Fondements du calcul numérique

Licence 2 MPI

Université Alioune DIOP – Département de Mathématiques

25 novembre 2025

Plan du chapitre

Motivation du calcul numérique

Le calcul numérique peut être introduit de manière naturelle en partant d'un constat simple : de nombreux problèmes mathématiques admettent une solution, mais *pas de solution analytique sous forme d'une formule simple*.

Motivation du calcul numérique

Le calcul numérique peut être introduit de manière naturelle en partant d'un constat simple : de nombreux problèmes mathématiques admettent une solution, mais *pas de solution analytique sous forme d'une formule simple*. Par exemple, l'équation

$$xe^x = 3$$

ne possède aucune solution exprimable à l'aide des fonctions usuelles. Pourtant, la solution existe. Ce type de situation motive l'usage d'approximations systématiques : c'est là que le calcul numérique intervient.

Motivation du calcul numérique

Le calcul numérique peut être introduit de manière naturelle en partant d'un constat simple : de nombreux problèmes mathématiques admettent une solution, mais *pas de solution analytique sous forme d'une formule simple*. Par exemple, l'équation

$$xe^x = 3$$

ne possède aucune solution exprimable à l'aide des fonctions usuelles. Pourtant, la solution existe. Ce type de situation motive l'usage d'approximations systématiques : c'est là que le calcul numérique intervient. Par Exemple : équations non linéaires, EDO, systèmes de grande taille, etc

Motivation du calcul numérique

Ce type de situation motive l'usage d'approximations systématiques : c'est là que le calcul numérique (**méthodes numériques**) intervient.

- On utilise alors des **méthodes numériques** :

Motivation du calcul numérique

Ce type de situation motive l'usage d'approximations systématiques : c'est là que le calcul numérique (**méthodes numériques**) intervient.

- On utilise alors des **méthodes numériques** :
  - approximations, itérations, algorithmes.

Motivation du calcul numérique

Ce type de situation motive l'usage d'approximations systématiques : c'est là que le calcul numérique (**méthodes numériques**) intervient.

- On utilise alors des **méthodes numériques** :
  - approximations, itérations, algorithmes.
- Ces méthodes s'exécutent sur des ordinateurs :

Motivation du calcul numérique

Ce type de situation motive l'usage d'approximations systématiques : c'est là que le calcul numérique (**méthodes numériques**) intervient.

- On utilise alors des **méthodes numériques** :
  - approximations, itérations, algorithmes.
- Ces méthodes s'exécutent sur des ordinateurs :
  - représentation finie des nombres ;

Motivation du calcul numérique

Ce type de situation motive l'usage d'approximations systématiques : c'est là que le calcul numérique (**méthodes numériques**) intervient.

- On utilise alors des **méthodes numériques** :
  - approximations, itérations, algorithmes.
- Ces méthodes s'exécutent sur des ordinateurs :
  - représentation finie des nombres ;
  - erreurs d'arrondi inévitables.

Motivation du calcul numérique

Ce type de situation motive l'usage d'approximations systématiques : c'est là que le calcul numérique (**méthodes numériques**) intervient.

- On utilise alors des **méthodes numériques** :
  - approximations, itérations, algorithmes.
- Ces méthodes s'exécutent sur des ordinateurs :
  - représentation finie des nombres ;
  - erreurs d'arrondi inévitables.

Certaines formules, pourtant rigoureuses sur le plan analytique, souffrent de pertes de précision en machine.

Motivation du calcul numérique

Ce type de situation motive l'usage d'approximations systématiques : c'est là que le calcul numérique (**méthodes numériques**) intervient.

- On utilise alors des **méthodes numériques** :
  - approximations, itérations, algorithmes.
- Ces méthodes s'exécutent sur des ordinateurs :
  - représentation finie des nombres ;
  - erreurs d'arrondi inévitables.

Certaines formules, pourtant rigoureuses sur le plan analytique, souffrent de pertes de précision en machine. Un exemple classique est la formule quadratique :

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Motivation du calcul numérique

Ce type de situation motive l'usage d'approximations systématiques : c'est là que le calcul numérique (**méthodes numériques**) intervient.

- On utilise alors des **méthodes numériques** :
  - approximations, itérations, algorithmes.
- Ces méthodes s'exécutent sur des ordinateurs :
  - représentation finie des nombres ;
  - erreurs d'arrondi inévitables.

Certaines formules, pourtant rigoureuses sur le plan analytique, souffrent de pertes de précision en machine. Un exemple classique est la formule quadratique :

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Lorsque $b^2 \gg 4ac$, on observe une annulation catastrophique. Le calcul numérique vise alors à reformuler ou remplacer ces expressions pour obtenir un résultat stable.

