Chapitre 1
Généralités sur le calcul scientifique

I. Motivation

La résolution rigoureuse des problèmes via méthodes numérique est l'un des objectifs de l'analyse numérique. Le traitement des erreurs de ces méthodes est un sujet majeur.
Les solutions numériques sont affectées par des erreurs de différentes origines, essentiellement :

- les erreurs sur les données, causées par une connaissance imparfaite des données du problème à résoudre, comme lorsqu'elles sont issues de mesures physiques soumises à des contraintes expérimentales;
- les erreurs de troncature, d'approximation ou de discrétisation, introduites par les schémas de résolution numérique utilisés, comme par exemple le remplacement d'une dérivée par une différence finie,
- les erreurs d'arrondi dans les opérations arithmétiques conséquences de la représentation en précision finie.

On s'intéresse en ce chapitre, aux résultats approchés par une arithmétique de nombres en virgule flottante.

II. Représentation des nombres réels en virgule flottante normalisée

II.1. Principe

Tout nombre réel est représentable sur machine sous la forme $s\ m\ \beta^e$.

Où m : le significande ou la mantisse est un réel positif, composé d'au plus t chiffres en base β,
e : l'exposant, un entier signé, compris entre deux bornes e_{min} et e_{max} (généralement $e_{min} < 0$ et $e_{max} > 0$).

En autorisant l'exposant e à changer de valeur, on laisse le séparateur (la virgule ou le point selon la convention) « flotter ». Ainsi, sur une même mantisse, on peut représenter de grandes ou de petites valeurs. Les nombres ainsi définis sont dits à virgule flottante, et l'entier t est dit la précision ou encore nombre de chiffres significatifs du nombre à virgule flottante.

L'exposant dépend des conventions adoptées sur le significande, notamment la place du séparateur dans ce dernier. On parle de représentation normalisée lorsque le chiffre de poids fort du significande est non nul, ce qui assure, une fois la position du séparateur fixée, que tout réel non nul représentable ne possède qu'une seule représentation.
En base binaire, une conséquence intéressante est que le premier bit du significande d'un nombre à virgule flottante normalisé est toujours égal à 1. On peut alors décider de ne pas le stocker physiquement et on parle de bit de poids fort implicite ou caché. Le codage du signe se fait par la valeur 0 du bit correspondant si le nombre est positif et 1 s'il est négatif (le nombre 0 étant considéré signé, il a deux représentations).

La norme IEEE 754-1985 définit deux formats pour la représentation des nombres réels en machine:

1) un format simple précision sur un mot mémoire de 32 bits (1 bit de signe, 8 bits d'exposant, 23 bits de significande, avec bit de poids fort implicite),
2) un format double précision sur un mot-mémoire de 64 bits (1 bit de signe, 11 bits d'exposant, 52 bits de significande, avec bit de poids fort implicite),

L'exposant est biaisé d'une valeur b= 2^{e-1} -1 afin de le stocker sous la forme d'un entier non signé. Pour les nombres en double précision, b=1023.
La restriction imposée par la normalisation avec bit de poids fort implicite fait que le nombre 0 n'est pas représentable par un nombre normalisé. Pour le représenter, la norme réserve l'exposant nul avec bit de signe mis à 0 et à 1.

II.2. Intervalle de représentation des nombres en précision finie

Construisons l'ensemble F des nombres f à virgule flottante à partir d'une représentation normalisée $s\ m\ \beta^e$ mais avec bit implicite. F est un sous ensemble fini de points de la droite des réels $\mathbb{R}$.

La mantisse sur t bits d'un nombre à virgule flottante normalisée avec bit de poids fort caché en base 2 vérifie :

0	0	...	0	0	$\leq$	m	$\leq$	1	1	...	1	1
0					$\leq$	m	$\leq$	$2^t - 1$				

Alors

$$0 \leq m \times 2^{-t} \leq 1 - 2^{-t}$$

En considérant le bit caché :

$$1 \quad \leq \quad 1 + (m \times 2^{-t}) \quad \leq \quad 2 - 2^{-t}$$

D'où : $f \in F,\ |f| = 1, m \times 2^e$ avec $e \in [e_{min}, e_{max}]$, vérifie

$$\begin{array}{ccccc} val\_min & \leq & |f| & \leq & val\_max \\ 2^{e_{min}} & \leq & |f| & \leq & (2 - 2^{-t}) \times 2^{e_{max}} \end{array}$$

On note $F(2, t,\ e_{min}, e_{max})$ l'ensemble discret constitué de l'union de ces nombres f avec le singleton $\{0\}$.

