Serge Rochain

Le problème des deux échelles qui se croisent dans un couloir

Parmi les problèmes amusants et délectables, celui des deux échelles dans un couloir tient une place de choix, car sous une apparence de grande simplicité il s'y cache une énigme redoutable dont la recherche de solution en a dérouté plus d'un. En effet, la solution implique une équation du quatrième degré, ce qui n'est pas un problème spécialement simple, mais il nous donne l'occasion de mettre en évidence les vertus de l'informatique.

Le problème :
Deux échelles sont appuyées chacune sur les parois opposées d'un couloir. L'une d'elle fait trois mètres de haut et l'autre deux. Elles se croisent un mètre au-dessus du sol.
La question est : **Qu'elle est la largeur du couloir ?**

L'informatique permet de trouver la solution du problème d'une façon plus rapide que celle qui consiste à résoudre manuellement l'équation du quatrième degré à laquelle un raisonnement d'algébriste va nous conduire. Les solutions informatiques sont multiples et différentes, et même très différentes, mettant à contribution à la fois la capacité de raisonnement de l'homme et la performance au calcul de l'ordinateur, et ceci dans des proportions très variées selon la voie choisie pour trouver la solution, en fonction notamment de la puissance de l'ordinateur disponible. Je vous propose ci-dessous deux solutions qui mettent chacune en valeur des qualités très différentes de l'informatique tant dans ses concepts de structures d'ordinateurs que dans ceux faisant les spécificités des langages de programmation qui servent à les mettre en œuvre.

La première solution ne fait que faiblement appelle à la puissance de l'ordinateur puisqu'il est très peu sollicité. En revanche, il faudra que le programme qu'on lui demandera d'exécuter lui « mâche » le travail, c'est-à-dire qu'il faudra lui donner une suite d'instructions à exécuter qui consiste à lui dire « comment faire pour aboutir à **la** solution ». Ce n'est donc pas lui qui trouve la solution mais celui qui lui dit comment faire, l'auteur du programme. A l'occasion de l'étude de cette solution j'en profiterai pour vous faire découvrir l'algorithmique, puisque la solution repose sur l'exploitation de cette technique de logique mathématique. Avec cette méthode, la solution aurait été obtenue en moins de 2 secondes sur un PC des années 70, et quelques centièmes de secondes sur une machine d'aujourd'hui.

La deuxième solution que j'appelle solution FMB (Fait par Machine Bolide), mettra moins à l'épreuve votre capacité d'analyse de méthode mais sollicitera fortement la performance de l'ordinateur. Pour résumer, elle consiste à dire dans un langage très concis à l'ordinateur de calculer toutes les solutions possibles et de ne donner le résultat que pour la solution où la largeur du couloir est cohérente avec les conditions imposées (croisement à un mètre au-dessus du sol d'une échelle de 2m avec une autre de 3m). Il n'y a là aucune subtilité dans le programme soumis à l'ordinateur, juste une demande d'égalité à l'équation qui lui est soumise. L'ordinateur calculera les milliers de solutions possibles dans la fourchette de largeur de couloir que l'auteur estime possible, et dont une seule correspond à l'égalité entre ces 4 variables (1m ; 2m ; 3m ; et largeur du couloir). Cette solution auraient été inacceptable pour des PC d'il y a quarante ans car il aurait fallu plus de 100 heures de calcul à un PC de cette époque pour trouver la solution alors que le résultat est atteint en moins de deux secondes sur une machine d'aujourd'hui.

Le schéma de la situation est visible sur la figure 1A ci-dessous à gauche. Intuitivement on voit que la démarche à engager doit nous conduire à une des deux figures 1B ou 1C à partir

desquelles la résolution du théorème de Pythagore doit nous fournir la réponse attendue. Connaissant l'hypoténuse du triangle de la figure 1B ou celui de la figure 1C qui ne sont autre que les longueurs des échelles il « suffit » de connaître la hauteur depuis le sol du point des parois sur lesquels reposent les échelles pour en conclure la largeur du couloir.

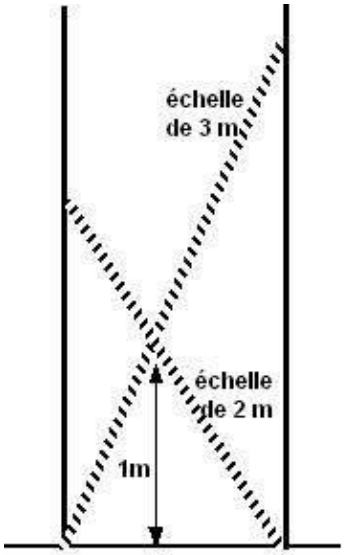


Figure 1A

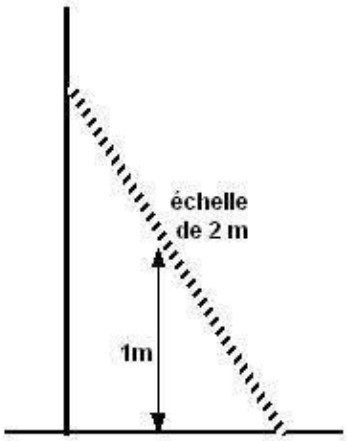


Figure 1B

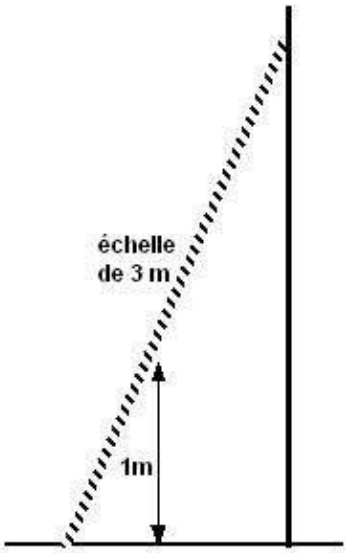


Figure 1 C

De ces hauteurs, nous savons qu'elles sont situées à plus d'un mètre et moins de deux pour celle de la figure 1B et à plus d'un mètre et moins de trois pour celle de la figure 1C. Nous connaissons les deux hypoténuses mais surtout nous savons qu'une variable est commune aux deux triangles, la base qui n'est autre que la largeur du couloire, la réponse à la question. Cette variable commune est elle-même la somme de deux variables **a** et **b** (voir figure 2A ci-dessous) dont les proportions relatives sont fixées par le point de croisement des deux échelles, donc de leur longueur, et par suite des deux hauteurs dont on se propose de déterminer la valeur. Le rapport de ces deux variables **a** et **b** est donc la clef du problème de la détermination de c et d auxquels l'ajout d'un mètre fourni le côté du triangle nous permettant de calculer sa base.

Isolons chaque segment identifiable sur la figure 2A et proposons nous de déterminer la valeur relative de d par rapport à a, b, et c et la valeur relative de c par rapport à a, b, et d.

