MDI224

Interpolation par splines cubiques

Travaux Pratique 1 - Deuxième semestre de 2011

PROFESSEUR: ROLAND BADEAU

TIAGO CHEDRAOUI SILVA   CASIER: 214

Décembre 15, 2011

Table des matières

1 Résolution du système linéaire

1.1 Méthode de Jacobi

1.1.1 Implémentation

Pour voir la méthode de Jacobi, on a fait le code suivant :

Méthode de Jacobi

```
% % % % % % % % % % % % % % % % % % %
% xx/12/11
% Chedraoui Silva,Tiago
% Casier: 214
% TP1: interpolation par splines
    cubiques
% Description: Methode jacobi
% pour resoudre un systeme lineaire
% % % % % % % % % % % % % % % % % % %

function x = jacobi(A,b,x0,eps,maxit)

% Entree
% A : matrice
% b : vecteur
% x0: vecteur d'initialisation
% esp: critere de convergence
% maxit: nombre maximal d'iterations

% A in [N X N]
N = size(A);

% Iniatialization sortie
x = x0;

% Decomposition de A:
% A = M - K
% M = D
% K = L + U

% Pour rappeler:
% ----------
% | d -u -u|
% |-l  d -u|
% |-l -l  d|
% ----------
% Donc:
D = diag(diag(A));
K = A-D;

for i=1:maxit,

  xn=D\(b-K*x(:,i));

  x=[x xn];

  % sauvegarder les valeurs pour faire
  % le plot log(erreur) X iteres
  err=norm( x(:,i+1) - x(:,i) );

  % si l'erreur d'approximation est
      plus petite
  % que eps on doit arreter
  if (err <= eps)
    break;
  end;

  % sinon je doit repeter l'iteration
      jusqu'a convergence

end;
```

Pour voir le convergence de la méthode de Jacobi, on a pris, pour chaque itération, chaque valeur de x calculé jusqu'au moment que méthode arrête. Après, on a fait la comparaison de chaque x avec le valeur optimal (xex) en prenant le log de la différence :

1.1.2 Convergence

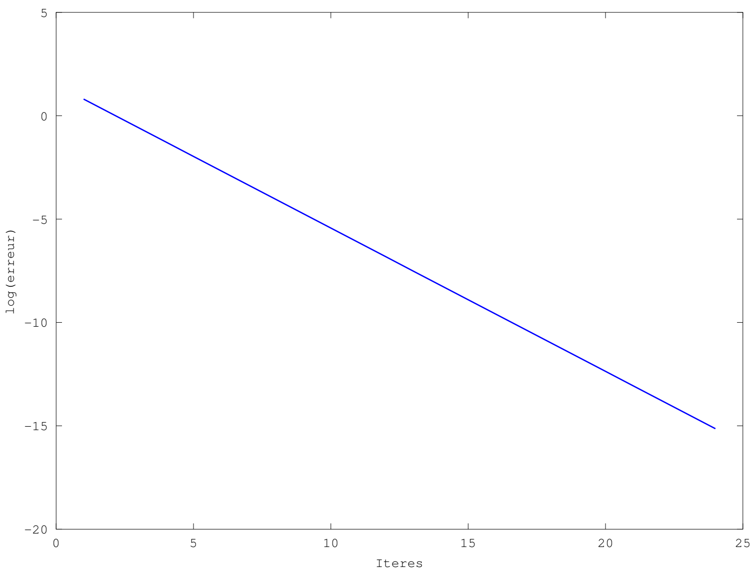

FIG. 1: Convergence de la méthode de Jacobi

En utilisant la fonction polyfit, nous avons trouvé le polynome : $p(x) = -0.693147x + 1.497866$

1.2 Méthode de relaxation

Pour voir la méthode SOR (relaxation), on a fait le code suivant :