Considérons l'équation quadratique :

$$x^2 + 1000x + 1 = 0.$$

Considérons l'équation quadratique :

$$x^2 + 1000x + 1 = 0.$$

Par la formule classique :

$$x = \frac{-1000 \pm \sqrt{1000^2 - 4}}{2}.$$

Considérons l'équation quadratique :

$$x^2 + 1000x + 1 = 0.$$

Par la formule classique :

$$x = \frac{-1000 \pm \sqrt{1000^2 - 4}}{2}.$$

On a :

$$\sqrt{1000000 - 4} =$$

Considérons l'équation quadratique :

$$x^2 + 1000x + 1 = 0.$$

Par la formule classique :

$$x = \frac{-1000 \pm \sqrt{1000^2 - 4}}{2}.$$

On a :

$$\sqrt{1000000 - 4} = \sqrt{999\,996}$$

Considérons l'équation quadratique :

$$x^2 + 1000x + 1 = 0.$$

Par la formule classique :

$$x = \frac{-1000 \pm \sqrt{1000^2 - 4}}{2}.$$

On a :

$$\sqrt{1000000 - 4} = \sqrt{999\,996} \approx 999.997999999.$$

Ainsi, pour la petite racine :

$$x_1 = \frac{-1000 + 999.997999999}{2},$$

Considérons l'équation quadratique :

$$x^2 + 1000x + 1 = 0.$$

Par la formule classique :

$$x = \frac{-1000 \pm \sqrt{1000^2 - 4}}{2}.$$

On a :

$$\sqrt{1000000 - 4} = \sqrt{999\,996} \approx 999.997999999.$$

Ainsi, pour la petite racine :

$$x_1 = \frac{-1000 + 999.997999999}{2},$$

ce qui implique une **soustraction entre deux nombres très proches**, menant à une **perte de précision importante** (annulation catastrophique).

Fondements du cours

Trois idées centrales :

1. **Arithmétique flottante** : comment la machine représente les réels ;

Fondements du cours

Trois idées centrales :

1. **Arithmétique flottante** : comment la machine représente les réels ;
2. **Analyse des erreurs** : absolues, relatives, propagation ;

Fondements du cours

Trois idées centrales :

1. **Arithmétique flottante** : comment la machine représente les réels ;
2. **Analyse des erreurs** : absolues, relatives, propagation ;
3. **Conditionnement et stabilité** :
   - sensibilité du problème (conditionnement) ;

Fondements du cours

Trois idées centrales :

1. **Arithmétique flottante** : comment la machine représente les réels ;
2. **Analyse des erreurs** : absolues, relatives, propagation ;
3. **Conditionnement et stabilité** :
   - sensibilité du problème (conditionnement) ;
   - comportement de l'algorithme (stabilité).

Représentation flottante (IEEE 754)

Un nombre flottant s'écrit sous la forme :

$$x = \pm(1.m_1 m_2 \dots m_p) \times 2^e,$$

Représentation flottante (IEEE 754)

Un nombre flottant s'écrit sous la forme :

$$x = \pm(1.m_1 m_2 \dots m_p) \times 2^e,$$

où :

- $m_1, \dots, m_p$: bits de la mantisse ;

Représentation flottante (IEEE 754)

Un nombre flottant s'écrit sous la forme :

$$x = \pm(1.m_1 m_2 \dots m_p) \times 2^e,$$

où :

- $m_1, \dots, m_p$: bits de la mantisse ;
- p : précision (nombre de bits) ;

Représentation flottante (IEEE 754)

Un nombre flottant s'écrit sous la forme :

$$x = \pm(1.m_1 m_2 \dots m_p) \times 2^e,$$

où :

- $m_1, \dots, m_p$: bits de la mantisse ;
- p : précision (nombre de bits) ;
- e : exposant, borné ($e_{\min} \leq e \leq e_{\max}$).

Conséquence : l'ensemble des réels représentables est **fini** et **discret**.

Epsilon machine

Définition : L'epsilon machine ε_m est la plus petite valeur positive telle que

$$1 + \varepsilon_m \neq 1$$

en arithmétique flottante.

Epsilon machine

Définition : L'epsilon machine ε_m est la plus petite valeur positive telle que

$$1 + \varepsilon_m \neq 1$$

en arithmétique flottante.

En double précision (64 bits) :

$$\varepsilon_m \approx 2^{-53} \approx 1{,}1 \times 10^{-16}.$$

Epsilon machine

Définition : L'epsilon machine ε_m est la plus petite valeur positive telle que

$$1 + \varepsilon_m \neq 1$$

en arithmétique flottante.