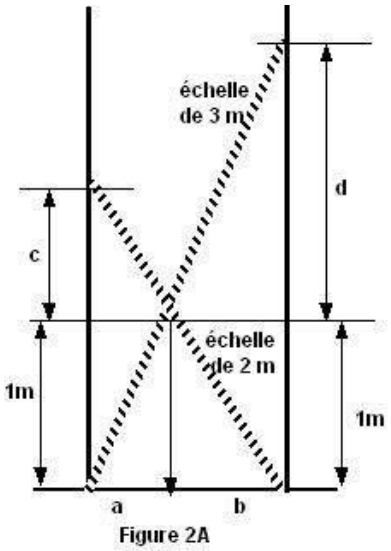


Figure 2A

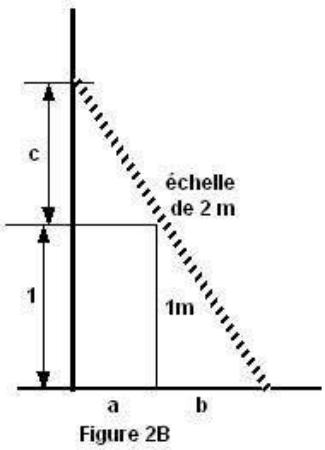


Figure 2B

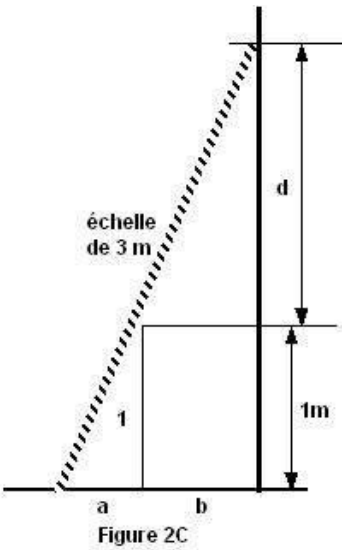


Figure 2C

Première étape (Figure 2C) : Détermination de d par rapport à a et b.
La surface du grand triangle, dont l'échelle de 3 m est l'hypoténuse, est égale à la largeur du couloir (a+b) multiplié par la hauteur (1+d) et divisé par 2.
Mais c'est aussi la somme de la surface des deux petits triangles et du rectangle. Le sommet en haut à gauche du triangle est le point de croisement des deux échelles.

Nous avons donc l'équation suivante :

$(a+b)(1+d)/2 = bd/2 + a/2 + b$ en effectuant les produits nous avons :
$(a+ad+b+bd)/2 = (bd + a + 2b)/2$ et en multipliant les deux membres par deux :
$a+ad+b+bd = bd + a + 2b$ en supprimant les termes identiques dans les deux membres :
$ad = b$

En développant le même raisonnement sur la figure 2B qui est symétrique à 2C nous arriverons à l'équation $bc = a$.

Ces deux derniers résultats peuvent aussi s'écrire :
$a/b = c$ et $b/a = d$ ce qui conduit à l'identité fondamentale **$d = 1/c$**
qui nous permettra de restreindre le problème à une seule inconnue : ***c***

Mais nous avions de la même façon $c = 1/d$ qui nous permettrait de restreindre le problème à la seule inconnue ***d*** ce qui semble équivalent à priori puisque par rapport au point de croisement des échelles la situation est symétrique et équivalente. Nous pouvons donc choisir entre les deux solutions mais je choisi l'inconnue c pour une raison qui apparaîtra évidente plus loin.

La largeur du couloir étant la partie commune aux deux échelles nous avons la relation :

$9-(d+1)^2 = 4-(c+1)^2$ et en remplaçant **d par 1/c** nous n'avons plus qu'une seule inconnue : c
$9-(1/c+1)^2 = 4-(c+1)^2$ en effectuant les produits :
$9- 1/c^2 -1- 2/c = 4 – c^2 -1- 2c$ en multipliant par c^2
$9c^2 -1- c^2- 2c = 4c^2 - c^4- c^2 - 2c^3$ en regroupant par ordre décroissant de puissance :
$c^4 + 2c^3 + 5c^2 - 2c -1 = 0$ On reconnaît l'équation générale du quatrième degré.
En résolvant cette équation on trouve $c = 0{,}5761207$

La légitime question est : Comment trouve t-on ce résultat ? Voir plus bas, mais continuons pour répondre à la question d'origine : **Qu'elle est la largeur du couloir ?**

Puisque nous avons déterminé la longueur du segment c, il suffite de l'augmenter de 1 mètre pour connaître la hauteur du triangle rectangle dont l'hypoténuse est représenté par l'échelle de 2m.

Le théorème de Pythagore vient aisément à bout du problème (pour autant que vous lui fassiez confiance… Êtes-vous certain qu'il est exact ?) :

Largeur du couloir = $(4 – 1{,}5761207^2)^{1/2} = 1{,}231$ mètres.

Revenons sur la solution de l'équation du quatrième degré qui est la véritable difficulté de ce problème.

C'est le mathématicien italien Ludovico Ferrari (1522-1565) qui a résolu l'équation du quatrième degré. Elève puis collaborateur de Cardan, qui avait résolu l'équation du troisième degré, Ferrari ramène celle du quatrième degré à l'équation du troisième degré par un artifice remarquable. Je ne vais pas vous détailler la méthode que vous pouvez trouver dans de nombreux ouvrages et sur Internet, mais pour résoudre une équation du 4^{e} degré, je vais vous donnez une méthode moderne qui exploite les moyens d'aujourd'hui, c'est-à-dire la puissance de l'ordinateur, bien que pour l'exemple dont il s'agit qui est très simple une calculette suffise.
C'est d'ailleurs ce que j'ai fait pour atteindre les trois premiers chiffres après la virgule (le mm) avant d'avoir l'idée de rédiger un petit programme pour vous montrez la puissance des méthodes algorithmiques.

Le principe de l'algorithme consiste à attribuer une valeur arbitraire à l'inconnue dont on cherche à déterminer la valeur réelle satisfaisant à l'équation dont elle fait partie, puis à la suite d'un premier calcul basé sur cette hypothétique valeur, constater l'inadéquation résultante. Cette inadéquation se traduit par un déséquilibre de l'équation (inéquation) dont l'amplitude est proportionnée à l'écart entre l'inconnue satisfaisant à l'équation et la valeur arbitraire qui lui a été donnée. Le sens de l'écart permet de savoir si l'estimation était trop faible ou trop importante et la valeur de l'écart permet de juger de l'amplitude de la modification à appliquer sur la valeur estimée précédente. En pratique il suffit souvent de calculer la moyenne entre la valeur choisie arbitrairement et celle obtenue par son application dans le calcul pour obtenir une valeur plus proche de la réalité que la précédente.
En recommençant avec une valeur corrigée de l'amplitude estimée par le calcul précédent, on obtient un déséquilibre plus faible que dans le calcul précédent, et il suffit de recommencer autant de fois que nécessaire jusqu'à ce que l'équation soit satisfaite.

Dans l'équation ci-dessus, un moyen simpliste consiste à commencer le premier calcul avec une valeur suffisamment basse pour nous permettre d'atteindre à coup sur le résultat souhaité avec jusqu'à 5 chiffres après la virgule, par exemple 0,000001 puis recommencer en l'incrémentant de cette valeur. Mais l'on voit, puisque nous connaissons le résultat (0,5761207), qu'il nous faudra exécuter près de 600 000 fois le calcul avant d'obtenir une équation équilibrée. Même si les ordinateurs sont rapides, ce n'est pas la peine de gaspiller leur temps, d'autant plus que pour certains problèmes relevant de l'algorithmique cette solution ne pourrait pas être retenue tant l'amplitude du domaine possible de la valeur réelle de la variable est importante. Il va de soit que l'algorithmique à d'autres tours dans son sac.

Pour ne pas vous troubler avec le problème de l'équation du quatrième degré je vous invite d'abord à faire connaissance avec les outils de cette branche des mathématiques sur un problème plus simple qui met en œuvre certains de ses principes. Nous reviendrons ensuite sur son application dans le cadre du problème des échelles.

Une méthode permettant d'extraire une racine carrée, va bien au-delà de ce que l'on peut croire en étendant à cette « curiosité » les supposées vertus que les « anciens » attribuaient à la connaissance des tables de multiplication, elles aussi dépassées par les calculettes les plus rudimentaires. Je vous accorde que la maîtrise des tables de multiplication n'est qu'une démonstration du « par cœur », tandis que celles permettant de savoir extraire une racine carrée (du moins pour certaines méthodes) ouvrent l'esprit sur la logique mathématique et notamment sur l'algorithmique, branche ancienne et oubliée durant deux siècles jusqu'à ce que l'informatique la sorte de sa torpeur en la découvrant comme étant la clef du calcul numérique appliqué aux processus automatiques.