1.2.1 Implémentation

Méthode de Jacobi

```
% % % % % % % % % % % % % % % % % % %
% 12/12/11
% Chedraoui Silva,Tiago
% Casier: 214
% TP1: interpolation par splines
    cubiques
% Description: Methode relaxation
% pour resoudre un systeme lineaire
% % % % % % % % % % % % % % % % % % %

function [x,rho] = relax(A,b,x0,w,eps,
    maxit)

% Entree
% A : matrice
% b : vecteur
```

```matlab
% x0: vecteur d'initialisation
% w : parametre de relaxation
% esp: critere de convergence
% maxit: nombre maximal d'iterations

% A in [N X N]
N = size(A);

% Iniatialization sortie
x = x0;

% Decomposition de A:
% A = M - K
% M = D/w - L
% K = (1-w)D/w + U

% Pour rappeler:
% ----------
% | d -u -u|
% |-l  d -u|
% |-l -l  d|
% ----------

% Donc:
D = diag(diag(A));
U = (-1)*triu(A,1);
L = (-1)*tril(A,-1);

% Obs: Le deuxieme valeur de triu et tril
% est utilise por mettre des zeros au
% lieu de la diagonal principal

M = D/w - L;
K = (1.0-w)*D/w + U;

Rw = M\((1.0-w)*D/w + U);
rho = max(abs(eig(Rw)));

for i=1:maxit,

  xn = M\((K*x(:,i))+b);

  x = [x xn];

  % sauvegarder les valeurs pour faire
  % le plot log(erreur) X iteres
  err=norm( x(:,i+1) - x(:,i) );

  % si l'erreur d'approximation est plus petite
  % que eps on doit arreter
  if (err <= eps)
    break;
  end;

  % sinon je doit repeter l'iteration jusqu'a convergence

end;
```

Pour voir le convergence de la méthode SOR, on a pris, pour chaque itération, chaque valeur de x calculé jusqu'au moment que méthode arrête (pour w=1.0). Après, on a fait la comparaison de chaque x avec le valeur optimal (xex) en prenant le log de la différence :

1.2.2 Convergence

Pour voir le taux, il existe deux méthodes, le premier, il utilise les erreurs et la fonction polyfit de matlab. Le deuxième il calcule le rayon spectral de la matrice R_w. Ainsi, on change les valeurs de w de manière a trouve le meieller taux de convergence. Les graphes pour les deux approches sont vue dans les graphes en 3 et 4 bas. On a trouve un $w_{optmal} \approx 1.1$