En double précision (64 bits) :

$$\varepsilon_m \approx 2^{-53} \approx 1{,}1 \times 10^{-16}.$$

C'est un indicateur de la **finesse** de la représentation des réels en machine.

Exemple corrigé : calcul de ε_m

Idée algorithmique :

- on part de $\varepsilon = 1$;
- on divise par 2 tant que $1 + \varepsilon/2 \neq 1$;
- la dernière valeur de ε est une approximation de ε_m.

Exemple corrigé : calcul de ε_m

Idée algorithmique :

- on part de $\varepsilon = 1$;
- on divise par 2 tant que $1 + \varepsilon/2 \neq 1$;
- la dernière valeur de ε est une approximation de ε_m.

Pseudo-code :

1. $\varepsilon \leftarrow 1$;

Exemple corrigé : calcul de ε_m

Idée algorithmique :

- on part de $\varepsilon = 1$;
- on divise par 2 tant que $1 + \varepsilon/2 \neq 1$;
- la dernière valeur de ε est une approximation de ε_m.

Pseudo-code :

1. $\varepsilon \leftarrow 1$;
2. Tant que $1 + \varepsilon/2 \neq 1$:

Exemple corrigé : calcul de ε_m

Idée algorithmique :

- on part de $\varepsilon = 1$;
- on divise par 2 tant que $1 + \varepsilon/2 \neq 1$;
- la dernière valeur de ε est une approximation de ε_m.

Pseudo-code :

1. $\varepsilon \leftarrow 1$;
2. Tant que $1 + \varepsilon/2 \neq 1$:
   $$\varepsilon \leftarrow \varepsilon/2$$
3. Retourner ε.

Exemple corrigé : calcul de ε_m

Idée algorithmique :

- on part de $\varepsilon = 1$;
- on divise par 2 tant que $1 + \varepsilon/2 \neq 1$;
- la dernière valeur de ε est une approximation de ε_m.

Pseudo-code :

1. $\varepsilon \leftarrow 1$;
2. Tant que $1 + \varepsilon/2 \neq 1$:
$$\varepsilon \leftarrow \varepsilon/2$$
3. Retourner ε.

Interprétation : $1 + \varepsilon/2 = 1$ signifie que $\varepsilon/2$ est trop petit pour affecter la mantisse.

Exemple TP : epsilon machine en Scilab

Code minimal :

```
eps = 1.0 ;
while 1.0 + eps/2 > 1.0:
    eps = eps/2 ;

 end
mprintf("Dernier eps utile : %e\n", eps_old);

mprintf("Premier eps inutile (1+eps == 1) : %e\n", eps);
```

Exemple TP : epsilon machine en Scilab

Code minimal :

```
eps = 1.0 ;
while 1.0 + eps/2 > 1.0:
    eps = eps/2 ;

 end
mprintf("Dernier eps utile : %e\n", eps_old);

mprintf("Premier eps inutile (1+eps == 1) : %e\n", eps);
```

En double précision, on obtient typiquement :

$$\varepsilon_m \approx 2.22 \times 10^{-16}$$

(selon la plateforme / implémentation).

Erreur absolue et relative

L'erreur absolue mesure *combien* on se trompe en valeur brute, tandis que l'erreur relative mesure *l'importance* de cette erreur par rapport à la valeur réelle.

Erreur absolue et relative

L'erreur absolue mesure *combien* on se trompe en valeur brute, tandis que l'erreur relative mesure *l'importance* de cette erreur par rapport à la valeur réelle.
Si la valeur vraie est x et l'approximation est $\tilde{x}$, alors :

$$E_{\text{abs}} = |\tilde{x} - x|.$$

Erreur absolue et relative

L'erreur absolue mesure *combien* on se trompe en valeur brute, tandis que l'erreur relative mesure *l'importance* de cette erreur par rapport à la valeur réelle.
Si la valeur vraie est x et l'approximation est $\tilde{x}$, alors :

$$E_{\text{abs}} = |\tilde{x} - x|.$$

C'est simplement la distance entre le vrai résultat et l'approximation.

Erreur absolue et relative

L'erreur absolue mesure *combien* on se trompe en valeur brute, tandis que l'erreur relative mesure *l'importance* de cette erreur par rapport à la valeur réelle.
Si la valeur vraie est x et l'approximation est $\tilde{x}$, alors :

$$E_{\text{abs}} = |\tilde{x} - x|.$$

C'est simplement la distance entre le vrai résultat et l'approximation.