Imaginez le problème de l'extraction de racine de la façon suivante :

A) Conditions du problème.
1) Je dispose d'une figure carrée dont je connais la surface et pour laquelle je souhaite connaître le côté.
2) Je peux dire n'importe quel côté puisqu'ils sont identiques.

B) Quelques considérations indépendantes du problème (pour l'instant).
1) Si je divise la surface d'un rectangle par son petit côté, j'obtiens le grand côté.
2) Si je divise la surface de ce même rectangle par son grand côté, j'obtiens le petit.

C) Intégrations de ces considérations dans le problème de l'extraction de racine carrée.
1) Si je divise la surface de mon carré par un nombre quelconque et que j'obtiens un quotient, par pur hasard, égal à ce diviseur arbitraire, du même coup j'ai la racine recherchée.
2) Si je divise la surface de mon carré par un nombre quelconque trop grand pour être la racine, j'obtiens un quotient qui sera d'autant plus petit par rapport à ce que doit être la racine que le diviseur choisi arbitrairement était trop grand.
3) Réciproquement, si je divise la surface de mon carré par un nombre quelconque trop petit pour être la racine, j'obtiens un quotient qui sera d'autant plus grand par rapport à ce que doit être la racine que le diviseur choisi arbitrairement était trop petit.
4) Ainsi, dans les cas 2) et 3) si je fais une moyenne entre le diviseur arbitraire et le quotient qui en résulte, j'obtiens un résultat plus proche de la racine que ne l'était le diviseur arbitraire.
5) Il ne reste qu'à recommencer l'opération avec ce « nouveau » diviseur beaucoup moins arbitraire que le précédent.
6) Il suffit de répéter ce processus itératif en prenant à chaque fois pour nouveau diviseur la moyenne du quotient obtenu à l'opération précédente et du diviseur ayant permis de l'obtenir jusqu'à ce que le nouveau quotient soit égal au diviseur qui a permis de l'obtenir sur un nombre de décimales correspondant à la précision recherchée.

C'est exactement ce qui se passe quand vous appuyez sur la touche racine de la calculette après avoir introduit un carré C. Cet algorithme qui met à profit les possibilités de calcul rapide des processeurs s'exprime de la façon suivante :

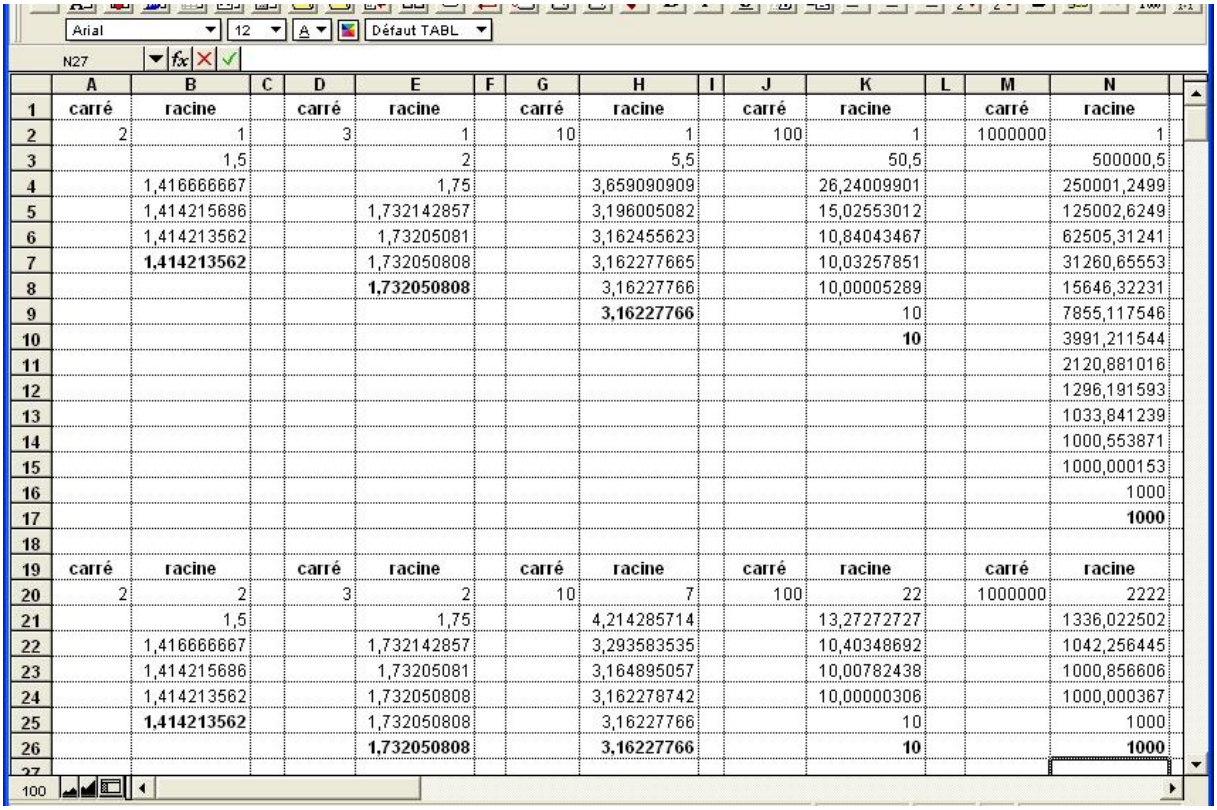
$x_1 = ½ * (C/x_0 + x_0)$ qui représente la première des itérations successives.

La suivante étant : $x_2 = ½ * (C/x_1 + x_1)$ etc,

jusqu'à $x_{n+1} = ½ * (C/x_n + x_n)$ tel que $x_{n+1} = x_n$

Un tableur permet de visualiser facilement cette notion de convergence, comme dans l'exemple ci-dessous (figure 3). 5 exemples différents (2, 3, 10, 100, et 1000000) sont donnés dans la largeur de la feuille de calcul. 2 colonnes sont utilisées pour chacun d'eux. La première indique le carré dont on cherche à connaître la racine et la seconde colonne montre l'évolution de la variable x depuis $x_0 = 1$ jusqu'à x = n lorsque deux valeurs consécutives sont égales, ce qui arrête le calcul. 1 est choisi arbitrairement comme valeur de x_0 et donc d'autant plus éloigné de la valeur réelle de la racine que celle ci est grande, ce qui se traduit par un nombre d'itérations plus élevé pour atteindre le résultat exact. Ce qui ne fait que 5 itérations pour racine de 2 en fait 15 pour 1 million. Il n'est donc pas judicieux de partir

systématiquement de la même valeur quel que soit le carré dont on cherche la racine. Dans la partie basse du tableau les mêmes 5 exemples sont repris avec des choix différents et plus judicieux pour la variable d'initialisation x_0. L'objectif est d'obtenir une valeur aussi proche que possible de la racine exacte par un moyen déterministe.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	carré	racine		carré	racine		carré	racine		carré	racine		carré	racine
2	2	1		3	1		10	1		100	1		1000000	1
3		1,5			2			5,5			50,5			500000,5
4		1,416666667			1,75			3,659090909			26,24009901			250001,2499
5		1,414215686			1,732142857			3,196005082			15,02553012			125002,6249
6		1,414213562			1,73205081			3,162455623			10,84043467			62505,31241
7		**1,414213562**			1,732050808			3,162277665			10,03257851			31260,65553
8					**1,732050808**			3,16227766			10,00005289			15646,32231
9								**3,16227766**			10			7855,117546
10											**10**			3991,211544
11														2120,881016
12														1296,191593
13														1033,841239
14														1000,553871
15														1000,000153
16														1000
17														**1000**
18														
19	carré	racine		carré	racine		carré	racine		carré	racine		carré	racine
20	2	2		3	2		10	7		100	22		1000000	2222
21		1,5			1,75			4,214285714			13,27272727			1336,022502
22		1,416666667			1,732142857			3,293583535			10,40348692			1042,256445
23		1,414215686			1,73205081			3,164895057			10,00782438			1000,856606
24		1,414213562			1,732050808			3,162278742			10,00000306			1000,000367
25		**1,414213562**			1,732050808			3,16227766			10			1000
26					**1,732050808**			**3,16227766**			**10**			**1000**