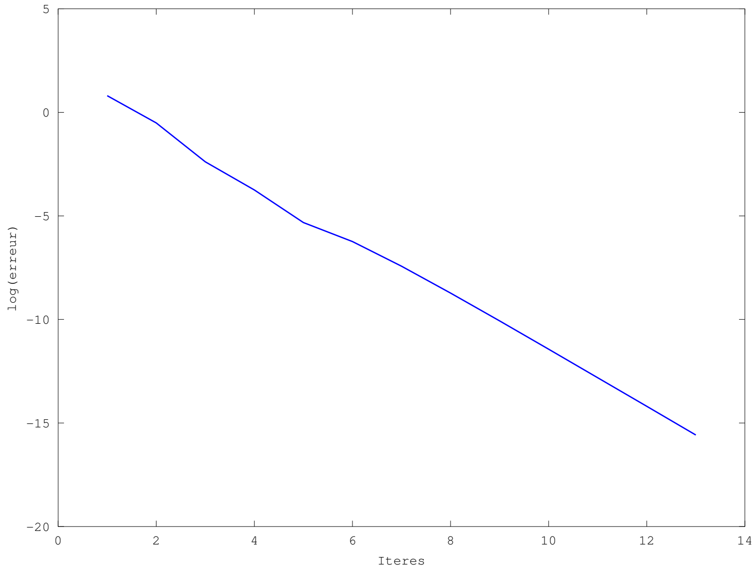

FIG. 2: Convergence de la méthode de Relaxation pour w=1.0

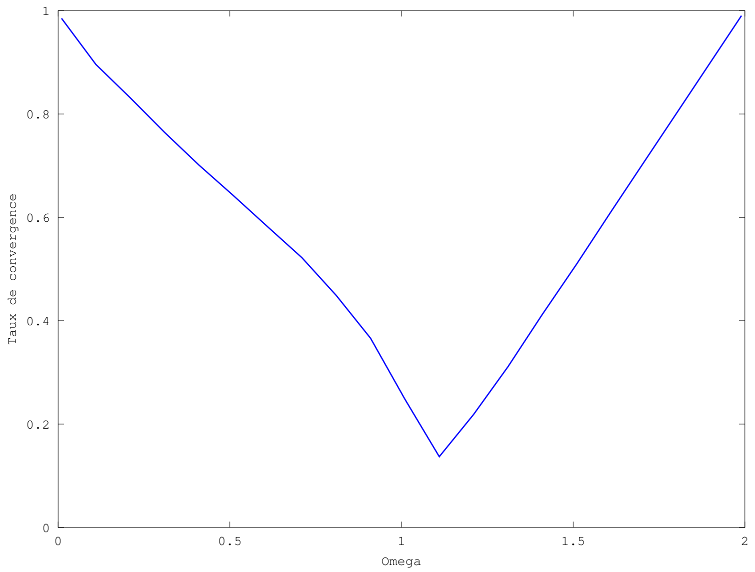

FIG. 3: Approche 1 :Taux de convergence en fonction du paramètre de relaxation w

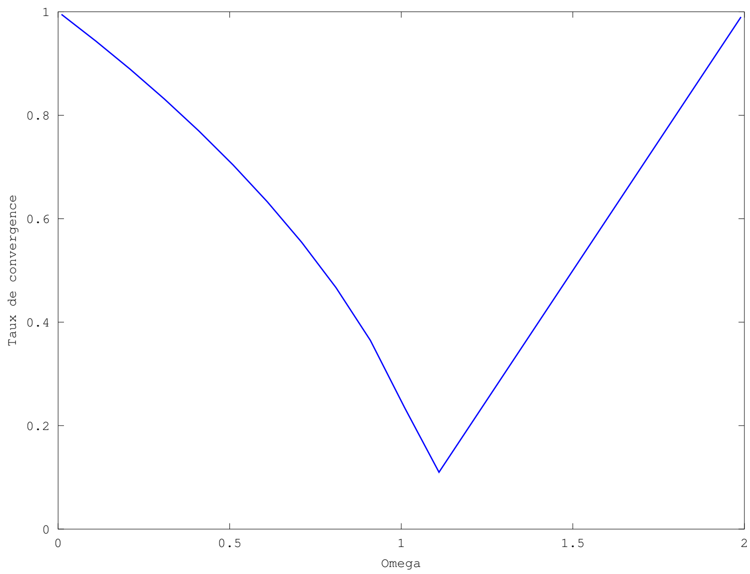

FIG. 4: Approche 2 : Calcule du rayon spectral de la matrice R_w

1.3 Méthode de Cholesky

1.3.1 Implémentation

La fonction suivante était fournit par le problème.

Méthode de Jacobi

```
function x = cholesky(A,p,b)

N = size(A,1);
L = zeros(N);
for i=1:N,
    L(i,i) = sqrt(A(i,i) - sum(L(i,max(1,i-p):i-1).^2));
    for j=i+1:min(i+p,N);
        L(j,i) = (A(i,j) - sum(L(i,max(1,j-p):i-1).*L(j,max(1,j-p):i-1))) / L(i,i);
    end;
end;
x = L'\(L\b);
```

Pour le même exemple précedent, on a trouve :

1.3.2 Complexité

En général le méthode Relaxation (SOR) est la méthode plus rapide pour un système tridiagonal, parce que il fait moins itération que Jacobi, peut-être le code n'est pas optimal c'est pour cela que le temps pour N grand est grand.

p	x
4	[1.00000 ;1.00000 ;1.00000 ;1.00000 ;1.00000]
3	[1.00000 ;1.00000 ;1.00000 ;1.00000 ;1.00000]
2	[1.00000 ;1.00000 ;1.00000 ;1.00000 ;1.00000]
1	[1.00000 ;1.00000 ;1.00000 ;1.00000 ;1.00000]

TAB. 1: Vérification function Cholesky

N	50	100	150	200	250	300
Cholesky ($p = 1$)	0.011554	0.023093	0.036068	0.047571	0.061026	0.074059
Jacobi	0.0026100	0.0031020	0.0041549	0.0048039	0.0078550	0.0097679
Relaxation ($w = 1.1$)	0.0020840	0.0025171	0.0036389	0.0065949	0.0083960	0.018082