Exemple. Longueur vraie : 20 cm ; mesure : 21 cm.

$$E_{\text{abs}} = |21 - 20| = 1 \text{ cm}.$$

L'erreur relative donne une mesure **sans dimension**, plus adaptée pour comparer des grandeurs de tailles différentes.

Erreur absolue et relative

Soient x la valeur exacte et $\tilde{x}$ une approximation.
l'**Erreur relative** compare l'erreur absolue à la taille de la valeur vraie :

$$E_{\text{rel}} = \frac{|\tilde{x} - x|}{|x|}.$$

Erreur absolue et relative

Soient x la valeur exacte et $\tilde{x}$ une approximation.
l'**Erreur relative** compare l'erreur absolue à la taille de la valeur vraie :

$$E_{\text{rel}} = \frac{|\tilde{x} - x|}{|x|}.$$

Elle indique la proportion d'erreur, souvent exprimée en pourcentage.

Erreur absolue et relative

Soient x la valeur exacte et $\tilde{x}$ une approximation.
l'**Erreur relative** compare l'erreur absolue à la taille de la valeur vraie :

$$E_{\text{rel}} = \frac{|\tilde{x} - x|}{|x|}.$$

Elle indique la proportion d'erreur, souvent exprimée en pourcentage.
L'erreur relative donne une mesure **sans dimension**, plus adaptée pour comparer des grandeurs de tailles différentes.

Exemple corrigé : approximation de π

On prend :

$$x = \pi \approx 3{,}14159265, \quad \tilde{x} = 3{,}14.$$

Exemple corrigé : approximation de π

On prend :

$$x = \pi \approx 3{,}14159265, \quad \tilde{x} = 3{,}14.$$

- Erreur absolue :

$$E_a = |3{,}14159265 - 3{,}14| \approx 0{,}00159265.$$

Exemple corrigé : approximation de π

On prend :

$$x = \pi \approx 3{,}14159265, \quad \tilde{x} = 3{,}14.$$

- Erreur absolue :

$$E_a = |3{,}14159265 - 3{,}14| \approx 0{,}00159265.$$

- Erreur relative :

$$E_r \approx \frac{0{,}00159265}{3{,}14159265} \approx 5{,}07 \times 10^{-4} = 0{,}05\%.$$

Exemple corrigé : approximation de π

On prend :

$$x = \pi \approx 3{,}14159265, \quad \tilde{x} = 3{,}14.$$

- Erreur absolue :

$$E_a = |3{,}14159265 - 3{,}14| \approx 0{,}00159265.$$

- Erreur relative :

$$E_r \approx \frac{0{,}00159265}{3{,}14159265} \approx 5{,}07 \times 10^{-4} = 0{,}05\%.$$

Lecture : l'approximation est correcte à environ 10^{-4} près, mais insuffisante pour un calcul scientifique haute précision.

Exemple corrigé : approximation de π

On prend :

$$x = \pi \approx 3{,}14159265, \quad \tilde{x} = 3{,}14.$$

- Erreur absolue :

$$E_a = |3{,}14159265 - 3{,}14| \approx 0{,}00159265.$$

- Erreur relative :

$$E_r \approx \frac{0{,}00159265}{3{,}14159265} \approx 5{,}07 \times 10^{-4} = 0{,}05\%.$$

Lecture : l'approximation est correcte à environ 10^{-4} près, mais insuffisante pour un calcul scientifique haute précision.
L'erreur absolue mesure *la différence brute*. L'erreur relative mesure *l'importance de cette différence*. Deux erreurs absolues identiques peuvent être insignifiantes ou catastrophiques selon la taille du nombre mesuré.

exemples

Pour chaque situation suivante, calculer :

1. l'erreur absolue ;
2. l'erreur relative ;
3. une brève interprétation.

exemples

Pour chaque situation suivante, calculer :

1. l'erreur absolue ;
2. l'erreur relative ;
3. une brève interprétation.

1. Température réelle : 22.5°C ; mesure : 23.1°C.

exemples

Pour chaque situation suivante, calculer :

1. l'erreur absolue ;
2. l'erreur relative ;
3. une brève interprétation.

1. Température réelle : 22.5°C ; mesure : 23.1°C.
2. Poids réel : 12.0 kg ; mesure : 11.4 kg.

exemples

Pour chaque situation suivante, calculer :

1. l'erreur absolue ;
2. l'erreur relative ;
3. une brève interprétation.

1. Température réelle : 22.5°C ; mesure : 23.1°C.
2. Poids réel : 12.0 kg ; mesure : 11.4 kg.
3. Distance réelle : 0.85 m ; estimation : 0.80 m.

