de l'utilité des mathématiques et des mathématiciens

par J.J. Duby
Directeur Scientifique d'IBM Europe

Je voudrais remercier l'A.P.M.E.P. de m'avoir permis de m'exprimer ici, tout d'abord pour des raisons professionnelles, puisqu'en tant que responsable scientifique d'une grande entreprise, membre de la Commission Recherche du C.N.P.F., je suis très soucieux, comme mes collègues, des problèmes de recrutement de scientifiques, et en particulier de mathématiciens : nous savons en effet que les jeunes que vous formez seront les ingénieurs qui viendront dans nos entreprises dans cinq, dix ou quinze ans, et représentent donc l'avenir industriel du pays. Ensuite, pour une raison très personnelle : je suis en quelque sorte un prof de maths défroqué et j'ai gardé vis-à-vis de l'enseignement des mathématiques un sentiment de nostalgie et peut-être même de culpabilité : j'ai un peu le sentiment aujourd'hui de m'acquitter d'une certaine dette...

Il y a deux choses que je voudrais dire aux professeurs de mathématiques, avec toute ma conviction : la première, c'est que l'Industrie a besoin de mathématiques ; la seconde, c'est que l'Industrie a besoin de mathématiciens (quand je parle d'Industrie, je considère l'industrie au sens large, c'est-à-dire pas seulement les secteurs primaires et secondaires, mais également le secteur tertiaire).

Pour illustrer mon propos, je commencerai par deux anecdotes. La première se passe à IBM, au milieu des années 70. Une équipe d'électroniciens avait alors conçu un multiplicateur-diviseur extrêmement rapide. Vous savez que, pour faire des multiplications et des divisions dans un ordinateur, on finit toujours par des opérations élémentaires ("et", "ou") sur des chiffres binaires ou bits, et il faut combiner ces opérations binaires d'une certaine manière pour que cela se passe le plus vite possible. Notre équipe avait donc défini un multiplicateur-diviseur qui allait nettement plus vite que tous ceux qui existaient, ayant quasiment gagné un facteur 10 en ordre de grandeur en vitesse d'exécution. Ce multiplicateur-diviseur avait été inclus dans l'ordinateur scientifique le plus puissant de la gamme IBM de l'époque, qui avait été un grand succès commercial. Et les dirigeants du laboratoire de développement

avaient dit : puisqu'avec dix personnes pendant un an, on a réussi à diviser par dix le temps de multiplication, mettons cent personnes et on va diviser par cent. Or, il s'est trouvé qu'un mathématicien du laboratoire voisin, qui faisait de la théorie des groupes et s'intéressait aux problèmes de complexité d'algorithmes, est tombé un jour, par hasard, sur l'algorithme mis au point par ses collègues électroniciens et a constaté qu'il était déjà remarquablement proche de l'optimum fourni par la théorie, et qu'il était donc inutile d'investir un dollar de plus. Il a réussi — avec quelque mal — à convaincre les dirigeants d'IBM, économisant ainsi plusieurs dizaines de millions de dollars à la compagnie.

La deuxième anecdote se passe aussi à IBM, au début des années 60. Il y avait à l'époque un ordinateur d'IBM qui s'appelait le 1401 ; c'était le premier mini, et aussi le premier qui ait été construit à des milliers d'exemplaires. Ce 1401 était donc un excellent ordinateur, et a eu des concurrents, comme il est normal. Un de ces concurrents était (déjà...) fabriqué par Honeywell, qui l'avait appellé... "Liberator". Ce Liberator marchait très bien, il avait notamment un avantage important sur la machine IBM, à savoir qu'il faisait des tris beaucoup plus rapides. Après avoir comparé les deux machines, les gens d'IBM se sont dit : ça, ce n'est pas ordinaire ; pourtant les disques ne vont pas plus vite, l'unité centrale ne va pas plus vite, les bandes ne vont pas plus vite, il doit y avoir un truc... Le truc, c'était le programme de tri, qui était très astucieux, et utilisait pour la première fois des outils mathématiques, et plus précisément les nombres de Fibonacci (nous verrons tout à l'heure comment les nombres de Fibonacci interviennent dans les algorithmes de tri). Les spécialistes d'IBM, après avoir étudié le problème, ont fait un rapport qui expliquait que, si Honeywell était meilleur, c'était parce qu'il utilisait les nombres de Fibonacci. L'histoire dit que ce rapport est remonté très haut dans la hiérarchie, et qu'il en est redescendu avec une note en marge : "Hire Fibonacci(1)..."

