Résolution numérique de l'équation de Poisson et application à l'étude de la déformation d'une membrane

Travaux pratiques du cours de méthodes numériques, ENSIMAG 1ère année

4 Avril 2008

Résumé

Ces travaux pratiques concernent la résolution de l'équation de Poisson en deux dimensions, sur un domaine rectangulaire, avec des conditions aux limites de Dirichlet homogènes. Le problème discrétisé par la méthode des différences finies conduit à la résolution d'un système linéaire de grande taille. La matrice du système est à la fois creuse et symétrique définie positive. Deux méthodes vues en cours sont donc bien adaptées à sa résolution, l'une itérative (méthode de relaxation) et l'autre directe (méthode de Cholesky). Dans ce TP vous allez programmer la méthode des différences finies et la résolution du système linéaire en utilisant ces deux méthodes, et comparer leur efficacité sur cet exemple.

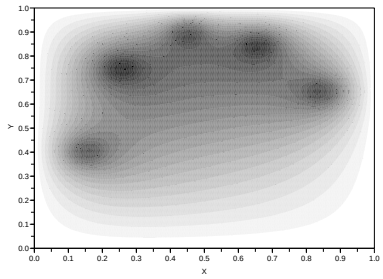

FIG. 1 – Déformation d'une membrane rectangulaire sous l'effet de la pression de cinq doigts. Le modèle utilisé est une équation de Poisson pour le déplacement de la membrane. La déformation est indiquée en niveaux de gris, le blanc correspondant à un déplacement presque nul et le noir à un déplacement maximal. Le calcul numérique est effectué sur un maillage de 400×400 points.

1 Informations pratiques

BUT du TP : Un élément central du cours de méthodes numériques est la résolution de grands systèmes linéaires issus de la modélisation de problèmes physiques. Bien qu'il soit crucial d'étudier théoriquement les méthodes les plus classiques pour connaître notamment leurs conditions de convergence, il est tout aussi important de savoir les utilliser correctement.

Le but de ce TP est de vous faire toucher du doigt les difficultés pouvant intervenir dans l'implémentation d'une méthode numérique et dans l'analyse des résultats obtenus. Le travail sera effectué en binôme uniquement. Ce TP sera réalisé avec l'aide de SCILAB, logiciel librement distribué et largement utilisé. La rédaction d'un rapport vous demandera d'être précis et clair tout en restant concis.

Contenu du rapport : Vous devez rédiger un compte-rendu de TP dans lequel vous répondrez à toutes les questions de l'énoncé, expliciterez les méthodes employées, présenterez et commenterez les résultats obtenus. Il n'y a qu'un seul rapport à rendre par binôme. La qualité de la rédaction, de la synthèse, de l'analyse des résultats obtenus sont des critères importants pour la note. Notez que ce sujet ne constitue que la base de ce qui vous est demandé : soyez critique par rapport à vos résultats, proposez d'autres idées, solutions ou tests. La dernière page de votre compte-rendu devra être une sorte de manuel d'utilisation où vous expliquerez comment utiliser vos programmes. Le compte-rendu sera dactylographié. Nous conseillons fortement d'utiliser le logiciel LaTeX, qui est un outil extrêmement utilisé pour la rédaction d'articles scientifiques. Ce compte-rendu n'excèdera pas 10 pages et ne comportera pas de programmes. Les programmes Scilab nous seront envoyés par email ; la lisibilité du code et la pertinence des commentaires seront pris en compte dans la note du TP.

Remise du rapport et des programmes :
Le TP est à rendre **au plus tard le lundi 11 Mai 2009, à 17h00**. Il faudra :
* déposer votre compte-rendu imprimé dans le casier prévu à cet effet,
* envoyer un fichier .pdf de votre compte-rendu ainsi que vos fichiers `Scilab` à l'adresse *Adrien.Magni@imag.fr*, avec copie à *tpmnensimag@yahoo.fr*.

Conseils et contacts : Une séance de questions-réponses concernant le TP est prévue durant les cours d'amphi des 20 et 21 Avril. Plus généralement n'hésitez pas à demander des conseils à votre enseignant à la fin de chaque cours. Par ailleurs, pour toute question, précision sur le TP vous pouvez à tout moment vous adresser à Adrien.Magni@imag.fr. Evitez de faire le TP au dernier moment : ce travail demande du temps, et lancer trop de calculs simultanément risque de saturer les machines au mauvais moment...