1) Température.
Erreur absolue : 0.6.
Erreur relative : $0.6/22.5 \approx 0.0267$ (2.67%).
Interprétation : erreur modérée.

1) Température.
Erreur absolue : 0.6.
Erreur relative : $0.6/22.5 \approx 0.0267$ (2.67%).
Interprétation : erreur modérée.

2) Poids.
Erreur absolue : 0.6.
Erreur relative : $0.6/12.0 = 0.05$ (5%).
Interprétation : erreur notable.

1) Température.
Erreur absolue : 0.6.
Erreur relative : $0.6/22.5 \approx 0.0267$ (2.67%).
Interprétation : erreur modérée.

2) Poids.
Erreur absolue : 0.6.
Erreur relative : $0.6/12.0 = 0.05$ (5%).
Interprétation : erreur notable. **3) Distance.**

Erreur absolue : 0.05.
Erreur relative : $0.05/0.85 \approx 0.0588$ (5.9%).
Interprétation : erreur importante.

Propagation des erreurs

Pour une fonction $y = f(x)$, si x est perturbé de Δx, alors :

Propagation des erreurs

Pour une fonction $y = f(x)$, si x est perturbé de Δx, alors :

$$\Delta y = f(x + \Delta x) - f(x) \approx f'(x)\Delta x$$

lorsque Δx est petit.

Propagation des erreurs

Pour une fonction $y = f(x)$, si x est perturbé de Δx, alors :

$$\Delta y = f(x + \Delta x) - f(x) \approx f'(x)\Delta x$$

lorsque Δx est petit.

Conséquence :

- la sensibilité de y aux erreurs sur x dépend de la dérivée $f'(x)$;

Propagation des erreurs

Pour une fonction $y = f(x)$, si x est perturbé de Δx, alors :

$$\Delta y = f(x + \Delta x) - f(x) \approx f'(x)\Delta x$$

lorsque Δx est petit.

Conséquence :

- la sensibilité de y aux erreurs sur x dépend de la dérivée $f'(x)$;
- si $|f'(x)|$ est grand, de petites erreurs sur x produisent de grandes erreurs sur y.

Exemple corrigé : $f(x) = \sqrt{x}$

Soit $f(x) = \sqrt{x}$, $x = 4$ and $\Delta x = 10^{-4}$.

Exemple corrigé : $f(x) = \sqrt{x}$

Soit $f(x) = \sqrt{x}$, $x = 4$ and $\Delta x = 10^{-4}$.

Approximation linéaire :

$$f'(x) = \frac{1}{2\sqrt{x}} \Rightarrow f'(4) = \frac{1}{4}.$$

Exemple corrigé : $f(x) = \sqrt{x}$

Soit $f(x) = \sqrt{x}$, $x = 4$ and $\Delta x = 10^{-4}$.

Approximation linéaire :

$$f'(x) = \frac{1}{2\sqrt{x}} \Rightarrow f'(4) = \frac{1}{4}.$$

Donc

$$\Delta y \approx f'(4)\Delta x = \frac{1}{4} \times 10^{-4} = 2{,}5 \times 10^{-5}.$$

Exemple corrigé : $f(x) = \sqrt{x}$

Soit $f(x) = \sqrt{x}$, $x = 4$ and $\Delta x = 10^{-4}$.

Approximation linéaire :

$$f'(x) = \frac{1}{2\sqrt{x}} \Rightarrow f'(4) = \frac{1}{4}.$$

Donc

$$\Delta y \approx f'(4)\Delta x = \frac{1}{4} \times 10^{-4} = 2{,}5 \times 10^{-5}.$$

Contrôle :

$$\Delta y_{\text{exact}} = \sqrt{4 + 10^{-4}} - 2 \approx 2{,}00005 - 2 \approx 2{,}5 \times 10^{-5}.$$

L'approximation différentielles est donc très bonne.

Conditionnement : idée

Conditionnement = sensibilité de la **solution** aux perturbations des **données**.

Conditionnement : idée

Conditionnement = sensibilité de la **solution** aux perturbations des **données**.
Pour un problème $y = f(x)$:

$$\frac{\Delta y}{y} \approx \kappa(x) \frac{\Delta x}{x},$$

Conditionnement : idée

Conditionnement = sensibilité de la **solution** aux perturbations des **données**.
Pour un problème $y = f(x)$:

$$\frac{\Delta y}{y} \approx \kappa(x) \frac{\Delta x}{x},$$

où $\kappa(x)$ est le **nombre de conditionnement**.

Conditionnement : idée

Conditionnement = sensibilité de la **solution** aux perturbations des **données**.
Pour un problème $y = f(x)$:

$$\frac{\Delta y}{y} \approx \kappa(x) \frac{\Delta x}{x},$$

où $\kappa(x)$ est le **nombre de conditionnement**.

