Modélisation/Calculs scientifiques - GME5

Leçon 1

Algorithmique illustrée

G. Vinsard

Gerard.Vinsard@univ-lorraine.fr

19 avril 2017

Objectifs

▶ Introduire les notions et un langage d'algorithmique pour des problèmes de calcul scientifique ;

▶ Décrire les problèmes des résolution de systèmes linéaires et notamment la question du conditionnement ;

▶ Décrire les méthodes de dichotomie et du point fixe (essentiellement pour illustrer les notions d'algorithmique)

▶ Discuter de quelques langages informatiques et des moyens pour s'approprier leur syntaxe.

Système d'équations linéaires

- Un système d'équations linéaires se présente sous la forme

$$\begin{pmatrix} a_{11} & \dots & a_{1N} \\ \vdots & & \vdots \\ a_{N1} & \dots & a_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_N \end{pmatrix} \Longleftrightarrow A\,X = B$$

- Il admet une solution unique si (et seulement si) le déterminant de la matrice A est non-nul mais la recherche de cette solution par la méthode des cofacteurs n'est facile que dans le cas de petites dimensions, par exemple en dimension 2

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} \Longleftrightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{a\,d - b\,c} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

- Dans les autres cas il faut utiliser une méthode numérique.

Méthodes numériques de résolution de systèmes linéaires

- Les méthodes numériques ne cherchent jamais à : 1) calculer l'inverse A^{-1} de A ; 2) multiplier A^{-1} par B. Ce serait inefficace.

- Au lieu de quoi soit
  - elles modifient le système linéaire pour trouver la solution, comme par exemple la méthode de Gauss-Jordan (pivot maximum) ; ces méthodes sont dites directes ;
  - elles donnent une valeur X_0 *a priori* au vecteur X et elles construisent une suite X_p de la forme

$$X_p = F(X_{p-1})$$

    qui converge vers la solution du système, comme par exemple la méthode du point fixe par la diagonale ; ces méthodes sont dites itératives.

Méthodes actuelles dans leur contexte

▶ Lorsque le système linéaire n'est pas très gros ($N < 10$ voire 100), on utilise plutôt des méthodes de résolution directe : si la matrice A est symétrique, définie, positive (toutes ses valeurs propres sont strictement positives) la méthode de Cholesky ; si elle ne l'est pas, une décomposition dite $L\ U$.

▶ Lorsqu'il devient plus gros ($N < 10000$) on peut continuer à utiliser ces méthodes, mais il devient intéressant d'utiliser des méthodes itératives comme la méthode du gradient conjugué (dans le cas d'une matrice symétrique définie positive) ou gmres dans le cas contraire.

▶ Si le système devient encore plus gros ($N \approx 10^6$ voire 10^9) seule restent possible ces méthodes itératives qui sont modifiées par de véritables stratégies dépendant du type de problème traité.

Le conditionnement d'une matrice

► Le choix de la méthode de résolution peut avoir un impact sur le résultat attendu.

► Mais quelle que soit cette méthode de résolution utilisée, un problème mal posé fournira des résultats incertains.

► Un problème mal posé correspond à un très grand écart entre les valeurs propres maximum et minimum de la matrice ; c'est formalisé par le nombre de conditionnement qui peut se calculer par

$$\left(\max_{i,j} a_{ij}\right) \left(\max_{i,j} a_{ij}^{-1}\right) \text{ où } a_{ij}^{-1} \text{ sont les coef. de } A^{-1}$$

(bien entendu ce calcul n'est possible que pour les petites matrices)
Si ce nombre (sans dimension physique) est trop grand le problème est mal posé et il convient de choisir des variables différentes (ce point sera illustré dans les ateliers AMOQ).

Résolution d'équation non-linéaire (mais à une inconnue)

- Une fonction $f : \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R} \\ x & \longrightarrow & f(x) \end{array}$ est donnée et il s'agit de trouver x_∞ tel que

$$f(x_\infty) = 0$$

- C'est un problème qui n'admet pas de réponses simples : par exemple si

$$f(x) = \sin\left(\frac{1}{x}\right)$$

il n'y a pas de solution unique mais des solutions de la forme

$$x_\infty = \frac{1}{k\ \pi} \text{ pour } k \in \mathbb{Z}/\{0\}$$

- Aussi ne doit-on pas croire qu'on peut disposer de méthodes capables de fournir des solutions dans tous les cas.

