MINFO0402

COMPTE RENDU DE TP2

Quentin JUILLIARD
Corentin MACHET
Licence 2e Année, Informatique – URCA

SOMMAIRE

I. PREAMBULE 2
II. MATRICES TRIDIAGONALES 2

1. HYPOTHESES 2
2. FONCTION RESOUTRI 2
3. FONCTION PRODMATTIR 3
4. RESULTATS 4

III. METHODE MAGIQUE 5

1. HYPOTHESES 5
2. FONCTION MAGIQUE 5
3. RESULTATS 6

IV. METHODE D'APPROXIMATION 12

1. HYPOTHESES 12
2. FONCTIONS A DISPOSITION 12
3. DETAILS DU SCRIPT 12
4. RESULTATS 13

I. PREAMBULE

Ce travail pratique tend à inculquer les manipulations algorithmiques de base via le langage Scilab. Il cherche à mettre en évidence les aspects mathématiques implicites des opérateurs disponibles nativement au sein de l'interpréteur. Ici, nous nous proposons d'étudier les résolutions matricielles pour les matrices tridiagonales, la méthode de résolution dite « magique » et l'application des concepts vu lors des TP 1 et 2 (modélisation graphique d'une approximation de fonction grâce aux équations différentielles).

L'exercice 1 reprend les consignes de l'exercice 6 du premier TP.

A l'issue de ce travail, vous disposerez des fichiers suivants :

- A_MACHET_CORENTIN.txt et A_JUILLIARD_QUENTIN.txt
- ALIRE.txt (détail des fichiers de l'archive)
- Fonction.sci, Exercice1.sce, Exercice2.sce et Exercice3.sce (fonctions et scripts des programmes)
- Le présent compte rendu

Les fichiers sources feront l'objet de commentaires.

NB : L'exercice 3 sera divisé en deux étapes successives comme stipulé dans le sujet : la première pour les fonctions Ua et Fa et la seconde pour les fonctions Ub et Fb.

II. MATRICES TRIDIAGONALES

1. HYPOTHESES

Il s'agit tout d'abord de résoudre la matrice tridiagonale de l'exercice 1. Nous supposons donc l'existence d'une matrice M dont la représentation de ses diagonales se fait par les vecteurs A, B et C – par soucis de place en mémoire et de rapidité d'exécution.

A et C sont des vecteurs de *n* valeurs -1 ; B un vecteur de *n* valeurs 2. On dispose aussi d'un vecteur D de *n* valeurs 1, et d'un vecteur X à *n* inconnus, tel que X soit la solution de l'équation $MX=D$.

Nous cherchons à déterminer une fonction pour résoudre l'équation précédente, puis nous calculerons le temps moyen de résolution et la marge d'erreur moyenne que nous afficherons en fonction de la taille de la matrice M de départ.

2. FONCTION RESOUTRI

En nous basant sur les indications de l'énoncé, nous calculons d'abord des valeurs intermédiaires que nous conservons dans les vecteurs E et F.

Il suffit ensuite de déterminer les valeurs de X par « remontée ». La suite numérique est définie comme suit :

$$\begin{cases} x_i = e_i * x_{i+1} + f_i \\ x_n = f_n \end{cases}$$

```
8  //·Routine·permettant·de·calculer·le·coef·e[i]·du·vecteur·E·pour·a[i],b[i],c[i],e[i-1]·don
   nés
1  function·res·=·calcE(a,b,c,e)
2  ····res·=·-c/(a*e+b)
3  endfunction
12
13 //·Routine·permettant·de·calculer·le·coef·f[i]·du·vecteur·F·pour·a[i],b[i],c[i],e[i-1],f[i
   -1]·donnés
1  function·res·=·calcF(a,b,d,e,f)
2  ····res·=·(d-a*f)/(a*e+b)
3  endfunction
17
18 //·Résoud·X·pour·la·matrice·tridiagonale·M·composée·des·vecteur·A,B,C·suivant·l'équation·M
   X=D
1  function·X·=·RESOUTRI(A,B,C,D,n)
2  ····E·=·zeros(n,1)·//·on·initialise·E,·F·et·X·des·vecteurs·nuls
3  ····F·=·zeros(n,1)
4  ····X·=·zeros(n,1)
5  ····//·Comme·la·résolution·suit·un·principe·de·récurrence,·on·calcule·d'abord·les·premiers
   ·termes·donnés·de·chq·vecteur
6  ····E(1)·=·-C(1)/B(1)
7  ····F(1)·=·D(1)/B(1)
8  ····for·i·=·2:n-1
9  ········E(i)·=·calcE(A(i),B(i),C(i),E(i-1))
10 ········F(i)·=·calcF(A(i),B(i),D(i),E(i-1),F(i-1))
11 ····end
12 ····F(n)·=·calcF(A(n),B(n),D(n),E(n-1),F(n-1))
13 ····X(n)·=·F(n)·//·d'après·la·déclaration·de·la·suite,·cf·énoncé
14 ····for·i·=·n-1:-1:1
15 ········//·Il·ne·reste·plus·qu'a·calculer·la·soluction·de·l'équation·grace·à·E·et·F·calcul
   és·précédemment
16 ········X(i)·=·E(i)*X(i+1)+F(i)
17 ····end
18 endfunction
```

