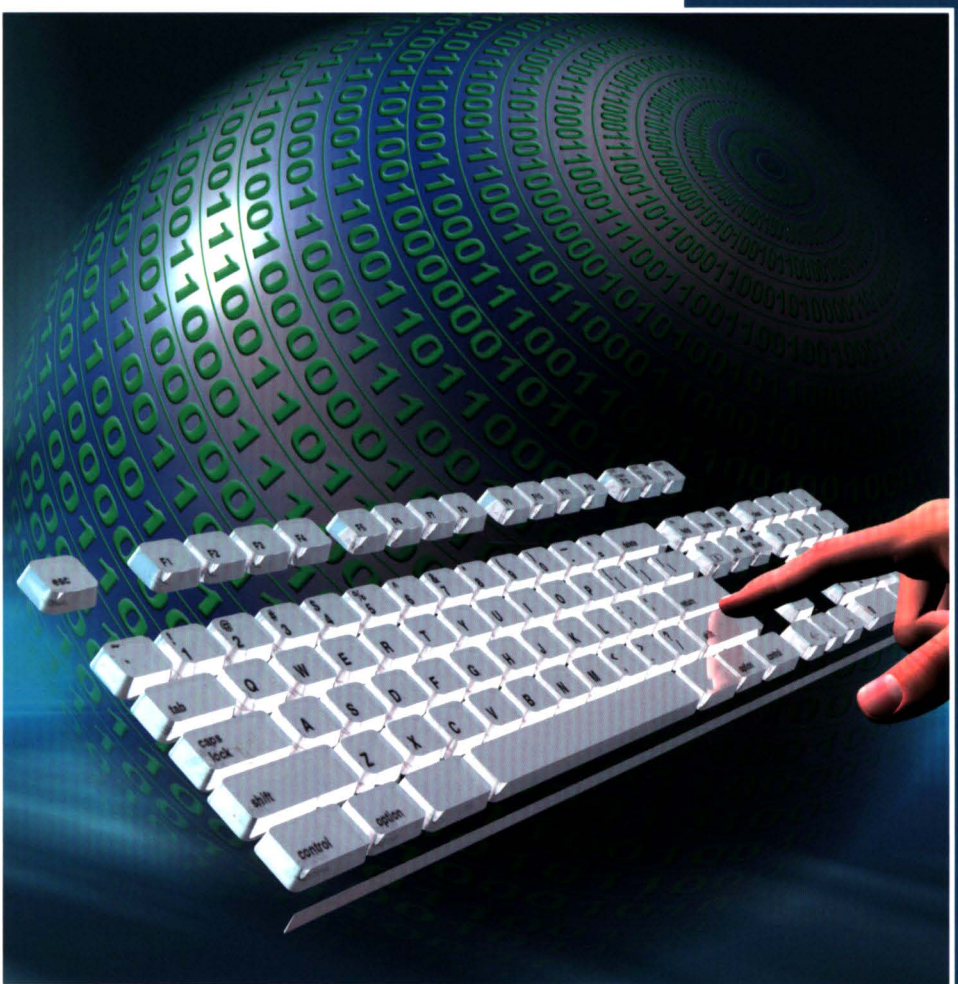
Tangente

Bibliothèque Tangente
L'aventure mathématique

Mathématiques & informatique



Une nouvelle ère numérique

EDITIONS POLE

HS n° 52

ISSN 2263-4908

Bibliothèque Tangente
L'aventure mathématique

Tangente Hors-série n° 52

Mathématiques et informatique

Une nouvelle ère numérique

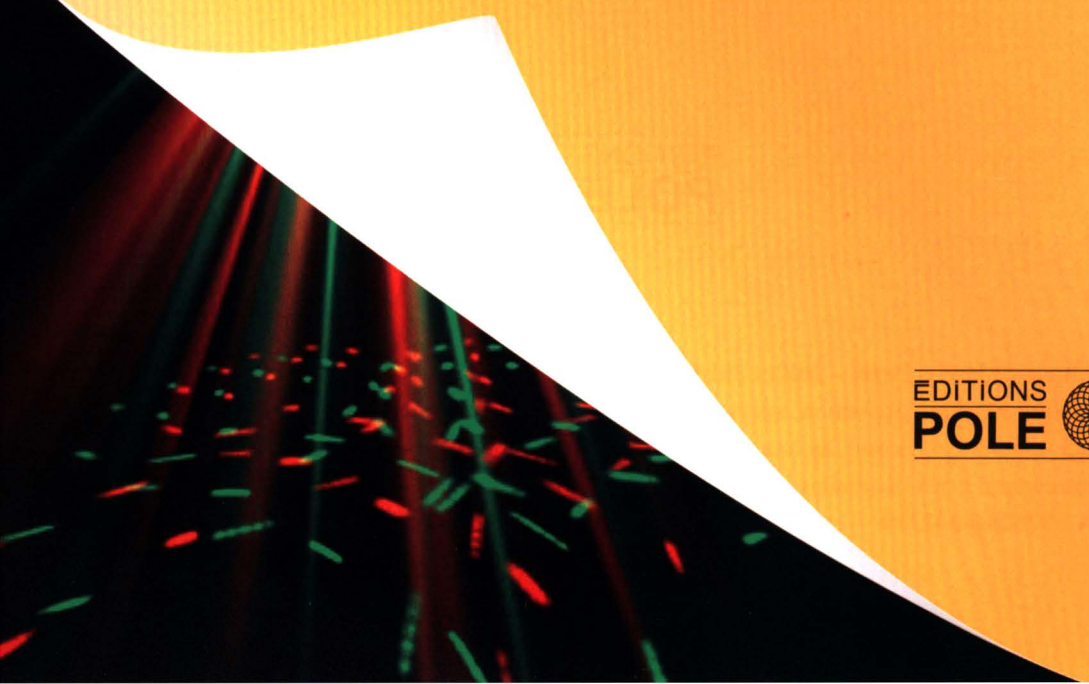
EDITIONS POLE

ISBN : 9782848841519 **ISSN : 2263-4908** **Commission paritaire : 1016K80883**

Bibliothèque Tangente
L'aventure mathématique

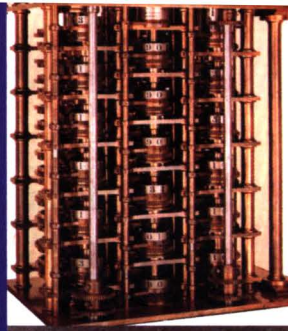
Prochainement dans la Bibliothèque Tangente

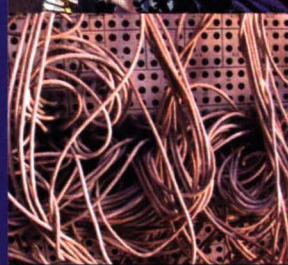
Les angles

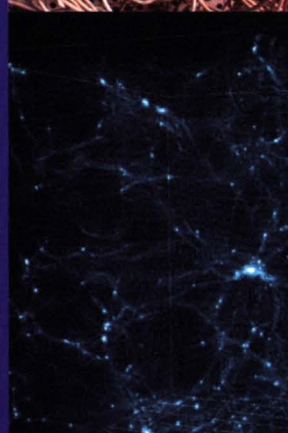


EDITIONS POLE

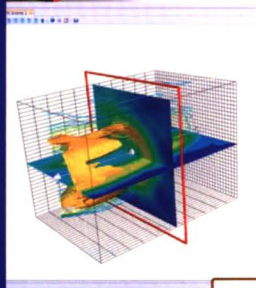
Mathématiques et informatique

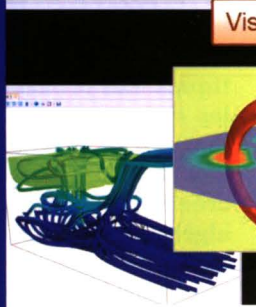


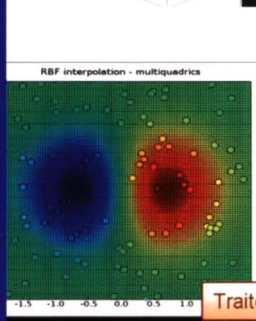


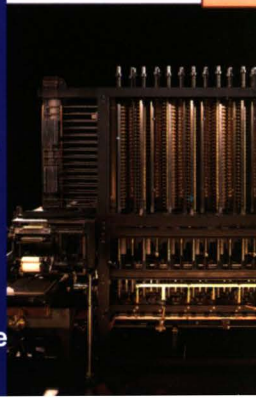


(suite du sommaire au verso)









par H. Lehning et M. de Ruelle

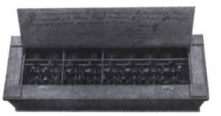
Le « Teorem » de Thales

Ne cherchez pas une ou deux erreurs d'orthographe dans le titre de cette brève, la société Thales a bien créé un objet qu'elle a nommé Teorem. Rien à voir avec le célèbre théorème de Thalès sauf, peut-être, l'idée d'un monde parallèle et proportionnel au nôtre, celui de l'espionnage. Il s'agit en effet d'un téléphone portable hautement sécurisé dont même le prix est secret, quelques milliers d'euros selon nos informations. Ne cherchez pas à l'acheter dans une boutique, vous ne l'y trouverez pas.

© H. Lehning

Le Teorem de Thales ouvert. Le petit écran supplémentaire informe l'utilisateur du niveau de sécurité de la communication.

Ce téléphone est réservé aux gouvernants et hauts responsables de l'armée. Son usage est plus lourd que celui des téléphones usuels, c'est pourquoi longtemps les ministres ont préféré utiliser leurs téléphones usuels, inconscients qu'ils étaient de l'importance de la sécurité. Il a fallu les révélations d'Edward Snowden pour qu'ils comprennent que leurs conversations à travers un téléphone portable ordinaire étaient écoutées par la NSA… et d'autres services d'intelligence sans aucun doute. Il est difficile d'en dire plus car, bien entendu, les algorithmes de chiffrement utilisés sont classés « secret défense » !



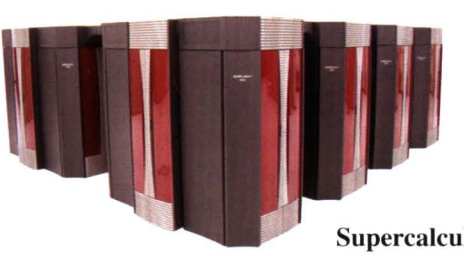
Une pascaline.

L'Amstrad CPC464.

L'informatique, d'hier à aujourd'hui

L'informatique a totalement envahi le monde moderne, qui ne peut plus s'en passer, pas plus que de l'eau courante ou de l'électricité. Plus encore que ces éléments indispensables de notre environnement, l'histoire témoigne d'une accélération exponentielle du temps qui permet à chaque instant de progresser autant que pendant les dix instants précédents.

De l'algorithme d'Euclide à la machine à calculer de Pascal, du premier ordinateur potentiel de Babbage à la machine de Turing, des ordinateurs géants aux PC, des disquettes au Web généralisé, tout va tellement vite que personne ne sait ce que réserve l'avenir. Voilà pourquoi, tant du côté de la recherche que dans la médiation scientifique vers le public, des efforts sont indispensables pour ne pas laisser une partie de la population sur le bas-côté de la route.



Supercalculateur Cray X1.

par B. Hauchecorne & A. Zalmanski

La naissance d'un nouveau vocabulaire

À nouvelle discipline, nouvelle terminologie. L'apparition de l'informatique dans la seconde moitié du XX^e siècle est associée à la création de nouveaux termes. Curieusement, aucune règle générale ne s'est appliquée, montrant l'absence de méthodologie dans la création de néologismes dans les langues actuelles (voir la note de lecture page 43).

Tant que l'informatique est restée l'apanage des entreprises d'informatique, les créations se sont faites consciemment. Ainsi John Tukey, mathématicien travaillant chez IBM, forgea à la fin de la guerre les termes *software* et *bit* en s'appuyant sur la terminologie anglaise. C'est en reprenant des racines latines que Jacques Perret et Karl Steinbruch introduisent les mots *ordinateur* et *informatique*. Citons le mot *souris* désignant ce petit boîtier qui nous permet de communiquer avec l'ordinateur : on doit son invention, en 1963, à l'ingénieur américain Douglas Engelbert. Dans la plupart des langues, sauf peut-être en italien et en japonais, on a tout simplement traduit le nom de ce petit rongeur familier ; ainsi, un Espagnol manie *el ratón* là où un Allemand utilise *die Maus*.

Le mot français *logiciel* est l'une des rares réussites de francisation de termes venus d'Outre-Manche sinon d'Outre-Atlantique. Ce mot-valise, contraction des termes *logique* et *matériel*, fut introduit en 1972 par l'Académie française pour remplacer le mot anglais *software* que rien ne semblait pouvoir arrêter.

L'arrivée des ordinateurs personnels dans les années 1980 fut synonyme d'une invasion de nouveaux termes venus de l'anglais qu'aucun organisme référent ne pouvait plus canaliser dans notre langue. Cette terminologie concerne moins les fondements de l'informatique que son utilisation courante. Ainsi on parle de *bug*, de *spam*, de *mail* et le préfixe *e-* (prononcer *i*, si vous ne voulez pas passer pour *has-been*) envahit le vocabulaire. Sans être hostile à l'emprunt linguistique, prenons garde à ce qu'il ne défigure pas notre langue !

Informatique

Karl Steinbuch, l'un des pionniers de l'informatique en Allemagne, forgea en 1957 le mot Informatik où l'on reconnaît bien sûr l'influence du mot français informer qui signifiait, à l'origine, *façonner l'esprit*. Le suffixe *-ique* (*-ik* en allemand) qui l'agrémente provient du latin *-icus* : relatif à. Certains estiment cependant que l'on est en présence d'un *mot-valise*, c'est-à-dire le début d'un terme et la fin d'un autre, abréviation de l'expression *information automatique*. Professeur d'informatique à Harvard puis directeur du Centre national de calcul électronique de la société Bull, Philippe Dreyfus reprend en français en 1962 et popularise le mot sous sa forme francisée *informatique*.

De *bit* à *byte* en passant par l'octet

À la fin des années 1940, John Tukey appelle *bit* l'unité élémentaire de stockage dans la mémoire d'un ordinateur. Il s'agit de la contraction de *binary digit*, c'est-à-dire *chiffre binaire*. Sans doute faut-il y voir un clin d'œil au terme anglais *bit* qui désigne un *morceau*. L'utilisation du terme *bit* a été popularisée par Claude Shannon, l'un des fondateur de la théorie de l'information.

Cependant, la plus petite unité adressable se compose de plusieurs *bits*, en général 8. C'est pourquoi, on utilise souvent le mot *octet*, introduit vers 1920 en chimie pour désigner une couche composée de 8 électrons. Le mot *byte*, introduit en 1956 par Werner Buchholz, ingénieur américain d'origine allemande travaillant pour IBM, lui est (presque) synonyme : il désigne la plus petite unité adressable (qui autrefois pouvait varier en fonction du matériel).

Ordinateur *versus* computer

Le mot *ordinator* existait déjà en latin. Il désignait celui qui met de l'ordre, qui règle les choses. On l'utilisait par exemple en droit pour nommer celui qui instruisait en justice. Pour les premiers chrétiens, l'*ordinator* était celui qui dirigeait les cérémonies, ce qui explique que, de nos jours encore, on parle de l'ordination des prêtres. Le mot apparaît en français vers 1600, dans des sens voisins.

Les anglophones utilisent le mot *computer*. Emprunté également au latin, il vient du verbe *computare* qui signifie *calculer*, *compter* (qui a évidemment la même origine). Alors que cet anglicisme commençait à se répandre dans le monde, IBM France fit appel à un linguiste et latiniste, Jacques Perret, pour trouver un terme français. Bien astucieusement, Perret fait revivre le mot *ordinateur*, qui s'est implanté sans difficulté dans notre langue, ainsi qu'en espagnol sous la forme *ordenador*. On peut s'en réjouir pour deux raisons : d'abord il remet au goût du jour un vieux mot français et ne perturbe donc pas notre langue, mais surtout sa connotation bien plus large correspond mieux aux tâches gérées par les *ordinateurs* que le mot *computer*, c'est-à-dire *calculateur*, beaucoup plus réducteur. Le mot anglais a cependant fait le tour du monde puisque seules les rives de la Méditerranée y ont partiellement échappé (français, espagnol, grec, turc et arabe).

Un programme palindromique

Le programme ci-contre, palindromique en chaque ligne, lit la ligne 9 et écrit son inversion. Le concepteur de ce programme a remporté le concours de code impénétrable Obfuscated CContest de 1987.

```
char rahc
[ ]
=
"\n/"
,
redivider
[ ]
=
"La marine en ira maL"
,
*
deliver,reviled
=
1+1
,
niam ; main
( )
{/*\}
\*/
int tni
=
0x0
,
rahctup,putchar
( )
,LACEDx0 = 0xDECAL,
rof ; for
(;(int) (tni);)
(int) (tni)
= reviled ; deliver =
redivider
;
for ((int)(tni)++,++reviled;reviled* *deliver;deliver++,++(int)(tni)) rof
=
(int) -1- (tni)
;reviled--;--deliver;
(tni) = (int)
- 0xDECAL + LACEDx0 -
rof ; for
(reviled--,(int)--(tni);(int) (tni);(int)--(tni),--deliver)
rahctup = putchar
(reviled* *deliver)
;
rahctup * putchar
((char) * (rahc))
;
/*\
{\*/}
```

par S. Alayrangues, C. Froidevaux et C. de la Higuera

Société Informatique de France

SIF : La plus jeune des sociétés savantes

En France, les sciences ont toutes une société savante. Pour les plus installées d'entre elles, ces sociétés savantes sont des institutions centenaires qui jouent un important rôle d'organisation et d'animation. Elles permettent à la science qu'elles représentent d'être visible, aux médias de localiser des interlocuteurs, aux pouvoirs publics de discuter d'évolutions dans les cursus. Une société savante peut contribuer à la valorisation des travaux conduits dans la discipline, et à sa vulgarisation. Elle a également une responsabilité vis-à-vis des jeunes : leur montrer tout l'intérêt d'orienter leurs études dans sa branche scientifique.

La Société informatique de France (SIF) a été créée pour cela le 31 mai 2012. Ses adhérents en partagent les objectifs : promouvoir l'informatique, une informatique ouverte, qui se retrouve pleinement au sein des sciences de l'information et du numérique. Il s'agit également de contribuer à la diffusion d'une culture scientifique, et de répandre l'idée que l'enseignement de l'informatique, en tant que discipline scientifique et pas seulement en tant qu'outil, est une nécessité aujourd'hui.

Comme le montre brillamment Milad Doueihi dans son livre *Qu'est-ce que le numérique ?* (Hermès, 2013), l'informatique est née science, s'est développée en industrie et a donné lieu à une culture qui a changé de nom et est devenue *numérique*. Mais cela ne signifie pas, bien au contraire, que l'informatique ait disparu ! Elle est l'essence même de cette culture. Avec l'aide de son Conseil scientifique, de ses adhérents, des associations d'informaticiens, des grandes institutions, la SIF entend contribuer au développement et au rayonnement de cette science.

Site de la SIF :
www.societe-informatique-de-france.fr

Colin de la Higuera, président de la SIF : « *L'enseignement de l'informatique en France est insuffisant* »

Dans l'ensemble des pays se pose la question de l'enseignement de l'informatique. Certains, comme la France, font le pari de former au numérique, c'est-à-dire, essentiellement aux usages. D'autres, comme le Royaume-Uni, les États-Unis et la majorité des pays asiatiques, jugent indispensable de former à l'informatique, c'est-à-dire à la programmation, au codage et à l'ensemble de ce qui permet de détenir les clés du monde numérique.

Si l'on part du principe qu'un enseignement de l'informatique est correct quand la discipline est enseignée de façon stable et pérenne par des spécialistes formés pour cela, le constat est qu'en 2014 la science informatique n'est réellement enseignée en France que dans certains cursus de l'enseignement supérieur.

Pourtant, quel que soit leur niveau de formation, techniciens et ingénieurs passés par ces cursus informatiques sont régulièrement plébiscités par les entreprises : les salaires d'embauche sont presque systématiquement toujours les plus hauts, et les embauches ont souvent lieu avant la fin des études.

Mais le contingent d'informaticiens formés est loin d'être suffisant pour répondre aux besoins actuels du marché de l'emploi et toutes les études prospectives à disposition montrent que cette tendance n'est pas prête de s'inverser. Malgré ces constats, aucun dispositif permettant de généraliser l'enseignement de l'informatique n'est mis en place et les obstacles s'accumulent notamment avec des problèmes de mise en œuvre de programmes à la hauteur des enjeux scientifiques, culturels et sociétaux actuels au lycée, en BTS, et dans les classes préparatoires. Malgré le très fort investissement de ceux qui sont impliqués, le manque d'enseignants ayant reçu une formation spécifique en informatique empêche la mise en place d'une politique de formation ambitieuse, comme cela peut être le cas dans d'autres pays.

Le Prix Bernard-Novelli

Un concours pour lycéens accros d'informatique et de jeu

Depuis l'avènement de l'enseignement de spécialité ISN (informatique et sciences du numérique), le magazine *Tangente* organise un concours de projets informatiques autour du jeu, intitulé Prix Bernard-Novelli en souvenir de son collaborateur disparu en 2011. La Société informatique de France, l'association Prologin, Magma Mobile, Casio, INRIA et plusieurs universités et écoles d'ingénieurs en sont partenaires. Conçu en priorité pour les élèves en spécialité ISN, le concours est cependant ouvert à tout lycéen accro d'informatique et de jeu. L'idée est de concilier la passion que peuvent avoir les lycéens pour le jeu ou pour l'informatique en joignant l'utile (la préparation du projet ISN) à l'agréable (le fait de se voir reconnaître dans l'univers du jeu numérique).

Les lauréats remportent un trophée d'art mathématique, mais surtout la possibilité de voir durant l'année suivante leur jeu développé de manière professionnelle et proposé sur les *smartphones*.

Références :

- *Tangente* 141, dossier « Bernard Novelli », 2011.
- *Tangente Éducation* 22, dossier « ISN », 2012.

© É. Thomas © É. Thomas

Les deux lauréats 2013 arborant leur trophée : Marc Coudriau (à gauche) et Maxime Gourghoulon.

Prix 2014 : dans le cadre des Trophées Tangente

En 2014, le Prix Novelli sera organisé dans le cadre des Trophées Tangente, décernés au Sénat le mercredi 19 novembre. Les candidats doivent préalablement s'inscrire sur le site www.tropheestangente.com en précisant leurs coordonnées complètes, leur établissement scolaire et leur classe, ainsi qu'un résumé de leur projet en quelques lignes. Ils ont alors le temps de se consacrer à leur bac avant d'envoyer leur dossier complet avant le 31 août.

Ils devront alors envoyer un dossier numérique comportant, outre les pièces administratives :

– La règle du jeu qui fait l'objet du projet. Cette règle n'est pas forcément originale, mais doit faire intervenir des qualités mathématiques chez le joueur (logique, habileté numérique ou géométrique…) ;

– Un document précisant les éléments descriptifs du projet : le but du jeu, les algorithmes mis au point par le candidat… ;

– La partie du projet développée (non compilée et compilée).

Nouveau : deux acteurs du domaine de l'éducation, du jeu et de l'informatique, les calculatrices Casio et la société Magma Mobile (voir en page 31), parrainent le concours, qui prend une importance médiatique particulière.

Renseignement et inscription : http://www.tropheestangente.com/

Les perspectives de l'informatique au XXIe siècle

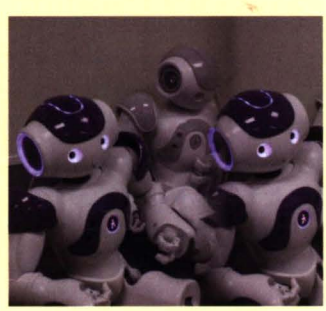
Depuis l'avènement de l'informatique, que de chemin parcouru ! Les applications, déjà innombrables, ne constituent qu'une infime partie du potentiel de cette science. Au niveau théorique, la recherche en science informatique a fait émerger des concepts inédits. Sur quoi porte-t-elle aujourd'hui, sur quoi portera-t-elle demain ?

Toutes les activités humaines, économiques, scientifiques ou industrielles présentent aujourd'hui des enjeux liés, de manière plus ou moins importante, aux progrès scientifiques et technologiques des champs informatiques et mathématiques des sciences. Bien entendu,

Les sciences du numérique

Le terme de *sciences du numérique* désigne les sciences de l'information et de la communication sur leurs volets matériels et logiciels. Cette terminologie inclut les sciences informatiques (*computer science*) et les mathématiques appliquées liées à ces sujets et représente une nouvelle façon de parler des sciences de l'information et de la communication, à l'aube du XXIe siècle. Elle se nourrit de disciplines telles que l'automatique, le traitement du signal, ou la robotique. Les grands objets d'études des sciences du numérique sont les systèmes, données, interfaces et modèles.

Parmi les systèmes, on peut citer les réseaux. Dans cet exemple, les formalismes vont pouvoir s'appliquer aux réseaux informatiques, mais être ensuite transformés pour étudier des réseaux biologiques (réseaux de neurones, réseaux de gènes, *etc.*) ou des réseaux humains (réseaux sociaux).



Les *sciences du numérique* se distinguent des *sciences numériques* (*computational sciences*), qui désignent une approche scientifique reposant sur un recours massif aux modélisations informatiques et mathématiques ainsi qu'à la simulation. Ici (pour simplifier), ce ne sont plus uniquement des équations mathématiques qui décrivent les lois de la nature, mais des algorithmes qui représentent ces mécanismes naturels, et permettent de prédire leur évolution et d'ajuster leurs variables pour les contrôler.

et c'est aussi ce qui fait leur richesse et leur intérêt, elles interagissent fortement avec les autres disciplines. Il est courant que les avancées d'un autre domaine scientifique irriguent ces sciences du numérique, ou qu'une question applicative débouche sur un problème fondamental inédit à résoudre.

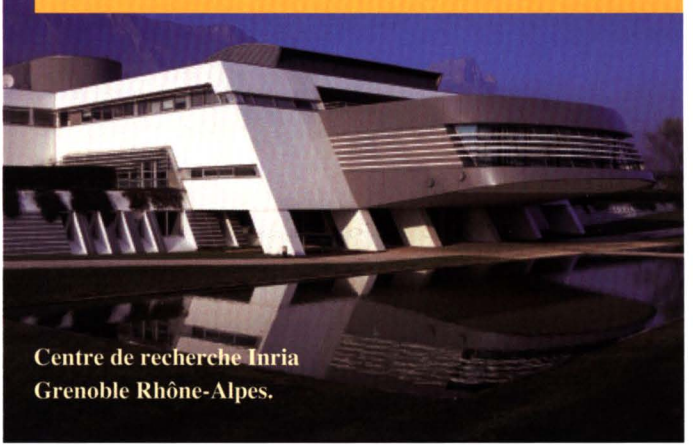
Les instituts de recherche français dédiés à ces sujets sont Inria et l'institut du CNRS qui se nomme l'INS2I. La majorité des chercheurs de ce domaine travaillent au sein des universités. D'autres organismes comme les grandes écoles d'ingénieurs ou le CEA sont des acteurs majeurs sur ces sujets. Beaucoup d'équipes de recherche sont communes à plusieurs de ces structures.

Les défis de la recherche informatique

La recherche en informatique porte essentiellement sur deux branches. L'une, de nature théorique, concerne la définition de concepts et modèles, l'autre, de nature plus technologique, s'intéresse aux techniques concrètes d'implantation et de mise en œuvre des abstractions précédentes.
Mais *justement*, ce qui distingue cette science des mathématiques c'est qu'elle est *incarnée* dans la technologie qui en émane : ces deux facettes en sont inséparables. Science formelle du calcul au sens algorithmique, en rapport avec tout type d'information que l'on peut représenter de manière symbolique ou numérique, ce calcul va s'implémenter dans des machines dont l'étude théorique est tout aussi importante et cette information sera codée dans des langages, eux aussi objets formels de cette discipline.

Inria

Créé le 3 janvier 1967 suite au lancement du Plan Calcul, l'*Institut national de recherche en informatique et en automatique* (Inria) est l'institut de recherche français qui se consacre uniquement aux sciences du numérique (mathématiques et informatique). Il a le statut d'établissement public à caractère scientifique et technologique. Les équipes-projets Inria, rassemblant 1800 chercheurs de l'institut et près de 1600 universitaires ou chercheurs d'autres organismes, mènent des recherches fondamentales et appliquées qui contribuent aux technologies numériques de demain, en partenariats étroits avec les acteurs de la recherche publique et privée en France et à l'étranger. Chaque année, elles publient près de 5000 articles. Elles sont aussi à l'origine de la création de nombreuses « start-ups » (110 à cette date).



Centre de recherche Inria Grenoble Rhône-Alpes.

Une partie des recherches est directement stimulée par les contextes sociétaux, économiques et environnementaux. Les chantiers majeurs sur lesquels les principaux instituts s'engagent (Inria et INS2I, voir encadrés) mettent l'humain au cœur des problématiques du numérique.

Dans ce cadre, les chercheurs d'aujourd'hui doivent résoudre de nombreux types de défis scientifiques. En voici quelques exemples.

• Les défis de la modélisation multi-échelle et des incertitudes associées. Ils concernent les très grands systèmes numériques, embarqués ou enfouis, les systèmes de systèmes, la programmation des très grands logiciels... Ils doivent prendre en compte les impératifs de fiabilité, de sûreté et de sécurité, qui sont les principaux sujets actuels de recherche au cœur de l'informatique.

• Les défis liés à la transformation du déluge de données en bibliothèques de connaissances dignes de confiance. Les enjeux sont de taille : la cyber-communication généralisée dans laquelle on s'immerge doit être sûre et respectueuse de la vie privée ; l'interaction entre les mondes réels et numériques débouche par exemple sur des problématiques d'apprentissage qui renvoient à des thématiques pluridisciplinaires. L'élaboration même de la connaissance en devient un axe majeur.

• Les défis liés à la modélisation du vivant et à ses applications : santé, bien-être, relations entre l'humain et ses environnements. Le paradigme diffère selon qu'il s'agisse de développements théoriques ou d'expériences de laboratoire qui sont les formes traditionnelles de la science et de l'ingénierie. L'approche est ici de gagner en compréhension, principalement grâce à l'analyse de modèles mathématiques mis en œuvre à travers des simulations numériques. De plus, des algorithmes représentent les mécanismes naturels étudiés et permettent de les représenter, de les prédire, et d'ajuster leurs variables pour les contrôler.

L'INS2I : le plus jeune institut du CNRS

Avec ses dix instituts thématiques, le Centre national de la recherche scientifique (CNRS) couvre l'ensemble du champ scientifique. Créé en 2009, l'*Institut des sciences de l'information et de leurs interactions* (INS2I) est le plus jeune d'entre eux. Il soutient et coordonne la communauté autour d'un réseau d'une soixantaine de laboratoires associés aux centres universitaires et répartis sur l'ensemble du territoire. Les professionnels des métiers de la recherche présents dans ces laboratoires sont au nombre de 11 000. Une moitié est constituée de doctorants et de post-doctorants, et l'autre moitié de permanents, exerçant des métiers variés : ingénieurs, techniciens, enseignants-chercheurs de l'université et chercheurs du CNRS. Ces derniers, au nombre d'environ 600, sont des chercheurs à plein temps.

Les recherches de l'INS2I se répartissent en six domaines principaux : l'informatique fondamentale ; le traitement et la sécurité des données ; les réseaux ; le traitement des signaux et des images ; l'automatique ; l'intelligence artificielle et la robotique.
Les trois premiers domaines relèvent de l'informatique, les trois derniers vont au-delà, l'ensemble constituant les « sciences de l'information ».
Mettant à profit la pluridisciplinarité du CNRS, l'INS2I tisse des relations fortes avec les mathématiques, la biologie, l'ingénierie ou encore les sciences humaines et sociales.
Vous trouverez davantage d'informations sur le site www.cnrs.fr/ins2i.

Michel Bidoit, directeur de l'Institut

Des concepts nouveaux jeunes de moins d'un siècle

De grands résultats scientifiques sont offerts par cette science.
Ainsi, dès qu'une machine (électronique, abstraite, *etc.*) qui calcule peut exécuter les ingrédients de base des algorithmes, alors elle peut exécuter

Partenariat maths-info : le GDR IM http://www.gdr-im.fr/

Une communauté se consacre au développement des nouvelles mathématiques liées à l'informatique, celle du GDR IM (*Groupement de recherche Informatique et mathématiques*).
Deux événements importants marquent la vie du GDR :

- *Les Journées nationales* : ce rassemblement annuel (en 2014 à l'université Paris-Diderot) donne l'occasion à une dizaine de chercheurs du GDR et à des invités prestigieux de présenter leur recherche ;
- *L'École des jeunes chercheurs en informatique mathématique* : cette semaine de cours de haut niveau donnés par des spécialistes français du domaine permet de compléter la formation de jeunes chercheurs (étudiants en Master2, doctorants ou docteurs récemment diplômés). Celle de 2014 a eu lieu à Caen du 31 mars au 4 avril.

Arnaud Durand, Jean-Michel Muller et Brigitte Vallée

Les 92 participants de l'école des jeunes chercheurs en informatique mathématique à Rennes, 2012.

tous les logiciels du monde : c'est une machine universelle, comme définie par la thèse de Church-Turing (voir article en page 40). Ce résultat implique par exemple que toutes les « intelligences mécaniques » (ordinateur, robot, *etc.*) sont qualitativement équivalentes : elles seront juste plus ou moins rapides ou performantes.
Un autre résultat montre le lien entre la science informatique et de grands concepts. C'est la théorie algorithmique de l'information. La complexité d'un message en terme d'information (au sens où " blablabla... " même répété une infinité de fois contient moins d'information qu'une page de longueur finie) se définit par *« la longueur du plus petit programme écrit avec une machine universelle qui génère le message en question »*. Bref, la longueur du plus petit programme indique la complexité du message. Il n'est pas du tout évident, mais c'est le cas, que cette définition est pertinente, bien fondée, ne dépend de la machine choisie qu'à une constante près, et offre une formalisation profonde de cette idée de « complexité ».

De même la notion de suite aléatoire prend un visage nouveau : ce n'est plus une suite « générée par le hasard » (au sens précis de la théorie des probabilités), mais une suite qui ne peut être... générée par un programme de taille finie (une suite « imprédictible » en quelque sorte). Nous voilà donc devant une vision nouvelle de cette notion d'aléa.

Au fur et à mesure que cette science se popularisera, la pensée informatique enrichira les connaissances humaines et élargira notre vision sur ces sujets.

La relation avec les mathématiques

La modélisation et la résolution de nombreux problèmes rencontrés en informatique offrent des défis nouveaux pour les mathématiques pour les deux disciplines informatiques et mathématiques.
Il est en effet rare que les techniques mathématiques usuelles puissent être utilisées « clés en main ». Les objets manipulés tout comme les types de réponses attendues (par exemple en termes d'effectivité) apparaissent souvent comme non classiques aux mathématiciens.

L'informatique théorique intervient, entre autres, dans les domaines suivants : calcul formel, arithmétique des ordinateurs, géométrie pour l'image, algorithmique (du texte, sur les arbres, graphes), automates, systèmes dynamiques, analyse d'algorithmes, complexité du calcul, informatique quantique, logique, bases de données, preuves de programmes, vérification logicielle et matérielle, codes correcteurs, cryptographie...
Tout un programme !

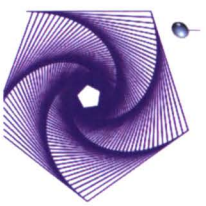
ARTICLE COLLECTIF COORDONNÉ
PAR THIERRY VIÉVILLE
ET CHRISTINE FROIDEVAUX

INTERNAUGRAPHIE

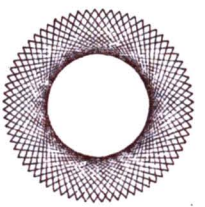
- https://fr.wikipedia.org/wiki/Sciences_du_numérique
- https://www.allistene.fr
- http://www.inria.fr/content/download/24371/605690/version/5/file/Plan-strategique-Objectif-2020.pdf

Art mathématique et informatique

Les outils informatiques peuvent aussi servir à la construction d'œuvres d'art mathématique à base de courbes. L'application la plus connue est, bien sûr, l'art fractal. Mais bien avant son apparition, les premiers langages permettaient déjà de construire des courbes ornementales comme les *jolygones*. La construction algorithmique de ces courbes a été initiée par feu *Le Petit Archimède* dans son n°14 en 1975, et favorisée ensuite par l'avènement de certains langages explicites, comme le Logo.



$k = 0{,}00$ $\alpha = 61°$

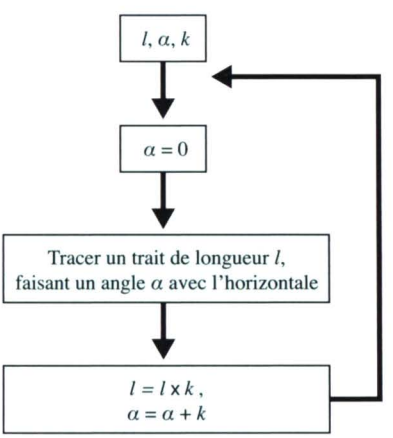


$k = 1$ $\alpha = 115°$

Le principe est simplissime :

- on trace un segment horizontal de longueur l, puis un autre faisant un angle α avec le premier et dont la longueur (l) a été multipliée par un nombre donné k compris entre 0 et 1,
- on recommence cette opération à partir de la nouvelle extrémité jusqu'à obtention du jolygone (α, k).

En voici l'organigramme :



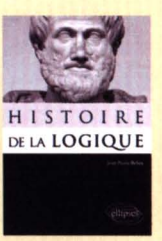
La longue et riche histoire de la logique

Suivant un plan historique en six chapitres, Jean-Pierre Belna nous fait suivre, au cours des siècles, la construction d'une certaine idée de la logique, car cette notion abstraite, qui vise à l'universalité, échappe à une définition consensuelle. Les termes les plus courants qui lui sont associés tout au long de cet ouvrage sont les notions de vérité, de raisonnement, de loi, de règle, de forme et de validité.

Née en Grèce, fille de la dialectique, en tant que pratique de la discussion raisonnée, on lui attribue Aristote pour père adoptif. Le Stagirite a été le premier à considérer la logique comme une discipline autonome. Sa syllogistique, dont les procédés, avec leurs limites, nous sont clairement explicités, sera reprise au Moyen Âge et servira longtemps de référence, plongeant dans les limbes de l'histoire la bien plus fine logique mégarico-stoïcienne. La Scolastique médiévale fut à son tour rejetée à l'âge classique. Pascal retisse le lien logique-langage avec la logique de Port-Royal et Leibniz anticipe certaines idées de la logique moderne, qui apparait au XIX^{e} siècle. La logique symbolique se lie alors intimement aux mathématiques avec Boole et Frege, intervenant profondément dans les débats sur les fondements des mathématiques. Au siècle de Russell et Gödel se développe, à côté de la logique dite standard, des logiques plurielles, dont certaines s'affranchissent du principe du tiers exclus. Enfin, un chapitre évoque les logiques orientale, indienne et chinoise.

Sans faire appel à trop de connaissances préalables, cet ouvrage, par sa rigueur, la simplicité et la clarté de sa rédaction, requiert néanmoins une certaine concentration de lecture. Par son panorama historique assez complet, ce livre-manuel est une bonne initiation à la logique et devrait intéresser les étudiants de deux domaines scientifiques dont la logique est à l'intersection : la philosophie, qui l'a produite, et les mathématiques, qui l'ont réformée.

F. L.



Histoire de la logique.
Jean-Pierre Belna, Ellipses,
128 pages, 2014, 16 euros.

Informatique, popularisation et médiation

La « fracture informatique » ne concerne pas seulement les générations ou l'accès au matériel. Au sein d'une même catégorie d'utilisateurs d'équipements informatiques, des inégalités parfois rédhibitoires sont constatées en fonction de la culture scientifique et technique liée au numérique. La médiation scientifique est là pour tenter de répondre à ce défi de société.

Le numérique façonne aujourd'hui le monde dans lequel nous évoluons. Des activités professionnelles, quel qu'en soit le secteur (industrie, tertiaire, enseignement, commerce, *etc.*), aux activités ludiques, domestiques et sociales, toutes font appel au moins en partie aux technologies issues de l'informatique et des sciences du numérique. Il est par conséquent essentiel que les citoyens maîtrisent ces technologies dans leurs usages, mais aussi qu'ils acquièrent la culture scientifique suffisante pour en comprendre les fondements et pouvoir ainsi contribuer à la mutation de la société engendrée par leur diffusion rapide dans le tissu social.

La médiation scientifique

La *médiation scientifique* sert à favoriser l'appropriation de cette nouvelle dimension de l'existence, nourrir la curiosité vis-à-vis des applications innovantes et d'intérêt commun de ces technologies, encourager la participation ou l'implication dans la création de ces applications, former des citoyens éclairés et contribuer à lutter contre la fracture numérique.

En quoi consiste-t-elle ? En un certain nombre de pratiques qui permettent au public visé de rencontrer les informations dont il a besoin, et ce dans des contextes qui ne sont pas forcément institutionnels ou éducatifs.

L'offre de médiation est d'autant plus variée qu'elle doit être adaptée à toutes les typologies des connaissances, et à la curiosité de publics divers : enfants et jeunes, curieux de science, grand public, scientifiques de toutes disciplines, décideurs politiques, partenaires socio-économiques…

Pour faire passer le message, on peut distinguer trois volets principaux.

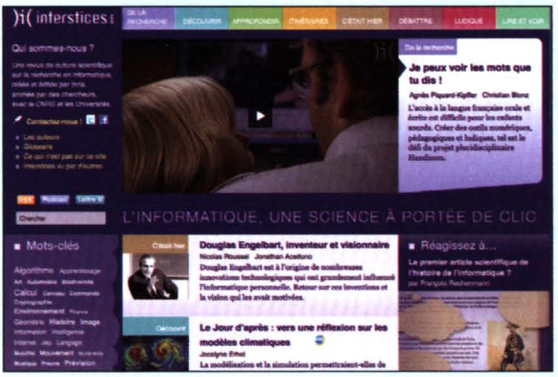
- un *volet éducatif* à destination des élèves et étudiants : interventions dans les établissements scolaires devant de larges groupes, visites d'élèves dans des laboratoires de l'industrie ou de

la recherche, mais aussi accueil personnalisé de lycéens ou d'étudiants, par exemple à l'occasion de leurs travaux personnels encadrés (TPE et TIPE), accompagnement de projets d'élèves ou d'enseignants liés à l'option ISN (informatique et sciences du numérique) de terminale, à des stages « MathC2+ », à des ateliers « MATh.en.JEANS » ou dans d'autres contextes…

- un *volet citoyen* à destination du grand public : diffusion de pépites de science informatique lors d'expositions, de conférences ou de rencontres, participation à des événements comme la Fête de la science, les cafés-sciences, la Nuit des chercheurs, apport de connaissances scientifiques pour alimenter les débats citoyens ou les cafés scientifiques sur les sujets de type science et société, *etc*. Il s'agit ici aussi de susciter l'intérêt du citoyen pour des sujets non directement scientifiques, comme une exposition artistique utilisant les nouvelles technologies, et de profiter de cet intérêt pour emmener le citoyen vers la science.

- un *volet participatif* à destination des curieux de sciences : diffusion de savoirs et de contenus de culture scientifique *via* des magazines comme *Tangente*, *Phosphore* ou *Okapi*, des revues comme *DocSciences* ou *TDC* (*Textes et documents pour la classe*), ou comme la revue en ligne *Interstices* (voir encadré), d'outils du numérique, soutien de clubs (club-robotique, maker-club, fab-labs) ou d'éducation populaire en science, par exemple en aidant à trouver des ressources et des contacts scientifiques. L'organisation de concours, destinés ou non aux scolaires, est un mode de médiation très efficace.

La revue en ligne *Interstices* : http://interstices.info



Interstices, revue de culture scientifique en ligne, invite les curieux de sciences à explorer les sciences du numérique. Créée à l'initiative d'Inria, cette revue animée par des chercheurs se développe depuis dix ans en partenariat avec le CNRS, les universités et des associations professionnelles du domaine, notamment le groupe ITIC-EPI-SIF. Son but ? Mettre les connaissances scientifiques en informatique et en mathématiques appliquées à la portée d'un large public francophone. Elle jouit d'une notoriété croissante sur son domaine et recense 3000 visites d'internautes par jour.

Algorithmes, langages, information, modèles… Pour faire comprendre les notions fondamentales de l'informatique et des mathématiques appliquées, permettre d'en mesurer les enjeux pour la société et d'aller à la rencontre des acteurs de la recherche, *Interstices* propose en accès libre et gratuit plus de 400 documents. Ces contenus très divers, ressources pour les lycées, articles didactiques, dossiers thématiques, portraits, documents approfondis, sujets de réflexion et d'histoire des sciences, tirent pleinement parti des technologies Web.
Vidéos, *podcasts* audios ou encore animations interactives viennent ainsi enrichir les textes rédigés par les chercheurs.
Quatre à cinq nouveaux sujets sont à découvrir chaque mois sur http://interstices.info.

Christine Leininger

ISN : un partenariat réussi pour faire entrer les sciences du numérique à l'école

Promouvoir l'enseignement de l'informatique et des sciences du numérique, en vue de donner aux jeunes une formation leur permettant d'être créatifs et productifs dans un domaine aux enjeux économiques forts dont notre pays ne peut se désintéresser, telle est la grande cause dans laquelle un certain nombre de médiateurs se sont investis.

Déjà depuis le milieu des années 2000, de nombreux acteurs ont milité pour la promotion de l'enseignement de l'informatique et des sciences du numérique (ISN) au lycée, au collège et à l'école. Ils se sont engagés, en partenariat avec les académies et les universités, à aider à concevoir cet enseignement, puis à le mettre en place, ce qui fut fait à la rentrée 2012.
Les résultats sont encourageants : dès la rentrée 2012, 750 établissements proposaient la spécialité informatique et sciences du numérique (ISN) aux classes de terminale S et plus de 10 000 élèves se sont inscrits. Cet enseignement sera étendu progressivement, les années suivantes, aux terminales des filières L et ES ainsi qu'aux classes préparatoires.
Ce partenariat se prolonge avec le ministère de l'Éducation nationale et l'inspection générale pour réfléchir à la mise en place d'une formation initiale pour les enseignants. De nombreux chercheurs en informatique ont ainsi contribué à la formation de la première vague de professeurs qui assurent la spécialité ISN au lycée, à l'élaboration des programmes et des supports de cours, et même à la rédaction de manuels pour les professeurs et les élèves.
Inria, la SIF, l'EPI, Pasc@line et tous leurs partenaires ont contribué à la création de contenus (manuels, supports d'activités, outils logiciels, ressources culturelles, *etc.*), à la formation d'enseignants, à la construction d'une plateforme d'échanges et de ressources «SIL:O!», à l'animation du biotope (site Facebook ISN pour les élèves) et aux réflexions sur le contenu des enseignements.
Tangente Éducation a réalisé lors de cette rentrée 2012 un numéro consacré à cette nouvelle option, lançant dès la première année un concours de projets informatiques autour du jeu, le concours Bernard-Novelli dont les premiers lauréats ont été récompensés en novembre dernier (voir page 9).

Ces trois volets se croisent en pratique, une intervention en classe (volet éducatif) par exemple donnant lieu à des discussions familiales le soir (volet citoyen).

Un engagement au service de causes nationales

Qui sont les médiateurs ? Institutionnels, associatifs, chercheurs, acteurs médiatiques ou éducatifs ou simples individualités, ils ont parmi leurs objectifs essentiels de faire connaître aux jeunes les secteurs de l'économie associés à ces sciences et d'augmenter ainsi leurs chances de trouver ou de créer un emploi.

Développer l'égalité des chances devant le numérique, proposer à des jeunes qui n'ont pas accroché, en mathématiques par exemple, une seconde chance avec cette matière nouvelle qu'est l'informatique (qui contient aussi des abstractions nouvelles, comme l'information ou l'algorithme), telle est une de leurs missions. Elle devra déboucher sur un apprentissage différent, *via* une *machine* qui autorise de nombreux essais et erreurs sans porter de jugement.
De tels projets, comme justement « L'école de la seconde chance », ont

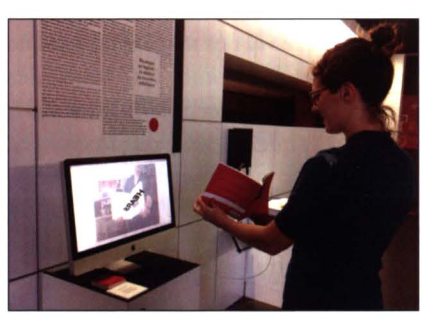
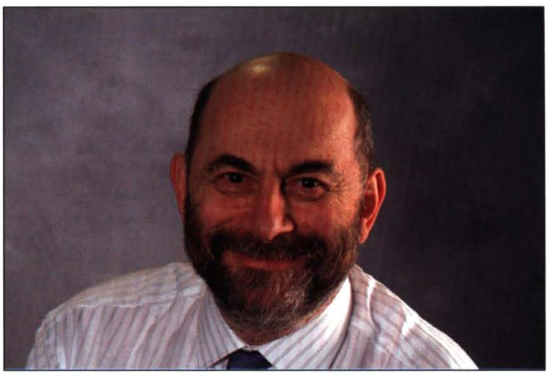


Photo prise durant l'exposition « Les littératures numériques d'hier à demain » qui s'est tenue à la Bibliothèque nationale de France du 24 septembre au 1er décembre 2013.

pu être financés notamment grâce au plan « Investissements d'avenir ».
Pour les organismes de recherche, c'est aussi une manière de montrer l'importance et l'utilité de l'investissement public en matière de recherche. Construire et renforcer des relais de médiation constituent un engagement des acteurs de la recherche en mathématiques et informatique qui se mettent en partenariat au service de grandes causes nationales.
Introduite dans l'article premier du Code de la recherche comme un des objectifs de la politique nationale de recherche et de développement technologique, la médiation est devenue une mission de la recherche. Elle est de plus en plus prise en compte, dans l'évaluation des individus, des équipes et des organismes de recherche, ainsi que par les instances nationales. Elle est, par exemple, inscrite dans le plan stratégique d'Inria, cette mission correspondant à une implication d'un jour par an en moyenne pour ses chercheurs.

Faire de la médiation scientifique entre ainsi dans le champ des missions de service public.
C'est un devoir mais aussi un plaisir, celui *« d'allumer l'étincelle dans les yeux des enfants »*, se plaisait à rappeler Gilles Kahn, premier informaticien à être entré à l'Académie des sciences.

Article collectif coordonné par Thierry Viéville



Gilles Kahn (1946–2006).

Le GICS, association en développement pour le partage des sciences

Le *Groupe pour l'initiative et la culture scientifiques* (GICS) est une association créée il y a trois ans par des étudiants rassemblés autour de l'idée que les sciences, en particulier les sciences du numérique, doivent se partager, dès le lycée, pour propager la culture scientifique, permettre de comprendre et d'apprécier un monde profondément scientifique et, avec un peu de chance, créer des vocations scientifiques.
Le constat de départ est le suivant : beaucoup d'étudiants en sciences sont prêts à passer une après-midi à partager leur enthousiasme avec des lycéens avec une approche proche des élèves et différente de celle adoptée en classe, sur des thèmes qui les passionnent. En face, nous avons dans tous les lycées de France des élèves prêts à se laisser entraîner par leur curiosité hors des sentiers battus des programmes de lycée.
Le GICS s'est donc donné pour mission de créer un échange entre étudiants et lycéens. Il propose à tous les passionnés de sciences de venir donner un peu de leur temps pour donner une fois un atelier ou un cours, traiter de problèmes concrets, de théorie ou de jeux d'esprit dans les lycées où il est implanté. Et il est prêt à s'implanter dans tous les lycées intéressés, il suffit de motiver les intervenants potentiels à proximité !
Lycéens désireux de voir le GICS s'implanter, passeurs de sciences prêts à donner quelques après-midis, professeurs souhaitant voir le projet s'installer dans leur lycée, amateurs de culture scientifique sont invités à se faire connaître sur le site http://gics.fr !

Adrien Dufour et Gabriel Gouvine

par M. Brilleaud et R. Lamarche-Perrin

Intelligence artificielle et philosophie, deux cousines éloignées

La notion d'intelligence artificielle débouche sur des interrogations que seule une réflexion philosophique peut structurer. Réciproquement, l'informatique peut apporter à la philosophie une approche expérimentale novatrice. Regards croisés sur deux disciplines qui ne sont lointaines qu'en apparence.

Statue d'Alan Turing à Bletchley Park.

Comment construire une machine *intelligente* ? Cette question intéresse tout autant les spécialistes en intelligence artificielle que les chercheurs en philosophie de l'esprit. Pour les premiers, il s'agit de créer une machine capable de résoudre des problèmes techniques habituellement traités par l'homme. L'objectif est, pour les seconds, de comprendre la nature même de l'intelligence en s'interrogeant sur ses propriétés essentielles : une machine peut-elle être consciente, avoir des émotions, des intentions, *etc.* ?

Ces qualités ne sont pas vraiment pertinentes pour l'intelligence artificielle puisque les similitudes entre la machine et l'être humain ne sont pas *a priori* nécessaires pour résoudre les problèmes complexes tels que jouer aux échecs, prouver des théorèmes, percevoir l'environnement et s'y déplacer sans encombre… De la même manière qu'un ingénieur en aéronautique n'a pas pour objectif d'imiter le vol des oiseaux, la question de la « véritable intelligence » ne présente que peu d'intérêt pour un spécialiste en intelligence artificielle (voir *Le test de Turing* en encadré). Pourtant, même si à première vue les deux disciplines ne s'intéressent pas aux mêmes problèmes scientifiques, des liens profonds existent entre philosophie et informatique.

L'atomisme logique et le joueur d'échecs

L'histoire de l'informatique montre que cette jeune discipline est parfois considérée comme de la *philosophie appliquée*. Elle emprunte alors à la philosophie de l'esprit des modèles et des théories pour procéder à ses développements techniques. Pour l'*atomisme logique*, courant majeur de la philosophie occidentale du début du XX^{e} siècle, toute connaissance peut être décomposée en propositions logiques élémentaires afin de décrire et de raisonner sur le monde réel. Le *computationnalisme*, courant dominant de l'intelligence artificielle jusqu'aux années 1980, est née de la volonté d'utiliser des calculateurs pour automatiser ces raisonnements. Par exemple, en formalisant les règles du jeu d'échecs et les différentes stratégies possibles, les premiers joueurs artificiels ont rapidement égalé les joueurs humains, jusqu'à battre aujourd'hui les meilleurs professionnels. Cependant, s'il est relativement facile de décomposer le « micro-monde » du jeu d'échecs en propositions logiques élémentaires, les applications informatiques de l'atomisme logique ont rencontré de lourdes difficultés pour résoudre les problèmes plus complexes du monde réel : comprendre et parler le français, reconnaître des objets sur une photographie, se déplacer dans la rue sans heurter les piétons...

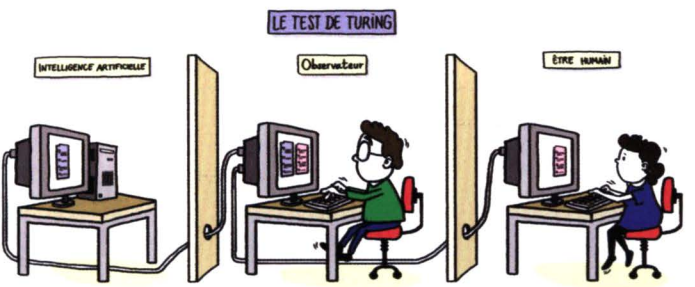
Le philosophe Hubert Dreyfus montre, dans son ouvrage intitulé *What Computers Can't Do*, comment l'idée d'un programme générique, capable de résoudre « tout type de problèmes », est finalement abandonnée par le *computationnalisme*.

L'énaction et l'apprentissage des robots

Vu le nombre de paramètres à prendre en compte lors de la décomposition logique, « se déplacer » est bien plus difficile pour un robot que « jouer aux échecs ».

Apparu dans les années 1980, le modèle *énactif* de l'esprit tient compte de

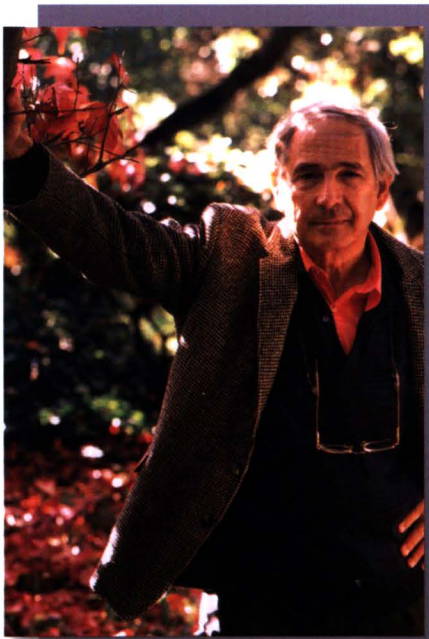
Le test de Turing et l'intelligence simulée



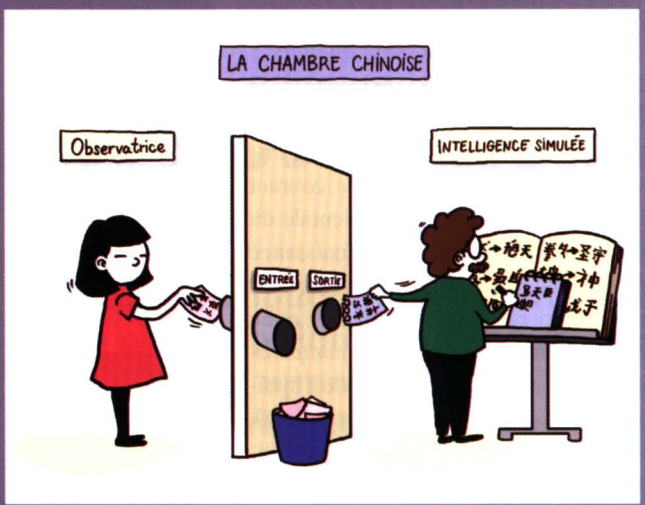
Alan Mathison Turing (1912–1954) est un mathématicien et informaticien britannique reconnu comme étant l'un des pères fondateurs de l'intelligence artificielle. Dans un article publié en 1950, il propose de répondre à la question épineuse « est-ce que les machines peuvent penser ? » par un dispositif extrêmement simple : si un observateur, discutant avec une machine et un autre être humain, se retrouve incapable de distinguer, au cours de la conversation, lequel de ses deux interlocuteurs est humain, alors il est nécessaire d'admettre que la machine est aussi intelligente que cet être humain. Le test de Turing s'appuie donc sur une définition *comportementale* de l'intelligence, c'est-à-dire indépendante du *fonctionnement* interne de la machine.

Si Turing s'intéresse néanmoins à des questions philosophiques telles que la conscience, les émotions et la créativité, le dispositif qu'il propose répond à une volonté de résoudre un problème propre à l'informatique : comment déterminer si une machine est ou non intelligente ? La réponse qu'il donne, par la seule observation des comportements, est une posture propre aux spécialistes de l'intelligence artificielle qui cherchent à simuler les comportements intelligents, sans exiger de qualités internes particulières. Le 7 juin 2014, lors d'un test organisé par la *Royal Society* à l'occasion du soixantième anniversaire de la mort d'Alan Turing, un programme informatique a réussi à se faire passer pour un garçon de 13 ans auprès de plus de 30 % des observateurs interrogés après des conversations libres de cinq minutes.

Searle et la chambre chinoise



John Searle (né en 1932).



John Searle est un philosophe britannique qui a beaucoup critiqué la définition comportementale de l'intelligence donnée par le test de Turing. Sa célèbre expérience fictive de la « chambre chinoise » consiste à dire que, si un locuteur anglais était enfermé dans une pièce avec à sa disposition un dictionnaire de conversation chinoise, lui permettant de répondre intelligemment à des caractères chinois par d'autres caractères chinois sans jamais comprendre le sujet de la conversation, un observateur extérieur serait convaincu que l'homme à l'intérieur de la pièce parle parfaitement chinois. Pourtant, celui-ci ne comprend pas un mot ! Il y a donc un écart entre les comportements que l'on observe et l'état interne de la personne observée : l'intelligence n'est pas juste une affaire de comportement et il est possible de concevoir une machine « stupide » qui se comporte *comme si* elle était intelligente. Searle a par ailleurs une posture propre aux philosophes de l'esprit concernant le concept d'intelligence artificielle : s'il s'interroge sur l'existence chez les machines de qualités internes telles que la conscience et l'intentionnalité, il ne s'intéresse pas aux techniques informatiques permettant de simuler les comportements intelligents ou de résoudre des problèmes complexes. Entre Turing et Searle, le fossé séparant les questions d'ordre informatique et celles d'ordre philosophique semble infranchissable.

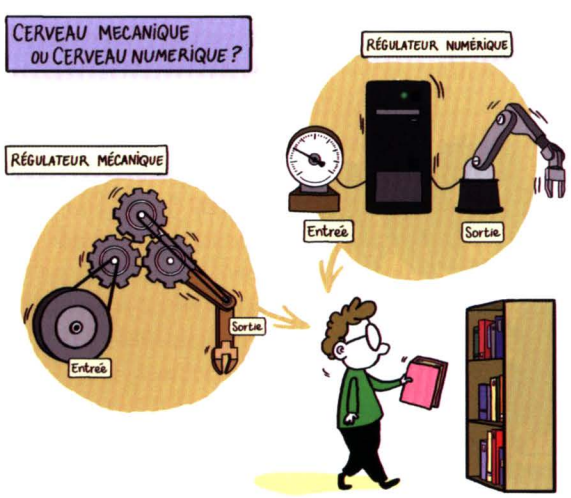
l'importance des perceptions dans la cognition et rompt ainsi avec le modèle cartésien affirmant que le corps et l'esprit sont deux substances indépendantes. Appliqué à la robotique, cette position philosophique invite à concevoir des machines qui ne sont pas programmées pour une tâche ou un problème spécifique, mais qui apprennent de manière autonome à percevoir et à résoudre le problème sans décomposition logique préalable de la part du concepteur. Pour un robot qui apprend à marcher, le mouvement des différentes parties du corps produit des changements dans la perception de l'environnement. Certains de ces changements ne sont pas prédictibles, tels que bouger toutes les pattes en même temps pour une araignée mécanique. D'autres mouvements, plus organisés, permettent cependant au robot de découvrir son environnement de manière efficace. En tâtonnant, il peut donc trouver de lui-même les mouvements

optimaux et apprendre à interagir avec l'environnement en fonction des moteurs et des capteurs dont il dispose. En appliquant le modèle *énactif* de l'esprit, l'intelligence artificielle propose donc des programmes intelligents autonomes sans passer par une décomposition formelle du problème par le concepteur.