Méthode de dichotomie

► Une des premières méthodes auxquelles on peut penser est celle de dichotomie : Un intervalle $[a, b]$ dans lequel doit se trouver le point x_∞ est donné et on procède ainsi :

1. on calcule $f(a)$ et $f(b)$
2. deux possibilités peuvent apparaître
   a. $f(a) > 0$ et $f(b) > 0$ ou $f(a) < 0$ et $f(b) < 0$
   b. $f(a) < 0$ et $f(b) > 0$ ou $f(a) > 0$ et $f(b) < 0$

Dans le cas a, la méthode échoue
Dans le cas b, on calcule $f((a+b)/2)$ et alors

α. si $f(a) < 0$, $f(b) > 0$, $f((a+b)/2) < 0$ alors $x_\infty \in [(a+b)/2, b]$;

β. si $f(a) < 0$, $f(b) > 0$, $f((a+b)/2) > 0$ alors $x_\infty \in [a, (a+b)/2]$;

γ. si $f(a) > 0$, $f(b) < 0$, $f((a+b)/2) > 0$ alors $x_\infty \in [(a+b)/2, b]$;

δ. si $f(a) > 0$, $f(b) < 0$, $f((a+b)/2) < 0$ alors $x_\infty \in [a, (a+b)/2]$.

Lorsque la méthode n'échoue pas elle fournit un sous-intervalle ($2 \times$ fois plus petit) de l'intervalle de départ à partir duquel il est possible de recommencer les opérations et donc la solution se trouve par encadrement.

Dichotomie en langage d'algorithme

► La description faite de la méthode de dichotomie est très verbeuse. On préfère souvent la faire dans un langage codé qui peut être appelé un langage d'algorithme. Soit :

Données : la fonction f, les bornes de l'intervalle $[a, b]$ dans lequel on cherche la solution x_∞ de l'équation $f(x_\infty) = 0$, la précision ϵ
Résultat : $\tilde{x} \in [a, b]$ tel que si x_∞ satisfait à $f(x_\infty) = 0$ alors $\tilde{x} - \epsilon \leq x_\infty \leq \tilde{x} + \epsilon$

```
début
    fa ⟵ f(a) ; fb ⟵ f(b);
    si fa = 0 alors
        x̃ ⟵ a
    sinon si fb = 0 alors
        x̃ ⟵ b
    sinon si fa × fb > 0 alors
        x̃ ⟵ "échec de l'algorithme"
    sinon
        tant que |b − a| > ε faire
            m ⟵ (a + b)/2 ; fm ⟵ f(m);
            si fm = 0 alors
                a ⟵ m ; b ⟵ m
            sinon
                si fa × fm > 0 alors
                    a ⟵ m ;fa ⟵ fm
                sinon
                    b ⟵ m ;fb ⟵ fm
        x̃ ⟵ (a + b)/2
```

Langage d'algorithme

▶ Le langage d'algorithme doit décrire les données dont il part et le résultat qu'il permet d'obtenir ; d'où les parties « Données » et « Résultat »

▶ il doit permettre de présenter une synthèse de tous les cas possibles ; pour cela il utilise des structures conditionnelles (les « si » « sinon » « alors ») ;

▶ il doit de plus expliciter les calculs de la façon la plus élémentaire possible ; c'est pour cela qu'il comporte des variables intermédaires (qui sont affecté par le symbole « ← »

▶ Il est un intermédaire à partir duquel pourra se faire le codage de l'algorithme dans un langage informatique (compréhensible par l'ordinateur via un interpréteur ou un compilateur).

Langage informatique

▶ Les langages informatiques se ressemblent tous : il sont formés d'une suite d'instruction exécutées séquentiellement avec des branchements possibles. Toutefois il y une façon particulière à chacun d'eux pour

- séparer les instructions ;
- placer un commentaire (une partie de texte lisible par l'humain et qui ne sera pas prise en compte par la machine) ;
- affecter une variable (faire que le symbole « a » puisse représenter une valeur) ;
- effectuer les structure de contrôle : les branchements conditionnels et les boucles ;
- identifier la syntaxe des variables composées (les tableaux) ; mais là il n'y en a pas.

▶ Cela va être illustré par l'examen de l'implantation de l'algorithme de dichotomie : en Fortran, en Lisp, en Javascript et bien sûr en Maxima qui sera le langage utilisé dans ce module...

