Seul l'être humain calcule dans l'ensemble des nombres réels !
Dès qu'on demande à une machine de calculer à partir d'une représentation finie approchée d'un nombre réel, nous devons nous attendre à obtenir des résultats inexacts qui ne seraient utilisables dans une démonstration mathématique que si on pouvait quantifier l'erreur commise par la machine.
Je ne me sens donc pas capable de répondre à une question de problème ou d'exercice me demandant de calculer avec une calculatrice ou un tableur un nombre réel ou une des ses valeurs approchées sans connaître d'une part la représentation interne des nombres dans les registres du microprocesseur et les mémoires, d'autre part la précision des algorithmes utilisés pour les opérations et les fonctions élémentaires.
Dans le cas des microprocesseurs des ordinateurs de type PC, une norme existe pour la représentation des nombres sous la forme dite "virgule flottante" et les opérations de base sur ces nombres : c'est la norme IEEE754 qui, si elle est rigoureusement utilisée, permet d'établir des preuves de la précision des résultats obtenus.
L'atelier se déroulait en deux parties : d'abord l'utilisation de petits programmes en JavaScript dans des pages HTML pour mettre en évidence les effets de l'utilisation de la norme et quelques-unes de ses propriétés, puis la réalisation de feuilles de calcul pour montrer que les programmeurs des tableurs, en voulant cacher les effets de la représentation des nombres à l'utilisateur, ont doté leurs logiciels d'incohérences permettant par exemple de prouver que 0 = 1 !

I. Javascript et HTML.

Dès qu'un ordinateur dispose d'un éditeur de texte et d'un navigateur internet, l'utilisateur possède un environnement de programmation rudimentaire, mais efficace, le navigateur intégrant un interpréteur du langage JavaScript. (J'utilise cet environnement avec mes élèves car il a l'avantage de leur permettre de publier leurs compte-rendu de TP sur l'intranet du lycée.)

a. squelette de document HTML utilisé :

```
<html>
<head><title>Expérimentons...</title></head>
<body>
<pre>
<script language="JavaScript">
a=Math.sqrt(2);
document.write(a.toPrecision(60));
</script>
</pre>
</body>
</html>
```

Commentaires :
HTML est un langage à balises. Les balises sont encadrées par < et >. Elles vont souvent par paires, une balise ouvrante et une fermante, comme des parenthèses, la balise fermante a le nom de la balise ouvrante précédé de /.
Dans le squelette ci-dessus, <html>...</html> encadre tout le document,
<head>...</head> est un en-tête contenant des informations à transmettre au navigateur, qui ne sont pas affichées dans la page Web,
<title>...</title> indique au navigateur le texte à afficher dans la barre de titre de la fenêtre,
<body>...</body> encadrent le contenu à afficher dans la page,
<pre>...</pre> sont utilisées pour afficher rigoureusement le contenu, espaces, tabulations et sauts de lignes sont rigoureusement reproduits,
<script>...</script> encadrent le programme rédigé dans le langage JavaScript (Attention, il est sensible à la casse : "math" et "Math" ne sont pas interchangeables).
Ce document doit afficher la valeur exacte de la représentation de la racine carrée de 2 par l'ordinateur.
Les dernières décimales non nulles mettent en évidence une représentation par une fraction binaire, et le décompte des décimales amène à penser que ce nombre est connu par l'ordinateur avec 52 bits après la virgule.

b. Expérience frappante :

Calcul des termes de la suite géométrique de raison phi' ((1-sqrt(5))/2) en utilisant la relation de récurrence de Fibonacci : u_n = u_{n-1} + u_{n-2}.

1. Boucle pour afficher les 20 premiers termes :

On remplace les 2 lignes de script du squelette par :

```
u=1;
v=(1-Math.sqrt(5))/2;
for(k=1;k<=20;k++){
document.write(k,"\t",v,"\n");
w=u+v;u=v;v=w;
}
```

L'instruction "for" se traduit "pour k variant de 1 à 20 de 1 en 1".

"\t" est une tabulation, "\n" un saut de ligne. (J'utilise ces codes ici avec la balise <pre> pour ne pas avoir à apprendre toutes les balises permettant de formatter un tableau en HTML.)

On observe un comportement habituel pour une suite géométrique dont la raison est négative et de valeur absolue strictement inférieure à 1.

2. Affichage de 200 termes :

Il suffit de rajouter un "0" dans l'instruction "for".

L'effet de la représentation finie des nombres utilisés est frappant : la suite que devait converger vers 0, semble maintenant diverger vers l'infini négatif !!.

3. Affichage d'un terme sur 10 :

Pour voir une seule page, on ne demande l'affichage que des termes de rang multiple de 10 en ajoutant un test devant la commande d'affichage "document.write". La ligne devient :

```
if(k%10==0) document.write(k,"\t",v,"\n");
```

"%" étant ici l'opération "modulo".

4. Estimation de l'erreur sur la représentation de phi' :

Dans la mémoire de l'ordinateur, on ne peut pas stocker phi', mais le rationnel (phi' + epsilon).

La suite calculée par l'ordinateur est dans ce cas asymptotiquement équivalente à (epsilon/sqrt(5))*phi^n, phi étant le nombre d'or ((1+sqrt(5))/2).(La démonstration est laissée au lecteur !)

On peut donc estimer epsilon en calculant u_200/phi^200*sqrt(5).