Plus sérieusement, je voudrais développer les deux points que j'ai affirmés plus haut, c'est-à-dire que l'industrie a besoin de mathématiques, et que l'industrie a besoin de mathématiciens.

En ce qui concerne le premier point, je ne parlerai pas directement des mathématiques appliquées (ou dites appliquées), parce que c'est alors une tautologie. Je voudrais au contraire parler des mathématiques dites pures, celles qui ne servent à rien, ou plutôt sont censées ne servir à rien : car ce que je veux montrer, c'est que toutes les mathématiques, même celles qui paraissent le plus ésotériques, dont on ne voit aucune utilisation a priori, peuvent trouver une application dans l'industrie

(1) Embauchez Fibonacci.

Tout d'abord, les mathématiques pures sont la base des mathématiques appliquées. Les exemples abondent, depuis les techniques de calcul de Heaviside, qui ont été utilisés par les physiciens de la première moitié du siècle sans aucune justification théorique avant que des mathématiciens comme Laurent Schwartz n'en trouvent non seulement les fondements théoriques, mais également, en conséquence, des prolongements et des possibilités d'utilisation plus puissantes. On peut citer aussi le calcul par éléments finis, qui a commencé par des bricolages d'informaticiens, avant que des mathématiciens comme Jean-Louis Lions n'en fassent la théorie et n'en permettent ainsi le développement de la pratique.

Je n'insisterai pas sur ce rôle "classique" des mathématiques pures, et je montrerai maintenant plusieurs exemples où elles interviennent directement dans des utilisations industrielles. Le premier de ces exemples, c'est celui des nombres de Fibonacci, que j'ai évoqués tout à l'heure. Que vient donc faire Fibonacci dans les tris sur ordinateur ? Lorsqu'on a un très grand nombre N d'"enregistrements" à trier (chacun de ces enregistrements pouvant être un mot d'un dictionnaire, la fiche d'un client, le relevé d'un compte bancaire, etc...), le problème est qu'ils ne peuvent pas rentrer tous ensemble dans la mémoire de l'ordinateur. On les regroupe donc en un petit nombre n d'enregistrements qui, eux, peuvent entrer en mémoire. On a ainsi p groupes de n enregistrements (avec pn = N). On va alors classer séparément, en mémoire, chacun de ces p groupes. On aura donc p groupes de n enregistrements ordonnés par ordre croissant ou décroissant dans chaque groupe. On va ensuite répartir ces p groupes sur deux disques, à savoir q groupes sur un disque et r groupes sur l'autre disque. Pour que le tri soit le plus efficace possible, il faut que p, q et r soient des nombres de Fibonacci.

Montrons-le sur un exemple simple en prenant p = 13. On répartit ces 13 groupes de n enregistrements en plaçant 8 sur un premier disque D1 et 5 sur un second disque D2. On va ensuite lire les 5 premiers groupes de D1 et les 5 groupes de D2, les interclasser deux à deux et écrire le résultat, c'est-à-dire 5 groupes de 2n enregistrements, sur un troisième disque D3. On aura alors 3 groupes de n enregistrements sur D1, 0 groupe sur D2 et 5 groupes de 2n enregistrements sur D3, chacun de ces groupes étant ordonné. On recommence en permutant D1, D2 et D3, c'est-à-dire qu'on va lire 3 groupes sur D1 et D3 respectivement, les interclasser et écrire les 3 enregistrements ordonnés de 3n enregistrements ainsi obtenus sur D2. Et ainsi de suite, jusqu'à ce qu'on arrive à un groupe unique, ordonné, de 13 enregistrements — ce qu'il fallait obtenir — en passant par les étapes intermédiaires schématisées ci-dessous :