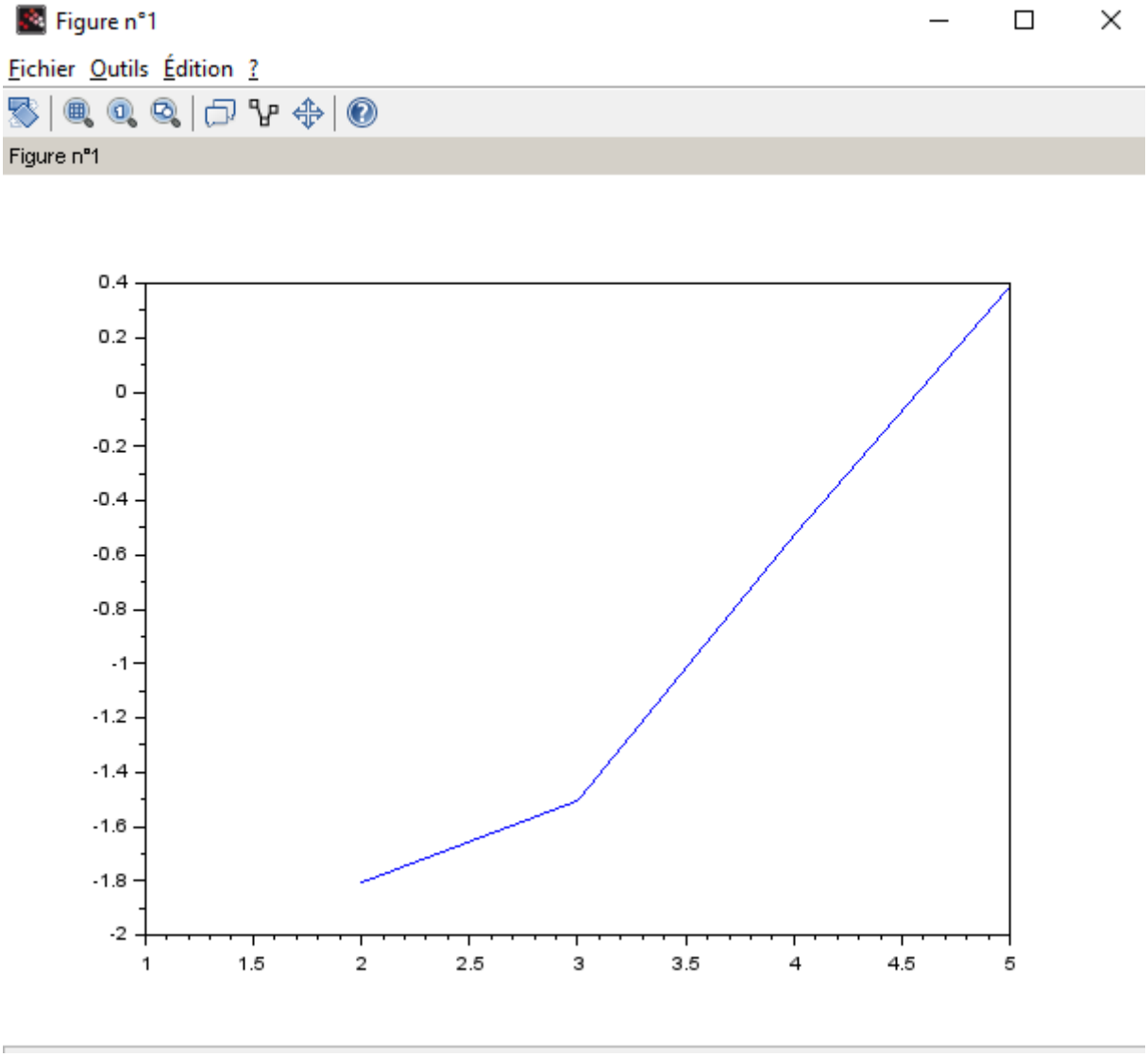
3. FONCTION PRODMATTIR

Pour pouvoir vérifier que la résolution est correcte, puis pour calculer la marge entre la valeur théorique et la valeur calculée par le logiciel, il nous faut d'abord être en mesure de faire le produit de M (qui est en fait A, B et C) et de X.