- κ petit : problème bien conditionné ;

Conditionnement : idée

Conditionnement = sensibilité de la **solution** aux perturbations des **données**.
Pour un problème $y = f(x)$:

$$\frac{\Delta y}{y} \approx \kappa(x) \frac{\Delta x}{x},$$

où $\kappa(x)$ est le **nombre de conditionnement**.

- κ petit : problème bien conditionné ;
- κ grand : problème mal conditionné.

Nombre de conditionnement (scalaire)

Pour f dérivable, on définit :

$$\kappa(x) = \left| \frac{x f'(x)}{f(x)} \right|.$$

Nombre de conditionnement (scalaire)

Pour f dérivable, on définit :

$$\kappa(x) = \left| \frac{x f'(x)}{f(x)} \right|.$$

Interprétation :

$$\left| \frac{\Delta y}{y} \right| \approx \kappa(x) \left| \frac{\Delta x}{x} \right|.$$

Nombre de conditionnement (scalaire)

Pour f dérivable, on définit :

$$\kappa(x) = \left| \frac{x f'(x)}{f(x)} \right|.$$

Interprétation :

$$\left| \frac{\Delta y}{y} \right| \approx \kappa(x) \left| \frac{\Delta x}{x} \right|.$$

- si $\kappa(x) \approx 1$: relative stabilité ;

Nombre de conditionnement (scalaire)

Pour f dérivable, on définit :

$$\kappa(x) = \left| \frac{x f'(x)}{f(x)} \right|.$$

Interprétation :

$$\left| \frac{\Delta y}{y} \right| \approx \kappa(x) \left| \frac{\Delta x}{x} \right|.$$

- si $\kappa(x) \approx 1$: relative stabilité ;
- si $\kappa(x) \gg 1$: petites erreurs sur x génèrent de grandes erreurs relatives sur y.

Exemple : $f(x) = \log(x)$

On considère $f(x) = \log(x)$.

$$f'(x) = \frac{1}{x}, \quad \kappa(x) = \left| \frac{x \cdot \frac{1}{x}}{\log(x)} \right| = \left| \frac{1}{\log(x)} \right|.$$

Exemple : $f(x) = \log(x)$

On considère $f(x) = \log(x)$.

$$f'(x) = \frac{1}{x}, \quad \kappa(x) = \left| \frac{x \cdot \frac{1}{x}}{\log(x)} \right| = \left| \frac{1}{\log(x)} \right|.$$

Comportement :

- lorsque $x \to 1^+$, $\log(x) \to 0 \Rightarrow \kappa(x) \to +\infty$;

Exemple : $f(x) = \log(x)$

On considère $f(x) = \log(x)$.

$$f'(x) = \frac{1}{x}, \quad \kappa(x) = \left| \frac{x \cdot \frac{1}{x}}{\log(x)} \right| = \left| \frac{1}{\log(x)} \right|.$$

Comportement :

- lorsque $x \to 1^+$, $\log(x) \to 0 \Rightarrow \kappa(x) \to +\infty$;
- le problème est **très mal conditionné** près de $x = 1$.

Exemple : $f(x) = \log(x)$

On considère $f(x) = \log(x)$.

$$f'(x) = \frac{1}{x}, \quad \kappa(x) = \left| \frac{x \cdot \frac{1}{x}}{\log(x)} \right| = \left| \frac{1}{\log(x)} \right|.$$

Comportement :

- lorsque $x \to 1^+$, $\log(x) \to 0 \Rightarrow \kappa(x) \to +\infty$;
- le problème est **très mal conditionné** près de $x = 1$.

Calculer $\log(x)$ à partir d'une mesure de x près de 1 est donc numériquement délicat.

Exemple corrigé : soustraction de nombres proches

Problème :

$$y = x - y.$$

Si x et y sont très proches, alors :

- la valeur exacte $x - y$ est petite ;

Exemple corrigé : soustraction de nombres proches

Problème :

$$y = x - y.$$

Si x et y sont très proches, alors :

- la valeur exacte $x - y$ est petite ;
- en machine, les bits significatifs sont partagés ;

Exemple corrigé : soustraction de nombres proches

Problème :

$$y = x - y.$$

Si x et y sont très proches, alors :

- la valeur exacte $x - y$ est petite ;
- en machine, les bits significatifs sont partagés ;
- la soustraction provoque une **perte de chiffres significatifs** (catastrophe de la soustraction).