2 Description du problème

2.1 Equation de Poisson pour la déformation d'une membrane

On considère une membrane élastique horizontale de forme rectangulaire. On met la membrane sous tension en tirant sur ses bords et on exerce sur elle une force transversale. La membrane à l'équilibre correspond à une surface $z = u(x, y)$, où u est solution d'une équation de Poisson bidimensionnelle

$$-\Delta u = f, \quad (x, y) \in]0, 1[\times]0, 1[, \tag{1}$$

avec $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$. Le second membre $f(x, y)$ est la pression exercée sur la surface divisée par la tension de la membrane. On complète (1) par les conditions aux limites

$$u(0, y) = u(1, y) = u(x, 0) = u(x, 1) = 0 \tag{2}$$

(conditions aux limites de Dirichlet homogènes), qui signifient que le bord de la membrane est maintenu fixe. Lorsque $f = 0$ on a simplement $u = 0$ et la membrane est horizontale (elle correspond alors au carré $[0, 1] \times [0, 1]$). On souhaite ici étudier la déformation de la

membrane sous l'action des forces transverses. On peut montrer que le problème aux limites (1)-(2) admet une solution unique lorsque f est continue sur $[0,1]\times[0,1]$, et nous allons calculer numériquement cette solution.

Remarque : le modèle (1) est en fait valable pour de petites déformations de la membrane. Notez aussi que l'équation de Poisson apparaît dans beaucoup d'autres contextes : conduction de la chaleur, diffusion d'espèces chimiques, électromagnétisme, mécanique des fluides...

2.2 Approximation par différences finies

On considère une discrétisation du domaine $[0,1]\times[0,1]$ suivant une grille régulière formée par des points de coordonnées (x_i, y_j), avec $x_i = i\,h$, $y_j = j\,h$, $h = 1/(N+1)$ et $0 \le i, j \le N+1$. On note $u_{i,j}$ une approximation de $u(x_i, y_j)$, obtenue par un schéma aux différences finies que nous allons décrire. On fixe $u_{0,j} = u_{N+1,j} = u_{i,0} = u_{i,N+1} = 0$ (conditions aux limites (2)).

Lorsque u est de classe C^4 on a

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \frac{1}{h^2}\left(u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)\right) + O(h^2)$$

(utiliser la formule de Taylor comme dans le 1er chapitre du cours). De même

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \frac{1}{h^2}\left(u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})\right) + O(h^2).$$

L'équation (1) donne donc en (x_i, y_j)

$$-\frac{1}{h^2}\left(u(x_{i+1}, y_j) + u(x_{i-1}, y_j) + u(x_i, y_{j+1}) + u(x_i, y_{j-1}) - 4u(x_i, y_j)\right) = f(x_i, y_j) + O(h^2). \quad (3)$$

On remplace donc (1)-(2) par le problème approché

$$-\frac{1}{h^2}\left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}\right) = f(x_i, y_j), \quad 1 \le i, j \le N, \quad (4)$$

$$u_{0,j} = u_{N+1,j} = u_{i,0} = u_{i,N+1} = 0, \quad 0 \le i, j \le N+1. \quad (5)$$

On définit le vecteur

$$v = (u_{1,1}, u_{2,1}, \ldots, u_{N,1}, u_{1,2}, u_{2,2}, \ldots, u_{N,2}, \ldots, u_{1,N}, u_{2,N}, \ldots, u_{N,N})^t \quad (6)$$

contenant les valeurs approchées de u à calculer. On note de même $f_{i,j} = f(x_i, y_j)$ et

$$b = h^2\,(f_{1,1}, f_{2,1}, \ldots, f_{N,1}, f_{1,2}, f_{2,2}, \ldots, f_{N,2}, \ldots, f_{1,N}, f_{2,N}, \ldots, f_{N,N})^t. \quad (7)$$

Le problème (4)-(5) s'écrit sous la forme

$$A\,v = b \quad (8)$$

avec $A \in M_{N^2}(\mathbb{R})$. La matrice A s'écrit par blocs de taille N

$$A = \begin{pmatrix} S & -I & 0 & \cdots & 0 \\ -I & S & -I & 0 & \vdots \\ 0 & -I & \ddots & \ddots & 0 \\ \vdots & 0 & \ddots & S & -I \\ 0 & \cdots & 0 & -I & S \end{pmatrix},$$

où I désigne la matrice identité d'ordre N et $S \in M_N(\mathbb{R})$ s'écrit

$$S = \begin{pmatrix} 4 & -1 & 0 & \cdots & 0 \\ -1 & 4 & -1 & 0 & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & 0 & \ddots & 4 & -1 \\ 0 & \cdots & 0 & -1 & 4 \end{pmatrix}.$$