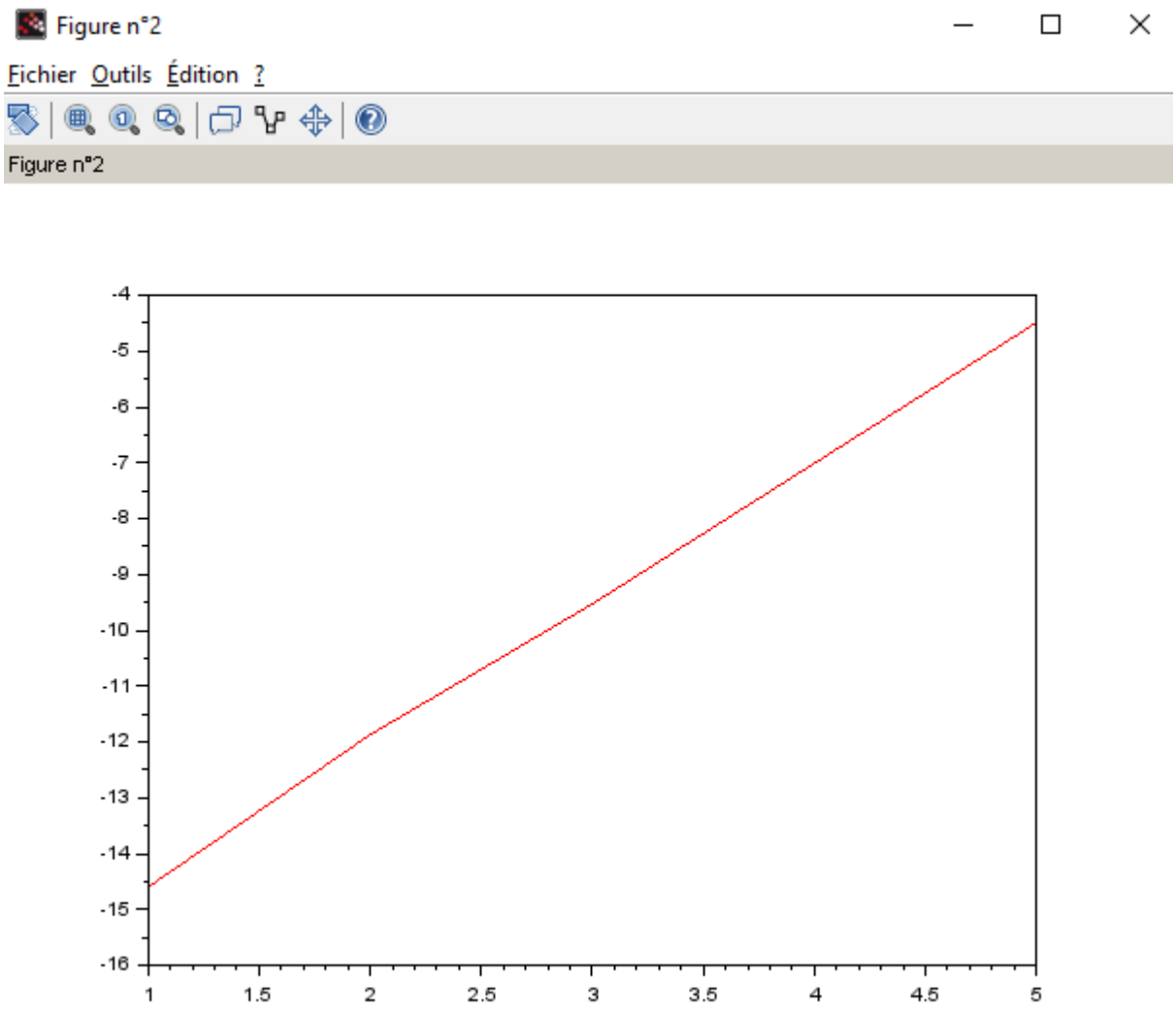
D'où la fonction suivante qui calcule MX :

```
38 //·Résoud·le·produit·matriciel·MX·où·M·est·une·matrice·tridiagonale
1  function·Y·=·PRODMATTRI(A,B,C,X,n)
2  ····Y·=·zeros(n,1)·//·initialisation·
3  ····Y(1)·=·B(1)*X(1)·+·C(1)*X(2)·//·calcul·du·premier·terme
4  ····for·i·=·2:n-1
5  ·····Y(i)·=·A(i)*X(i-1)·+·B(i)*X(i)·+·C(i)*X(i+1)·//·résolution·par·récurrence
6  ····end
7  ····Y(n)·=·A(n)*X(n-1)·+·B(n)*X(n)·//·calcul·du·dernier·terme
8  endfunction
47
```