Exemple corrigé : soustraction de nombres proches

Problème :

$$y = x - y.$$

Si x et y sont très proches, alors :

- la valeur exacte $x - y$ est petite ;
- en machine, les bits significatifs sont partagés ;
- la soustraction provoque une **perte de chiffres significatifs** (catastrophe de la soustraction).

Exemple :

$$x = 1{,}23456789, \quad y = 1{,}23456788,$$

Exemple corrigé : soustraction de nombres proches

Problème :

$$y = x - y.$$

Si x et y sont très proches, alors :

- la valeur exacte $x - y$ est petite ;
- en machine, les bits significatifs sont partagés ;
- la soustraction provoque une **perte de chiffres significatifs** (catastrophe de la soustraction).

Exemple :

$$x = 1{,}23456789, \quad y = 1{,}23456788,$$

$$x - y = 10^{-8},$$

Exemple corrigé : soustraction de nombres proches

Problème :

$$y = x - y.$$

Si x et y sont très proches, alors :

- la valeur exacte $x - y$ est petite ;
- en machine, les bits significatifs sont partagés ;
- la soustraction provoque une **perte de chiffres significatifs** (catastrophe de la soustraction).

Exemple :

$$x = 1{,}23456789, \quad y = 1{,}23456788,$$

$$x - y = 10^{-8},$$

alors que chaque terme est d'ordre 1. La moindre erreur d'arrondi sur x ou y se répercute fortement sur la différence.

Stabilité : idée

En calcul numérique, un algorithme doit produire des résultats fiables même lorsque les données ou les calculs intermédiaires comportent de petites perturbations (arrondis machine, erreurs de mesure, troncature). On introduit alors la notion de **stabilité**.

Stabilité : idée

En calcul numérique, un algorithme doit produire des résultats fiables même lorsque les données ou les calculs intermédiaires comportent de petites perturbations (arrondis machine, erreurs de mesure, troncature). On introduit alors la notion de **stabilité**.
Un algorithme est dit **stable** si une petite erreur en entrée produit une petite erreur en sortie.

Stabilité : idée

En calcul numérique, un algorithme doit produire des résultats fiables même lorsque les données ou les calculs intermédiaires comportent de petites perturbations (arrondis machine, erreurs de mesure, troncature). On introduit alors la notion de **stabilité**.
Un algorithme est dit **stable** si une petite erreur en entrée produit une petite erreur en sortie. Autrement dit, il n'amplifie pas de façon excessive les perturbations inévitables.

Stabilité : idée

En calcul numérique, un algorithme doit produire des résultats fiables même lorsque les données ou les calculs intermédiaires comportent de petites perturbations (arrondis machine, erreurs de mesure, troncature). On introduit alors la notion de **stabilité**.
Un algorithme est dit **stable** si une petite erreur en entrée produit une petite erreur en sortie. Autrement dit, il n'amplifie pas de façon excessive les perturbations inévitables.

Un algorithme est **instable** si une perturbation minuscule des données entraîne une grande variation du résultat, même si les calculs sont corrects du point de vue mathématique.

Stabilité : idée

Stabilité d'un algorithme :

- propriété de ne pas **amplifier** excessivement les erreurs d'arrondi ;

Stabilité : idée

Stabilité d'un algorithme :

- propriété de ne pas **amplifier** excessivement les erreurs d'arrondi ;
- un algorithme stable donne une solution calculée $\tilde{y}$ qui est la *solution exacte* d'un problème légèrement perturbé.

Stabilité : idée

Stabilité d'un algorithme :

- propriété de ne pas **amplifier** excessivement les erreurs d'arrondi ;
- un algorithme stable donne une solution calculée $\tilde{y}$ qui est la *solution exacte* d'un problème légèrement perturbé.

Problème bien conditionné + algorithme stable $\Rightarrow$ résultats fiables.

Stabilité : idée

Stabilité d'un algorithme :

- propriété de ne pas **amplifier** excessivement les erreurs d'arrondi ;
- un algorithme stable donne une solution calculée $\tilde{y}$ qui est la *solution exacte* d'un problème légèrement perturbé.

Problème bien conditionné + algorithme stable $\Rightarrow$ résultats fiables.
Problème mal conditionné + algorithme stable $\Rightarrow$ résultats intrinsèquement fragiles.