Disque D1	Disque D2	Disque D3
8 gr de n	5 gr de n	0 gr
3 gr de n	0 gr	5 gr de 2n
0 gr	3 gr de 3n	2 gr de 2n
2 gr de 5n	1 gr de 3n	0 gr
1 gr de 5n	0 gr	1 gr de 8n
0 gr	1 gr de 13n	0 gr

Petit détail technique qui n'aura pas échappé au lecteur informaticien : en fin de compte, on arrive toujours à des groupes d'enregistrements trop nombreux pour tenir en mémoire... C'est exact, mais l'astuce de la méthode consiste à tenir compte du fait que ces groupes sont tous ordonnés, et qu'il est possible de les interclasser en les lisant par morceaux en séquence et en écrivant les enregistrements interclassés au fur et à mesure.

Cela étant dit, le lecteur même non informaticien aura reconnu la suite de Fibonacci 1 2 3 5 8 13 dans le nombre d'enregistrements de chaque groupe (croissant) et dans le nombre de groupes sur chaque disque (décroissant). La leçon la plus frappante de cet exemple, c'est que les nombres de Fibonacci ont attendu cette application plus de sept siècles...

Considérons un autre exemple, dont on pourra tirer une autre leçon. Cet exemple a trait à la théorie du codage. Lorsque l'on stocke ou transmet de l'information, il faut éviter les erreurs, si possible les détecter et, mieux, les corriger. L'idée pour ce faire est de coder les informations en ajoutant de la redondance. Plus précisément, en supposant que les informations traitées sont des mots de longueur l, on va adjoindre à chaque mot r caractères de redondance. On obtient donc un mot codé de longueur $l+r$. L'idée est de prendre un r aussi petit que possible, mais avec un algorithme extrêmement sophistiqué du passage du mot de longueur l au code longueur $l+r$, afin de détecter et de corriger le plus grand nombre d'erreurs possible. Cette théorie du codage a bien sûr pris un grand essor d'abord avec les télécommunications, puis avec l'informatique. La leçon à tirer ici, c'est que plus les outils mathématiques utilisés sont sophistiqués et font appel à des résultats fins de mathématiques pures, plus les codes sont performants. Mettons-le en évidence par trois cas.

Parmi les premiers codes réalisés sont ceux qu'on appelle les codes linéaires. Ces codes sont obtenus en considérant un mot de longueur l comme un élément d'un espace vectoriel de dimension l et en le codant simplement par une application linéaire de l'espace de dimen-

sion l dans l'espace de dimension $l+r$. Une application des codes linéaires est le code de Hamming, découvert dans les années 50 et utilisé depuis les années 60 dans beaucoup d'ordinateurs pour corriger une erreur et détecter parfois deux erreurs dans un mot.

Quelques années plus tard, on a fait appel à des outils mathématiques plus sophistiqués, comme la théorie de Galois, qui est à la base des codes de Reed. L'idée de Reed est de considérer le mot à coder comme un polynôme de degré $l-1$ sur un corps fini, le codage étant les $l+r$ valeurs que prend le polynôme sur les $l+r$ valeurs du corps, le décodage étant obtenu par la division euclidienne. C'est un code de Reed qui est utilisé sur les disques compacts pour corriger jusqu'à quatre erreurs : les codes de Reed sont donc nettement plus performants que les simples codes linéaires.

Les études les plus récentes pour améliorer les codes utilisent non plus la théorie de Galois, mais la géométrie algébrique. L'idée, due au mathématicien soviétique Goppa, est de se placer toujours sur un corps fini, d'y représenter un mot non plus par un polynôme, mais par une fraction rationnelle, et d'encoder le mot par les valeurs de sa fraction rationnelle associée pour certains éléments du corps qui sont les points rationnels d'une courbe algébrique sur le corps. Les codes de Goppa sont susceptibles de performances très supérieures aux codes de Reed, et a fortiori aux linéaires.