Figure 3

La seule valeur connue au départ est le carré lui-même, c'est donc en fonction de ses caractéristiques qu'il faut choisir la variable initialisant x_0.
L'arsenal mathématique possède une branche appelée théorie des nombres. Celle ci considère les mathématiques comme un objet d'étude pour lui-même. On lui doit de nombreux acquis comme les logarithmes, par exemple, ou encore des méthodes visant à catégoriser les nombres comme la démonstration de la transcendance de π ou l'irrationalité de racine de 2
Ici, il s'agit de trouver une valeur de x_0 qui ne soit pas trop éloignée de la racine réelle afin de minimiser le nombre d'algorithmes nécessaire à l'obtention du résultat.

Nous savons que le carré d'un nombre qui s'écrit avec n chiffres conduira au maximum à un résultat qui s'écrira avec 2n chiffres s'il est grand et avec 2n-1 chiffres s'il est petit. Ceci nous catégorise déjà la racine qui sera constituée avec des chiffres de poids fort entre 1 et 5 ou entre 5 et 9, nous choisirons 2 pour le premier cas et 7 pour le second. Par ailleurs la valeur donnée à x0 aura un nombre de chiffre égal à la moitié de celui du carré ou la moitié moins 1. Ainsi, pour un carré ne comprenant qu'un seul chiffre, x0 = 2
S'il en contient 2 $x_0 = 7$, s'il en contient 3, $x_0 = 22$, s'il en contient 4, $x_0 = 77$, s'il en contient 5, $x_0 = 222$, etc.

En initialisant x_0 avec cette méthode avant de lancer l'algorithme convergent qui permettra d'atteindre la valeur exacte de la racine, on restreint le nombre d'algorithme à un nombre voisin quelle que soit l'amplitude du carré. C'est ce que l'on peut constater sur les 5 exemples

donnés en partie basse du tableur Figure 3 ci-dessus. On peut voir que le bénéfice de cette initialisation intelligente de la valeur d'origine donnée à la variable x_0 est d'autant plus important en économie d'itérations que le carré dont on cherche à connaître la racine est grand, comme le soulignent les exemples à la droite du tableau en partie basse (Figure 3).

Les méthodes algorithmiques disposent de différentes propriétés comme la rapidité de convergence par exemple. Lorsque la précision gagnée à chaque itération double par rapport à la précédente on dit quelle est d'ordre deux, c'est le cas de l'algorithme d'extraction de racine ci-dessus. Ces méthodes permettent toutes de converger vers le résultat mais pas toujours de la même façon. Dans certains cas, dits alternés, les résultats intermédiaires convergent vers la cible en l'encadrant de plus en plus prés alternativement par défaut et par excès, alors que d'autres méthodes peuvent tendre vers le résultat par valeurs toujours inférieure ou toujours supérieure selon la valeur initiale imposée à la variable. Ce sont les caractéristiques du problème à résoudre qui conditionnent les propriétés de l'algorithme à imaginer.

Par exemple, dans notre problème d'échelles qui se croisent dans un couloir nous pouvons essayer d'imaginer un algorithme permettant de déterminer la valeur ***c*** à tous les niveaux (toutes les équations) dans lesquelles ***c*** apparaît comme la seule inconnue. C'est-à-dire dans la liste suivante reprise du raisonnement développé plus haut :

$9-(1/c+1)^2 = 4-(c+1)^2$
$9- 1/c^2 -1- 2/c = 4 – c^2 -1- 2c$
$9c^2 -1- c^2- 2c = 4c^2 - c^4- c^2 - 2c^3$
$c^4 + 2c^3 + 5c^2 - 2c -1 = 0$

Nous ne sommes pas contraints de rechercher l'algorithme uniquement au niveau de la forme normalisée de l'équation du quatrième degré, c'est-à-dire la dernière équation. Ce ne serait d'ailleurs pas un choix très judicieux car elle se présente sous la forme de termes dont les premiers sont à des puissances élevées (4 et 3) ce qui amplifierait considérablement les écarts entre les valeurs approchées donnée à ***c*** et sa valeur réelle, ce qui entraînerait une faible vitesse de convergence. Par ailleurs le résultat de l'équation conséquent à la variable convergente ***c*** serait toujours comparé à la valeur nulle (0) du second membre sans que ce second membre prenne une part active à la détermination de ***c***.

En revanche la première équation $9-(1/c+1)^2 = 4-(c+1)^2$ présente beaucoup d'intérêts.
En effet, les deux membres utilisent ***c*** à la même puissance (2, c'est-à-dire au carré) et les deux membres sont du même ordre de grandeur ce qui assure une certaine régularité dans la convergence. C'est donc certainement le bon choix.

Plus haut j'ai dit qu'après une petite hésitation j'ai choisi de prendre ***c*** comme inconnue au lieu de ***d*** avec de bonnes raisons qui apparaîtront évidente plus loin.
C'est le moment de comprendre la raison de ce choix.

Puisque l'échelle coté de la verticale 1+c a une longueur de 2 mètres, nous savons que ***c*** est supérieur à 0 et inférieur à 1, alors que l'échelle coté de la verticale 1+d a une longueur de 3 mètres, ce qui implique pour ***d*** une valeur supérieure à 0 et inférieure à 2. L'incertitude sur la valeur de ***c*** est donc moitié moindre que pour celle de ***d***, d'où la raison de choisir ***c*** pour unique inconnue. L'inconnue dont il faut cerner la valeur se situe dans une fourchette deux fois plus petite avec ***c*** plutôt qu'avec ***d***.

Le principe algorithmique doit mettre en œuvre trois caractéristiques fondamentales :

- Le principe qui permet à l'inconnue de tendre vers la solution.
- Le moyen de savoir que la solution est atteinte.
- Le moyen de faire évoluer la variable représentant l'inconnue.

Pour bien illustrer la méthode algorithmique appliquée au problème des deux échelles croisées j'ai imaginé deux procédés partiellement différents que je détails successivement :

Premier procédé :
Ici nous allons faire d'une pierre, deux coups en répondant aux deux premières exigences avec un moyen commun. Nous allons utiliser l'équation elle-même pour satisfaire ces deux critères.

Pour les deux méthodes nous savons que $0,1<c<0,9$ et considérons globalement l'équation $\alpha = \beta$ symbolisant $9-(1/c+1)^2 = 4-(c+1)^2$

Définissons le domaine des variables α et β pour les valeurs extrêmes approximatives de c
Pour c = 0,1 nous obtenons α = -112 et β = +2,79
Pour c = 0,9 nous obtenons α = +4,54 et β = +0,39

Pour des valeurs croissantes de c on voit que α est croissant tandis que β décroît. Mais quelques part entre ces extrêmes ils sont égaux par définition de ce qu'est une équation et ce cas particulier ce réalise lorsque la valeur attribuée à c est bien racine de l'équation.
Lorsque α sera égal à β à travers l'évolution de c sur un nombre de décimales fixé à l'avance (précision demandée) la recherche de la solution sera terminée et cette dernière sera égale à la valeur courante de c à ce moment précis.