Les nombres à virgule flottante en simple (resp. double) précision standard correspondent aux éléments de l'ensemble F(2,23,-126,127) (resp. F(2,52,-1022,1023)) .

D'où, les nombres à virgule flottante en simple (resp. double) précision standard peuvent décrire des nombres réels dont la valeur absolue est comprise entre $2^{-126} \approx 1{,}175494351\ 10^{-38}$ et $(1 - 2^{-24})\ 2^{128} \approx 3{,}4028235\ 10^{38}$
(resp. $2^{-1022} \approx 2{,}2250738585072020\ 10^{-308}$ et $(1 - 2^{-53})\ 2^{1024} \approx 1{,}7976931348623157\ 10^{308}$).

II.3. Arrondis en précision finie

Les ensembles de nombres à virgule flottante sont de cardinalité finie ce qui pose le problème de la représentation exacte en machine des nombres réels. La solution consiste à remplacer le nombre par un autre admettant une représentation en virgule flottante selon le système considéré.

Pour un nombre réel dont la valeur absolue est comprise entre les bornes des nombres en virgule flottante, son remplacement peut être effectué de différentes manières :

1) Ecrire le nombre et ne conserver que les *t* premiers chiffres (selon la convention de représentation d'une mantisse). On parle alors de troncature ou d'arrondi vers zéro (un arrondi dirigé).
2) Substituer au nombre réel le nombre à virgule flottante qui lui est le plus proche; c'est l'arrondi au plus proche. Lorsque le nombre se situe à égale distance des deux nombres à virgule flottante qui l'entourent, on choisit la valeur de l'arrondi en faisant appel à des arrondis dirigés. On peut alors prendre :
   • le nombre à virgule flottante le plus petit (resp. grand), c'est l'arrondi par défaut (resp. excès) ;
   • le nombre à virgule flottante le plus petit (resp. grand) en valeur absolue, c'est l'arrondi vers zéro (resp. vers l'infini)
   • le nombre à virgule flottante dont le dernier chiffre de la mantisse est pair (resp. impair), c'est l'arrondi au chiffre pair (resp. impair).

En IEEE754, les modes d'arrondi proposés sont au nombre de quatre (1.l'arrondi au plus proche, qui est celui utilisé par défaut, avec une stratégie d'arrondi au chiffre pair quand un choix est nécessaire, 2.l'arrondi vers zéro, 3 et 4. les arrondis vers $\pm\infty$).

II.4. L'opérateur float : fl(x)

Pour décrire l'ensemble des nombres réels qui trouvent une représentation en virgule flottante sur F, on définit l'ensemble :

$$G = \quad \{x \in \mathbb{R} \text{ tel que } \quad val\_min \leq |x| \leq val\_max\} \cup \{0\}$$

Ou encore : $G = \quad [-(2^{-t} - 2) \times 2^{e_{max}}, -2^{e_{min}}] \cup \{0\} \cup [2^{e_{min}}, (2 - 2^{-t}) \times 2^{e_{max}}]$

Et l'opérateur $fl\ (x)$: l'arrondi au plus proche d'un nombre réel *x*, définissant ainsi l'application :

$$\begin{array}{lll} fl: & G \rightarrow & F(\beta, t,\ e_{min}, e_{max}) \\ & x \rightarrow & f, \text{ plus proche } f \in F \end{array}$$

La valeur de l'arrondit est décidé comme suit :

$$fl(x) = (-1)^s (1.a_1 a_2 \ldots\ldots\ \hat{a}_t)\beta^e, \qquad \hat{a_t} = \begin{cases} a_t & \text{si } a_{t+1} < \beta/2 \\ a_t + 1 & \text{si } a_{t+1} \geq \beta/2 \end{cases}$$

II.5. Débordement

Dans le cas d'un nombre réel non nul dont la valeur absolue n'appartient pas à F, il n'est pas possible d'effectuer un remplacement par un arrondi. Si la valeur absolue du nombre est supérieure à la valeur maximale représentable, ce dépassement de la capacité de stockage est appelé débordement vers l'infini. Il est dit débordement vers zéro si le nombre est inférieur à la valeur positive minimale représentable.