Mon expérience de l'industrie m'a convaincu que le phénomène illustré par l'amélioration des performances des codes avec la sophistication des outils mathématiques est loin d'être isolé : des problèmes industriels de plus en plus difficiles sont résolus en utilisant des outils de mathématiques pures de plus en plus puissants.

Je considérerai maintenant le problème inverse, si j'ose dire : je voudrais passer en revue rapidement quelques domaines des mathématiques les plus abstraites, et je me propose de montrer que pour tous ces domaines, il existe des applications, même pour ceux qui en paraissent le plus éloignés.

- Les foncteurs et catégories commencent à être utilisés pour modéliser les architectures d'ordinateurs.
- La topologie algébrique est utilisée dans le dessin des circuits VLSI (Very Large Scale Integration) sur une micro-plaquette de silicium afin d'obtenir des graphes planaires. Elle est également utilisée au niveau du packaging, pour minimiser les connections entre micro-plaquettes sur un module (ensemble de micro-plaquettes interconnectées sur un substrat de céramique).
- Les algèbres de Lie sont utilisées en physique corpusculaire.
- Les algèbres de Artin sont appliquées à l'optique.

- La géométrie algébrique sert en CAO (Conception Assistée par Ordinateur) de robotique. Le célèbre problème du déménageur, qui consiste à déterminer par le calcul si un objet de forme et dimension données va parvenir à passer dans un système de couloirs, de pièces et d'obstacles, est en effet fondamental en robotique : quand on définit le mouvement d'un robot pour souder une pièce ou mettre en place le tableau de bord d'une automobile à l'intérieur de la carrosserie, il faut s'assurer que le bras du robot ou la pièce qu'il transporte ne se cogne pas dans le reste. C'est un problème type de géométrie algébrique réelle.

Je pourrais prolonger l'énumération, mais je préfère l'écourter en terminant sur deux exemples qui portent sur des domaines très spécialisés des mathématiques.

Mon premier exemple porte sur l'Analyse Non Standard, qui vient d'être utilisée avec succès en Intelligence Artificielle. Le rapport entre Analyse Non Standard et Intelligence Artificielle n'est pas évident et mérite d'être expliqué succinctement. Un des problèmes rencontrés en IA est d'étendre les possibilités des systèmes experts aux logiques non aristotéliciennes, c'est-à-dire à des logiques plus puissantes que la simple logique vrai-faux : on a ainsi mis au point des logiques ternaires, probabilistes, etc... Récemment, un jeune chercheur français a défini une logique appelée qualitative, qui permet de raisonner sur des ordres de grandeur. Pour obtenir ce résultat, extrêmement important pour l'IA, il a d'abord défini un système formel, ensemble d'axiomes et de règles de déduction, puis il l'a validé, c'est-à-dire qu'il s'est assuré (dans la mesure du possible car c'est en général non décidable) que ce système était non contradictoire. C'est dans la phase de validation que l'Analyse Non Standard intervient, puisqu'elle consiste à établir un homomorphisme entre le système formel et l'Analyse Non Standard qui conserve l'implication : si le système formel était contradictoire, l'Analyse Non standard le serait aussi, et ça se saurait...

Mon deuxième exemple porte sur les systèmes dynamiques symboliques de Poincaré — domaine pointu et étroit, on en conviendra. Ces systèmes dynamiques symboliques ont été sortis de leur superbe isolement pour résoudre le problème du codage de l'information sur un disque magnétique. Ce problème vient du fait que, pour lire une information sur un disque magnétique, on va détecter un signal s'il y a un 1, et on ne détecte rien s'il y a un 0 ; lorsqu'il y a une trop longue succession de 0, le lecteur ne reçoit aucun signal et perd sa synchronisation. Il est nécessaire de coder l'information de manière à éviter toute suite de plus de n 0. C'est la théorie des systèmes dynamiques qui a permis la mise au point de codes possédant une telle propriété.