On peut vérifier que A admet comme valeurs propres

$$\lambda_{k,l} = 4(\sin^2 \frac{k\pi h}{2} + \sin^2 \frac{l\pi h}{2}), \quad 1 \leq k, l \leq N,$$

avec des vecteurs propres correspondant à

$$u_{i,j}^{(k,l)} = \sin(k\pi hi)\, \sin(l\pi hj).$$

La matrice A est donc inversible et symétrique définie positive.

Pour calculer numériquement la solution du système (8), nous allons utiliser la méthode de Cholesky et la méthode SOR (méthode de relaxation). Cette dernière est intéressante ici car la matrice A est creuse (ses coefficients non nuls sont localisés sur 5 diagonales).

2.3 Complément sur la méthode SOR

On pose $A = M - N$ avec $M = \frac{D}{\omega} + L$ et $N = \frac{1-\omega}{\omega} D - U$, les matrices D, L, U étant les parties diagonale, triangulaire inférieure et triangulaire supérieure de A définies en cours, et ω le paramètre de relaxation. Lorsque $\omega \in]0, 2[$, pour toute condition initiale x_0 la suite $(x_k)_{k\geq 0}$ définie par

$$Mx_{k+1} = Nx_k + b \tag{9}$$

converge vers la solution de (8) lorsque $k \to +\infty$. La condition $\omega \in]0, 2[$ est nécessaire et suffisante pour la convergence de la méthode car A est symétrique définie positive. La question est maintenant de connaître la valeur optimale de ω, i.e. celle donnant la convergence la plus rapide de la méthode. Cette valeur de ω est celle qui minimise le rayon spectral de la matrice $\mathcal{L}_\omega = M^{-1}N$.

Pour la matrice A particulière intervenant dans (8), les valeurs propres de $\mathcal{L}_\omega$ sont connues explicitement. En effet, $\mathcal{L}_\omega z = \lambda z$ équivaut à

$$-Uz - \lambda Lz = \frac{1}{\omega}(\lambda + \omega - 1)Dz,$$

soit en notant comme précédemment $z_{i,j}$ $(1 \leq i, j \leq N)$ les composantes du vecteur $z \in \mathbb{R}^{N^2}$

$$z_{i+1,j} + \lambda z_{i-1,j} + z_{i,j+1} + \lambda z_{i,j-1} = \frac{4}{\omega}(\lambda + \omega - 1)z_{i,j}, \quad 1 \leq i, j \leq N,$$

$$z_{0,j} = z_{N+1,j} = z_{i,0} = z_{i,N+1} = 0, \quad 0 \leq i, j \leq N + 1.$$

Lorsque λ est une valeur propre non nulle, le calcul de λ et $z_{i,j}$ se simplifie grâce au changement de variable $z_{i,j} = \lambda^{(i+j)/2} w_{i,j}$. En effet on obtient

$$w_{i+1,j} + w_{i-1,j} + w_{i,j+1} + w_{i,j-1} = \frac{4}{\sqrt{\lambda}\,\omega}(\lambda + \omega - 1)w_{i,j}, \quad 1 \leq i, j \leq N,$$

$$w_{0,j} = w_{N+1,j} = w_{i,0} = w_{i,N+1} = 0, \quad 0 \leq i, j \leq N+1,$$

autrement dit $4\left[1 - \frac{1}{\sqrt{\lambda}\,\omega}\left(\lambda + \omega - 1\right)\right]$ est valeur propre de A. Les valeurs propres de A étant connues explicitement, on obtient explicitement $\sqrt{\lambda}$ en résolvant une équation du second degré, puis on en déduit λ. Cela permet de déterminer la valeur propre λ de plus grand module, et donc le rayon spectral de $\mathcal{L}_\omega$. On trouve alors que $\rho(\mathcal{L}_\omega)$ est minimal lorsque

$$\omega = \frac{2}{1 + \sin\left(\pi h\right)}. \tag{10}$$

3 Questions

3.1 Implémentation de la méthode de relaxation (SOR)

Question 1 *Ecrire une fonction qui prend comme arguments ω, le second membre b de (9), un vecteur x_k, et calcule la solution x_{k+1} de (9). Ce programme ne doit pas stocker les matrices A, M ou N qui deviennent rapidement trop volumineuses lorsque h est petit. On utilise à la place la formule donnant explicitement les composantes de x_{k+1} en fonction des arguments ω, b, x_k et des coefficients non nuls de A (il y a au maximum 5 coefficients non nuls sur chaque ligne de A).*