L'occurrence d'un débordement vers l'infini est un problème sérieux, notamment lorsque le nombre qui le provoque est le résultat d'une opération, et devrait conduire à l'interruption du calcul en cours. Néanmoins, dans le standard IEEE, il produit les valeurs spéciales +Inf et –Inf (dont l'exposant est une valeur réservé à 2047 pour représentation en double précision).

Un débordement vers zéro est moins grave et un remplacement par 0 (la valeur la plus proche) est en général effectué, mais cette solution n'est cependant pas toujours satisfaisante.

II.6. Epsilon machine

Le epsilon machine est par définition, la distance entre le nombre 1 et le nombre à virgule flottante le plus proche qui lui est supérieur, définit par l'expression : $fl\,(1 + \text{eps}) > 1$

La mantisse de la valeur 1 sur une machine avec des mots normalisée en IEEE754 sur t bits, est de : $\overbrace{0\ \ 0\ \ \cdots\ \ 0\ \ 0}^{t}$, alors que celle du nombre suivant : $fl\,(1 + \text{eps})$ sur t bits est de $\overbrace{0\ \ 0\ \ \cdots\ \ 0\ \ 1}^{t}$. L'exposant de 1 et de 1+eps est à 0, leur bit caché étant identiquement à 1.

D'où, la distance qui sépare le nombre 1 du plus proche nombre flottant suivant est égale à la différence des mantisses des deux nombres. Ainsi pour une mantisse comportant t bits on a :

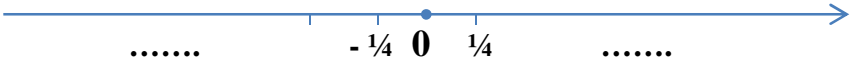
$$\text{eps}_{\text{machine}} = \varepsilon_M = 2^{-t}$$

Dans le cas des nombres représentés en double précision, notamment avec Matlab, qui utilise t = 52 bits pour la mantisse et un arrondit au plus proche, le epsilon machine est alors identifiée par : $\text{eps} = 2^{-52} \approx 2.22 \times 10^{-16}$.

Exemple

Déterminer les points du système F(2,3,-2,2) =l'union du singleton {0} et de l'ensemble des nombres à virgule flottante construit à partir d'une représentation normalisée avec bit implicite, sur la droite des réels.

$f = n \times 2^{e-t}$									
n					2^{e-t}				
					$e=-2$	$e=-1$	$e=0$	$e=1$	$e=2$
					1/32	1/16	1/8	1/4	1/2
1	0	0	0	=8	1/4	1/2	1	2	4
1	0	0	1	=9	9/32	9/16	9/8	9/4	9/2
1	0	1	0	=10	5/16	5/8	5/4	5/2	5
1	0	1	1	=11	11/32	11/16	11/8	11/4	11/2
1	1	0	0	=12	12/32	3/4	3/2	3	6
1	1	0	1	=13	13/32	13/16	13/8	13/4	13/2
1	1	1	0	=14	14/32	7/8	7/4	7/2	7
1	1	1	1	=15	15/32	15/16	15/8	15/4	15/2



II.7. Mesure de l'erreur

Pour estimer la précision sur une valeur, on mesure l'erreur entre sa valeur approchée et la valeur exacte en utilisant l'une des formules suivantes :

l'erreur absolue est définie entre un scalaire réel x et son approximation $\hat{x}$ par : $|x - \hat{x}|$,

l'erreur relative utilisée lorsque x est non nul, est donnée par $\frac{|x-\hat{x}|}{|x|}$.