En conclusion de cette partie, je voudrais, comme nous sommes entre mathématiciens, vous proposer trois conjectures :

1. Il n'y a pas de domaine des mathématiques pures qui ne soit, un jour ou l'autre appliqué à un problème industriel.

2. Plus un problème industriel est difficile, plus sa solution exigera des outils de mathématiques pures sophistiqués.

3. Il n'y a pas de frontières entre mathématiques pures et mathématiques appliquées, mais un continuum entre mathématiques plus ou moins appliquées, ou encore, entre mathématiques déjà appliquées et d'autres qui ne le sont pas encore.

J'espère avoir convaincu le lecteur que l'industrie a besoin de mathématiques et j'en arrive à ma deuxième proposition : l'industrie a besoin de mathématiciens. Plus précisément, je voudrais montrer deux choses : que l'industrie a besoin des connaissances du mathématicien, mais aussi qu'elle a besoin de son état d'esprit, de sa démarche intellectuelle. Ici encore, je m'appuierai sur des exemples issus de mon expérience personnelle dans IBM, et je commencerai par raconter deux anecdotes, illustrant ce que peut apporter une formation de mathématicien dans un environnement industriel.

Dans le milieu des années 60, on commençait à essayer d'utiliser des ordinateurs pour faire des calculs algébriques, que l'on appelle calculs formels par opposition aux calculs numériques. De telles possibilités existent aujourd'hui sur certaines calculettes, mais posaient à l'époque beaucoup de problèmes du fait de la faible puissance des ordinateurs. Un de ces problèmes était de reconnaître l'identité de formules telles que x, sin(Arcsin(x)) et exp(Log(x)) : cette identité, triviale pour le mathématicien, est très compliquée à mettre en évidence pour un ordinateur. Tous les informaticiens que nous étions à l'époque essayions de mettre ces formules sous des formes canoniques plus ou moins polynomiales, mais qui étaient toutes fort lourdes à manipuler et ne garantissaient pas toujours l'exactitude du résultat. J'ai proposé alors une idée toute bête, qui partait de la constation que toutes les fonctions manipulées en calcul formel (du moins à ses débuts), étaient de braves fonctions, bien analytiques : des polynômes, des fractions rationnelles, des sinus, des cosinus, des exponentielles, les logarithmes... bref, des fonctions dont l'ensemble des zéros est de mesure nulle. Si alors, en donnant au hasard une valeur à la variable, ou aux variables, on tombe sur zéro, c'est bien le diable si ce zéro-là est précisément un des zéros de l'ensemble de mesure nulle et on peut conclure, sans grande chance de se tromper, que la fonction est identiquement nulle. Comme le calcul d'une fonction pour une valeur numérique de la variable se fait très rapidement et simplement sur un ordinateur, on peut même, pour plus de sûreté, faire une petite boucle sur une dizaine de valeurs différentes. La méthode peut paraître expéditive, mais elle a été beaucoup utilisée. Bien entendu, le lecteur informaticien aura tout de suite vu sa faiblesse congénitable : la précision finie des calculs sur machine ne permet pas toujours de

déterminer si un résultat est nul ou seulement très petit. C'est pourquoi la méthode a été considérablement améliorée par un mathématicien américain du nom d'Arthur Martin (ça ne s'invente pas...), qui a eu l'idée de faire les calculs dans un corps fini, pour lequel la mesure du zéro, si elle n'est pas nulle, est parfaitement définie, ce qui permet d'évaluer avec précision le risque pris. Il est évident que seules des connaissances en mathématiques pouvaient conduire à une telle approche pour résoudre ce problème quasiment insoluble avec des seules connaissances informatiques.