L'informatique pour évaluer les théories philosophiques et en créer de nouvelles

Si la philosophie apporte donc les bases conceptuelles pour les développements de l'intelligence artificielle, la collaboration ne reste pas unilatérale. Pour Daniel Andler, chercheur en mathématiques et en philosophe des sciences, l'intelligence artificielle invite notamment la philosophie à *« préciser ses choix, à dissiper des malentendus millénaires, à formuler et à tester de manière scientifique de nouvelles hypothèses »*. L'informatique s'immisce ainsi dans les controverses philosophiques en proposant, par l'expérimentation, de nouveaux arguments. Elle obtient le statut inattendu de *philosophie expérimentale*. Les difficultés pratiques des programmes *computationnalistes* ont par exemple mis en évidence des présupposés erronés dans les théories philosophiques qu'ils tentaient d'appliquer. S'il est impossible de concevoir un robot capable de se déplacer en adoptant les mêmes méthodes que celles utilisées pour le joueur d'échecs, c'est qu'il est peut-être impossible en vérité de décomposer *toute* connaissance en propositions logiques élémentaires. L'intelligence artificielle, grâce à une longue série d'expériences infructueuses, a permis de mettre l'accent sur la nécessité de prendre en compte le corps et l'environnement pour concevoir des machines intelligentes. Plus généralement, puisque les différentes théories philosophiques affectent les choix de conception informatique, elles induisent une différence *en pratique*. Il est donc possible d'évaluer ces théories indirectement, à partir de leurs applications.

« Faire de la philosophie de l'esprit avec un tournevis. » Cette amusante formule du spécialiste en intelligence artificielle Inman Harvey invite à observer toutes sortes de machines pour s'interroger sur l'homme et son cerveau. Les nombreux développements de la robotique fondée sur le modèle *énactif* de l'esprit encouragent en retour la philosophie à considérer ce modèle comme une véritable alternative au *computationnalisme* et relancent le débat sur le rôle du corps dans l'apprentissage et les autres fonctions cognitives.
Timothy van Gelder, chercheur en sciences cognitives, donne également de nouvelles pistes philosophiques en s'intéressant à un problème technique de la révolution industrielle : comment réguler l'énergie produite par une machine à vapeur pour alimenter un métier à tisser ? Le « régulateur à boules » imaginé par James Watt à l'époque utilise la force centrifuge d'un volant pour ajuster une valve d'ad-



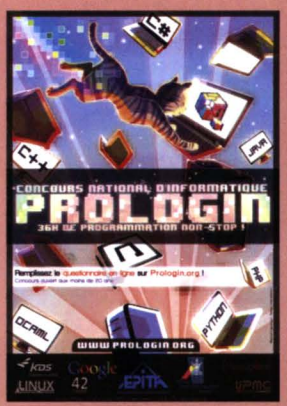
mission. Il s'agit d'un dispositif entièrement mécanique. Aujourd'hui, on utiliserait au contraire un dispositif numérique, c'est-à-dire un ordinateur capable de *mesurer* la vitesse de rotation du volant et de *calculer* la position de la valve pour apporter la bonne quantité d'énergie. Ces deux dispositifs parviennent néanmoins à résoudre le problème posé. Ils ouvrent ainsi deux voies possibles pour comprendre d'autres systèmes de régulation, tels que le cerveau : les fonctions cognitives de l'homme sont-elles similaires à celles d'un régulateur numérique ou à celles d'un régulateur mécanique ? À celles de l'ordinateur jouant aux échecs ou à celles du robot apprenant à se déplacer par essais et erreurs ? Si l'observation de ces deux systèmes de régulation ne permet pas de trancher en faveur de l'une ou de l'autre des deux hypothèses, elle a cependant le mérite de révéler et d'expliciter ces hypothèses par des exemples concrets, de proposer des pistes de validation théoriques, tout laissant à la philosophie de l'esprit le soin de les départager.

L'informatique peut également participer à l'édification de nouvelles théories philosophiques. Un exemple de cette démarche est donné par l'utilisation des tables de multiplication. Leur apprentissage est coûteux en mémoire : il faudrait un peu moins de 5,000 Go pour stocker les tables des nombres à six chiffres. L'algorithme traditionnel que l'on apprend à l'école primaire consiste à décomposer les multiplications à plusieurs chiffres en multiplications à un seul chiffre et en additions. Ce procédé, économe en termes d'espace mémoire, nécessite néanmoins un peu plus de temps de calcul. Mais il repose sur une compréhension plus profonde de la notion de multiplication dans la mesure où il fait intervenir le produit croisé de puissances décimales.
Ainsi l'analyse de l'algorithme permet de se rendre compte de ce qu'est *véritablement* une multiplication en donnant une intuition mathématique. Mieux, il permet d'ouvrir des pistes pour la compréhension des mécanismes cognitifs humains. L'argument est le suivant : *un programme simple ne peut pas, en pratique, résoudre un problème complexe*. Personne, pas même un supercalculateur, ne peut exécuter une tâche donnée (par exemple, multiplier des grands nombres) sans employer une méthode « intelligente ». Dès lors, tout programme résolvant un problème complexe est lui-même assez sophistiqué pour mériter que l'on s'y intéresse.

Dans le cas de l'intelligence artificielle, contrairement à ce qu'en dit John Searle dans son expérience de la « chambre chinoise » (voir encadré), une machine passant le test de Turing avec succès ne peut pas, en pratique, être complètement « stupide ». Une telle machine constitue dès lors une piste expérimentale pour modéliser et expliquer la complexité des facultés humaines.

M.B. et R.L.-P.

Bibliographie
- Dreyfus, H.L. 1979. *Intelligence Artificielle : mythes et limites*. Paris : Flammarion.
- Lamarche-Perrin, R. 2012. *Des collaborations possibles entre philosophie et Intelligence Artificielle*. Mémoire de Master de philosophie de l'Université Pierre-Mendès-France, Grenoble.
- Searle, J.R. 1984. *Du cerveau au savoir : conférences Reith 1984 de la BBC*. Paris : Hermann, 1985.
- Turing, A.M. 1950. « Computing Machinery and Intelligence. ». *Mind*, vol. 59, p. 433-460.
- Varela, F.J. 1988. *Invitation aux sciences cognitives*. Paris : Seuil, 1996.



Le concours Prologin

L'association Prologin organise depuis 1992 un concours national d'informatique qui s'adresse aux jeunes de 20 ans et moins.

Le concours débute d'octobre à janvier avec une épreuve de sélection composée d'un questionnaire de culture informatique et de problèmes algorithmiques de difficulté croissante.

À l'issue de cette phase, les meilleurs candidats sont qualifiés pour les épreuves régionales organisées dans plusieurs villes de France et de Belgique : le matin, une épreuve écrite d'algorithmique assortie d'un entretien et l'après-midi, une épreuve « machine » de programmation.

Dernière étape, les 100 meilleurs candidats sont conviés à la grande finale qui se déroule pendant trois jours à l'EPITA à Paris : chaque candidat dispose de trente-six heures pour programmer une intelligence artificielle (appelée « champion »). Les champions des candidats s'affrontent ensuite deux à deux au sein d'un tournoi et les dix meilleurs passent devant un jury et remportent les lots.

Pour plus d'informations, rendez-vous sur le site du concours : http://prologin.org

Le concours Castor Informatique

Au mois de novembre, l'ENS Cachan, l'association France-IOI et Inria ont organisé la troisième édition française du concours Castor Informatique.

Le principe est simple : seuls ou en binômes, des collégiens et lycéens ont 45 minutes pour résoudre 18 exercices permettant de découvrir des concepts informatiques. La plupart des questions se présentent sous la forme d'animations interactives où les élèves construisent petit à petit leur réponse en interagissant avec l'ordinateur.

Ne nécessitant aucune connaissance préalable en informatique, le Castor vise à faire découvrir aux jeunes ce qu'est l'informatique en tant que science, ainsi qu'à suggérer le type de raisonnement mis en œuvre en programmation et en algorithmique. Et ça marche !

Depuis trois ans, le nombre de participants français a quasiment doublé chaque année. Ils étaient plus de 170 000 en 2013, dont 48 % de filles.

Le concours, entièrement gratuit, se déroule début novembre dans une salle informatique de chaque établissement participant sous la surveillance d'un enseignant. Comme dans les 28 autres pays où il est organisé, le Castor est décliné en France en différents niveaux : 6e/5e, 4e/3e, 2de et 1re/Terminale. Peu après la fin du concours, des corrections sont publiées, ainsi que des informations pour aller plus loin sur les thèmes abordés. Tous les participants reçoivent un diplôme, et les meilleurs gagnent un lot.

Pour plus d'informations, ainsi que pour jouer les sujets des éditions précédentes, n'hésitez pas à vous rendre sur le site du concours : http://castor-informatique.fr

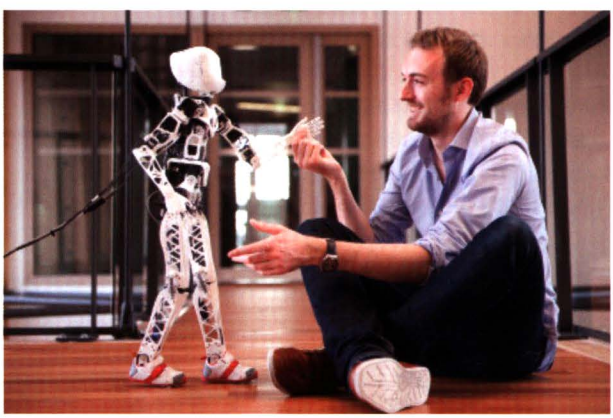
Poppy, premier robot humanoïde imprimé en 3D

Le robot Poppy a été présenté en 2014 à François Hollande à l'Élysée, dans le cadre d'un évènement FrenchTech pour soutenir le développement de l'économie du numérique (des vidéos sont disponible en ligne sur les sites de DailyMotion et de Vimeo). Poppy est le premier robot humanoïde complet au monde à être la fois *open-source* et imprimé en 3D. Il est destiné principalement au monde de l'éducation, des FabLabs (associatifs ou dans les entreprises), des *geeks* et des artistes.

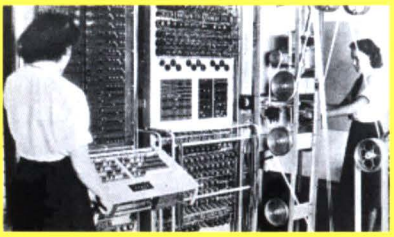
Dès 2014, les premiers lycées, écoles d'ingénieur et FabLab commenceront à l'utiliser dans leurs formations. Il est prévu que cet usage se multiplie rapidement afin de contribuer à l'appropriation du monde numérique et de l'impression 3D par le plus grand nombre.

Tout le monde peut télécharger librement les plans de Poppy et le logiciel associer, le construire, le « hacker », et inventer de nouvelles version !

Référence :
http://www.poppy-project.org



La révolution cryptographique



Colossus, premier ordinateur à Bletchley Park pendant la Seconde Guerre mondiale.

L'informatique est née en partie du besoin de décrypter les messages allemands de la Seconde Guerre mondiale. Des méthodes pour casser facilement le code de la machine Enigma, comme celle de *l'indice de coïncidence*, existaient mais n'ont pu être utilisées du fait de l'absence de moyens de calculs suffisants. C'est pourquoi Alan Turing et son équipe optèrent pour une méthode moins systématique, la recherche de mots probables, qu'heureusement l'armée allemande fournissait en abondance, dans les bulletins météo par exemple. Le premier ordinateur (Colossus) fut construit par les Britanniques à la même époque pour décrypter le code d'une autre machine de chiffrement, Lorenz, réservée aux hauts dirigeants allemands alors qu'Enigma servait sur le champ de bataille, en particulier dans les sous-marins. Le code de Lorenz et le relatif faible nombre de messages le permettaient.

Après la guerre, ces progrès menèrent à l'introduction de nouvelles méthodes, comme les clefs asymétriques où, contrairement aux clefs symétriques, savoir coder ne suffit pas pour savoir décoder. La plus célèbre et la plus utilisée d'entre elles, en particulier dans les cartes bancaires et sur Internet, est la méthode RSA, du nom des trois inventeurs : Rivest, Shamir et Adleman. Elle repose sur la difficulté pratique de factoriser les nombres quand ils sont grands.

Mathématiques pour l'informatique

L'informatique n'existerait pas sans les mathématiques. Les procédures (et les processeurs) qu'elle utilise, les algorithmes et les structures des langages qui permettent la programmation, la vérification des programmes et le calcul de leur durée, s'appuient sur des théories mathématiques dont certaines ont été créées spécialement dans ce dessein. Système binaire, algèbre de Boole, notion de complexité, et bien d'autres, ont apporté à l'informatique le support théorique qui en fait une science à part entière.

La préhistoire de l'informatique

De la procédure algorithmique aux premières machines à calculer, les origines de l'informatique sont bien plus anciennes que ce qu'on pourrait imaginer, même si ce sont les progrès de la physique qui ont permis le passage à l'acte. Histoire de précurseurs.

Le développement de l'informatique a été rendu possible par les progrès immenses des siècles derniers en sciences physiques, en particulier dans le domaine des matériaux. Mais il n'aurait pas été possible sans des avancées mathématiques ou logiques préalables. C'est même depuis très longtemps que certains ont imaginé des méthodes itératives pouvant aboutir à un résultat ou des machines capable de réaliser des opérations automatiques.

Les premiers algorithmes

L'ordinateur ne réalisant que des opérations programmées à l'avance et s'adaptant en fonction des résultats obtenus aux étapes précédentes, la notion d'algorithme y prend tout son sens. C'est sans doute chez Euclide qu'on voir apparaître pour la première fois ce type de méthode.

Dans le livre VII des *Éléments*, il explique, dans ses deux premières propositions, la méthode effective pour obtenir le PGCD de deux nombres : c'est ce qu'on appelle de nos jours *l'algorithme d'Euclide*.
Dans l'Antiquité, on peut citer aussi le crible imaginé par Ératosthène pour obtenir la liste des nombres premiers. Les deux hommes ont vécu à Alexandrie au troisième siècle avant J.-C., le premier au début, le second au milieu.

Mais d'où vient le mot même d'algorithme, curieuse anagramme du mot logarithme ? Il provient de la déformation du nom du mathématicien arabe du début du dixième siècle Mohammed al-Khwarizmi.
Dans son célèbre ouvrage *Kitab al-jabr* (on y reconnaît une autre origine, celle du mot *algèbre*), il a donné une méthode mécanique de la résolution des équations du second degré ; pour lui, il y en avait six types une fois ôtées toutes les expressions négatives. L'attribution de son nom pour désigner ce type de raisonnement n'est donc pas usurpée.

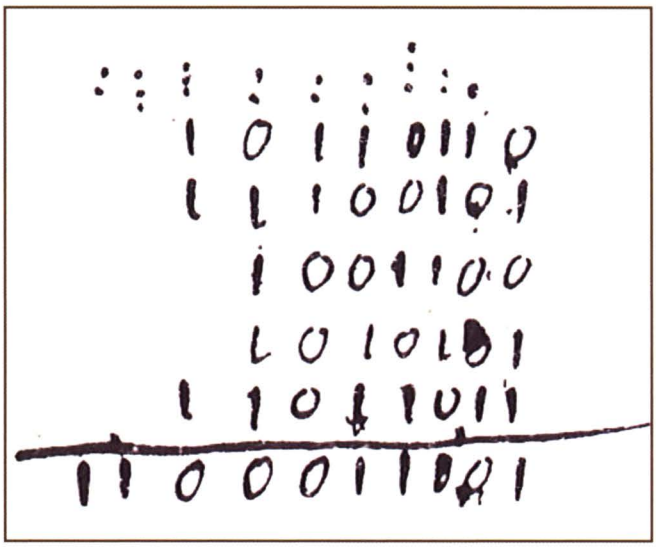
Les premières machines à calculer

Les premiers instruments mécaniques pour permettre le calcul sont les bouliers, ou abaques, conçus dans différentes civilisations dès l'Antiquité ; on ne peut cependant pas les considérer comme des machines.
Gerbert d'Aurillac, érudit éclairé, rapporta de ses voyages chez les Maures d'étranges machines et en conçut lui-même avec tant d'habileté qu'on le soupçonna d'avoir vendu son âme au diable, ce qui ne l'empêcha pas d'être élu pape en 999 sous le nom de Sylvestre II.

John Napier, l'inventeur des logarithmes en 1614, mit au point un système de baguettes coulissantes sur lesquelles étaient notées les tables de multiplication des entiers de 1 à 9 qui permettaient ainsi d'effectuer des multiplications. Peu après, Wilhelm Schickhard s'en inspira pour imaginer une *horloge calculante* ; d'après le courrier qu'il envoya à son ami Kepler, le haut était constitué de baguettes coulissantes alors que des roues dentées en bas permettaient d'enregistrer les retenues. Malheureusement sa machine non encore achevée fut détruite l'année suivante dans un incendie.

Blaise Pascal, âgé de 19 ans, se détache du principe de l'horloge. Celui-ci ne permettait pas d'effectuer des retenues en cascade pour des raisons mécaniques (trop forte poussée qui bloquait la machine). Il conçoit des rouages miniaturisés, dits *à lanternes*, qui résistent à des secousses très fortes alors que des systèmes de sautoirs permettent les retenues. Cette machine fonctionnait sans intervention humaine intermédiaire. C'est pourquoi on considère Blaise Pascal comme le premier inventeur d'une machine à calculer.

Sylvestre II.



Addition en binaire extrait de http://www.bibnum.education.fr/files/Leibniz-analyse.pdf

Nommée *Pascaline*, cette machine fut construite et commercialisée mais sans grand succès. Certes le prix en était élevé, mais on pense surtout qu'elle inspirait à l'époque une certaine méfiance, ce qui fit dire à Voltaire : *« Pascal, fou sublime né un siècle trop tôt. »*

Pour plusieurs raisons, Leibniz peut être considéré comme un précurseur de l'informatique. Dans un manuscrit *De Progressione Dyadica*, datant de 1673 mais non publié, il introduit le système binaire et effectue des opérations. Il conçoit alors la possibilité de mécanisation du calcul. Il reprend en détail ces idées dans son article *Explication de l'arithmétique binaire* publié en 1703.

Fort de ces idées, Leibniz conçoit en 1673, une machine dont le principe est assez différent de celui de la Pascaline. Elle se compose de cylindres cannelés qui entraînent des roues dentées. Seul deux exemplaires ont été construits, l'un en 1694 et l'autre en 1706. Il en explique le fonctionnement en détail dans un article paru en 1710 et se réjouit en affirmant : *« grâce à ma Machine (...) les calculs pouvaient être menés à bonne fin par un petit enfant. »*

En fait, cette machine ne marchera jamais réellement. L'ingénieur Thomas de Colmar l'incorporera en 1820 dans son arithmomètre, une nouvelle machine à calculer qui aura un certain succès commercial.

C'est à la même époque que Charles Babbage imagine une machine à cartes perforées (voir article suivant) ; même si elle ne vit jamais le jour, on peut la considérer comme ancêtre des premiers ordinateurs.

B.H.

La Pascaline, collection du musée des Arts et métiers.

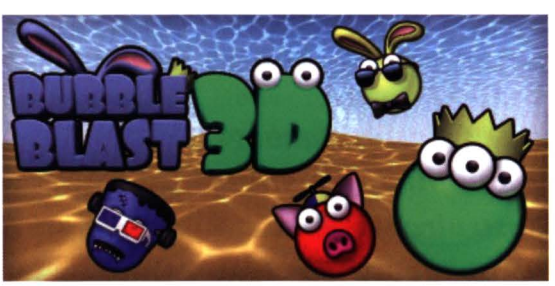
Nicolas Sorel, créateur de Magma Mobile : « Restez dans les clous et persévérez ! »

Magma Mobile connaît un fort développement depuis sa création en 2009. Comment expliquez-vous ce succès ?

« Magma Mobile a effectivement développé environ cent cinquante jeux, toutes plateformes confondues (Android, Iphone et Windows Phone). Nous étions là au bon moment, et avons su saisir les opportunités : en 2009, il n'y avait pas d'Android en France et on a commencé à faire des applications pour ce système d'exploitation mobile. On a écrit un livre, fait des formations vidéo... On s'est lancé à trois, maintenant nous sommes une quinzaine, stagiaires compris. Aujourd'hui on a plus de trois cents millions de téléchargements de nos jeux ! »

Les applications que vous développez font-elles appel à des algorithmes complexes ou à des notions mathématiques sophistiquées ?

« Tout à fait ! Depuis le début, notre savoir-faire c'est les jeux du type "brain and puzzle" [NDLR : cerveau et casse-tête] *et les jeux de réflexions. Dans Connect'em par exemple, on doit relier, de façon stratégique et selon des considérations arithmétiques, des petits personnages marqués par des entiers naturels. Nous créons aussi des jeux de type Bubble Blast, où il faut calculer la longueur d'une réaction en chaîne. Nous avons également développé un jeu d'échecs très pointu, un jeu de Reversi ainsi que tous les jeux classiques qui font appel à l'intelligence artificielle. »*



En tant que partenaire du prix Bernard-Novelli et des Trophées Tangente, quels conseils donneriez-vous aux jeunes développeurs ?

« Rester dans les clous ! S'ils se fixent un objectif, qu'ils ont une idée, et qu'ils veulent la réaliser, il faut que les jeunes développeurs se donnent les moyens de le faire. Il n'y a rien de magique ! Ce n'est pas parce qu'on va faire un jeu et qu'on va le "mettre dans un store" que la magie va opérer. C'est un peu une vue de l'esprit actuellement : déposer un jeu, mettre beaucoup d'efforts dedans, le déposer et ensuite constater que ça ne marche pas et passer à autre chose.

Le conseil, c'est de mettre beaucoup d'énergie dans la création d'un jeu ou d'un programme. Après, il faut continuer, trouver des moyens de le faire connaître, d'en faire parler, ne pas lâcher l'affaire. Par contre, le chemin ne sera pas facile : il y a une grosse concurrence, avec un million d'applications sur les stores. Pendant les moments durs, il faut persévérer et faire confiance à son intuition. »

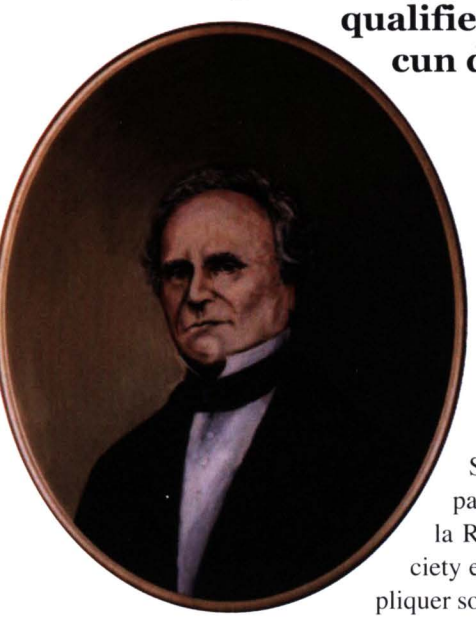
Quelles sont selon vous les grandes tendances qui semblent se dessiner aujourd'hui ?

« Avant, les jeux payants étaient les plus rémunérateurs. Les choses ont changé : les Coréens, je crois, ont introduit l'achat intégré dans l'appli ("in app purchase"). En ce moment, c'est ce mode-là qui prévaut : l'application est gratuite, puis au bout d'un moment, une fois qu'on est bien accroché, on va passer à la caisse. Chez nous, tous les jeux sont gratuits, et on les monétise avec de la publicité. On est dans un autre modèle. »

par Hervé Lehning

Babbage
et le premier ordinateur potentiel

Au XIX^e siècle, Charles Babbage conçut la première calculatrice scientifique ainsi que le premier instrument que l'on puisse qualifier d'ordinateur. Malheureusement, aucun des deux ne fonctionna à l'époque.



Charles Babbage (1791–1871). Portrait par Virginia Kolence.

Charles Babbage se consacra très jeune aux mathématiques. Après des études au Trinity College de Cambridge, il devint membre de la Royal Society en 1816 et fut parmi les fondateurs de la Royal Astronomical Society en 1820, ce qui peut expliquer son intérêt pour les tables de fonctions numériques, indispensables aux calculs astronomiques, en particulier.

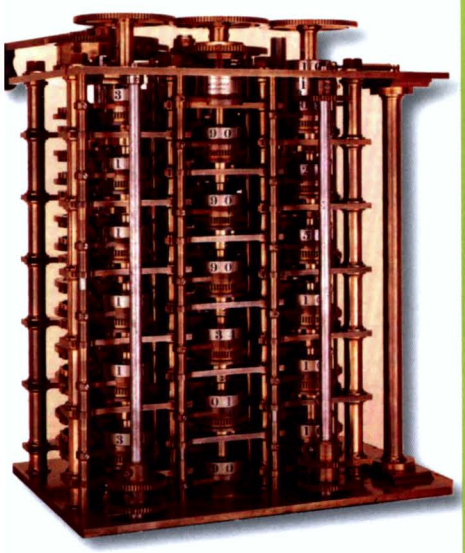
Les tables de l'époque, confectionnées à la main, contenaient un grand nombre d'erreurs, ce qui conduisait parfois à des catastrophes, en particulier des naufrages puisqu'elles étaient utilisées dans la navigation. Le but initial de Babbage fut donc de créer des tables de fonctions sans erreur de calcul. Pour cela, il conçut une machine capable de calculer les valeurs d'un polynôme à partir de ses différences finies (voir l'encadré). En utilisant l'approximation des fonctions par des polynômes, cela permet de dresser les tables des fonctions usuelles, comme les fonctions trigonométriques ou logarithmiques.

La méthode utilisée explique le nom donné à ce calculateur : *la machine à différences*.

La calculatrice de Charles Babbage a été réalisée à la fin du XX^e siècle et on projette de construire l'ordinateur en ce début de XXI^e siècle…

En 1821, un premier modèle partiel, mais fonctionnel, de la machine à différences fut présenté à la Royal Astronomical Society. Celle-ci s'intéressa au projet et, en 1823, le gou-

vernement britannique accorda une bourse de 1 500 £ afin que Babbage puisse construire l'ensemble complet. Malheureusement, ce ne fut pas le cas de son vivant, en partie parce qu'il perfectionnait sans cesse sa machine, sans attendre qu'un modèle soit achevé.
Les principes de cette machine restent classiques. Comme les précédentes, celle de Blaise Pascal en particulier, elle fonctionne avec des roues dentées, correspondant aux chiffres de 0 à 9.



Un élément de la machine à différences, construit en 1832, on remarque qu'elle calcule sur des nombres décimaux.

L'élément construit fut utilisé pour montrer que l'ensemble était réalisable, même s'il ne le fut jamais du vivant de Babbage. En fait, celui-ci avait conçu deux modèles de sa machine. C'est la seconde que l'on a construit à la fin du XX^{e} siècle, après avoir corrigé quelques erreurs mineures. La plus gênante concernait le mécanisme des retenues, qui ne pouvait fonctionner tel qu'il était décrit… même si son principe était correct.

La méthode des différences finies

Soit Δ la fonction qui, à un polynôme P, associé le polynôme ΔP défini par : $\Delta P\,(x) = P\,(x + 1) - P\,(x)$.
On remarque immédiatement que le degré de ΔP est égal à celui de P diminué d'une unité.
Ainsi, si P est de degré 3, ΔP est de degré 2, $\Delta^2 P = \Delta(\Delta P)$ est de degré 1 et $\Delta^3 P = \Delta(\Delta^2 P)$, de degré 0, c'est-à-dire est une constante, $\Delta^4 P = 0$.

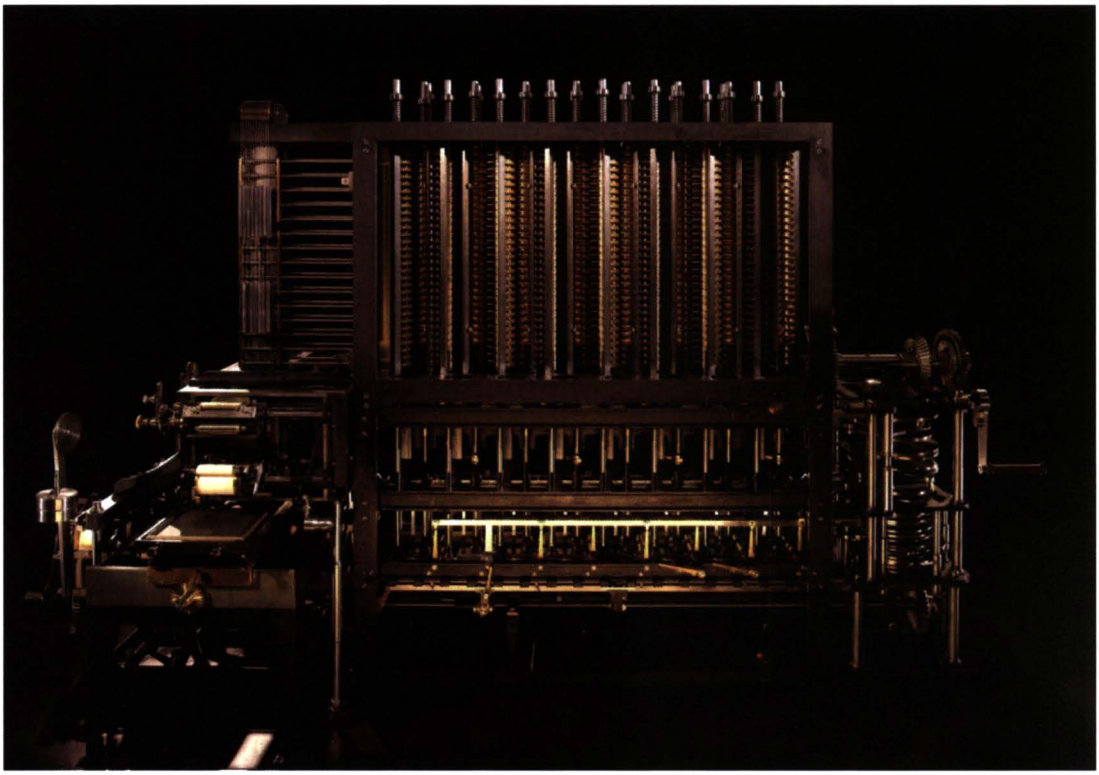
En inversant la méthode, de simples additions suffisent pour calculer les valeurs de P à condition de connaître les valeurs sur la diagonale, en jaune dans le tableau. La machine à différences de Babbage, si elle avait été réalisée, aurait été capable d'effectuer ces calculs pour des polynômes de degré au plus 7 et des nombres comportant jusqu'à 31 chiffres.

x	x^3	Δ	Δ^2	Δ^3	Δ^4
0	**0**				
		1			
1	1		**6**		
		7		**6**	
2	8		12		**0**
		19		6	
3	27		18		0
		37		6	
4	64		24		
		61			
5	125				

Différences finies de x^3 appliquées aux premiers entiers (lire verticalement l'image de 0, 1, 2,…)

La machine analytique

Sans avoir réussi à construire sa machine à différences, Charles Babbage entreprit un projet plus ambitieux encore : *la machine analytique*, premier ordinateur qui reste… potentiel, car lui, il ne fut jamais réalisé. Pour comprendre ses principes, mieux vaut savoir que Babbage s'est intéressé au métier à tisser de Jacquard, qui utilisait des cartes perforées : les



La machine à différence, construite selon les plans de Babbage en 1991. Sur la droite, on remarque la manivelle qui actionne la machine, sur la gauche, l'imprimante et sa manivelle.

pleins des cartes servaient à repousser les aiguilles, le contraire pour les creux qui les laissaient passer…
C'est ainsi que le tissage d'une pièce était commandé par une séquence de cartes, qui constituait ainsi un programme avant la lettre.

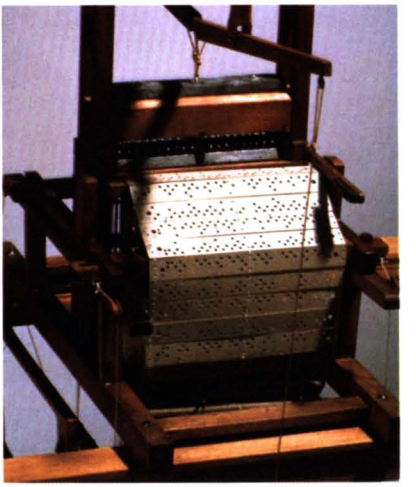
Contrairement à la machine à différences, qui ne pouvait effectuer qu'un calcul, la *machine analytique* devait pouvoir suivre n'importe quel algorithme compatible avec sa conception. Ainsi, elle prévoyait des boucles et des structures conditionnelles.
Babbage choisit d'entrer ses algorithmes dans la machine, comme Jacquard entrait ses motifs : avec des cartes perforées. Il nomma *unités d'entrée* les parties permettant de communiquer l'algorithme à effectuer ainsi que les données. Pour stocker ces données et les résultats intermédiaires, ainsi que le résultat final, il conçut une autre unité, qu'il nomma le *magasin*, qui fonctionnait avec des roues dentées et des cartes perforées. C'est l'équivalent, de nos jours, de la mémoire. Le magasin comportait également des unités de sortie : perforatrices (ou plutôt leurs ancêtres) et imprimantes.

C'est également en s'inspirant des métiers Jacquard que Babbage décida de séparer le mécanisme de commande de la machine, qu'il nomma l'*unité de commande*, du mécanisme d'exécution des opérations logiques et

arithmétiques, qu'il appelait le *moulin*. Les deux correspondent aux processeurs des ordinateurs modernes.
Cette machine, entièrement numérique donc mathématique, ne fut jamais construite car, outre les difficultés techniques, l'époque était plutôt favorable au calcul analogique, plus inspiré par la physique (voir ci-dessous).

En ce début de XXI[e] siècle, un projet a vu le jour pour construire une machine analytique. Il porte le nom de *plan 28*, qui est le numéro du dessin de Babbage représentant le mieux son projet. On peut trouver le projet et ses détails sur Internet.

H.L.

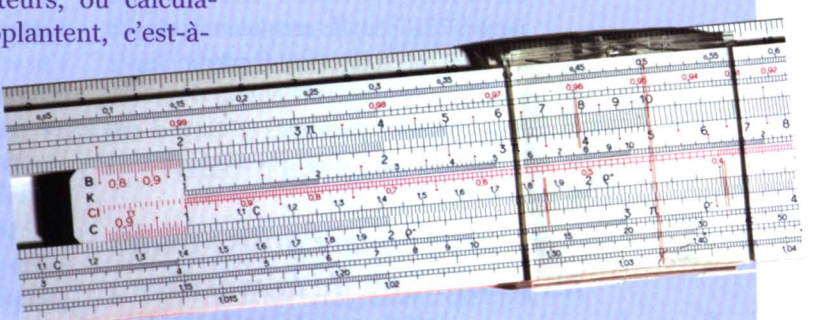


Détail d'un métier Jacquard montrant les cartes perforées.

Le calcul analogique

Pour donner un exemple simple, voire simpliste, de calcul analogique, une autre façon d'effectuer les additions est d'utiliser les propriétés des longueurs : deux mètres plus trois mètres font cinq mètres. Ainsi, avec deux règles graduées, on peut facilement opérer une addition. La fonction logarithme transformant une multiplication en addition donne alors une méthode analogique pour calculer un produit. Il suffit de transformer l'échelle linéaire en échelle logarithmique. On obtient un instrument de calcul, autrefois symbole de l'ingénieur.

De façon plus générale, l'idée du calcul analogique est de représenter les nombres par des grandeurs géométriques (longueurs, aires, volumes, angles) ou physiques (mécaniques, électriques, hydrauliques, chimiques), et d'exploiter des phénomènes géométriques ou physiques dont la modélisation mathématique est fondée sur les équations que l'on veut résoudre.
En particulier, des systèmes électriques permettent de résoudre automatiquement certaines équations : celles qui les régissent.
Les calculateurs analogiques ont été en usage jusqu'à ce que les ordinateurs, ou calculateurs numériques, les supplantent, c'est-à-dire jusqu'au début des années 1970. Dans le domaine du calcul scientifique, numérique est ainsi devenu l'opposé d'analogique.



Une règle à calculs est composée de trois réglettes dont une coulisse entre les deux autres. En faisant coïncider la graduation 1 de l'une et la graduation 2 de l'autre, puis en alignant le curseur sur la graduation 5 de la première, on lit le résultat de la multiplication 2 × 5 sur la seconde.

Le programme de Lady Ada King, comtesse de Lovelace

Lady Ada King est le personnage romantique par excellence : jeune, belle, indépendante, intelligente, unique fille légitime du célèbre poète britannique lord Byron, elle décédera comme lui dans la fleur de l'âge, à 36 ans. L'informatique moderne la rendra célèbre.

Lady Ada King est certes la fille du célèbre poète Lord Byron, mais également de la mathématicienne Annabella (ou Anne Isabella) Milbanke. Ses parents se sépareront avant sa naissance et en fait Ada ne connaîtra jamais son père... Annabella choisit pour sa fille une éducation musicale et scientifique. Le 5 juin 1833, elle rencontre Charles Babbage lors d'une réception, elle a alors 17 ans. Babbage expose les principes de sa première machine à Ada, qui se trouve immédiatement fascinée. Ils deviennent rapidement amis et Babbage lui parle d'une nouvelle machine, la Machine analytique, beaucoup plus élaborée, dont il passera le reste de sa vie à peaufiner les plans dans l'espoir de la faire construire (il ne la verra jamais réalisée). L'architecture de son appareil est étonnement moderne. Jugez plutôt : elle s'articule autour d'un lecteur de cartes perforées (l'unité d'entrée), inventé quelques décennies auparavant par Jacquard pour les métiers à tisser, d'un magasin (la mémoire) et d'un moulin (le processeur).

Le programme utilise toutes les possibilités de la machine : branchements conditionnels et boucles.

Un siècle d'avance !

En 1840, Babbage présente, pour la première et unique fois, sa machine à un groupe d'ingénieurs et mathématiciens italiens, dont Luigi Federico Menabrea,

Le pseudo-code du programme d'Ada

```
1 -> Bernoulli[0]
Pour n de 1 à n faire
  0 -> Somme
  (2n−1)/(2n+1) -> A[0]                          # évaluation de A0
  A[0] + Bernoulli[0] -> Somme                   #AiBi -> Somme
  Si n == 1 alors −Somme -> Bernoulli[1]         # B1 = −AiBi
  Pour i de 1 à n faire
    Si i == 1 alors n -> A[1]                    #A1 = n
    Sinon
      A[1] -> Produit                            # Produit = A1
      Pour j de 0 à 2(i−1) faire
        Produit*(2n−1−j)/(3+j) -> Produit        # Produit = A1(2n−1−j)/(3−j)
      Produit -> A[i]                            # Ai = A1(2n−1)(2n−2).../3 /4...
    Somme + A[i]*Bernoulli[i] -> Somme           # Somme = Somme + AiBi
    Si i == n−1 alors −Somme -> Bernoulli[i]     # Bi = −Somme
```

qui publiera un article en français, *Notions sur la machine analytique de Charles Babbage*. Ada lit, traduit et annote cet article grâce aux échanges qu'elle continue d'avoir avec Babbage, puis elle publie ce texte. Il est trois fois plus long que l'article de Menabrea, car augmenté de sept notes. Ce sera pendant près d'un siècle le seul article de ce genre.

Les notes d'Ada sont consacrées à la programmation. En particulier, la dernière décrit un programme permettant de calculer les nombres de Bernoulli : ce programme utilise toutes les possibilités de la machine, comme les branchements conditionnels et les boucles. Il est bien plus complexe que ceux qui avaient été envisagés par Babbage, et reste considéré par beaucoup comme le premier vrai programme de l'histoire.

Les nombres de Bernoulli B_n sont des constantes que l'on rencontre lors du développement en polynômes des fonctions trigonométriques. Ils sont définis par le développement en série entière de la fonction définie qui au réel strictement positif x associe $x/(e^x - 1)$. Mais ce sont des formules de récurrence qui ont été à la base du programme d'Ada Lovelace. Ces dernières sont de la forme suivante :

$A_0B_0 + A_1B_1 + A_2B_2 + \ldots + A_nB_n = 0$, avec les coefficients A_i qui peuvent dépendre de n. La formule de récurrence la plus employée aujourd'hui est $\sum_{k=0}^{n} \binom{n+1}{k} B_k = 0$, mais en son temps Ada Lovelace utilisait l'équation suivante :

$$0 = -\frac{1}{2}\frac{2x-1}{2x+1} + B_1\frac{2x}{2!} + B_2\frac{(2x)(2x-1)(2x-2)}{4!} + \ldots + B_n\frac{(2x)(2x-1)\ldots(2x-2n+2)}{(2n)!}$$

où elle posait $x = n$.

On obtient alors en effet :

$$A_n = \frac{(2n)(2n-1)\ldots(2n-2n+2)}{(2n)!} = \frac{(2n)!}{(2n)!} = 1$$

et

$$A_{n+p} = \frac{(2n)(2n-1)\ldots(0)\ldots(-2p+2)}{(2n)!} = 0.$$

Ada utilise des variables $V_1, V_2\ldots$ recevant les valeurs. V_1 est initialisée avec 1, V_2 avec 2. Le programme qu'elle conçoit est une grande boucle itérant

sur n calculant les nombres de Bernoulli B_i de proche en proche.
Pour $n = 1, 0 = A_0 + B_1$
donc $B_1 = -A_0$.
Pour $n = 2, 0 = A_0 + B_1A_1 + B_2$
donc $B_2 = -A_0 - B_1A_1$.
Pour $n = 3, 0 = A_0 + B_1A_1 + B_2A_2 + B_3$
donc $B_3 = -A_0 - B_1A_1 - B_2A_2$.
V_3 est initialisée avec n. Une autre boucle de sommation est nécessaire pour ajouter à A_0 les A_iB_i. Le compteur de cette boucle est la variable V_{10}.
Encore faut-il évaluer les A_i ! Pour ce faire, elle commence par calculer

$$A_0 = -\frac{1}{2}\frac{2n-1}{2n+1}, \text{ puis } A_1 = \frac{2n}{2}.$$

Son évaluation du coefficient A_1 n'est pas optimale, mais elle se sert de la valeur $2n$ déjà stockée dans une variable lors de l'évaluation du coefficient A_0.
Pour évaluer les A_n suivants (où $n > 2$), elle réécrit l'expression de A_n comme suit :

$$A_n = \frac{2n}{2}\frac{2n-1}{3}\frac{2n-2}{4}\cdots\frac{2}{2n}$$
$$= A_1\frac{2n-1}{3}\frac{2n-2}{4}\cdots\frac{2}{2n}$$

Elle construit donc une nouvelle boucle, matérialisée dans son programme par une accolade. Le numérateur est initialisé avec $2n - 1$ et le dénominateur avec 3 ; à chaque itération, on divise le numérateur, diminué de 1, par le dénominateur, augmenté de 1, et on le multiplie avec le résultat. Ce morceau de programme est le suivant si la variable V_1 contient 1, V_7 contient 2, V_6 contient $2n$ et V_{11} contient A_1 :

$V_6 - V_1 \rightarrow V_6$
(V_6 contient désormais $2n - 1$),
$V_1 + V_7 \rightarrow V_7$
(V_7 contient désormais $2 + 1 = 3$),
$V_6 / V_7 \rightarrow V_8$
(V_8 contient désormais la nouvelle fraction),
$V_8 \times V_{11} \rightarrow V_{11}$
(V_{11} contient le nouveau produit).

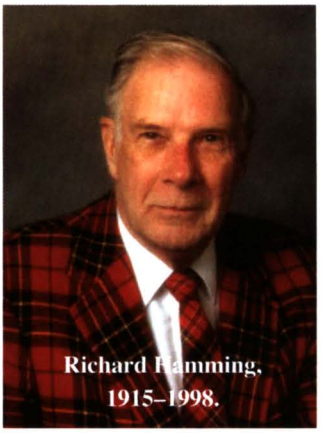
Les résultats des A_i sont conservés dans les variables V_{11}, V_{12} et V_{13} alors que les résultats des nombres de Bernoulli utilisent les variable V_{21}, V_{22}, V_{23}, V_{24}…
Ce programme est incroyablement moderne et abstrait (utilisation de variables, boucles, branchements). Ada avait bien compris l'idée de ce que nous appelons le *branchement conditionnel* (la possibilité qu'a un programme de sélectionner des instructions suivant une condition). Elle s'est intéressée à la notion de calculabilité, faisant la distinction entre ce qui est théoriquement possible de calculer de ce qui l'est en pratique. Enfin, elle avait parfaitement perçu l'intérêt de la mécanisation du calcul. Hélas, elle décédera à la fleur de l'âge d'un cancer et ses travaux seront oubliés pendant plus d'un siècle. Elle sera redécouverte lors de l'avènement des premiers ordinateurs ; un langage informatique portera même son nom.

J.-J. D.

Références

- *Lady Ada et le premier ordinateur*. Eugene Eric Kim et Betty Alexandra Toole, *Pour La Science* 261, juillet 1999.
- *Lady Augusta Ada King comtesse de Lovelace*. Bibliothèque Tangente 37, *Les algorithmes*, 2013.
- *Informatique : la préhistoire est anglaise*. *Tangente* 137, 2010.

Les messages qui se corrigent tout seuls


Richard Hamming, 1915–1998.

De nos jours, tout message, que ce soit un texte, une image ou une vidéo, est une longue suite de bits, c'est-à-dire de 0 et de 1. Il peut être envoyé par des canaux de communication divers : câbles, fibres optiques, ondes radio, *etc.* Quelle que soit la ligne de transmission utilisée, elle ne saurait être à l'abri d'erreurs. Pour remédier à ce défaut, l'idéal est que les messages erronés se corrigent d'eux-mêmes. La tâche semble impossible… et pourtant l'idée est simple : enrichir le message d'informations redondantes.

La première solution qui vient à l'esprit est la répétition, par exemple trois fois. Ainsi, 0 est codé en 000, et 1, en 111. À la réception, le message est découpé en blocs de trois bits. Les groupes 000 et 111 sont corrects et ne posent aucun problème. Pour les autres, on remplace le bit minoritaire par le bit opposé. Ainsi 100, 010 et 001 sont remplacés par 000, 011, 101 et 110, par 111. Inconvénient : la correction s'avère exacte si une seule erreur a été commise. Autrement dit, le code proposé ne peut corriger qu'une erreur tous les trois bits. De plus, il est assez lourd puisqu'il triple la longueur des messages.

Avant d'aller plus loin, notons qu'une notion importante se dégage, celle de distance linguistique entre deux messages, c'est-à-dire le nombre de bits à modifier pour passer de l'un à l'autre.

Les mathématiciens ont inventé de nombreux codes correcteurs d'erreurs utilisant des notions d'algèbre. Par exemple, un codage dû à Richard Hamming consiste à transformer chaque groupe de quatre bits (x_1, x_2, x_3, x_4) en un mot de huit bits qui s'écrit :
$(x_1, x_2, x_3, x_4, x_1 + x_2, x_3 + x_4, x_1 + x_3, x_2 + x_4)$.

Dans cette transformation, l'addition se fait suivant la table suivante :

+	0	1
0	0	1
1	1	0

Ainsi, le mot 0110, où $x_1 = 0$, $x_2 = 1$, $x_3 = 1$ et $x_4 = 0$, est transformé en 01101111. Le mot 0010 est transformé en 00100110. Alors que les deux mots 0110 et 0010 sont à une distance linguistique égale à 1, les deux mots images 01101111 et 00100110 sont à une distance égale à 3. Ce résultat est général : deux codes distincts génèrent deux images à distance au moins égale à 3. Le code de Hamming permet donc de corriger une erreur en étant plus économe que la répétition triple. On peut dresser une table de correction comme dans le cas précédent mais on peut également donner une formule de correction…

RÉFÉRENCES :

L'Univers des codes secrets de l'Antiquité à Internet, Hervé Lehning, Ixelles

De nombreux articles (voire même numéros entiers) de *Tangente*, de ses hors-séries et de *Tangente Sup* ont été consacrés à la cryptographie et aux codes correcteurs d'erreurs. Voir par exemple :

- *Tangente* 147, pages 42 à 50, 2012.
- *La cryptographie*. Bibliothèque Tangente 26, réédité en 2013.
- *Transfert et échange*. *Tangente SUP* 70–71, 2013.

par David Delaunay

Alonzo Church, Alan Turing et la calculabilité

S'inspirant l'un et l'autre des travaux fondateurs du logicien Kurt Gödel, Alonzo Church et Alan Turing dégagent deux notions clés : le λ-calcul et la machine de Turing. Leur but est de préciser la notion de calculabilité. Ils ont en fait défini un seul et même concept !

À la fin du XIX^{e} siècle, les fondements des mathématiques sont remis en cause par l'introduction de paradoxes logiques et ensemblistes analogues au paradoxe du menteur affirmant *« Je suis un menteur »*. Cette crise a participé à l'essor de la logique mathématique à travers l'axiomatisation de la théorie des ensembles, le calcul des prédicats ou la théorie des modèles. En 1930, le logicien Kurt Gödel énonce un premier théorème de complétude affirmant que, si un énoncé est vérifié dans tous les modèles d'une théorie, alors cet énoncé est démontrable. Peut-on alors en « calculer une démonstration » ? Que signifie être calculable ? Avec Alonzo Church et Alan Turing, on obtient deux approches qui vont converger vers une même définition.

Alonzo Church écrivait initialement ^x au lieu de λx, puis le symbole s'est transformé avec l'usage…

Un nouveau langage

Le mathématicien et logicien américain Alonzo Church (1903–1995) conduit des travaux parallèles à ceux de Gödel et s'intéresse en particulier à définir ce qui est « algorithmiquement calculable ». Il introduit pour cela ce que l'on appelle de nos jours le λ-calcul (lire *lambda calcul*). Il s'agit d'un langage servant à décrire et à composer des fonctions afin de déterminer ce que celles-ci peuvent calculer.

Dans ce langage, toutes les lettres servent à désigner des fonctions. Si x et y désignent deux fonctions, écrire xy signifie composer celles-ci (ce que l'on écrit aujourd'hui $x \circ y$). Si e est un mot de ce langage où pourrait apparaître la lettre x, Alonzo Church écrit $\lambda x.e$ pour désigner la fonction qui à x associe e (ce que l'on écrirait $x \mapsto e$). Ainsi, $\mathrm{I} = \lambda x.x$ désigne la fonction identité, tandis que $\lambda x.y$ désigne la fonction constante égale à y. Plus généralement, on peut aussi écrire $\lambda xy.e$, pour définir la fonc-

tion qui à x et y associe e. Par exemple, $\lambda xy.xyx$ se comprend comme une fonction qui envoie x et y sur le résultat de la composition $x \circ y \circ x$. La fonction $K = \lambda xy.x$ se comprend comme celle associant à un couple (x, y) son premier élément (projection sur la première composante).

En λ-calcul, une fonction définie par $\lambda x.e$ est composée par y en écrivant $(\lambda x.e)y$. Cette fonction est alors « équivalente » à la fonction obtenue en remplaçant chaque occurrence libre de x dans e par y. On définit ainsi une règle de transformation des expressions du λ-calcul donnant par exemple

$(\lambda x.xx)y \longrightarrow yy$, $(\lambda xy.x)z \longrightarrow \lambda y.z$,

$(\lambda xy.yx)ab \longrightarrow ba$

ou encore la succession

$(\lambda x.xx)(\lambda y.y) \longrightarrow (\lambda y.y)(\lambda y.y) \longrightarrow \lambda y.y$.

Ces calculs, *a priori* simplificateurs, peuvent néanmoins boucler à l'infini dans certaines situations. C'est le cas si l'on considère $\Omega = \Delta\Delta$ avec $\Delta = \lambda x.xx$, où l'on obtient :

$\Omega = (\lambda x.xx)\Delta \longrightarrow \Delta\Delta = \Omega \longrightarrow \Omega \longrightarrow \ldots$

Parfois même, il n'y a simplification qu'en fonction de l'organisation du calcul :

$KI\Omega = (\lambda x.x)I\Omega \longrightarrow I$ alors que

$KI\Omega = KI(\lambda x.xx)\Delta \longrightarrow KI\Omega \longrightarrow KI\Omega \longrightarrow \ldots$

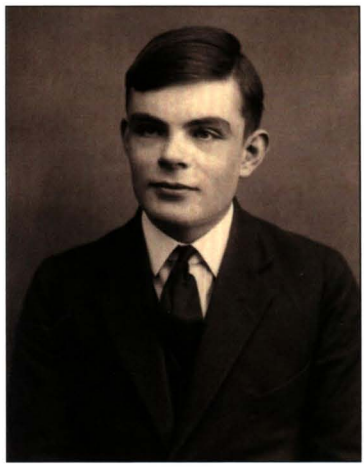
En λ-calcul, les lettres ne désignent que des fonctions et nullement des objets mathématiques : les expressions écrites en λ-calcul ne comportent que les lettres du vocabulaire servant à nommer les fonctions, le symbole λ et les symboles utiles au parenthèsage. Alonzo Church parvient cependant à reconstruire dans ce langage les objets du monde mathématique, à commencer par les entiers naturels ou les valeurs booléennes. Il parvient aussi à définir les constructeurs de la programmation informatique que sont les instructions conditionnées, les itérateurs et les appels récursifs. À ce titre, le λ-calcul peut être considéré comme l'un des premiers langages permettant de coder les algorithmes. Ainsi, Alonzo Church accède aux fonctions λ-calculables, c'est-à-dire à celles pouvant être obtenues par λ-calcul.

Alonzo Church.

Une machine à calculer universelle

Parallèlement, le mathématicien et logicien britannique Alan Turing complète lui aussi les travaux de Gödel afin de déterminer ce qui est « mécaniquement calculable ». Il définit pour cela une « machine à calculer universelle ». Celle-ci est particulièrement rudimentaire, mais ses capacités théoriques égalent, et même surpassent, celles de n'importe quel supercalculateur moderne !

Une machine de Turing est constituée d'un ruban de longueur infinie (équivalent d'une mémoire) sur lequel évolue une tête de lecture et d'écriture. Une machine de Turing est aussi constituée d'un processeur pouvant prendre un nombre fini d'états et commandant le comportement de la tête de lecture. La programmation d'une machine de Turing se fait préalablement à son fonctionnement. Elle



Alan Turing.

consiste à définir, en fonction de l'état du processeur et de la valeur lue sur le ruban, des actions d'écriture sur le ruban, de déplacement de la tête de lecture, de changement d'état du processeur.
À partir d'une information initialement écrite sur un nombre fini de cases du ruban, la machine de Turing opère selon le programme prédéterminé et, si celui-ci conduit le processeur à un état qualifié de *final*, on dit que la machine de Turing s'est *arrêtée*. L'utilisateur lit alors le contenu du ruban, qui se comprend comme étant la réponse calculée par la machine de Turing à partir du ruban d'entrée. Au final, une machine de Turing se comprend comme une fonction mécanique transformant l'état du ruban en un autre. Alan Turing a ainsi défini ce que pouvait être un ordinateur ayant été programmé. Il participera d'ailleurs activement à la création d'un des premiers ordinateurs en 1949.

Bien que ce modèle paraisse élémentaire (et c'est ce qui en facilite l'étude), il suffit pour mettre en place des démarches algorithmiques complexes. Alan Turing peut ainsi étudier ce que sont les fonctions T-calculables, c'est-à-dire les fonctions dont les valeurs sont fournies par l'exécution d'une machine de Turing.

La « λ-calculalibilité » et la « T-calculalibilité » sont deux approches *a priori* différentes, car l'une est plutôt logicielle alors que l'autre est plus matérielle. Elles visent cependant toutes deux à définir le plus généralement possible ce que pouvait être une *fonction calculable*. En 1936, Alonzo Church démontre que ces deux notions déterminent exactement les mêmes fonctions calculables ! Il affirme même qu'il n'est sans doute pas possible de proposer un ensemble plus large de fonctions calculables. Cette affirmation (un peu vague) est appelée la *thèse de Church*. Ce n'est pas un théorème, car la notion de fonction calculable n'est qu'intuitive. C'est plutôt un constat : on ne parvient pas à définir une classe plus large de fonctions dont les valeurs pourraient être calculées par un procédé de nature algorithmique.

Après ses théorèmes de complétude, Kurt Gödel a énoncé des théorèmes d'incomplétudes. Schématiquement, l'un d'eux affirme que, dans toute théorie capable de définir l'arithmétique des entiers, il existe un énoncé qui ne peut (ni lui, ni sa négation) être démontré. Cela entraîne qu'on ne peut définir d'algorithme prenant la décision de savoir si un énoncé arithmétique est, ou non, vérifié : on dit qu'il s'agit d'un problème *indécidable*. Parallèlement, Alonzo Church démontre qu'il est indécidable de savoir si deux phrases du λ-calcul sont équivalentes. Alan Turing établit quant à lui qu'il n'est pas non plus décidable de savoir si le fonctionnement d'une machine de Turing donnée va ou non s'arrêter.
Ces études ont fortement participé aux progrès des mathématiques dans le domaine de la logique et ont développé les outils contribuant à la formalisation de l'informatique moderne.

D. D.

L'avènement d'un nouveau langage

Vocabulaire informatique, de la norme à l'usage **Maria Centrella, Hermann, 214 pages, 2013, 23 euros.**

Le livre de Maria Centrella est beaucoup plus qu'un simple dictionnaire ou thésaurus, car son auteur y étudie les fondements linguistiques de la révolution culturelle que constitue l'informatique. Dans la liste des termes reconnus de la Commission générale de terminologie, et dont l'usage est normatif, c'est-à-dire obligatoire dans les textes officiels, Maria Centrella extrait 362 entrées dont elle dissèque les origines (anglaises pour 64 % d'entre elles, latines ou anglo-latines), et les calques structuraux (*base de données*), sémantiques (**fenêtre, passerelle**), syntagmatiques (cheval de Troie, liste de diffusion), transpositionnels (*internaute, mémoire vive, page d'accueil*) qui engendrent parfois des néologismes. Les commentaires métalinguistiques sont clairs et pertinents comme dans l'étude des acronymes et sigles anglais (le choix de l'usage de www plutôt que « toile d'araignée mondiale »).

Une deuxième partie est consacrée à l'implantation des termes officiels dans l'usage réel de la langue avéré par les dictionnaires courants.

Le travail, fin et rigoureux, extrait de la thèse de doctorat de l'auteur, est étayé par de nombreux tableaux de fréquences d'apparition des termes dans des ouvrages spécialisés ou de langue générale avec 3 200 textes et 1 250 000 mots répertoriés et analysés. Il montre clairement que si la norme conduit à l'usage, l'usage peut conduire à la norme de fait, tel un droit coutumier. L'exemple d'échecs de termes comme *mél* (créé sur la topologie de *tél.*) ou *courriel* au profit de *e-mail, email* et maintenant *mail*, voire *courrier électronique* est caractéristique.

A.Z.

Quand l'informatique revisite les classiques

La reliure spirale donne à ce livre un petit côté cahier d'écolier, mais d'écolier studieux car tout est ici traité avec le plus grand sérieux. On part de problèmes ludiques classés en trois catégories : « Avec des nombres », « Avec des lettres », « Avec des images ». Tous bien connus des amateurs de mathématiques récréatives, comme le problème de Josèphe, la suite de Collatz, les nombres parfaits, les techniques de cryptographie et de stéganographie, la spirale d'Ulam ou l'ensemble de Julia, ils sont plus traités par l'informatique que par le raisonnement mathématique. Le livre en devient un véritable manuel d'informatique, avec exposé des notions, programmes (en langage Ruby), exercices et solutions. Il n'est pas nécessaire au départ de savoir programmer pour étudier les problèmes proposés, l'information nécessaire étant donnée au fur et à mesure de la lecture. Le néophyte devra donc le lire dans l'ordre.

Sous la jolie couverture elle-même, on trouve ainsi plus de mathématiques et surtout plus d'informatique que de divertissements. Les problèmes évoqués donnent tous lieu à de grands développements informatiques que seuls les amateurs éclairés apprécieront. Enfin, au livre est associé un site, www.divmath.fr, offrant de nombreuses références culturelles et historiques.

É. B.

Divertissements mathématiques et informatiques. **Laurent Signac, H&K, 176 pages, 2011, 14,90 euros.**

par François Lavallou

Algèbre de Boole

Fruit d'une longue et lente algébrisation de la logique, l'algèbre de Boole est utilisée pour la première fois hors du champ mathématique par Shannon, père fondateur de la théorie de l'information. Elle a depuis de nombreuses applications en électronique et informatique.

Les philosophes grecs furent les premiers à étudier la logique, qui régente, souvent inconsciemment, nos processus mentaux. Dès le siècle de Périclès, la structure idéale d'un texte mathématique est établie et la notion de démonstration chez Euclide, Archimède ou Apollonius est totalement moderne. Deux logiques complémentaires se développent avec l'école péripatéticienne d'Aristote et l'école du portique de Chrysippe. Si la logique aristotélicienne, ou scolastique, fut longtemps la référence des logiciens, celle des stoïciens, qui tomba peu à peu dans l'oubli, peut être considérée comme une première version du calcul moderne des propositions.