L'erreur relative est invariante au changement d'échelle, c'est pourquoi, elle est privilégiée pour évaluer la précision d'un résultat. Son utilisation pose des problèmes lorsque x prends des valeurs qui sont proches de zéro. Pour pallier à cet inconvénient on utilise souvent dans la pratique la mesure $\frac{|x-\hat{x}|}{|x|+1}$. Cette mesure a les caractéristiques de l'erreur relative si$|x| \gg 1$, et les caractéristiques de l'erreur absolue si $|x| \ll 1$

II.8. Erreurs sur fl(x)

L'erreur absolue sur la représentation d'un nombre x en virgule flottante est dite erreur d'arrondit et notée E(x) =$|x - fl(x)|$. C'est la modification de la mantisse dû au dernier bit à l'extrême droite,

Pour le calcul de l'erreur relative E_{rel} sur la représentation des réels en flottant, posons le réel *x* strictement positif (pour simplifier les calculs sans perte de généralité), écrit sous la forme : $s\ m\ \beta^k$, avec $\beta = 2$ et une mantisse *m* ayant la partie fractionnaire sur t bit significatif et un bit explicit pour partie entière,

Alors on note $m = n.\beta^{-t}$, avec n entier vérifiant $\beta^t \leq n < \beta^{t+1}$ d'où $|x| = n \cdot \beta^{-t} \cdot \beta^k$

On observe que *x* se trouve entre les deux nombres à virgule flottante adjacents $y_1 = \lfloor n \rfloor \cdot \beta^{-t+k}$ et $y_2 = \lceil n \rceil \cdot \beta^{-t+k}$, $\lfloor n \rfloor$(resp. $\lceil n \rceil$) désigne la partie entière par défaut (resp. par excès) du réel n.

Par conséquent, l'arrondit au plus proche est soit $fl(x) = y_1$ ou $fl(x) = y_2$

D'où $|x - fl(x)| \leq \frac{|y_2 - y_1|}{2} \leq \frac{\beta^{-t+k}}{2}$

l'erreur absolue est majorée par : $E(x) \leq \frac{1}{2}\beta^{-t+k}$

On a alors : $\frac{|x-fl(x)|}{|x|} \leq \frac{\beta^{-t+k}}{2} \cdot \frac{1}{n \cdot \beta^{-t+k}} \leq \frac{\beta^{-t}}{2}$

D'où le résultat fondamental de l'arithmétique en virgule flottante suivant :

$\forall x \in G$, pour lequel l'arrondi vers le double le plus proche est défini dans F, il est approché avec une erreur relative en valeur absolue ne dépassant pas la valeur $u = \frac{1}{2}eps_{mach}$, dite précision machine ou unité d'arrondit.

$$\frac{|x - fl(x)|}{|x|} \leq u = \frac{1}{2}eps_{mach}$$

La relation qui définit l'erreur relative peut aussi s'écrire :

$$fl(x) = x(1 + \boldsymbol{\delta}) \quad \text{avec} \quad |\boldsymbol{\delta}| \leq \boldsymbol{u}$$

Cette relation est à la base de toute étude des erreurs d'arrondi.

Remarque :

Pour caractériser la précision d'un système de représentation des nombres réels en virgule flottante, on présente souvent les mesures suivantes :

- précision machine ou epsilon machine
- erreur d'arrondit
- erreur relative

III. Arithmétique en précision finie

L'ensemble des nombres représentables sur une machine étant limité, il faut définir sur celui-ci une arithmétique reproduisant de manière aussi fidèle que possible celle existant sur R. On parle alors d'arithmétique en précision finie.
En effet, toute opération arithmétique avec des nombres en virgule flottante implique une erreur d'arrondi ou de troncature, selon l'architecture de la machine, dont il importe de tenir compte lors de la mise en œuvre de certains algorithmes de résolution.

Cette section présente quelques grands principes d'arithmétique en virgule flottante ainsi que de bons usages pour minimiser les erreurs d'arrondi.

1) Modèle d'arithmétique à virgule flottante

Le modèle d'arithmétique, dû à Wilkinson, classiquement utilisé pour l'analyse d'erreur d'un algorithme possède la propriété suivante : en désignant par le symbole « op » n'importe quelle opération arithmétique de base (addition, soustraction, multiplication, division), et soient x et y deux nombres à virgule flottante tels que le résultat x op y ne provoque pas de dépassement de capacité, alors on a

$$\mathrm{fl}(x \text{ op } y) = (x \text{ op } y)(1 + \boldsymbol{\delta}),\ |\delta| \leq u,\ \text{op} = +, -, *, /$$

L'opérateur fl(·) est appliquée à une expression arithmétique pour représenter la valeur effectivement fournie par la machine pour l'évaluation de cette expression.