Par la suite on fixe $x_0 = 0$ comme condition initiale de (9).
On désigne par $\|\ \|$ la norme $\|\ \|_\infty$ sur $\mathbb{R}^{N^2}$.

Question 2 *On note $r_k = b - Ax_k$ (ce vecteur est appelée le résidu de la méthode itérative). Ecrire un programme qui calcule x_k jusqu'à ce que*

$$\frac{\|r_k\|}{\|b\|} < \epsilon \quad ou \quad k \geq k_{max}. \tag{11}$$

Les paramètres du programme sont le pas de discrétisation h, le second membre b de (8), la tolérance ϵ sur la norme relative du résidu, le nombre d'itérations maximal k_{max} et le paramètre de relaxation ω. Le programme affiche en sortie le nombre d'itérations réalisées et la norme relative du dernier résidu.

Question 3 *On se propose de tester le programme précédent sur un exemple. On choisit $h = 10^{-2}$ et on fixe ω avec (10) (valeur optimale théorique). On définit*

$$u(x, y) = 16\, x\, y\, (1 - x)\, (1 - y) \tag{12}$$

et $u_{i,j} = u(x_i, x_j)$. On considère le vecteur v défini par (6) et $b = A\, v$. Calculer l'erreur relative

$$\frac{\|v - x_k\|}{\|v\|} \tag{13}$$

à la fin de l'exécution de programme précédent, lorsque $\epsilon = 10^{-4}$ et $k_{max} = 1000$.

Question 4 *Montrer que l'erreur relative (13) décroît exponentiellement avec k (indication : tracer le graphe de l'erreur en fonction de k en échelle logarithmique). Donner une approximation de la vitesse de convergence.*

Question 5 *On suppose que la solution u de (1) est de classe C^4 sur $[0,1]\times[0,1]$ et on note $\tilde{u}_{i,j} = u(x_i, x_j)$. On note $\tilde{v}$ le vecteur défini de manière analogue à (6) en remplaçant $u_{i,j}$ par $\tilde{u}_{i,j}$, contenant les valeurs de la solution exacte u aux points (x_i, x_j). On admet le résultat de stabilité*

$$\frac{1}{h^2} A\, w = c \;\Rightarrow\; \|w\| \leq \kappa \, \|c\|, \quad \kappa > 0 \text{ indépendant de } h, \tag{14}$$

et le résultat de convergence quand $h \to 0$

$$\|\tilde{v} - v\| = O(h^2). \tag{15}$$

Montrer que prendre $\epsilon = h^2$ dans (11) garantit que $\|\tilde{v} - x_k\| = O(h^2)$.

Question 6 *Reprendre le calcul de la question 3 avec $h = 1/65$, $\epsilon = 10^{-7}$, $k_{max} = 1000$, pour différentes valeurs de ω dans l'intervalle $[1.7, 2[$. Tracer le graphe du nombre d'itérations effectuées en sortie du programme en fonction de ω. Pour quelle valeur de ω réalise-t-on le moins d'itérations ? Comparer cette valeur à la valeur théorique (10).*

Question 7 *On propose de tester deux critères d'arrêt en remplacement de (11). Le premier est*

$$\frac{\|x_k - x_{k-1}\|}{\|x_k\|} < \epsilon \quad ou \quad k \geq k_{max} \tag{16}$$

et le second

$$0 \leq \frac{\|x_k - x_{k-1}\|^2}{\|x_{k-1} - x_{k-2}\| - \|x_k - x_{k-1}\|} < \epsilon \quad ou \quad k \geq k_{max} \tag{17}$$

(ce dernier critère s'obtient sous l'hypothèse $\lim_{k\to+\infty} \frac{\|x_k - x_{k-1}\|}{\|x_{k-1} - x_{k-2}\|} = \lambda \in]0,1[$). Comparer l'efficacité des critères (11), (16) et (17) sur des exemples de votre choix.