E	Classes ($\alpha, \beta, \gamma, \ldots$)	Propositions ($p, q, r, \ldots$)
1	Univers (1)	Tautologie (V)
0	Vide (0)	Contradiction (F)
+	Union ($\cup$)	Disjonction ($\vee$)
×	Intersection ($\cap$)	Conjonction ($\wedge$)
–	Complément (–)	Négation ($\neg$)
$\subset$	Inclusion de classe ($\subset$)	Implication matérielle ($\supset$)
=	Identité entre classe (=)	Equivalence matérielle ($\equiv$)

Interprétations de l'algèbre de Boole

Des péripatéticiens à Boole

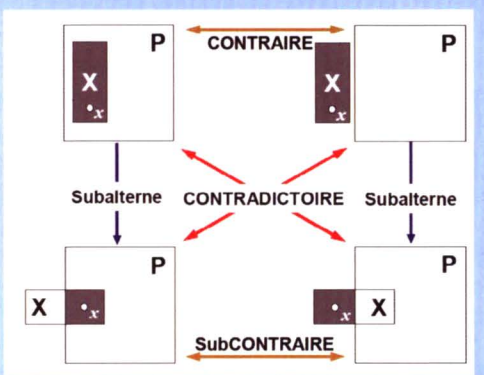
En quête d'un langage universel commun à la logique et à l'algèbre, Leibniz recherche, sans succès, un « alphabet des pensées humaines » qui permette de transcrire symboliquement tout raisonnement déductif en algorithme. Gergonne tente bien d'utiliser la combinatoire pour retrouver la syllogistique des anciens mais, fondamentalement, la logique reste toujours une branche de la philosophie n'appartenant pas au corpus mathématique.

C'est à George Boole (1815–1864), un mathématicien autodidacte, que revient le mérite d'affirmer que la logique doit être rattachée aux mathématiques et non à la philosophie. Il réunit les logiques aristotélicienne et stoïcienne en un unique système algébrique, mais qui impose de choisir entre une interprétation ensembliste ou propositionnelle (voir tableau). Malgré ses défauts, cette mathématisation de la logique est une révolution, plus qu'une évolution, et aura des répercussions sur les fondements même des mathématiques.

Syllogismes et carré logique

Les syllogismes sont des associations de trois propositions prises parmi quatre types possibles, notés par les quatre premières voyelles : A, E, I, O, qui sont aussi celles d'Aristote ! Ces propositions références sont les suivantes, avec les notations ensembliste et booléenne, où $x \in X$ et $y \in P$:

- **Proposition A, universelle affirmative : Tout x est P ; $\forall x\ x \subset P$; $x.(1-y) = 0$**
- **Proposition E, universelle négative : Tout x est non-P ; $\forall x\ x \not\subset P$; $x.y = 0$**
- **Proposition I, particulière affirmative : Quelque x est P ; $\exists x\ x \subset P$; $x.y \neq 0$**
- **Proposition O, particulière négative : Quelque x est non-P ; $\exists x\ x \not\subset P$; $x.(1-y) \neq 0$**



Le carré logique illustre les oppositions logiques entre ces propositions.

Des propositions peuvent s'opposer par leur quantité (universelle ou particulière) et/ou leur qualité (affirmative ou négative). Chaque proposition de référence est ainsi en opposition avec trois autres selon le schéma dit du carré logique. Ces relations sont dénommées :

- **contradictoires, pour opposition en quantité et qualité,**
- **contraires, pour des propositions universelles (∀) opposées en qualité,**
- **subcontraires, pour des propositions particulières (∃) opposées en qualité,**
- **subalternes, pour opposition en quantité.**

Les syllogismes sont composés de deux propositions, appelées prémisses, ayant en commun un moyen terme, et d'une proposition conclusive sans le moyen terme. Pour des prémisses données et supposées vraies, le syllogisme valide la véracité de la conclusion. Il y a $4^3 = 64$ propositions pour un syllogisme, $2 \times 2 = 4$ positions pour le moyen terme, donc en tout $4 \times 64 = 256$ structures potentielles de syllogismes. Mais seuls vingt-quatre sont valides. Le célèbre syllogisme :

Tous les hommes sont mortels
Or Socrate est un homme
Donc Socrate est mortel

est constitué de trois propositions de type A, avec « homme » pour moyen terme. Un tel syllogisme est appelé « barbara », car les voyelles correspondent aux types des propositions. De même, le syllogisme :

Tous les lecteurs de Tangente sont sympathiques
Certains mathématiciens ne sont pas sympathiques
Donc certains mathématiciens ne lisent pas Tangente

est de type « baroco ».

La contribution de Boole à la logique symbolique se résume dans les propositions suivantes :

- la logique peut être remplacée par un calcul qui « ne dépend pas de l'interprétation des symboles employés mais seulement des lois qui régissent la manière dont ces symboles sont combinés »,
- l'algèbre appliquée aux nombres peut être étendue aux classes (ensembles),
- les quatre propositions fondamentales d'Aristote (voir encadré) peuvent être exprimées par des équations algébriques,

- la validité des syllogismes peut être établie par un calcul algébrique sur ces équations.

Le système de Boole repose sur l'utilisation de lettres pour désigner les objets et de symboles pour traduire les relations effectuées par la pensée entre ces objets. La conjonction des termes, qui correspond au « et », est symbolisée par la multiplication, et la disjonction, pour le « ou » exclusif, par l'addition.

Les syllogismes en équations

L'algèbre de Boole repose sur des propriétés fondamentales, dont l'associativité, la commutativité, $x.y = y.x$, la distributivité, $x.(y + z) = (x.y) + (x.z)$ et une loi d'idempotence $x^2 = x$, spécifique à cette algèbre.

En notant 1 la classe universelle, ou univers du discours, et 0 la classe vide, alors 1 est un élément neutre, $1.x = x$ et 0 est un absorbant, $0.x = 0$.

L'algèbre de Boole vérifie le principe de dualité : tout théorème demeure vrai quand les symboles « . » et « + », ainsi que les éléments neutres 0 et 1 sont échangés dans l'énoncé. Les résultats multiplicatifs précédents sont ainsi valables pour l'addition et l'idempotence s'écrit : $x + x = x$.

En notant $\overline{x}$ la classe complémentaire de x, le principe de contradiction des Métaphysiciens, qui affirme impossible pour un objet de posséder et ne pas posséder une qualité, s'écrit $x.\overline{x} = 0$. Avec le principe de dualité, nous obtenons l'expression $\overline{x} + x = 1$, qui n'est autre que le principe du tiers exclu : un objet appartient à la classe x ou à son complémentaire. Boole en tirera son expression du complémentaire additif : $\overline{x} = 1 - x$, qui permet d'identifier principe de contradiction et loi d'idempotence : $x^2 = x \Leftrightarrow x.(1 - x) = x.\overline{x} = 0$.

Par le « principe de transfert », qui « relie » logique et algèbre formelle, on peut établir algébriquement le théorème d'absorption : $a + a.b = a.(1 + b) = a$.

Et avec les théorèmes duaux de Morgan, $\overline{a + b} = \overline{a}.\overline{b}$ et $\overline{a.b} = \overline{a} + \overline{b}$, vous avez tous les éléments pour simplifier une expression telle que $E = \overline{a}.b + a.c + b.c$ en $E = \overline{a}.b + a.c$.

Avec son algèbre, Boole peut mettre en équations les quatre propositions références d'Aristote. Le syllogisme correspond alors à la réduction de deux équations (les prémisses) à une seule (la conclusion) en éliminant la variable moyen terme.

L'algèbre de Boole a trouvé naturellement un domaine d'application en sciences informatiques dont la logique repose sur un système binaire.

Ironie de l'histoire, la logique pour laquelle s'est passionné George Boole a causé sa perte ! Ayant pris froid, il meurt d'une pneumonie après que sa femme Mary, croyant au principe d'analogie (théorie à la logique discutable), l'ait aspergé d'eau pour le guérir.

F. L.

Référence

• Paul Gochet, Pascal Gribomont, *Logique, méthodes pour l'informatique fondamentale*, Hermès.

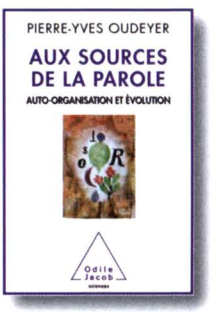
La tour de Babel reconstituée par l'informatique

D'Arcy Thompson a été le premier à utiliser les mathématiques pour expliquer les sciences du vivant. La sélection naturelle, la forme des coquillages et autres organes pouvaient dès lors être mis en équation. Avec les ordinateurs et les algorithmes de simulation, le climat, la morphogenèse, l'autoreproduction et l'auto-organisation des structures furent mis en évidence par Fermi, Lorenz, Turing puis Von Neumann. C'est cet héritage que revendique Pierre-Yves Oudeyer qui, partant de la robotique, une nouvelle science n'existant que par l'informatique, cherche à modéliser les représentations cognitives du cerveau, et en particulier la parole et la création du langage par une communauté vivante autarcique. Le livre aborde déjà l'étude de la structure de la parole et des langages ainsi que les principes de l'auto-organisation. À partir des différentes théories sur les origines de la parole, il justifie le modèle choisi pour l'auto-organisation de systèmes de vocalisation qui vont conduire ses robots à faire le lien entre des sons et des objets ou des réactions motrices jusqu'à créer un « vrai » langage et de sa syntaxe.

Outre la reconstitution d'une société humaine, l'expérimentation permet, en modifiant les algorithmes, de se rapprocher de la réalité, et par là-même de faire des progrès sur la connaissance de la construction du monde dans lequel on vit.

G.C.

Aux sources de la parole, auto-organisation et évolution, **Pierre-Yves Oudeyer, Éditions Odile Jacob 2013, 230 pages, 24,90 euros.**



TIC : repérez les mythes

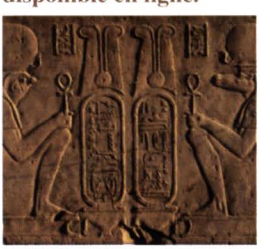
Les mythes et légendes réfèrent à la religion et à l'Antiquité ; comment peut-on les associer à des questions ultra modernes, comme les TIC (technologies de l'information et de la communication) ? Gérard Peliks et son équipe d'auteurs du forum Atena nous montrent pourtant que ce domaine, *a priori* éloigné de toute religion, fonctionne bien sur un mode mythique et légendaire. L'un des mythes principaux concerne la compétence informatique supposée de ces personnes « nées avec un ordinateur dans les mains », qui seraient donc toutes des génies de l'informatique. Bien sûr, elles ont acquis des savoir-faire, mais ceux-ci sont limités, peu explicités ou explicitables et ne prêtent pas à la conceptualisation. Les entreprises du secteur des TIC ont d'ailleurs bien du mal à recruter… Un véritable enseignement de l'informatique pour tous serait nécessaire pour cela.

Autre mythe parmi d'autres, dans un domaine très différent : le système des adresses IP (*Internet protocol*), qui permettrait d'identifier les utilisateurs… mais qui est loin de le faire si ceux-ci savent la modifier, en la localisant à l'étranger par exemple, ce que certains logiciels gratuits réalisent à volonté. De même, le calcul intensif n'est plus le domaine réservé des supercalculateurs ; les « bonnes » méthodes de chiffrement ne sont pas fondées sur des algorithmes tenus secrets mais sur des algorithmes connus de tous les spécialistes, dont on change souvent les clefs ; *etc.*

L'ouvrage intéressera toutes les personnes concernées par l'informatique, Internet et les TIC. Écrit dans une langue simple et compréhensible, très loin de celle qu'utilisent ceux qui colportent les mythes et légendes, il illustre à merveille cette citation de Boileau : *« Ce qui se conçoit bien s'énonce clairement. »*

H.L.

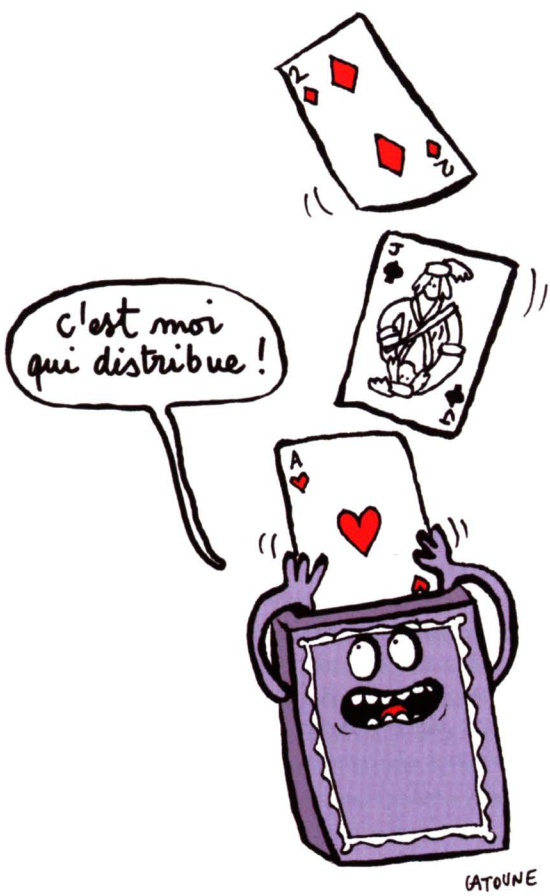
Mythes et légendes des TIC. **Collectif, sous la direction de Gérard Peliks, Forum Atena, 296 pages, 2011, disponible en ligne.**



par Hervé Lehning

Langages et récursivité

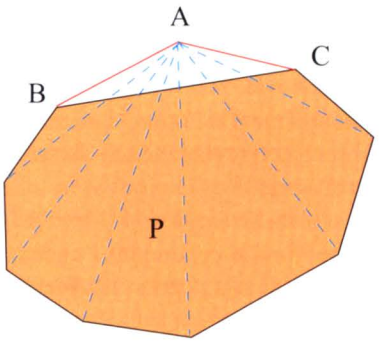
La notion d'autoréférence fascine les amateurs de paradoxes et de jeux mathématiques. En informatique, c'est la récursivité qui joue ce rôle. Elle consiste à appeler au sein d'un programme qui calcule la fonction *f*... la fonction *f* elle-même ! Il n'y a pourtant rien de plus sûr qu'une telle démarche.



Les premiers langages de programmation comme COBOL pour la gestion ou FORTRAN pour le calcul scientifique se passaient de récursivité, ils avaient bien tort ! Si aujourd'hui, sans doute, tous les langages sont récursifs, l'une des raisons est probablement la sûreté, les programmes récursifs étant pratiquement les seuls dont on puisse démontrer qu'ils font bien ce qu'ils sont censés faire... Dans les dictionnaires, la définition la plus courante du mot *récursif* est sibylline : *Un programme informatique est dit récursif s'il demande sa propre exécution au cours de son déroulement*. Sans aucun exemple, cette phrase est bien difficile à comprendre...

Récurrence et induction

Pour comprendre cela, il est nécessaire d'expliquer effectivement ce qu'est la récursivité... et pour cela la récurrence, la notion mathématique dont elle est issue.



Un polygone convexe à *n* sommets est obtenu d'un polygone convexe P à *n* – 1 sommets en ajoutant un point A entre deux sommets B et C (côtés en rouge). Les diagonales sont alors de trois types : celles de P, BC et celles passant par A (en bleu).

Voyons un exemple où ce principe est à l'œuvre : combien de diagonales un polygone convexe à n côtés possède-t-il ? Tout d'abord, nous lui donnons un nom : v_n. Nommer est prendre un pouvoir sur ce que l'on nomme. Le raisonnement s'opère en liant le cas d'un polygone à n côtés à celui d'un polygone à $n - 1$ côtés. On considère donc un polygone P à $n - 1$ côtés et on y ajoute un point A entre deux sommets B et C.
Les diagonales se scindent en trois groupes. Le premier est formé des diagonales du polygone P, au nombre de v_{n-1} par hypothèse, le second du côté BC de P et le troisième des droites joignant A aux points de P autres que B et C, au nombre de $n - 3$.
Ainsi : $v_n = v_{n-1} + 1 + (n - 3)$

Une telle relation est dite *de récurrence* car elle lie v_n à v_{n-1}. Si on connaît v_{n-1}, on en déduit v_n. La connaissance d'un terme permet de calculer, les suivants. Comme un triangle n'a aucune diagonale, $v_3 = 0$ ce qui donne les suivants de proche. On peut même en déduire une formule donnant v_n. On trouve : $v_n = n(n-3)/2$. Comment la trouver ? Le détail ne sera pas donné ici, mais il faut savoir que les méthodes sont diverses. L'une d'entre elles, qui n'est pas la plus négligeable, est l'*induction*.

On émet une hypothèse par intuition, on la vérifie expérimentalement pour les premières valeurs, et, si aucun contre-exemple ne vient la démentir, on la démontre.
La *démonstration par récurrence* (ce n'est pas par hasard qu'elle est appelée *induction* par les Anglo-Saxons) est la preuve de cette formule *a posteriori*. Sa démarche :
• Elle est vérifiée pour $n = 3$.

La récursivité en mémoire

La procédure ci-dessous décrit en détail comment la fonction Factorielle fonctionne pour $n = 3$:
- Comme 3 n'est pas égal à 0, l'appel de Factorielle (3) provoque l'affectation de 3 × Factorielle (2), qui demande le calcul de Factorielle (2), et on recommence.
- Comme 2 n'est pas égal à 0, l'appel de Factorielle (2) provoque l'affectation de 2 × Factorielle (1), qui demande le calcul de Factorielle (1), et on recommence.
- Comme 1 n'est pas égal à 0, l'appel de Factorielle (1) provoque l'affectation de 1 × Factorielle (0), qui demande le calcul de Factorielle (0), et on recommence.
- Comme 0 est égal à 0, l'appel de Factorielle (0) provoque l'affectation de 1. Les calculs envisagés et laissés de côté peuvent alors être effectués. On obtient successivement :
  Factorielle (1) = 1 × Factorielle (0) = 1 × 1 = 1,
  Factorielle (2) = 2 × Factorielle (1) = 2 × 1 = 2,
  Factorielle (3) = 3 × Factorielle (2) = 3 × 2 = 6.

Chaque procédure récursive peut se détailler ainsi. Elle occupe une place en mémoire importante car tous les appels récursifs doivent être stockés dans un sens, avant d'être exécutés dans l'autre. On comprend que cette description est inutile dans la pratique, car l'un des intérêts primordiaux des fonctions récursives est d'être facile à prouver par récurrence.

• Admettons qu'elle le soit pour $n - 1$ (*hypothèse de récurrence*) :
$v_{n-1} = (n - 1)(n - 4)/2$.
• Alors, d'après la relation de récurrence :

$$\begin{aligned} v_n &= v_{n-1} + n - 2 \\ &= (n - 1)(n - 4)/2 + n - 2 \\ &= (n^2 - 5n + 4 + 2n - 4)/2 \\ &= (n^2 - 3n)/2 = n(n - 3)/2 \end{aligned}$$

La formule est donc vraie pour n.
• De proche en proche (on dit encore *par récurrence*), on en déduit qu'elle est vraie pour tout $n \geq 3$.
Il s'agit de l'archétype du raisonnement par récurrence. Comme on va le voir, il s'applique à l'identique pour prouver que les *fonctions récursives* remplissent bien leur office.
Il est donc à la base de la *récursivité*.

Description d'une fonction récursive

La démarche précédente permet de créer des algorithmes avant de les prouver. L'important est de rester au cœur du principe : si on sait passer de l'étape n à l'étape $n + 1$ et que l'on sait *démarrer*, on peut traiter toutes les étapes.
Quand on l'applique en informatique, on ne parle plus de récurrence mais de *récursivité*.
On peut ainsi définir des *fonctions récursives* dans pratiquement tous les langages modernes.
L'exemple le plus souvent donné est celui de la fonction factorielle dont voici la description du calcul :

Fonction Factorielle, argument *n* : nombre
Si *n* = 0 alors Factorielle := 1
sinon Factorielle := *n* × Factorielle (*n* – 1)

Comme dans un grand nombre de langages, on a fait précéder la description de cette fonction par une partie déclarant son nom (Factorielle) et son (ou ses) argument : n qui est un nombre. Le corps de la fonction décrit comment elle est définie. Cette définition peut sembler étonnante, car la factorielle y est définie à partir d'elle-même. On la comprend mieux en pensant à une délégation de tâche. Vouloir suivre son exécution est fastidieux et, de plus, ne sert à rien (l'encadré ci-contre simule cependant l'exécution complète de la procédure). L'intérêt est de pouvoir prouver que la fonction Factorielle remplit bien son rôle. Cela se fait par récurrence sur son argument n. On remarque que l'écriture de la fonction est elle-même la preuve qu'elle donne bien le bon résultat. En effet, si $n = 0$, le résultat est bien 1. Admettons que le résultat soit bien $(n - 1)$! pour $n - 1$, la relation Factorielle := $n \times$ Factorielle $(n - 1)$ donne $n\ (n - 1)\ ! = n\ !$ pour n, ce qui est le bon résultat. On a ainsi montré par récurrence que la fonction Factorielle renvoie donc toujours le bon résultat.

Le tri par récursivité

Les fonctions récursives sont particulièrement intéressantes si elles s'appliquent à des *structures récursives* comme les listes ou les arbres. Voici la définition, *a priori* surréaliste, de la structure de liste :

Une liste d'objets est soit la liste vide, soit un objet (la tête) plus une liste (la queue). Une liste de nombres entiers peut donc être : {10, 5, 8, 2}. Dans ce cas, sa tête est le nombre 10 et sa queue, la liste {5, 8, 2}. Pour ce qui suit, on notera $t :: q$ la liste de tête t et de queue q. Un exemple de fonction récursive sur les listes va déboucher sur un tri. Comment trier une liste par ordre croissant ? Comme on va le voir, il suffit de savoir insérer un nombre dans une liste déjà triée. La fonction suivante, notée Insère, sera utilisée pour ce faire :

Fonction Insère, arguments n : nombre, l : liste

Si la liste l est vide, Insère :=$\{n\}$

Si la liste l n'est pas vide, on la décompose en tête t et queue q

Si $n < t$ alors Insère := n :: l sinon Insère := t :: Insère (n, q)

L'algorithme est simple,
On peut démontrer que cette fonction réalise bien une insertion d'un nombre n dans une liste triée l par récurrence sur la longueur de l.

• Si cette longueur est nulle, la liste est vide, la première ligne du corps de la fonction s'applique, le résultat est donc correct.

• Admettons que le résultat soit correct pour une liste de longueur p et considérons une liste de longueur $p + 1$. La liste se décompose en tête t et queue q. Si $n < t$, il est mis en première position, sinon il est inséré dans la queue.

Cette queue est de longueur p donc Insère (n, q) donne le bon résultat... on en déduit que Insère (n, l) aussi. Ainsi, par récurrence, la fonction renvoie bien toujours le résultat attendu. L'intérêt de la récursivité est là : la facilité et la clarté de la programmation, la brièveté du programme... et surtout la possibilité de prouver que les fonctions fournissent bien les résultats attendus !

À partir de cette fonction d'insertion, il est facile de créer une fonction de tri :

Fonction Tri, argument : l : liste

Si la liste l est vide alors Tri := l

Si la liste l n'est pas vide, on la décompose en tête t et queue q alors :

Tri := Insère (t, Tri (q))

De même que pour la fonction d'insertion, cette fonction se prouve par récurrence sur la longueur de la liste l :

– Si cette longueur est nulle, la liste est vide, la première ligne du corps de la fonction s'applique, le résultat est donc correct.

– Admettons que le résultat soit correct pour une liste de longueur p et considérons une liste de longueur $p + 1$. La liste se décompose en tête t et queue q. Cette queue est de longueur p donc Tri (q) donne le bon résultat. Comme Insère donne également le bon résultat... on en déduit que Tri (l) aussi. On conclut à nouveau par récurrence.

Ces quelques exemples justifient l'affirmation : avec la récursivité, programmer, c'est prouver.

H. L.

par Jill-Jênn Vie

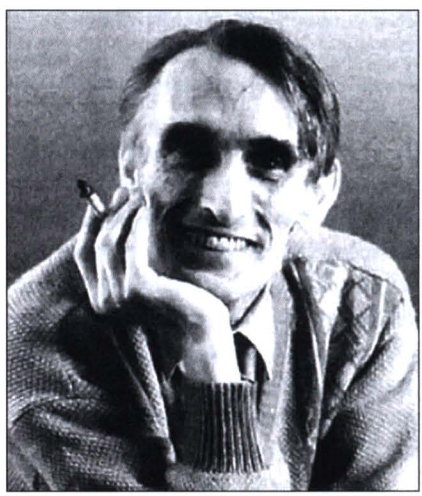
Langages rationnels et automates finis

La théorie des langages formels est une modélisation du langage naturel. Fondamentale pour décrire et analyser les langages de programmation et la calculabilité, elle a de nombreuses applications en linguistique : correction orthographique, reconnaissance vocale, traduction.

Parmi les langages formels se trouve la sous-classe remarquable des *langages rationnels*, indissociables des automates finis ou des machines abstraites (à la base des fameuses machines de Turing). Les automates trouvent eux-mêmes des applications en informatique, telles que la modélisation de processus, l'analyse lexicale effectuée par les compilateurs et la recherche de motifs (par exemple la recherche d'occurrences d'un mot dans un texte, ou de détection de séquences de bases azotées dans l'ADN).

Des lettres, des mots... un langage !

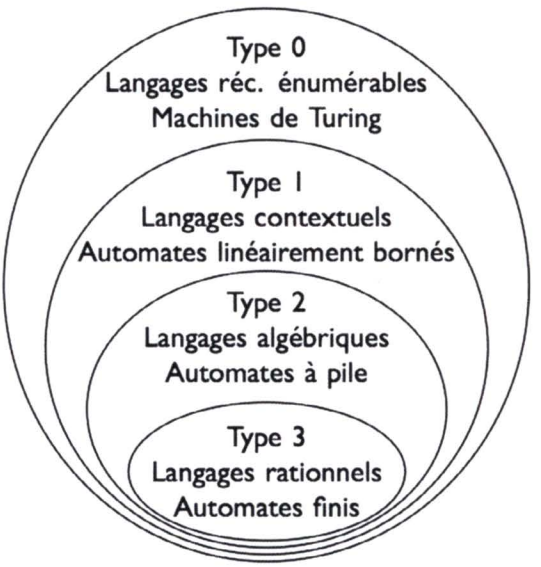
Un *langage* est défini comme un ensemble de *mots*, eux-mêmes suites d'éléments d'un *alphabet*, ensemble de *lettres* souvent supposé fini. L'ensemble des mots finis sur l'alphabet A est noté A*. Sur l'alphabet $\{a, b, n\}$, les ensembles $\{baba, nabab, banana\}$ et $\{a, aa, aaa\ldots\}$ sont des langages (le premier est fini, le second est infini). De même, les séquences de bases azotées sont des mots sur l'alphabet {A, T, C, G}.



Marcel-Paul Schützenberger (1920–1996), fondateur de *la combinatoire des mots* (1983).

On introduit ensuite la *concaténation*, opération (naturelle) qui à deux mots $u = u_1u_2\ldots u_m$ et $v = v_1v_2\ldots v_n$ associe le mot $uv = u_1u_2\ldots u_mv_1v_2\ldots v_n$ obtenu en « ajoutant » ou « collant » v après u. La notation $(ab)^3 = ababab$ est utilisée pour simplifier les notations. Enfin, on note

ε le *mot vide* (qui ne contient aucune lettre), élément neutre pour la concaténation. Plusieurs opérations peuvent être effectuées sur les langages. Si L et L' sont deux langages, on peut définir L∪L' l'*union* (au sens de la théorie des ensembles) des deux langages. LL' est par définition le langage formé par les mots qui sont les concaténations des mots de L avec les mots de L'. Ainsi, LL' = {*uv*, avec *u* dans L et *v* dans L'}. Ensuite, l'*étoile de Kleene* L* de L est le langage défini par {ε}∪L∪LL∪... Enfin, L^C est le *complémentaire* du langage L (c'est l'ensemble des mots finis qui appartiennent à A* mais pas à L).
Amusons-nous sur un exemple.
Le langage {*aa* + *b*}* contient par exemple *aab* ou *bbaa* mais pas *ab*. Saurez-vous le démontrer à l'aide d'un raisonnement rigoureux ?



La hiérarchie de Chomsky est une classification des langages formels et des grammaires formelles. Les automates finis reconnaissent les langages rationnels tandis que les machines de Turing peuvent calculer tout ce qui est calculable.

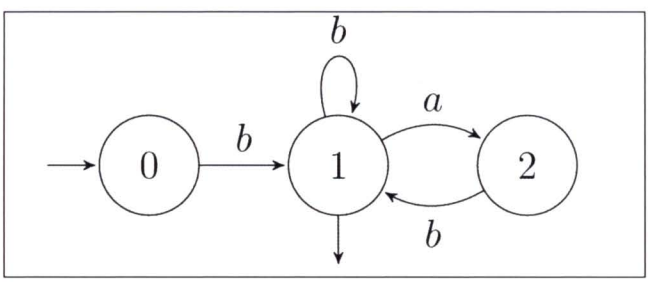
Parmi tous les langages concevables, les *langages rationnels* sont définis ainsi : pour chaque lettre *a* dans A, {*a*} est un langage rationnel ; le complémentaire d'un langage rationnel est un langage rationnel ; l'union ou la concaténation de deux langages rationnels est un langage rationnel, et l'étoile d'un langage rationnel est un langage rationnel. On représente généralement les langages rationnels par des expressions rationnelles :

- *a* est l'expression rationnelle dénotant le langage {*a*} pour une lettre *a* appartenant à A,
- *e* + *e*' dénote l'union des langages dénotés par *e* et *e*',
- *ee*' dénote la concaténation des langages dénotés par *e* et *e*',
- *e** dénote l'étoile de Kleene du langage dénoté par *e*,
- A**e* dénote le complémentaire du langage dénoté par *e*.

Par exemple, si on considère l'alphabet A = {*a*, *b*}, le langage des mots finissant par *ab* est dénoté par A**ab* et le langage des mots commençant par *b*, finissant par *b* et ne contenant jamais deux *a* consécutifs est dénoté par *b*(*ab* + *b*)*, signifiant : « d'abord, un *b*, suivi d'un nombre arbitraire de mots parmi *ab* et *b* » (convainquez-vous bien que nécessairement tout mot ainsi construit finira par *b*).
Le langage {$a^n b^n$, avec *n* un entier quelconque}, qui est simplement {ε, *ab*, *aabb*, *aaabbb*...}, est un exemple célèbre de langage non rationnel.
Les notations sur les langages rationnels ne sont pas sans rappeler les expressions régulières : lorsque je veux vérifier dans cet article que je n'ai pas commencé un paragraphe par une lettre minuscule, mon éditeur de texte me permet d'utiliser l'expression régulière \n[a-z], qui correspond à un caractère de retour de ligne (*newline*) suivi d'une lettre minuscule, pour rechercher toutes les occurrences incriminées.

Des automates dans tous leurs états

Un mot étant donné, on peut se demander s'il fait partie ou non d'un langage. Pour cela, il existe une machine abstraite (appelée *automate fini*), composée d'un nombre fini d'états reliés par des flèches étiquetées par des lettres. Les automates *déterministes* n'ont qu'un seul état initial et une unique flèche étiquetée par une lettre pour chaque état. Pour tester un mot, il faut partir de l'état initial et suivre les flèches correspondant aux lettres du mot les unes après les autres. Si l'automate est déterministe, il ne possède qu'un état initial et il existe, à chaque étape, au plus une flèche étiquetée par chaque lettre. Le chemin (on dit aussi *calcul*) ainsi construit à partir d'un mot sur un automate déterministe est donc unique. Après avoir épuisé les lettres du mot, si on se trouve sur un état final, le mot est accepté, sinon il est rejeté par l'automate. Le *langage reconnu* par cet automate est l'ensemble des mots acceptés par cet automate. Dans le même ordre d'idées, un langage est *reconnaissable* s'il existe un automate qui le reconnaît. De manière remarquable, un langage est reconnaissable si, et seulement si, il est rationnel. C'est le théorème de Kleene, dont il existe une pléthore de démonstrations.



Un automate qui reconnaît le langage $b(ab + b)^*$. Par exemple, pour tester *bba*, on part de l'état initial 0 (indiqué par la flèche entrante). La flèche étiquetée par *b* nous mène en 1, la flèche suivante nous laisse en 1 et on finit enfin en 2, qui n'est pas un état final : le mot est donc rejeté. Le mot *a* est directement rejeté. Le mot *bbab*, en revanche, fait arriver en 1, qui est final (indiqué par la flèche sortante), donc le mot est accepté. Et en effet, le langage dénoté par $b(ab + b)^*$ contient *bbab* mais pas *bba*.

En particulier, pour chaque expression rationnelle on peut construire un automate qui reconnaît le langage dénoté par cette expression. Une intuition pour comprendre pourquoi $\{a^n b^n$, avec n un entier quelconque$\}$ n'est pas rationnel est de remarquer que les états d'un automate peuvent renfermer une information bornée. Or, pour accepter un mot de ce langage, lorsqu'on a lu un nombre arbitrairement grand de a, on a besoin de vérifier qu'il y a autant d'occurrences de b.

Pour compter les occurrences du mot *chaton* dans un texte, on peut construire un automate qui reconnaît le langage rationnel A**chaton* (les mots qui finissent par *chaton*) sur l'alphabet naturel, qui peut lire le texte à travers l'automate et incrémenter un compteur à chaque fois que l'on visite un état final (qui correspondra à une occurrence de *chaton* dans le texte).

On peut autoriser un automate à posséder plusieurs chemins étiquetés par un mot. On obtient ce qu'on appelle un automate *non déterministe* et un mot est accepté si au moins un des chemins qu'il étiquette est acceptant, rejeté sinon.

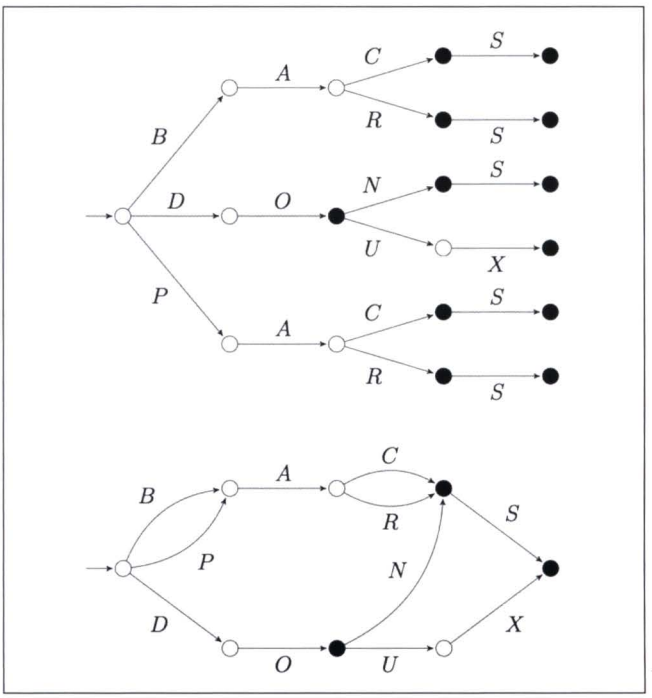
Bonne nouvelle, il est possible de construire pour chaque automate non déterministe un automate déterministe qui reconnaît le même langage ! Mauvaise nouvelle, l'automate ainsi construit peut avoir exponentiellement plus d'états… Ces deux classes d'automates ont donc étonnamment le même pouvoir expressif : ils sont équivalents en ce qu'ils sont capables de calculer, mais ils diffèrent dans la manière dont il le calculent. Il faut trouver un compromis entre le faible nombre d'états des auto-

mates non déterministes et l'unicité du chemin des automates déterministes. En particulier, un résultat dû à Valentin M. Antimirov permet de construire, à partir d'une expression rationnelle à n lettres, un automate non nécessairement déterministe à $n + 1$ états au pire qui le reconnaît. Un tel automate est appelé *automate aux termes dérivés*.

Parmi tous les automates qui reconnaissent le même langage, il en existe un unique, déterministe, à nombre d'états minimal, appelé… *automate minimal*. De plus, cet automate est calculable efficacement.

Les langages contenant un nombre fini de mots sont rationnels puisque la somme de tous les mots d'un tel langage constitue une expression rationnelle finie qui le dénote. Le dictionnaire électronique des formes fléchies du français, réalisé par les linguistes de l'université de Marne-la-Vallée, contient huit cent deux mille neuf mots sur un alphabet de quatre-vingt-dix lettres (minuscules et majuscules accentuées, chiffres et autres signes). Sa représentation par arbre lexicographique nécessite 2 203 261 nœuds, tandis que l'automate minimal le reconnaissant ne possède que 273 716 états. Les automates constituent donc un excellent moyen de compresser l'information contenue dans un dictionnaire, et donc d'accepter des mots validés ou rejetés dans une version électronique du jeu de Scrabble par exemple.

Ainsi, ces machines très simples permettent de décrire une vaste classe de langages. Et c'est ce qui est à la base de la complexité algorithmique : savoir ce que l'on peut décrire à partir de ressources limitées, ou réciproquement trouver la capacité de calcul suffisante pour répondre à un problème donné. À noter que si les automates finis peuvent être vus comme des programmes qui prennent en entrée un mot et retournent 0 ou 1 selon s'il est accepté ou rejeté, il en existe une généralisation (les *transducteurs finis*) qui transforme l'entrée en une sortie sur un autre alphabet, ce qui a des applications en compilation, en reconnaissance de la parole et en analyse grammaticale.



Une représentation par arbre lexicographique. Une représentation équivalente par automates.

J.-J. V.

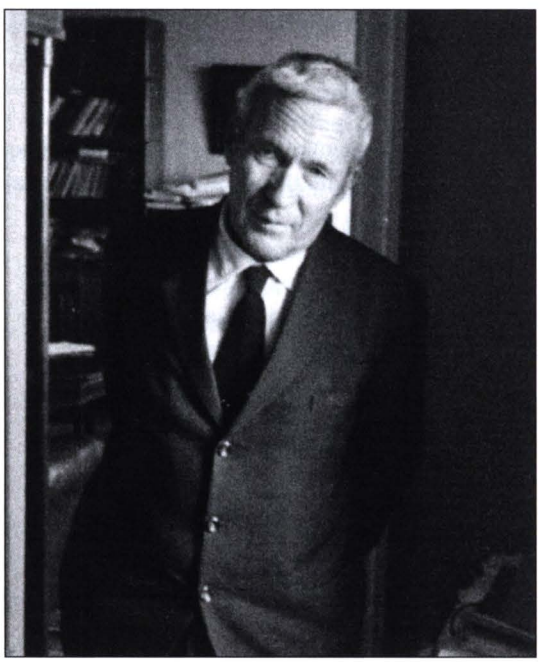
Références

- *Éléments de théorie des automates*. Jacques Sakarovitch, Vuibert, 2003.
- *Fondations mathématiques de la théorie des automates*. Jean-Éric Pin et Olivier Carton, disponible en ligne.
- *Combinatorics on Words*, M. Lothaire, Cambridge University Press, 1997.

par Jean-Paul Delahaye

Complexité de Kolmogorov et profondeur logique de Bennett

Pour mesurer la complexité d'un objet numérique, selon que l'on considère son contenu en informations, ou son contenu en structures, deux notions très différentes sont obtenues. À l'origine de ces outils se trouvent les travaux du mathématicien Andreï Kolmogorov.



Andreï Nikolaïevitch Kolmogorov (1903–1987), photographié par Paul Halmos en 1965.

Les tentatives de mathématisation des notions naturelles de « simple » et de « complexe » n'ont abouti à des résultats intéressants que depuis quelques années, grâce à la théorie algorithmique de l'information proposée par Andreï Kolmogorov en 1965. Charles Bennett a depuis donné un sens mathématique à une distinction tout aussi naturelle et importante, mais qui jusqu'à présent échappait à la formalisation, la distinction entre ce qui est « complexe car aléatoire » (comme un gaz ou un tas de cailloux) et ce qui est « complexe car très organisé ou très structuré » (comme une puce informatique ou un être vivant). Ces deux concepts – complexité de Kolmogorov et profondeur logique de Bennett – concernent toutes les sciences.

Que signifie « complexe » ?

Complexe peut vouloir dire « long à décrire en détail » ou « riche en structures et subtilement organisé ». Les deux idées que sont le « contenu en informations » et le « contenu en structures » sont relativement indépendantes. « Une allée rectiligne empierrée » est difficile à décrire entièrement dans le détail, car il faut indiquer l'emplacement et la forme de chaque caillou. Pourtant elle est facile à décrire pour ce qui est de sa structure générale, puisque seule sa forme géo-

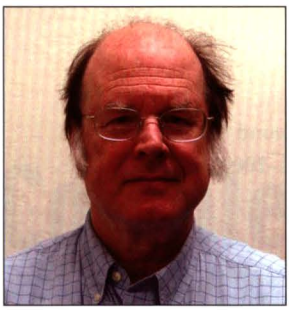
métrique (longueur, largeur…) est importante alors.
Il y a bien deux concepts de complexité à ne pas confondre : la « complexité aléatoire » et la « complexité organisée (ou structurelle) ». Pour l'illustrer plus précisément, considérons le problème de la description au millimètre près d'une maison dont les murs sont couverts de crépi. Le plan de la maison correspond à la complexité organisée de la maison. Ce plan ne précise pas les dessins du crépi sur les murs. La description complète de la maison, qui contient tous les détails du crépi, comporte bien plus d'informations que celle du plan. La maison possède une *complexité organisée* de taille moyenne (un plan n'est pas très compliqué comparée par exemple à un être vivant) et une *complexité aléatoire* assez grande.

Programmes courts et temps de calcul

Le concept mathématique de complexité aléatoire a été identifié dans les années 1960, grâce aux travaux préliminaires de Ray Solomonoff, et à ceux d'Andreï Kolmogorov, de Gregory Chaitin et de Leonid Levin. Cette complexité se nomme *complexité de Kolmogorov*. Elle est définie comme la taille du plus petit programme pour un ordinateur de référence – appelé *machine universelle* – capable de produire l'objet numérique auquel on s'intéresse et que l'on a supposé écrit sous la forme d'une suite de 0 et de 1 (image numérique, son numérique…). Une suite d'un milliard de 0 a une faible complexité de Kolmogorov, de même qu'un milliard de chiffres binaires de π (car des programmes courts permettent de les calculer). Une suite aléatoire d'un milliard de 0 et de 1, à l'inverse, possède une complexité de Kolmogorov d'environ un milliard.

Raymond Solomonoff (1926–2009).

La définition de la profondeur logique de Bennett s'appuie sur la théorie de la calculabilité et est assez technique à énoncer ; elle fait en particulier intervenir les notions de machine de Turing universelle et de fonctions récursives. Mais cela n'empêche pas de comprendre assez bien intuitivement de quoi il s'agit. Pour trouver le « bon » concept mathématique associé à la complexité structurelle, Charles Bennett a mené un travail d'analyse. Pour lui, un objet fortement organisé contient nécessairement en lui la trace d'un long processus d'élaboration, de réflexion ou d'évolution qui correspond à une forme de calcul. Définir la complexité organisée d'un objet se ramène donc au problème de la définition d'une notion de *contenu en calcul*. En informatique théorique, les travaux sur les algorithmes prennent bien en compte les temps de calcul (classes P, NP, EXP…), mais ces études s'attachent surtout aux comportements asymptotiques des algorithmes, alors qu'ici on n'a à faire qu'à des objets numériques finis, ou que l'on ramène à des objets finis en fixant un niveau de précision considéré comme suffisant pour la numérisation. Pour définir le contenu en calcul d'un objet numérique (c'est-à-dire sa complexité organisée), Bennett propose de considérer le temps de calcul que prend le programme minimal (celui dont la taille définit la complexité de Kolmogorov) pour produire l'objet auquel on s'intéresse. C'est une



Charles Henry Bennett (né en 1943).

sorte de « temps de décompression » car le programme minimal est la forme comprimée la meilleure de l'objet numérisé, et exécuter ce programme minimal c'est décompresser l'information compressée de manière extrême dans le programme minimal. Ce temps de calcul, Bennett l'appelle *profondeur logique* de l'objet numérique.

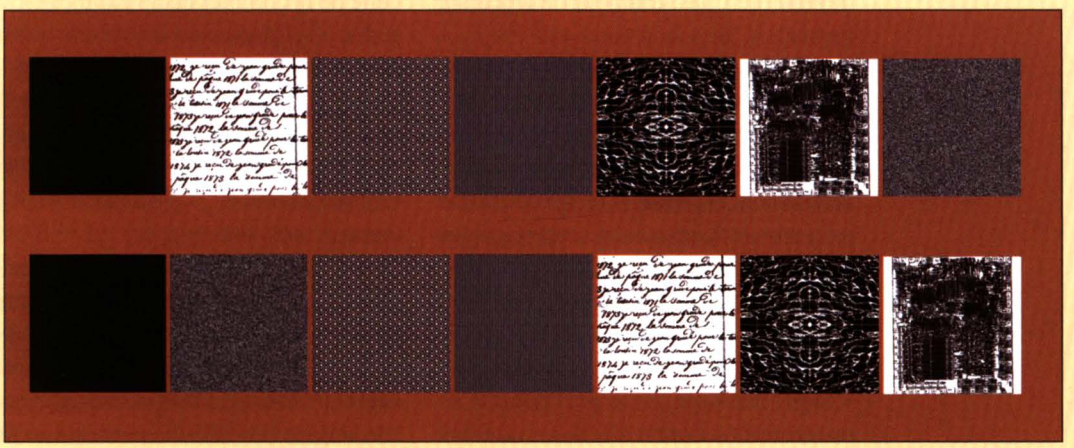
La profondeur logique de Bennett n'a été proposée que dans les années 1980, car l'idée la plus tentante pour définir le *contenu en calcul* d'un objet est de mimer la définition de la complexité de Kolmogorov et donc de définir le contenu en calcul d'un objet numérique comme « le temps de calcul du programme le plus rapide capable de produire l'objet numérique ». Cette définition naturelle ne marche pas : en effet, ce temps minimal de calcul est toujours donné par le programme « imprimer Ob » où Ob est l'objet numérisé auquel on s'intéresse. Cette remarque est d'ailleurs bien connue des programmeurs, qui savent tous que le programme le plus rapide pour obtenir les vingt premières décimales de π est le programme « imprimer 14159265358979323846 ». La définition naturelle du contenu en calcul par le temps minimal de calcul d'un objet n'a donc pas de sens et ne mesure rien. C'est cette difficulté d'une définition tentante mais inopérante que Bennett a surmontée en faisant référence au programme minimal. Sa définition, qu'il ne faut pas confondre avec celle envisagée en termes de programme « le plus rapide », est vraiment que le contenu en calcul (ou profondeur logique) d'un objet numérique Ob doit être considéré comme « le temps de calcul du programme minimal (en taille) de Ob ». Un objet « profond », c'est-à-dire ayant une grande profondeur logique, est un objet dont l'origine la plus probable est un long calcul. C'est un objet qui contient des redondances parfois profondément cachées. Pour tester si la définition de Bennett correspond bien à notre attente intuitive, considérons divers exemples.

Un bloc de cristal possède clairement une faible complexité aléatoire (puisqu'il n'est pas du tout aléatoire !) et une faible complexité en organisation (puisque son organisation est une simple répétition). En utilisant les définitions formelles, on constate que, conformément à cette intuition, la complexité de Kolmogorov est petite puisque le programme minimal pour décrire la version numérisée du bloc de cristal est court, et que sa profondeur logique aussi est petite, puisque le programme minimal est un programme d'itérations élémentaires du genre « mille fois de suite, reproduire le motif de base du cristal », qui s'exécute rapidement.
Comme deuxième exemple, prenons un litre de gaz (supposé numérisé avec une précision de l'ordre du millionième de millimètre par exemple). C'est un objet qui possède une très grande complexité aléatoire. Les molécules du gaz sont réparties au hasard, on ne peut rien faire de mieux pour décrire le litre de gaz qu'utiliser un programme du type « imprimer Ob ». Il n'y a pas ou très peu de raccourcis possibles dans la description. La complexité organisée est faible (le programme minimal « imprimer Ob » ne fait pas de calculs subtils). À nouveau, les définitions mathématiques s'accor-

Sur un exemple

Un programme de compression de données permet d'évaluer à la fois la complexité de Kolmogorov et la profondeur logique de Bennett. L'idée est que la version compressée d'un fichier doit être vue comme un court programme engendrant le fichier. La taille de ce fichier indique donc une valeur approchée de la complexité de Kolmogorov. De plus, le temps nécessaire à la décompression du fichier est assimilable au temps de calcul de ce court programme pour produire le fichier initial, donc peut être vu comme une évaluation de sa profondeur logique de Bennett.

Voici concrètement ce que l'on obtient avec des images. Les expériences ont été réalisées par Hector Zenil, Cédric Gaucherel et l'auteur. On a pris sept images de même format que l'on a compressées par utilisation d'un algorithme de compression sans perte (l'image que l'on retrouve après décompression est exactement celle avant compression).



La première série indique le classement des images par ordre croissant de taille du fichier compressé. Le classement est donc celui par complexité de Kolmogorov K($\boldsymbol{s}$) croissante. Sans surprise, l'image toute noire (image 1, en haut à gauche) a le contenu en information le plus petit, et l'image composée de pixels tirés aléatoirement (image 7) est celle qui exige la plus grande quantité de mémoire (K($\boldsymbol{s}$) est maximal). Les autres images sont classées à peu près comme on s'y attend : le texte écrit à la main (image 2) demande assez peu de mémoire car il y a beaucoup de blanc ; un réseau périodique (image 3) ; une courbe de Peano (image 4) ; une image irrégulière mais avec deux axes de symétrie (image 5) ; un microprocesseur (image 6).

La seconde série d'images (ligne du bas) reprend les sept mêmes images, mais cette fois par ordre croissant de temps de décompression, ce qui donne des valeurs approchées de la profondeur logique de Bennett, P($\boldsymbol{s}$), et donc un classement par complexité structurelle croissante. L'image 7 du premier classement (qui est parfaitement aléatoire) est maintenant parmi les premières, conformément à l'idée qu'un hasard parfait est sans structure. Le microprocesseur est bien identifié comme le plus complexe structuralement. La courbe de Peano est sans surprise toujours considérée comme contenant des structures un peu plus riches que le motif périodique. Le texte écrit et le motif symétrique changent de position, donnant au total un classement compatible avec ce qui, intuitivement, correspond à une complexité structurelle croissante.

Kurt Gödel (1906–1978).

dent avec nos attentes : la complexité de Kolmogorov de l'objet est grande (le programme minimal est long) et sa profondeur logique est faible (le programme minimal n'a pas vraiment de calculs à mener).

Pour l'instant, on n'a rencontré que des objets de petite profondeur logique. Ce n'est pas le cas des cent mille premières décimales de π, qui constituent un objet subtilement organisé ! On sait écrire des programmes relativement courts capables d'engendrer ces cent mille décimales, et donc, contrairement au cas où l'on s'intéresse uniquement à vingt décimales de π, ce n'est plus le programme « imprimer » qui est le plus court. Ces programmes courts doivent calculer longtemps et ne produisent leurs décimales qu'à petite vitesse (contrairement à un programme « imprimer Ob »). Cela signifie que la profondeur logique est grande. Conformément à notre attente, les cent mille premières décimales de π ont une faible complexité de Kolmogorov et une assez grande profondeur logique de Bennett.

Comme dernier exemple, considérons un mouton. Sa complexité aléatoire est grande car (par exemple) la répartition de la laine sur sa peau ne suit pas un motif parfaitement régulier. Sa profondeur logique, elle aussi, est grande car on pourrait (en théorie) décrire le mouton, en donnant son génome et en demandant au programme de simuler le processus de développement, ce qui prendrait beaucoup de temps. Le mouton est un objet complexe, aussi bien en complexité aléatoire qu'en complexité organisée.

Croissance lente et indécidabilité

Les développements mathématiques que Bennett a donnés à ses idées sont intéressants sous plusieurs aspects. D'abord, il a montré que, moyennant une bonne définition des ordinateurs de référence, la définition qu'il propose ne dépend pratiquement pas de l'ordinateur choisi : sa notion est donc (comme celle de la complexité de Kolmogorov) stable et globalement invariante quand on change la machine utilisée pour la mesurer. Ensuite, il a montré que la notion de profondeur logique vérifie ce qu'il appelle une *loi de croissance lente* : l'augmentation de la profondeur ne peut être que très lente (ou encore : il n'y a qu'une très faible probabilité pour que, durant un court processus dynamique, un objet profond apparaisse spontanément). Ceci confirme que, face à un objet profond, on doit considérer que son origine probable ne peut être qu'un long calcul : un objet profond porte (implicitement) en lui la trace d'un long processus d'élaboration.

Plus malheureuses sont les conséquences des résultats d'indécidabilité de Gödel (toujours eux !) qui, aussi bien pour la complexité de Kolmogorov que pour la profondeur logique de Bennett, montrent que calculer avec certitude les valeurs de ces deux mesures de complexité est une tâche d'une extrême difficulté, qui sera infaisable de manière exacte dès que l'on devra traiter des objets non triviaux. Ce n'est peut-être pas surprenant, car on comprend bien que face à un objet profond (pensons aux décimales de π entre la cent millième et la deux cent millième) il est difficile de décider entre les explications « c'est un objet de grande complexité aléatoire »

ou « c'est un objet de petite complexité aléatoire mais profond ». Pour trancher, il faut avoir identifié en quoi la complexité est organisée. Ce qui est profond peut avoir l'apparence de l'aléatoire. Il n'est pas absurde de considérer que le travail de la recherche scientifique est l'identification de la complexité organisée, là où apparemment ne se trouve que de la complexité aléatoire.

Aujourd'hui, les algorithmes de compression de données (sans pertes) sont les meilleurs outils pour mener des mesures de complexité :

- La taille du fichier compressé est une approximation de la complexité de Kolmogorov.
- Le temps de calcul pris pour décompresser un fichier compressé donne une évaluation de sa profondeur logique de Bennett (elle n'est vraiment satisfaisante que si l'algorithme de compression a bien su repérer les structures du fichier qu'on lui a confié).

Les résultats de Bennett montrent aussi que l'apparition lente de la complexité organisée ne contredit aucunement la seconde loi de la thermodynamique, qui concerne l'accroissement de la complexité aléatoire. Le fait que la complexité organisée s'accroisse au cours du temps (comme on le constate sur Terre) est simplement le signe que dans le monde physique se déroulent des processus assimilables à du calcul (avec mémorisation). Ce n'est pas choquant : les mouvements mécaniques, les interactions chimiques, les processus de sélection, les mouvements culturels sont des sortes de calculs. Si on adopte le point de vue de Bennett, l'augmentation de la complexité organisée sur Terre depuis quatre milliards d'années est compatible avec la thermodynamique, et cela sans avoir recours à des pirouettes comme quand on confondait complexité aléatoire et complexité organisée.

Un autre problème est de savoir si les lois du monde physique sont telles que, nécessairement, se produit un accroissement de la complexité organisée. Ce n'est pas parce qu'une sorte de calcul se déroule dans le monde physique qu'un autre calcul plus rapide n'est pas possible ou que les résultats de ce calcul ne peuvent pas être détruits (auquel cas bien sûr aucune croissance de profondeur logique ne se produit). Le fait que notre monde physique autorise de longs calculs ne prouve donc pas qu'il est le lieu d'une augmentation inévitable de profondeur logique. Les bons concepts mathématiques semblent identifiés. La question « pourquoi assiste-t-on à un accroissement de la complexité organisée ? » possède maintenant un sens purement mathématique, et peut donc recevoir une réponse mathématique. Si on y arrive, peut-être saurons-nous alors pourquoi la vie *devait* apparaître sur Terre !

J.-P. D.

Références

- *Mesurer la complexité des objets numériques*. Jean-Paul Delahaye, *Bulletin de la Société informatique de France* (1), 2013, disponible en ligne.
- *Image Characterization and Classification by Physical Complexity*. Hector Zenil, Jean-Paul Delahaye et Cédric Gaucherel, *Complexity* 17 (3), 2012.
- *Complexité aléatoire et complexité organisée*. Jean-Paul Delahaye, Quæ, 2009.
- *Logical Depth and Physical Complexity*. Charles Bennett, in *The Universal Turing Machine: A Half-Century Survey*, Oxford University Press, 1988.
- *How to define complexity in physics, and Why*. Charles Bennett, in *Complexity, Entropy and the Physics of Information*, SFI Studies in the Sciences of Complexity (VIII), Addison-Wesley, 1990.
- *Information, Randomness and Incompleteness: Papers on Algorithmic Information Theory*. Gregory Chaitin, World Scientific, 1987.
- Information, complexité et hasard. Jean-Paul Delahaye, Hermès, 1999.
- *Three Approaches for Defining the Concept of Information Quantity*. Andreï Kolmogorov, *Information Transmission* (1), 1965.

par Nicolas Delerue

Comment éliminer les *spams* ?

Comment rejeter les courriers électroniques non sollicités sans perdre de messages ? La solution la plus efficace à l'heure actuelle repose sur l'utilisation d'un théorème de mathématiques : le théorème de Bayes.

Un inconnu vous propose une importante somme d'argent en échange d'une transaction sans risque. Une société vous propose des pilules miracles pour presque rien. Un inconnu vous signale l'existence d'un site Web vendant des produits de luxe à des prix défiant toute concurrence. Ce genre de messages peut faire rêver. C'est en fait le cauchemar des internautes dont la boîte à lettres électronique se remplit chaque matin de courriers non sollicités. Ils sont tellement nombreux que l'on a trouvé un nom pour les désigner : il s'agit des *spams* (ou pourriels en français).

Les auteurs de pourriels profitent de la quasi gratuité de l'envoi de messages sur Internet pour inonder toutes les boîtes dont ils trouvent l'adresse de propositions de toutes sortes. Une étude statistique a montré que le taux de réponses positives à ces pourriels est de quinze pour un million. C'est infime mais le prix de l'envoi l'est encore davantage, si bien que, aussi incroyable que cela puisse paraître, ce genre de pratique est très profitable !

Pour les destinataires de ces messages, il reste à nettoyer quotidiennement leurs

boîtes aux lettres. Le coût de ces nettoyages est facile à calculer. À raison d'une seconde par message, un million de messages demande 278 heures pour être effacés, ce qui représente deux mois de travail d'un salarié. C'est exorbitant. C'est pourquoi des algorithmes ont été mis au point pour identifier et détruire automatiquement les pourriels.

Des mots clefs à la reconnaissance automatique

La première idée venant à l'esprit pour identifier les pourriels est de tester la présence de quelques mots clefs caractéristiques de ces messages. Les meilleurs candidats sont sans doute « viagra » ou « sexe » ainsi que d'autres mots dans

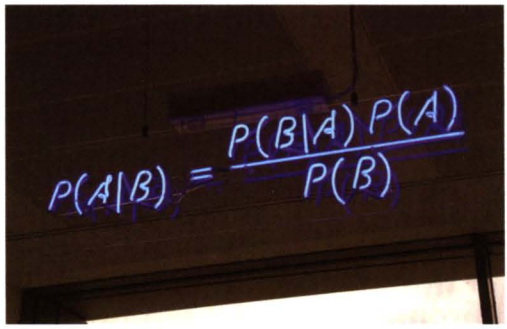
le même registre. Cependant, cette idée montre rapidement ses limites pour plusieurs raisons.
Tout d'abord, un courrier normal peut très bien utiliser ces mots et vous risquez de les éliminer automatiquement. Ensuite, certains pourriels ne les utilisent pas. D'autre part, il suffit de changer « viagra » en « Viagra », « VI4GR4 » ou « VlAGRA » pour contourner la difficulté.
Des filtres plus développés ont été mis au point pour tenir compte de ces problèmes. Cependant, les techniques employées par les auteurs de pourriels évoluent sans cesse, ce qui nécessite une mise à jour permanente des règles employées. Et bien que ces règles soient très efficaces pour se débarrasser d'une bonne partie du *spam*, le filtrage des messages restants est souvent très difficile.

Est-il si difficile de distinguer les pourriels des vrais messages ? Pour un être humain en ayant déjà reçu quelques-uns, un coup d'œil suffit. Ce n'est pas la présence d'un mot particulier qui les rend facile à identifier, mais plutôt l'accumulation de mots dans un même registre. Dans le cas des pourriels, il s'agit souvent du registre pornographique. La différence entre un texte normal même très vulgaire sur le sujet et un pourriel est l'accumulation. Même de mauvaise qualité, un texte normal respecte certaines contraintes littéraires.
À l'heure actuelle, les logiciels les plus performants se fondent sur ce constat. Il faut analyser les mots dans leur contexte pour savoir si un message est du *spam* ou non. La lecture de tous les mots du message permet en effet d'avoir une vue plus globale de son contenu et donc de sa nature. Pour cela, chaque mot reçoit une probabilité : celle qu'il figure dans un *spam*. Lors de l'arrivée d'un message, tous ses mots sont lus ; il est ainsi transformé en un ensemble de probabilités. Ces probabilités sont combinées pour donner un indice indiquant la probabilité que le message soit du *spam* ou non. Comme ce calcul de probabilités fait appel au théorème de Bayes, ces filtres sont appelés *filtres bayésiens*.

Pour calculer la probabilité qu'un mot apparaisse dans un pourriel, rien de très difficile : prenez un grand nombre de pourriels, utilisez un logiciel pour trier les mots s'y trouvant et comptez le nombre d'occurrences de chacun. Faites la même chose avec un ensemble de message n'étant pas du *spam* (vos messages personnels par exemple) et vous obtenez, pour chaque mot, la probabilité qu'il apparaisse dans un pourriel et la probabilité qu'il apparaisse dans un message normal. Après l'analyse d'un message, vous pouvez, en multipliant les probabilités que chaque mot apparaisse dans un pourriel, en déduire la probabilité que l'ensemble des mots du message apparaissent dans un pourriel. Mais attention, la probabilité que l'ensemble des mots d'un message apparaissent dans un pourriel n'est pas égal à la probabilité que cet ensemble de mots forme un pourriel !
Appelons P(S) la probabilité qu'un message soit du *spam*. Pour chaque mot M, il est facile de calculer la probabilité conditionnelle $P(M \mid S)$ que le mot M apparaisse dans un message de *spam*. Il est donc facile d'en déduire la probabilité qu'un ensemble de mots M_i formant un courrier électronique C apparaissent dans du *spam* :

$$P(C \mid S) = \prod_i P(M_i \mid S)$$

Par contre, pour connaître $P(S \mid C)$, c'est-à-dire la probabilité que ce message soit effectivement du *spam*, il est indiqué d'utiliser le théorème de Bayes.