Poue l'afficher, il suffit d'ajouter à la fin du script :

```
phi=(1+Math.sqrt(5))/2;
document.write(u/Math.pow(phi,200)*Math.sqrt(5));
```

5. Recherche de la fraction binaire représentant phi' :

Script utilisé :

```
a=(sqrt(5)-1)/2;
n=0;
while(Math.floor(a)!=a){
n=n+1;a=2*a;
}
document.write("ce nombre est représenté par la fraction -",a,"/2^",n);
```

Commentaires : le travail est fait sur l'opposé de phi' pour utiliser facilement la fonction "floor" (partie entière). On multiplie par 2 jusqu'à obtenir un entier, en comptant le nombre de multiplications.

On peut maintenant vérifier la valeur approchée de epsilon obtenue précédemment avec un logiciel de calcul formel (par exemple xcas, présenté dans les ateliers D6 et ??).

En tapant Digits:=30;evalf((1-sqrt(5))/2+??????????????/2^??), on obtient une validation du calcul précédent.

c. Informations sur la représentation normalisée :

1. Recherche de la taille maximum de la mantisse :

sqrt(2) est représenté par un fraction de dénominateur 2^52, phi' par une fraction de dénominateur 2^49, quel est le nombre maximum de bits utilisables ?

Script utilisé :

```
a=0.5;
n=1;
while((1+a)!=1){
a=a/2;n=n+1;
}
document.write("nombre maximum de chiffres binaires après la virgule : ",n);
```

Conclusion : La norme utilise une mantisse de 53 chiffres, mais comme on décale par multiplications ou divisions par 2 jusqu'à obtenir une mantisse commençant par 1, seuls 52 bits sont utilisés pour la mantisse.

2. Recherche de l'exposant maximum :

Script utilisé :

```
a=2;
n=1;
while (isFinite(a)){
a *= 2;
n += 1}
document.write("Exposant de 2 Maximum : ",n-1);
```

3. Conclusion :

Les exposants semblent être codés sur 11 bits (permettant d'utiliser la plage de -1024 à 1023),
52+11=63, en ajoutant le bit de signe, les nombres flottants sont stockés sur 8 octets.

II. Les incohérences des tableurs.

Le processeur a des capacités mais il ne fera que ce que le programmeur lui a dit de faire.
Les programmeurs des tableurs (celui de OpenOffice et StarOffice, et Excel) en essayant de cacher à l'utilisateur certaines "erreurs" dues à la représentation finie des nombres, provoquent des résultats incompatibles avec la logique mathématique.
Feuilles de calcul réalisées :

1. vérifier à partir de quel entier naturel n le tableur affirme que 1+2^(-n)=1 :

A1: 1	B1: 0.5	C1: =1+B1	D1: =(C1=1)
A2: =A1+1	B2: =B1/2	C2: =1+B2	D2: =(C2=1)

puis recopie vers le bas.
Les valeurs diffèrent de celles attendues si la norme IEEE754 est utilisées.

2. vérifier que le tableur connait bien 53 chiffres binaires de la racine carrée de 2 :

A1: =racine(2)	B1: =ent(A1)
A2: =2*(A1-B1)	B2: =ent(A2)

puis recopie vers le bas.
On trouve bien "1" comme 53-ième bit !!

3. Première contradiction :

D'après W. Kahan, qui a participé à la création des coprocesseurs mathématiques pour Intel, la norme IEEE754 a pour conséquence que 30*2.54 et 3*25.4 ont des représentations différentes.

À l'aide de la feuille de calcul précédente, nous pouvons vérifier que la représentation du premier nombre se termine par "1101" alors que celle du second se termine par "1100".
Un des axiomes de l'égalité affirme que si a=b, alors f(a)=f(b) pour toute fonction f pour laquelle les écritures f(a) et f (b) ont un sens. Le fait d'obtenir deux résultats différents pour le même calcul prouve par contraposée de l'affirmation précédente, que les valeurs initiales sont différentes.
Réalisons la feuille de calcul suivante :
A1: =3*25.4 B1: =30*2.54 C1: =(A1=B1)
Le tableur affirme que les 2 nombres dont les représentations en mémoire sont différentes sont égaux !!
Complétons la feuille :
A2: =A1/B1 B2: 1 C2: =B1/A1 D2: =(A2=B2) E2: =(A2=C2) F2: (B2=C2)
A3: =A2^1e15 B3: =B2^1e15 C3: =C2^1e15
A4: =A3^1e30 B4: =B3^1e30 C4: =C3^1e30
Nous avons maintenant 3 nombres égaux qui, lorsqu'on leur applique la même fonction, donnent 3 résultats différents et, si on utilise l'axiome cité précédemment, les 3 antécédents étant déclarés égaux, on peut conclure 0=1=infini.

4. Deuxième contradiction :
Un autre axiome de l'égalité affirme que si a=b et b=c, alors a=c.
Feuille de calcul pour OpenOffice ou StarOffice :
A1: =1e15 B1: =1e15+3 C1: =1e15-3
A2: =(A1=B1) B2: =(A1=C1) C2: =(B1=C1)
L'axiome étant contredit, aucun raisonnement logique ne peut être mené en utilisant les résultats de ce logiciel.
Je vous laisse expérimenter avec Excel. Avec les versions que j'ai utilisées, je n'arrive pas directement à obtenir la même contradiction, mais en passant à des tests utilisant la comparaison de la différence de deux nombres avec 0, j'ai trouvé des nombres déclarés égaux dont la différence n'est pas nulle !!!