3.2 Méthode de Cholesky

Dans cette partie du TP nous allons résoudre le système (8) en utilisant la méthode de Cholesky. Il ne s'agira pas de programmer la méthode elle-même (c'est à dire la factorisation $A = T\,T^t$ et la résolution de systèmes triangulaires), mais d'utiliser des fonctions préprogrammées effectuant les calculs (voir plus bas). Le but de ces questions est de vous faire connaître différents outils (matrices creuses, factorisation de Cholesky,...) et de vous habituer à utiliser l'help de Scilab.

Question 8 *Ecrire un programme définissant la matrice A sous forme de matrice creuse (cela permet d'économiser espace mémoire et temps de calcul). On pourra utiliser les fonctions : speye, spzeros, l'opérateur colon (exemple : $A(:,i)$, ou $A(:,2:N)$), sparse.*

Question 9 *Ecrire un programme qui calcule la solution v de (8) en utilisant la méthode de Cholesky. Le programme utilise la matrice creuse A définie dans la question précédente et les fonctions chfact et chsolve disponibles sous Scilab. Les paramètres du programme sont le pas de discrétisation h et le second membre b de (8). Le programme affiche en sortie la norme relative du résidu $\frac{\|b - A\,v\|}{\|b\|}$.*

Question 10 *Tester votre programme sur l'exemple (12) et comparer les performances des méthodes de relaxation et de Cholesky. On pourra comparer la précision du résultat, le temps de calcul CPU (fonction timer), l'espace mémoire utilisé (fonctions stacksize, who).*

3.3 Calcul de la déformation d'une membrane

Dans cette partie vous pouvez utiliser la méthode de votre choix (SOR ou Cholesky) pour résoudre l'équation de Poisson qui détermine la déformation d'une membrane. Il s'agit de tester la convergence de la méthode des différences finies sur un exemple où la solution de (1) est explicite, puis d'étudier un exemple plus complexe.

Question 11 *On définit*

$$u(x,y) = 4x(1-x)y(1-y)(1 - cos(2q\pi x) - cos(2q\pi y) + cos(2q\pi x)cos(2q\pi y)), \qquad (18)$$

q étant un entier ≥ 1. Calculer $f = -\Delta u$. Tracer les graphes des fonctions u et f pour $q = 1$ et $q = 6$ (voir help plot3d).

Question 12 *On considère le cas $q = 1$ de la question précédente. Pour différentes valeurs de h, résoudre numériquement le système (8) avec second membre b défini par (7). Tracer le graphe de l'erreur relative $\frac{\|\tilde{v}-v\|}{\|\tilde{v}\|}$ en fonction de h (on rappelle que $\tilde{v}$ contient les valeurs de la solution exacte u aux noeuds) et retrouver numériquement le résultat de convergence (15). On utilisera une échelle logarithmique pour tracer le graphe de l'erreur (voir help plot2d).*

Question 13 *Dans l'exemple des questions 11 et 12, comment le choix de h est-il lié à q si on veut obtenir u avec une bonne précision ?*

Question 14 *On modélise la pression exercée par un doigt sur la membrane en un point de coordonnées (x_0, y_0) par*

$$f(x,y) = \begin{cases} -\dfrac{1}{r^2} \text{ si } \|(x,y)-(x_0,y_0)\|_2 < r \\ 0 \text{ sinon.} \end{cases}, \quad r = 0.05,$$

Calculer le déplacement u de la membrane sous l'effet de la pression des 5 doigts d'une main. On représentera le graphe de u (fonction plot3d) ainsi que ses valeurs en niveaux de gris comme dans la figure 1 (fonctions grayplot, Sgrayplot). Préciser les valeurs des paramètres choisis dans vos calculs (position des doigts, pas de discrétisation h). Remarque : si nécessaire, l'espace mémoire utilisable pour les calculs peut être augmenté avec la fonction stacksize.

A Rappel de quelques commandes utiles en SCILAB

Après avoir lancé Scilab, vous pouvez tester les commandes suivantes :
`help` pour ouvrir l'aide en scilab
`help mot-clé` pour obtenir la description de la fonction mot-clé
`apropos mot-clé` pour obtenir la liste des pages d'aide contenant `mot-clé`

Les commandes `clear`, `clc` et `clf` permettent d'effacer respectivement les données mises en mémoire, l'écran de commandes et les figures. Elles doivent être exécutées régulièrement pour éviter les erreurs et libérer la mémoire.