La troisième exigence consistant déterminer le moyen permettant de faire évoluer la variable c repose sur la moyenne de deux variables dont la première est issue des deux extrêmes 0,1 et 0,9, soit 0,5 qui nous donne un résultat de calcul des premiers α et β. L'évolution suivante de c dépend du résultat de ce calcul.

Si $\alpha < \beta$, alors c était estimé trop petit et sa nouvelle valeur sera la moyenne entre cette trop petite valeur et la dernière plus grande valeur connue de c, en l'occurrence 0,9 pour le début de l'algorithme. Ceci donnera pour c une nouvelle valeur approximative de (0,5+0,9)/2 = 0,7.
Si $\alpha > \beta$, alors c était estimé trop grand et sa nouvelle valeur sera la moyenne entre cette trop grande valeur et la dernière plus petite valeur connue de c, en l'occurrence 0,1 pour le début de l'algorithme. Ceci donnerait une nouvelle valeur approximative de (0,5+0,1)/2 = 0,3, mais en l'occurrence ce ne peut pas être le cas à l'initialisation car α = -112 et β = +2,79 comme vu plus haut. En revanche cette éventualité se produira durant le déroulement de l'algorithme avant que α ne soit égal à β.

Nous avons satisfait au trois exigence d'un algorithme devant converger vers la solution et sa logique se récapitule de la façon suivante :
Les abréviations suivantes sont utilisées dans les propositions logiques ci-dessous :
DPG pour dernière plus grande valeur de c
DPP pour dernière plus petite valeur de c
EC pour valeur en cours de c

α pour $9-(1/c+1)^2$
β pour $4-(c+1)^2$

0,1 → DPP
0,9 → DPG
Boucle itérative.
Calcul de la moyenne entre DPP et DPG → EC
Calcul $\alpha \int$ EC
Calcul $\beta \int$EC
Si $\alpha = \beta$ alors **résultat = EC. Fin de programme.**
Si $\alpha < \beta$ alors EC → DPP sinon EC → DPG
Retourner à Boucle itérative

Cette logique se retrouve dans le programme suivant rédigé en Basic (Figure 4) :

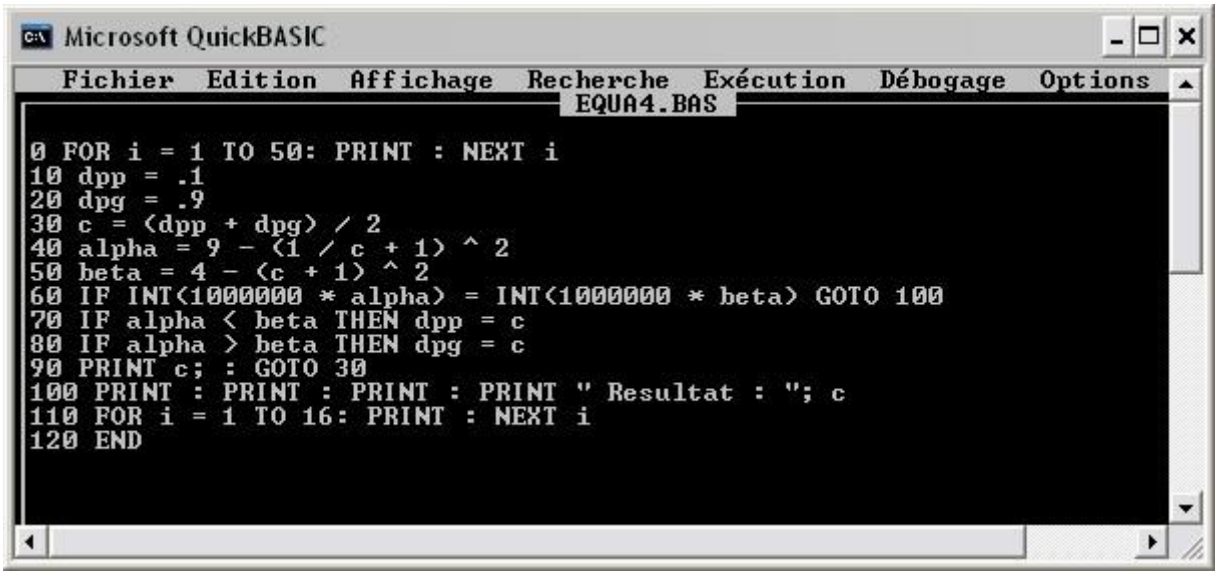


Figure 4

Dans ce programme, les lignes notées 0, 110 et 120 n'ont qu'une fonction de présentation lors de l'exécution du programme et n'ont aucune utilité pour la solution du problème.
L'exécution de ce programme dont la rédaction est infiniment plus rapide que la recherche de la solution mathématique de l'équation du quatrième degré par la méthode de Ferrari est représentée figure 5 :

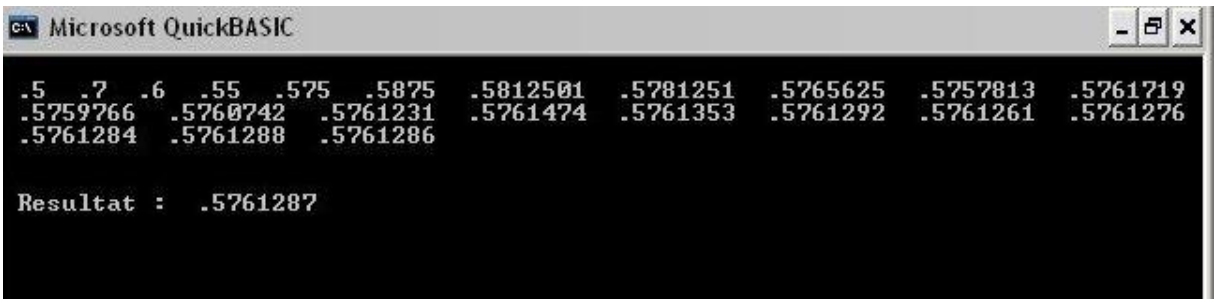


Figure 5

L'évolution de la variable ***e*** dont la valeur évolutive aboutie à la racine de l'équation du quatrième degré est listée afin de montrer la façon dont elle a convergée pour atteindre 0,5761287 (Figure 5). On voit qu'il aura suffit de 23 itérations de la boucle de calcul pour

atteindre le résultat, soit quelques centièmes de secondes sur un micro-ordinateur du commerce. On remarque aussi que la variable ***e*** oscille à plusieurs reprises autour de la valeur cible de temps en temps au-dessus et à d'autres moments au-dessous, avant d'atteindre le résultat. Onze des résultats intermédiaires se situent sous la valeur réelle et douze d'entre eux se situent au-dessus, la convergence est parfaitement symétrique.

J'ai également rédigé un autre algorithme pour vous montrer la richesse des ressources de l'algorithmique. Ici, pour déterminer la fin de la boucle itérative, je n'utilise pas l'identité de résultat entre α et β comme dans le programme précédent mais l'identité entre deux valeurs consécutives de ***e*** appelées c et c2 dans le programme (Figure 6).