TAB. 2: Comparaison de temps entre les trois méthodes pour un système tridiagonal

2 Application

2.1 Calcul d'une spline d'interpolation

Méthode spline cubique

```
% % % % % % % % % % % % % % % % % % %
% 12/12/11
% Chedraoui Silva,Tiago
% Casier: 214
% TP1: interpolation par splines cubiques
% Description: Calculer la spline cubique
% d'interpolation
% % % % % % % % % % % % % % % % % % %

function sp = sinterp(y)

% Entree
% y : vecteur

% y in [N X 1]
N = size(y,1);

% Sortie
sp =  zeros(N);

disp(N);
disp(sp);

% Iniatialization matrice A (Ax=b)
A = zeros(N,N);
A(1,1)=2; A(N,N)=2;
A(1,2)=1; A(N,N-1)=1;

for i=2:N-1,
  A(i,i-1)=1;
  A(i,i)=4;
  A(i,i+1)=1;
end;

% Iniatialization vecteur b (Ax=b)
b = zeros(N,1);

% cas donne par C3
b(N)=y(N)-y(N-1);
b(1)=y(2)-y(1);

for i=2:N-1,
  b(i)=y(i+1)-y(i-1);
end;

% Soit h egal a 1
h=1;
b=(3/h)*b;

sp=A\(b);

end;
```

2.2 Évaluation d'une fonction spline cubique

Méthode spline cubique

```matlab
% % % % % % % % % % % % % % % % % % %
% 12/12/11
% Chedraoui Silva,Tiago
% Casier: 214
% TP1: interpolation par splines cubiques
% Description: evaluer une fonction spline
% cubique aux points donnes par x
% % % % % % % % % % % % % % % % % % %

function y = speval(a,b,s,sp,x,h)

% Entree
% [a,b] : valeurs intervale
% s : valeurs splines
% sp :valeurs derive premier
% x :vecteur de points a value la fonction
% h: pas interaction

% Sortie

% x in [N X 1]
N = size(x,2);
N2 = size(s,2);

% Sortie
y =  zeros(N,1);

% Theorique: Poly page 28/43
% P(i)= diff1*L+diff2*M
% diff1= (t-t_i+1)^2/(h*h)
% diff2= (t-t_i)^2/(h*h)
% L = (s_i+(s_i' + 2 s_i/h)(t-t_i))
% M = (s_{i+1}+(s_{i+1}' - 2 s_{i+1}/h)(t-t_{i+1}))

for j=1:N,

  % Quel points sont les plus proches duquel je voudrais calculer?
  % t1 et t2 vont donner le index du vecteur
  % t11 et t22 vont donner les valeur temporel plus proche
  t1=1;
  t2=2;
  t11=a;
  t22=a+h;

  for k=1:N2-1,
    if(x(j)>(b-k*h))
      t1=N2-k;
      t2=N2-k+1;
      t11=b-k*h;
      t22=b+h*(1-k);
      break;
    end;
  end;

  diff1 = ((x(j)-t22)^2)/(h*h);
  diff2 = ((x(j)-t11)^2)/(h*h);
  L = (s(t1))+((sp(t1)+2*s(t1)/h)*(x(j)-t11));
  M = (s(t2))+((sp(t2)-2*s(t2)/h)*(x(j)-t22));

  y(j)= diff1*L+diff2*M;

end;
```