Langage informatique : Fortran

```
      subroutine  f(x,fx)
      implicit real*8 (a-h,q-z)
      fx=x-0.3333
      end

      program dichotomie
      implicit real*8 (a-h,q-z)
C données
      a=0
      b=1
      e=1.0d-03
      call f(a,fa)
      call f(b,fb)
C algorithme
      i_echec=0
      if (fa.EQ.0.0d0) then
         x_tilde=a
      else if (fb.EQ.0.0d0) then
         x_tilde=b
      else if ((fa*fb).GT.0.0) then
         i_echec=1
      else
 1       if (abs(b-a).LE.e) goto 2
         xm=(a+b)/2.0d0
         write(*,*) a,b
         call f(xm,fm)
         if (fm.EQ.0) then
            a=xm
            b=xm
         else if (fa*fm.GT.0.0d0) then
            a=xm
            fa=fm
         else
            b=xm
            fb=fm
         endif
         goto 1 2       x_tilde=(a+b)/2.0d0
         write(*,*) a,b
      endif
      if (i_echec.EQ.0) then
         write(*,*) x_tilde
      else
         write(*,*)
     &'echec de l''algorithme'
      endif
      end
```

Langage informatique : Lisp (d'Emacs)

```
; LISP d'emacs -> dichotomie.el
(let (a b f e fa fb m fm x-tilde)
; données
  (defun f (x) (- x 0.3333))
  (setq a 0)
  (setq b 1)
  (setq e 1.0e-5)
```

```
; algorithme
  (setq a (float a))
  (setq b (float b))
  (setq fa (float (f a)))
  (setq fb (float (f b)))
  (cond
   ((= fa 0.0) (setq x-tilde a))
   ((= fb 0.0) (setq x-tilde b))
   ((> (* fa fb) 0.0)
    (setq x-tilde "échec de l'algorithme"))
   (t
    (while (> (abs (- b a)) e)
      (setq m (/ (+ a b) 2.0))
      (setq fm (float (f m)))
      (cond
       ((= fm 0.0) (setq a m) (setq b m))
       ((> (* fa fm) 0.0) (setq a m) (setq fa fm))
       (t (setq b m) (setq fb fm))
       )
      )
    (setq x-tilde (/ (+ a b) 2.0))
    )
   )
  x-tilde)
```

Langage informatique : Maxima

```
/* MAXIMA -> dichotomie.mc */
f(x):=x-0.3333$
a:0 $ b:1 $ e:1.0e-5 $
fa:f(a) $ fb:f(b) $
if fa=0 then  (x_tilde:a)
elseif fb=0 then (x_tilde:b)
elseif fa*fb > 0 then (x_tilde:"echec de l'algorithme")
else
(while abs(b-a)>e do
 (m:(a+b)/2,fm:f(m),
  if fm=0 then (a:m,b:m)
  elseif fa*fm > 0 then (a:m,fa:fm)
  else (b:m,fb:fm)),
 x_tilde:(a+b)/2
);
```

Langage informatique : Javascript

```
<HTML><HEAD><TITLE>Entrées/sorties en javascript</TITLE>
<script type="text/javascript">
<!-- window.onerror = mon_erreur;
function mon_erreur(nouvelle,fichier,ligne)
{document.entree.message.value = "Erreur: "+nouvelle;
 return true;}

function inite(){with(Math){x=document.entree.bi.value;
           document.entree.fbi.value=eval(document.entree.expression.value);
           x=document.entree.bs.value;
           document.entree.fbs.value=eval(document.entree.expression.value);
   document.entree.message.value="";};};

function dichotomie(){
var a = parseFloat(document.entree.bi.value);
var b = parseFloat(document.entree.bs.value);
var fa= parseFloat(document.entree.fbi.value);
var fb= parseFloat(document.entree.fbs.value);
document.entree.message.value="";
if (fa == 0.0) {b=a;fb=0.0;document.entree.message.value="la solution est trouvée"}
else if (fb == 0.0) {a=b;fa=0.0;;document.entree.message.value="la solution est trouvée"}
else if (fa*fb > 0) {document.entree.message.value="l'algorithme ne peut pas continuer";}
else {with(Math)
 {var x=(a+b)/2.0; var fx = eval(document.entree.expression.value);
  if (fx==0.0) {b=x;a=x;fa=fx;fb=fx;document.entree.message.value="la solution est trouvée"}
  else if (fx*fa > 0) {a=x;fa=fx;}
  else if (fx*fb > 0) {b=x;fb=fx;}
  else {document.entree.message.value="il doit y avoir des variables symboliques en trop"}}};
document.entree.bi.value=a;
document.entree.fbi.value=fa;
document.entree.bs.value=b;
document.entree.fbs.value=fb;};
//--></script></HEAD>
```

Choix du langage

- Quelques langages disponibles de nos jours sont :

$\approx$ dates de naissance	Langages généraux	de Logiciels dédiés
1950	Fortran	
1960	Lisp	
1970	C	reduce
1980	Pascal †	matlab, octave
1990	C++, java	maple, maxima, mathematica
2000	javascript, php,...	scilab, giac/xcas

- Le choix fait dans le cadre de l'enseignement est d'utiliser Maxima. Non parce que ce serait le $\ll$ meilleur $\gg$ mais parce qu'il présente un compromis acceptable entre la difficulté de programmation propre et l'efficacité en termes de calcul scientifique.