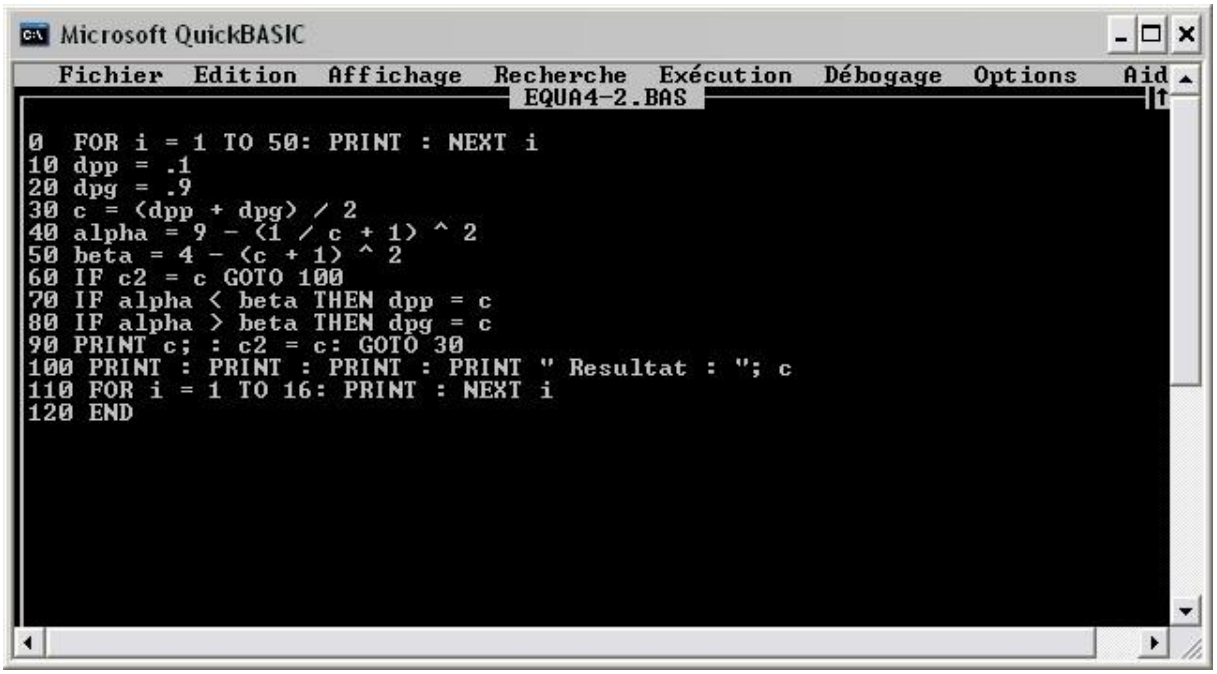


Figure 6

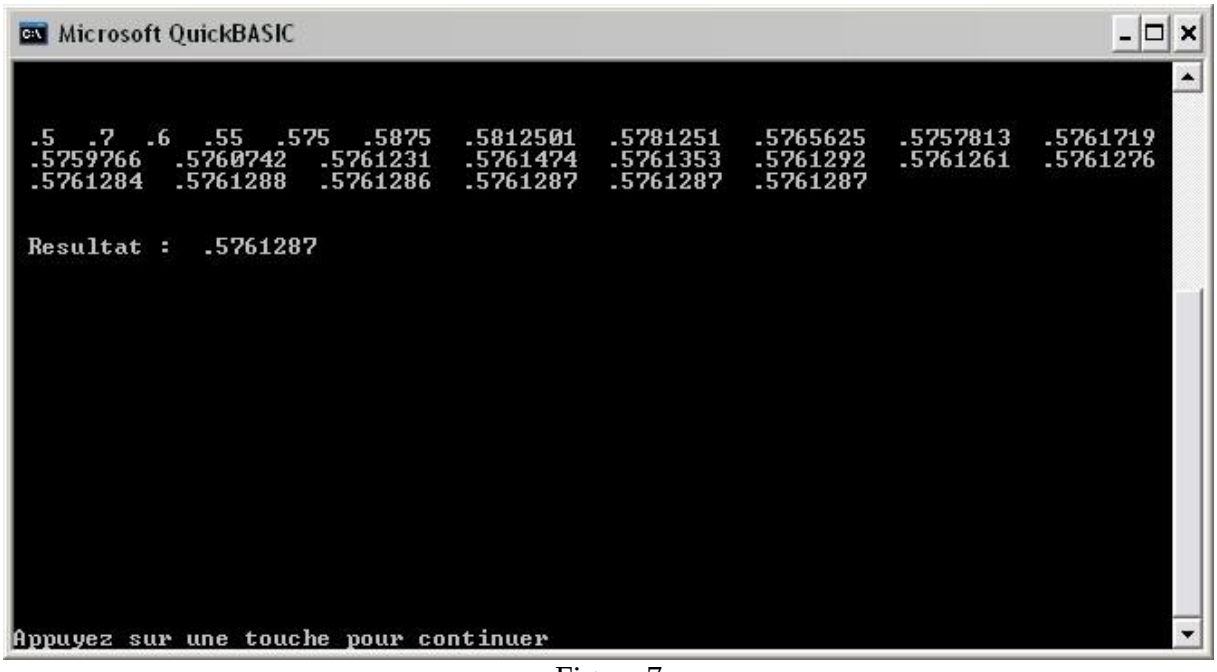


Figure 7

Le résultat de ce programme figure 7 ne diffère du précédent que sur la fin de l'exécution car l'égalité des deux variables e et e2 est plus tardive que l'égalité entre a et b limitée à 6 décimales. Cette égalité semble être déjà atteinte 2 itérations avant la fin mais ce n'est qu'une apparence car le dernier chiffre n'apparaît pas à l'affichage.

Pour le problème dont il s'agit ici, il n'est pas utile de pousser la précision des calculs aussi loin et 3 décimales suffisent puisqu'elles permettent de calculer la largeur du couloir en millimètres. J'avais fait ce calcul à la main à l'aide d'une calculette et 10 itérations avaient suffies pour obtenir le résultat, soit quelques cinq minutes de calculs à peu près.

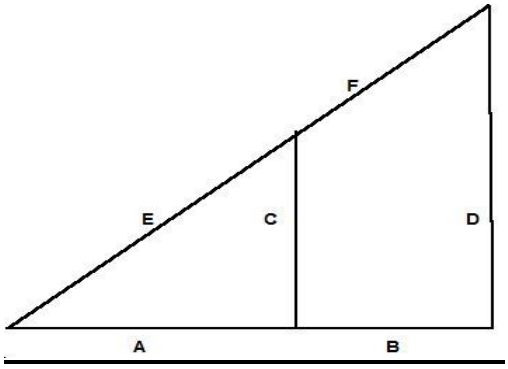
Il faut surtout retenir que l'algorithmique est souvent la meilleure solution pour résoudre un problème complexe dans lequel l'algèbre peine à conduire vers la solution. Pour information, le temps que j'ai pris pour écrire ces programmes, les exécuter et rédiger ce mémo sur le problème des deux échelles du couloir est certainement moindre que celui qu'il en aurait fallu pour résoudre l'équation du quatrième degré en appliquant la méthode de Ferrari… et sans erreurs de calcul.

Voyons maintenant la solution FMB page suivante.

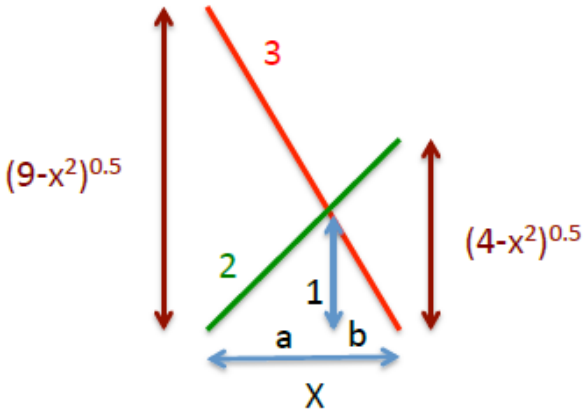
La solution FMB

La solution FMB est en deux parties, la première est une approche géométrique basée sur les propriétés dégagées par le théorème de Thales qui permet d'établir une équation exprimant la largeur du couloir sous une forme au carré (x2) pour un nombre limité de valeurs de x, en l'occurrence 2000, déjà largement surestimé puisque qu'à ce compte l'échelle de 2 mètres se retrouverait à l'horizontal au sol dans la largeur du couloir.