2.3 Application

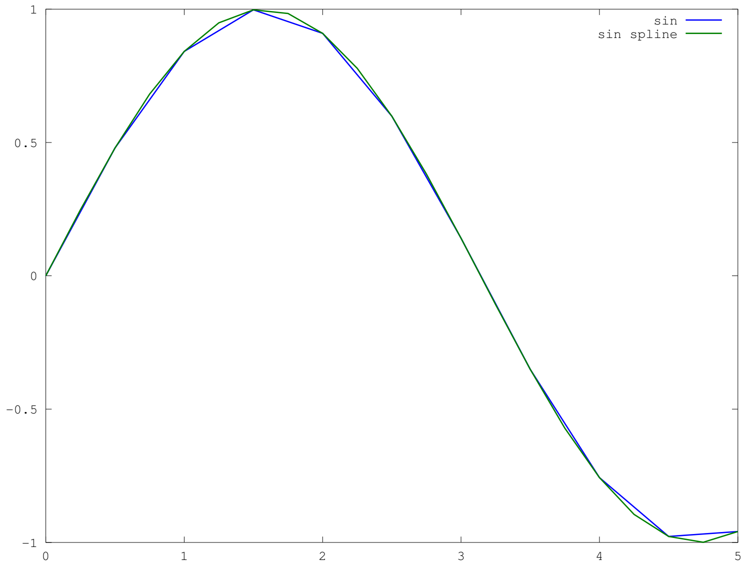

FIG. 5: Spline Cubique - fonction sinus pas original 0.5, pas spline 0.25

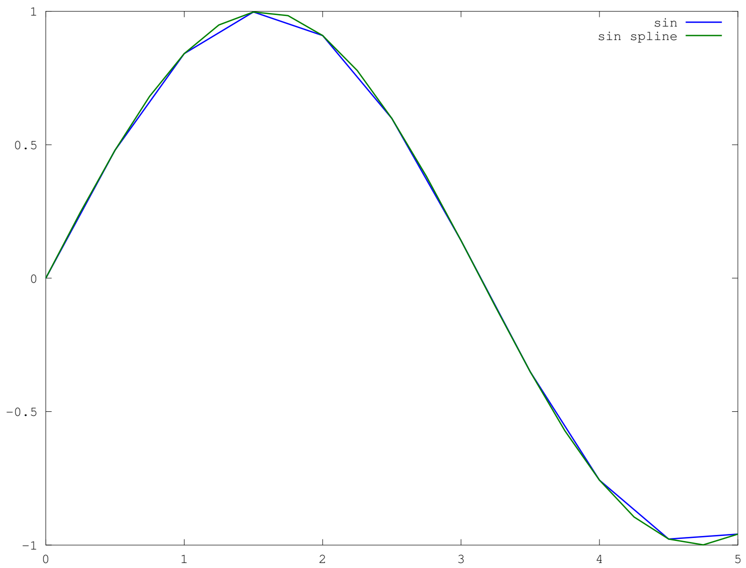

FIG. 6: Spline Cubique - fonction sinus pas original 0.5, pas spline 0.1

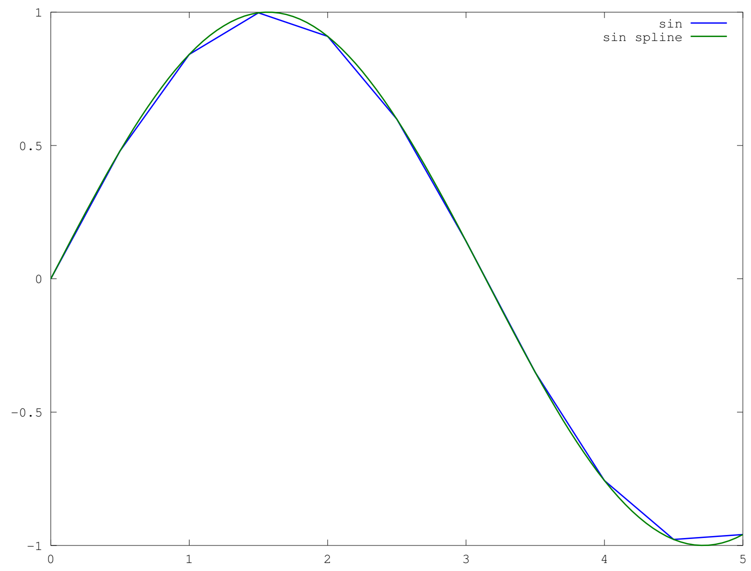

FIG. 7: Spline Cubique - fonction sinus pas original 0.5, pas spline 0.05