Méthode du point fixe

- Revenons au problème pour lequelle une fonction

$$\begin{array}{rcccc} f & : & \mathbb{R} & \longrightarrow & \mathbb{R} \\ & & x & \longrightarrow & f(x) \end{array}$$

étant donnée il s'agit de trouver x_∞ tel que

$$f(x_\infty) = 0$$

- La méthode du point fixe s'appuie sur le théorème du point fixe selon lequel une fonction

$$g \quad : \quad [a,b] \quad \longrightarrow \quad [a,b]$$

continue

$$\forall x \in [a,b], \forall \epsilon > 0, \exists \nu > 0 \text{ tel que } |x' - x| < \nu \Longrightarrow |g(x) - g(x')| < \epsilon$$

et contractante

$$\exists L < 1 \text{ tel que } \forall (x,x') \in [a,b]^2, |g(x) - g(x')| < L|x - x'|$$

admet un point fixe

$$x_\infty \in [a,b] \text{ tel que } g(x_\infty) = x_\infty$$

unique.

Algorithme du point fixe

- L'algorithme du point fixe est

Données : la fonction g, un point initial x_0, on cherche la solution x_∞ de l'équation $g(x_\infty) = x_\infty$, la précision ϵ
Résultat : x_N tel que $|g(x_N) - x_N| \leq \epsilon$
début
  $gx_0 \longleftarrow g(x_0)$;
  tant que $|gx_0 - x_0| > \epsilon$ **faire**
    $x_0 \longleftarrow gx_0$; $gx_0 \longleftarrow g(x_0)$

- Il ne résout pas l'équation

$$f(x_\infty) = 0 \text{ mais l'équation } g(x_\infty) = x_\infty$$

Il est donc nécessaire de l'adapter au problème de la recherche de zéros.

Algorithme de résolution d'équation par point fixe

- On peut poser par exemple

$$g(x) = x - f(x)$$

et l'adaptation est

Données : la fonction f, un point initial x_0, on cherche la solution x_∞ de l'équation $f(x_\infty) = 0$, la précision ϵ
Résultat : x_N tel que $|f(x_N)| \leq \epsilon$
début
  $fx_0 \longleftarrow f(x_0)$;
  tant que $|fx_0| > \epsilon$ **faire**
    $x_0 \longleftarrow x_0 - fx_0$, $fx_0 \longleftarrow f(x_0)$

- Mais on aurait tout aussi bien pu choisir $g(x) = x + f(x)^2$ ou encore d'autres formes qui auraient conduit à d'autres adaptations de l'algorithme du point fixe.

Le groupement des algorithmes en fonctions

► Pour ne pas multiplier les variantes de, par exemple l'algorithme du point fixe, tout en laissant la possibilité de multiplicité des choix intacte, on définit des fonctions dans lesquelles sont rangés les algorithmes.

► Ici on définit la *fonction* ≪ recherche de point fixe ≫

---

recherche_de_points_fixes(fonction,point_initial,precision) **début**

$x_0 \longleftarrow$ *point_initial* ; $gx_0 \longleftarrow$ *fonction*(x_0);

tant que $|gx_0 - x_0| > precision$ **faire**

$x_0 \longleftarrow gx_0$; $gx_0 \longleftarrow$ *fonction*(x_0)

Retourner x_0

► Et cette fonction peut être appelée par tout autre algorithme comme

---

Données : la fonction f, un point initial x_0, on cherche la solution x_∞ de l'équation $f(x) = 0$, la précision ϵ

Résultat : x_N tel que $|f(x_N)| \leq \epsilon$

début

Définir $g : \mathbb{R} \longrightarrow \mathbb{R}$; $x \longrightarrow g(x) = x - f(x)$

Retourner *recherche_de_points_fixes*$(g, x0, \epsilon)$

Évaluation des méthodes

► Les méthodes ont un champ d'application où elles sont efficaces ; elles ne le sont pas voire même sont défectueuses en dehors.

► Par exemple la méthode de dichotomie suppose qu'il existe une et une seule racine dans l'intervalle où elle est cherchée ; s'il y en a plusieurs l'une d'entre elle peut être trouvée ($cos(3\ \pi\ x)$ dans $[0, 1]$) ou la méthode peut échouer ($cos(2\ \pi\ x)$ dans $[0, 1]$).

► De la même façon, la méthode du point fixe suppose que la fonction g ne soit pas plus croissante que x, elle ne trouve pas le zéro de $2\ x - 1$.

► D'autres méthodes seront expliquées à la leçon No 2 mais elles souffriront toujours d'un inconvénient de cette nature.