Ma seconde anecdote porte sur la résolution d'un vrai problème industriel : celui de la minimisation du nombre de locomotives sur un réseau SNCF. La SNCF a une série de trains à tirer, et pour chaque train, il faut une locomotive. Une fois qu'un train est arrivé à son terminus, on décroche la locomotive, et on l'envoie tirer un autre train. Comment peut-on affecter à chaque train une locomotive pour tirer tous les trains avec le nombre minimum de locomotives ? Ce problème d'affectation est bien connu des spécialistes de recherche opérationnelle ; la méthode classique est de considérer le diagramme espace-temps (niveau Cours Moyen) comme un graphe sur lequel il faut faire passer un flot de locomotives qui soit au minimum de 0 dans les gares et de 1 le long des trains : minimiser le nombre de locomotives revient à minimiser le flot total qui traverse le graphe, ce que l'on sait faire grâce à un algorithme dû à Ford et Fulkerson. J'ai cherché à améliorer la méthode, en évitant de laisser une locomotive trop longtemps en attente dans une gare, mais au contraire en l'envoyant haut-le-pied, c'est-à-dire à vide, chercher un train dans une autre gare. Le problème, c'est qu'il faut générer tous les mouvements haut-le-pied possibles, et qu'il y en a une infinité, ou au moins un très grand nombre, beaucoup trop grand pour qu'on puisse faire tourner Ford-Fulkerson. Comme Zorro, mes connaissances de mathématiques sont arrivées : j'ai défini une relation d'équivalence entre les différents mouvements haut-le-pied, montré que le flot sur le graphe était conservatif pour cette relation d'équivalence, appliqué Ford-Fulkerson au graphe quotient. Dès la première application de la méthode, on a pu diminuer le nombre de BB 16000 sur le réseau nord de 23 à 17 — ce qui représente une économie très substantielle... L'histoire ne s'arrête pas là, car j'ai dû pour résoudre ce même problème faire appel à d'autres connaissances mathématiques, tout aussi inattendues. Les horaires de la SNCF ont une périodicité apparente de 24 heures, mais la périodicité réelle est de 7 jours (en fait, c'est encore plus compliqué à cause des fêtes mobiles et des trains qui circulent sauf dimanche et fêtes, lundi et lendemain de fêtes...). Fort heureusement pour moi, le nombre de jours dans la semaine est premier, car il y avait un moment crucial dans ma méthode où je faisais intervenir le théorème de Bezout. Je serais bien en peine aujourd'hui de me rappeler ce que venait faire Bezout dans cette galère, toujours est-il que lorsque ma

méthode donnait un nombre de locomotives minimum multiple de 7, il fallait en rajouter une pour pouvoir appliquer l'horaire. J'avais eu du mal à expliquer cela aux responsables de la SNCF.

Ces deux exemples montrent que les connaissances mathématiques, et pas seulement celles acquises dans le supérieur, mais aussi dans le secondaire (comme le montre mon anecdote ferroviaire) peuvent se révéler utiles, voire indispensables, dans l'exercice d'une profession industrielle. N'importe quel mathématicien dans l'industrie peut en témoigner. Mais ce ne sont pas seulement ces connaissances qui servent au mathématicien dans l'industrie, c'est aussi sa forme de pensée, sa démarche intellectuelle, toute la méthodologie de travail qu'il a acquise en faisant des mathématiques.

On peut faire en effet un parallèle entre l'activité du mathématicien et celle de l'ingénieur, qui m'amène à la conclusion qu'un mathématicien a les qualités demandées à un ingénieur. Qu'est-ce qu'un mathématicien ? Un mathématicien, c'est quelqu'un qui sait :

1. poser des problèmes,
2. trouver des solutions,
3. s'assurer que les solutions marchent dans tous les cas.

Reprenons chacun de ces points :

1. Poser les problèmes, cela veut dire reconnaître quelles sont les hypothèses importantes et celles qui sont inutiles ; savoir d'où on part et où on veut aller — c'est le travail quotidien de l'ingénieur et du mathématicien.