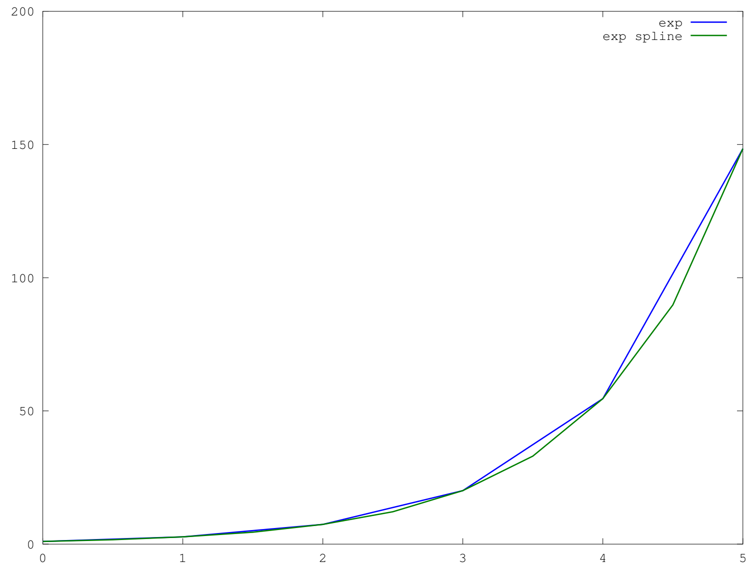

FIG. 8: Spline Cubique - fonction exp pas original 1, pas spline 0.5

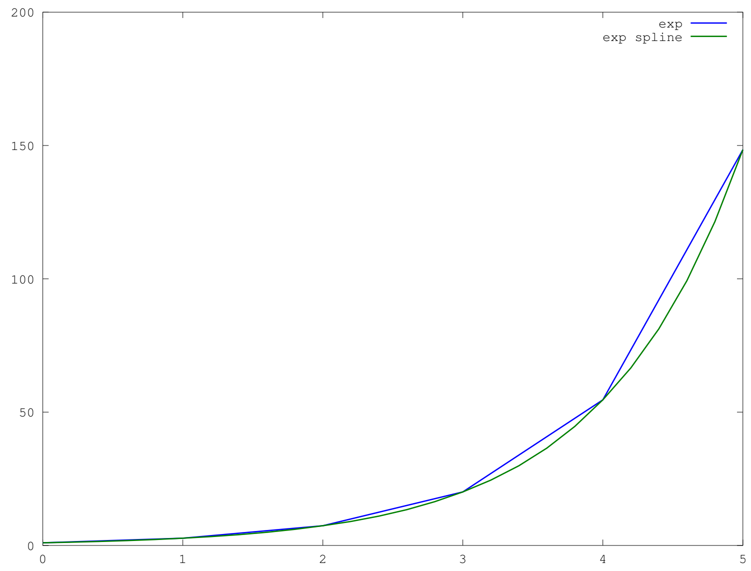

FIG. 9: Spline Cubique - fonction exp pas original 1, pas spline 0.02

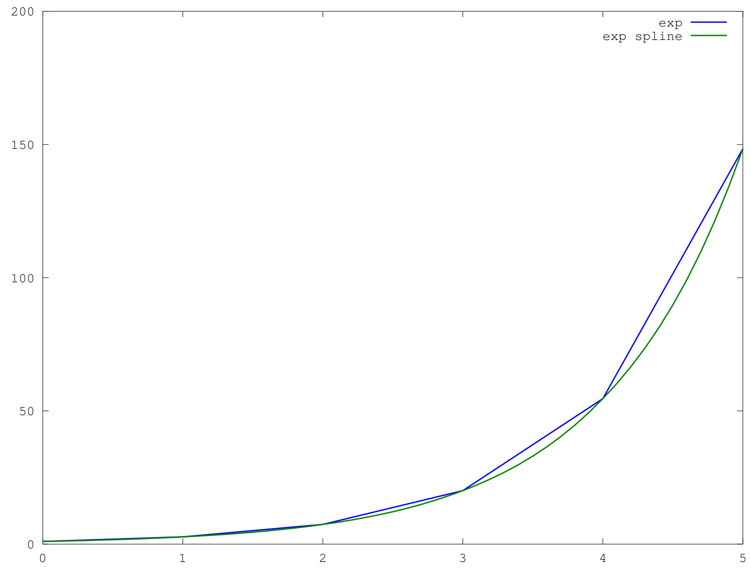

FIG. 10: Spline Cubique - fonction exp pas original 1, pas spline 0.1