***Le théorème de Bayes permet de suspecter si un message est du* spam.**

Ce théorème permet en effet d'inverser une probabilité : connaissant P(C | S), il permet de calculer P(S | C), à l'aide de la formule $P(S|C) = \frac{P(S)P(C|S)}{P(S) + P(\overline{S})}$.

Ici, P(S) est la probabilité qu'un message soit du *spam* et $P(\overline{S})$ la probabilité qu'il n'en soit pas (pour connaître ces probabilités, il suffit de compter le nombres de messages dans chacun des deux ensembles utilisés pour calculer les probabilités P(M | S)).

La probabilité P(S | C) est donc un indice permettant de suspecter qu'un message est du *spam*. Plus l'indice est élevé, plus le message est suspect. La spécificité des pourriels fait que cet indice atteint souvent des valeurs très élevées pour la plupart d'entre eux, et des valeurs très basses pour les autres messages. Il est aussi possible d'utiliser pour chaque mot la probabilité que celui-ci ne soit pas du *spam* pour en déduire la probabilité que ce message ne soit pas du *spam*. En comparant les deux indices, il est possible de conclure si le message est probablement du *spam* ou non.

Chaque algorithme utilisant un filtre bayésien a une efficacité différente (dépendant de la manière exacte dont les probabilités sont calculées et combinées). Le programmeur britannique Paul Graham, l'un des pionniers de cette méthode, affirme que son filtre ne laisse passer que 0,5 % des *spams* et ne bloque jamais un message standard. Quoi qu'il en soit, les performances du filtre bayésien dépendent de la qualité des *spams* utilisés pour le calcul des probabilités. Si vous n'utilisez que des *spams* vous vantant les mérites d'un certain produit pharmaceutique pour calculer les probabilités, ne vous étonnez pas qu'un *spam* se faisant passer pour un ami qui veut vous donner un million d'euros ne soit pas classé comme tel !

Il est possible de faire mieux en utilisant des méthodes d'intelligence artificielle. Par exemple, un réseau de neurones pourrait probablement, après apprentissage, être plus efficace qu'un filtre bayésien. Cependant, l'utilisation d'un réseau de neurones est assez complexe et il n'est pas sûr que le gain possible justifie l'augmentation de la complexité.

Le *spam* est apparu il y a quelques années et progresse rapidement. Les techniques utilisées par les auteurs de pourriels évoluent pour tenter de contourner les filtres qui leurs sont opposés. Les filtres anti-*spams* doivent donc en permanence s'adapter. Les filtres bayésiens sont l'une des dernières innovations en date, mais de nombreuses autres sont à venir. Probablement que ces nouvelles méthodes utiliseront d'autres théorèmes mathématiques plus ou moins connus !

N. D.

Informatique pour les mathématiques

Comme toute progéniture reconnaissante, l'informatique a bien rendu aux mathématiques ce qu'elle leur doit. Les interrogations nées de cette nouvelle science, les apports technologiques qu'elle permet, ont ouvert aux mathématiques des horizons nouveaux. L'expérimentation, la simulation, le calcul haute performance, la démonstration automatique... sont quelques-unes des nombreuses portes ouvertes par l'informatique aux mathématiciens.

par Jean-Paul Delahaye

Mathématiques expérimentales

Les mathématiques science expérimentale ? Cette affirmation, qui ferait bondir plus d'un mathématicien pur et dur, trouve de plus en plus d'échos grâce à l'outil informatique.

Plusieurs philosophes dont Imre Lakatos et Thomas Tymoczko ainsi que de nombreux mathématiciens ont insisté sur les aspects expérimentaux et inductifs de l'activité mathématique et sur certaines similitudes entre le travail du physicien et celui du mathématicien. Carl Frederich Gauss expliquait qu'il atteignait la vérité mathématique par l'expérimentation systématique. C'est de cette façon qu'il découvrit que le nombre de nombres premiers inférieurs à n est approximativement $n / \ln n$, affirmation qui ne fut prouvée qu'un siècle plus tard. Le logicien Kurt Gödel, cohérent avec ses positions réalistes – il croyait que les objets mathématiques ont une existence indépendante de nous –, remarquait que « *si les mathématiques décrivent un monde objectif, comme le fait la physique, il n'y a aucune raison pour que la méthode inductive ne puisse être appliquée en mathématiques comme elle l'est en physique* ». L'idée chez Gödel d'une induction analogue à celle des sciences empiriques concerne la découverte de nouveaux axiomes et le choix entre systèmes d'axiomes concurrents, opérations qui ne peuvent résulter des raisonnements déductifs seuls à l'œuvre dans les démonstrations mathématiques usuelles.

Parfois, l'idée de faits et d'expérimentations mathématiques va au-delà, surtout depuis que l'ordinateur s'est ajouté à la feuille, au crayon et aux instruments de tracé géométrique qui ont longtemps été les seuls outils des mathématiciens. Plusieurs livres récents sont consacrés à cette façon nouvelle de pratiquer les mathématiques avec un ordinateur comme outil (voir la bibliographie).

Comme l'affirme Hardy (voir encadré), les mathématiques n'auraient pas pour but général de découvrir des démonstrations, mais des connaissances sûres !

« J'ai toujours considéré qu'un mathématicien était en premier lieu un observateur »
Godfrey Hardy

Chacun des exemples des usages de l'informatique en mathématiques qui suivent illustrera un aspect de ces jeux expérimentaux dont l'importance va en s'accroissant au fur et à mesure que les outils informatiques se perfectionnent et se diffusent.

L'ordinateur pour forger l'intuition

Tout d'abord l'ordinateur sert à façonner l'intuition en créant une familiarité avec des objets et situations qui ne peuvent être réalisées matériellement. C'est la fonction *didactique* de l'expérimentation dont non seulement les élèves et les étudiants peuvent bénéficier, mais dont le chercheur lui-même tirera profit.

Manipuler des billes aide à se construire une image précise du monde des nombres entiers ; réaliser des découpages en carton donne une compréhension affinée de ce que sont les longueurs, les aires, les polyèdres, etc. De même, les simulations massives de tirages au hasard qu'on peut effectuer avec un ordinateur sont un bon moyen de développer le sens des probabilités.

Le *Principe du jeu direct* (« *bold play* ») affirme que «la méthode la plus efficace de jeu pour passer de A euros à B euros en jouant sur pair ou impair à la roulette s'obtient en misant toujours – par exemple sur pair – la somme maximale possible, sans dépasser le but visé». Par exemple : pour passer de 100 euros à 1 000 euros : misez 100 si vous avez 100, misez 300 si vous avez 300, misez 400 si vous avez 600, *etc*.

Ce principe, démontré par Dubbins et Savage en 1956, est difficile à établir rigoureusement. En revanche il est facile à mettre à l'épreuve expérimentalement. Il suffit d'essayer toutes sortes de stratégies de jeu non conformes au *Principe du jeu direct* et de mesurer leur efficacité par simulation, en comparant au résultat donné par le *jeu direct*. La loi devient petit à petit naturelle pour celui qui réalise ces simulations. Il en comprend la raison profonde : puisque les tirages élémentaires sont défavorables au joueur, son intérêt est de poser sur le

Les démonstrations ne servent qu'à convaincre de ce qu'on observe ?

Godfrey Hardy (1877–1947).

Le mathématicien Godfrey Hardy (qui fit venir Ramanujan en Europe) formula en 1928, à une époque où pourtant l'ordinateur n'était pas encore entré dans le jeu, une idée étonnante :

« J'ai toujours considéré qu'un mathématicien était en premier lieu un observateur, un homme qui, situé assez loin de paysages montagneux, décrit ce qu'il y voit. [...] L'analogie est un peu brutale, mais je suis certain qu'elle n'est pas trompeuse. En la poussant à son extrême, nous arrivons à la conclusion plutôt paradoxale que nous pouvons, en dernière analyse, nous contenter de noter ce que nous observons ; que les démonstrations sont ce que Littlewood et moi appelons des effets rhétoriques destinés à frapper les esprits, des images sur un tableau lors d'une conférence, des trucs pour stimuler l'imagination des étudiants. La vérité n'est pas exactement ainsi, mais ne s'en écarte pas beaucoup. L'image donne une idée aussi bien de ce qu'est la pédagogie mathématique que de ce qu'est la découverte mathématique. Il n'y a que les personnes étrangères aux sciences et mal informées qui imaginent que les mathématiciens font des découvertes en tournant la manivelle d'une machine miraculeuse. L'image en fin de compte donne une vision crue des démonstrations telles que les concevait Hilbert, qui ne sont en réalité que certains arguments en faveur de leurs conclusions et dont le but est seulement de convaincre.»

tapis de jeu le moins d'argent possible et donc d'aller droit au but.

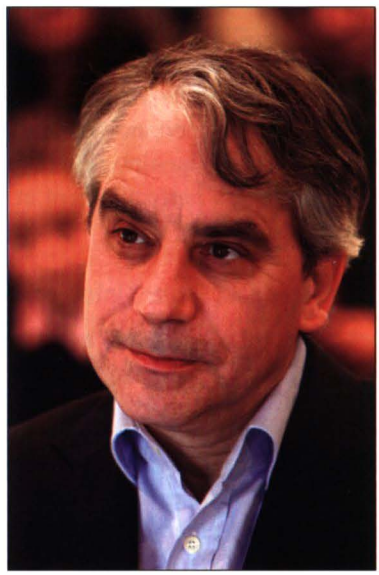
Cet exemple d'expérimentation dans un dessein didactique n'en est qu'un parmi une multitude d'autres. L'idée n'est pas nouvelle et ne fait que reprendre en l'adaptant la proposition de créer des *Laboratoires de Mathématiques*, proposition défendue par Émile Borel en 1904, soutenue depuis par Jean Dieudonné à propos de l'enseignement de la géométrie et reprise de nombreuse fois en didactique des mathématiques.

L'ordinateur producteur de faits mathématiques

Un second type d'expérimentations mathématiques est celui où l'on demande à l'ordinateur de produire un grand nombre de «faits mathématiques» qu'on analyse ensuite jusqu'à y découvrir des régularités, qu'il sera peut-être possible de démontrer. C'est ainsi qu'un ami mathématicien a redécouvert, il y a quelques années par l'expérimentation informatique la stratégie assurant de gagner au Jeu de Marienbad (ou jeu de Nim) basée sur la Nim-addition.

Le repérage informatique de régularités dans des faits mathématiques énumérés par la machine admet un cas particulier : la recherche de nouvelles formules par examen des chiffres décimaux qu'elle produit. La technique consiste à calculer avec une précision de plusieurs dizaines de chiffres diverses formules et à comparer les résultats obtenus. Lorsque les résultats de deux formules coïncident, on essaie de démontrer l'égalité repérée. En pratique, pour mener très rapidement de nombreuses comparaisons, des algorithmes *spéciaux* sont utilisés.

La plus fameuse découverte obtenue par



Simon Plouffe.

cette méthode est celle que fit Simon Plouffe en 1995 de l'égalité :

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k}\left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right).$$

Cette nouvelle formule de série pour π permet d'en calculer les chiffres binaires indépendamment les uns des autres, ce qui étonna tout le monde. Elle permet par exemple de calculer le milliardième chiffre binaire de π sans calculer les précédents.

Notons bien que la nouvelle formule a été prouvée rigoureusement par une démonstration au sens traditionnel. Faire travailler l'ordinateur et généraliser sans prendre la peine de démontrer se révélerait catastrophique. La nécessité de faire des démonstrations traditionnelles n'est contestée par personne.

L'ordinateur produit des faits mathématiques en grande quantité dont on peut lui demander quelles lois les organisent. Les régularités repérées deviendront alors des théorèmes si on réussit à les démontrer, ou seront des conjectures si on n'y parvient pas.

C'est ainsi qu'en calculant avec l'aide de programmes, les décimales des nombres

$\pi, e, \sqrt{2}$ et de bien d'autres nombres irrationnels, et en examinant —toujours avec l'ordinateur— leur répartition, on a été amené à formuler la conjecture que ces nombres sont *normaux* : chaque décimale se présente avec la fréquence 1/10, chaque série de 2 chiffres (par exemple 37) se présente avec la fréquence 1/100, *etc*.

L'ordinateur falsificateur

Karl Popper a défendu l'idée qu'une théorie a d'autant plus de contenu qu'elle est facile à mettre à l'épreuve – ou, avec son vocabulaire *à falsifier*. Pour falsifier des conjectures, rien de tel qu'un ordinateur et ici l'expérimentation mathématique ressemble d'assez près à l'expérimentation physique : on mène des expériences (de calcul) avec des données différentes, en observant si la loi qu'on teste est satisfaite à chaque fois.

L'énoncé du Grand Théorème de Fermat indique par exemple qu'il n'existe pas de quadruplets d'entiers a, b, c, n avec a, b, et c non nuls et $n > 2$, tels que : $a^n + b^n = c^n$. Un tel énoncé serait falsifié par n'importe quel quadruplet a, b, c, n satisfaisant les conditions citées. Le grand théorème de Fermat était donc falsifiable avant qu'il ne soit démontré par Andrew Wiles. De nombreux essais avaient d'ailleurs été faits pour le falsifier, et aujourd'hui encore certains mathématiciens lancent sans doute des programmes qui recherchent des *falsificateurs* pour ce théorème célèbre. Ces tentatives ne sont pas absurdes, car au fur et à mesure du temps, leur échec confirme indirectement que la démonstration de Wiles est juste.

Diverses généralisations du théorème de Fermat ont été envisagées dont celle-ci proposée par Leonhard Euler : une somme de moins de n puissances n-ièmes de nombres non nuls n'est jamais une puissance n-ième sauf dans les cas évidents $n = 2$ ou $a^n = a^n$.
Il se trouve que cette généralisation est fausse pour $n = 5$. En effet, en 1966, L. Lander et T. Parkin ont trouvé – bien sûr en utilisant un ordinateur – que :
$27^5 + 84^5 + 110^5 + 133^5 = 144^5$.
Pour $n = 4$ la généralisation d'Euler du théorème de Fermat est encore fausse et en 1988, N. Elkies a démontré qu'il y avait une infinité de solutions (non multiples les unes des autres) dont la plus petite, qui fut trouvée par ordinateur par R. Fries de la Thinking Machine Corporation, est :
$95\,800^4 + 217\,519^4 + 414\,560^4 = 422\,481^4$.
Pour $n = 6$ et au-delà, la conjecture reste aujourd'hui irrésolue.

Si l'ordinateur n'a servi à rien dans la démonstration du *Grand Théorème de Fermat*, sa capacité à falsifier d'autres conjectures proches doit être vue comme cruciale. Sans lui, des mathématiciens continueraient à chercher une démonstration de la généralisation proposée par Euler... que d'ailleurs Euler pensait vraie.

Bien sûr, le plus souvent, les conjectures résistent et le travail de ceux qui tentent de les falsifier apparaît absurde : si la conjecture est vraie, ils ne trouveront jamais rien, quel que soit le soin ou le génie qu'ils auront pu mettre dans leurs programmes. On peut penser que c'est ce que se passe pour la *conjecture de Syracuse* dont la vérification chaque année progresse grâce à des programmes de plus en plus complexes et subtils.
Cette conjecture affirme que la fonction $f(n) = n/2$ si n pair, $f(n) = 3n+1$ si n impair conduit toujours à 1 quand on l'applique de manière répétée à un entier : $n \to f(n) \to f(f(n)) \to f(f(f(n))) \to \ldots$
Cette conjecture a été vérifiée pour tous

les entiers inférieurs à :
$5 \times 2^{60} \approx 5{,}764 \times 10^{18}$.

L'ordinateur assistant aux capacités multiples

Récemment, des progrès ont été faits à propos de ce qui est sans doute la plus ancienne conjecture mathématique : « il n'existe aucun nombre parfait impair ».
Les nombres parfaits sont les nombres qui comme 6 ou 28 sont égaux à la somme de leurs diviseurs stricts :
$6 = 1 + 2 + 3$; $28 = 1 + 2 + 4 + 7 + 14$.
On connaît aujourd'hui 48 nombres parfaits pairs, ce sont les nombres de la forme $2^{n-1}(2^n - 1)$ avec $2^n - 1$ nombre de Mersenne premier, mais aucun nombre parfait impair. La recherche infructueuse de nombres parfaits impairs a suggéré aux mathématiciens qu'il n'existe pas de nombre parfait impair (c'est la conjecture).
Depuis plus de deux millénaires la question est posée.
Les progrès à ce sujet méritent d'être mentionnés car ils constituent un bon exemple de ce que sont les interactions entre mathématiciens et ordinateurs. Ils consistent principalement en résultats du type : s'il existe un nombre parfait impair, il possède au moins F facteurs premiers, et est plus grand que N.
Le dernier résultat record de ce type est dû à Pascal Ochen et Michael Rao qui ont établi que s'il existe des nombres parfaits impairs, ils possèdent au moins 101 facteurs dans leurs décompositions en facteurs premiers et ils sont plus grands que 10^{1500}.
Tout cela a été établi en utilisant des ordinateurs, mais pas d'une manière naïve en faisant défiler les nombres impairs les un après les autres et en s'assurant qu'aucun n'est pas égal à la somme de ses diviseurs propres (d'ailleurs on ne sait pas factoriser en temps acceptables tous ces nombres). La méthode exhaustive, en l'état actuel de la technologie, ne pourrait même pas faire défiler tous les nombres jusqu'à 10^{25}.
La méthode utilisée par les chercheurs s'intéressant aux nombres parfaits impairs associe des raisonnements arithmétiques, conduisant à des lemmes et théorèmes, qui permettent de découper le problème en cas qu'on traite alors soigneusement, parfois en utilisant un ordinateur auquel on confie par exemple un travail de factorisation. Le tout conduit à la conclusion via un raisonnement par l'absurde qui serait d'une taille colossale si on en écrivait toutes les étapes.

Même lorsqu'il s'agit de tester des conjectures, le mathématicien expérimentaliste doit faire preuve d'intelligence, et c'est en entremêlant raisonnements usuels et calculs confiés à la machine qu'il avance. Notons encore que dans l'exemple des nombres parfaits impairs les calculs pour obtenir les factorisations nécessaires aux étapes du raisonnement s'appuient sur des algorithmes résultants eux-mêmes de longues recherches utilisant des théorèmes ayant demandé des calculs informatiques pour leur mise au point.

Dans les mathématiques expérimentales, l'ordinateur apparaît parfois comme un simple exécuteur de corvées, dont les résultats viennent s'insérer à l'intérieur d'une preuve par ailleurs complexe. Deux cas sont célèbres de cet usage au sein d'un raisonnement délicat et difficile.
Il s'agit de la démonstration du théorème des quatre couleurs en 1976 (toute carte peut être coloriée avec quatre couleurs sans que deux pays voisins portent la même couleur) et de la conjecture de Kepler démontrée en 1998 (l'empilement le plus serré que l'on puisse

obtenir de sphères dans l'espace est celui utilisé pour faire les piles de boulets de canon ou les tas d'oranges).
Un bon assistant ne sert pas seulement à exécuter docilement des corvées simples qu'on pourrait faire soi-même à la main si on était près à y consacré des années ou des siècles. Il sera d'autant plus précieux qu'on pourra lui demander des tâches subtiles. Grâce aux logiciels de calcul formel et de démonstration automatique, il n'est pas rare qu'une partie délicate de démonstration ou un morceau difficile de calcul soit confié à l'ordinateur.
Aujourd'hui, on voit donc fréquemment des recherches où l'ordinateur intervient en fournissant son aide au moment de la découverte des nouveaux énoncés, étape suivie par une aide à la mise au point des démonstrations et éventuellement à leur contrôle (sans parler de l'aide qu'apporte l'ordinateur pour éditer les textes mathématiques, les imprimer et les faire circuler entre chercheurs).

L'ordinateur pour confirmer et contrôler

Signalons encore qu'un usage important de l'ordinateur semble en vue à cause justement de la complexité des preuves que l'interaction entre ordinateurs et mathématiciens engendre. La démonstration que Thomas Hales a mise au point de la conjecture de Kepler et dont certaines parties font intervenir des calculs informatiques a été publiée avec une mise en garde du comité d'experts chargé d'en fournir la garantie : le comité a indiqué qu'il ne pouvait pas assurer qu'aucune erreur n'était restée. Pour lever cette incertitude, Thomas Hales a entrepris de produire une version formalisée de sa démonstration, c'est-à-dire une version dont chaque pas est soigneusement explicité et peut être contrôlé mécaniquement. Bien sûr le travail de mise au point de la version formalisée se fait en s'aidant d'ordinateurs, et une fois que ce travail sera effectué, ce sera encore l'ordinateur qui sera chargé du contrôle final de la justesse de chaque pas de la preuve formalisée. Un tel travail d'écriture formelle d'une preuve associée à sa vérification a été mené pour le théorème des quatre couleurs par Georges Gonthier.
Cet usage des ordinateurs pour valider des démonstrations complexes est une forme nouvelle d'expérimentation mathématique. Elle s'ajoute aux nombreuses autres formes d'expérimentations en train d'envahir l'enseignement et la recherche mathématique.

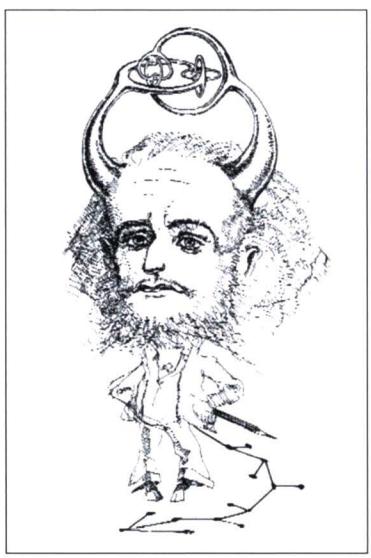
J.-P. D.

Bibliographie

- J. Borwein, D. Bailey, R. Girgensohn. Experimetation in Mathematics : Computational Paths to Discovery. Natick, MA, A. K. Peters, 2004.
- D. Bailey, J. Borwein, K. Devlin. The Experimental Mathematician : a Computational Guide to the Mathematical Unknown. Natick, MA, A. K. Peters, 2002.
- D. Bailey, J. Borwein, N. Girgenshn, Experimental Mathematics in Action, A. K. Peter, 2006.
- K. Chaurasya Experimental Mathematics, Campus Books International, 2012.
- G. Gonthier, Formal Proof, The Four-Color Theorem. Notices of the AMS 55 (11), 1382–1393, 2008.
- K. Hare. New Techniques for Bounds on the total number of primes factors of an odd Perfects number, Mathematics of Computation, 76:260, 2241–2248, 2007.
- P. Ochem, M, Rao,. Odd Perfect Numbers are Greater than 10^{1500}, Mathematics of Computation, 81, 1869–1877, 2012.

Les automates cellulaires et le jeu de la vie

Le *jeu de la vie* fait partie des *automates cellulaires*. Ces êtres virtuels sont nés dans les années 1960, de l'imagination fertile du mathématicien britannique John Horton Conway. Des cellules d'un quadrillage régulier vivent, meurent et naissent selon des règles définies à l'avance.



John Conway croqué par Simon Fraser en 1975.

Les automates cellulaires ont fait leur apparition dans les années 1940, suite à des travaux des mathématiciens Stanislaw Ulam et John von Neumann qui recherchaient des systèmes abstraits capables de se répliquer eux-mêmes.

Ils sont régulièrement utilisés pour modéliser les processus d'expansion des épidémies ou des épizooties. Ils sont aussi pertinents que leur définition est simple : des pions sont placés sur un damier infini, des règles de naissance, de survie et de mort sont précisées, et on laisse agir le processus. Dans le cas d'une application à la propagation d'une maladie, on suppose au départ que toutes les cases du damier contiennent des individus sains. On déclare ensuite que certains individus (ou cases) sont infectés, et on précise en plus une règle probabiliste de type : les cellules voisines de la cellule infectée sont elles-mêmes contaminées à l'étape suivante avec la probabilité p ; l'individu meurt ou est immunisé à l'étape suivante. La question qui intéresse alors autant les épidémiologistes que le grand public est alors : pour quelles valeurs de p la maladie se propage-t-elle au monde entier ?

Naissance, vie et mort

Une automate cellulaire peut se définir comme un ensemble de cellules pouvant prendre des états successifs (le nombre d'états possibles étant fini) en fonction des états des cellules voisines à chaque étape, le temps étant *discrétisé* en instants t, $t + 1$, $t + 2$, ... Les cel-

lules d'un automate cellulaire sont les cases d'un réseau régulier dans un espace à n dimensions, le cas le plus connu et le plus étudié étant celui d'un quadrillage à mailles carrées dans le plan. On peut aussi concevoir des automates cellulaires dans un réseau à mailles triangulaires ou hexagonales ou dans le pavage régulier d'un espace à trois dimensions ou davantage.

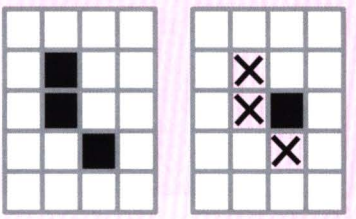
Ainsi, pour définir un automate cellulaire, il faut un réseau de cases (les cellules) pouvant prendre un nombre fini d'états selon une règle précisant le devenir de chaque état d'une cellule à l'instant $t + 1$ en fonction de son état et de celui de ses cellules voisines à l'instant t. Les mathématiciens Stephen Wolfram et David Eppstein ont tenté de proposer une classification des automates cellulaires.

L'étude d'un automate cellulaire consiste à partir d'une configuration initiale donnée et à observer son devenir. La population va-t-elle s'éteindre et disparaître, s'étendre indéfiniment, converger vers une configuration particulière constante ou périodique, fixe ou en déplacement, ou encore devenir chaotique ? Les automates cellulaires peuvent être considérés comme des *systèmes dynamiques* discrets. L'étude de ces systèmes s'est considérablement développée, on le comprend aisément, avec l'avènement de l'informatique qui permet une approche expérimentale.

Le plus connu des automates cellulaires est « le jeu de la vie ». John Horton Conway l'a introduit dans les années 1960, en en précisant les règles de naissance, de survie et de mort.

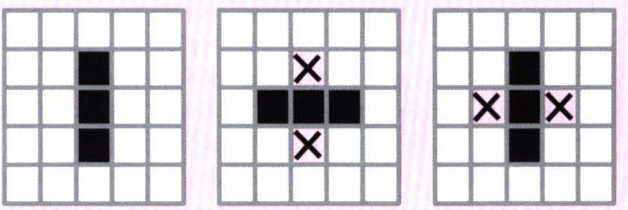
En octobre 1972, Martin Gardner consacre sa rubrique du *Scientific American* au jeu de la vie de Conway et l'engouement pour ce jeu prend un essor extraordinaire au tout début du développement de l'informatique.

Quelques exemples d'évolutions de populations au jeu de la vie



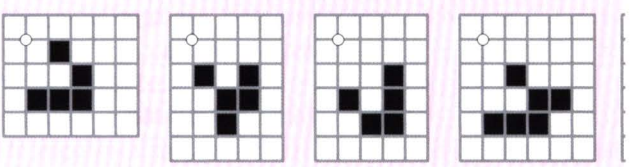
• Dans le premier exemple, on part d'une population de trois cellules. Deux de ces cellules ont un seul voisin et la troisième a deux voisins. Toutes les trois vont donc mourir. Mais une cellule vide possède trois voisins (avant leur mort). Cette cellule va donc voir une naissance. À l'instant 2, notre population sera donc éteinte.

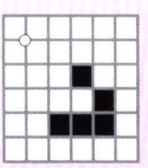
• Prenons maintenant l'exemple de trois cellules formant un rectangle.



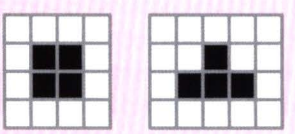
On constate que le rectangle va être alternativement horizontal, puis vertical, et que l'on a affaire à une transformation de période 2 : la population va prendre deux états. On l'appelle parfois un pulsar.

• Voici maintenant un autre type de population, appelé glisseur, qui revient périodiquement à sa forme initiale mais en se déplaçant ou en glissant selon un vecteur défini.





• Voici pour finir deux questions faciles laissées à votre sagacité : comment les deux configurations ci-dessous vont-elles évoluer ?



La coquille du conus textile présente des motifs complexes qui sont similaires à certains automates cellulaires.

Le jeu de la vie, comme tous les automates cellulaires, est un jeu entièrement déterministe. Une population de départ constituée d'êtres monocellulaires (les cellules), va vivre, donner naissance à de nouvelles cellules vivantes et parfois mourir selon des règles intangibles. En quoi s'agit-il alors d'un jeu ? La seule intervention du joueur consiste à créer l'univers de ce jeu : la grille destinée à recevoir les cellules, les règles de vie, de naissance et de mort de ces cellules, puis à placer dans cet univers des populations et à observer leur devenir, en n'ayant plus aucune prise sur ce devenir. On peut également procéder à une analyse rétrograde et rechercher les configurations « parentes » (les antécédents) d'une configuration donnée, certaines configurations n'ayant d'ailleurs pas d'antécédent (on appelle ces populations des « edens »).

Les règles du jeu de la vie

Le jeu de la vie fonctionne sur un quadrillage régulier formé de carrés unitaires (ou cellules) pavant le plan. On pourrait tout aussi bien le généraliser à un plan pavé de triangles ou d'hexagones réguliers, ou encore dans un pavage de l'espace à trois dimensions, mais sa version originelle offre déjà un vaste terrain d'exploration et d'amusement. Des créatures vont vivre et mourir dans ces cellules. Le temps dans lequel elles vivent n'est pas le nôtre : il s'agit d'un temps discret, qui s'écoule en « étapes » successives. On passe de l'instant 0 à l'instant 1, puis à l'instant 2, et ceci sans aucune transition. Le quadrillage est constitué de cellules qui peuvent être habitées, ou non, par des êtres vivants, à raison d'au plus un par cellule. Une cellule habitée est dite *vivante* (sur les diagrammes, elle est coloriée), et une cellule vide est *morte* (elle n'est pas coloriée). Chaque cellule est entourée par huit cellules voisines, deux cellules voisines ayant un côté ou un sommet en commun. Si, à l'instant n, une cellule est vivante, alors elle le reste à l'instant $n + 1$ si, et seulement si, deux ou trois des cellules voisines sont vivantes. Dans tous les autres cas, la cellule meurt à l'instant suivant. Si, à l'instant n, une cellule est n'est pas vivante, elle s'anime (ou elle naît) à l'instant $n + 1$ si, et seulement si, exactement trois des cellules voisines sont vivantes. Dans les autres cas, la cellule demeure vide. En particulier, une cellule isolée ne peut survivre. Une population de départ constituée d'êtres monocellulaires va vivre (et parfois mourir) selon ces règles intangibles. Les morts et les naissances se produisent simultanément, et, juste avant de mourir, une cellule peut participer à la naissance d'une autre.

L'étude du jeu de la vie de Conway peut conduire à de multiples questions, parfois difficiles : trouver des *edens* ; trouver des populations de période 2 (comme le *pulsar*, voir encadré), de période 3, de période 4, *etc.* ; explorer d'autres règles de survie, de mort et de naissance des cellules ; explorer d'autres type de pavage du plan : triangulaire, hexagonal.

Vous allez le constater, la richesse et la complexité des automates cellulaires valent bien celles de la vie elle-même !

M. C.

Références

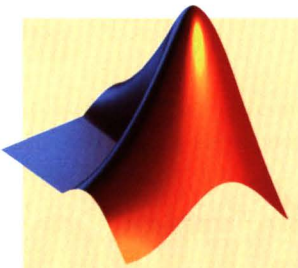
- *Les algorithmes*. Bibliothèque Tangente 37, 2013.
- *Théorie des jeux*. Bibliothèque Tangente 46, 2013.
- *Wheels, life and other mathematical amusements*. Martin Gardner, Freeman, 1983.

Au commencement, les matrices...

Matlab est un logiciel de calcul numérique interactif développé par la société américaine MathWorks. Utilisé aujourd'hui par plusieurs millions d'ingénieurs et de chercheurs, il est issu, au milieu des années 1970, du projet pédagogique d'un professeur de mathématiques, Cleve Moler, soucieux d'aider ses élèves à faire du calcul matriciel sans connaissance préalable de Fortran (Matlab est la contraction de *Matrix Laboratory*). Matlab est donc, nativement, optimisé pour le traitement des matrices, qui en sont d'ailleurs les variables par défaut. Son langage, destiné initialement à des étudiants, est simple, intuitif et plus concis que les langages historiques Basic, C, Fortran ou Pascal.

Si Matlab est d'abord un langage, c'est aussi un environnement de développement doté d'une interface graphique puissante pour l'affichage des courbes et des données, ainsi que de nombreuses boîtes à outils (fonctions dédiées à des applications spécifiques, comme le traitement du signal, l'analyse statistique, l'optimisation...). Outre une utilisation standard en algèbre linéaire, le logiciel permet ainsi l'analyse et la visualisation de données, la modélisation et la simulation de systèmes dynamiques, l'optimisation numérique, la mise en œuvre et le test d'algorithmes, et trouve des applications aussi bien en traitement du signal qu'en statistique ou encore en traitement des images.

Les trois fondateurs de MathWorks (Cleve Moler et les ingénieurs Jack Little et Steve Bangert) pouvaient-ils imaginer, lors de la commercialisation, en 1984, de la toute première version de Matlab, que leur logiciel deviendrait non seulement un outil incontournable pour la recherche, mais serait de plus en plus utilisé et prisé par le monde de l'industrie et de la finance ?



Matlab pour l'enseignement

Les dernières versions de Matlab permettent aux utilisateurs de créer des applications prêtes à l'emploi et de les empaqueter automatiquement afin de les partager facilement *via* la plateforme communautaire de MathWorks (www.mathworks.fr/matlabcentral). Autre signe d'une volonté d'interopérabilité accrue, les possibilités d'interfaçages avec d'autres langages comme le C++ ou Java.

Au-delà de l'usage dans la recherche et l'industrie, cet environnement constitue également un outil novateur pour l'enseignement. En sciences du vivant par exemple, le recours à Matlab peut permettre d'aborder la problématique de la bio-informatique et des systèmes biologiques à travers l'analyse de séquences génomiques, d'arbres phylogénétiques, de structures protéiniques, *etc*.

Matlab se veut aussi le partenaire privilégié du baccalauréat « Sciences et technologies de l'industrie et du développement durable » (STI2D). Certaines ressources pédagogiques exploitent ainsi les capacités de simulation et de modélisation du logiciel pour proposer aux élèves l'étude de systèmes réels, comme la lampe solaire Mona, développée par la jeune entreprise Solar 21.

par Jean-Christophe Filliâtre

Démonstration
L'ordinateur à la rescousse

Une intelligence artificielle comparable à celle de l'être humain reste encore à ce jour hors de notre portée. Pour autant, l'ordinateur peut nous aider à mener bien des raisonnements, parfois même très complexes.

Puzzle Artifact illustrant le théorème des quatre couleurs (design : Tara Flannery).

L'ordinateur est encore loin des performances humaines en termes d'intelligence, mais il a des atouts que l'être humain ne possède pas : une capacité de calcul incomparable, une docilité totale et une fiabilité sans faille dès lors que la mission qui lui a été confiée l'a été avec rigueur.
Il semble alors normal que le mathématicien cherche à s'appuyer dessus pour l'aider dans ses démonstrations, surtout celles dont il a du mal à se tirer tout seul. Dans ce but, il a donc imaginé plusieurs outils informatiques.

L'assistant de preuve

L'un des outils les plus notables de la démonstration assistée par ordinateur est l'assistant de preuve. Un assistant de preuve est un logiciel qui vérifie les étapes d'un raisonnement logique. Il

Il n'est pas possible d'écrire un programme qui prend en entrée un énoncé mathématique et qui détermine, en un temps fini, si cet énoncé est vrai ou faux.

peut également effectuer des calculs, numériques ou symboliques. L'un des plus utilisés est le logiciel Coq, développé par des chercheurs d'Inria depuis 1984. L'utilisateur interagit avec Coq pour introduire des définitions, énoncer des théorèmes et construire des preuves. Le système vérifie alors la validité de ces divers éléments.

Le logiciel Coq a été utilisé pour vérifier des démonstrations de grande ampleur. Ainsi on peut citer la preuve du théorème fondamental de l'algèbre (tout polynôme non constant à coefficients dans $\mathbb{C}$ admet au moins une racine) en 2000 par une équipe de chercheurs de Nimègue (Pays-Bas) ou encore la preuve du théorème des quatre couleurs (toute carte planaire peut être coloriée avec seulement quatre couleurs sans que deux pays ayant une frontière commune soient de la même couleur) en 2005 par deux chercheurs de Microsoft Research et Inria (Georges Gonthier et Benjamin Werner). Il existe d'autres d'assistants de preuve, développés dans des laboratoires de recherche du monde entier : PVS (SRI, Californie), HOL-Light (Intel, Oregon), Isabelle (TUM, Munich)…

La démonstration automatique

Deuxième outil : le démonstrateur automatique. C'est un programme qui prend en entrée un énoncé mathématique et tente d'en établir une preuve automatiquement. En cas de succès, on a la garantie que l'énoncé est vrai. En cas d'échec, en revanche, il se peut que l'énoncé soit tout de même vrai mais que le logiciel n'a pas su en trouver une preuve.

On a montré qu'il n'est pas possible d'écrire un programme qui prend en entrée un énoncé mathématique et qui détermine, en un temps fini, si cet énoncé est vrai ou faux. En quelque sorte, c'est là une assurance anti-chômage pour les mathématiciens. Il n'y a pas de contradiction pour autant à chercher à développer des démonstrateurs automatiques. Il faut seulement être conscient du fait qu'un démonstrateur automatique pourra toujours ne pas terminer ou répondre « je ne sais pas », y compris sur des énoncés vrais.

Coq en action

Voici un exemple d'utilisation de Coq. On commence par définir la propriété « être un multiple de 5 » pour un entier relatif x.

Definition mult5 (x: Z) := exists k: Z, x = 5 * k.

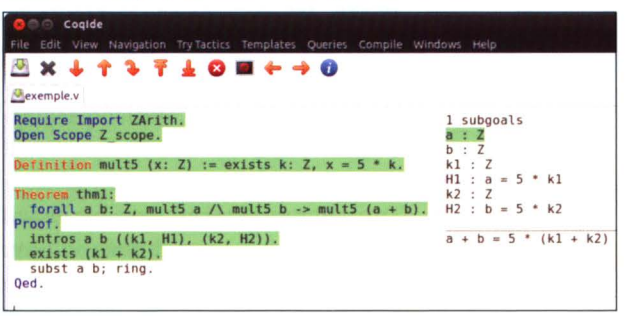
Ici le symbole Z dénote l'ensemble des entiers relatifs, fourni par la bibliothèque de Coq. La syntaxe (x: Z) peut se lire comme « x est un entier relatif ». On peut ensuite énoncer un théorème stipulant que si deux entiers a et b sont multiples de 5, alors leur somme l'est également.

Theorem thm1: forall a b: Z,
mult5 a ∧ mult5 b ! mult5 (a + b).

Ici « thm1 » est le nom que l'on a choisi de donner à ce théorème, par exemple pour y faire référence plus tard.

Pour procéder à la preuve de ce théorème, on commence par la commande « intros » qui permet de séparer les hypothèses (a et b sont des entiers multiples de 5) et la conclusion à prouver ($a + b$ est multiple de 5). On poursuit la preuve avec la commande « exists » qui nous permet de donner l'entier k justifiant que a + b est multiple de 5, c'est-à-dire tel que a+b est égal à $5*k$.

On peut terminer la preuve automatiquement avec la commande « ring », car il ne reste que du calcul.



La figure montre l'interface graphique du logiciel Coq, avec notamment les commandes saisies par l'utilisateur dans la partie gauche et le but à prouver dans la partie droite.

Un exemple de démonstrateur automatique est le logiciel Alt-Ergo, développé par des chercheurs de l'Université Paris Sud depuis 2007. Dans la syntaxe d'Alt-Ergo, on peut énoncer que (la partie entière de) la moyenne de deux entiers relatifs est toujours comprise entre ces deux entiers avec la syntaxe suivante :

```
goal thm2: forall x, y: int.
x <= y -> x <= (x+y)/2 <= y
```

Il ne faut que quelques centièmes de seconde à Alt-Ergo pour démontrer un tel énoncé.

De nombreux démonstrateurs automatiques sont développés de par le monde depuis plus d'un demi siècle. En 1996, une conjecture de la théorie des groupes qui résistait aux mathématiciens depuis 1933 a été prouvée par le démonstrateur automatique EQP, après huit jours de calcul.

La preuve de programmes

Troisième outil : il est possible de faire la preuve qu'un programme informatique est correct, c'est-à-dire qu'il fait bien ce qu'il est censé faire, en exprimant cela comme un énoncé mathématique. Plus précisément, un outil prend en entrée un programme, ainsi qu'une propriété que ce programme doit vérifier, et produit en sortie un ensemble d'énoncés mathématiques qui, s'ils sont prouvés, garantissent la *correction** du programme. C'est notamment là que la démonstration assistée par ordinateur, dans les deux acceptions précédentes, prend tout son sens, car de telles preuves peuvent être gigantesques et sont en général extrêmement fastidieuses.

* Le fait d'être correct.

Un exemple significatif de preuve de programme est celui de la preuve d'un compilateur C, réalisée en 2008 par un chercheur d'Inria (voir http://compcert.inria.fr). Un autre exemple est celui de la preuve d'une partie du logiciel de la ligne 14 du métro parisien, par la société Matra en 1998.

Un des outils les plus utilisés en France de preuve de programmes est le logiciel Why3, développé à l'Université Paris Sud depuis 2001. Il peut être utilisé conjointement avec de nombreux assistants de preuve, dont Coq, et de nombreux démonstrateurs automatiques, dont Alt-Ergo.

Considérons le programme **expo** figurant dans l'encadré ci-dessous et utilisons un logiciel comme Why3 pour montrer que ce programme élève bien x à la puissance n.

On commence par énoncer précisément la propriété que l'on souhaite vérifier. On appelle cela *spécifier* le programme. Comme il s'agit ici d'une fonction, cette spécification a deux composantes : une propriété attendue à l'entrée de la fonction, appelée *pré-condition*, et une propriété attendue à la sortie de la fonction, appelée *post-condition*. Ici la pré-condition stipule que $n \geq 0$ et la post-condition que le résultat est égal à x^n. L'ensemble de la pré-condition et de la post-condition forme ce que l'on appelle le *contrat* de la fonction.

Même si cet exemple est très simple, la spécification est une étape importante et parfois difficile. En particulier, le lecteur doit pouvoir se persuader que c'est bien là la propriété que l'on souhaite pour le programme. Une erreur peut facilement se glisser dans l'énoncé et la machine ne nous aidera pas.

On passe alors à l'étape de preuve. À ce stade, il faut aider l'outil Why3 en lui donnant une indication, sous la forme

d'une propriété qui reste vraie à chaque tour de boucle du programme. On appelle cela un *invariant de boucle*. Plus précisément, on indique que la propriété $r \times p^e = x^n$ doit être vérifiée chaque fois que le programme atteint la ligne 5 (quel que soit le résultat du test $e > 0$). L'invariant de boucle est un peu l'analogue de l'hypothèse de récurrence en mathématiques.

L'outil Why3 ne prend pas cet invariant de boucle pour argent comptant. Au contraire, il va exiger de nous de montrer qu'il s'agit bien là d'une propriété vraie à chaque tour de boucle, en supplément de la propriété finale que nous cherchons à montrer. Plus précisément, l'outil Why3 va nous demander de prouver les trois propriétés suivantes :

- que l'invariant de boucle est vrai initialement, c'est-à-dire quand on atteint la ligne 5 du programme pour la première fois. Ceci revient à montrer $1 \times x^n = x^n$, ce qui est immédiat.
- que l'invariant de boucle est maintenu par une exécution du corps de la boucle. Cela revient à supposer $e > 0$ (le test de la boucle est positif puisque le corps est exécuté) et l'invariant $r \times p^e = x^n$, et à montrer que l'invariant est toujours vrai après l'exécution des trois instructions lignes 6, 7 et 8. Il y a là deux cas de figure, selon que le test e **mod** 2 = 1 est ou non positif. Si oui, c'est-à-dire si e est impair, il faut montrer :
  $r \times p \times (p \times p)^{(e-1)/2} = x^n$ (car e **div** 2 désigne ici la partie entière de la division de e par 2).
  Sinon, c'est-à-dire si e est pair, il faut montrer $r \times (p \times p)^{e/2} = x^n$. Dans les deux cas, un petit peu d'algèbre suffit.
- enfin, que la post-condition est satisfaite quand on atteint l'instruction **renvoyer** à la fin du programme, c'est-à-dire que $r = x^n$. Vu que $e = 0$ à la sortie de la boucle, l'invariant $r \times p^e = x^n$ se simplifie en la propriété voulue.

Le programme expo calculant x^n

```
fonction expo(x, n) =
    r ← 1
    p ← x
    e ← n
    tant que e > 0 faire
        si (e mod 2) = 1 alors r ← r × p
        p ← p × p
        e ← e div 2
    renvoyer r
```

L'outil Why3 produit ces divers énoncés dans le format d'entrée des outils Coq et Alt-Ergo, qui peuvent alors être utilisés pour obtenir une preuve complètement mécanisée de la correction du programme **expo**. Une fraction de seconde suffit pour rejouer une telle preuve.

Il convient enfin de montrer que notre programme s'exécute en un temps fini. Pour cela, on majore le nombre de tours de la boucle **tant que**, par exemple par l'entier e. L'outil Why3 exige alors de nous de montrer que la quantité e reste toujours positive ou nulle et qu'elle décroît strictement à chaque tour de boucle, ce qui ne pose aucune difficulté. C'est bien entendu un majorant grossier du nombre de tours de boucle de ce programme, mais cela suffit à en garantir la terminaison.

J.-C. F.

Références

On trouvera plus de détails sur les outils Coq, Alt-Ergo et Why3 sur les sites suivants :

- http://coq.inria.fr
- http://alt-ergo.lri.fr
- http://why3.lri.fr

par Alain Valette

Espaces de Banach
et informatique théorique

Un problème d'analyse difficile peut être reformulé en théorie des graphes et se trouver lié à certains concepts de l'informatique théorique. Ce phénomène fréquent est une manifestation forte de l'unité des mathématiques !

Les termes suivis d'un astérisque sont définis en encadré, dans le glossaire.

L'émergence de l'ordinateur, dans la seconde moitié du XX^{e} siècle, a induit le développement d'une nouvelle branche des mathématiques, l'informatique théorique, et donc aussi l'apparition d'une nouvelle espèce de scientifiques, les « informaticiens théoriciens », qui ont la particularité d'être aussi à l'aise devant un clavier qu'au tableau noir. Il y a des aspects de l'informatique théorique que l'on imagine bien : théorie des graphes, algorithmique, théorie de la complexité… Il est peut-être plus surprenant de voir des informaticiens se passionner pour des problèmes d'analyse fonctionnelle liés à la géométrie des espaces de Banach* !

Une notion subtile : la distorsion

Au départ se trouve la notion d'espace métrique* fini (X, d). Une façon de la comprendre est d'essayer de plonger X dans un espace de Banach* B bien connu, par exemple un espace euclidien de grande dimension. Le prix à payer est qu'une application injective $f : X \to B$ n'a aucune raison de préserver les distances. Pour mesurer comment f déforme les distances entre X et B, on introduit la *distorsion* de f, un nombre dans $[1 ; \infty[$ défini par :

$$dist(f) = \max_{(x,y)} \frac{\|f(x) - f(y)\|_B}{d(x,y)} \times \max_{(x,y)} \frac{d(x,y)}{\|f(x) - f(y)\|_B}.$$

La distorsion de f est un invariant très robuste : il est invariant par exemple par les homothéties de B. Pour avoir un invariant ne dépendant que de X, on prend la borne inférieure sur toutes les applications injectives telles que f, et on définit la *distorsion* de X à valeurs dans B :

$$c_B(X) = \inf_{f : X \to B} dist(f).$$

En 1986, dans un article qui fonde la théorie non linéaire des espaces de Banach, le mathématicien belge Jean Bourgain (né en 1954, médaille Fields 1994) montre que, si B est l'espace $L^p([0 ; 1])$, avec $1 \leq p < +\infty$, il existe une constante $C(p) > 0$ (ne dépendant que de p) telle que :

$$c_{L^p([0;1])}(X) \leq C(p) \ln|X|.$$

Autrement dit, il existe une application injective $X \to L^p$ dont la distorsion est au plus logarithmique en le nombre de points de X.

Passons maintenant à la théorie des graphes : si X est un graphe fini, d'en-

semble de sommets V et d'ensemble d'arêtes E, et S est une partie de V, le *bord* de S, noté δS, est l'ensemble des arêtes connectant S à son complémentaire V – S. C'est donc l'ensemble des arêtes qu'il faut enlever pour déconnecter S de V – S. Pour cette raison, la paire {S ; V – S} s'appelle une *coupure* du graphe X.

La *constante de Cheeger* de X est :

$$\Phi^*(X) = \min_{\substack{S \subset V, \\ 0<|S|\leq|V|}} \frac{|\delta S|}{|S| \times |V - S|}.$$

Le problème consistant à calculer $\Phi^*(X)$ est connu sous le nom de *problème de la coupure de densité minimum* (ou *sparsest cut problem*). C'est un problème NP-difficile, ce qui signifie que l'on ne connaît essentiellement pas de meilleure solution, pour calculer $\Phi^*(X)$, que d'énumérer les $2^{|V|} - 2$ parties S apparaissant dans la définition. D'où un temps de calcul exponentiel en |V|.

On va se contenter d'une solution approximative, qui approcherait $\Phi^*(X)$ à une constante multiplicative près. Une première solution a été fournie en 1995 par les Israéliens Nati Linial, Eran London et Yuri Rabinovitch. Ils donnent d'abord une réinterprétation de Φ^* en termes de plongements de X (vu comme espace métrique*, avec les arêtes de longueur 1) dans L^1. Une distance* d sur X est une distance-L^1 s'il existe une application injective $f : X \to L^1$ avec $d(x, y) = \|f(x) - f(y)\|_1$, normalisé par :

$$\sum_{x \in V} \sum_{y \in V} d(x, y) = 1.$$

Les trois mathématiciens ont démontré que

$$\Phi^*(X) = \inf_{d \text{ distance-}L^1 \text{ sur } X} \sum_{(x,y) \in E} d(x, y).$$

Leur idée est de relaxer le problème en oubliant la condition L^1, c'est-à-dire en travaillant avec toutes les métriques sur X. On définit ainsi une constante $\Phi_{PL}(X)$, dont le calcul est un problème de programmation linéaire, qui se résout en temps polynomial avec une précision arbitraire. On a clairement $\Phi_{PL}(X) \leq \Phi^*(X)$, mais un risque existe, évidemment : faire une erreur gigantesque en remplaçant $\Phi^*(X)$ par $\Phi_{PL}(X)$. C'est précisément le résultat de Jean Bourgain qui nous assure que $\Phi_{PL}(X)$ est une bonne approximation de $\Phi^*(X)$! En effet, ce résultat implique

$$\Phi_{PL}(X) \leq \Phi^*(X) \leq C \ln |X| \, \Phi_{PL}(X)$$

Ce résultat remarquable a provoqué une série d'interactions passionnantes impliquant informaticiens, analystes, théoriciens des graphes, théoriciens des groupes : une manifestation de plus de l'unité des mathématiques !

A. V.

Petit glossaire des espaces de Banach

Espace de Banach : espace vectoriel sur ℝ ou ℂ muni d'une norme* pour lequel il est complet*.

Norme sur un espace vectoriel : application qui, à tout vecteur V, fait correspondre un nombre positif ||V|| qui vérifie certaines propriétés, comme $||k.V|| = |k| \times ||V||$ ou $||V + W|| \leq ||V|| + ||W||$.

Espace métrique : ensemble muni d'une distance d vérifiant certaines propriétés, en particulier l'inégalité triangulaire : $d(A, C) \leq d(A, B) + d(B, C)$. Tout espace normé est un espace métrique : on pose pour cela $d(V, W) = ||W - V||$, mais il n'est pas forcément complet.

Espace métrique complet : dans un espace métrique complet, toute suite de Cauchy* est convergente. Ainsi, ℝ ou $\mathbb{R}^n$, munis par exemple d'une norme euclidienne, sont des espaces de Banach, mais pas ℚ, qui n'est pas complet.

Suite de Cauchy : suite $(u_n)_n$ pour laquelle la distance $||u_m - u_n||$ entre deux éléments tend vers 0 quand leurs indices tendent vers l'infini. ℚ n'est pas complet car une suite de Cauchy de ℚ peut ne pas converger dans ℚ mais dans ℝ (par exemple, la suite décimale de π est une suite de Cauchy de ℚ qui ne converge pas dans ℚ).

Espaces L^p : on appelle $L^p(I)$ l'espace des fonctions dont la puissance p est intégrable sur I (au sens de Lebesgue). L^∞ est l'espace des fonctions bornées.

par Jean-Paul Delahaye

Le problème fondamental de l'informatique théorique

P est-il égal à NP ?

Le problème découvert par Gödel paraissait facile ; pourtant, il résiste depuis cinquante ans, et on a surtout compris qu'il ne fallait pas s'attendre à en trouver rapidement la solution. De quel problème fondamental s'agit-il ?

Les problèmes combinatoires classiques (« savoir si un mot contient un sous-mot donné », « savoir si un chemin donné est le plus court chemin reliant A et B dans un graphe », « savoir si l'entier N est un carré parfait »…) se traitent parfois rapidement, ou à l'inverse demandent beaucoup de calculs. Les classes de complexité définies en informatique théorique servent à les ranger en catégories. C'est étrange, mais on ignore si les deux principales classes, P et NP, sont égales !

La classe P

Savoir si les nœuds d'un graphe donné ayant n nœuds sont coloriables à l'aide de deux couleurs (par exemple bleu et rouge) sans que deux nœuds liés l'un à l'autre portent la même couleur est facile. On obtient la réponse rapidement par la méthode suivante. On choisit un nœud, que l'on colorie en rouge, on colorie tous les nœuds qui lui sont liés en bleu, on colorie tous les nœuds liés à un nœud bleu en rouge, et on poursuit ainsi de proche en proche en alternant les couleurs ; quand il existe plusieurs composantes connexes au graphe, on procède de la même façon pour chaque composante. Si l'on rencontre une impossibilité, c'est qu'aucun coloriage bicolore n'est possible, car tous les coloriages faits après le premier sont inévitables. Si on aboutit, c'est que la réponse est oui.

Aucun retour en arrière n'est nécessaire dans l'utilisation de la méthode (les nœuds une fois colorés ne changent plus de couleur) et donc la méthode de coloriage prend un « temps » (c'est-à-dire un nombre d'étapes) proportionnel en gros au nombre de nœuds, n. On dit que le problème de la 2-coloriabilité est *polynomial* (ou *appartient à la classe polynomiale*).

Certains problèmes de décision (la réponse doit être « oui » ou « non ») ne peuvent être résolus qu'en un nombre d'étapes majoré par n^2 (ou par toute autre puissance de n, n mesurant la taille des données). On considère encore que ce sont des problèmes « efficacement traitables » et ils constituent la classe P des problèmes que l'on peut résoudre en temps polynomial.

Bien évidemment, un problème demandant un nombre d'étapes de l'ordre de n^4 est plus difficile (en un sens) qu'un problème demandant un nombre d'étapes

de l'ordre de n^2. Cependant, pour une première analyse, on les considère tous les deux comme relativement faciles. Voici quelques exemples de problèmes de la classe de complexité P :

- Savoir si une suite de n entiers est rangée en ordre croissant.
- Savoir si un entier de n chiffres est un carré parfait.
- Savoir si un entier de n chiffres est un nombre premier (problème de la *primalité* : c'est seulement en 2002 que l'on a prouvé qu'il est dans P).
- Savoir si un mot contient un sous-mot donné.

Savoir si les nœuds un graphe possédant n nœuds sont coloriables à l'aide de trois couleurs (par exemple : bleu, rouge, jaune) sans que deux nœuds liés l'un à l'autre portent la même couleur est plus difficile qu'avec deux couleurs. On ne connaît aucune méthode polynomiale (c'est-à-dire demandant un temps de calcul majoré par un polynôme de la variable n) conduisant, de manière certaine, soit à la réponse oui, soit à la réponse non. On soupçonne qu'il n'existe pas de tel algorithme polynomial pour ce problème, mais on ne sait pas le prouver.
En revanche, il n'y a aucune difficulté à colorer les n nœuds selon une règle arbitraire, puis à examiner si cela convient. Si vous avez de la chance, vous trouverez une solution dès le premier essai et ce sera fini. Sinon vous recommencerez. Il y a 3^n tentatives à faire (car il y a n nœuds pouvant chacun prendre trois couleurs différentes). Lorsque vous les aurez toutes essayées en utilisant un procédé d'énumération systématique, soit vous aurez trouvé une solution (vous saurez que la réponse est « oui, le graphe est 3-coloriable »), soit vous n'en aurez pas trouvé (et vous saurez que la réponse est « non, le graphe n'est pas 3-coloriable »).
Si vous êtes comme Gontran, le personnage de Walt Disney à qui le hasard est toujours favorable, alors la méthode « essayer une fois au hasard et vérifier » est parfaite. Cette méthode vous donne la réponse en un temps proportionnel en gros à n. Si vous n'êtes pas Gontran, mais que vous disposez d'un ordinateur parallèle au parallélisme illimité, vous vous en sortirez aussi en un nombre d'étapes en gros proportionnel à n car vous lancerez 3^n tentatives en parallèle et donc saurez, aussi rapidement que Gontran, si le graphe est 3-coloriable ou pas.
On dit que le problème de la 3-coloriablité est un problème *de la classe NP* (pour *non déterministe, polynomial*) car, en ayant une chance parfaite et en menant un essai de manière non déterministe, ou si l'on dispose d'un ordinateur au parallélisme illimité, on le résout en temps polynomial. On ne sait pas, par contre, si ce problème est dans la classe P, car les seuls algorithmes déterministes (et non parallèles) que l'on connaît sont du type de celui décrit précédemment, qui procèdent par énumération et demandent un temps de travail exponentiel à un ordinateur non parallèle (ici 3^n essais).
D'une manière générale, on définit la classe NP comme la classe des problèmes de décision (la réponse est « oui » ou « non ») que l'on sait résoudre en temps polynomial si on a une chance parfaite : on utilise un algorithme dont le nombre d'étapes est majoré par un polynôme de la variable n (la taille du problème), qui fait des choix au hasard, et qui vérifie (une fois que les choix ont été faits) que c'est bon. Cela est équivalent à utiliser un algorithme lançant des calculs en parallèle (sans limitation), chacun d'eux ne travaillant qu'un nombre d'étapes majoré par un même polynôme de la variable n.

Quelques problèmes NP-complets

Plusieurs milliers de problèmes NP-complets sont connus et on en découvre chaque année de nouveaux. Si vous voulez tenter votre chance pour gagner un million de dollars, en voici une petite liste.

Problème du circuit hamiltonien
Un graphe G, de taille n, étant donné, peut-on suivre les arcs du graphe de façon à passer par tous les nœuds du graphe, sans passer deux fois par le même nœud et en revenant au point de départ ?

Problème du voyageur de commerce
Un graphe G, de taille n, étant donné avec un nombre sur chaque arc indiquant sa longueur, et un nombre M étant fixé, peut-on trouver un chemin du graphe ayant une longueur totale inférieure à M et passant par tous les nœuds du graphe ?

Problème du sous-graphe planaire
Un graphe G, de taille n, étant donné, ainsi qu'un entier k, peut-on trouver k nœuds du graphe G tels qu'en ne retenant que ces k nœuds et les arcs qui les relient on obtienne un graphe *planaire* (représentable sur un plan sans que deux arcs se coupent) ?

Problème des ensembles disjoints
Une famille finie d'ensembles finis F, de taille n, étant donnée, ainsi qu'un nombre k, peut-on trouver k ensembles dans la famille F qui soient disjoints deux à deux ?
Exemple : F = $\{\{a, b, c\}, \{a, e, c, f, g\}, \{d, e, f\}, \{a, c, e, i\}, \{c, f, i\}, \{g, h, i\}, \{b, f, i\}, \{j, k, l, m\}, \{b, g, h, i\}\}$ avec $k = 4$.
Réponse : oui, en considérant $\{a, b, c\}$, $\{d, e, f\}$, $\{g, h, i\}$, $\{j, k, l, m\}$.

La grande question

La question la plus fondamentale de l'informatique théorique est celle de savoir si P = NP. Autrement dit, ce que l'on peut faire en temps polynomial non déterministe quand on a une chance parfaite (classe des problèmes NP) peut-il *toujours* être fait en temps polynomial par un algorithme n'utilisant ni le hasard, ni le parallélisme (classe des problèmes P) ?
Preuve de son importance, le problème « P = NP ? » est l'un des sept problèmes que l'Institut Clay a sélectionnés en l'an 2000 (dont un seul a été résolu à ce jour). Comme pour les six autres, une somme d'un million de dollars attend celui ou ceux qui sauront le résoudre. Certains affirment que c'est le plus important des sept problèmes, et donc le plus important aujourd'hui en mathématiques ! Il est vrai qu'il est *a priori* le seul dont la résolution pourrait avoir des conséquences pratiques car des centaines de problèmes concrets sont concernés. Il est aussi celui dont la portée philosophique est la plus grande : il concerne la nature de ce qu'est la recherche de solutions dans un ensemble exponentiel de possibilités, ce qui est le problème même de la recherche scientifique. Dit en termes simples, la question « P = NP ? » signifie « est-ce que ce que nous pouvons trouver rapidement, lorsque nous avons de la chance, peut être trouvé rapidement par un calcul intelligent ? ». Sous forme très brève : l'intelligence peut-elle remplacer la chance ?
Une autre formulation encore de ce problème est : tout ce que l'on peut vérifier facilement peut-il être découvert facilement ? Vérifier qu'un chemin dans un graphe passe par tous les nœuds du graphe sans jamais passer deux fois par le même nœud (chemin *hamiltonien*) est facile, donc, si P = NP, savoir s'il existe des chemins hamiltoniens sera facile (on ne connaît pour l'instant aucun algorithme efficace qui le permet).