La propriété assure alors que cette valeur est « aussi bonne » que l'arrondi du résultat exact, au sens où l'erreur relative possède la même borne dans les deux cas.

2) Propagation des erreurs

Nous considérons le cas le plus simple de propagation d'erreurs : celle ayant lieu dans des opérations arithmétiques élémentaires comme la multiplication, la division, l'addition ou la soustraction. Pour l'étudier, nous allons supposer que les opérandes contiennent des erreurs.

Cas de la multiplication
On considère les valeurs $x + \delta x$ et $y + \delta y$, représentant deux réels non nuls x et y respectivement entachés des erreurs absolues $|\delta x|$ et $|\delta y|$. En supposant que les erreurs relatives δx/x et δy/y sont suffisamment petites en valeur absolue pour que les termes d'ordre deux en ces quantités soient négligeables, on obtient :

$$(x + \delta x)(y + \delta y) = x\,y\,(1 + \delta x/x + \delta y/y + \delta x.\delta y\,/x\,y) \approx x\,y\,(1 + \boldsymbol{\delta x/x + \delta y/y}).$$

L'erreur relative sur le résultat est donc approximativement égale à la somme des erreurs relatives sur les opérandes.

Cas de la division
Pour la division, on trouve, sous les mêmes hypothèses que précédemment,

$$\frac{x + \delta x}{y + \delta y} = \frac{x}{y}\left(1 + \frac{\delta x}{x}\right)\left(1 - \frac{\delta y}{y} + \left(\frac{\delta y}{y}\right)^2 - \cdots\right) \approx \frac{x}{y}\left(1 + \boldsymbol{\frac{\delta x}{x} - \frac{\delta y}{y}}\right)$$

Dans ce cas, l'erreur relative sur le résultat est de l'ordre de la différence entre les erreurs relatives sur les opérandes.

Cas de l'addition et de la soustraction
Les nombres réels x et y pouvant être positifs ou négatifs, on ne va s'intéresser qu'à l'addition. On a, en supposant que la somme $x + y$ est non nulle,

$$(x + \delta x) + (y + \delta y) = (x + y)\left(1 + \frac{x}{x+y}\frac{\delta x}{x} + \frac{y}{x+y}\frac{\delta y}{y}\right)$$

Lorsque les opérandes sont de même signe, l'erreur relative sur le résultat est majorée par :

$$max\left\{\left|\frac{\delta x}{x}\right|, \left|\frac{\delta y}{y}\right|\right\}$$

et reste donc du même ordre les erreurs relatives sur les opérandes.

En revanche, si ces derniers sont de signes opposés, au moins l'un des facteurs |x/x+ y| et |y/x+ y| est plus grand que 1 et au moins l'une des erreurs relatives sur les opérandes est amplifiée, de manière d'autant plus importante que ces opérandes sont presque égaux en valeur absolue, donnant alors lieu au phénomène d'annulation catastrophique.

3) Perte d'associativité et de distributivité

Aux problèmes déjà posés par les arrondis s'ajoute le fait que ces derniers rendent plusieurs des propriétés fondamentales de l'arithmétique « exacte » non espectées en arithmétique à virgule flottante, ce qui tend à faire de l'analyse d'erreur de calculs effectués sur une machine un travail passablement compliqué. Ainsi, si l'addition et la multiplication de nombres à virgule flottante sont bien commutatives, elles ne sont en général plus associatives. La propriété de distributivité de la multiplication par rapport à l'addition est également perdue.

IV. Analyse de sensibilité et conditionnement d'un problème

L'étude de la propagation des erreurs sur de simples opérations arithmétiques a montré que la sensibilité d'un problème à une perturbation des données peut présenter deux tendances opposées, de petites variations des données pouvant entraîner, selon les cas, de petits ou de grands changements sur la solution. Ceci nous amène à introduire un cadre d'étude général, dans lequel on identifie la résolution d'un problème donné à une application définie sur l'ensemble des données possibles pour le problème et à valeurs dans l'ensemble des solutions correspondantes.