Exemples de stabilité / instabilité

Considérons la fonction :

$$x = \frac{1}{1-y}.$$

Exemples de stabilité / instabilité

Considérons la fonction :

$$x = \frac{1}{1-y}.$$

Si $y = 0.999999$, alors :

$$x = 1{,}000{,}000.$$

Exemples de stabilité / instabilité

Considérons la fonction :

$$x = \frac{1}{1-y}.$$

Si $y = 0.999999$, alors :

$$x = 1{,}000{,}000.$$

Mais si une erreur d'arrondi modifie légèrement y en $\tilde{y} = 0.999998$, on obtient :

$$\tilde{x} = 500{,}000.$$

Exemples de stabilité / instabilité

Considérons la fonction :

$$x = \frac{1}{1-y}.$$

Si $y = 0.999999$, alors :

$$x = 1{,}000{,}000.$$

Mais si une erreur d'arrondi modifie légèrement y en $\tilde{y} = 0.999998$, on obtient :

$$\tilde{x} = 500{,}000.$$

Ici :

- l'erreur d'entrée est infime : 10^{-6} ;

Exemples de stabilité / instabilité

Considérons la fonction :

$$x = \frac{1}{1-y}.$$

Si $y = 0.999999$, alors :

$$x = 1{,}000{,}000.$$

Mais si une erreur d'arrondi modifie légèrement y en $\tilde{y} = 0.999998$, on obtient :

$$\tilde{x} = 500{,}000.$$

Ici :

- l'erreur d'entrée est infime : 10^{-6} ;
- l'erreur de sortie est énorme : facteur 2.

Exemples de stabilité / instabilité

Considérons la fonction :

$$x = \frac{1}{1-y}.$$

Si $y = 0.999999$, alors :

$$x = 1{,}000{,}000.$$

Mais si une erreur d'arrondi modifie légèrement y en $\tilde{y} = 0.999998$, on obtient :

$$\tilde{x} = 500{,}000.$$

Ici :

- l'erreur d'entrée est infime : 10^{-6} ;
- l'erreur de sortie est énorme : facteur 2.

La formule utilisée *amplifie* les erreurs : le calcul est donc **numériquement instable**.

Exemple : sommation de termes

On considère :

$$S = 10^6 + \underbrace{0{,}001 + 0{,}001 + \cdots + 0{,}001}_{10^6 \text{ fois}}.$$

Exemple : sommation de termes

On considère :

$$S = 10^6 + \underbrace{0{,}001 + 0{,}001 + \cdots + 0{,}001}_{10^6 \text{ fois}}.$$

- Si on commence par 10^6 et qu'on ajoute les 0,001 un par un :

Exemple : sommation de termes

On considère :

$$S = 10^6 + \underbrace{0{,}001 + 0{,}001 + \cdots + 0{,}001}_{10^6 \text{ fois}}.$$

- Si on commence par 10^6 et qu'on ajoute les 0,001 un par un :
  - beaucoup de petits termes peuvent être « avalés » par l'arrondi ;

Exemple : sommation de termes

On considère :

$$S = 10^6 + \underbrace{0{,}001 + 0{,}001 + \cdots + 0{,}001}_{10^6 \text{ fois}}.$$

- Si on commence par 10^6 et qu'on ajoute les 0,001 un par un :
  - beaucoup de petits termes peuvent être « avalés » par l'arrondi ;
- Si on somme d'abord tous les 0,001, puis on ajoute 10^6 :

Exemple : sommation de termes

On considère :

$$S = 10^6 + \underbrace{0{,}001 + 0{,}001 + \cdots + 0{,}001}_{10^6 \text{ fois}}.$$

- Si on commence par 10^6 et qu'on ajoute les 0,001 un par un :
  - beaucoup de petits termes peuvent être « avalés » par l'arrondi ;
- Si on somme d'abord tous les 0,001, puis on ajoute 10^6 :

$$\sum 0{,}001 = 10^3,$$

le résultat est beaucoup plus précis.

Exemple : sommation de termes

On considère :

$$S = 10^6 + \underbrace{0{,}001 + 0{,}001 + \cdots + 0{,}001}_{10^6 \text{ fois}}.$$

- Si on commence par 10^6 et qu'on ajoute les 0,001 un par un :
  - beaucoup de petits termes peuvent être « avalés » par l'arrondi ;
- Si on somme d'abord tous les 0,001, puis on ajoute 10^6 :

$$\sum 0{,}001 = 10^3,$$

  le résultat est beaucoup plus précis.

Morale : l'ordre de sommation influe sur la stabilité numérique.

Synthèse du chapitre

Idées clés :

- les nombres réels sont représentés de manière approchée (flottants) ;
- toute opération comporte des erreurs d'arrondi ;
- le conditionnement mesure la sensibilité du problème ;
- la stabilité mesure la qualité numérique de l'algorithme ;
- une bonne pratique numérique combine :
  - problèmes bien posés ;
  - algorithmes stables ;
  - implémentations prudentes (ordre des opérations, etc.).

Merci de votre attention