Tout problème de la classe P est également dans la classe NP. Appartenir à la classe NP n'est donc pas un gage de difficulté ! Un tel gage ne s'obtient qu'en considérant la classe des problèmes NP-complets.

Les problèmes vraiment difficiles

Certains problèmes de la classe NP concentrent en eux *toute* la difficulté de la classe NP, en ce sens que :

- savoir en résoudre *un seul* en temps polynomial permettrait de résoudre *tout* problème NP en temps polynomial,
- prouver qu'il est impossible d'en résoudre un seul en temps polynomial prouverait définitivement que P ≠ NP.

On les appelle les *problèmes NP-complets*. Cette notion fut introduite au début des années 1970 indépendamment par Leonid Levin en Russie (alors Union des républiques socialistes soviétiques) et Stephen Cook au Canada, qui prouvèrent chacun de leur côté qu'il existe effectivement des problèmes NP-complets, ce qui est loin d'être une évidence.

Le problème de la 3-coloriabilité est NP-complet (cela fut démontré en 1972 par Richard Karp). En conséquence, si vous découvrez un algorithme qui le résout en temps polynomial, vous aurez prouvé que P = NP. C'est bien sûr la voie la plus tentante pour résoudre l'énigme « P = NP ? ». Si vous *démontrez* qu'il n'existe pas d'algorithme polynomial pour ce problème, vous aurez *démontré* que P ≠ NP.

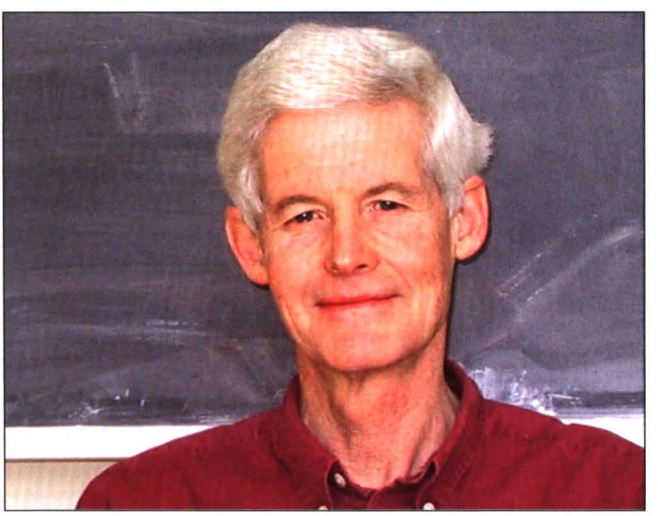
On connaît des problèmes de décision dont on a démontré qu'ils demandaient un temps de calcul exponentiel (par exemple savoir si un programme ou une machine de Turing s'arrête avant avoir fait n étapes de calcul). De tels problèmes appartiennent à la classe notée EXP, mais ne sont pas dans NP, ni ne sont NP-complets. Cependant, ces problèmes sont plus rares et, très souvent en algorithmique, on tombe sur des problèmes

Problème de la séparation équitable
Une suite finie de nombres entiers étant donnée, de taille n, peut-on la séparer en deux paquets ayant la même somme ?
Exemple : (1, 2, 2, 2, 3, 4, 4).
Réponse : oui, car 2 + 2 + 2 + 3 = 1 + 4 + 4.

Les équations quadratiques
Trois nombres entiers a, b et c étant donnés, peut-on trouver deux entiers x et y tels que $ax^2 + by = c$?

Problème du Sudoku généralisé
Au lieu de considérer des problèmes de Sudoku composés de neuf carrés de neuf cases regroupés en un grand carré de neuf lignes et neuf colonnes, on considère des problèmes composés de n^2 carrés de n^2 cases regroupés en un grand carré de n^2 lignes et n^2 colonnes avec les mêmes règles de remplissage. La question posée est : un énoncé étant donné, possède-t-il une solution ?

Le problème des mots croisés
Une liste finie de mots (un dictionnaire), D, étant donnée, ainsi qu'une grille de mots croisés de taille n^2 (c'est-à-dire une grille carrée vide avec quelques cases noircies), peut-on remplir la grille de mots croisés en utilisant des mots pris dans D ?



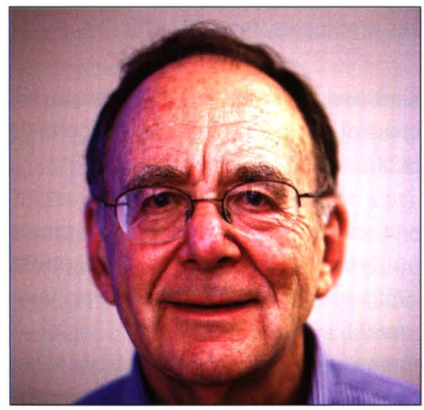
Stephen Arthur Cook (né en 1939).

Kurt Gödel interroge John von Neumann

Kurt Gödel, dans une lettre retrouvée parmi ses papiers et qu'il envoya à John von Neumann en 1956 quelques mois avant sa mort, est le premier à avoir formulé clairement la question « est-ce que P = NP ? » en insistant sur son importance concrète en mathématiques. Il y explique que si P = NP (sans employer cette notation, qui sera introduite en 1972) alors bien des questions mathématiques deviendront faciles par le procédé suivant. Pour résoudre une question ouverte Q, il suffira de rechercher parmi toutes les démonstrations de longueur n (n un entier fixé), dans un système donné d'écriture des démonstrations (par exemple dans celui de la théorie des ensembles), s'il y en a une conduisant à la réponse cherchée. S'il y en a une, on aura résolu le problème. Si on n'en trouve pas, et que le n essayé est assez grand, alors « il n'y aura plus de raisons sérieuses de rester préoccupé par le problème ».

L'indécidabilité algorithmique des systèmes logiques (c'est-à-dire l'affirmation qu'il n'existe pas d'algorithmes indiquant, en un temps fini, pour toute formule F, si elle est démontrable ou non dans un système fixé assez puissant) est un résultat négatif central en logique qui fut établi dans la décennie 1930 par Alonzo Church et Alan Turing. Pour Kurt Gödel, sa version concrète est l'affirmation P ≠ NP. On le voit, l'enjeu est capital.

La preuve que P = NP serait une surprise. Les chercheurs sont aujourd'hui à peu près tous persuadés qu'en vérité P ≠ NP (plus de 80 % de ceux qui ont un avis pensent que P ≠ NP). Il est étrange que, bien qu'en apparence très simple, la question résiste autant. Les recherches menées depuis plus de quarante ans à son sujet ont peu fait avancer vers la solution. Elles ne sont cependant pas restées totalement vaines, car à défaut de suggérer ce qu'il faut faire, elles donnent une meilleure compréhension des raisons des échecs et de l'inutilité de l'exploration de certaines voies.



Richard Manning Karp (né en 1935).

des classes P ou NP. C'est aussi pour cette raison que savoir si P est identique à NP est si important.

On entend dire parfois que le problème « P = NP ? » est, des sept problèmes récompensés par l'Institut Clay, celui le plus susceptible d'être résolu par un amateur. C'est exact, en ce sens que son énoncé est plus simple à comprendre que celui des autres problèmes et qu'il est envisageable (bien que peu probable…) qu'une solution élémentaire soit proposée demain par un génial passionné, par exemple en découvrant un algorithme polynomial pour un problème NP-complet. La situation était la même pour le grand théorème de Fermat, dont l'énoncé est compréhensible par tous. Cependant, comme on l'a vu, cela ne signifie pas que la solution était facile ! Pour le grand théorème de Fermat, d'ailleurs, c'est un professionnel qui a résolu l'énigme. Aujourd'hui, on a de sérieuses raisons de craindre que la question « P = NP ? » est d'une profonde et extrême difficulté.

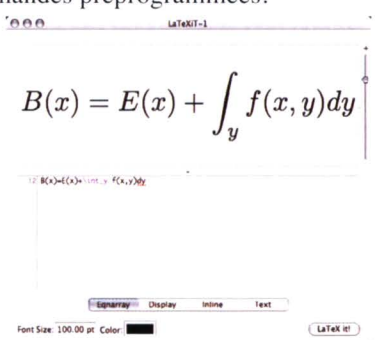
J.-P. D.

Références

- *Is P Versus NP Formally Independant?* Scott Aaronson, *Bulletin of the European Association of Theoretical Computer Science* 81, 2003.
- *The P versus NP Problem*. Stephen Cook, Clay Mathematics Institute, 2000 (disponible en ligne).
- *Les problèmes NP sont-ils si compliqués ?* Jean-Paul Delahaye, Dossier Pour La Science « Les grands problèmes mathématiques », 2012.
- *Mathématiques discrètes et combinatoire*. Bibliothèque Tangente 39, 2010.
- *P, NP and the NP-Completeness, The Basics of Computational Complexity*. Oded Goldreich, Cambridge University Press, 2010.

Latex

La composition des formules mathématiques est un cauchemar pour beaucoup d'utilisateurs de maths, de l'étudiant au chercheur, mais plus encore pour les éditeurs, en recherche permanente de synthétiseurs de formules. En 1977 le mathématicien et informaticien Donald Knuth crée le langage Tex (prononcez « Tek ») qui, après compilation, affiche la plupart des formules dont on peut avoir besoin. Quelques années plus tard, en 1983, Leslie Lamport intègre ce langage dans un logiciel de composition de pages qui prend le nom de LaTeX. Aujourd'hui, LaTeX est utilisé dans le monde entier, dans toutes les langues, sous tous les systèmes d'exploitation, et possède de nombreuses *extensions* semblables aux *bibliothèques* de *Maple* ou *Mathematica*, qui regroupent des commandes préprogrammées.



Un fichier LaTeX contient du texte où figurent des *commandes de marquage*. Il est ensuite converti grâce à un compilateur, qui permet une mise en page sobre, « spartiate » disent certains, mais avec une grande qualité des formules mathématiques.

Malheureusement, cette mise en page, suffisante pour des thèses ou des publications sans prétention graphique, ne peut convenir pour une mise en page sophistiquée ou en couleur, si ce n'est pour récupérer les formules sous forme d'images. Mais dans ce cas, il n'y a pas vraiment plus d'avantage à utiliser LaTeX que les autres éditeurs d'équations comme ceux qui sont intégrés dans les traitements de texte ou comme MathType.

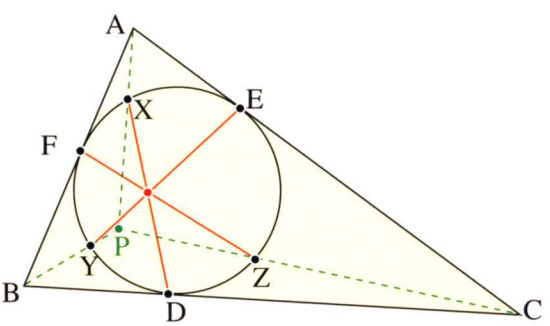
Cabri
pour vérifier une conjecture

Les logiciels de géométrie interactive sont une des manifestations importantes de l'apport de l'informatique aux mathématiques.

Si le plus connu d'entre eux, Cabri, s'adresse naturellement au monde de l'éducation (la société Cabrilog vient de plus de sortir Cabri Factory pour le collège), des chercheurs, amateurs ou professionnels, se le sont rapidement approprié pour en faire un outil de conjecture et de recherche dans le domaine de la géométrie élémentaire.

C'est ainsi que l'ingénieur et mathématicien américain Stanley Rabinowitz, connu pour son inépuisable base de problèmes (il a créé dans les années 1990 une société d'édition spécialisée dans le *problem solving* et un site internet intitulé « 20 000 problems under the sea »), a choisi un logo illustrant une propriété toute simple à énoncer qu'il a découverte en 1990 grâce au logiciel Cabri-géomètre.

Soit P un point quelconque intérieur à un triangle ABC. On trace les segments [PA], [PB] et [PC] qui coupent respectivement en X, Y et Z le cercle inscrit dans ce triangle. Ce dernier est tangent aux côtés du triangle en D, E et F (voir figure).



Alors, les droites (DX), (EY) et (FZ) sont concourantes, quel que soit le choix du point P.

Référence

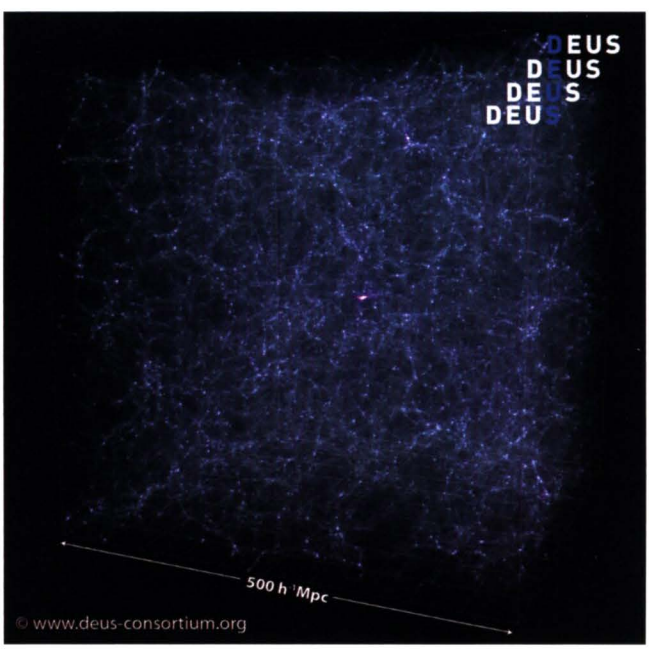
Stanley Rabinowitz, Problem 1364, *Mathematics Magazine* 64(1991).

par François Lavallou

La simulation numérique

Entre la théorie et l'expérience s'est glissée, depuis l'avènement de l'informatique, la simulation numérique. Elle est devenue un outil indispensable pour compenser les limites de l'expérimentation et tenter de représenter l'irreprésentable.

Où suis-je ? Que l'interrogation soit temporelle ou spatiale, l'apparente régularité des alternances jour nuit, la répétitivité des levers et couchers de soleil (bref, le *nycthémère*), la périodicité même des quartiers de lune semblent les signes objectifs d'une incroyable horlogerie astronomique. À défaut de comprendre, on a tous besoin de prédire son fonctionnement pour rythmer notre vie, que ce soit au jour le jour ou saison après saison. L'astrolabe, inventé par les Grecs, remplit cette fonction et est un bel exemple de la relation modélisation–simulation. Modèle de la sphère céleste par projection stéréographique, sa conception et son utilisation ne requièrent en aucun cas la connaissance du système héliocentrique. Il n'explique pas le monde, il le calcule. Il permet de prévoir, par une simple lecture, la position à venir des astres ou planètes. La précision de cette prévision dépend bien sûr de la qualité de réalisation de l'astrolabe qui détermine l'incertitude de lecture, mais surtout de la validité de ce modèle planisphérique.



Filaments entre amas de galaxies.

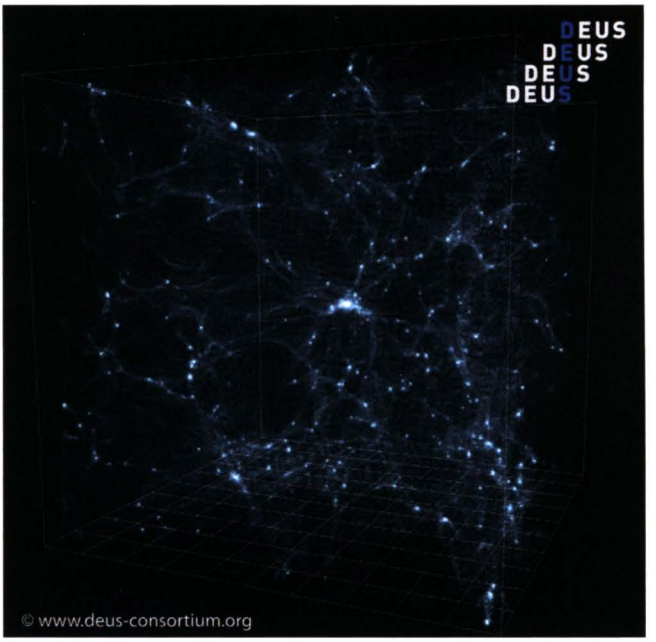
Un modèle à suivre

Méthodologiquement parlant, la science moderne est née à l'époque de Galilée. Dans la continuité de Pythagore, il cherche à expliquer le monde au moyen d'un langage, les mathématiques. Les lois physiques sont modélisées par des formules qui permettent de mettre en équation les mouvements des corps célestes (donc rapides). On résout analytiquement le mouvement de deux corps en attraction gravitationnelle, mais Poincaré démontre qu'il est vain de chercher une solution exacte pour un problème aussi simple (concep-

tuellement) que le problème des trois corps. Il est le premier à étudier la stabilité de systèmes dynamiques et à faire émerger la notion de chaos. Il était alors illusoire d'espérer prédire l'évolution du système solaire sur de « longs » temps ! Il faudra attendre la puissance calculatoire de l'informatique moderne pour pouvoir soumettre la stabilité du système solaire à l'expérimentation numérique.

La notion d'expérience numérique est un nouveau concept. Pythagore, en effectuant des expériences de reproduction des sons, faisait de la simulation. Tout laboratoire est un modèle d'univers, toute expérience est une simulation de la réalité, une réalité réduite dont on espère la représentativité acceptable. Quand on est sûr d'avoir incorporé dans le modèle tous les éléments nécessaires, on peut même s'affranchir de matérialiser l'expérience et effectuer alors des expériences de pensée, dont la puissance pédagogique a été merveilleusement illustrée par Albert Einstein. Mais quand on quitte notre domaine de connaissance pour aborder des échelles de temps ou d'espace auxquelles nous n'aurons sans doute jamais physiquement accès, seule une expérience numérique peut nous éclairer. Elle amplifie les possibilités de la recherche en permettant de comprendre les structures moléculaires aussi bien que les collisions de galaxie.

Ainsi, une équipe de l'Observatoire de Paris a utilisé le supercalculateur Curie du Genci (voir en pages suivantes) pour simuler la structuration de tout l'univers observable afin de comprendre la nature de la matière noire. Cette simulation DEUS (*dark energy universe simulation*) va permettre de comparer la prédiction de trois modèles cosmologiques de la matière noire aux observations sur la structure de l'univers aux grandes échelles. Elle montre en particulier la nécessité de l'existence d'une « matière noire » pour expliquer la vitesse expansionniste de l'univers primordial. Cette expérience illustre un rôle déterminant, et nouveau, de la simulation, qui induit des théories à partir des données d'observation. Elle peut permettre de déterminer, parmi les modèles théoriques proposés, celui qui fournit l'approximation la plus fiable, et potentiellement de l'optimiser.



Visualisation de simulations DEUS.

La méthode de Monte-Carlo

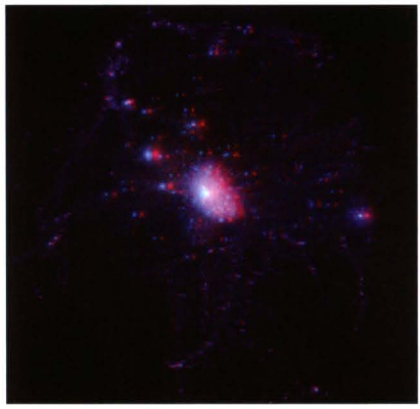
La méthode dite de Monte-Carlo a été conçue lors du projet Manhattan pour le transport neutronique. Au cours de sa vie hasardeuse, un neutron subit des évènements (émission, absorption, diffusion élastique ou inélastique...) dont les probabilités élémentaires sont connues. Comme pour un sondage d'opinion, le tirage au sort (d'où le nom de la méthode !) d'un nombre fini de neutrons sera représentatif de l'ensemble du flux neutronique (s'il est bien réalisé !). Le principe de cette méthode, dont la convergence a fait l'objet de nombreuses études mathématiques, s'applique bien sûr à de nombreux domaines (voir dans le dossier suivant).

L'informatique de la simulation

La précision des simulations numériques croît avec la finesse de discrétisation des objets étudiés. Pour gagner un facteur dix sur la résolution, il faut multiplier le nombre de mailles par mille ! Cette croissance exponentielle lie intimement le progrès des simulations numériques à celui des équipements informatiques. On est passé de quelques dizaines de mailles dans les années soixante à la dizaine de milliards de mailles aujourd'hui. À mailles égales, cette augmentation de performance a permis de passer des modèles sphériques à une dimension a des modèles à trois dimensions, plus proches de la réalité.

Les processeurs vectoriels, qui exécutent des opérations sur des vecteurs, sont particulièrement bien adaptés à la simulation numérique. Les performances d'un processeur actuel sont liées à sa fréquence de fonctionnement et à son *parallélisme interne* (sa capacité à effectuer des opérations simultanées). Le degré suivant est le *parallélisme entre processeurs*, dont la gestion est à la charge du programmeur. Le calcul doit alors être décomposé en tâches indépendantes, en veillant à maintenir la communication entre processeurs pour partager des résultats intermédiaires, et à l'équilibre des charges de travail de chaque processeur pour gagner en efficacité.

Plusieurs types d'architecture existent pour les supercalculateurs, dont les puissances se chiffrent aujourd'hui en téraflops (voir en pages suivantes). Les supercalculateurs vectoriels possèdent des processeurs haut de gamme, mais leur architecture rigide est peu évolutive. Les « grappes de PC » sont peu coûteuses, mais mal adaptées à une forte puissance de travail. Enfin, un excellent rapport performance / prix est offert par une structure intermédiaire : les « grappes de mini-ordinateurs à mémoire partagée ». Des briques de base – les nœuds – sont constituées de plusieurs microprocesseurs partageant une mémoire commune et sont reliées entre elles par un réseau d'interconnexion haute performance. Cette structure, la plus répandue parmi les supercalculateurs, est appelée *ordinateur massivement parallèle*.



La simulation numérique s'introduit donc dans le diptyque dialectique théorie–expérience, au cœur de la science classique, pour structurer la recherche moderne selon un triptyque théorie, modélisation–simulation et expérimentation. Elle permet bien sûr de limiter le coût et le danger par rapport aux expérimentations physiques, mais aussi de se donner les moyens de mieux comprendre, donc de mieux concevoir ou agir. Elle devient un guide conceptuel pour intuiter de nouvelles directions de recherche.

Les résultats de simulation numérique font maintenant partie de notre quotidien : prévisions météorologiques, prévisions démographiques, évolution du climat ou cours de la bourse.

Des mailles à répartir

La première étape d'une expérimentation numérique consiste à réunir les caractéristiques physiques du système à étudier dans un modèle établi avec des méthodes mathématiques et informatiques. Les grandeurs physiques recherchées (position, vitesse, température, pression…) et leurs variations sont liées par des équations, souvent aux dérivées partielles, qui modélisent le comportement de l'objet. Hors du champ analy-

tique, que l'on ne rencontre en fait que dans les exercices scolaires, le calcul est numérique et ne peut donc être effectué que sur un nombre fini de valeurs. Ce passage du monde infini de la continuité au monde des vecteurs de dimension finie s'appelle la *discrétisation*.

Deux familles de calcul se proposent pour résoudre les équations du modèle. La *méthode de Monte-Carlo* (voir dans le dossier suivant), qui est une méthode statistique, est particulièrement bien adaptée pour suivre les flux de particules, neutroniques, photoniques ou électroniques. Elle optimise le temps de calcul en s'intéressant aux particules qui contribuent le plus au signal (voir en encadré). Quant aux *méthodes déterministes*, leur principe a été établi bien avant l'avènement de l'informatique. Chaque point du domaine de calcul est associé à son voisinage pour constituer un volume élémentaire, dont la forme, qui pave l'espace, peut être un tétraèdre ou un hexaèdre. Ce maillage tridimensionnel est au cœur de la méthode dite *des volumes finis*, aisée à mettre en œuvre pour un calcul parallèle (voir en encadré). Il peut être fixe (maillage *lagrangien*) ou suivre l'écoulement d'un liquide (maillage *eulérien*), ou encore *adaptatif*, en ajustant la taille des mailles à la complexité des phénomènes physiques. L'étude de la sensibilité des résultats au type de maillage est équivalent à l'étalonnage pour un détecteur.

Effectuer un calcul revient donc à suivre le flot des valeurs des grandeurs physiques, d'une maille à l'autre, à travers leur surface commune, selon les principes contenus dans les équations du modèle. Les relations algébriques entre les paramètres d'une maille et ceux de ses voisines constituent un immense système linéaire dont la résolution, par l'analyse numérique, ne doit sa complexité qu'à sa taille.

Les résultats d'une expérimentation numérique doivent être confrontés à l'expérience, seul juge de paix en physique.

Les résultats d'une expérimentation numérique, une fois sélectionnés, extraits et visualisés, doivent ensuite être confrontés à l'expérience, seul juge de paix en physique. Ainsi, par exemple, c'est seulement par son incapacité à produire une expérience probante que la théorie des cordes tombe aujourd'hui en disgrâce.

F. L.

Références

- Les revues *Clefs CEA*, et notamment *Clefs CEA* 47 (2003), disponibles en ligne à cette adresse : www.cea.fr/le-cea/publications/les-clefs-du-cea/clefs-cea
- *Mathématiques et géographie*. Bibliothèque Tangente 40, 2011.
- *Les matrices*. Bibliothèque Tangente 44, 2012.
- *Le calcul intégral*. Bibliothèque Tangente 50, 2013.
- *Les angles*. Bibliothèque Tangente 53, bientôt disponible.

par J.-J. Dupas et F. Lavallou

Le calcul haute performance

Le besoin croissant de simulation numérique ne pourrait être assouvi sans les performances en continuelle évolution des technologies numériques. Le calcul intensif, ou haute performance, participe à ce développement de l'innovation et est devenu un enjeu majeur dans la course à la compétitivité.

Depuis que l'homme s'est mis à calculer, il n'a eu de cesse de chercher à produire plus de calculs. Utilisant tout d'abord de petits cailloux, les *calculi*, puis l'écriture, il comprit bien plus tard que certaines opérations pouvaient être mécanisées. Une des premières machines mécaniques est due à Pascal en 1645. Babbage a failli nous faire basculer dans un monde d'informatique mécanique dès le XIX^{e} siècle. Ce fut ensuite l'avènement de l'informatique moderne. Pendant ce temps, les besoins en calculs ont augmenté de façon exponentielle, au sens propre du terme ! Il faudrait des centaines, voire des milliers d'années à nos machines de bureau pour exécuter les applications imaginées par nos scientifiques. Le « calcul haute performance » devient une nécessité.

Le terme de calcul haute performance (ou HPC, pour *high performance computing*), ou calcul intensif, renvoie, à ce jour, aux machines pétaflopiques, c'est-à-dire de machines capables d'effectuer un million de milliards d'opérations à la seconde (voir encadré).

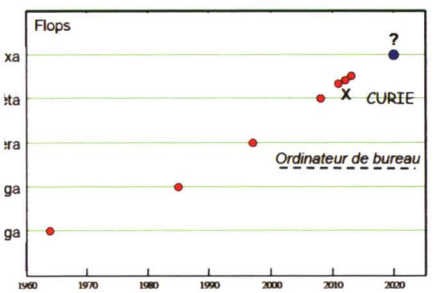
Le savoir-faire français dans le domaine du calcul intensif provient de la volonté de garantir la fiabilité des armes nucléaires par la simulation.

Un enjeu stratégique

De nombreux domaines académiques, mais aussi industriels, ont besoin de ces puissances de calculs : cosmologie, météorologie, médecine, chimie, physique nucléaire… Le calcul intensif est devenu un instrument de compétitivité pour nos entreprises. Il leur permet d'innover plus vite et de réduire le temps entre la conception et la mise sur le marché des produits. Pour toutes ces raisons, le pays devait se doter de moyens de calcul efficaces.

Il y a encore quelques années, la France accusait un retard certain dans ce domaine. La structure publique Genci (Grand Équipement national de calcul intensif) fut alors créée en 1987. Cette structure a financé et commandé un certain nombre de machines, dont Curie, d'une puissance de plus de 2 pétaflops, est la figure de proue, avec un accès ouvert à tous

les scientifiques européens. À titre indicatif, cette machine, implantée fin 2011, est constituée de plus de quatre-vingt-douze mille cœurs de calcul couplés à un système permettant de stocker l'équivalent de sept mille six cents ans de fichiers MP3, à une vitesse de 250 Go/s, soit cent mille fois supérieure aux meilleures liaisons ADSL. Elle peut effectuer en une journée cent cinquante années de travail de votre ordinateur de bureau ! En quelques années, la puissance de calcul disponible en France, pour le monde académique, passe de 20 téraflops à plusieurs pétaflops.



Progression exponentielle de la puissance des ordinateurs.

En 1995, le gouvernement français décide d'arrêter les essais nucléaires, et de garantir la fiabilité et la sûreté des armes nucléaires par la simulation. Pour satisfaire ces besoins, la Direction des applications militaires du Commissariat à l'énergie atomiques et aux énergies alternatives (CEA/DAM) a développé un nouveau savoir-faire dans le calcul intensif. Ce projet conduira en 2002 à la réalisation de la machine Tera (première machine téraflopique d'Europe, quatrième machine plus puissante du monde avec 1 téraflops), puis à Tera10 en 2006 (60 téraflops), et enfin à Téra100 en 2011, première machine pétaflopique d'Europe.

Le CEA/DAM de Bruyères-le-Châtel, fort de cette expérience dans la conception, la réalisation et l'exploitation des machines haute performance, a créé un complexe de calcul scientifique qui abrite le Centre de calcul recherche et technologique (CCRT), mis en service en 2003. Les missions du CCRT sont de répondre aux besoins du CEA et de ses partenaires en matière de grandes simulations numériques, proposer aux partenaires expertises et compétences dans le domaine du HPC, favoriser les échanges et les collaborations scientifiques entre le CEA et les partenaires industriels.

Vocabulaire arénaire

En informatique, le terme *flops* est l'acronyme de l'expression anglaise *floating point operations per second*, signifiant *opérations à virgule flottante par seconde*. C'est une des plus communes mesures de vitesse d'un système informatique. Dans cette unité, les capacités des ordinateurs sont des nombres gigantesques, qu'il a fallu dénommer pour en faciliter l'usage. Déjà Archimède s'était attaqué aux grands nombres en voulant estimer le nombre de grains de sable que pouvait contenir l'univers. Le système grec était alors limité à une myriade, soit dix-mille. Dans *l'Arénaire* (*arena* = sable), il dénomme *première octade* tous les nombres de un à une myriade de myriade (soit 10^8), quantité qu'il appelle *unité des nombres seconds*.

Il continue avec l'*unité des nombres troisièmes* (soit $10^{16} = 10^8 \times 10^8$), et ainsi de suite jusqu'à $\left(10^8\right)^{10^8} = 10^{8\times10^8}$. Il appelle alors *première période* l'ensemble de ces nombres, recommence le processus en remplaçant octade par période, pour arriver à un nombre qui n'existe pas dans la nature : $\left(\left(10^8\right)^{10^8}\right)^{10^8} = 10^{8\times10^{16}}$.

Pour les grands nombres informatiques, on utilise classiquement une base mille, dont vous connaissez les premiers éléments : kilo k (de *chilioi* = mille), méga M (de *megas* = grand), giga G (de *gigas* = géant). Pour le terme suivant, on utilise téra (T), qui vient de *teras* = monstre.
« Téra » peut être vu comme une déformation de *tetra*, quatre. En gardant ce principe, on obtient les termes suivants par modification du nom du nombre qui leur est associé dans l'exponentiation de k. Ainsi, $10^{15} = k^5$ a pour préfixe péta puisque *penta* = cinq et $10^{18} = k^6$, exa, car six = *hexa* en grec.

Le calculateur Curie.

Le TGCC (Très Grand Centre de calcul du CEA), nouvelle infrastructure verte, héberge, outre la première machine pétaflopique Curie, la nouvelle génération du CCRT. Cette structure a pour but d'accueillir des systèmes informatiques dédiés au calcul haute performance, offrir des espaces de communication pour de grands évènements scientifiques, et proposer un bâtiment modulaire et flexible dimensionné pour recevoir les futures évolutions des machines.

Les structures qui ont permis le développement de ces machines ne sont pas toujours simples à appréhender. Il existe bien sûr les structures européennes Prace (*partnership for advanced computing in Europe*) et ETP4HCP (*European technologie platform for HCP*), mais il est important de comprendre que la France, en s'appuyant sur l'expérience du CEA, reste au premier plan dans la course de ces technologies. Cependant, l'avenir sera au moins exaflopique (avec des machines capables d'effectuer 10^{18} d'opérations à la seconde).

Les défis du futur

Pour arriver à de tels résultats, les chercheurs doivent relever plusieurs défis. Le premier est la maîtrise de la consommation énergétique. En effet, la consommation électrique peut représenter jusqu'à 30 % du coût de fonctionnement, et il est impensable de multiplier par mille la consommation électrique des futures machines. D'autant que, de l'autre côté, une grande partie de cette électricité consommée est transformée par effet Joule en chaleur, et il faut alors refroidir les machines. Les techniques de refroidissement progressent, et on fait circuler de l'eau directement au plus près des processeurs, ce qui permet d'évacuer plus de calories. Mieux refroidis, les processeurs peuvent donc être rapprochés, ce qui diminue le temps de communication entre eux. Il est toutefois certain qu'il faudra concevoir de nouveaux microprocesseurs toujours plus puissants et moins énergivores. Enfin, il faudra augmenter la fiabilité des systèmes. On estime qu'une machine exaflopique contiendra des millions de cœurs de calculs, ce qui impliquera une panne par heure avec les fiabilités actuelles. L'exploitation de telles machines pourrait alors s'avérer difficile, sinon impossible. D'ores et déjà, les machines actuelles contiennent des redondances qui permettent de ne pas stopper les calculs à chaque panne.

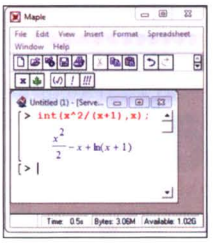
De nombreux obstacles restent à franchir, et la démocratisation du calcul haute performance, attendue avec impatience par les scientifiques et les industriels, sera longue, mais est en cours.

J.-J. D. & F. L.

Références

- *Les supercalculateurs relèvent le défi*. *La Recherche* 469, 2012 (disponible en ligne).
- www-hpc.cea.fr
- www.genci.fr

Maple



Capture d'écran de Maple montrant un calcul d'intégrale.

Maple est un logiciel de calcul symbolique dont le nom, qui signifie « érable » en anglais, vient sans aucun doute de son origine canadienne. Son ambition est de permettre des calculs comme les mathématiciens les font.

Ainsi, il est possible de calculer des intégrales comme : $\int \frac{x^2}{x+1}\,dx$.

Maple trouve un résultat exact :

$x^2/2 - x + \ln(x+1)$

mais ne précise pas les intervalles de validité car cela est en dehors de ses compétences.

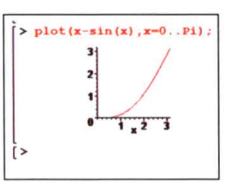
De plus, Maple calcule sur un espace de fonctions hors de portée de l'utilisateur moyen. Par exemple, il fournit des résultats comme :

$$\int_0^{\infty} \frac{dx}{x^2 - a^2} = \frac{\ln a - \ln(-a)}{2a}$$

qui ne pourra qu'étonner l'utilisateur. Un peu de curiosité permet de comprendre d'où vient un tel résultat : le logiciel calcule l'intégrale définie :

$$\int \frac{dx}{x^2 - a^2} = \frac{\ln(x-a) - \ln(x+a)}{2a}$$

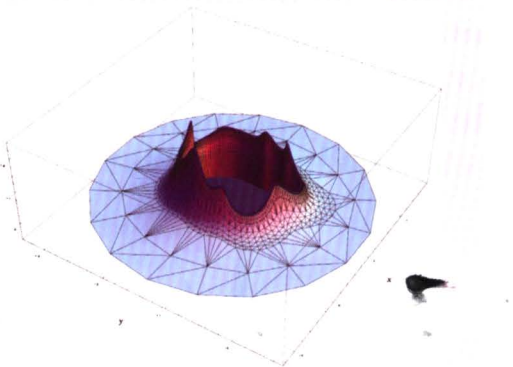
puis remplace x par les bornes de l'intégrale, sans vérification de sens. Cette vérification doit être faite par l'utilisateur, ce qui est normal mais ne doit pas être oublié. Maple permet également de visualiser des courbes et des surfaces, les instructions pour ce faire sont élémentaires (voir la figure). Pour finir ce bref tour d'horizon, Maple est programmable via un langage simple, ce qui permet en particulier d'utiliser des fonctions plus sophistiquées.



Tracé de la courbe d'équation : $y = x - \sin x$ entre 0 et p.

Mathematica : les maths de la fortune

Le logiciel de calcul formel Mathematica est édité par Wolfram Research depuis 1988. Arrivé sur le marché peu de temps après son concurrent Maple, il a immédiatement fait la fortune de son créateur, Stephen Wolfram (voir *Tangente* 134). Malgré une syntaxe pour le moins déroutante au premier abord, Mathematica a su s'imposer dans de nombreux milieux grâce à la qualité graphique des courbes et surfaces qu'il permet de générer et à sa souplesse d'utilisation.



Un exemple de graphique sous Mathematica.

Les autres atouts de Mathematica sont le calcul d'intégrales, la simplification de formules algébriques alambiquées, la manipulation de systèmes de logique formelle et la résolution exacte (en calcul symbolique) d'équations, grâce à une banque de données riche de nombreuses fonctions spéciales. En outre, les dernières versions du logiciel permettent de poser des questions ou de donner des instructions mathématiques en langue naturelle (l'anglais), directement dans l'invite de commande, sans passer par la syntaxe formelle !

Ce qui n'est pas toujours correctement interprété, on s'en doute : rien ne sait encore remplacer une bonne syntaxe pour interagir avec un logiciel…

par J.-F. Colonna et D. Justens

Quelques problèmes de calculs liés au caractère discret et fini de nos ordinateurs

Notre système de calcul est décimal, conséquence de nos deux mains à cinq doigts. La machine, elle, travaille en base deux, ce qui permet une représentation physique des nombres par le passage (1) ou non (0) d'une impulsion électrique. Ce qui a une incidence sur la façon de représenter les nombres.

Le système binaire utilisé en informatique permet la transition vers le système octal (base 8), qui synthétise toute suite de trois chiffres binaires en un seul. Théoriquement, tout nombre peut s'écrire indifféremment dans chaque système de comptage. Mais en pratique il est loin d'en être ainsi ! En effet, le fait pour un nombre de se présenter ou non sous une écriture finie va impliquer l'existence ou non d'un certain niveau d'erreur lors des différentes opérations à effectuer. Face aux nombres irrationnels, tous les systèmes semblent égaux : ces nombres ne sont jamais qu'approchés ! Mais il n'en va pas de même pour les nombres rationnels qui s'exprimeront de manière exacte ou non en fonction de la base choisie. Un changement de base implique alors une modification parfois significative du niveau d'erreur lors d'une succession d'opérations arithmétiques. Alors que dire des opérateurs non arithmétiques ?

a =	1023,1
b=a+1	1024,1

x =bx-a
1,000000000000000000
0,999999999988600000
0,999999999988347100000
0,999999886625850000
0,998877786553611000000
0,87484120955321000000
-127,17511729655700000000
-131263,137623404000000000000
-134427602,3401290000000000000
-137667308579,626000000000000000000
-140985090717418,000000000000000000000
-144382831403708000,00000000000000000000
-147862457640538000000,00000000000000000000
-151425942869675000000000,00000000000000000000
-155075308092834000000000000,00000000000000000000
-158812623017871000000000000000,00000000000000000000
-162640007232602000000000000000000,00000000000000000000
-166559631406908000000000000000000000,00000000000000000000
-170573718523814000000000000000000000000,00000000000000000000
-174684545140238000000000000000000000000000,00000000000000000000

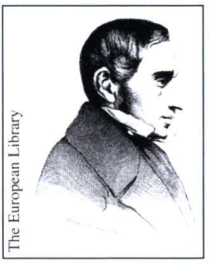
Notre confiance en la machine doit être adaptée en conséquence. Et parfois mise en doute. En voici un petit exemple. Considérons la suite de calculs établis comme suit. Soit a un nombre réel. Définissons $b = a + 1$ et $x_1 = 1$. Calculons $x_2 = bx_1 - a$. On vérifie que $x_2 = 1$ évidemment. Selon la même procédure, on peut calculer (par récurrence) $x_3, x_4 \ldots$

x_n. Et vérifier que cette suite est composée uniformément de « 1 », quel que soit le nombre a choisi au départ. Encodons à présent notre procédure dans un tableur quelconque et testons quelques valeurs particulières « bien choisies » pour a. La plupart des valeurs a produisent bien la suite de « 1 » attendue. Mais que se passe-t-il lorsque l'on introduit $a = 1023{,}1$? On arrive à la suite de valeurs présentées dans le tableau de la page 96, qui est bien loin de ce que l'on doit trouver !

Analysons ce résultat étonnant. Le nombre de zéros situés à droite des premières itérations est éclairant : il montre que le système travaille « en *double* précision », c'est-à-dire avec *deux* fois huit chiffres significatifs. Ceci peut se vérifier également en calculant la suite de nombres $(10^n + 1) - 10^n$, comme il est fait dans le second tableau. Dès la valeur $n = 15$, une erreur apparaît, limitant ainsi la précision de toute opération.

Certes, dans la plupart des applications, ce niveau de précision est suffisant. Mais ce n'est pas le cas dans une suite de calculs dans lesquels une erreur même extrêmement petite sur la valeur de l'un des x de la suite est amplifiée systématiquement. Quelle est donc la nature bien particulière du nombre 1023,1 qui conduit à ces résultats aberrants ? On obtient en fait ce type de suites de valeurs absurdes pour tout a de la forme $4^k - 0{,}9$ ou encore $2^{2k} - 0{,}9$, k étant un nombre entier naturel. Le lecteur curieux peut ainsi tester 3,1, 15,1 ou 63,1 pour constater que, lorsque k croît, la divergence devient plus rapide. La suite $(x_n)_n$ proposée est construite de telle sorte que toute différence initiale, si petite soit-elle, va s'amplifier de manière exponentielle, vu la multiplication introduite dans la procédure.

Pierre-François Verhulst



Gravure de Verhulst, par Léopold Flameng.

Le mathématicien bruxellois Pierre-François Verhulst (1804–1849) fut l'élève de Quetelet. Il est surtout connu pour sa construction de la courbe qu'il a lui même intitulée *« logistique »*, sans fournir d'explication satisfaisante à cette dénomination. Délaissant les modèles exponentiels de Thomas Malthus, Verhulst propose une modèle de croissance sous contrainte particulièrement bien adapté à l'évolution des populations puisqu'il conduit à l'existence d'une taille maximale pour toute population, fonction des conditions initiales. Verhulst a appliqué à plusieurs reprises son modèle à la population belge avec des conclusions variées. Détail amusant, la thèse de Verhulst, soutenue en 1825, portait sur la résolution des équations binomiales. Un demi-siècle plus tard, lorsque Conan Doyle décida de donner à Sherlock Holmes un ennemi mortel, il en fit un mathématicien... dont les travaux portaient précisément sur ce sujet. Verhulst a-t-il servi de modèle à Moriarty ?

Les nombres 2^{2k} s'écrivent de manière exacte en base 2. Qu'en est-il de la fraction 9/10 ? Un calcul simple montre que 0,9 s'écrit 0,111001100110011…, présentant bien une forme périodique illimitée. Que se passe-t-il lorsque l'on travaille en double précision ? On connaît bien l'approximation qui est souvent

n=	10^n+1	10^n	$(10^n+1) - 10^n$
1	11	10	1
2	101	100	1
3	1001	1000	1
4	10001	10000	1
5	100001	100000	1
6	1000001	1000000	1
7	10000001	10000000	1
8	100000001	100000000	1
9	1000000001	1000000000	1
10	10000000001	10000000000	1
11	100000000001	100000000000	1
12	1000000000001	1000000000000	1
13	10000000000001	10000000000000	1
14	100000000000001	100000000000000	1
15	1000000000000000	1000000000000000	0
16	10000000000000000	10000000000000000	0
17	100000000000000000	100000000000000000	0
18	1000000000000000000	1000000000000000000	0
19	10000000000000000000	10000000000000000000	0
20	100000000000000000000	100000000000000000000	0

faite et qui dit que $2^{10} = 10^3$ avec une erreur de l'ordre de 2 %. Pour obtenir une précision de l'ordre de quinze ou seize décimales (ce qui sémantiquement implique la base 10), il faut, en base 2, aller jusqu'à une cinquantaine de chiffres « après la virgule ». Les puissances de 1/2 donnent en fait les erreurs absolues maximales en base 2. Elles sont calculées, respectivement aux 16e et 17e décimales, dans le tableau ci-contre, qui donne également les erreurs absolues pour la valeur 0,9. On constate que ces erreurs sont maximales étant, pour une précision de 10^{-k}, approximativement égales à $(1/2) \times 10^{-k}$ (et même un peu supérieures) pour les valeurs $k = 16$ et $k = 17$. De là les résultats étonnant obtenus pour toute succession de calculs de nature divergente !

puissances de 1/2	Valeur décimale	erreur en base 2
50	8,88E-016	5,55E-016
51	4,44E-016	0,0000000000000011102
52	2,22E-016	
53	1,11E-016	
54	0,0000000000000005551	5,55E-017
55	0,0000000000000002776	0,0000000000000002776

Des problèmes vont donc également se poser lors de la mise en place de tout calcul générateur de chaos mathématique. Un deuxième exemple va nous montrer explicitement que les propriétés de distributivité ou d'associativité ne sont pas vérifiées lors de la mise en place de calculs effectifs utilisant la machine. Pour cela, introduisons la procédure chaotique connue sous l'appellation de *dynamique de Verhulst*, selon le nom du célèbre mathématicien belge. Le but de ce scientifique était de décrire l'évolution d'une population sous contraintes, de manière à échapper à l'absurdité du modèle exponentiel manquant de réalisme (voir *Mathématiques et Biologie*, Bibliothèque Tangente 42, 2011). Voyons ce qu'il en est en temps discret. Soit une population y mesurée à l'instant entier t (on note y_t cette population). L'instant suivant, cette population va croître proportionnellement à elle-même mais en subissant également un facteur d'atténuation expliqué par la limitation des ressources et la saturation de sa niche écologique. Verhulst étudie alors une procédure évolutive selon l'équation suivante :

$$y_{t+1} = y_t + a \times y_t \times \left(1 - \frac{y_t}{\mathrm{K}}\right).$$

Cette équation exprime une croissance proportionnelle à elle-même mais atténuée par un facteur proportionnel à la taille de la population. La constante K est supposée rendre compte des conditions naturelles propres à l'espèce concernée. Ceci n'est pas notre sujet. Concentrons-nous sur l'aspect formel de l'équation, qui peut s'écrire, en posant K = 1 :

$$y_{t+1} = y_t + ay_t + ay_t^2$$

On peut donner plusieurs notations formellement identiques de cette expression en utilisant les propriétés de commutativité, d'associativité et de distributivité dans les réels. Mais qu'en est-il des valeurs numériques obtenues ? Examinons et comparons les résultats obtenus pour les valeurs $a = 3$ en partant de $x_0 = 0{,}5$ et en donnant à notre expression les écritures suivantes, formellement équivalentes :

Dynamique de Verhulst			
a =	3,0000000000000000000		
	(a+1) yt – a yt^2	(a+1) yt – (ayt) . yt	Différences
x =	0,5000000000000000000	0,5000000000000000000	0,0000000000000000000
1	1,2500000000000000000	1,2500000000000000000	0,0000000000000000000
2	0,3125000000000000000	0,3125000000000000000	0,0000000000000000000
3	0,9570312500000000000	0,9570312500000000000	0,0000000000000000000
4	1,0803985595703100000	1,0803985595703100000	0,0000000000000000000
5	0,8198110957164320000	0,8198110957164320000	0,0000000000000000000
6	1,2629736848864000000	1,2629736848864000000	0,0000000000000000000
7	0,2665871533990120000	0,2665871533990130000	0,0000000000000000000
8	0,8531424825238830000	0,8531424825238850000	0,0000000000000000000
9	1,2290136436344900000	1,2290136436344900000	0,0000000000000000000
10	0,3846309658187850000	0,3846309658187930000	-0,0000000000000071054
11	1,0947009236750700000	1,0947009236750800000	-0,0000000000000119904
12	0,7836933578151340000	0,7836933578151030000	0,0000000000000310624
13	1,2922475940098600000	1,2922475940098800000	-0,0000000000000198242
14	0,1592788433666370000	0,1592788433665540000	0,0000000000000834877
15	0,5610061236339080000	0,5610061236336540000	0,0000000000000254241 07
16	1,2998408822714000000	1,2998408822712400000	0,0000000000001609823 4
17	0,1306045714133210000	0,1306045714139320000	-0,0000000000006110667 5
18	0,4712456234311120000	0,4712456234330780000	-0,0000000000019653723 1
19	1,2187651809155200000	1,2187651809178200000	-0,0000000000023043789 1
20	0,4188950250259720000	0,4188950250183390000	0,0000000000763300534

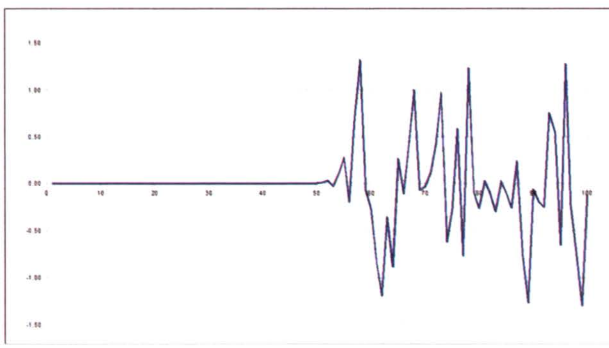
De nombreux autres exemples

Les deux exemples utilisés dans le corps de l'article (à savoir le traitement de l'équation de récurrence $x = bx - a$ et l'étude de la dynamique de Verhuslt) font partie des nombreux modèles que le premier auteur présente sur son site depuis... plus de vingt ans ! En effet, notre collaborateur, du Centre de mathématiques appliquées (CMAP) de l'École polytechnique, semble avoir été le premier à étudier le cas de l'exemple $x = bx - a$ (aucune mention antérieure n'a pu être trouvée dans la littérature) et certainement l'un des premiers à avoir soulevé le cas de la dynamique Verhulst. Pour plus de précisions, on pourra consulter cet article :
The Subjectivity of Computers. Jean-François Colonna, Technical Correspondence, *Communications of the Association for Computing Machinery (ACM)*, volume 36, numéro 8, pages 15 à 18, août 1993.
De nombreux exemples complémentaires (par exemple liés à l'attracteur de Lorentz, au problème des n corps ou à l'irréversibilité du temps) peuvent être trouvés en ligne à cette adresse :
www.lactamme.polytechnique.fr

$$y_{t+1} = (a+1)y_t - ay_t^2 \text{ et}$$
$$y_{t+1} = (a+1)y_t - (ay_t)\,y_t.$$

En notant bien qu'imposer à un ordinateur l'ordre des opérations à effectuer est chose difficile… On obtient le tableau de valeurs ci-contre. De petites erreurs apparaissent dès la dixième itération, montrant ainsi la non-associativité dans les calculs informatiques.

Ces différences vont aller en s'amplifier pour exploser littéralement après une cinquantaine d'itérations et donner dans les deux colonnes des résultats totalement indépendants. Le graphique ci-dessous représente ces différences successives entre les deux expressions formellement identiques de calculs.



D'autres expressions tout aussi équivalentes peuvent être testées avec des conclusions identiques. On peut ainsi calculer
$y_{t+1} = ((a+1) - ay_t)\,y_t$,
$y_{t+1} = ay_t + (1 - ay_t)\,y_t$ ou encore
$y_{t+1} = y_t + a(y_t - y_t^2)$

et effectuer une centaine d'itérations. On arrive à la ligne de valeurs suivantes :

	(a+1) yt – a yt^2	(a+1) yt – (ayt) . yt	((a+1) – (a yt)) yt	a yt +(1– (a yt)) yt	yt + a(yt – (yt)^2)
100	0,00006712709491283020	0,1194574394452290000	1,2564956762948600000	0,6580466479305640000	1,0428230333900900000

On peut légitimement se poser la question angoissante : quelle est *la vraie* valeur de x_{100} ?

J.-F. C. & D. J.

par Philippe Boulanger

Décimales de π et normalité

Les Japonais Alexander Yee et Shigeru Kondo ont calculé, le 16 octobre 2011, dix mille milliards de décimales de π. C'est à la fois un exploit et une misère, car nous savons que le nombre des décimales est infini. D'autre part, on n'y a pas reconnu quelques milliards de décimales consécutives de *e* ou d'un autre nombre remarquable. Et pourtant si π est *normal*, ce que tous les mathématiciens pensent mais qui n'a pas été démontré, alors toutes les suites finies de nombres y figurent.
L'absence de cette démonstration est exaspérante. Cette question aurait dû figurer dans la liste des problèmes de mathématiques dont la résolution est primée d'un million de dollars par la fondation Clay !

Nous supposerons que π est *normal* (on dit parfois aussi *universel*, mais on ne peut pas aller à l'encontre du vocabulaire politique !). Si donc toutes les suites de nombres existent dans π, le roman de votre vie y existe aussi, en français comme dans toutes les langues du monde, passées et à venir. Comment cela ? Il suffit de remplacer chaque couple de chiffres par une lettre ou un signe de ponctuation, et les couples de 00 à 99 y pourvoient largement. Hélas, avec cent milliards de signes dans π, sachant qu'un mot comporte en moyenne environ neuf lettres et avec les blancs, disons dix, il ne reste plus que dix milliards de mots. Un roman de vie peut comprendre cent mille mots, donc la bibliothèque avec le nombre de décimales calculées ne comprendra, en étant exagérément optimistes sur la signification des décimales de π dont aucune ne serait inutile, « que » cent mille livres, ce qui est bien insuffisant, car cent trente millions de livres ont été publiés depuis l'invention de l'imprimerie.

Alors pour l'histoire de votre vie en javanais, vous pourrez attendre… dix mille milliards de décimales de π, une misère. L'infini est un rêve.

Une ignominie révélée

Philippe Boulanger, chacun le sait, était directeur de la Rédaction au magazine Pour La Science. *Il raconte une anecdote liée aux décimales de π.*
« C'était il y a environ vingt ans (il y a prescription). Nous publiions, dans une autre revue que *Tangente*, une chronique mathématique sur le calcul des décimales de π. Et dans un souci d'illustration indispensable à l'attrait d'un journal, j'avais pensé illustrer par un portrait les principaux calculateurs de π, d'Archimède aux ordinateurs de l'époque.
J'en avais prévu huit et tout allait bien sauf pour un dénommé Machin, dont toute gravure restait introuvable. John Machin (1680–1751) est un mathématicien britannique dont la renommée est due au calcul, en 1706, de cent décimales de π grâce à la formule qui porte son nom, la formule de Machin :
$\pi/4 = 4 \arctan 1/5 - \arctan 1/239$.

Donc la place allouée à Machin, lequel devait trôner à côté de sa formule, était désespérément vide. Et la date fatidique de remise de copie approchait… Le drame couvait, rotatives en attente et lecteurs impatients. Pourquoi priver les lecteurs et les sept autres calculateurs de π de leur portrait ? Il fallait trancher. Je trouvais un portrait d'un perruqué anonyme de l'époque et lui attribuais la paternité de la formule. Qui peut prouver que ce n'était pas Machin ? Et comme il faut un crédit, je créais une Agence Truc qui nous avait procuré le portrait de Machin. Cette agence n'aurait eu qu'un client si des éditeurs de mathématiques ne nous avaient demandé le droit de publication de l'illustration ! Elle a été vendue deux fois. Pas pour rien, car la gratuité aurait allégué de notre turpitude. »

Des applications qui changent le monde

Mathématiques et informatique, une équipe gagnante. Que d'applications de ce partenariat talentueux voient régulièrement le jour ! Les exemples portent sur des sujets qui ponctuent notre quotidien : la compression des images, la représentation du son, la cryptographie, la sécurité informatique... L'utilisation de modèles mathématiques sophistiqués agissant sur les données massives fournies grâce à l'informatique dans de nombreux contextes (finance, biologie, commerce...) fait même débat dans la mesure où elle pose des questions d'éthique inédites qui, aujourd'hui, sont encore loin d'être toutes résolues.

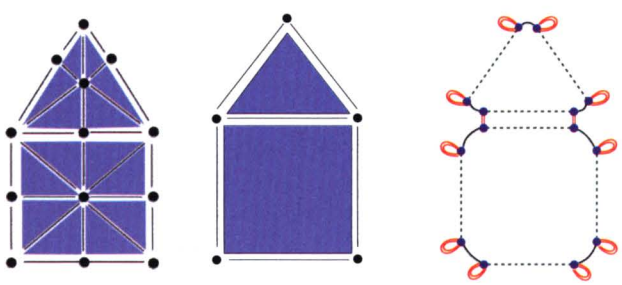
par Sylvie Alayrangues

Images numériques, du pixel à la topologie

Les *images numériques* sont partout ; mais sans les mathématiques, nos ordinateurs seraient bien démunis pour les exploiter. Au fait, qu'entend-on précisément par l'expression « image numérique » ?

Une image numérique peut être produite par deux grandes familles de procédés. Le premier d'entre eux est la *captation* d'une certaine réalité à l'aide d'outils très divers : appareil photo, scanner (2D ou 3D), caméras (unique ou multiples, 2D + temps ou 3D + temps), mais aussi appareils d'acquisition plus spécialisés comme pour l'imagerie médicale (IRM, tomographe) ou l'exploration des sous-sols (imagerie sismique). Le résultat de ces captations peut être, certes, une matrice de pixels (la représentation classique que l'on peut se faire d'une image numérique) mais pas seulement. On peut, par exemple, récupérer un nuage de points 3D (autrement dit un ensemble de points de $\mathbb{R}^3$).

Un complexe simplicial, un complexe cellulaire et une carte combinatoire représentant le même objet. Pour les deux premiers, l'objet est décomposé en cellules de dimension 0 (sommets), dimension 1 (arêtes) et dimension 2 (faces). La carte, elle, est fabriquée à partir d'éléments appelés brins reliés par trois involutions différentes (trait noir, double trait rouge, pointillés) et ce sont des compositions de ces involutions qui permettent de reconstituer les différentes cellules composant l'objet.



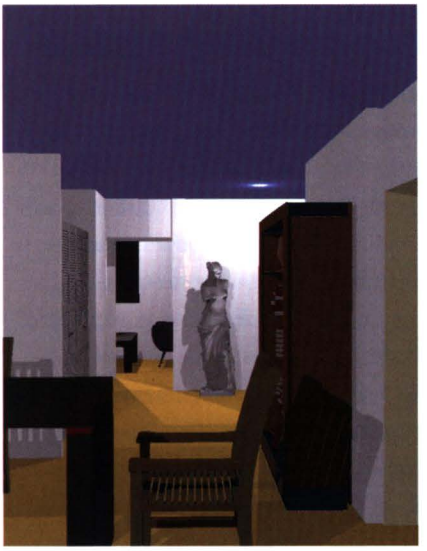
Au-delà des pixels

Une autre approche pour obtenir des images consiste tout simplement en la mise en œuvre d'un processus de *construction* (conception assistée par ordinateur ou CAO, architecture...). Dans ces processus constructifs, on utilise des objets mathématiques plus ou moins structurés pour représenter les objets contenus dans les scènes (« soupe » de triangles, autrement dit un ensemble de triangles sans

aucune information de connectivité entre eux, *complexes simpliciaux* voire cellulaires, graphes d'incidence, cartes combinatoires, courbes paramétriques...). L'idée sous-jacente derrière ces représentations consiste à modéliser une scène à l'aide de volumes ou de surfaces en mémorisant d'une manière plus ou moins fine selon les modèles comment ces volumes et ces surfaces sont agencés.

À ces objets mathématiques permettant de coder la structure de l'image peuvent être associées des données caractérisant les objets représentés. Il s'agit souvent d'informations de couleurs ou de textures mais l'on peut également utiliser des vecteurs de valeurs plus complexes ou des informations sémantiques. Ainsi, en imagerie médicale ou en télédétection, on peut avoir affaire à des images dites de *polarisation* dont chaque pixel est associé à un vecteur contenant 16 valeurs scalaires caractérisant cette propriété électromagnétique. En architecture, lorsqu'on modélise un bâtiment, il peut être utile d'associer aux différents volumes qui le composent des informations sémantiques : tel volume est une pièce, tel autre un mur, tel autre enfin une ouverture.

Bref, une image numérique, si elle peut être une simple matrice de pixels colorés, peut aussi être bien plus. En toute généralité, elle est définie par un objet mathématique, qui permet de coder la structure de la scène, objet auquel est associé un ensemble d'attributs qui caractérisent ses éléments. On pourra donc avoir des images peu structurées (construites sur des nuages de points ou des « soupes » de triangles...), des images structurées selon un modèle indépendant du contenu de l'image (immergées sur une grille régulière...) ou des images structurées par leur contenu (un maillage de la surface de l'objet représenté...).

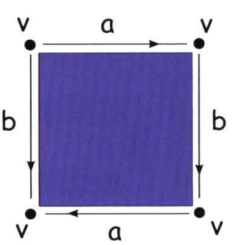


Une scène pour laquelle le calcul de l'éclairage exploite les propriétés topologiques du bâtiment (image extraite des travaux de Maxime Maria, Sébastien Horna et Lilian Aveneau).