Rappel du théorème de Thales :



Selon Thales, dans un triangle comme celui ci-dessus, mais en réalité quels qu'en soit les angles (j'ai choisi un triangle rectangle, car il correspond à la figure de notre problème d'échelles) nous avons **F/(E+F) = A/(A+B) = C/D** dans laquelle nous trouvons **A/(A+B) = C/D** qui correspond à $\mathbf{a/x = 1/(4-x^2)^{0,5}}$ dans le schéma de nos échelles ci-dessous :



$a+b=x$

$a/x = 1/(4-x^2)^{0.5}$

$b/x = 1/(9-x^2)^{0.5}$

Donc

$1/(4-x^2)^{0.5} + 1/(9-x^2)^{0.5} = 1$

Numériquement:

```
x=indgen(2000)/1000.
s= 1/SQRT(4-x*x) + 1/SQRT(9-x*x) -1
print,x[(sort(abs(s)))[0]]
1.231
```

Cette application du théorème de Thales permet de déterminer l'équation :
$\mathbf{1/(4-x^2)^{0,5} + 1/(9-x^2)^{0,5} - 1 = 0}$ à partir de laquelle l'auteur de la solution imagine le programme suivant écrit en IDL.

L'auteur de la méthode estime que la largeur du couloir est nécessairement inférieure à 2 mètres sans préjuger de sa largeur minimum. En effet, si la largeur du couloir était inférieure ou égale à 2 mètres, l'échelle de 2 mètre serait couchée par terre et ne pourrait pas croiser l'autre à une hauteur de 1 mètre.

A la précision du millimètre il estime donc 2000 cas de largeur possibles.
Il crée une table de 2000 postes qu'il appelle « **x** » de contenus initiaux allant de 0,000 à 1,999 (les données initiales sont en mètres et le résultat est demandé en mm, ce qui justifie la division par 1000).

Pour chacun de ces 2000 postes, il calcule la variable **S** dont une seule satisfera **S=0**.
Ensuite il conditionne l'impression du contenu **x** de chacun des 2000 postes dont la valeur **S** est égale à 0 condition à laquelle un seul poste de la table peut satisfaire.
C'est donc la valeur du **x** correspondant à **S=0** qui répond à la question posée : « Quelle est la largeur du couloir ? », ce qu'il imprime à la troisième ligne de son programme.

Critique de la méthode (avantages, inconvénients, opportunité, adéquation au futur) :

Le programme est d'une grande concision, 3 lignes ont suffi pour obtenir le résultat demandé.

En revanche, bien que l'auteur annonce que le temps de calcul a été de 1,6 secondes sur son PC, cette solution n'est pas généralisable à tous les problèmes possibles de cette nature, l'exigence étant ici mineur puisque 2000 postes sont suffisants pour analyser toutes les situations. De plus le processeur du PC utilisé est un modèle récent, mais sur les premiers PC apparus sur le marché il y a près de 40 ans, cette solution aurait nécessité plus de 100 heures de calcul à en juger par l'accroissement de performance des microprocesseurs durant ces 40 dernières années (0,64 MIPS pour l'Intel 8008 de 1974 contre 147600 MIPS pour l'Intel Core I7-990x de 2010 donc 231000 fois plus rapide). Le MIPS est le million d'instructions par seconde. Voir plus bas le tableau récapitulant la progression des performances des micro-processeurs de la marque Intel (mais les concurrents comme deux du fondeur AMD ont eu des progressions similaires). Le premier PC au monde, le Micral de la société R2E, a été créé par l'ingénieur français, François Gernelle qui a utilisé un Intel 8008.

Il reste qu'en raison des performances courantes des processeurs d'aujourd'hui cette solution est parfaitement adaptée à la nature du problème posé. Je pense même que ce type de solution est le model adapté à la structure des processeurs du futur, notamment si l'informatique quantique tient ses promesses.

Les microprocesseurs d'aujourd'hui, bien que d'une rapidité sans commune mesure avec les premiers microprocesseurs, comme nous venons de le voir, gardent la même structure et continuent à appliquer un processus de traitement séquentiel. C'est-à-dire bien que le programme de FMB n'affiche qu'une ligne de calcul pour $\mathbf{1/(4-x^2)^{0,5} + 1/(9-x^2)^{0,5} -1 = 0}$ l'interpréteur du langage IDL a effectué ce calcul en exécutant 2000 fois une séquence de plusieurs dizaines d'instructions résolvant l'équation pour chacune des 2000 valeurs possibles de x, dans une boucle balayant les 2000 postes de la table en faisant varier x d'un pas de 1 (0,001) à chaque passage de la boucle de programme. Et même chose pour la ligne unique d'affichage de résultat du programme. D'ailleurs, une fois la condition d'égalité à zéro obtenue et autorisant l'affichage du résultat, la boucle testant la condition d'affichage a continuée à s'exécuter 769 fois inutilement.

Date	Nom	Nombre de transistors	Finesse de gravure (nm)	Fréquence de l'horloge	Largeur des données	MIPS
1971	Intel 4004	2 300	10 000	108 kHz	4 bits/4 bits bus	0,06
1974	Intel 8008	6 000	6 000	2 MHz	8 bits/8 bits bus	0,64
1979	Intel 8088	29 000	3 000	5 MHz	16 bits/8 bits bus	0,33
1982	Intel 80286	134 000	1 500	6 à 16 MHz (20 MHz chez AMD)	16 bits/16 bits bus	1
1985	Intel 80386	275 000	1 500	16 à 40 MHz	32 bits/32 bits bus	5
1989	Intel 80486	1 200 000 (800nm)	1 000 à 800	16 à 100 MHz	32 bits/32 bits bus	20
1993	Pentium (Intel P5)	3 100 000	800 à 250	60 à 233 MHz	32 bits/64 bits bus	100
1997	Pentium II	7 500 000	350 à 250	233 à 450 MHz	32 bits/64 bits bus	300
1999	Pentium III	9 500 000	250 à 130	450 à 1 400 MHz	32 bits/64 bits bus	510
2000	Pentium 4	42 000 000	180 à 65	1,3 à 3,8 GHz	32 bits/64 bits bus	1 700
2004	Pentium 4 D (Prescott)	125 000 000	90 à 65	2.66 à 3,6 GHz	32 bits/64 bits bus	9 000
2006	Core 2 Duo (Conroe)	291 000 000	65	2,4 GHz (E6600)	64 bits/64 bits bus	22 000
2007	Core 2 Quad (Kentsfield)	2*291 000 000	65	3 GHz (Q6850)	64 bits/64 bits bus	2*22 000 (?)
2008	Core 2 Duo (Wolfdale)	410 000 000	45	3,33 GHz (E8600)	64 bits/64 bits bus	~24 200
2008	Core 2 Quad (Yorkfield)	2*410 000 000	45	3,2 GHz (QX9770)	64 bits/64 bits bus	~2*24 200
2008	Intel Core i7 (Bloomfield)	731 000 000	45	3,33 GHz (Core i7 975X)	64 bits/64 bits bus	?
2009	Intel Core i5/i7 (Lynnfield)	774 000 000	45	3,06 GHz (I7 880)	64 bits/64 bits bus	76 383
2010	Intel Core i7 (Gulftown)	1 170 000 000	32	3,47 GHz (Core i7 990X)	64 bits/64 bits bus	147 600
2011	Intel Core i3/i5/i7 (Sandy Bridge)	1 160 000 000	32	3,5 GHz (Core i7 2700K)	64 bits/64 bits bus	
2011	Intel Core i7/Xeon (Sandy Bridge-E)	2 270 000 000	32	3,5 GHz (Core i7 3970X)	64 bits/64 bits bus	1 ou 2
2012	Intel Core i3/i5/i7 (Ivy Bridge)	1 400 000 000	22	3,5 GHz (Core i7 3770K)	64 bits/64 bits bus	
2013	Intel Core i3/i5/i7 (Haswell)	1 400 000 000	22	3,8 GHz (Core i7 4770K)	64 bits/64 bits bus	
2014	Intel Core i3/i5/i7 (Broadwell)	1 400 000 000	14	3,8 GHz (Core i7 5775R)	64 bits/64 bits bus	