1) Problème bien posé

Considérons le problème suivant : trouver s tel que $F(s,d) = 0$,
où F désigne une relation fonctionnelle liant la solution s à la donnée d du problème, et les variables peuvent être des nombres réels, des vecteurs ou des fonctions.
Un tel problème est dit bien posé (ou stable) si pour une donnée d fixée, une solution s existe, qu'elle est unique et qu'elle dépend continûment de la donnée d. Un problème qui ne possède pas la propriété ci-dessus est dit mal posé ou instable.

2) Conditionnement

Le conditionnement mesure l'influence des erreurs d'arrondi sur la solution d'un problème donné. Il est mis en évidence par une légère perturbation des données initiales.
Un problème numérique est dit bien conditionné s'il est mathématiquement bien posé et que la continuité de la solution est suffisamment bonne par rapport aux conditions initiales pour que la solution ne soit pas perturbée par une erreur initiale ou de petites erreurs d'arrondi.

Soient δd une perturbation admissible des données et δx la modification induite sur la solution telles que :
$F(x + \delta x, d + \delta d) = 0$.
On veut alors que
$\exists \eta_0 = \eta_0(d) > 0$, $\exists K_0 = K_0(d)$ tels que
si $\|\delta d\| \leq \eta_0$ alors $\|\delta x\| \leq K_0 \|\delta d\|$. K_0 étant le conditionnement du problème.

V. Analyse d'erreur et stabilité des méthodes numériques

Une méthode (ou un algorithme) est dite numériquement stable si de petites perturbations aux données ne génère que de petites variations aux résultats données par cette méthode. Dans le cas échéant (la solution est sujette à une propagation importante des erreurs numériques de discrétisation et d'arrondi), la méthode est dite instable.

Un problème peut être bien conditionné alors que la méthode numérique choisie pour le résoudre est instable. Dans ce cas, il est impératif de changer de méthode numérique. Par contre, si le problème de départ est mal conditionné, aucune méthode numérique ne pourra y remédier. Il faudra alors essayer de trouver une formulation mathématique différente du même problème.

VI. Techniques pratiques

Les quelques principes de programmation ci-dessous, lorsque suivis par le programmeur prudent et consciencieux, permettent d'atténuer l'impact de l'erreur d'arrondi.

- ✓ L'addition et la soustraction en virgule flottante ne sont pas associatives. Additionner les nombres en ordre croissant de grandeur afin d'accumuler des bits significatifs.
- ✓ Éviter de soustraire un petit nombre d'un grand, ou alors le faire le plus tard possible dans la chaîne des calculs.
- ✓ Pour calculer une somme alternée, additionner tous les termes positifs et tous les termes négatifs, puis faire la soustraction.
- ✓ Multiplier et diviser des nombres d'un même ordre de grandeur. Si x et y sont presque égaux, le calcul x/y sera plus précis en virgule flottante que $y^{-1}x$.
- ✓ Chercher des formules mathématiques équivalentes évitant de devoir traiter des très grands ou des très petits nombres.
- ✓ Travailler en échelle logarithmique. Par exemple, le produit de deux grands nombres x et y dépassera la capacité moins rapidement et demeurera précis plus longtemps s'il est calculé sous la forme $e^{\log x+\log y}$

Les deux derniers principes ci-dessus permettent d'éviter des débordements de capacité et d'améliorer la précision des calculs.

Conclusion

Les erreurs d'arrondi sont inévitables et parfois présentes avant même qu'une seule opération ait eu lieu, puisque la représentation en machine des données d'un problème peut nécessiter de les arrondir. Prises isolément, ces erreurs sont généralement bénignes, mais leur propagation et leur accumulation au cours d'une série de calculs, notamment lorsque l'on cherche à résoudre un problème mal conditionné et/ou que l'on utilise un algorithme numériquement instable, peuvent faire perdre toute signification au résultat numérique obtenu. Notons de plus, qu'un un algorithme numériquement instable produira de mauvaises solutions pour des problèmes bien conditionnés.