Enrichissement de la topologie

Au-delà de la simple visualisation des images numériques sur un écran 2D (pas si simple que cela d'ailleurs lorsque celles-ci ne sont pas représentées par une matrice de pixels), de nombreuses applications ont besoin de calculer et d'utiliser des propriétés structurelles des scènes qu'elles contiennent.
Ainsi, être capable de reconnaître des formes est nécessaire pour pouvoir indexer des images de manière semi-automatique ou, en imagerie médicale, pour repérer d'éventuelles modifications structurelles des organes et détecter, par exemple, la présence de tumeurs.
En CAO, on doit pouvoir garantir que l'objet que l'on modélise peut être usiné. Si, suite à un mauvais recollement lors de la construction du modèle

Pas la peine de découper votre revue pour essayer de coller les arêtes selon les flèches, vous n'y arriverez pas : la bouteille de Klein ne peut pas être construite dans notre monde à trois dimensions.

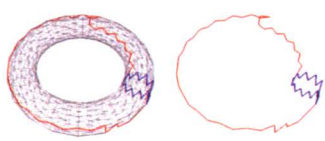


La topologie reconnue par l'ordinateur

Comment un ordinateur peut-il « voir », « manipuler » et « exploiter » la topologie d'une scène représentée dans une image ?

Tout d'abord, le modèle mathématique utilisé pour représenter l'image doit permettre d'encoder sa topologie : un nuage de points et une soupe de triangles en sont par exemple bien incapables. Extraire la topologie d'un objet représenté dans une matrice de pixels pose également un certain nombre de problèmes. Fort heureusement, des algorithmes ont été développés pour reconstruire des modèles topologiques à partir de telles données. En outre, il faut disposer d'informations topologiques calculables et exploitables (par exemple comparables) sur des images. La topologie algébrique est ici d'un grand secours puisqu'elle fournit toute une panoplie d'invariants : caractéristique d'Euler, groupes d'homologie, groupes d'homotopie...

Les groupes d'homologie permettent de caractériser les « trous » d'un objet. Cette information est calculable et peut être mise sous la forme d'un vecteur d'entiers (le nombre de Betti et d'éventuels coefficients de torsion). Il est donc aisé de comparer les groupes d'homologie de plusieurs objets. En outre, il est possible de calculer un représentant pour chaque trou. Le tore représenté ci-contre possède deux trous de dimension 1 (en rouge et en bleu).



La difficulté réside alors dans la définition, le calcul et l'utilisation de tels invariants sur les modèles mathématiques particuliers utilisés dans les images numériques. Il s'agit là d'un domaine de recherche très actif.

➜ Références (en France) concernant le calcul de l'homologie sur des images :
travaux de Sylvie Alayrangues, Dobrina Boltcheva, Guillaume Damiand, Laurent Fuchs, Jacques-Olivier Lachaud, Pascal Lienhardt, Samuel Peltier.

3D, votre machine-outil tente de produire *une bouteille de Klein*, il y a fort à parier qu'elle ne s'en remettra pas !

En synthèse d'images, lorsqu'il s'agit d'éclairer une scène, il est indispensable de savoir quels éléments de la scène vont être éclairés par une source de lumière et quels autres seront occultés. Il n'est ainsi plus utile, lorsqu'une scène représente un bâtiment entier, de chercher à calculer l'éclairage apporté par une ampoule allumée au sous-sol sur les objets présents dans le grenier.

Le calcul de caractéristiques topologiques permet de répondre partiellement à certaines de ces questions. Les propriétés topologiques sont, en effet, des propriétés qui ont trait à la structure même des objets ou des scènes représentées. Dans les applications de reconnaissance de forme, elles peuvent ainsi permettre d'éviter un certain nombre de biais.

Par exemple, un même objet capté sous différentes conditions d'éclairement, d'angles de vue, en utilisant éventuellement des modes de captation différents, garde (modulo les erreurs d'acquisition) la même topologie, même si sa géométrie est déformée, ses couleurs modifiées... Réciproquement, il est possible de distinguer certains objets en observant leur topologie. Ainsi, sans les goûter, il est possible de distinguer un *donut* d'un bretzel rien qu'en comptant le nombre de trous de chacun des deux objets

Lors de l'étude de séquences d'images de battement de cœur, être en mesure de détecter les changements de topologie provoqués par l'ouverture et la fermeture de la valve aortique peut permettre de déceler des anomalies.

Un *donut* et un bretzel différent par le goût mais aussi par la topologie.

En modélisation 3D, qu'il s'agisse de représenter un bâtiment ou bien des couches géologiques, il est primordial de connaître la topologie de l'agencement des volumes qui constituent de telles scènes. Quelle pièce dans un bâtiment est voisine de quelle autre ? Quels blocs du sous-sol proviennent d'une même couche géologique initiale ?

Attention cependant, la topologie, seule, ne peut répondre à toutes les questions. Un *donut* et une tasse avec une anse n'ont pas grand chose en commun si ce n'est qu'ils sont topologiquement parfaitement équivalents ! Et lorsqu'un architecte modélise la topologie d'une tour par exemple, c'est bien le plongement géométrique (entre autres) qu'il va lui associer qui déterminera si elle finira penchée comme la tour de Pise.

Il est donc toujours nécessaire de combiner l'utilisation de la topologie avec celle d'informations d'autres natures : géométriques, colorimétriques, mécaniques, sémantiques... Et bien souvent, l'expertise humaine reste indispensable pour obtenir le résultat souhaité.

S.A.

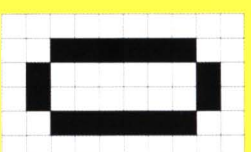
Topologie, image et paradoxe

Définir une topologie sur une image captée n'est pas si facile. Prenons un cas simple, une image 2D en noir et blanc. Les pixels blancs correspondent au fond de l'image, les pixels noirs aux objets présents. Plutôt que de tenter de définir *in extenso* une topologie sur une image, on peut essayer d'introduire des notions topologiques adaptée aux images. Par exemple, comment sait-on qu'un amas de pixels appartient à un même objet ? Probablement parce qu'ils sont « collés » les uns aux autres, autrement dit qu'ils constituent une composante connexe. Comment peut-on définir une connexité sur une grille ? Assez naturellement, en passant par la notion d'adjacence. Deux pixels noirs appartiennent au même objet s'ils sont voisins (adjacents) ou s'il existe entre eux un chemin composé de pixels noirs deux à deux adjacents. Ici, la connexité ressemble fort à une connexité par arc.

Mais qu'appelle-t-on des pixels « voisins »? Ceux qui se rencontrent sur une arête ? Ou bien sur au moins un sommet ? Chaque pixel a-t-il quatre ou huit voisins ? En fait, pourquoi choisir ? On peut définir les notions de 4- et de 8-adjacence qui, par extension, donnent naissance à la 4- et à la 8-connexité.

Une autre propriété topologique importante est celle de Jordan–Brouwer : une courbe fermée se doit de séparer l'espace en deux. Avec la notion d'adjacence, on peut définir une courbe comme un ensemble de pixels deux à deux adjacents. Une courbe sera ainsi 8-*connexe* si les pixels qui la composent sont 8-adjacents.

On arrive au paradoxe. Sur la figure, l'ensemble des pixels noirs forme une courbe 8-connexe. Sépare-t-elle bien l'espace en deux composantes 8-connexes distinctes ? Non ! Essayons avec la 4-connexité. Observons les pixels blancs : ils forment bien deux composantes 4-connexes distinctes, mais la courbe qui les sépare n'est pas 4-connexe. Comment contourner ou lever ce paradoxe ?



par Daniel Justens

La méthode de Monte-Carlo

Application à un investissement financier

Lorsque l'ensemble des paramètres d'un problème (par exemple la rentabilité d'un investissement financier) est trop important ou ne peut être entièrement décrit, des méthodes numériques statistiques deviennent les seuls moyens de modéliser la situation. La méthode de Monte-Carlo en fait partie.

Contrairement à certaines idées reçues, les mathématiques ne constituent pas une science définitive, au sein de laquelle tous les résultats sont établis une fois pour toutes. Bien au contraire, les maths sont en permanence en construction. De nouveaux résultats sont découverts chaque jour et complètent progressivement le grand livre de nos connaissances.

Tous les problèmes théoriques ne sont donc pas résolus. Et il en est de même pour de nombreux problèmes pratiques en manque de support mathématique utilisable concrètement.

La grande complexité de certains cas de figure en économie, en biologie, en physique, en finance, fait que nos mathématiques, toutes satisfaisantes qu'elles puissent sembler, s'avèrent souvent insuffisantes. C'est pour ce type de problèmes que l'informatique se révèle d'un secours incroyablement utile.

Quand des méthodes numériques suppléent à l'absence de solution mathématique…

Une application « théorique » au calcul d'intégrales

Parmi les méthodes numériques initiées depuis quelques décennies et rendues pragmatiques grâce à l'informatique, on retrouve la fameuse méthode de Monte-Carlo mise au point vers 1947 par le physicien gréco-américain Nicholas Constantine Metropolis (1915–1999), l'un des collaborateurs de John von Neumann (1903–1957). C'est aussi Nicholas Metropolis qui fut l'un des inventeurs d'un des tout premiers ordinateurs opérationnels qu'il baptisa du nom de MANIAC (Mathematical Analyzer, Numerical Integrator And Computer) pour couper court à la mode des acronymes étranges qui désignaient ce genre de machines. MANIAC I fut totalement opérationnel dès 1952.

La méthode de Monte-Carlo s'est révélée particulièrement utile pour le calcul de certaines intégrales portant sur de très larges domaines, donnant en pratique des résultats numériques nette-

ment plus précis que les méthodes classiques déterministes (voir l'article de François Lavallou pages 98 à 103 du HS « Bibliothèque » 34 de *Tangente* consacré aux statistiques).
Mais cette méthode est également employée en finance, domaine dans lequel elle permet de quantifier assez exactement le risque par une approche statistique adaptée. La suite de l'article présente la méthode dans ce cadre particulier, en liaison avec les méthodes numériques implémentées.

Considérons un projet d'investissement de montant connu, concrétisant par exemple le rachat d'une entreprise. Quels vont être les résultats de cette entreprise pendant les dix prochaines années, une durée qui correspond par exemple à la durée maximale d'utilisation de son outil de production ? En réalité nul n'en sait rien. Tout au plus dispose-t-on d'informations du passé, qui ne sont en rien garantes des résultats futurs.

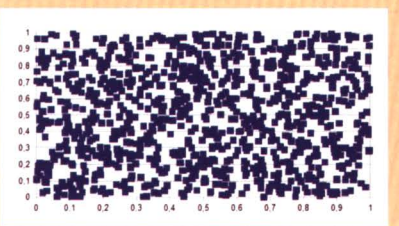
La modélisation statistique du problème

Pour gérer ce type de problème, on va considérer que les résultats futurs de l'entreprise sont des variables aléatoires de distributions connues. En effet, études de marché et résultats antérieurs peuvent nous donner une idée de la tendance des résultats attendus (moyenne) et également de leur niveau de variabilité (écart type). Reste à choisir un type de distribution pour cette variable aléatoire. Le *théorème central limite* conduit souvent à choisir une distribution normale. En effet, tout phénomène qui est la somme d'un grand nombre de petits phénomènes aléatoires indépendants de variance finie est distribué normalement. Et l'on considère généralement le résultat d'une entreprise sur un an comme la somme de ses résultats partiels.
L'idée de l'équipe de Nicholas Metropolis a été de proposer de simuler artificiellement un grand nombre de futurs possibles de l'entreprise conformément au choix de distribution de probabilité adoptée et ensuite de travailler statistiquement sur les résultats obtenus.
La première difficulté consiste à reproduire artificiellement le hasard. On parle alors de variable « pseudo-aléatoire ».
L'idée de départ est de commencer par simuler une distribution uniforme sur l'intervalle [0, 1] à partir de calculs résolument déterministes. Le tableur Excel fournit un opérateur de ce type sous la fonction ALEA().

Simuler une distribution uniforme

Considérons trois entiers (bien choisis) :
$A = 2051$, $B_1 = 2097153$ et $C = 4194304$.
On construit les nombres $D = A \times B_1$ puis D/C dont on prend la partie entière $E[D/C]$. On redéfinit B_2 (procédure itérative usuelle en informatique) :
$B_2 = D - E[D/C] \times C$.
Le nombre $x = B_2/C$ est bien compris entre 0 et 1.
On recommence alors la procédure à partir de B_2.
Numériquement les résultats obtenus se distribuent assez bien pseudo-aléatoirement de manière uniforme.
On peut visualiser le résultat en considérant le graphique suivant dans le plan, reprenant 1000 points obtenus par la suite $(x_1, x_{1001}), \ldots, (x_{1000}, x_{2000})$.

On imagine assez mal de faire ce genre d'opérations successives sans l'appui de l'informatique. La simulation décrite peut être téléchargée sous formats XLS et ODS sur le site www.infinimath.com.



On observe une répartition quasi uniforme des points. Notre œil est un assez bon juge de l'uniformité de la dispersion.

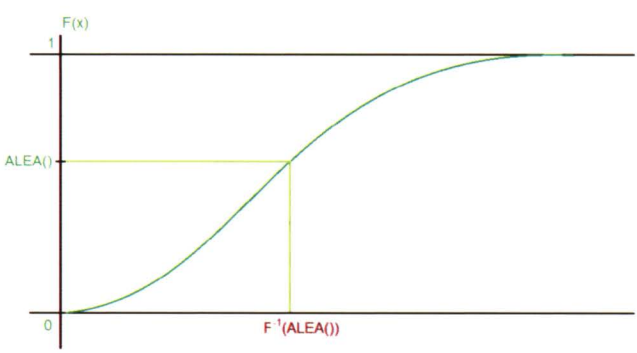
Mais on peut en construire soi-même assez facilement. L'encadré de la page précédente décrit un petit algorithme qui ne donne pas de trop mauvais résultats et que chacun peut implémenter facilement sur son PC.

Nous voici donc capables de simuler aléatoirement (ou presque) une valeur distribuée uniformément sur [0, 1] qui est précisément l'intervalle image de toute fonction de répartition. En effet, pour une variable aléatoire X quelconque, la fonction de répartition F quantifie la probabilité associée à l'ensemble des valeurs pouvant être prises par X et qui sont inférieures ou égales à X :

$$F(x) = P[\omega \text{ tels que } X(\omega) \leq x].$$

F est une fonction croissante dont l'image parcourt l'intervalle [0, 1].

Les mathématiciens George Edward Pelham Box et Mervin Edgar Muller ont alors proposé vers 1958 de simuler toute variable aléatoire à partir de la réciproque de cette fonction de répartition. Soit ALEA une variable aléatoire uniformément distribuée sur [0, 1]. Pour une distribution de probabilité de fonction de répartition F, la valeur $F^{-1}(ALEA)$ simule une valeur numérique compatible avec la distribution envisagée. Graphiquement, les choses sont visualisées sur la figure ci-dessous.



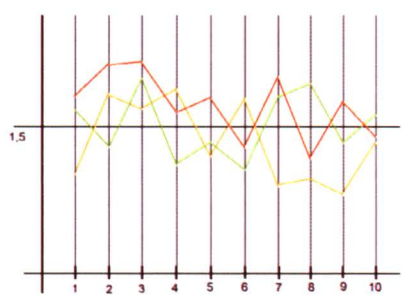
Mais que faire avec la distribution normale dont on sait que la fonction de répartition n'a pas une structure agréable puisque sa densité n'est pas intégrable ? Box et Muller suggèrent de travailler à deux dimensions et de passer aux coordonnées polaires.

Le détail du calcul est donné dans l'encadré de la page suivante.

Une application « pratique » au projet d'investissement

Voici l'application de ces résultats dans le cas de l'exemple de l'acquisition d'une entreprise. On va supposer que l'on acquiert une entreprise pour le prix de 10 U.M. (unité monétaire) et que les résultats futurs pour les dix années à venir peuvent être représentés par des variables aléatoires normales de moyenne 1,5 U.M. et d'écart type 0,5 U.M.

On entre ici dans l'univers incontrôlable des calculs informatiques. Les résultats futurs de l'entreprise sont simulés selon les relations établies, créant ainsi un grand nombre de futurs possibles également probables. Voici un graphe présentant trois trajectoires possibles dans l'espace des futurs compatibles avec nos hypothèses.



Pour chaque trajectoire, le futur de l'entreprise est connu et le rendement associé peut être calculé. Toutes ces procédures sont exclusivement numériques et se situent dans le monde informatique.

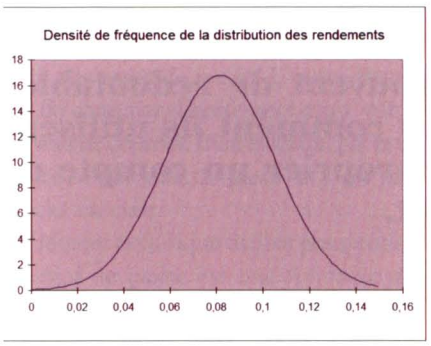
En simulant ainsi un millier de futurs possibles et en y associant les rendements correspondants, on crée une sta-

tistique de rendements que l'on peut soumettre aux méthodes usuelles d'interprétation.

Avec nos données numériques, on arrive à une distribution de rendements dont la densité observée est représentée ci-dessous.

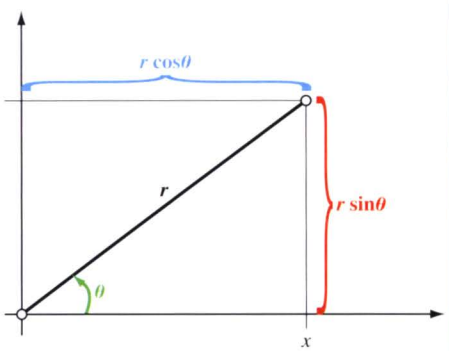
La moyenne statistique vaut 0,08154 et l'écart type 0,02374.

Il faut donc s'attendre à un rendement compris entre 2 et 14% avec fiabilité 95%.



La méthode Monte-Carlo nous livre ici un résultat inaccessible au moyen de méthodes traditionnelles mais que seule une étroite collaboration entre les deux domaines mathématiques et informatique a rendu possible.

D. J.



Représentation graphique d'un point en coordonnées polaires.

Simuler une distribution uniforme

Rappelons la fonction densité $f(x)$ d'une distribution normale de moyenne m et d'écart type σ :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2}.$$

Le passage à une variable centrée réduite simplifie la présentation : $z = \dfrac{x-m}{\sigma}$ donne $dz = \dfrac{dx}{\sigma}$.

Toute intégrale sur un domaine D se transforme en :

$$\int_D \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2} dx = \frac{1}{\sqrt{2\pi}} \int_{D'} e^{-\frac{z^2}{2}} dz.$$

En dimension 2 et en coordonnées polaires, on représente un point (x, y) du plan en joignant ce point à l'origine, et en mesurant d'une part l'angle entre ce segment et l'horizontale (θ) et d'autre part la distance entre le point et l'origine (r). On a : $x = r\cos\theta$ et $y = r\sin\theta$ (voir graphe en bas ci-contre). r appartient à $\mathbb{R}^+$ et θ parcourt l'intervalle $[0, 2\pi]$.

Le changement de variables dans l'intégrale double impose la multiplication par le *Jacobien*, déterminant de la matrice des dérivées partielles des deux variables de départ relativement aux deux nouvelles variables. Ici, il vaut :

$$J = \det\begin{vmatrix} \dfrac{\partial x}{\partial r} & \dfrac{\partial y}{\partial r} \\ \dfrac{\partial x}{\partial \theta} & \dfrac{\partial y}{\partial \theta} \end{vmatrix} = \det\begin{vmatrix} \cos\theta & \sin\theta \\ -r\sin\theta & r\cos\theta \end{vmatrix} = r\cos^2\theta + r\sin^2\theta = r.$$

L'intégrale double (sur un certain domaine D) de deux variables normales réduites indépendantes x et y se sépare naturellement en deux intégrales simples :

$$\frac{1}{2\pi}\iint e^{-\frac{1}{2}(x^2+y^2)} dxdy = \frac{1}{2\pi}\int d\theta \int re^{-\frac{r^2}{2}} dr.$$

La première quantifie une distribution uniforme de la variable θ sur l'intervalle $[0, 2\pi]$, ce qui est facile à simuler. Pour la deuxième, il suffit de poser $r^2/2 = t$ ce qui donne $r\,dr = dt$ pour arriver à une fonction intégrable dont l'image sur l'ensemble des réels positifs est l'intervalle $[0, 1]$, nous permettant à nouveau de recourir aux distributions uniformes :

$$\int re^{-\frac{r^2}{2}} dr = \int e^{-t} dt.$$

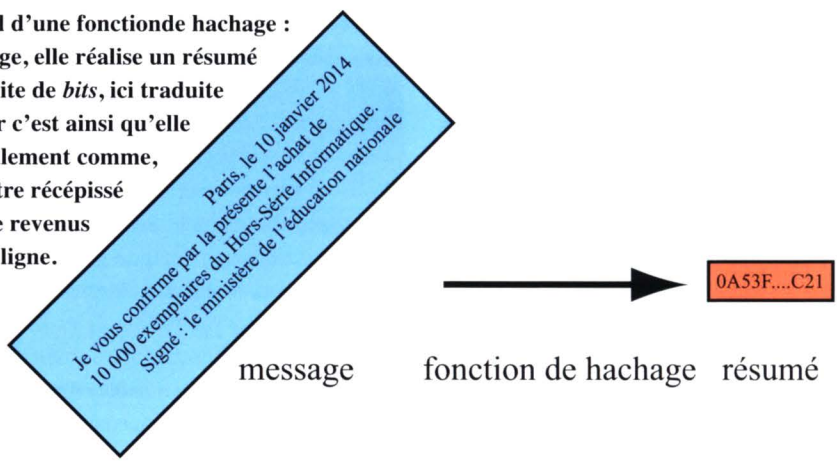
On pose donc $\theta = 2\pi \times \text{ALEA}$ et $e^{-\frac{r^2}{2}} = \text{ALEA}$, soit $r = \sqrt{-2\ln(\text{ALEA})}$. La simulation de variables normales réduites se fait alors au moyen de $z = \sin(2\pi\text{ALEA}) \times \sqrt{-2\ln(\text{ALEA})}$.

par Hervé Lehning

Pirater un site ou une messagerie

Les hackers peuvent être de simples passionnés d'informatique, ils sont aussi bien souvent de redoutables escrocs. Dans cet article, nous voyons comment ils utilisent les faiblesses d'un système pour s'approprier un compte de messagerie ou pénétrer un site interdit.

Principe général d'une fonctionde hachage : à partir d'un message, elle réalise un résumé qui est une suite de *bits*, ici traduite en hexadécimal car c'est ainsi qu'elle apparaît normalement comme, par exemple, sur votre récépissé de déclaration de revenus en ligne.



Vous avez sans doute tous reçu des messages du type : « *Bonjour*

J'espère que tu vas bien. J'ai besoin de ton aide, je suis en déplacement et je me trouve dans une situation très compliquée que je voudrais t'expliquer. Je reste cependant connectée en attendant ta réponse, puisque je ne suis joignable que par mail. »

Ce message est signé d'une amie et provient de son adresse électronique. Si vous êtes prudent, vous n'y répondrez pas et, éventuellement, lui téléphonerez… pour vous apercevoir qu'elle est chez elle, bien loin des soucis. Certains s'y font prendre et se font détrousser, parfois de quelques milliers d'euros ! Le bon sens suffit pour éviter ce piège. Nous nous intéressons ici aux procédures suivies par les escrocs pour s'approprier le compte de messagerie d'autrui, et comment l'empêcher. Pour le comprendre, il faut savoir comment les comptes sont protégés.

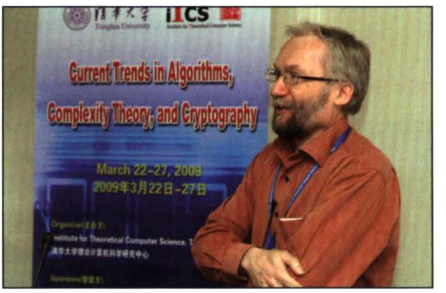
Hacher pour cacher

Pour utiliser un compte de messagerie, il faut son identifiant et son mot de passe. Le problème du hacker est de trouver votre mot de passe. Il existe deux types de méthodes pour cela. La première est l'espionnage de votre ordinateur. Cela peut être fait à partir d'un virus (*spyware*) ou d'un dispositif physique. Ce dernier cas est peu probable, sauf si vous êtes particulièrement ciblé. L'espion envoie à une adresse toutes vos frappes au clavier et, parmi elles, se trouveront tous vos mots de passe. L'autre façon est de chercher directement chez votre serveur de courrier. Bien entendu, par prudence, votre mot de passe n'y est pas stocké en clair.

L'élément technique utilisé pour coder un mot de passe est une *fonction de hachage*. Il s'agit d'une application transformant un texte, par exemple un mot de passe ou un contrat, en un résumé de longueur fixe, que certains nomment « haché » car la structure de ces fonctions fait effectivement penser à la technique de certains chefs pour découper les oignons.

Ces fonctions doivent répondre à quelques impératifs de sécurité. Tout d'abord, comme les résumés sont connus ou faciles à découvrir, ils ne doivent pas permettre de remonter aux originaux. Dans le cas contraire, cela pourrait entraîner des falsifications. Cette propriété indispensable est la *résistance à la pré-image*. Ce terme vient du langage mathématique associé aux fonctions. Le résumé est ici l'image du message par la fonction de hachage qui, lui, est la pré-image du résumé.

Cette résistance ne suffit pas pour éviter la fabrication de faux. Une bonne fonction de hachage doit également assurer la *résistance à la seconde pré-image*. Connaissant le message, il doit être difficile de trouver un autre message ayant le même résumé. Si ce n'est pas le cas, connaissant un de vos contrats signés, on peut en fabriquer un autre et prétendre que vous l'avez signé, puisqu'il a le même résumé ! Cette résistance est nécessaire pour éviter la contrefaçon.

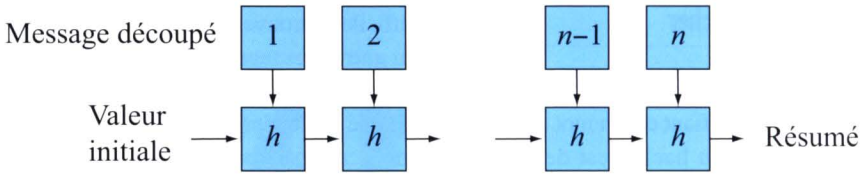


Ivan Damgård.

Enfin, une fonction de hachage doit assurer la *résistance aux collisions*. Il doit être difficile de trouver deux messages ayant le même résumé. Si ce n'est pas le cas, il est possible de vous faire signer un contrat et de prétendre que vous en avez signé un autre puisqu'il aura le même résumé. Remarquez que la résistance aux collisions implique la résistance à la seconde pré-image. En ce qui concerne les mots de passe, nous nous intéressons seulement à la résistance à la pré-image.

Ces principes étant posés, comment réaliser une fonction de hachage ? Comme souvent en informatique, l'idée est de la construire de façon progressive. Imaginons que nous disposions d'une fonction h fournissant un résumé sûr de 128 *bits* d'un message de 256 *bits*. On découpe alors le message donné M en blocs de 128 *bits* : M_1, M_2, *etc*. On combine un message initial de 128 *bits* à M_1 pour obtenir, grâce à h, un résumé de 128 *bits*, que l'on combine à M_2. On obtient un nouveau résumé de 128 *bits*, et on recommence avec M_3. En continuant ainsi, on obtient un résumé final de 128 *bits*, c'est-à-dire une fonction de hachage H résumant tout message en 128 *bits* (voir le graphique ci-contre en haut).

Ralph Merkle et Ivan Damgård ont montré que si h est résistante aux



Construction de Merkle–Damgård. Le message est découpé en blocs de 128 *bits* puis on résume succesivement des chaînes de 256 *bits* en partant d'une chaîne arbitraire, dite la *valeur initiale*. Bien entendu, le procédé est généralisable en remplaçant 128 par tout autre nombre.

collisions alors H l'est aussi ! On peut réaliser une telle fonction de hachage au moyen d'un chiffrement par blocs, comme le code AES (Advanced Encryption Standard), puisqu'il combine une clef secrète de 128 *bits* à un bloc de 128 *bits* pour obtenir un bloc de 128 *bits*. Il suffit d'introduire la clef secrète comme valeur initiale de l'itération. La fonction de hachage obtenue est aussi résistante que le chiffrement utilisé. L'ennui est que l'AES ou les fonctions de chiffrement en général sont longues à calculer. C'est pourquoi on a introduit des fonctions de hachage plus rapides, tels MD5 (Message Digest) sur 128 *bits* ou SHA-1 (Secure Hash Algorithm) sur 160 *bits*, qui utilisent des chiffrements simplifiés. Malheureusement les clefs des chiffrements sont trop courtes, ce qui autorise une attaque exhaustive. Le nombre de cas à analyser pour MD5 est de 2^{16}, ce qui est devenu beaucoup trop faible à l'heure actuelle. Pour SHA-1, il est de 2^{60}, ce qui peut être atteignable en mettant un très grand nombre d'ordinateurs en réseau. Il est possible d'améliorer ces attaques. Cependant, SHA-1 est encore considéré comme relativement sûr.

Recherche du mot de passe

Le hacker dispose donc du haché de votre mot de passe ainsi que de la fonction de hachage utilisée. Si MD5 est au cœur de cette fonction, il est possible de retrouver le mot de passe par recherche exhaustive (on dit aussi « par force brute »). S'il s'agit de SHA-1, il est mieux protégé. Une façon toute simple de casser un mot de passe est d'essayer tous les mots de passe classiques. Il existe ainsi des dictionnaires de mots de passe usuels : mots courants de la langue natale de l'utilisateur (comme « maison » ou « objet »), prénoms, suites de chiffres « logiques » comme 123456, mots spécialisés et rares du domaine de l'utilisateur comme « acatalectique », *etc*. Les hackers utilisent ainsi un grand nombre de dictionnaires que l'on peut trouver sur Internet.

Pour éviter le cassage par dictionnaire, il est nécessaire d'utiliser des mots de passe qui ne peuvent y figurer. Il est prudent aussi d'en avoir un différent pour chaque compte et site utilisé. Une idée simple pour ce faire est de partir d'un mot et d'un nombre, et de les mixer différemment pour chacun de vos comptes, mais de façon logique pour vous. Par exemple, si vous utilisez *Tangente* et votre date de naissance *25 12 1993*, vous pouvez les marier selon plusieurs lois, que vous pouvez même noter. Par exemple, 1-2-3-4-3-2-1 donne :

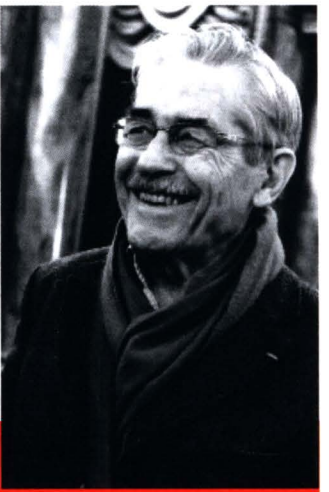
T25ang1219ent93e. Ceci n'est qu'un exemple, et il est évidemment possible de compliquer à loisir la méthode !

H. L.

Une brève histoire d'Internet

Le mythe des origines

Arpanet, l'ancêtre d'Internet, naquit en pleine guerre froide, en 1969. Sa structure décentralisée aurait eu pour but de résister à une destruction partielle du réseau, due à une attaque nucléaire. Cette légende est contredite par le fait que les premiers sites d'Arpanet étaient des universités, des sites de recherche et des entreprises et non pas des sites militaires. L'origine de ce mythe semble être un rapport sur le sujet, écrit par Paul Baran, parfois présenté comme à l'origine d'Arpanet, mais dont le projet fut un échec.



Louis Pouzin, né en 1931, précurseur d'Internet.

Le datagramme et Louis Pouzin

Le datagramme à la base du protocole d'Internet fut inventé par un Français, Louis Pouzin, au début des années 1970. L'administration des Télécoms préférant pousser l'éclosion du Minitel, le projet de Louis Pouzin fut abandonné, puis repris par les Américains. Il devint le fameux protocole IP (Internet Protocol) à la base d'Internet.

Les protocoles IP

Les protocoles IP, très proches des datagrammes de Louis Pouzin, sont destinés à assurer le transfert des communications à travers Internet. Conçus en 1974, ils devinrent la règle en 1983. Contrairement au réseau téléphonique, leur particularité est de ne pas fixer le chemin du transfert à l'avance. La communication passe par le chemin le moins saturé. Le principal défaut de ce système, aujourd'hui vieillissant, est d'avoir été conçu sans le moindre souci de sécurité. Autrement dit, avec ce protocole, si l'on souhaite avoir un minimum de sécurité, il est *nécessaire* de chiffrer les messages !

La gouvernance d'Internet

La gouvernance d'Internet concerne essentiellement l'attribution des adresses IP et des noms de domaines. Les adresses IP sont attribuées automatiquement par le réseau où l'on se connecte. La gestion des noms de domaine, comme www.infinimath.com, est faite par une société privée de droit californien créée en 1998, l'ICANN (Internet Corporation for Assigned Names and Numbers), ce qui est pour le moins curieux ! Cela amène même à des dysfonctionnements dangereux. Par exemple, en juin 2014, l'ICANN désire vendre au plus offrant les noms de domaines de suffixes « vin » ou « wine ». Ainsi, un particulier en Australie, ou ailleurs, pourrait acquérir le domaine www.bordeaux.vin ou www.champagne.vin sans respecter les indications géographiques protégées.

Est-il légitime qu'une société californienne vende le nom de domaine www.bordeaux.vin au plus offrant ?

par N. Anciaux et B. Nguyen

Limiter la collecte des données personnelles

Un problème juridique NP-difficile

Oui, un problème mathématique tout à fait théorique à l'origine peut avoir une application très pratique ! La preuve avec le problème de *n*-exposition, qui concerne la collecte de données personnelles, et la problématique de l'attribution d'aides sociales. Un défi basé sur des données réelles est proposé aux lecteurs.

Le concept de protection des données personnelles existe depuis longtemps. En France, par exemple, la loi relative à l'informatique, aux fichiers et aux libertés, dite *loi informatique et libertés*, date de 1978 et pose l'existence d'un certain nombre de principes fondamentaux liés au traitement des données personnelles, protégeant ainsi la vie privée des citoyens. Parmi ces nombreux principes, l'un d'entre eux régit le *traitement* des données personnelles : la collecte limitée de données personnelles en vue d'un traitement informatique (voir en encadré).

Traitement des données personnelles

Un *traitement de données personnelles* est un terme général, correspondant à un traitement automatique (informatique) ou non portant sur des données personnelles, quel que soit le procédé utilisé, et notamment la collecte, l'enregistrement, l'organisation, la conservation, l'adaptation ou la modification, l'extraction, la consultation, l'utilisation, la communication par transmission, diffusion ou toute autre forme de mise à disposition, le rapprochement ou l'interconnexion, ainsi que le verrouillage, l'effacement ou la destruction.

En effet, la loi de 1978 énonce dans ses articles 6 (sur la collecte et conservation des données) et 7 (sur le consentement) les conditions de licéité d'un traitement (informatique). On voit que la *finalité* du traitement doit être « *déterminée, explicite et légitime* », que les données traitées doivent être « *adéquates, pertinentes et non excessives* », mais également « *exactes et complètes* ». En d'autres termes, il n'est pas légal de collecter des données sans savoir si elles seront utiles ou pas pour le processus. Or, nous allons voir que ce principe (juridique) de *minimalité* des données traitées est très difficile à mettre en place de manière pratique puisqu'il pose des problèmes mathématiques (informatiques) complexes : il est en effet NP-difficile de décider, compte tenu d'un traitement modélisé sous la forme de règles logiques permettant une prise de décision multicritère, si un ensemble de données collecté est minimal ou non. À un problème de décision NP-complet (voir l'article consacré à ce thème dans ce hors-série) qui cherche à déterminer s'il existe une solution de taille n à un problème, on peut lier un problème d'optimisation qui cherche à

calculer la plus petite valeur acceptable pour n ; ce nouveau problème est qualifié dans ce cas de NP-difficile, et sa résolution est au moins aussi coûteuse que la résolution du problème de décision NP-complet associé.

Pour illustrer le problème, prenons comme exemple de traitement automatique de données personnelles un système informatique permettant d'attribuer des prêts bancaires personnalisés. Une banque propose des prêts à la consommation de 5 000 dollars à 10 % sur trois ans, personnalisables selon quatre critères : montant plus élevé du prêt, réduction du taux, augmentation de la durée de remboursement et réduction de l'assurance liée au prêt. La décision d'attribuer chacun de ces quatre critères est prise sur la base d'un ensemble de règles logiques, dit *modèle métier* de la banque, exprimées avec des variables Booléennes $b_{k,i,j}$, où chaque règle r_k permettant d'obtenir un bénéfice c_k est en forme normale disjonctive, et où certaines variables booléennes $b_{k,i,j}$ interviennent dans plusieurs règles. Dans l'exemple de l'encadré ci-contre, on a $b_{1,1,1} = b_{3,1,1} = b_{4,2,1} = p_1$, $b_{1,1,2} = b_{3,2,1} = b_{4,1,1} = p_2$ et ainsi de suite. Il est intéressant de noter que, dans de nombreux cas, les règles sont issues de techniques de fouille de données (*data mining*) sur la base de décisions prises précédemment par des experts.

Considérons la règle r_1 :
$(p_1 \wedge p_2) \vee (p_3 \wedge p_4) \Rightarrow c_1$.
Elle exprime le fait qu'un client peut obtenir un prêt plus élevé (l'avantage c_1) si son revenu annuel est supérieur à 30 000 $ et si son patrimoine est supérieur à 100 000 $ ou bien s'il possède une assurance vie et un nantissement supérieur à 100 000 $. Afin de bénéficier de cet avantage, un client doit exposer un ensemble d'assertions logiques $as_1, as_2, as_3 \ldots$ supposées de la forme *attribut* = *valeur*, tel que cet ensemble permet de prouver que le corps de la règle $(p_1 \wedge p_2) \vee (p_3 \wedge p_4)$ est vrai.
Dans l'exemple, $as_1 \Rightarrow p_1$, $as_2 \Rightarrow p_2$, $as_3 \Rightarrow p_3$ et $as_4 \Rightarrow p_4$. Il est donc évident que le corps de règle r_1 est vrai, et donc que ce client peut bénéficier d'un montant de prêt plus élevé. Toutefois, il est inutile que le client transmette l'intégralité de ces informations pour bénéficier de cet avantage (on dit également qu'il *expose* ces informations). Il est en réalité nécessaire et suffisant de ne transmettre que $\{as_1, as_2\}$ ou $\{as_3, as_4\}$ afin de limiter la collecte de données personnelles (pour simplifier, on considère que chaque attribut a la même sensibilité qu'un autre ; il est ainsi aussi « problématique » pour le client de transmettre son âge que son adresse). On préfère donc transmettre le plus petit nombre d'attributs.

Un exemple de prêt bancaire

Règles de collecte :

$r_1 : (p_1 \wedge p_2) \vee (p_3 \wedge p_4) \Rightarrow c_1$
$r_2 : (p_5 \wedge p_6 \wedge p_7) \vee (p_4 \wedge p_8 \wedge p_9) \Rightarrow c_2$
$r_3 : (p_1 \wedge p_6 \wedge p_7) \vee (p_2 \wedge p_4 \wedge p_{10}) \Rightarrow c_3$
$r_4 : (p_2 \wedge p_5 \wedge p_6 \wedge p_7) \vee (p_1 \wedge p_4 \wedge p_8 \wedge p_9) \Rightarrow c_4$

avec

p_1 : revenu_annuel > \$30.000, p_2 : patrimoine > \$100.000,
p_3 : nantissement > \$100.000, p_4 : assur_vie = 'oui',
p_5 : taux_impots > 10 %, p_6 : marié = vrai,
p_7 : enfants > 0, p_8 : edu = 'université',
p_9 : age < 30, p_{10} : frais_sinistres < \$5.000

et

c_1 = prêt_élevé, c_2 = taux_5 %,
c_3 = prêt_long, c_4 = assurance_réduite.

Assertions Client :

as_1 : revenu_annuel = \$35.000, as_2 : patrimoine = \$150.000,
as_3 : nantissement = \$175.000, as_4 : assur_vie = 'oui',
as_5 : taux_impots = 11.5 %, as_6 : marié = vrai,
as_7 : enfants = 1, as_8 : edu = 'univ',
as_9 : age = 25, as_{10} : frais_sinistres = \$250.

L'article 6 de la loi informatique et libertés

« Un traitement ne peut porter que sur des données à caractère personnel qui satisfont aux conditions suivantes :

1) Les données sont collectées et traitées de manière loyale et licite ;
2) Elles sont collectées pour des finalités déterminées, explicites et légitimes et ne sont pas traitées ultérieurement de manière incompatible avec ces finalités. Toutefois, un traitement ultérieur de données à des fins statistiques ou à des fins de recherche scientifique ou historique est considéré comme compatible avec les finalités initiales de la collecte des données, s'il est réalisé dans le respect des principes et des procédures prévus au présent chapitre, au chapitre IV et à la section 1 du chapitre V ainsi qu'aux chapitres IX et X et s'il n'est pas utilisé pour prendre des décisions à l'égard des personnes concernées ;
3) Elles sont adéquates, pertinentes et non excessives au regard des finalités pour lesquelles elles sont collectées et de leurs traitements ultérieurs ;
4) Elles sont exactes, complètes et, si nécessaire, mises à jour ; les mesures appropriées doivent être prises pour que les données inexactes ou incomplètes au regard des finalités pour lesquelles elles sont collectées ou traitées soient effacées ou rectifiées ;
5) Elles sont conservées sous une forme permettant l'identification des personnes concernées pendant une durée qui n'excède pas la durée nécessaire aux finalités pour lesquelles elles sont collectées et traitées.

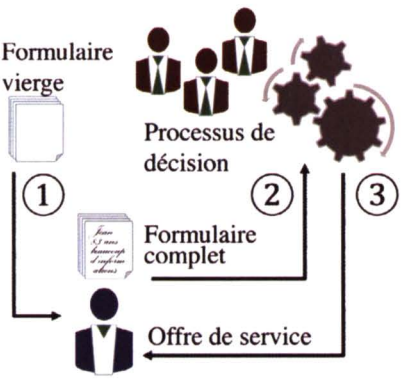
S'il n'y a qu'une seule règle de collecte, le problème est simple : il suffit de transmettre la conjonction impliquant le moins d'attributs. Le problème devient plus difficile lorsque le client souhaite bénéficier de *plusieurs* avantages c_k simultanément.

Afin de définir formellement le problème mathématique de la minimisation des données collectées (dit également *problème de n-exposition*), posons $E_R = \wedge_k(r_k) = \wedge_k(\vee_i(\wedge_j b_{k,i,j}))$, appelée *formule booléenne de l'ensemble de règles*. Le problème s'énonce formellement de la manière suivante :

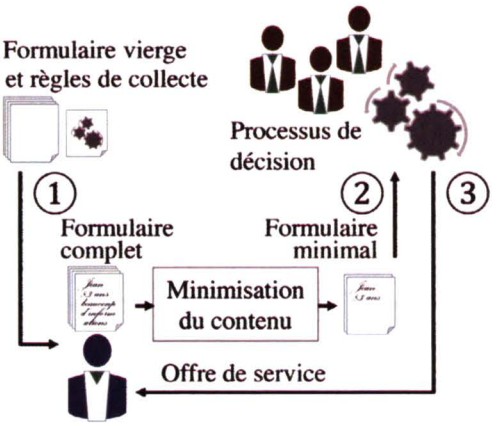
On se donne un ensemble de règles $R = \{r_1, r_2, r_3 \ldots\}$, un ensemble d'assertions $\text{data}_u = \{as_1, as_2, as_3 \ldots as_q\}$ tel que tous les r_k sont vrais, un ensemble de variables booléennes $B = \{p_1, p_2 \ldots p_q\}$ tel que p_m = vrai $\Leftrightarrow as_m$ est transmis, et enfin la formule booléenne de l'ensemble de règles R, notée $E_R = \wedge_k(\vee_i(\wedge_j b_{k,i,j}))$ où, quels que soient les indices k, i et j, $b_{k,i,j} \in B$. On dit que data_u est *n-exposable par rapport à* R si, et seulement si, il existe une affectation de valeurs booléennes aux variables de B telle que E_R est vraie, et le nombre de $p_1, p_2 \ldots p_q$ prenant la valeur vraie est inférieur n.

Pour les données présentées dans l'encadré précédent, on peut facilement vérifier que la solution $T_B = \{p_1 = V, p_2 = V, p_3 = F, p_4 = F, p_5 = V, p_6 = V, p_7 = V, p_8 = F, p_9 = F, p_{10} = F\}$ valide bien E_R, et donc que R est 5-exposable. Pour montrer que 5 est bien la plus petite valeur acceptable, une manière est de tester toutes les affectations ayant quatre valeurs vraies (il en existe C_{10}^4 soit 210), et montrer qu'aucune ne convient.

Il est possible de montrer que ce problème, dans sa généralité, est NP-complet, en se fondant sur une réduction à un problème assez connu en logique : le *problème de satisfiabilité minimale avec pondération* (ou Min Weighted Sat). On définit le problème d'optimisation associé, qui cherche à trouver le plus petit nombre d'éléments de B (on notera m cette valeur) qu'il faut fixer à vrai pour que E_R soit vraie. Le problème d'optimisation est NP-difficile, ce qui signifie qu'il peut être très coûteux de trouver le résultat exact à cette question pour de grandes valeurs de m. Pour calculer la solution exacte du problème, on peut utiliser une méthode de force brute,



Processus classique
Le demandeur renseigne l'intégralité du formulaire d'application, transmettant ainsi toutes les informations identifiées comme ayant un impact potentiel sur le processus de prise de décision.



Processus à exposition minimale.
Les informations renseignées par le demandeur sont minimisées conformément aux règles de collecte, ce qui conduit à prendre la même décision, à partir d'un formulaire contenant (beaucoup) moins d'informations.

Processus de prise de décision à exposition minimale d'information.

Application : l'attribution de l'aide sociale

En France, les Conseils généraux sont responsables d'attribuer une aide sociale aux personnes dépendantes, sous forme de support financier, matériel ou humain. L'aide est sollicitée au travers d'un ensemble de formulaires à renseigner décrivant la situation précise de la personne dépendante, et comportant plusieurs centaines de champs à saisir avec l'aide de son médecin, de son assistant social et de son entourage. Ainsi, les Conseils généraux peuvent traiter chacun plusieurs dizaines de milliers de demandes par an, nécessitant l'intervention de centaines d'employés dépouillant les formulaires pour extraire et vérifier les informations, et décider de la forme que devra prendre l'aide afin de répondre au mieux à la situation du requérant. Un processus identifiant, en amont du traitement de la demande, l'ensemble minimal d'assertions permettant de conduire à la décision limite la quantité de données personnelles exposées par le requérant conformément à la loi, et accélère le traitement des demandes.

Le défi. Voici un jeu de règles construit avec l'aide du Conseil général des Yvelines, impliquant 440 prédicats et 63 bénéfices potentiels. Les données sont factices pour des raisons de confidentialité, mais respectent la topologie des règles réelles. Ces règles ont été transposées au format AMPL (données disponibles sur le site https://project.inria.fr/minexp/, rubrique Challenge). Le solveur Couenne lancé sur la modélisation non linéaire ne parvient pas à trouver une solution minimale, même après plusieurs jours de fonctionnement. L'algorithme RAND* permet d'obtenir une solution en quelques secondes (230 éléments sont conservés sur 440, avec N = 1 000, et 215 éléments avec N=100 000). Cette solution se situe probablement assez loin de la solution optimale. Nos lecteurs sont invités à développer d'autres algorithmes basés sur des heuristiques plus efficaces, permettant de parvenir à une solution plus proche de l'optimale, tout en restant très rapides !

Références

- *Exposition minimum de données pour des applications à base de classifieurs*. Nicolas Anciaux, Benjamin Nguyen et Michalis Vazirgiannis, *Ingénierie des Systèmes d'Information* 18(4), 2013.
- Le défi : https://project.inria.fr/minexp/, rubrique Challenge (en anglais).

Un exemple de programme en AMPL

```
var b1 binary; var b2 binary; ... var b10 binary;
minimize EX:
b1 + b2 + b3 + b4 + b5 + b6 + b7 + b8 + b9 + b10;
subject to
r1 : b1*b2 + b3*b4 >= 1;
r2 : b5*b6*b7 + b4*b8*b9 >= 1;
r3 : b1*b6*b7 + b2*b4*b10 >= 1;
r4 : b2*b5*b6*b7 + b1*b4*b8*b9 >= 1;
```

L'algorithme approché RAND*

Entrées : ensemble de K règles r_k contenant q prédicats distincts p_i
nombre de passes N
Sortie : meilleur ensemble de valeurs de vérité p_i trouvé
Variables : $p\_optimal$, un tableau de q booléens
x, une variable de type entier

1. Mettre à VRAI toutes les valeurs de $p\_optimal_i$
2. Pour *compteur* allant de 1 à N faire
3. Mettre toutes les valeurs de p_i à FAUX
4. Pour k allant de 1 à K faire
5. $x \leftarrow$ une valeur aléatoire comprise entre 1 et le nombre de disjonctions dans r_k
6. Pour j allant de 1 au nombre de prédicats dans la $x^{ème}$ disjonction de r_k faire
7. mettre à VRAI la valeur du prédicat p_i correspondant à $b_{k,x,j}$
8. FinPour
9. FinPour
10. Si le nombre de prédicats p_i mis à VRAI est plus petit que le nombre de prédicats $p\_optimal_i$ mis à VRAI, alors
11. Pour i allant de 1 à q faire
12. $p\_optimal_i \leftarrow p_i$
13. FinPour
14. FinSi
15. FinPour
16. Retourner la liste des $p\_optimal_i$

c'est-à-dire tester toutes les solutions possibles : il y en a très exactement 2^q où q représente le nombre de prédicats booléens du problème. Le coût de cette méthode est donc hautement prohibitif.

Une autre manière de faire est d'utiliser un outil de résolution exact de problèmes d'optimisation. Le problème doit alors être écrit dans un langage particulier : le langage AMPL (langage de modélisation pour la programmation mathématique), qui permet d'exprimer une fonction à minimiser par rapport à un ensemble de variables (ici les prédicats booléens) et de contraintes (ici les règles). Un exemple d'un tel programme est donné en encadré. Les contraintes contiennent des multiplications, ce qui rend le problème non-linéaire. Tant que le nombre de variables n'est pas trop important (disons une centaine), ce programme peut être résolu par des logiciels comme Couenne (pour Convex over and under envelopes for non-linear estimation, disponible en ligne et *open-source*).

Une dernière approche est de proposer des algorithmes de complexité polynomiale, et donc faciles et rapides à calculer, permettant de trouver un résultat approché de la solution. L'encadré qui suit donne l'exemple d'un algorithme aléatoire, nommé RAND*, très simple, qui permet de calculer une solution acceptable au problème, en générant aléatoirement N solutions acceptables, puis en choisissant la meilleure. Toutefois, la minimalité de la solution n'est aucunement garantie !

Le lecteur est invité à chercher d'autres algorithmes pour résoudre le problème de manière approchée, et à comparer, à temps de calcul égal, leur qualité avec l'algorithme naïf RAND*. Ainsi, on voit que le problème de limiter la collecte de données est compliqué, puisque la simple identification d'une donnée potentiellement utile est un problème informatique difficile et coûteux en temps !

N. A. & B. N.

L'intelligence artificielle pour jouer aux échecs

Deep Blue, le premier à battre les champions

Alan Turing et Claude Shannon ont commencé dans les années 1950 les premiers travaux sur l'intelligence artificielle appliquée au jeu d'échec. Mais c'est à la fin des années 1990 qu'IBM marqua l'histoire en créant Deep Blue, une machine conçue spécifiquement pour jouer à ce jeu. En 1997, avec la victoire de Deep Blue sur le célèbre champion Kasparov, une machine se classait devant les humains lors d'un tournoi du plus haut niveau.
Depuis, de nombreux progrès ont été réalisés, permettant de mettre au point plusieurs programmes solides qui fonctionnent sur des machines personnelles (ordinateurs ou téléphones) et qui battraient aujourd'hui Deep Blue. Ces progrès sont dus à de grandes avancées importantes en algorithmique.

Le logiciel Chess sur iPad.

L'algorithme du logiciel Chess

La société Magma Mobile a suivi les traces de Deep Blue pour développer Chess, son Intelligence Artificielle pour téléphone, disponible sur http://m.magmamobile.com/Chess/
Les programmes d'échecs reposent tous sur le même principe : une fonction qui peut donner :

- une estimation de la valeur d'un plateau (l'estimation du plateau peut être le nombre de pièces blanches pondérées par leurs valeurs moins le nombre de pièces noires pondérées par leurs valeur).
- une fonction de recherche (voir en encadré le programme). Cette fonction simule chaque mouvement possible sur le plateau, et se rappelle récursivement. Quand elle arrive à une profondeur donnée ou à une fin de partie, elle décide d'une valeur à renvoyer.

Si on compte les plateaux d'échecs sur lesquels ont été joués moins de huit coups, on obtient déjà 84 milliards de plateaux. Ce qui veut dire qu'il faudrait faire 84 milliards d'appels à la fonction de recherche pour simuler seulement huit coups.
Malgré la puissance de calcul des ordinateurs modernes, ce nombre est trop grand : l'intelligence artificielle mettrait trop longtemps à jouer. Il existe heureusement un grand nombre de raffinements, dont le plus connu est l'*alphabeta*.

Le temps de calcul de cet algorithme dépend évidement de la profondeur et du nombre de coups par plateau, mais aussi de l'ordre de recherche des coups. Il permet néanmoins de parvenir à un résultat en un temps acceptable pour les joueurs.

LE PROGRAMME « RECHERCHE »

```
fonction recherche (p; plateau)
si la partie est finie alors renvoyer le score
si p = 0 alors
    renvoyer une estimation du score
soit max = +∞
pour tous les plateau_i faire
    soit score = −recherche(p − 1; plateau_i)
    max = max(score; max)
fin
renvoyer max
```

par J. Nicolas, C. Belleannée et F. Coste

Le langage des molécules du vivant

Les textes du génome des êtres vivants affluent. Il ne reste plus qu'un détail : en comprendre le sens ! Les outils informatiques doivent aller plus loin que la reconstitution des mots : comprendre comment leur assemblage sera interprété par les *machines* qui jouent un rôle prédominant dans les cellules vivantes.

Les auteurs sont chercheurs à IRISA / INRIA Centre de Rennes Bretagne Atlantique.

Depuis une quinzaine d'années, la biologie dispose de moyens d'observation sans précédent pour accéder au niveau moléculaire le plus intime à l'ensemble des informations présentes dans une cellule. La question centrale de la biologie est de passer de cette information de bas niveau à la compréhension des fonctions essentielles du vivant : la reproduction, l'autoréparation, l'organisation en structures emboîtées à l'aide de membranes isolant les systèmes cruciaux, et enfin l'évolution et l'adaptation au milieu de vie qui permettent aux organismes d'affronter des conditions d'environnement très variées.

Voyage au cœur des cellules.

Les données génomiques : ADN, ARN et protéines

Mais de quelles informations dispose-t-on exactement ?

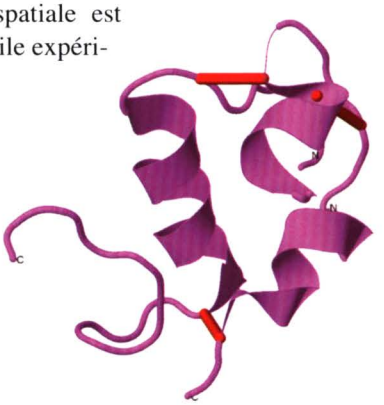
Le déferlement des données génomiques concerne non seulement le code des chromosomes humains – obtenu en 2003 –, mais également celui d'un nombre croissant d'organismes, des plus petits, les virus et les bactéries, aux plus gros comme l'éléphant africain. À la base, il y a donc *la cellule*, la brique de base de tout organisme vivant, et au cœur de la cellule, ces longues molécules d'ADN qui forment les *génomes* et qui recèlent toutes les données nécessaires à la maintenance et à la reproduction de ces cellules.

En termes mathématiques, un génome peut être vu de manière simplifiée comme une (grande) séquence écrite sur un alphabet à quatre lettres, *les bases a, c, g* et *t*. C'est un peu plus compliqué, la séquence est orientée (le texte a un sens de lecture) et c'est un agent double : l'ADN a deux brins enlacés, qui sont codés de manière

totalement complémentaire, c'est-à-dire en sens opposé et en associant les lettres qui se font face selon un schéma rigide de paires *a-t* ou *g-c*.

Nous savons maintenant que l'information nécessaire à la compréhension des comportements vivants est loin de s'arrêter aux chromosomes. La molécule d'ADN est une sorte de conservatoire, elle code et retient l'information héréditaire comme un notaire veillant sur le patrimoine de ses clients, mais il faut bien avouer qu'elle n'est pas très active. Une première molécule semblable à l'ADN mais beaucoup plus dynamique est l'ARN, probablement apparue très tôt dans la soupe primordiale de molécules ayant conduit aux formes de vie terrestre. Les molécules d'ARN sont les agents de traduction du message génétique en mesures concrètes et adaptées à l'intérieur de la cellule. Ainsi, les segments d'ADN d'une entité appelée *gène* pourront donner naissance à différentes variantes en ARN (appelées *transcrits*) suivant le contexte dans lequel se trouve la cellule. L'ARN participe également à la formation des machines moléculaires qui vont traduire puis réguler l'expression des gènes pour former des molécules actives. Formellement, l'ARN est aussi un texte orienté à quatre lettres, *a, c, g* et *u*. Simple brin cette fois et plus court, il peut se replier dans l'espace en s'appariant sur lui-même (paires *a-u* ou *g-c*) et prendre des formes variées caractéristiques de sa fonction. On sait maintenant observer finement la présence d'ARN dans les cellules. La structure spatiale est difficilement accessible expérimentalement et on s'appuie en général sur la modélisation mathématique pour la découvrir.

Il nous faut évoquer enfin le troisième grand type de (macro-)molécule des cellules, les *protéines*, qui sont les agents à tout faire de la cellule : elles ont un rôle structurel, elles s'occupent du transport des molécules, de la signalisation des évènements extérieurs, de la mobilité, elles catalysent les réactions chimiques et régulent l'expression des gènes. Les protéines sont produites par traduction d'un type d'ARN dit *messager*. Les textes en résultant sont codés sur un alphabet, les *acides aminés* (une vingtaine de lettres) et ne dépassent pas quelques milliers de caractères. Les propriétés physico-chimiques de ces molécules sont beaucoup plus variées que celles des molécules d'ADN et d'ARN, de même que les structures qu'elles adoptent spontanément dans l'espace. On sait observer l'ensemble des protéines présentes dans un échantillon. Connaître leur structure spatiale est un problème à la fois difficile expéri-



Dans une protéine : insuline.
Les protéines de la famille de l'insuline partagent toutes une structure spatiale commune de deux chaînes liées d'acides aminés, de séquence C_1-C_2- x(3)-C_3-[STDNEKPI]-x(3)-[LIVMFS]-x(3)-C_4, et C_5-x(12)-C6, où x(k) est un mot quelconque de taille k. De plus, les cystéines C_i forment des liaisons, les ponts disulfures (en rouge dans l'image) qui rigidifient la forme de la molécule dans l'espace : C_1 est reliée à C_3, C_2 est reliée à C_5, et C_3 est reliée à C_6.
Source : Auteur (J.Nicolas) de la base de données PDB

mentalement et fondamental en pratique car la fonction des protéines est intimement liée à cette structure. Comme pour l'ARN, la modélisation mathématique et informatique se révèle donc indispensable à leur étude, en particulier pour prédire la structure à partir du texte.

Des textes naturels et troublants...

Pour un informaticien, observer la complexité des mécanismes du vivant est une source infinie d'étonnement et d'inspiration. Car il ne s'agit pas uniquement de physique et de chimie. Les différents *textes* à l'intérieur des cellules, qui sont le fruit d'une longue maturation tout au long de l'évolution, ont un impact majeur sur le comportement cellulaire. Dans des cas extrêmes, le changement d'une seule base d'ADN conduit à des états pathologiques, comme c'est le cas pour la première maladie génétique de France, la *drépanocytose*, où la mutation d'une base d'un gène entraîne un changement d'acide aminé de la protéine associée, déformant les globules rouges.

Si on regarde ces textes d'un peu plus près, on s'aperçoit qu'ils sont tout sauf aléatoires. La première impression qui frappe est l'abondance de répétitions plus ou moins exactes qui peuplent les séquences biologiques. On sait maintenant qu'il existe des mécanismes dans les cellules commandés par des protéines propres – les transposases – ou importées de virus – les intégrases – qui permettent de copier certaines portions de génome, les *transposons*, à un autre endroit dans le génome.
D'un point de vue informatique, il s'agit ni plus ni moins d'une version du copier/coller ou du couper/coller des ordinateurs ou des mobiles, à ceci près que le texte copié contient aussi le programme de copie ! C'est un mécanisme très efficace qui provoque la croissance des génomes et apporte un éclairage sur leur étonnante plasticité et leur capacité à évoluer et à se diversifier au-delà de ce que permettraient les seuls accidents de transmission du patrimoine génétique (les mutations) et les recombinaisons entre individus lors de la reproduction. Le record de croissance, $1{,}5 \times 10^{11}$ bases d'ADN, est actuellement détenu par une plante, *Paris japonica*, qui pousse sur les montagnes de l'île de Honshu (le génome humain n'a « que » 3×10^9 bases).