Les processeurs multi-cœurs d'aujourd'hui sont cependant déjà capables de traiter l'équation ci-dessus avec un certain niveau de parallélisme. L'interpréteur IDL, comme le ferait également un compilateur, sait très bien que les deux groupes de variables de l'équation séparés par le signe + sont indépendant et leur calcul confié à deux cœurs de processeurs différents, chacun fournissant son résultat partiel dont la somme sera faite ultérieurement par un seul processeur avec la soustraction du 1 qui clôture l'équation. Ainsi, la « plusieurs » dizaines d'instructions à exécuter séquentiellement a été traitée en deux fois moins de « plusieurs » dizaines que ce qu'aurait fait un processeur scalaire mono-cœur d'autrefois.

Mais déjà aujourd'hui, il existe des ordinateurs structurés autour de microprocesseurs organisés sous forme de matrices (array processeurs) capables de traiter une équation, éventuellement la même, simultanément sur plusieurs milliers de processeurs avec des valeurs de variables différentes.
Sur une telle machine, l'équation $\mathbf{1/(4-x^2)^{0,5} +1/(9-x^2)^{0,5} -1 = 0}$ serait traitée simultanément par 2000 processeurs mais chacun avec une variable x différente de valeurs 0,000 à 1,999. Ce sont les compilateurs et/ou interpréteurs du langage dans lequel est rédigé le programme source qui développe le code machine en éclatant la boucle temporelle en réseau spatial. Sur une telle machine, la solution FMB n'aurait pris que quelques microsecondes. Ces machines n'existent qu'à quelques exemplaires dans le monde aujourd'hui. L'une d'elles et parmi les moins puissantes de ces machines supergéantes se trouve au CEA, il s'agit du CURIE

construit par BULL qui n'atteint « que » 2 petaflops alors que le TITAN de Cray Rechearch exécute 27 petaflops.

Le « flops » qui signifie « opérations en virgule flottante par seconde » est l'unité de mesure pour comparer les performances des ordinateurs à vocation scientifique qui remplace « L'ips » qui signifie « instructions par seconde » et utilisé plus haut pour comparer les performance des microprocesseurs depuis leur origine jusqu'à maintenant. Bien que depuis la fin du XXe siècle tous les microprocesseurs soient dotés d'instructions traitant aussi bien des variables en virgules fixes qu'en virgules flottantes les comparaisons en absolue ne sont plus très significatives. Les mesures objectives de comparaison doivent préciser nécessairement le contexte d'emploi, c'est-à-dire la nature du type de variables sur lesquelles porte la comparaison de performance. Un processeur **Pa** peut se montrer supérieur à un processeur **Pb** sur un test de type gestion traitant de variables en virgules fixe alors que sur un test traitant d'une application scientifique traitant de variables en virgules flottantes**, Pb** sera plus performant que **Pa**.

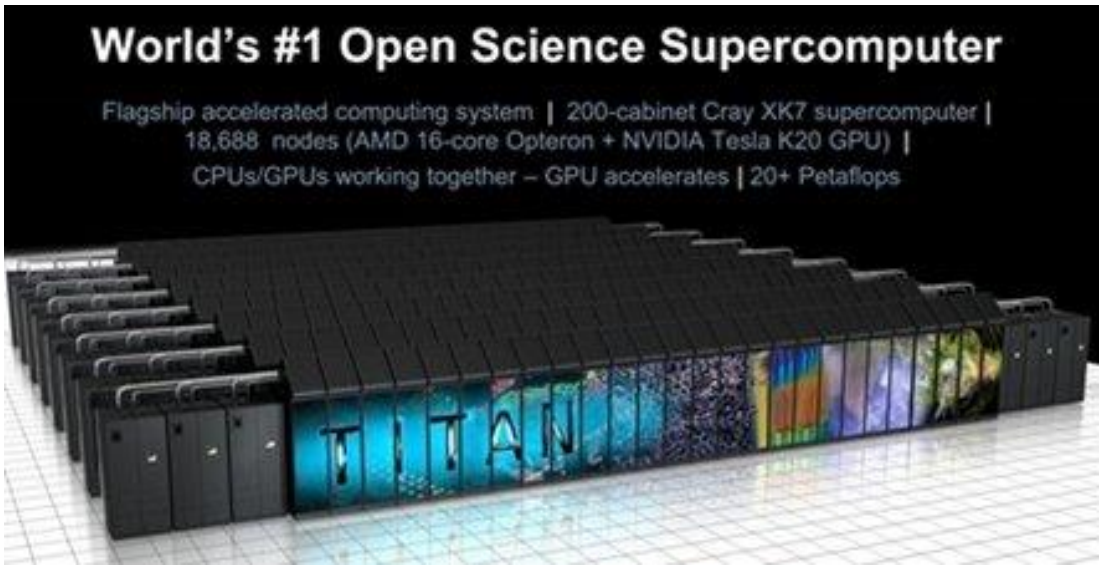
Mais là encore, le fonctionnement de ces milliers de microprocesseurs travaillant en parallèle est-il toujours le même que celui du processeur classique, celui qui équipe votre PC. C'est-à-dire que chaque processeur traite les instructions séquentiellement, au fur et à mesure de leur ordre d'écriture dans le programme dont l'exécution leur est soumis, sauf dans le cas de branchements, bien sûr.

Ce que l'avenir nous réserve peut-être, c'est l'architecture quantique du processeur, un mode naturelle d'exécution de toutes les solutions possibles d'une équation. Ce serait l'état de superposition quantique qui serait mis à contribution pour traiter toutes les solutions possibles simultanément. Mais cela, c'est certainement encore loin. J'ai même de sérieux doute sur la faisabilité de ce genre de processus quantiques appliqués à des portes logiques traitant de variables représentant un nombre important de bit quantiques (Qbit), les propriétés quantiques semblant se dissoudre par décohérence lorsqu'elles s'étendent sur des espaces macroscopiques.

Cependant, je pense que la fonction mémoire qui peut être répartie sur des îlots de Qbits indépendants et isolés, les mettant à l'abri du phénomène de décohérence dû aux interactions de voisinages massifs, peut bénéficier un jour de cette avancée technologique et cela dans un avenir prévisible. J'ai une plus grande confiance dans cette application quantique de l'informatique que dans les opérateurs quantiques de traitement proprement dit de l'information.

Quoi qu'il en soit ces considérations n'étant pas encore à l'ordre du jour, je souhaitais seulement vous montrer deux façons très différentes d'utiliser efficacement les ressources de l'informatique d'aujourd'hui pour résoudre un problème dont la résolution uniquement humaine est un problème à la limite de l'insoluble sur le plan pratique. En effet, même si sur un plan théorique, l'équation du 4e degré est aujourd'hui résolue, les multiples racines imaginaires disséminées dans les racines réelles de ce genre d'équation sont de véritables chausse-trappes pour un esprit humain qui s'y perd inéluctablement.

Le CURIE de Bull au CEA



Le TITAN de Cray

Le K de Fujitsu