2. Trouver des solutions, c'est aussi le travail quotidien de l'ingénieur et du mathématicien. Mais le mathématicien a une grande qualité dans la recherche de solutions : il sait trouver des solutions qui ne sont pas dans l'énoncé du problème. Je m'explique : nombreux sont les exemples où la solution à un problème posé dans un certain domaine des mathématiques est tirée d'un autre domaine — c'est Descartes faisant appel à l'algèbre pour résoudre des problèmes de géométrie, Riemann à l'analyse pour des problèmes d'arithmétique... De plus en plus souvent les théorèmes les plus puissants sont obtenus par l'application à une certaine branche des mathématiques de méthodes développées et de résultats obtenus dans une autre branche. Et dans l'industrie, nombreux sont les problèmes qui n'ont pas le bon goût de se laisser résoudre en faisant appel à une seule discipline ou une seule technologie : l'ingénieur électronicien sur la chaîne de fabrication de microplaquettes devra être capable de déterminer si une baisse de rendement soudaine n'est pas due à une pollution chimique ou à un problème d'optique.

3. Enfin, le mathématicien sait s'assurer que les solutions marchent partout, que tous les lemmes techniques ont été démontrés, tous les cas particuliers prévus. Mais c'est aussi ce que l'on va demander à l'ingé-

nieur : si le procédé qu'il a mis au point ne marche que dans certains cas ou si la machine qu'il a réparée retombe en panne dès qu'il a le dos tourné, ce n'est pas un bon ingénieur.

L'industrie n'a pas seulement besoin des connaissances qu'a acquises le mathématicien, mais aussi des habitudes de travail — méthodes, rigueur — qu'on lui a inculquées.

Ce qui m'amène à ma conclusion, dans laquelle je voudrais m'adresser aux professeurs de mathématiques. Si l'industrie a besoin de mathématiques et de mathématiciens, est-ce que cela a des implications sur l'enseignement des mathématiques ? Je serai bien sûr très prudent, car enseigner les mathématiques est un métier difficile — et ce n'est pas mon métier. Mais je profiterai de la tribune qui m'est offerte ici par l'A.P.M.E.P. pour dire aux professeurs qui me liront : "les mathématiques, ce n'est pas seulement un outil de sélection pour les jeunes que vous formez, mais ce sera aussi — et surtout — un outil de travail pour les adultes qu'ils deviendront. Et pas seulement pour l'infime minorité d'entre eux qui feront des mathématiques leur profession, mais pour tout le monde". Et je me risquerai à tirer deux conséquences.

La première, c'est que, si les mathématiques sont un outil, il faut apprendre à s'en servir. Et avant même d'apprendre à s'en servir, il faut apprendre que ça sert. Et c'est là, à mon avis, un point sur lequel l'enseignement des mathématiques en France peut être amélioré : je suis choqué quand je vois ma fille, qui est en prépa, apprendre deux sortes de mathématiques, les mathématiques pour les mathématiques et les mathématiques pour la physique. Les sacro-saints programmes ont une grande part de responsabilité, qui ne facilitent pas la tâche à l'enseignant qui voudrait montrer comment et à quoi les mathématiques peuvent servir. C'est à mes yeux une raison supplémentaire pour que les professeurs de mathématiques redoublent d'efforts pour montrer aussi les applications des théories qu'ils et elles enseignent.

La seconde conséquence que je tirerai, c'est qu'un outil, pour bien s'en servir, il ne faut pas en avoir peur. Encore une fois, j'illustrerai mon propos à l'aide de l'informatique. En tant qu'informaticien, je suis aussi choqué de la manière dont on enseigne l'informatique, plus précisément dont les programmes d'enseignement forcent à enseigner l'informatique : on enferme les élèves dans un carcan de règles, on les terrorise avec un formalisme digne de la scolastique... Alors que l'informatique, avec les ordinateurs et les logiciels modernes, c'est un outil merveilleux, qu'il ne faut surtout pas craindre d'utiliser, avec lequel on peut expérimenter, "voir ce que ça donne" — oserais-je dire même : "essayer d'abord, réfléchir ensuite" ? Je n'irai certes pas aussi loin en ce qui concerne les mathématiques, mais le dernier message que j'adresserai aux enseignants qui me lisent, c'est de leur demander d'inculquer à leurs élèves, non pas la crainte des mathématiques, mais leur respect.