De façon générale, il existe d'autres mots que l'on retrouve systématiquement dans ces textes et qui les charpentent, presque toujours associés à des machines plus ou moins complexes (à base d'ARN et de protéines), qui vont permettre la lecture et la transformation des textes, un peu à la manière des têtes de lecture/écriture de nos enregistreurs électroniques. On pourrait multiplier les exemples. Ainsi, les fameux gènes, unités génomiques qui vont être traduites en protéines, sont découpés chez les organismes supérieurs en tronçons appelés exons et introns qui s'alternent le long de la molécule. Le texte utile pour la protéine sera constitué d'une sélection d'exons mis bout à bout. Sans comprendre en détails comment la machine associée, le *splicéosome*, effectue cette sélection, on connaît déjà des mots qui se retrouvent majoritairement aux jonctions entre les tronçons : *gu* au début et *agg* à la fin des introns par exemple.

Pour le passage du texte en ARN au texte en acides aminés, la machine s'appelle ribosome et le début du texte à traduire commence par *aug* et parfois *gug* ou *uug* et finit invariablement par *uag*, *uga* ou *uaa*. Systématiquement, les triplets de lettres d'ARN vont être transformés en une lettre d'acide aminé. On observe là des phénomènes très différents de ce qui se passe en chimie classique : de véritables codes sont interprétés de manière rigoureuse dans la cellule. Autre exemple de machine en action sur les cellules germinales ou les cellules cancéreuses, les *télomérases*. Il s'agit d'un assemblage de protéines et d'ARN chargé d'ajouter une séquence répétée spécifique à l'extrémité des chromosomes, pour éviter de perdre des gènes lors de leur réplication pour produire de nouvelles cellules.

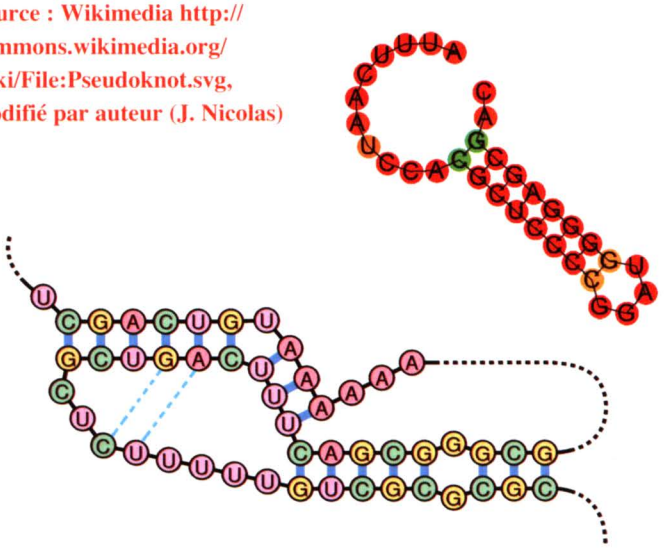
Il reste encore de telles machines à découvrir au cœur des cellules. Parmi les découvertes récentes, évoquons juste les structures de *CRISPR*, au nom improbable mais dont l'existence est encore plus surprenante : on trouve dans bon nombre de bactéries des séries de mots courts répétés qui forment une ossature dans laquelle vont s'intercaler d'autres mots très différents, qui semblent issus d'une langue étrangère à la bactérie. En fait, tout comme nous, les bactéries sont sujettes à des attaques de virus et cette structure se comporte comme un ensemble de cases mémoires, qui retiennent des morceaux de séquences de virus rencontrés et les reconnaissent lors de futures attaques : ce n'est ni plus ni moins qu'un système immunitaire auquel sont associés un ensemble de mots et une machinerie spécifique de gènes associés.

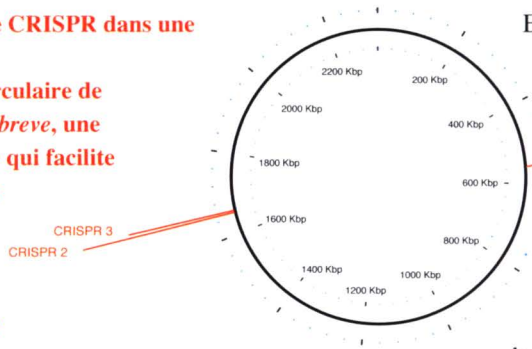
De façon claire et remarquable, les mots ainsi que leur assemblage et leur interprétation par des machines spécialisées jouent un rôle prédominant dans les cellules vivantes. Au-delà des phénomènes de thermodynamique, d'électrostatique ou autres lois à caractère continu, il existe des représentations et des comportements de nature symbolique qui contrôlent le devenir des cellules.

Langages, structures et interactions

Des mots qui s'enchaînent de façon contrôlée, y aurait-il donc un langage, voire des langages à l'œuvre dans les organismes ? Précisons les termes employés : il y a généralement deux niveaux d'écriture dans une langue, le

Dans un ARN : télomérase.
Une structure de pseudo-nœud dans la partie ARN de la télomérase humaine : cette structure permet de positionner une matrice qui va générer des répétitions TTAGGG à la fin des chromosomes et éviter le raccourcissement de leurs extrémités lors de leur réplication (mécanisme également utilisé dans les cancers). En pointillés, une structure alternative.
Source : Wikimedia http://commons.wikimedia.org/wiki/File:Pseudoknot.svg, modifié par auteur (J. Nicolas)





Une structure de CRISPR dans une bactérie
Chromosome circulaire de *Bifidobacterium breve*, une bactérie lactique qui facilite la digestion chez les nourrissons. Emplacement de CRISPR et de la machinerie associée.

niveau lexical qui est celui des mots du dictionnaire, et le niveau syntaxique, qui est celui des enchaînements possibles au sein de phrases.

L'ensemble des mots est généralement fini et on parlera ici de *vocabulaire* pour indiquer l'ensemble des symboles manipulés. Le vocabulaire peut être constitué de simples caractères (des idéogrammes comme les hiéroglyphes, ou des acides aminés pour les protéines) ou de succession de caractères (un mot en français, une balise HTML), peu importe, on le suppose figé. Une façon particulièrement simple de décrire un langage est d'établir la liste des phrases qu'il contient, c'est-à-dire des combinaisons permises d'éléments du vocabulaire. Mais contrairement aux vocabulaires, les langages peuvent la plupart du temps être considérés comme infinis. On ne peut donc pas se contenter d'énumérer leurs éléments. L'informatique a été très tôt confrontée à ce problème car il y a un lien intime entre langages et calcul. Après tout, la logique mathématique n'est rien d'autre que l'étude des mathématiques en tant que langage et un ordinateur est une machine de manipulation de langages. La *théorie des langages* a pour but de décrire formellement la notion de langage. Elle utilise un outil commun des linguistes et des informaticiens pour générer avec un système formel fini un nombre potentiellement non borné de phrases : les grammaires. Formellement, une grammaire est un ensemble de règles de récriture qui repose à la fois sur l'utilisation d'un ensemble de symboles spécifiques appelé non-terminaux et d'un vocabulaire dont les éléments sont dits terminaux. Pour générer toutes les phrases du langage, il suffit de transformer itérativement une chaîne réduite initialement à un symbole fixé, l'axiome, et d'appliquer de toutes les manières possibles les règles en changeant une occurrence d'une partie gauche de la règle dans cette chaîne par la partie droite de la règle (voir en encadré un exemple de ce qu'on appelle un langage *régulier*).

Cependant la classe des langages réguliers reste trop limitée pour décrire ce qui se passe en biologie. Il n'est par exemple pas possible de décrire les structures en forme de tige qui naissent lors du repliement de l'ARN dans l'espace pour former des doubles brins, comme dans certaines structures de CRISPR (une structure qui ressemble fort aux palindromes du français). Pour cela, il faut pouvoir mettre en correspondance des symboles éventuellement distants. On étend donc les règles de grammaires régulières en permettant un assemblage quelconque de terminaux et de non terminaux dans la partie droite. Comme les non terminaux se récrivent directement où qu'ils soient, on parle de *grammaire hors contexte*.

Grammaires « régulières »

En prenant S comme axiome, une grammaire possible de génération d'une séquence d'ARN bactérien codant pour une protéine est la suivante :
$\{\mathbf{1}: S \to aX_1,\ \mathbf{2}: X_1 \to uX_2,\ \mathbf{3}: X_2 \to gX_3\} \cup \{\mathbf{4}: X_3 \to \alpha X_4,\ \mathbf{5}: X_7 \to \alpha X_4 \mid \alpha \in \{a,c,g\}\} \cup \{\mathbf{6}: X_4 \to \beta X_6,\ \mathbf{7}: X_6 \to \beta X_7 \mid \beta \in \{a, c, g, u\}\} \cup \{\mathbf{8}: X_5 \to \gamma X_6,\ \mathbf{9}: X_8 \to \gamma X_7 \mid \gamma \in \{c, g, u\}\} \cup \{\mathbf{10}: X_3 \to uX_5,\ \mathbf{11}: X_7 \to uX_5,\ \mathbf{12}: X_5 \to aX_8, \mathbf{13}: X_8 \to a\}$.

Elle pourra par exemple générer la séquence *augccguaa* en utilisant successivement les règles 1, 2 et 3 (*aug*), puis 4, 6 et 7 (*ccg*), et enfin 11, 12 et 13 (*uaa*). Observez que la structure des règles est très régulière : elles sont toutes de la forme « un non terminal se récrit en un symbole terminal suivi éventuellement d'un non terminal ». On nomme *réguliers* les langages générés avec cette forme de règles. Ces langages ont une foule de bonnes propriétés qui en font un outil précieux en informatique (système Unix, traitements de texte...). Ainsi, savoir si une phrase appartient à un langage régulier demande un nombre d'opérations proportionnel à la taille de la phrase. De plus ils forment une classe stable au sens où l'intersection, l'union, le complémentaire, la différence ou l'application d'un homomorphisme sur un langage régulier continuent à donner un langage régulier.

Les langages peuvent aussi être décrits, de façon équivalente, à l'aide de machines : c'est un modèle plus proche de ce qui se passe en biologie où de nombreuses machines sont en œuvre ainsi qu'en informatique où on définit plusieurs types de machines abstraites en fonction du type de langages, la plus générale, la *machine de Turing*, étant le fondement de nos ordinateurs.
Pour reconnaître un langage régulier, on utilise ainsi ce qu'on appelle des *automates d'états finis*. La machine part d'un état initial et lit les symboles de la phrase de gauche à droite. En utilisant une fonction de transition fixée qui associe à chaque état et chaque symbole lu un nouvel état, la machine progresse tant que c'est possible d'états en états. La phrase est reconnue si la machine termine dans un état final. On représente graphiquement les états par des cercles, un état final par un double cercle et une transition en lisant un symbole par une flèche depuis l'état de départ jusqu'à l'état d'arrivée surmontée du symbole. Nous effleurons juste en passant la notion de probabilité qu'on peut introduire dans les langages et leurs représentations : rien n'empêche de considérer un langage comme une distribution de probabilités sur l'ensemble des enchaînements possibles. D'un point de vue grammaire ou machine, ceci suppose d'associer des probabilités aux règles ou aux transitions. Nous nous contentons ici de considérer que toutes les phrases ont la probabilité 0 ou 1. En pratique, il peut exister en biologie différentes alternatives d'analyse (on parle d'ambiguïté) avec des probabilités différentes pour de mêmes phrases, comme par exemple dans le cas des télomérases qui oscillent entre deux états stables.

Ainsi, une grammaire hors contexte pour la reconnaissance de tiges-boucles dans l'ARN peut être décrit par deux ensembles de règles (l'un pour la tige, le deuxième pour la boucle) :

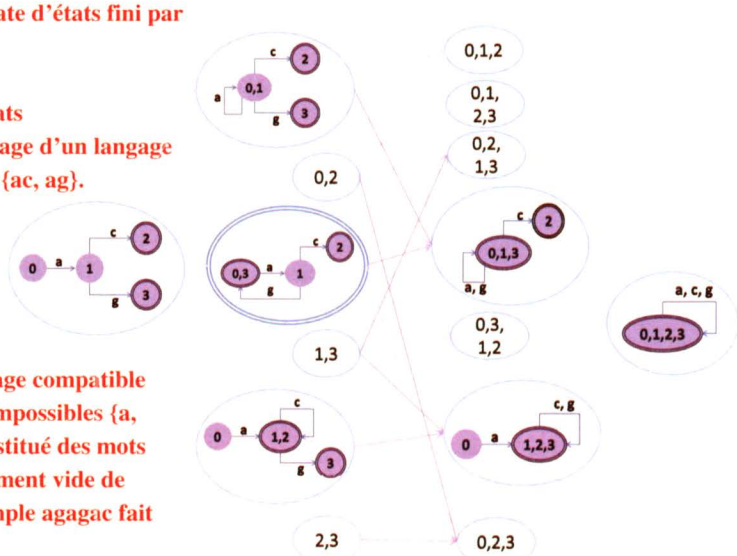
$\{S \to aSu,\ S \to cSg,\ S \to gSc,\ S \to uSa\} \cup \{S \to \alpha X,\ X \to \alpha \mid \alpha \in \{a,c,g,t\}\}$.

De même, il faut améliorer la machine précédente en ajoutant une mémoire

Découvrir un langage, c'est automatisable ?

L'acquisition d'un langage, c'est un jeu d'enfant : l'essentiel est achevé avant l'âge de 3 ans pour la plupart. Si on arrive à apprendre n'importe quelle langue naturelle, ne peut-on arriver à apprendre cette autre langue naturelle qu'est le langage du génome ? On est cependant loin de comprendre les mécanismes internes qui permettent cette acquisition ni même de savoir comment nous représentons mentalement les langages et il nous faut inventer une approche rationnelle de l'apprentissage. Notons que la démarche n'est pas tout à fait celle d'un Champollion déchiffrant les hiéroglyphes : on ne dispose pas d'un langage de référence sur lequel s'appuyer pour comprendre un nouveau langage, il s'agit bien d'apprendre un nouveau langage simplement à partir d'un échantillon de phrases.
On sait actuellement assez bien modéliser l'apprentissage de langages réguliers. Une idée simple mais mathématiquement intéressante est de partir d'un langage qui reconnaît uniquement les phrases autorisées. C'est par exemple l'automate de gauche dans la figure. Ensuite, on peut généraliser le langage correspondant (c'est-à-dire accepter plus de phrases) en fusionnant deux états quelconques (ils deviennent égaux et on conserve l'union des transitions auxquelles ils participent). La structure mathématique de l'ensemble des possibilités est ce que l'on appelle un *treillis*, c'est-à-dire un ensemble partiellement ordonné où tout couple d'éléments admet une borne supérieure et une borne inférieure. Si on considère l'ensemble de tous les états E, c'est en fait le treillis des partitions sur E, c'est-à-dire l'ensemble des façons de partitionner E en sous-ensembles distincts, ordonné par l'inclusion sur les partitions. Généraliser, c'est aller de la gauche vers la droite dans le treillis. Pour savoir jusqu'où le faire et éviter de reconnaître n'importe quel enchaînement (automate de droite sur la figure), on peut par exemple dialoguer avec l'utilisateur en proposant l'automate le plus général et en demandant une phrase impossible si le langage est jugé trop permissif, afin de filtrer progressivement la bonne solution.

L'apprentissage d'un automate d'états fini par fusion.
Source : Auteur (J. Nicolas)
Le treillis des partitions d'états correspondant à l'apprentissage d'un langage régulier à partir des phrases {ac, ag}.
On a effectué un zoom sur certains éléments du treillis. Pour les autres, on a simplement indiqué les états fusionnés. La double ellipse est le plus grand langage compatible avec l'ensemble de phrases impossibles {a, aag, agg, aca, acg}. Il est constitué des mots formant une suite éventuellement vide de ag terminée par ac (par exemple agagac fait partie du langage).



(supposée infinie) où on peut empiler des symboles spéciaux qui sont ensuite utilisés pour définir la fonction de transition. La richesse de description a un coût : reconnaître une phrase d'un langage peut demander un nombre d'opérations de l'ordre de n^3 si n est la longueur de la phrase.

Crispr detail

Show repeat palindromic structure

Select all units | Select all spacer | Clear all | Download selected data | Extract flanking sequences | Extract CRISPR sequence

		BEGIN	END	SEQUENCE	SIZE
	unit 1	1654072	1654104	ATTTCAATCCACGCTCCCCG9AT9999A9C9AC .....((((((....))))))..	33
	spacer 1	1654105	1654140	GACCTCGCG9ACCTCATCCTCGACACCATCAGCAGG	36
	unit 2	1654141	1654173	ATTTCAATCCACGCTCCCCG9AT9999A9C9AC .....((((((....))))))..	33
	spacer 2	1654174	1654206	9GT9GC9999AACATCATC99TC9T99ACAAAT	33
	unit 3	1654207	1654239	ATTTCAATCCACGCTCCCCG9AT9999A9C9AC .....((((((....))))))..	33
	spacer 3	1654240	1654274	T9C9CCATATCCCCCTTTGATCT9CCCGTCAA99C9	35
	unit 4	1654275	1654307	ATTTCAATCCACGCTCCCCG9AT9999A9C9AC .....((((((....))))))..	33
	spacer 4	1654308	1654342	CGTTCACTGCGCGGCGATGTGCGCCGTGTACAAG	35
	unit 5	1654343	1654375	ATTTCAATCCACGCTCCCCG9AT9999A9C9AC .....((((((....))))))..	33

Détail des répétitions (*unit*) d'un CRISPR, sa structure emboîtée, et les segments viraux (*spacer*). La structure en tige-boucle des répétitions du CRISPR précédent. (Source : Auteur (J.Nicolas), base de données http://crispi.genouest.org/)

Notons ici que les langages permettent de structurer les textes de façon logique mais également spatiale dans le cas des macromolécules : à une correspondance entre symboles correspond vraiment une proximité physique et/ou une interaction chimique. Au bout du compte, un langage est porteur de sens et savoir le décrire donne des indications précieuses pour comprendre les phénomènes à l'œuvre.

Les langages de programmation et la description des pages Web reposent sur ce type de grammaire, mais est-ce suffisant pour la biologie ? La réponse est clairement non. Les appariements qu'acceptent les grammaires hors contexte doivent être nécessairement emboîtés (par exemple si j'ouvre une parenthèse [voire ici une deuxième], je suis obligé de mettre le crochet fermant avant la parenthèse fermante). Or si on considère des structures d'ARN comportant des pseudo-nœuds comme dans les télomérases, ou des protéines comportant plusieurs acides aminés appelés cystéines qui ont tendance à s'assembler spontanément pour former des liaisons fortes, les appariements peuvent être dans un ordre quelconque par rapport à l'ordre dans la phrase. Il faut encore augmenter la complexité des langages et des grammaires nécessaires. Les *langages contextuels* sont engendrés par des grammaires étendues où le symbole non terminal de gauche et la partie droite de la règle peut être encadrée d'autant de symboles que nécessaire (il a un contexte d'application) du moment qu'on les retrouve à droite. Les langues naturelles comme les langages moléculaires du vivant se rangent dans cette catégorie, sans exiger cependant toute la puissance offerte par les grammaires contextuelles. Savoir découvrir et modéliser au juste niveau ces langages de façon à analyser automatiquement leurs phrases reste un passionnant défi de recherche.

J.N., C.B. et F.C.

POUR ALLER PLUS LOIN...

- *The language of genes*, David Searls, nov. 2002. Nature 420 211-217 Version accessible sur ftp://ftp.cis.upenn.edu/pub/cse140/public_html/2002/Searls.pdf
- *La grammaire de la vie*. Antoine Danchin, 2009.
- http://www.normalesup.org/~adanchin/causeries/grammaire.html
- *Langages formels, calculabilité et complexité*. Olivier Carton, Paris, Vuibert, coll. « Capes-agrég », oct. 2008. Version accessible sur http://www.normalesup.org/~bisson/tea/lfcc.pdf

par Jill-Jênn Vie

GroLopin et les plans projectifs finis

Un petit problème de combinatoire peut cacher de grandes théories. Arithmétique des congruences, corps finis, espaces projectifs (en passant par une conjecture non résolue), voilà des notions qu'il vaut mieux connaître pour trouver l'algorithme qui construise la solution optimale.

Prologin (voir page 25) est une association d'étudiants de l'EPITA (une école d'ingénieurs en informatique), des Écoles normales supérieures et de l'École polytechnique, qui organise chaque année un concours national d'informatique ouvert à tous les jeunes de 20 ans et moins.
Cette association a organisé durant trois semaines, à la rentrée 2012, un mini-jeu sur le site www.grolopin.com (le pourquoi de ce nom est laissé en exercice au lecteur), dont on peut tirer de nombreux enseignements.
Les premiers à trouver tous les pin's remportaient, entre autres, les éditions spéciales Alan Turing et Doctor Who du jeu Monopoly.

Le jeu des pin's compatibles

Voici pour commencer la règle du jeu :

> Le laboratoire d'Okabé ne cesse de compter de nouveaux membres, et il s'attache à distribuer à chacun d'eux un pin's, en respectant les contraintes suivantes :
> - Les pin's sont tous composés d'un engrenage de M dents dont N percées d'un trou.
> - La superposition de deux pin's quelconques doit toujours laisser apparaître un unique trou.
>
> Les concurrents devaient afficher le plus de pin's possible qui respectent la propriété.
> Et il fallait se dépêcher, car les autres réfléchissaient au même défi !

Les pin's étaient représentés par des chaînes de M caractères contenant N fois le symbole « O » correspondant à un trou. Par exemple, pour trois « O » parmi sept, le jeu suivant de trois pin's est correct (deux pin's quelconques ont toujours exactement un O à la même place) :

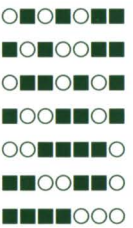
OOO■■■■
O■■OO■■
O■■■■OO

Mais celui-ci ne l'est pas, car les premier et dernier pin's ont deux trous à la même position :

OO■O■■■
O■O■O■■
O■■O■O■

Les couples (N ; M) pour lesquels il fallait résoudre l'énigme étaient (3 ; 7), (4 ; 13), (6 ; 31), (9 ; 73) puis finalement (28 ; 757) (notez qu'on a toujours $M = N^2 - N + 1$).

Pour (N ; M) = (3 ; 7), le nombre maximal de pin's constructibles est 7, dont voici un jeu admissible :



L'avantage de ce problème est que chacun, peu importe son bagage mathématique, pouvait soumettre des pin's et contribuer à dévoiler des pixels de la carte.

M est un majorant du nombre de pin's

Le problème consiste donc à réaliser un tableau de $M = N^2 - N + 1$ colonnes et le plus de lignes possible, tel que chaque ligne comporte N trous et que deux lignes quelconques aient exactement un trou à la même position.

On peut commencer par remarquer qu'il est impossible d'obtenir plus de M pin's. En effet, si on disposait d'un tel jeu, on aurait strictement plus de MN trous à répartir parmi les M colonnes. D'après le « principe des tiroirs », il existerait une colonne comportant au moins N +1 trous. Considérons alors les N + 1 pin's ayant un trou dans cette colonne, il reste pour chacun d'entre eux N – 1 trous à répartir parmi les $M - 1 = N^2 - N$ colonnes restantes, soit en tout $(N + 1)(N - 1) = N^2 - 1$ trous parmi $N^2 - N$ colonnes. Une colonne comportera au moins deux trous : les deux pin's ainsi mis en évidence auront deux trous aux mêmes positions, ce qui constitue une contradiction.

L'intuition qui se cache derrière la solution est de voir les pin's comme des droites du plan et les trous comme les points par lesquels elles passent : deux droites quelconques sont sécantes ou parallèles. Ainsi, ajouter un point « à l'infini » pour chaque classe de droites parallèles permet de garantir un point d'intersection pour chaque paire de droites, donc un unique trou en commun pour chaque paire de pin's. C'est la base de ce qu'on appelle la géométrie projective.

Pour construire un jeu de pin's maximal, on distinguera deux cas :

- $N = p + 1$ où p est un nombre premier,
- $N = p^k + 1$ pour p premier et $k > 1$.

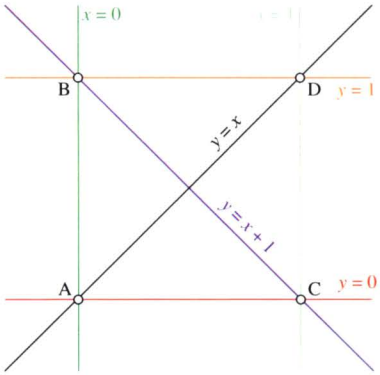
La résolution pour N = 3

Commençons par le cas de l'exemple, soit N = 3.

Considérons toutes les droites à coeffi-

Code Python résolvant le cas N = *p* + 1 où *p* premier

```
N = 3
p = N - 1
Np = range(p)
points = [(x, y) for x in Np for y in Np] \
        + [('infini', k) for k in Np] \
        + [('infini', 'infini')]
for a in Np :
    for b in Np :
        pins = "" # droite y = ax + b
        for x, y in points :
            pins += 'o' if (x == 'infini' and y == a) \
                        or (x != 'infini' and y == (a * x + b) % p) else ' '
        print(pins)
for a in Np :
    pins = "" # verticale x = c
    for x, y in points :
        pins += 'o' if ((x, y) == ('infini', 'infini') or x == a) else ' '
        print(pins)
pins = "" # infini
for x, y in points :
    pins += 'o' if x == 'infini' else ' '
print(pins)
```



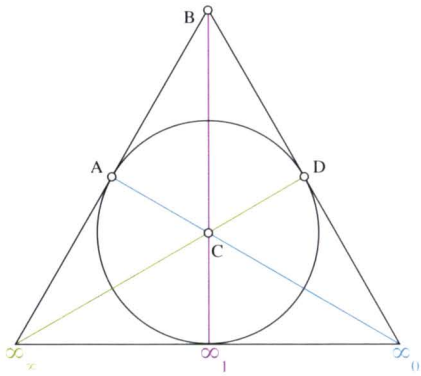


Figure 1. Intersections des droites à coefficients dans {0, 1} modulo 2 à gauche, avec les points à l'infini à droite. À noter que sur le graphique à gauche, les droites en croix sont parallèles : modulo 2, les droites $y = 1 + x$ et $y = 1 - x$ sont identiques ; à droite, la droite $y = x$ est représentée par un cercle. On a ainsi construit une représentation du plus petit plan projectif (intitulé *plan de Fano*).

cients dans {0; 1}.
Elles ont pour équations $y = ax + b$ pour a et b dans {0; 1} et $x = c$ pour c dans {0; 1}.
Considérons le reste de la division euclidienne par 2 (on dit la valeur « modulo 2 ») de chaque opération arithmétique : ainsi, 1 + 1 sera équivalent à 0 et donc la droite $y = x + 1$ passera (*cf.* figure 1) par le point (1 ; 0).
La figure 2 dévoile un jeu de pin's maximal pour N = 3.
On ajoute le point à l'infini ∞_a à toutes les droites parallèles de pente a ; ainsi, toutes les droites de pente 0 ($y = 0$ et $y = 1$) se couperont en un nouveau point nommé ∞_0, toutes les droites de pente 1 ($y = x$ et $y = x + 1$) se couperont en ∞_1 et toutes les droites de pente infinie ($x = 0$ et $x = 1$) se couperont en ∞_∞.

La résolution pour N – 1 premier

Plaçons-nous maintenant où $N - 1 = p$ est un nombre premier.
On appelle $\mathbb{N}_p = \{0 ; 1 ; \ldots ; p - 1\}$ l'ensemble des entiers *modulo p* (tous les calculs y seront effectués en identifiant a et $a + p$).
On considère le cadre général de droites $D_{a;b}$: $y = ax + b$ et de verticales V_c : $x = c$ où a, b et c appartiennent à $\mathbb{N}_p$. Les points d'intersection sont les p^2 points du plan $\{(x ; y), x, y \in \mathbb{N}_p\}$ auxquels il faut ajouter les $p + 1$ points à l'infini $\{\infty_a , a \in \mathbb{N}_p\} \cup \{\infty_\infty\}$.
Dénombrons-les :
- chacune des p^2 droites $D_{a;b}$ passe par les p points $(x ; ax + b)$ pour $x \in \mathbb{N}_p\}$ ainsi que le point à l'infini ∞_a ; soit $p + 1 = N$ points en tout ;
- chacune des p verticales V_c passe par les p points $(c ; y)$ pour $y \in \mathbb{N}_p$ ainsi que le point à l'infini ∞_∞ soit $p + 1 = N$ points en tout ;
- la droite infinie passe par les $p + 1 = N$ points à l'infini.

Analysons les intersections :
- les verticales V_c se coupent en ∞_∞ par définition ;
- pour a fixé $\in \mathbb{N}_p$, les droites $D_{a;b}$ se coupent en ∞_a par définition ;
- chaque droite (sauf la droite infinie) passe par un unique point à l'infini, donc coupe la droite infinie en un unique point ;
- toute droite $D_{a;b}$ coupe toute droite V_c en le point $(c ; ac + b)$
- si $a \neq a'$, $D_{a;b}$ coupe $D_{a';b'}$ en un point $(x ; y)$ vérifiant $y = ax + b = a'x + b'$ soit $(a - a')x = b' - b$ mais comme p est premier, le théorème de Bézout nous affirme l'existence d'entiers relatifs u et v tels que $u(a - a') + vp = 1$ et donc : $u(a - a')x = u(b' - b) = x \pmod p$ est l'unique solution de l'équation. Donc les droites se coupent en un unique point.

On construit ainsi $p^2 + p + 1 = M$ pin's à $p + 1 = N$ trous parmi $p^2 + p + 1 = M$ dents, ce qui est donc un jeu de pin's maximal.

Le cas $N = p^k + 1$ où p premier, $k > 1$

On peut observer à l'analyse qui précède que la seule chose dont on a besoin pour garantir l'intersection entre deux droites quelconques à coefficients dans une certaine structure, c'est de garantir l'existence d'un inverse pour chaque élément, ce qui est caractéristique de la structure de corps fini. Ainsi, à la section précédente, $\mathbb{N}_p$ joue en fait le rôle du corps à p éléments noté $\mathbb{Z}/p\mathbb{Z}$.
Comme il existe un corps à p^k éléments, on va pouvoir construire un tel plan, mais il ne s'agira pas de $\mathbb{Z}/p^k\mathbb{Z}$ puisque par exemple le nombre p n'y a pas d'inverse (c'est un diviseur de zéro : $p \bullet p^{k-1} = 0 \bmod p^k$, $\mathbb{Z}/n\mathbb{Z}$ a une structure de corps si et seulement si n est premier).
Certains candidats ont ainsi pu constater que leur algorithme fonctionnant pour N = 2, 3 et 6 ne fonctionnait pas pour $N = 9 (= 2^3 + 1)$.
Pour éviter d'avoir à construire un corps, on peut recourir à un logiciel de calcul formel tel que SAGE ; le code est presque exactement le même que pour le cas $N = p + 1$ (il suffit de changer la structure et de retirer un modulo p).

Cas général et dénouement

Ce n'est pas parce qu'il n'existe pas de corps de cardinal q non primaire (puissance de nombre premier) qu'il n'existe pas de plan projectif d'ordre q, c'est-à-dire à $q^2 + q + 1$ points et droites et $q + 1$ points par ligne. Mais leur existence est un problème ouvert. Une épopée de calcul par ordinateur menée à Concordia University par Clement Lam de 1980 à 1989 a ainsi conduit à la conclusion qu'il n'en existe pas pour $q = 10$.
Seules six personnes ont trouvé tous les pin's possibles, parmi lesquels Ryan Lahfa, qui avait 14 ans lors du déroulement du concours.

Le classement est disponible sur www.grolopin.com.

J.-J. V.

Code SAGE résolvant le cas $N = p^k + 1$ où p premier, $k > 1$

```
N = 9
q = N - 1
Fq = GF(q, 'x')
points = [(x, y) for x in Fq for y in Fq] \
        + [('infini', k) for k in Fq] \
        + [('infini', 'infini')]
for a in Fq :
    for b in Fq :
    pins = "" # droite y = ax + b
    for x, y in points :
        pins += 'o' if (x == 'infini' and y == a) \
                    or (x != 'infini' and y == a * x + b) else ' '
        print(pins)
for a in Fq :
    pins = "" # verticale x = c
    for x, y in points :
        pins += 'o' if ((x, y) == ('infini', 'infini') or x == a) else ' '
    print(pins)
pins = "" # infini
for x, y in points :
    pins += 'o' if x == 'infini' else ' '
print(pins)
```

	(0 ; 0)	(0 ; 1)	(1 ; 0)	(1 ; 1)	∞_0	∞_1	∞_∞
$y = 0$	○	■	○	■	○	■	■
$y = 1$	■	○	■	○	○	■	■
$y = x$	○	■	■	○	■	○	■
$y = x + 1$	■	○	○	■	■	○	■
$x = 0$	○	○	■	■	■	■	○
$x = 1$	■	■	○	○	■	■	○
∞	■	■	■	■	○	○	○

Figure 2.
Grille du plan projectif d'ordre 2. Pour chaque droite, on met un ○ en dessous des points par lesquels elle passe. On voit ainsi apparaître un jeu de pin's maximal pour N = 3 (*cf.* le jeu admissible plus haut).

par François Lavallou

Le traitement du signal

Aux frontières des mathématiques et de l'informatique, le traitement du signal, à l'intitulé médicalisé, est injustement méconnu. Avec l'arrivée des ordinateurs, il a basculé dans l'informatique moderne et envahi l'intégralité de nos outils de communication.

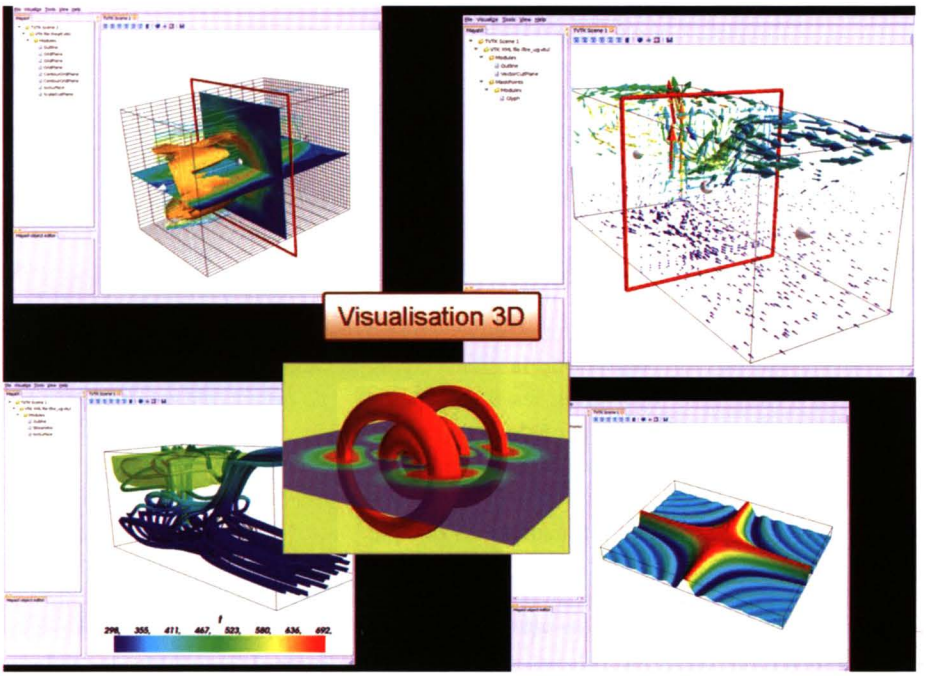
Nous sommes tous des Monsieur (ou Madame) Jourdain du traitement de signal. En lisant ces lignes, vous effectuez ce que vous allez découvrir… en lisant ces lignes ! Nos sens sont en veille perpétuelle et captent en continu les informations de notre environnement par leurs interactions avec notre enveloppe corporelle, qu'elles soient physiques, chimiques, électromagnétiques, phoniques ou thermiques. À partir de ces perceptions, notre cerveau nous offre la possibilité de nous faire une représentation de notre milieu environnant. Notre corps est un système complexe de détection dont l'unité centrale de traitement est notre cerveau. Si notre ordinateur neuronal vient à se dérégler et perd une partie de ses logiciels de traitement, alors, malgré un bon fonctionnement de nos organes sensoriels, peuvent apparaître des cas d'anosmie, d'agueusie, de surdité ou de cécité.
Nous allons nous intéresser aux signaux obtenus par des détecteurs non biologiques. La théorie de la communication ou les outils techniques du traitement du signal (comme l'analyse numérique ou des approches probabilistes) ne sera, par contre, pas abordée ici.

« Donnez-moi cent paramètres et je vous fais un éléphant. Donnez-moi un cent-unième et je lui ferai remuer la queue. »

Jacques Hadamard

Les problèmes inverses

Qui dit traitement, dit informatisation. On quitte le monde analogique réel pour entrer dans son simulacre numérique. Chaque future victime d'un calcul est alors représentée par un ensemble fini de valeurs, mis habituellement sous forme de vecteur ou matrice. Prenons



l'exemple de domaines physiques tels que l'électromagnétisme (optique, radar, radioastronomie, laser), les contrôles non destructifs, la spectroscopie, et l'imagerie en général pour lesquels se pose le problème de déterminer la distribution spatiale d'un objet à partir d'un ensemble de mesures, que nous appellerons génériquement *images*. Notons x l'objet, y l'image obtenue par la mesure et H la transformation représentative du système de détection. H est souvent appelée un *opérateur* (voir en encadré). Ces trois grandeurs sont liées par la relation $y = H(x)$, pour une expérimentation non bruitée, donc utopique. On ne considèrera que le cas très courant d'une transformation H linéaire : le niveau de l'image est proportionnel à celui de l'objet et l'image d'une somme d'objet est la sommes des images de chaque objet. On note alors Y = H.X, où X et Y sont les matrices colonnes de l'objet et de l'image avec une dimension égale à l'échantillonnage (le nombre de points de mesures). On n'enlève rien à la généralité du problème en supposant objet et image échantillonnés de la même façon, de sorte que les matrices X et Y contiennent le même nombre de valeurs et donc que la matrice H associée à l'opérateur H soit carrée.

Les opérateurs usuels

Les plus simples et usuels des opérateurs utilisés en traitement du signal sont les suivants :

- **translation :** $\Phi_t[f](x) = f(x+t)$,
- **taux d'accroissement :** $\Delta_{x_0}[f](x) = \dfrac{f(x) - f(x_0)}{x - x_0}$,
- **différentiel (ou dérivée) :** $D[f](x) = f'(x)$.

L'intégration, opérateur inverse de D, permet de construire de nombreux autres opérateurs, et en particulier l'importante famille des *opérateurs à noyaux*, de la forme $H[f](x) = \int K(x,y)f(y)dy$.

Deux exemples courants sont :

- **la convolution :** $H_g[f](x) = \int f(u)g(x-u)du$.
- **l'opérateur tomographique :** $A[f](x) = 2\int_x^{+\infty} \dfrac{uf(u)du}{\sqrt{u^2 - x^2}}$.

Le problème se trouve ainsi mathématisé, et son étude est ramenée à celle des matrices d'opérateur, dont le cadre mathématique est celui des espaces de Hilbert. Ces espaces de fonctions, généralisations des espaces euclidiens, sont munis d'un produit scalaire. On peut alors parler d'orthogonalité de fonctions et définir des projections. Ceci explique que l'intuition géométrique est un moteur puissant de la créativité en traitement du signal, les théorèmes y étant analogues à ceux de la géométrie classique. Un élément d'un espace de Hilbert, une fonction (1D) ou une image (2D), est défini de manière unique par ses coordonnées dans une base, que nous supposerons orthogonale. Un espace de Hilbert peut alors être vu comme un ensemble de suites infinies, mais de carrés sommables, puisque les fonctions qu'elles représentent sont d'énergie finie (voir encadré). Ainsi, l'approximation d'une courbe par un polynôme de degré n est un problème d'optimisation qui consiste à rechercher la projection d'un objet de dimension infinie dans un espace de dimension $n + 1$.

Dans la plupart des cas concrets, les opérateurs linéaires d'un espace de Hilbert sont des transformations qui peuvent être décomposées en somme d'homothéties de rapports (valeur propres) et de directions (vecteurs propres) différents. L'ensemble des valeurs et vecteurs propres résume toutes les propriétés de l'opérateur et est appelé son *spectre*. La théorie spectrale est au cœur du traitement du signal.

Un changement de base dans un espace vectoriel peut simplifier les coordonnées d'un vecteur. Il en est de même pour la structure d'une matrice ! Par un changement de base de vecteurs propres orthogonaux, la plupart des matrices peuvent s'écrire sous la forme diagonale $H_{n,n} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \lambda_n \end{bmatrix}$, où les $(\lambda_i)_{i=1,2\ldots n}$ sont les valeurs propres. Toute la problématique du traitement du signal est résumée dans cette représentation.

On distingue les *problèmes directs*, lorsque l'objet x et l'opérateur H sont connus, et les *problèmes inverses*, pour lesquels seul l'objet x est inconnu.

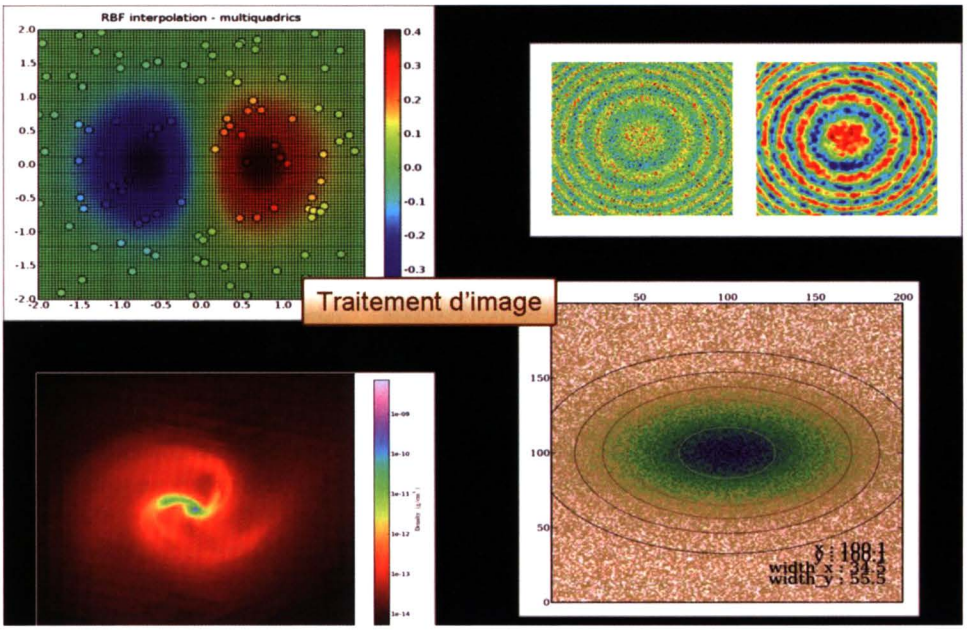
Les mesures de température, pression, hygrométrie et pluviométrie recueillies par une station météorologique sont porteuses d'informations qui vont permettre,

D'Euclide à Hilbert

La notion d'*espace de Hilbert*, ou *hilbertien*, généralise celle d'espace euclidien, lente mathématisation de notre environnement. David Hilbert a prolongé les méthodes mathématiques des espaces euclidiens de dimensions inférieures ou égales à trois à des espaces de dimension quelconque, voire infinie (mais dénombrable).

En 1926, John von Neumann développe cette idée en considérant qu'un système quantique constitué de N particules, chacune déterminée par six paramètres (trois de position et trois de vitesse), est un point d'un espace de Hilbert de dimension 6N. Il axiomatise la mécanique quantique, associe aux grandeurs physiques traditionnelles des opérateurs linéaires et leurs applique les techniques de l'analyse mathématique.

Les espaces hilbertiens apparaissent fréquemment en physique, et sont incontournables en traitement du signal. Puisque les phénomènes physiques étudiés sont d'énergie finie, et que l'énergie est proportionnelle au carré d'une grandeur (la vitesse pour un mouvement, $E_c = mv^2/2$; la position pour l'énergie potentielle d'un ressort, $E_p = kx^2/2$; le champ électrique E ou magnétique B pour les phénomènes électromagnétiques, $E_{em} = \varepsilon_0 E^2/2 + B^2/2\mu_0$; l'intensité I ou la tension U en électricité, $E = RI^2 = U^2/R$, *etc.*), la plupart des grandeurs physiques sont de carrés intégrables. Ceci explique que les exemples les plus courants d'espace de Hilbert en physique sont les espaces de fonctions de carré intégrable !



en utilisant les résultats d'expériences passées, de prédire l'évolution future de phénomènes atmosphériques. C'est un exemple de problème direct qui prédit le futur à partir du présent, sous réserve d'avoir établi un bon modèle.

Le problème inverse impose de pouvoir inverser l'opérateur H pour obtenir la solution $X = H^{-1}Y$,

où $H^{-1} = \begin{bmatrix} \lambda_1^{-1} & 0 & \cdots & 0 \\ 0 & \lambda_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \lambda_n^{-1} \end{bmatrix}$.

L'existence de cet opérateur inverse nécessite donc qu'aucune valeur propre ne soit nulle, c'est-à-dire que $HX = 0$ implique que $X = 0$ soit la seule solution. L'opérateur est alors *injectif*, ce qui garantit l'unicité de la solution, si elle existe. C'est la plus incontournable condition imposé par Hadamard pour qu'un problème inverse soit *bien posé* (voir en encadré). Si un problème est bien posé, il existe certainement un algorithme stable permettant de déterminer une solution « proche » de la solution exacte.

Une mesure du degré avec lequel un problème est bien posé est le *critère de conditionnement* $\sqrt{\frac{\lambda_{max}}{\lambda_{min}}} \geq 1$,

où $c = \sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$ représente la racine carrée du rapport des valeurs propres extrêmes. Même si un problème est bien posé, il peut être *mal conditionné* lorsque ce critère est « nettement supérieur » à l'unité. La solution sera alors peu robuste, et variera beaucoup même pour une très faible erreur de mesure. Une caractéristique commune des problèmes d'imagerie est qu'ils sont souvent mal posés ou mal conditionnés. Il convient alors de modifier le problème originel avant d'envisager un traitement numérique. Des hypothèses supplémentaires concernant les caractéristiques de la solution peuvent ainsi s'avérer nécessaires.

Le rôle de la régularisation

Les outils mathématiques utilisés en traitement du signal proviennent de plusieurs domaines des mathématiques :

les espaces vectoriels, l'algèbre linéaire, les distributions, l'analyse fonctionnelle, les probabilités, les statistiques, l'optimisation, l'analyse numérique... D'un point de vue méthodologique, deux directions de recherche sont utilisées pour l'étude des problèmes inverses. Une voie, très en vogue, utilise les méthodes probabilistes qui incorporent des informations *a priori* sur la solution (c'est toujours plus facile quand on connaît la solution !).

L'autre voie date des années 1960 avec les travaux de Tikhonov (1906–1993) en physique mathématique, qui traite, en dimension infinie, des problèmes d'existence, d'unicité et de stabilité. La résolution numérique s'effectue alors par projection sur des espaces de dimension finie.

Andreï Nikolaïevitch Tikhonov.

Résoudre un problème inverse revient à déterminer un objet x dont l'image $H(x)$ par le système de détection est « la plus proche possible » de la mesure y. Cette notion de distance entre éléments de Hilbert a un sens grâce à la norme associée au produit scalaire, analogue de la distance euclidienne en géométrie classique. D'où la célèbre *méthode des moindres carrés*, qui consiste à chercher une solution x, possédant les caractéristiques attendues de l'objet, qui minimise la quantité $Q(x) = \|y - H(x)\|^2$. La solution, identique à l'inverse généralisée, est $x = \left({}^tHH\right)^{-1} {}^tHy$, où ${}^tH$ est la matrice transposée de H. Bien sûr, $x = H^{-1}y$ si la matrice H est carrée et inversible.

Dans le cas d'un système linéaire de matrice mal conditionnée, la régularisation de Tikhonov incorpore de plus dans le critère de minimisation des conditions sur la solution. Son principe est de minimiser une quantité qui dépend de deux termes et d'un paramètre :
$Q(x) = D(y - H(x)) + \alpha S(x)$.
Comme pour la méthode des moindres carrés, le premier terme est fidèle aux données et s'annule si $H(x) = y$. Le second terme tient compte de conditions *a priori* sur la solution et se doit d'être « petit » quand elles sont remplies.

Le poids des données ($\alpha = 0$) ou de l'*a priori* ($\alpha = +\infty$) sur la solution dépend de la valeur du paramètre α. Pour une valeur du paramètre fixée, on cherche la solution x qui minimise $Q(x)$. Classiquement, $Q(x) = \|Y - HX\|^2 + \alpha\|CX\|^2$ et la solution généralisée est $x = \left({}^tHH + \alpha {}^tCC\right)^{-1} {}^tHy$. On retrouve bien sûr la solution des moindres carrés pour $\alpha = 0$. Tout dépend donc du choix de l'opérateur de régularisation. Pour l'identité C = I, les solutions de normes faibles sont favorisées, et on évitera les grandes fluctuations dues à des valeurs propres faibles. De fait, les valeurs propres λ ont été translatées et changées en $\lambda + \alpha$, ce qui régularise la solution en atténuant l'amplification λ^{-1} due aux faibles valeurs propres. Le conditionnement est amélioré car

$$\frac{\lambda_{max} + \alpha}{\lambda_{min} + \alpha} < \frac{\lambda_{max}}{\lambda_{min}} \text{ pour } \alpha > 0.$$

On a effectué un *filtrage*, c'est-à-dire une légère manipulation des valeurs propres.

En prenant pour C l'opérateur de dérivation, on favorise les solutions qui n'ont pas de variations brusques, quand on sait par exemple que la solution doit être régulière.

Ce principe de régularisation (on en invente chaque jour, pour chaque problème traité) illustre la créativité dont on doit faire preuve en traitement du signal pour incorporer une idée de la solution, plus ou moins contraignante, selon les conditions expérimentales, dans le processus de calcul. Ce côté expérimental de la recherche d'une solution optimale, jamais exacte, ne pourrait, bien sûr, jamais se faire sans l'apport de la puissance informatique. La conception et l'étude de nouveaux moyens de diagnostic fait appel aux mathématiques, mais la mise en œuvre opérationnelle nécessite l'informatique.

Le traitement du signal étant une discipline extrêmement vaste, on ne peut qu'effleurer ici quelques principes généraux, sans entrer dans le vif des méthodes de traitement. Ce monde, à la routine absente, fait le bonheur des ingénieurs-chercheurs et est une belle école d'humilité. Quel travail il reste à accomplir quand on voit, dans les séries télévisées, qu'en moins de temps qu'il n'en faut pour appuyer sur une touche, on extrait d'un unique pixel un visage !

F. L.

Les illustrations ont été réalisées par Pierre Raybaut.

Références

- *Les matrices*. Bibliothèque Tangente 44, 2012.
- *Le calcul intégral*. Bibliothèque Tangente 50, 2014.
- *Les angles*. Hors-série 53 de *Tangente*, 2014.

Le Bien selon Hadamard

Soit y l'image d'un objet inconnu x par une mesure dont l'action est représentée par l'opérateur linéaire H. Le problème inverse est donc de résoudre l'équation matricielle $Y = HX$. Deux ensembles sont caractéristiques d'un opérateur H : son *noyau* Ker(H), ensemble des vecteurs d'image nulle, et son *image* Im(H), ensemble des images pour l'ensemble des objets possibles.

La première chose à faire est de s'assurer qu'il n'y a pas d'ambiguïté pour l'image réciproque, c'est-à-dire qu'une image a un seul antécédent. Si deux objets X_1 et X_2 donnent la même image ($Y = HX_1 = HX_2$), alors ils doivent être égaux ($X_1 = X_2$). Puisque l'opérateur est linéaire, on doit avoir $H(X_1 - X_2) = 0 \Rightarrow (X_1 - X_2) = 0$.

Le noyau de l'opérateur H se trouve réduit au strict minimum, le vecteur nul : l'application H est injective. La solution est unique, si elle existe !

Le deuxième critère est d'assurer l'existence. L'image doit pouvoir être celle d'un objet possible : $y \in \text{Im}(H)$.

À ces deux conditions d'unicité et d'existence, qui garantisse une bijection entre l'ensemble des objets et celui des images, Hadamard ajoute, comme critère suffisant pour l'inversion d'un opérateur, une condition de continuité nécessaire pour les problèmes bruités de la physique.

Quand ces trois conditions sont remplies, on parle de *problème bien posé au sens de Hadamard*. Dans ce cas, il existe en général un algorithme stable permettant de construire une solution approchée de la solution exacte. Très souvent, un problème direct est bien posé et un problème inverse est mal posé (c'est-à-dire « non bien posé »).

par Hervé Lehning

Protégez-vous des hackers !

Les menaces sur Internet portent des noms poétiques comme *spams*, *phishing*, chevaux de Troie, zombies, *botnets*, dénis de service et bien d'autres encore tant l'imagination des criminels est fertile. Heureusement, une bonne hygiène suffit pour éviter l'essentiel car l'élément faible d'un système informatique reste l'être humain.

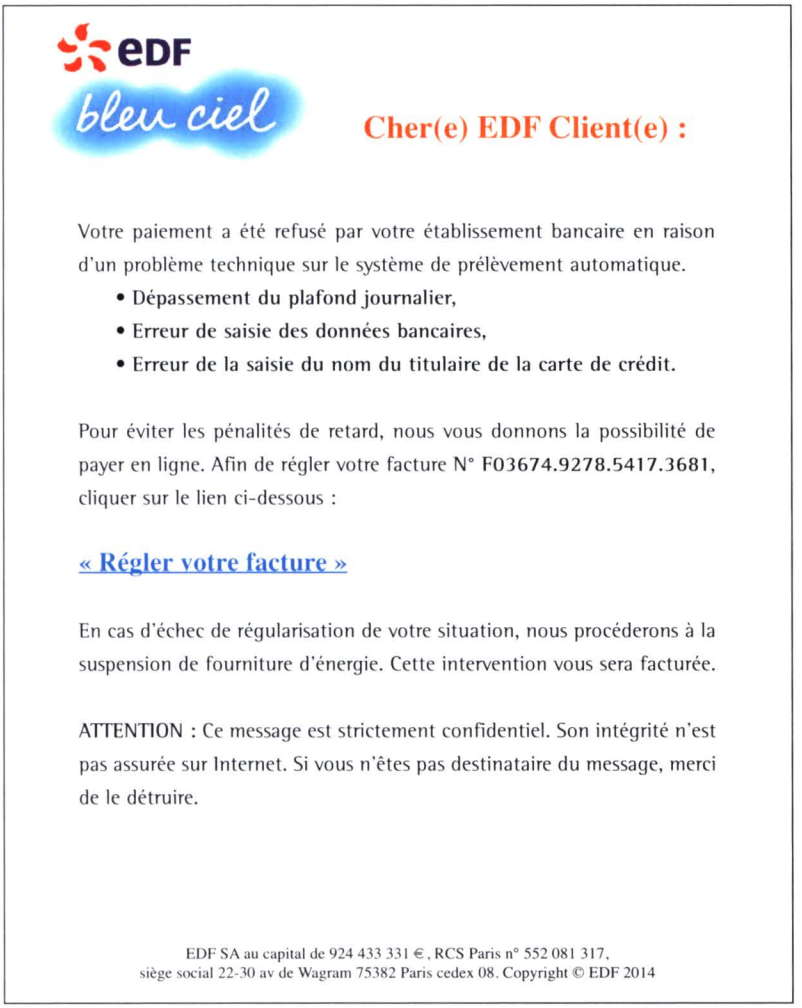
Comment prendre le contrôle d'un ordinateur ? Pas du vôtre mais de celui d'autrui. Bien entendu, il ne s'agit pas de vous l'apprendre ici, ce serait illégal, mais on peut vous expliquer comment les *hackers* s'y prennent. Leur idée est de vous inciter à lancer vous-même un programme malveillant qui le fera pour eux. Comment ? Soit en vous amenant à ouvrir une pièce jointe à un courrier électronique, soit en vous faisant cliquer sur un lien vers un site piraté au préalable. Cela peut se produire sur des sites de téléchargement. À ce niveau, il est conseillé de toujours vérifier si le nom du domaine sur lequel vous cliquez est bien le nom officiel. Par exemple, si vous voulez télécharger un logiciel de Microsoft, méfiez-vous si le site proposé porte un autre nom. Pensez aussi à décocher les cases prévoyant des téléchargements non désirés.

Internet : un monde impitoyable ?

En ce qui concerne les pièces jointes, le mieux est de ne pas ouvrir les documents non sollicités. Pensez que la boîte aux lettres électronique de celui que vous croyez être votre correspondant peut avoir été piratée et que derrière elle se trouve en fait un escroc.

Un exemple de *phishing*

L'idée du *phishing*, ou « hameçonnage », est de se faire passer pour un site de confiance pour utiliser la naïveté de la victime et l'amener à donner des renseignements importants ou payer des factures imaginaires. On peut souvent s'apercevoir de la contrefaçon par la présence de fautes d'orthographe grossières. Dans l'exemple qui suit, elles ont cependant été éliminées pour rendre le courrier plus vraisemblable. La facture utilisée est normalement véritable et montre que le service en question a été piraté.



EDF bleu ciel

Cher(e) EDF Client(e) :

Votre paiement a été refusé par votre établissement bancaire en raison d'un problème technique sur le système de prélèvement automatique.

- Dépassement du plafond journalier,
- Erreur de saisie des données bancaires,
- Erreur de la saisie du nom du titulaire de la carte de crédit.

Pour éviter les pénalités de retard, nous vous donnons la possibilité de payer en ligne. Afin de régler votre facture N° F03674.9278.5417.3681, cliquer sur le lien ci-dessous :

« Régler votre facture »

En cas d'échec de régularisation de votre situation, nous procéderons à la suspension de fourniture d'énergie. Cette intervention vous sera facturée.

ATTENTION : Ce message est strictement confidentiel. Son intégrité n'est pas assurée sur Internet. Si vous n'êtes pas destinataire du message, merci de le détruire.

EDF SA au capital de 924 433 331 €, RCS Paris n° 552 081 317,
siège social 22-30 av de Wagram 75382 Paris cedex 08. Copyright © EDF 2014

Un exemple de *phishing*. Si vous cliquez sur le lien en bleu, vous vous retrouvez sur un site où on vous fait payer au moyen de services du type Paypal, ce qu'aucune administration ne fait.

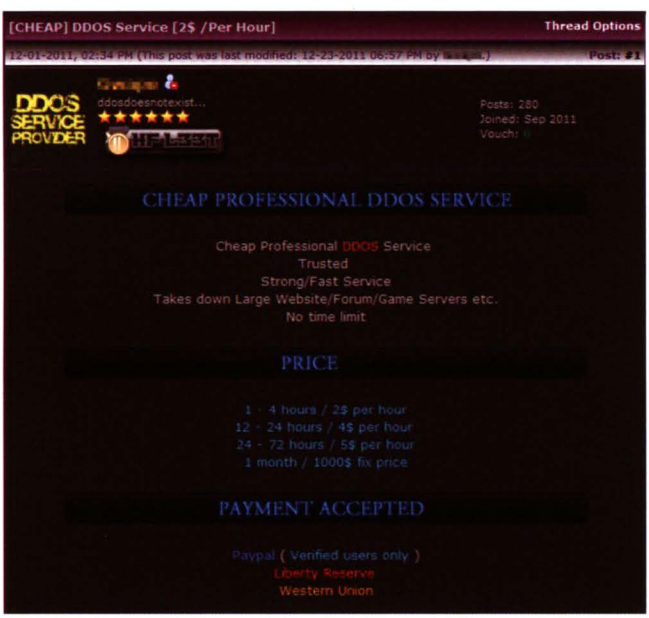
Le piratage d'un service officiel peut se faire en obtenant le mot de passe d'un administrateur du système, ce qui permet d'en prendre la maîtrise. Une fois le système sous contrôle, il est possible de récupérer les données contenues dans ses mémoires et donc d'envoyer des courriers électroniques aux clients en se faisant passer pour le système lui-même.

Une attaque bien plus sérieuse consiste à prendre les données contenues sur votre ordinateur en otage et vous demander une rançon pour vous les rendre. Ne payez surtout pas, cela ne servirait à rien. En général, les *hackers* ne rendent rien. L'ironie de l'histoire est qu'ils utilisent un dispositif destiné à sécuriser Internet pour vous rançonner. En effet, le logiciel malveillant qu'ils vous amènent à exécuter sur votre ordinateur chiffre vos fichiers avec un système de chiffrement asymétrique. Une clef tenue secrète est nécessaire pour les déchiffrer. Bien entendu, les *ransomwares* (ou « rançongiciels ») peuvent s'attaquer aux entreprises comme aux particuliers. Les plus vulnérables sont les PME car elles sont des objectifs plus tentants que les particuliers mais souvent presque aussi inconscients qu'eux des dangers.

Les grandes entreprises se protègent généralement mieux.

Zombies et *botnets*

Pour des attaques de grande envergure, les *hackers* se constituent des réseaux d'esclaves : vos ordinateurs transformés en zombies. La prise de contrôle de votre ordinateur n'est alors que partielle. Ces réseaux de zombies ou *botnets* sont en vente sur Internet pour monter des attaques. La cybercriminalité a ses entreprises ! À quoi peut servir un *botnet* ? Tout d'abord à envoyer des *spams*, ou courriers non sollicités (« pourriels »). Ils contiennent en général de la publicité pour des « médicaments » (même si les criminels n'ont en l'occurrence aucun intérêt à tuer leurs clients, il convient de se méfier des substances vendues par ce genre d'individus), des sites pornographiques ou autres.



Annonce de services pour *hackers* sur Internet. Il semblerait que ce site vende des dénis de service clefs en main, à moins que cela soit un piège… Mais il est avéré que ce genre de service existe sur Internet.