A.1 Exécuter sous Scilab

Les commandes Scilab peuvent être tapées directement en ligne, par exemple :
$-->$ x=1
$-->$ A=ones(3,4) ;
$-->$ x+A
ou bien, écrites dans un fichier de commmandes "***.sce**". Dans ce cas,

1. Ouvrir un fichier intitulé par exemple **test.sce**, comportant les instructions suivantes :
   //(Symbole commentaire) Programme test.sce
   ```
   clc ;clf ;clear ;
   A=ones(3,4)
   1+A
   ```
2. Sous scilab, tapez : $-->$ exec('test.sce')

On peut également définir des *fichiers de fonctions* nommés "***.sci**". Pour cela,

1. Ouvrir un fichier intitulé par exemple **carre.sci**, comportant les instructions suivantes :
   //Fonction carre.sci
   ```
   function d = carre(x)
   d= x.*x
   endfunction
   ```
2. Sous scilab, chargez et compilez le fichier carre.sci :
   $-->$ getf('carre.sci') //si le fichier est dans le répertoire courant.

La fonction `carre` est maintenant définie sous scilab :
$-->$ x=[0,1,2,3,4]
$-->$ carre(x)

Un calcul trop long peut etre arrêté en cliquant sur la fenetre `control` puis `abort` et `stop`.

A.2 Vecteurs et matrices

La façon la plus simple de définir une matrice $n \times m$ en Scilab est d'entrer au clavier la liste de ses éléments :

$$A = [a_{1,1}, \ldots, a_{1,m}; \ldots; a_{n,1}, \ldots, a_{n,m}]$$

Opérations élémentaires A tester sur des exemples !

```
-->  A+B    //somme
-->  A*B    //produit
-->  A.*B   //produit terme à terme
-->  A^ 2   //équivalent à A*A
-->  A.^ 2  //équivalent à A.*A
-->  det(A) //déterminant de A
-->  A'     //transposée de A
-->  inv(A) //inverse de A
```

A.3 Fonctions échantillonnées

Une fonction peut être définie par rapport à une discrétisation de la variable x, ainsi :
`x=0 :0.1 :1 ;`
correspond à une discrétisation par pas de 0.1, de x=0 à x=1, soit 11 valeurs. On définit des fonctions sur cette grille discrète, par exemple :

```
y = sin(2 * %pi * x) + cos(%pi * x)  //Somme de deux sinusoïdes
z = x.^2                             //parabole
```

A.4 Tracé de courbes

Pour tracer une courbe y=f(x) sur l'intervalle $[a, b]$:

```
n = ...          //nombre de points de discrétisation
dx=(b-a)/(n-1)   //pas de la discrétisation
x=[a :dx :b] ;   //x est échantillonné entre a et b avec un pas de dx
plot2d(x,f(x))
```

A noter : on pourra également utiliser la commande `x=linspace(a,b,n)` pour définir la grille discrète.

Pour rajouter un titre

```
xtitle('Graphe de la fonction sin')
```

Pour mettre une légende

```
x=0 :0.1 :10
plot2d(x,exp(x),leg="exp")
```

Pour tracer plusieurs graphes dans une fenêtre

La commande `subplot(m,n,p)` placée avant chaque tracé de courbe, subdivise la fenêtre du graphe en une matrice $m \times n$ de sous-fenêtres et sélectionne la p-ième pour dessiner le graphe courant : l'élément (i,j) de la matrice correspond au graphe numéro (i-1)*m + j.

Pour dessiner dans une fenêtre graphique

Tapez `scf(1)` par exemple pour que la fenêtre graphique courante devienne la fenêtre 1. Si celle-ci n'existe pas, elle est créée.

Pour superposer deux courbes

```
plot(x,[f(x'),g(x')])    //même discrétisation
```

Pour exporter une figure

Dans la fenêtre graphique à exporter, cliquer sur le menu *File*, puis *Export*. Dans la fenêtre qui s'ouvre alors :

1. choisir l'extension du fichier image : pour mettre vos figures en latex, choisir *Postcript* ("**.ps**").
2. choisir entre *color* et *Black&White*
3. choisir l'orientation. Attention, par défaut, l'image est en landscape (format paysage) ! !
4. Entrer le nom du fichier image : *myfig* par exemple, sans l'extension.