Certains peuvent se demander comment prospère l'industrie du *spam*. L'arnaque semble tellement évidente ! La réponse tient dans le calcul des probabilités. Statistiquement, les *spams* obtiennent une réponse positive dans une dizaine de cas sur un million. La location des services d'un réseau de mille zombies pour une heure coûte environ 10 €. Cela suffit pour atteindre le million de *spams*. Avec 10 €, vous pouvez donc atteindre dix clients. La démarche ne coûte qu'environ 1 € par client, auquel il faut ajouter l'achat du fichier d'adresses, en vente également sur Internet pour des prix similaires. Si vous vendez à chacun pour 100 € de produits avec une marge de 50 %, l'opération procure un bénéfice de 490 € par million de *spams* envoyés, hors taxes, bien entendu… sauf celles éventuelles d'autres *hackers*. Comme le disait Georges Brassens, il y a quelque chose de pourri au royaume de Truanderie…

Malgré cela, les *spams* restent une forme de menace très anodine, et ne constituent qu'une taxe sur l'excessive crédulité. Les *botnets* sont également utilisés pour saturer des services en ligne et les empêcher de fonctionner. On parle alors d'« attaques en déni de services », ou DoS (pour *denial of service*), ou encore DDoS (pour *distributed denial of service*). Le motif de ces attaques peut être l'activisme politique, comme ceux qui peuvent cibler des organisations gouvernementales de type Pentagone, Élysée ou Maison Blanche, mais aussi la mise en danger d'entreprises commerciales, éventuellement concurrentes. Ainsi, en 2000, Amazon a été victime d'une attaque, qui a immobilisé ses services commerciaux pendant dix heures, ce qui a coûté 600 000 $. Le motif peut aussi être financier, pour rançonner une entreprise, affaiblir un concurrent, *etc.*

Les failles de sécurité

Il est possible d'augmenter ainsi les divers types d'attaque, et de parler alors d'espionnage économique. Le renseignement cherché peut parfois sembler anodin mais, par exemple, connaître l'offre d'un concurrent peut aider une entreprise à remporter un marché. Inutile d'imaginer le vol de plans de matériels sophistiqués pour cela.
Mais voyons plutôt maintenant les vulnérabilités des systèmes d'information que les *hackers* exploitent. Chaque pays possède un organisme les collectant, les CERT (pour *computer emergency response team*). Par exemple, le CERT-FR publié par l'Agence nationale de la sécurité des systèmes d'information (Anssi) comporte plus de cinquante alertes nouvelles par mois ! (*cf.* le tableau ci-contre).

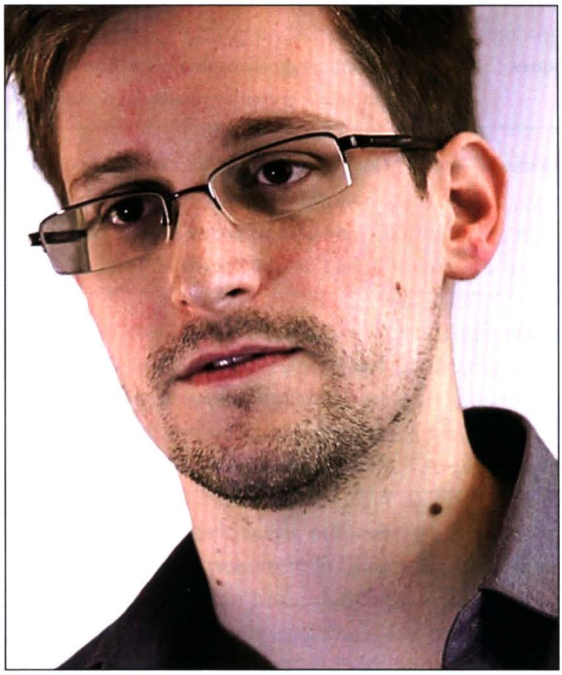
Risques	Exécution de code arbitraire à distance Atteinte à la confidentialité des données
Systèmes affectés	Google Chrome versions antérieures à 33.0.1750.154 pour Windows Google Chrome versions antérieures à 33.0.1750.152 pour Mac et Linux
Résumés	De multiples vulnérabilités ont été corrigées dans Google Chrome. Elles permettent à un attaquant de provoquer une exécution de code arbitraire à distance et une atteinte à la confidentialité des données.
Solutions	Se référer au bulletin de sécurité de l'éditeur pour l'obtention des correctifs

Le bulletin du CERT-FR du 14 mars 2014.

Bien entendu, il s'agit toujours de failles découvertes par les services de l'éditeur ou d'autres et déjà corrigées. Pour cela, il suffit de mettre ses logiciels à jour… en prenant garde de les solliciter soi-même car certains *hackers* créent de faux sites de mise à jour ! Certains se spécialisent dans la découverte de failles de sécurité et les vendent sur Internet. Ces failles non encore découvertes par les services de sécurité sont appelés des *zero day* (ou « jour zéro »). Elles permettent des attaques difficiles à parer, puisque encore inconnues.
L'origine de ces failles tient à une mauvaise conception des logiciels, à des erreurs de programmation. En particulier, les entrées de données sont rarement sécurisées correctement et certains langages de programmation ne s'y prêtent pas. Par exemple, un logiciel peut accepter une entrée dépassant l'espace qui devrait normalement lui être alloué, ce qui permet d'écraser des instructions nécessaires au logiciel en cours d'exécution et les remplacer par d'autres… un code malveillant, bien sûr ! Pour éviter cela, on peut utiliser des langages de programmation adaptés, comme Caml (dans sa version OCaml) ou des bibliothèques de composants logiciels sécurisés. Dans tous les cas, la sécurité des systèmes d'information mérite d'être effectuée *a priori* et non pas seulement *a posteriori* comme c'est le cas le plus courant actuellement.

Les portes dérobées

Un autre type de faille concerne également la programmation initiale mais n'est pas une erreur. Il s'agit de la présence de « portes dérobées », ou *back doors*. La métaphore envoie à l'image d'une citadelle imprenable… sauf qu'une petite porte dissimulée et non protégée permet d'y pénétrer. Les programmeurs créent souvent de telles portes pour tester leurs programmes. Normalement, ils les détruisent ensuite mais peuvent oublier de le faire, intentionnellement ou non.



Edward Snowden (né en 1983), traître, héros ou naïf ?

Les antivirus

Les antivirus sont des logiciels destinés à détecter et détruire les virus. Pour cela, ils se servent de plusieurs techniques. La plus utilisée est la recherche d'un morceau de code du virus le caractérisant et que l'on appelle, pour cette raison, sa « signature ». Pour ce faire, le virus doit se trouver dans la base de données de l'antivirus. Une deuxième méthode analyse les codes des programmes inconnus en simulant son fonctionnement. Cette méthode, dite « heuristique », peut provoquer de fausses alertes. Enfin, une dernière méthode consiste à surveiller le comportement des logiciels actifs. S'il détecte une anomalie, il avertit l'utilisateur.

En cas de détection, les antivirus offrent trois possibilités : réparer le fichier, le supprimer ou le mettre en quarantaine. La meilleure option est la première mais elle n'est pas toujours possible.

Pour les particuliers et les très petites entreprises, il existe des antivirus gratuits tout à fait suffisants. D'autre part, on peut également analyser son ordinateur en ligne, pour disposer des dernières mises à jour. C'est ce qu'il convient de faire en cas de doute (sur un site dûment répertorié).

Un grand nombre de personnes ont été scandalisés par les révélations d'Edward Snowden. Ce consultant de la NSA (National Security Agency) a révélé que les services d'espionnage des États-Unis… espionnaient le monde entier, ce qui, normalement, ne devrait pas passer pour un scoop. Tous les pays font de même, sinon pourquoi payer des espions ?

Peu de gens ont vu que l'affaire était potentiellement bien plus grave. La suspicion réelle serait que, pour espionner plus facilement, la NSA aurait fait placer des portes dérobées dans les logiciels de sécurité fabriqués aux États-Unis. Les services américains ont répondu qu'ils n'étaient pas assez fous pour cela. En effet, s'il existe des portes dérobées, les services des grands pays sont tous capables de les trouver. Espérons que la NSA ait effectivement eu cette sagesse. Il n'en reste pas moins que cela devrait éveiller une méfiance envers les logiciels de sécurité provenant des États-Unis, ou des circuits intégrés venant d'ailleurs puisqu'il est également possible d'y disposer des portes dérobées, ou d'y affaiblir les circuits créant les suites de nombres pseudo-aléatoires, au cœur des systèmes cryptographiques.

H. L.

Les problèmes de l'informatique dans les nuages

Le *cloud computing* consiste à externaliser certaines taches informatiques.

L'avantage essentiel des nuages (informatiques) est la mise en commun de ressources. Plus besoin de disposer de tous les logiciels sur son ordinateur, il suffit d'utiliser ceux se trouvant dans le nuage qui, de plus, sont toujours à jour. De même, vous disposez de toute la mémoire nécessaire. Bien des gens le font en déposant des vidéos sur des sites comme YouTube, par exemple. Il en est de même de la puissance de calcul. Cela peut sembler sans intérêt pour le commun des mortels, qui cherche rarement à calculer des milliards de décimales de nombres tels que π. Ce raisonnement oublie cependant que les logiciels pour fabriquer des vidéos sont, avec les jeux, parmi les plus gourmands en puissance de calcul !

Malgré ces avantages, le *cloud computing* a deux gros inconvénients. Le premier est de dépendre d'Internet. En cas de coupure de réseau, vous n'avez plus accès à ce que vous avez externalisé. Le second est plus grave, il s'agit du manque de confidentialité. Vous ne savez plus où transitent vos données. Pourquoi pas chez vos pires ennemis ? Difficile de répondre totalement au premier inconvénient. Certaines fonctions vitales ne devraient jamais passer par les nuages de l'informatique. En revanche, on peut tenter de résoudre le second problème. Une solution passe par des méthodes de chiffrements possédant la propriété d'homomorphie.

Les chiffrements homomorphes

Un *chiffrement* correspond à une fonction f transformant un nombre en un autre, soit $x \mapsto f(x)$. Il est dit *homomorphe pour l'addition* s'il vérifie $f(x)+f(y)=f(x+y)$ pour tous x et y. Dans ce cas, la somme des chiffrés est le chiffré de la somme. Autrement dit, on peut faire les calculs impliquant l'addition sur les chiffrés puis déchiffrer le résultat. Le déchiffrement sera encore possible si $f(x)+f(y)=f(x * y)$ où $*$ est une autre loi, par exemple une multiplication. On parlera encore d'homomorphisme additif dans ce cas.

Il existe des chiffrements homomorphes additifs. En s'inspirant de l'idée de base de la méthode RSA, Pascal Paillier a inventé un chiffrement homomorphe pour l'addition en 1999. Il est opérationnel et peut être utilisé pour fabriquer des systèmes de vote électronique, comme Helios, qui est utilisé pour voter sur Internet. De manière plus subtile, on peut également imaginer consulter une base de données dans le nuage (c'est-à-dire à distance, *via* Internet) sans révéler la requête faite à l'aide d'un chiffrement homomorphe additif (voir *Tangente Sup* 70–71).

De plus, il existe des chiffrements homomorphes pour les deux opérations, comme celui inventé par Craig Gentry en 2009, mais ils ne sont pas encore opérationnels car trop lourds. Quand ils le seront, ils permettront d'effectuer dans les nuages tous les calculs impliquant les deux opérations.

par Hervé Lehning

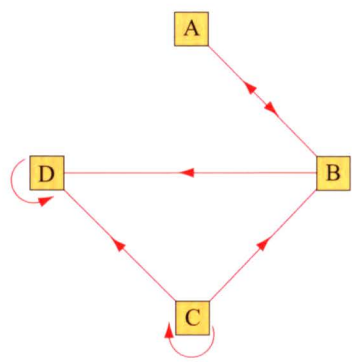
Le classement des pages
par les moteurs de recherche

Les pages du Web sont innombrables. Pourtant, les moteurs de recherche comme Google les classent en une fraction de seconde. Ils sont capables de repérer les pages les plus populaires, aussi bien que celles qui se cachent...

Les moteurs de recherche comme Google utilisent le graphe du Web (voir l'encadré) pour proposer une liste de pages traitant d'un sujet donné, en les classant selon leurs degrés de pertinence. Par exemple, si l'on effectue une recherche du mot « tangente » (le 4 juin 2014), on obtient 1,73 million de résultats en un quart de seconde ! Le premier est l'article de Wikipédia, qui concerne la géométrie, le deuxième concerne un magazine de culture mathématique. D'où vient ce classement ? La réponse est qu'il est dû aux internautes eux-mêmes, qui font pointer leurs pages ou non vers telle ou telle autre. Chaque page a ainsi un score selon le nombre de pages pointant vers elle, score qu'elle distribue à celles sur lesquelles elles pointent.

Le mode de calcul

Pour préciser les calculs, imaginons la partie du Web traitant du sujet cherché limitée à quatre pages.
Pour simplifier, les pages sont désignées par des lettres. Les flèches désignent les pages vers lesquelles chaque page pointe, c'est-à-dire les pages auxquelles on peut accéder en un clic de souris.



Le produit matriciel

Au centre du calcul effectué dans notre exemple se trouve la matrice $M = \begin{pmatrix} 0 & 1/2 & 0 & 0 \\ 1 & 0 & 1/3 & 0 \\ 0 & 0 & 1/3 & 0 \\ 0 & 1/2 & 1/3 & 1 \end{pmatrix}$ correspondant à la distribution des scores de chaque page. C'est pourquoi la somme des termes de chaque colonne est égale à 1. Partant de la distribution initiale $X = \begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix}$, les distributions suivantes sont MX, M^2X, M^3X, *etc.* où le produit matriciel s'opère de la façon suivante :

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 \\ b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 \\ c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 \\ d_1x_1 + d_2x_2 + d_3x_3 + d_4x_4 \end{pmatrix}.$$

La stabilisation des pages dépend donc des puissances de la matrice M. La question se trouve ainsi liée aux valeurs propres de M et on montre que, en général, cette suite converge (voir *le Théorème de Perron-Frobenius* dans *Tangente Sup* 44–45).

Pour déterminer le score de chaque page, on attribue d'abord à chacune le même score (égal à 25 % ici). Ensuite, on tient compte des interférences entre les pages. Ainsi, la page B pointe sur les deux pages A et D, auxquelles elle va donc donner la moitié de son score. De même, elle reçoit le score de la page A et le tiers de celui de la page C. Ces diverses formules sont appliquées plusieurs fois, et donnent le tableau suivant :

Page	Score 1	Score 2	Score 3
A	25 %	12,5 %	16,66 %
B	25 %	33,33 %	15,27 %
C	25 %	8,33 %	2,77 %
D	25 %	45,83 %	65,27 %
Page	**Score 4**	**Score 5**	**Score 6**
A	7,63 %	8,79 %	3,97 %
B	17,59 %	7,94 %	8,90 %
C	0,92 %	0,31 %	0,10 %
D	73,84 %	82,94 %	87,02 %

Il est inutile de continuer longtemps pour que le système se stabilise. Il en ressort le classement des pages : D, B, A et C, dans cet ordre. Les moteurs de recherche opèrent ainsi, mais avec des milliers de pages. Mathématiquement, cette façon de procéder appelle le calcul matriciel (voir en encadré).

Le principe est donc récursif, l'algorithme aussi. Les calculs sont énormes, infaisables à la main mais simples dans leurs principes. Si vous avez un site et que vous le désirez visible, ce principe montre qu'il est inutile de payer pour cela, il suffit de le faire connaître ! Certains sites pointeront dessus et l'affaire sera lancée.

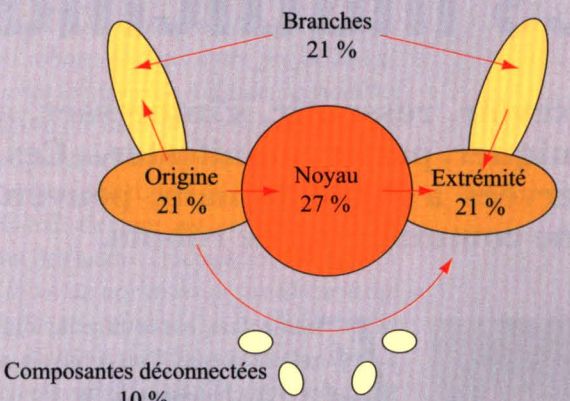
H. L.

Références

- *Comment Google classe les pages*. *Tangente Sup* 44–45, 2008.
- *Le théorème de Perron-Frobenius*. *Tangente Sup* 44–45, 2008.
- *L'algorithme PageRank de Google : promenade sur la Toile*. *Tangente* 130, 2009.
- *Les matrices*. Bibliothèque Tangente 44, 2012.

Le graphe du Web

Le Web peut être représenté comme un graphe gigantesque. Ce graphe énumère toutes les pages avec leurs connections, celles-ci correspondant aux liens que l'on peut atteindre en cliquant. Ce graphe est immense puisqu'il réunit plusieurs millions de pages. Vu de loin, il prend une forme de nœud papillon.



Le nœud papillon. Les flèches indiquent le sens des liens.

L'essentiel des pages (90 %) sont connectées entre elles, dans un sens, dans l'autre ou les deux. Les autres forment quelques composantes déconnectées. Celles-ci ne pointant qu'entre elles n'intéressent pas le surfeur moyen. En revanche, elles pourraient intéresser la police car il peut s'agir de réseaux de malfaiteurs, comme les pédophiles ou des terroristes par exemple. Les repérer est un problème intéressant aussi bien mathématiquement que pratiquement. Le critère est exactement l'inverse des moteurs de recherche qui, au contraire, classent les pages les plus populaires.

Si l'on exclut ces composantes déconnectées, le Web est constitué d'un gros noyau central, contenant 27 % des pages. Leur particularité est d'être accessibles les unes des autres en quelques clics. Autour de ce noyau, on trouve les pages liées au noyau dans un sens ou dans l'autre (on y va ou on en vient), plus deux branches non accessibles du noyau mais connectées en sens inverse.

À un niveau plus local, on peut reconnaître des structures particulières, comme des groupes de pages pointant toutes vers la même ou les mêmes, l'inverse étant faux. Ces structures représentent des groupes de fans, que l'on peut ainsi retrouver.

Entre le robot et l'homme, les mathématiques

Percevoir, ressentir, s'intéresser, apprendre, inventer, communiquer, parler, manifester... Ces fonctions, que l'on croyait réservées à l'être humain, peuvent être modélisées au sein d'une communauté de robots.

Pierre-Yves Oudeyer est chercheur à l'Inria Bordeaux-Sud-Ouest.

Ergo-robots est le titre de l'installation que l'on trouvait au rez-de-chaussée de la Fondation Cartier, à l'occasion de l'exposition « Mathématiques, un dépaysement soudain » dans un grand œuf de quatre mètres d'envergure qui accueille une communauté de robots curieux (voir *Tangente* 143, 2012). Curieux de leur environnement, de leurs congénères. Si ces robots en viennent un jour à croire en un dieu, ils pourraient lui donner le visage de Pierre-Yves Oudeyer, le chercheur de l'Inria qui, avec son équipe, les a conçus.

Au-delà de leur curiosité, les robots ont été programmés pour se prêter à l'expression orale. Tout se passe comme dans une civilisation naissante. Les individus disposent d'un système vocal qui leur permet de s'exprimer à l'aide d'une distribution de sons, au départ aléatoire, et comparable à celle des langues du monde humain. Pour désigner les objets de leur environnement ou les actions de leurs congénères, ils inventent des mots, pas forcément compris du premier coup par les autres robots. C'est là qu'intervient, au cours de jeux de langage, l'expression : exploration, acquiescement ou négation (selon le succès ou l'échec), impatience, joie ou colère… Elle est matérialisée par la couleur des lampes qui les éclairent de l'intérieur. Petit à petit, les individus se comprennent et la langue se construit.

Pierre-Yves Oudeyer dirige l'équipe Flowers Fields de l'Inria qui a mis au point, en collaboration avec le Labri (laboratoire bordelais de recherche en informatique), l'intelligence artificielle de ce modèle individuo-centré, physiquement et socialement situé. Il a expliqué à l'équipe de *Tangente* l'importance des mathématiques dans sa réalisation.

La deuxième vie de l'IA

IA : intelligence artificielle. Cette branche de l'informatique, qui a pour origine les travaux d'Alan Turing en 1950, a nourri bien des ambitions (des illusions ?) dans les années 1980. Pour faire réfléchir des ordinateurs comme des humains, on a inventé les systèmes

experts, ces logiciels capables de manipuler des règles introduites par les programmeurs et exploitées à l'aide de moteurs d'inférence, reproduisant le raisonnement humain.

***Tangente* : Où en est aujourd'hui l'intelligence artificielle ?**

« Dans un premier temps limitée par la complexité des algorithmes, comparée aux capacités des microprocesseurs, l'IA bénéficie aujourd'hui des progrès des capacités de mémoire des ordinateurs et des recherches mathématiques. Elle est à la base de technologies modernes telles que les moteurs de recherche de type Google, mais un obstacle subsistait : l'approche de la cognition est symbolique *; or la connexion à la réalité est une difficulté importante que l'on retrouve quand il s'agit de programmer des robots réels. Des tâches élémentaires pour les humains deviennent d'énormes gageures. On s'est alors retournés vers une vieille idée, déjà évoquée par Turing : faire apprendre comme un enfant. La notion d'auto-apprentissage, à laquelle on revient depuis une dizaine d'années, allait donner une seconde vie à l'IA. »*

Et les mathématiques dans tout cela ?

« Elles sont fondamentales. Certains outils ne sont pas très exotiques : l'apprentissage automatique est un domaine de recherche à l'intersection de l'IA, de l'inférence statistique et de l'optimisation. Les algorithmes, briques élémentaires de l'architecture, proviennent de diverses sources : pour comprendre le vivant, Turing avait déjà imaginé les automates cellulaires ; pour simuler le fonctionnement du cerveau, les réseaux de neurones permettent une approche reconstituant la façon dont il traite l'information. Pour les systèmes complexes, comme ceux de la physique statistique, on utilise des équations différentielles couplées, en particulier pour créer des boucles de rétroaction, qui permettent au système de s'autoorganiser de la bonne manière. Tous ces outils sont exploités pour modéliser l'apprentissage des robots. Et dans notre projet, la curiosité est simulée à l'aide de modèles encore au stade de la recherche. C'est ce qui nous a rapprochés de Misha Gromov [NDLR : le mathématicien Mikhaïl Gromov est en poste à l'Institut des hautes études scientifiques]. *Il a lu un de nos articles sur la curiosité artificielle, consacré à la modélisation du plaisir engendré par la croissance des connaissances : la problématique des modèles informatiques faisant la différence entre les motivations intrinsèques et extrinsèques était également au cœur de ses travaux sur les ergosystèmes ! »*

De la « curiosité artificielle » ? Voilà qui éveille la nôtre ! Pierre-Yves Oudeyer explique qu'elle est constituée de trois modules.

• Le premier module réalise un apprentissage statistique classique, qui consiste pour le robot à construire un modèle prédictif du résultat de ses actes, et à pouvoir comparer, pour chaque action, sa prédiction avec la réalité.

• Le deuxième module, au-dessus, est un système de méta-prédiction. Les prédictions de premier niveau sont soumises à une évaluation qui va servir à mesurer l'intérêt de chaque situation explorée par le robot. Plus cette mesure sera élevée, plus la curiosité du robot sera éveillée et l'incitera à poursuivre son apprentissage. Si la

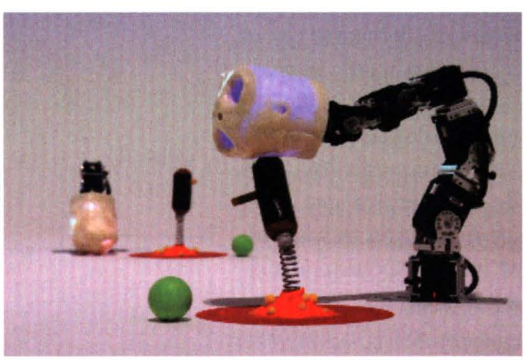
prédiction s'avère presque toujours exacte (situations simples), il n'y aura pas de surprise, il suffira d'entériner les conclusions. On pourrait alors penser, à l'inverse, à faire en sorte que ce soient les situations les plus complexes (où les prédictions sont toujours mauvaises) qui éveillent au maximum sa curiosité. C'est une mauvaise idée. Elle déboucherait sur un comportement qui ne ferait pas forcément faire de progrès. La mesure choisie, plus sophistiquée, repose sur l'étude de la dérivée d'une fonction mesurant l'évolution des erreurs de prédiction dans le temps. On part d'une situation imprédictible, puis on mesure les progrès. S'ils sont significatifs, cela crée de l'intérêt. Quand le progrès ralentit, on s'intéresse à autre chose.

• C'est l'objet du troisième module : optimiser son plaisir ! Le système optimise le choix de ses actions, donc des situations qu'il va explorer, de manière à provoquer le plus de progrès en apprentissage, donc de curiosité et de plaisir. Les activités sont automatiquement ordonnées suivant une échelle de complexité croissante, les plus simples et les plus complexes étant éliminées.

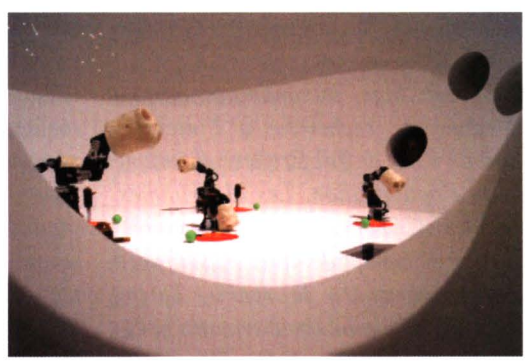
Voilà qui remet en selle la proximité entre science et art : comprendre, mais aussi créer une nouvelle représentation du monde. « Si on remplace dans le modèle le mot intérêt par le mot esthétique, on construit une relation mathématique entre objets extérieurs et modèles subjectifs du monde. En particulier, l'esthétique d'un résultat scientifique pourrait dériver d'une combinaison entre simplicité et amélioration des théories existantes. »

Propos recueillis par G.C.

Le design des têtes des robots a été conçu par David Lynch.



Les robots apprennent à interagir avec leur environnement.



Enfermés dans un espace confiné, les robots apprennent à communiquer en développant un langage.

Pour contribuer à la transmission de la culture mathématique,
LE MAGAZINE TANGENTE ORGANISE

LES TROPHÉES tangente

DÉPOSEZ VOS CANDIDATURES CHAQUE ANNÉE AVANT LE 31 AOUT

Prix Bernard-Novelli

Soutenir les jeunes créateurs

Vous êtes collègiens, lycéens ? Vous aimez concevoir des jeux logiques faisant intervenir les mathématiques ? Participez à ce concours et transformez votre projet en un produit mobile disponible sur l'App Store et Google Play

Prix Tangente du livre

Donner envie d'en savoir plus

A partir du 15 juin, participez à la sélection des ouvrages en votant sur : www.tropheestangente.com

Prix Tangente du meilleur article

Raconter les mathématiques

Envoyez-nous un article inédit vulgarisant un concept mathématique. Le meilleur sera publié au sein du magazine Tangente

CASIO Magma Mobile Your Joyful Escape Inria INVENTEURS DU MONDE NUMÉRIQUE

Inscription, informations et modalités de participation sur le site
www.tropheestangente.com

Prologin POUR LA SCIENCE SIF APMEP Société Mathématique de France SMF

par Hervé Lehning

La cryptographie, à l'origine de l'informatique

La réalisation des premiers ordinateurs a mobilisé des moyens énormes. La dépense était à l'époque justifiée par l'urgence de décrypter les messages chiffrés allemands, un travail qui demande de colossaux moyens de calcul.

Décrypter a toujours demandé de l'astuce et des calculs. Pour le montrer, prenons le code le plus simple qui existe, celui de Jules César, qui consiste en un décalage de lettres. La clef du chiffrement est donc ici le décalage, c'est-à-dire un nombre entre 1 et 25. Prenons un exemple simple, dont on ignore la clef :

WYMNM SGJUX YXYWI XYL,

où les lettres ont été regroupées par cinq comme c'est l'usage. L'idée la plus simple pour le décrypter est d'écrire, sous chaque lettre, l'alphabet dans l'ordre :

W	Y	M	N	M	S	G	J	U	X	Y	X	Y	W	I	X	Y	L
X	Z	N	O	N	T	H	K	V	Y	Z	Y	Z	X	K	Y	Z	M
Y	A	O	P	O	U	I	L	W	Z	A	Z	A	Y	K	Z	A	N
Z	B	P	Q	P	V	J	M	X	A	B	A	B	Z	L	A	B	O
A	C	Q	S	Q	W	K	N	Y	B	C	B	C	A	M	B	C	P
B	D	R	S	R	X	L	O	Z	C	D	C	D	B	N	C	D	Q
C	**E**	**S**	**T**	**S**	**Y**	**M**	**P**	**A**	**D**	**E**	**D**	**E**	**C**	**O**	**D**	**E**	**R**

Décryptement d'un message chiffré par le code de Jules César. On peut par exemple utiliser des bandes de papier portant chacune l'alphabet dans l'ordre. Avec une règle, on examine chaque alignement et on s'arrête dès que le texte a un sens.

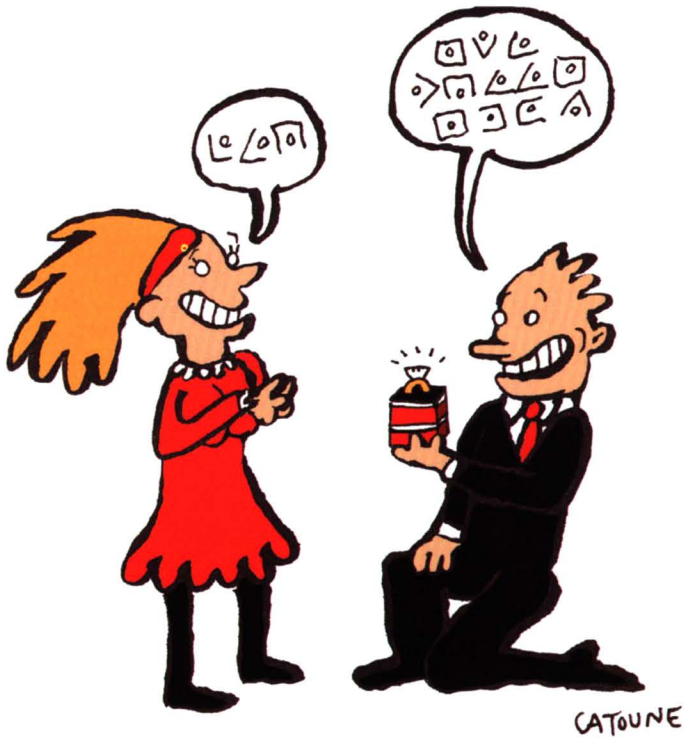
© Dominique Beudin

L'auteur explique à Cédric Villani comment utiliser une C-36 sous le regard du général Desvignes, président de l'Association des réservistes du chiffre et de la sécurité de l'information (ARCSI).

Une des vingt-cinq lignes que l'on peut écrire ainsi correspond au message en clair (voir le tableau). Une idée plus subtile pour décrypter ce message est d'utiliser la fréquence des lettres en français. Dans le message chiffré, le Y est la lettre la plus fréquente. Elle correspond probablement au E, la lettre la plus fréquente en français. Pour transformer Y en E, il faut le décaler de 6 dans l'ordre alphabétique : Y, Z, A, B, C, D, E. En décalant de même chaque lettre du message chiffré, on obtient bien le même décryptement.

Les chiffres à substitution poly-alphabétique

Le premier chiffre à substitution poly-alphabétique de l'histoire est connu sous le nom de chiffre de Vigenère car, même s'il fut utilisé avant lui, Blaise de Vigenère (1523–1596) fut le premier à le décrire clairement. Dans sa version actuelle, le chiffre de Vigenère est un César à décalage variable. En guise d'exemple, chiffrons le message « le chiffre de Vigenère est plus solide que celui de César » avec la clef « art ». Pour cela, on écrit le message dans un tableau sans espace ni ponctuation puis la clef en dessous, en la répétant autant de fois que nécessaire. Le décalage dépend du numéro d'ordre de la lettre considérée, 0 pour A, 1 pour B, 4 pour E, *etc*. On applique alors le décalage indiqué sous chaque lettre du message.



L	E	C	H	I	F	F	R	E	D	E	V
A	R	T	A	R	T	A	R	T	A	R	T
0	17	19	0	17	19	0	17	19	0	17	19
L	V	V	H	Z	Y	F	I	X	D	V	O

I	G	E	N	E	R	E	E	S	T	P	L
A	R	T	A	R	T	A	R	T	A	R	T
0	17	19	0	17	19	0	17	19	0	17	19
I	X	X	N	V	K	E	V	L	T	G	E
U	S	S	O	L	I	D	E	Q	U	E	C

A	R	T	A	R	T	A	R	T	A	R	T
0	17	19	0	17	19	0	17	19	0	17	19
U	J	L	O	C	B	D	V	J	U	V	V
E	L	U	I	D	E	C	E	S	A	R	
A	R	T	A	R	T	A	R	T	A	R	
0	17	19	0	17	19	0	17	19	0	17	
E	C	N	I	U	X	C	V	L	A	I	

Codage de Vigenère.
Le chiffre de Gronsfeld est identique mais la clef est donnée sous la forme d'une liste de nombres (« 0 17 19 » ici).

Le chiffrement est assez laborieux puisqu'il faut décaler chaque lettre de façon distincte, ce qui explique que la méthode ait été pratiquement ignorée pendant presque trois siècles. La moindre erreur pouvait rendre le message indéchiffrable pour son destinataire.

Les machines à chiffrer mécaniques ou électro-mécaniques de la Seconde Guerre mondiale effectuent des substitutions poly-alphabétiques de manière automatique et déchiffrent de même. La plus célèbre d'entre elles est la machine allemande Enigma, mais il en a existé un grand nombre comme, par exemple, la C-36 utilisée au niveau tactique par l'armée française pendant cette guerre. En la découvrant au Salon de la culture et des jeux mathématiques de 2014, Cédric Villani s'est d'ailleurs exclamé : « *Oh, un bébé Enigma !* » Il avait parfaitement raison, ces deux machines sont effectivement de la même famille.

Le maillon faible de la cryptographie est toujours l'être humain.

La notion d'indice de coïncidence

L'indice de coïncidence d'un texte T est la probabilité que deux lettres tirées au hasard dans ce texte coïncident. Si le nombre de A est égal à n_A, le nombre de couples de deux A est égal à $\frac{n_A(n_A-1)}{2}$, celui de B, $\frac{n_B(n_B-1)}{2}$, *etc.* En faisant la somme de tous ces nombres, on trouve le nombre de couples formés de deux lettres identiques. Le nombre de couples quelconques dans le texte est égal à $\frac{n(n-1)}{2}$ donc la probabilité pour que deux lettres d'un texte coïncident vaut :

$$I_C = \frac{n_A(n_A-1)+n_B(n_B-1)+\ldots+n_Z(n_Z-1)}{n(n-1)}.$$

Si on note $f_A, f_B \ldots f_Z$ les fréquences des lettres dans le texte ($n_A = nf_A$ par exemple), on peut écrire :

$$I_C = \frac{nf_A(nf_A-1)+\ldots}{n(n-1)} = \frac{n}{n-1}\left(f_A^2+\ldots\right) - \frac{1}{n-1}(f_A+\ldots).$$

On en déduit que $I_C = \frac{n}{n-1}S - \frac{1}{n-1}$ où S est la somme des carrés des fréquences des lettres.

Dans les applications, on peut donc employer S au lieu de l'indice de coïncidences car, si le texte est relativement long, n et $n-1$ sont à peu près égaux et 1 est négligeable devant n donc S et I_C peuvent être confondus.

Pour déterminer la longueur d'une clef de Vigenère, William Friedman (1891–1969) eut l'idée d'introduire un indice invariable par permutation des lettres. Ainsi, sa valeur n'est pas affectée par une substitution alphabétique simple, telle celles engendrées par le code de César. Pour obtenir un tel indice, l'idée la plus simple est d'additionner les carrés des fréquences des lettres. Plus précisément, dans un texte T donné, on compte le nombre d'occurrences de chaque lettre : n_A pour A, n_B pour B… et n_Z pour Z. On en fait la somme n, qui est donc la longueur du texte, puis la somme des carrés ${n_A}^2, {n_B}^2 \ldots {n_Z}^2$. On divise enfin cette dernière somme par le carré de n. Si on échange les lettres, les positions des carrés des fréquences sont modifiées dans la somme mais celle-ci reste identique puisque tous les carrés s'y retrouvent malgré la permutation. Ce nombre n'ayant pas de sens concret, on préfère en utiliser un autre, qui est très proche, appelé *indice de coïncidence* (voir l'encadré). Sans moyen électronique, le calcul est très long donc impraticable, d'où le besoin de créer un moyen de calcul puissant et fiable. À partir des fréquences usuelles, on peut déterminer l'indice de coïncidence moyen d'un texte français, ou dans d'autres langues, éventuellement codé par un chiffre tel celui de César ! Il suffit de faire la somme des fréquences moyennes au carré.

A	B	C	D	E	F	G
8,4	1,1	3	4,2	17,3	1,1	1,3
H	I	J	K	L	M	N
0,9	7,3	0,3	0,1	6	3	7,1
O	P	Q	R	S	T	U
5,3	3,0	1	6,5	8,1	7,1	5,7
V	W	X	Y	Z		
1,3	0,1	0,4	0,3	0,1		

Tableau de fréquences des lettres en français, exprimées en pourcentages.

En français, l'indice de coïncidence moyen est donc égal à la somme $0{,}084^2 + 0{,}011^2 + \ldots + 0{,}001^2$, soit 0,0746. Dans le cas d'un message où les lettres seraient choisies au hasard, on trouve une fréquence moyenne de 1/26. L'indice est donc égal à 26 fois $(1/26)^2$, c'est-à-dire à 1/26, soit 0,038. Il s'agit de l'indice de coïncidence moyen d'un texte aléatoire.

Prenons un message chiffré par la méthode de Vigenère, par exemple :

CVXIA UWQKU ZMEIO ITKTJ YWPFB RGTCP
VJUME JVNCI WRBVZ ZJYWK NVFRV NXIZF
ITKJX ZHLFX BAIOC SBIME RRGZR NHMHX
UZWEJ YMSKC GPJVA KGOFI YIKRA DAVGP
WTRAF GXZDW VABNY JZKPV AOFUL
VDWLM VBSSM PYEHV RLFDD PHYWA
MLZFV VOBRT LZTPY RAGKR NFFKV
DVYVM WITRA FGXZP RWYCF OVPCW
TRAFG XZDUB VARZI JFZLA BPUUZ DTHET
RIYDQ JYRLR IVNLV SNURZ YJOIK RASXV
LFIUP MFVVM XIAQM PUEXW YYR.

Son indice de coïncidence est égal à 0,042, plus proche de celui d'un texte aléatoire que celui d'un texte moyen. Il n'est donc pas chiffré au moyen d'une substitution mono-alphabétique, la clef n'est pas de longueur 1. On considère alors les textes obtenus en partant de la première lettre et en ne gardant qu'une lettre sur deux, une lettre sur trois, *etc.* On obtient des indices de coïncidence compris entre celui d'un texte aléatoire et celui d'un texte moyen :

Clef	1	2	3	4	5
Indice	0,04	0,04	0,06	0,05	0,04
Clef	6	7	8	9	
Indice	0,06	0,04	0,03	0,08	

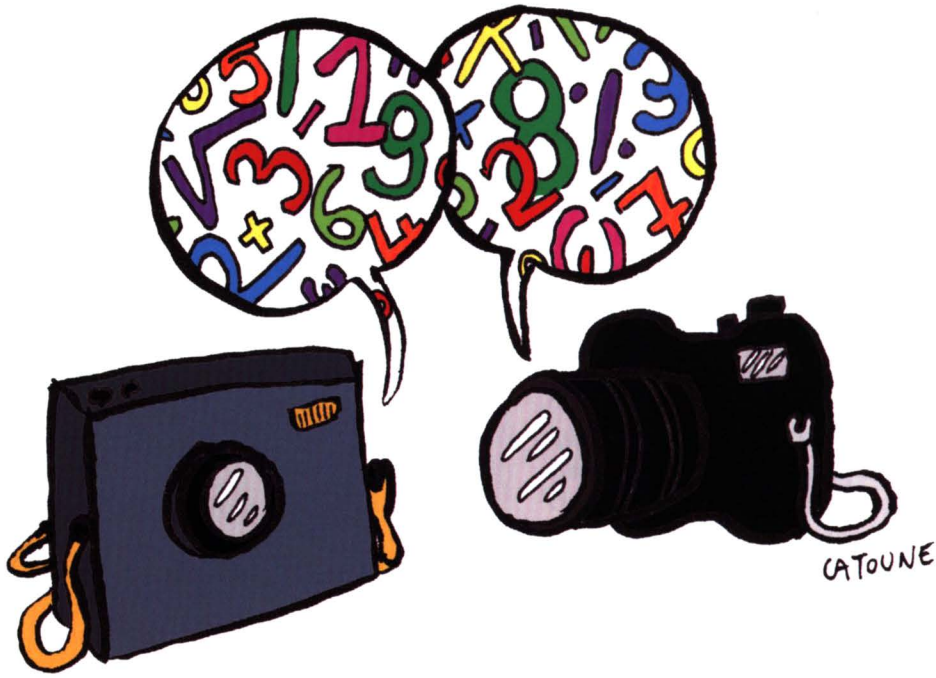
Mesure des indices de coïncidence.

Arrivé à 9, on trouve un indice proche de l'indice moyen. Ainsi, par un simple calcul, on découvre que la longueur de la clef est probablement égale à 9. Si cette hypothèse est correcte, on produit neuf messages chiffrés par simple décalage dont le premier est
CUTTV ZVJIR UCFVX JLSFL LRVXV XIUYV YVVE
(en partant de la première lettre puis en écrivant une lettre sur neuf). La lettre la plus fréquente est V (plus de 23 %), elle représente sans doute E. La première lettre de la clef est donc R. En faisant de même pour les huit autres groupes, la clef est RVLEHNING et le message donne une autre méthode de décryptement, moins systématique mais aussi moins calculatoire. Elle est due au précurseur de l'informatique Charles Babbage :
« La méthode de Babbage consiste à chercher des répétitions pour trouver la longueur de la clef. Elle fonctionne dès que les messages sont assez longs, ou assez nombreux. Une fois la longueur obtenue, il reste à subdiviser le message en plusieurs messages, qui se trouvent codés par le chiffre de César. La méthode des fréquences permet de conclure. »

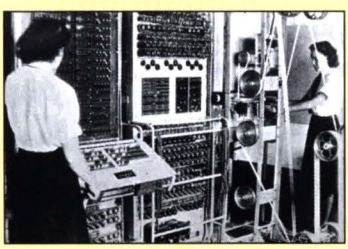
Ce type de calcul permet de décrypter tout message des machines à chiffrer de la famille de l'Enigma allemande, s'il est assez long. Les erreurs des opérateurs allemands, dont la principale était de chiffrer les bulletins météoro-

© H. Lehning

La C-36 ouverte. Elle sera améliorée en la C-38, qui deviendra la célèbre M-209 de l'armée américaine.



Colossus, le premier ordinateur



Colossus, premier ordinateur à Bletchley Park pendant la Seconde Guerre mondiale.

Le premier ordinateur (Colossus) fut construit par les Britanniques pendant la Seconde Guerre mondiale pour décrypter le code d'une machine de chiffrement, Lorenz, réservée aux hauts dirigeants allemands, alors qu'Enigma servait sur le champ de bataille, en particulier dans les sous-marins. Le code de Lorenz et le relatif faible nombre de messages le permettaient.

Après la guerre, ces progrès menèrent à l'introduction de nouvelles méthodes, comme les clefs asymétriques où, contrairement aux clefs symétriques, savoir coder ne suffit pas pour savoir décoder. La plus célèbre et la plus utilisée d'entre elles, en particulier dans les cartes bancaires et sur Internet, est la méthode RSA, du nom des trois inventeurs : Rivest, Shamir et Adleman. Elle repose sur la difficulté pratique de factoriser les nombres quand ils sont grands.

logiques, forcément prévisibles, permirent à Alan Turing d'utiliser la méthode du mot probable avec succès.

Autrement dit, une machine peut être idéale et potentiellement impossible à décrypter, mais le maillon faible reste toujours l'être humain. C'est pourquoi, il est important que les opérateurs soient bien formés et les notices d'utilisation claires. Ainsi, celle de la C-36, machine à chiffrer de l'armée française au cours de la Seconde Guerre mondiale, précise de limiter les messages à quatre-vingts caractères et de changer la clef après chacun d'eux…

Références

- *Cryptographie et codes secrets*. Tangente Bibliothèque 26, 2013.
- *L'univers des codes secrets, de l'Antiquité à Internet*. Hervé Lehning, Ixelles, 2012.
- Dossier « Codes secrets », *Tangente* 147, 2012.

H. L.

Avec ou sans clavier

HS5201 – Un petit calcul R ❍

Sans l'aide d'un ordinateur ou d'un logiciel de calcul, **pouvez-vous calculer cette expression ?**

$$\frac{1\,234\,567\,890}{1\,234\,567\,891^2 - 1\,234\,567\,890 \times 1\,234\,567\,892}$$

HS5202 – Un nombre à trouver ✔✔

Un jeu sur ordinateur fonctionne de la manière suivante. L'ordinateur choisit aléatoirement un nombre entier entre 1 et n. Le joueur a droit à k essais pour trouver le nombre choisi. Après chaque essai, l'ordinateur répond « Exact », « Trop grand » ou « Trop petit ».
Pour des valeurs fixées de k et de n, déterminez la probabilité de réussite et trouvez la stratégie optimale.

HS5203 – Mathématiciens et informaticiens (1) ✔

Trois personnes, A, B et C, chacune d'elle étant soit mathématicien(ne) soit informaticien(ne), ont la discussion suivante :

A : C et moi sommes mathématiciens ;
B : C n'est pas mathématicien ;
C : B est mathématicien ou A est informaticien.
Sachant que les mathématiciens disent toujours la vérité alors que les informaticiens mentent systématiquement, pouvez-vous dire qui est quoi ?

HS5204 – Mathématiciens et informaticiens (2) ✔

Trois autres personnes, D, E et F, chacune d'elle étant soit mathématicien(ne) soit informaticien(ne), ont la discussion suivante :
D : F est informaticien ;
E : D et F sont mathématiciens ;
F : E est mathématicien.
Sachant que les mathématiciens disent toujours la vérité alors que les informaticiens mentant systématiquement, pouvez-vous dire qui est quoi ?

HS5205 – La calculatrice de l'année ✔✔

Cette calculatrice ne sait faire qu'une opération : la multiplication de deux nombres. Il est impossible de lui entrer un nombre au clavier ! Les seuls nombres qu'elle puisse utiliser sont ceux qui sont dans sa mémoire, et elle garde systématiquement en mémoire les résultats de tous les calculs qu'elle effectue. On peut rappeler à tout moment n'importe quel nombre se trouvant dans sa mémoire. Au départ, seul le nombre 2014 est dans sa mémoire.

SOURCES DES PROBLÈMES

- *Ontario Mathematical Gazette* (HS5201)
- D'après l'*American Mathematical Monthly* (HS5202)
- D'après *Ontario Secondary School Mathematics Bulletin* (HS5203, HS5204)
- D'après Championnat des jeux mathématiques et logiques (HS5205, HS5206)
- D'après la revue *Parabola* (HS5207)
- D'après la revue *Function* (HS5208)
- D'après le *Journal of Recreational Mathematics* (HS5209, HS5210)

Quel est le nombre minimum de multiplications qu'elle doit effectuer pour calculer 2014 × 2014 ?

HS5206 – Un clavier défectueux ✔✔

Sur les neuf touches 1, 2, 3, 4, 5, 6, 7, 8, 9 du clavier de Mathias, seules trois fonctionnent, et la touche 0 ne fonctionne pas. Sur son ordinateur, Mathias additionne les six nombres s'écrivant avec trois chiffres distincts qu'il peut encore taper avec les trois touches rescapées, et il constate avec amusement que le total s'écrit en n'utilisant que les chiffres de ces trois touches. **Quelles sont les trois touches numériques qui fonctionnent encore sur le clavier de Mathias ?**

HS5207 – Un ordinateur et ses nombres ✔✔✔

Un ordinateur est capable de calculer l'inverse de n'importe quel nombre non nul qu'il a en mémoire et la somme de deux nombres quelconques qu'il a mémorisés. Il peut également mémoriser tous les nombres qu'il a calculés. Au départ, il ne connait que le nombre 100. **Quels sont tous les nombres qu'il peut produire ?**

HS5208 – Puissances successives ✔✔

Si $(1 + \sqrt{2})^n = a + b\sqrt{2}$ où a, b et n sont des entiers strictement positifs, **démontrez que a est l'entier le plus proche de $b\sqrt{2}$.**

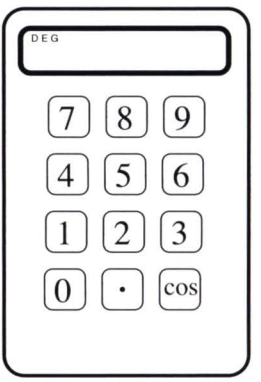
HS5209 – Des irrationnels presque entiers ? ✔✔✔

Le professeur Cosinus calcule à l'aide d'un tableur les puissances successives du nombre $3 + \sqrt{8}$. Il constate avec stupeur qu'à partir d'un certain rang, les résultats sont tous entiers, alors qu'il sait pertinemment que les puissances de ce nombre sont toutes irrationnelles. Il recommence avec le nombre $4 + \sqrt{15}$ et fait le même constat.
Comment pouvez-vous expliquer cela ?

Niveau de difficulté

❍	très facile
✔	facile
✔✔	pas facile
✔✔✔	difficile
✔✔✔✔	très difficile

HS5210 – Le singe et la calculatrice ✔✔



Un jour, un singe tapa dix fois sur des touches d'une calculatrice. Un nombre s'afficha à l'écran.
Le calculateur simiesque posa alors son doigt sur la touche « cos », mais rien ne se produisit. Tout en grimaçant et en poussant de petits cris, il appuya à nouveau plusieurs fois sur la touche « cos » ; hélas, le nombre affiché restait identique.
Au, fait, quel était ce nombre ? (On donnera la troncature de ce nombre avec six chiffres après la virgule.)
Remarque : comme l'indiquent les lettres DEG sur l'écran de la calculatrice, celle-ci est en mode degrés. Une calculatrice scientifique est nécessaire pour résoudre ce problème.

HS5211 – Les bonbons ✔✔✔

Cette sorte de bonbons existe en k arômes et j'ai n exemplaires de chaque arôme. Je veux les déguster un par un en ne mangeant jamais trois bonbons ayant le même arôme à la suite.
De combien de manières puis-je le faire si j'ai trois arômes différents et trois bonbons de chaque arôme ?

HS5201 –
$1234567891^2 - 1234567890 \times 1234567892$
$= 1234567891^2 - (1234567891^2 - 1) = 1$.
On en déduit l'expression demandée, égale à **123 467 890**.

HS5202 – Si $n \leq 2^k - 1$, il existe une stratégie optimale permettant de trouver le nombre à coup sûr. On procède par dichotomie, en proposant systématiquement le nombre situé au milieu de l'intervalle considéré. Si $n > 2^k - 1$, on procède de même, avec une probabilité de trouver le nombre cherché égale à $(2^k - 1)/n$.

HS5203 – Si A disait la vérité, B et C mentiraient tous les deux, ce qui entraînerait une contradiction. Donc **A est informaticien**. On en déduit que **C**, qui dit la vérité, **est mathématicien et que B est informaticien**.

HS5204 – Si D ment, F et E sont mathématiciens, et on aboutit à une contradiction pour D qui ne peut être mathématicien et mentir. Si D dit la vérité, F ment et E est informaticien. Il ment bien puisque D et F ne sont pas tous deux mathématiciens. Donc **D est mathématicien et E et F sont informaticiens**.

HS5205 – 16 multiplications suffisent :
$2014^{2014} = (((((((2014 \times 2014 \times 2014)^3)^2)^2 \times 2014)^3)^3 \times 2014^2)^3 \times 2014^2)^2$.

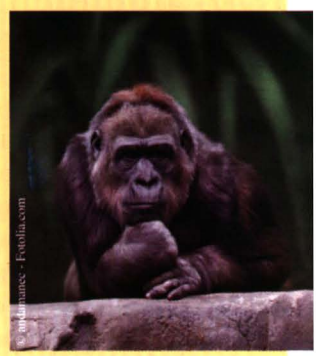
HS5206 – Si a, b, c sont les trois chiffres cherchés, le total des six nombres est égal à $222(a + b + c)$. La somme des trois chiffres peut prendre toutes les valeurs entières de 6 à 24, soit dix-neuf valeurs. Seules les treize suivantes utilisent trois chiffres distincts non nuls : 1332, 1554, 1776, 1998, 2664, 2886, 3552, 3774, 3996, 4662, 4884, 5772, 5994. Trois sont des solutions au problème posé : $1332 = (1+2+3) \times 222, 2664 = (2+4+6) \times 222$ et $3996 = (3 + 6 + 9) \times 222$.

HS5207 – L'ordinateur peut calculer l'ensemble des multiples de 0,01. Il peut donc obtenir, 1 000, 10 000, 100 000, *etc.* et leurs inverses, donc tous les nombres décimaux strictement positifs, puis, en inversant les nombres entiers non nuls et en les additionnant, tous les nombres rationnels strictement positifs.

HS5208 – Si $(\sqrt{2} + 1)^n = a + b\sqrt{2}$ où a, b et n sont des entiers strictement positifs, $0 < (\sqrt{2} - 1)^n < 0{,}5$. On a donc :
$2a < (\sqrt{2} + 1)^n + (\sqrt{2} - 1)^n < 2a + 0{,}5$ et
$2b - 0{,}5 < (\sqrt{2} + 1)^n - (\sqrt{2} - 1)^n < 2b$.
On en déduit la propriété demandée.

HS5209 – Le nombre $(3 + \sqrt{8})^n + (3 - \sqrt{8})^n$ est un nombre entier. Or le second terme de cette somme tend rapidement vers 0 quand n augmente. On en déduit que $(3 + \sqrt{8})^n$ tend rapidement vers un nombre entier quand n tend vers l'infini. On fait le même raisonnement avec $4 + \sqrt{15}$ et avec tous les nombres de la forme $a + \sqrt{(a^2 - 1)}$.

HS5210 – Un essai à partir de (presque) n'importe quel nombre a saisi sur une calculatrice scientifique à huit chiffres, donne, après au plus trois appuis successifs sur la touche COS, 0,9998477. En effet, $-1 < \cos a < 1$, et cos est croissante entre – 1 et 0 puis décroissante entre 0 et 1. Donc, dans les deux cas, $0{,}9998476951 < \cos(\cos a°) < 1$. En itérant, on obtient : $0{,}9998476951 < \cos(\cos(\cos a°)) < 0{,}9998477416$.



HS5211 – En choisissant les bonbons tour à tour, on a $9! / (3!)3 = 1680$ choix possibles. Il faut déduire de ces choix les cas où trois bonbons ayant le même arôme se suivent. Le premier de ces trois bonbons peut occuper sept places différentes et il reste vingt choix pour les six autres bonbons. Ce faisant, on enlève deux fois les cas où il y a deux séries d'arômes et trois fois ceux il y a trois séries de trois arômes identiques qui se suivent. Il faut donc les rajouter. Un peu de dénombrement montre que le nombre demandé est égal à $1680 - 420 + 42 + 12$, soit **1314 possibilités**.

Abonnez-vous à tangente

l'aventure mathématique

••• Tangente le magazine des mathématiques •••

Pour mieux comprendre le monde : *Tangente*
Le seul magazine au monde sur les mathématiques.
Tous les deux mois depuis 25 ans.

••• Les hors-séries Bibliothèque Tangente •••

Ce sont de magnifiques ouvrages d'en moyenne 160 pages (prix unitaire 19,80€), richement illustrés, approfondissant le sujet du dernier numéro des HS « kiosque » de *Tangente*.
Disponibles
- chez votre libraire
- avec **l'abonnement SUPERPLUS**
- avec **l'abonnement Math++**

à un prix exceptionnel (33% de réduction).

••• Tangente Sup •••

6 numéros par an (ou 4 dont 2 doubles), destinés à ceux qui veulent aller plus loin ou aux étudiants de premier cycle. Dans chaque numéro, un dossier : Poincaré, Coniques et quadriques…

••• Les hors-séries « kiosque » •••

4 fois par an, un hors-série « kiosque » d'au moins 56 pages, explorent l'actualité des grands dossiers du savoir ou de la culture mathématique.

Le calcul intégral, Mathématiques et informatique, Les angles
Disponibles
- chez votre marchand de journaux
- avec **l'abonnement PLUS**
- avec **l'abonnement Math+.**

••• Spécial Logique •••

Nouveau ! Dans la collection Tangente Jeux et Stratégie, un trimestriel contenant près de 200 jeux : tests de logique, grilles à remplir, énigmes mathématiques… Accès numérique gratuit pour les abonnés à la version papier.

••• Tangente Éducation •••

Trimestriel qui traite de thèmes pédagogiques variés : les programmes, les TICE, la formation des enseignants, MathC2+, l'informatique et les sciences du numérique… **Permet l'accès à de nombreuses ressources en ligne.**

---

codif : BIB52

Bulletin d'abonnement à retourner à :

Espace Tangente - 80, Bd Saint-Michel - 75006 PARIS

Nom ..... Prénom .....

Établissement .....

Adresse .....

Code Postal ..... Ville .....

Profession ..... E-mail .....

Oui, je m'abonne à	FRANCE MÉTROPOLITAINE		EUROPE	AUTRES
	1 AN	2 ANS	Supplément par an	
TANGENTE	☐ 36 €	☐ 68 €	☐ + 12 €	☐ + 15 €
TANGENTE PLUS	☐ 56 €	☐ 108 €	☐ + 20 €	☐ + 25 €
TANGENTE SUPERPLUS	☐ 88 €	☐ 172 €	☐ + 24 €	☐ + 30 €
TANGENTE SUP	☐ 25 €	☐ 46 €	☐ + 6 €	☐ + 8 €
TANGENTE ÉDUCATION	☐ 12 €	☐ 22 €	☐ + 2 €	☐ + 3 €
SPÉCIAL LOGIQUE	☐ 19,50 €	☐ 37 €	☐ + 8	☐ + 10,50 €
ABONNEMENT MATH+ *	☐ 105 €	☐ 199 €	☐ + 30	☐ + 30 €
ABONNEMENT MATH++ **	☐ 135 €	☐ 260 €	☐ + 32	☐ + 32 €
ABONNEMENT SOUTIEN ***	☐ 155 €	☐ 300 €	☐ + 35 €	☐ + 35 €

* Tous les titres avec les HS « kiosque ». ** Tous les titres avec les HS Bibliothèque. *** Tous les titres avec les deux HS.

Total à payer ☐

Je joins mon paiement par (établissements scolaires, joindre bon de commande administratif) :

❑ Chèque (uniquement **payable en France**)

❑ Carte (à partir de 30 €) numéro : ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

Date et Signature : crypto : ☐☐☐ Expiration le : ...../.....

Tangente Hors-série n° 52
Mathématiques et informatique

Tangente

Publié par les Éditions POLE
SAS au capital de 42 000 euros

Siège social
80 bd Saint-Michel – 75006 Paris
Commission paritaire : 1016 K 80883
Dépôt légal à parution

Directeur de Publication et de la Rédaction
Gilles COHEN

Rédacteur en chef adjoint
Hervé LEHNING

Secrétaire de rédaction
Édouard THOMAS

Ont collaboré à ce numéro
Sylvie ALAYRANGUES, Nicolas ANCIAUX, Maxime AUDOUIN, Catherine BELLEANNÉE, Michel BIDOIT, Philippe BOULANGER, Martine BRILLLEAUD, Élisabeth BUSSER, Arthur CHARGUÉRAUD, Jean-François COLONNA, François COSTE, Michel CRITON, Franck DANINOS, Jean-Paul DELAHAYE, Colin DE LA HIGUERA, David DELAUNAY, Nicolas DELERUE, Adrien DUFOUR, Jean-Jacques DUPAS, Arnaud DURAND, Jean-Christophe FILLIÂTRE, Christine FROIDEVAUX, Gabriel GOUVINE, Bertrand HAUCHECORNE, Daniel JUSTENS, Robin LAMARCHE-PERRIN, François LAVALLOU, Hervé LEHNING, Christine LEININGER, Jean-Michel MULLER, Benjamen NGUYEN, Jacques NICOLAS, Alexandre TALON, Alain VALETTE, Brigitte VALLÉE, Jill-Jênn VIE, Thierry VIÉVILLE, Alain ZALMANSKI

Maquette
Guillaume GAIDOT, Natacha LAUGIER, Claude LUCCHINI

Visuel couverture : © www.picardieweb.com
Photos : droits réservés
Dessins : Julie Lambert (catoune.com)

Abonnements
abo@poleditions.com
01 47 07 51 15 – Fax : 01 47 07 88 13

Achevé d'imprimer pour le compte des Éditions POLE
sur les presses de l'imprimerie SPEI à Pulnoy (54 France)
Dépôt légal — Juillet 2014

Tangente

Mathématiques & informatique

Une nouvelle ère numérique

- Mathématiques pour l'informatique
- L'informatique pour les mathématiques
- Des applications qui changent le monde

L'informatique est un système de représentation de l'information. L'algèbre booléenne et l'algorithmique sont les outils qui permettent de numériser (« mettre sous forme de nombres »), représenter et manipuler l'information. La logique formelle comme la sémantique cherchent à préciser ce qui peut être formalisé et expliqué à un ordinateur. La théorie du signal permet de faire circuler des données d'un ordinateur à l'autre. La cryptologie vise à étudier la sécurité de ces données qui transitent.
La vérification des programmes s'appuie sur la logique mathématique. De fait, la démonstration automatique, l'expérimentation et la simulation numérique quittent le domaine du rêve pour devenir réalité. Ainsi, l'outil informatique envahit notre quotidien, de la biologie à la finance en passant par l'ingénierie, et bouscule notre vision du monde en soulevant des questions (scientifiques ou éthiques) qui n'avaient jamais été envisagées.
Sans les mathématiques, aucun de ces progrès ne serait possible !

782848 841519

Prix : 19,80 €

EDITIONS POLE