4. RESULTATS



Logarithme décimal du temps d'exécution en fonction du logarithme décimal de n



Logarithme décimal de la différence (erreur) en fonction du logarithme décimal de n

III. METHODE MAGIQUE

1. HYPOTHESES

Il s'agit à présent d'inverser une matrice grâce à la méthode dite « magique ». On admet uniquement que A est inversible sans nécessité de permuter les lignes de celle-ci.

2. FONCTION MAGIQUE

A chaque itération, nous fixons la valeur pivot (i.e. les coefficient diagonaux). Nous traitons, pour la colonne du pivot, les coefficients qui doivent s'annuler, en appliquant ligne par ligne la combinaison linéaire suivante :

$$L_i \leftarrow L_i - (c_a / p) * L_p$$

Evidemment, on ne traitement pas la ligne du pivot.

Lorsqu'on obtient une matrice diagonale à gauche, il ne reste plus qu'à diviser par les coefficients diagonaux de celle-ci, afin d'obtenir la matrice identité à gauche et la matrice inverse à droite.

```
50 // Résolution matrice inverse (en supposant que A est inversible), grace à la méthode magi
   que Gauss-Jordan
1  function X = MAGIQUE(A,n)
2      X = eye(n,n) // init X une matrice identité de taille n
3      for k = 1:n // k est l'index du pivot
4          for i = 1:n // i est l'indice des lignes
5              if i <> k // on traite chaque ligne sauf celle du pivot
6                  tmp = A(i,k)/A(k,k) // on garde temporairement le facteur pour ne pas le p
   erdre à mesure que l'on modifie les valeurs de la matrice A
7                  for j = 1:n // j est l'indice des colonnes
8                      A(i,j) = A(i,j) - A(k,j)*tmp
9                      X(i,j) = X(i,j) - X(k,j)*tmp
10                 end
11                 disp([A,X]) // on affiche la grande matrice (n,2n)
12             end
13         end
14     end
15     for i = 1:n
16         for j = 1:n
17             X(i,j) = X(i,j)/A(i,i) // enfin on divise les lignes par les coef qu'il reste
   dans la matrice de gauche pour obtenir la matrice identité à gauche et la matrice inverse
   à droite
18         end
19     end
20 endfunction
```

3. RESULTATS

```
exec("Exercice2.sce")
--> // Exercice 2
--> pathname = get_absolute_file_path("Exercice2.sce");
--> exec(pathname+'\Fonction.sci',-1);
--> n = 5; // taille de la matrice
--> min = 0; // borne inférieure de l'intervalle de valeurs aléatoires
--> max = 10; // borne supérieure de l'intervalle de valeurs aléatoires
--> // Matrice carrée initiale
--> A = rand(n, n)*(max-min)+min
 A  =

  3.3217189   6.3257449   2.639556    2.1190304   8.4155184
  5.9350947   4.051954    4.1481037   1.1213547   4.0620248
  5.0153416   9.1847078   2.806498    6.856896    4.0948255
  4.3685876   0.4373343   1.2800585   1.5312167   8.7841258
  2.6931248   4.8185089   7.7831286   6.9708506   1.1383597
```

Pour tester notre méthode magique, on initialise A, une matrice de taille 5x5 à valeurs entre 0 et 10.

```
--> // Calcul de l'inverse de A
--> C = MAGIQUE(A,n);

       column 1 to 5
  3.3217189   6.3257449   2.639556    2.1190304   8.4155184
  0.         -7.2505962  -0.5681337  -2.6648316  -10.974437
  5.0153416   9.1847078   2.806498    6.856896    4.0948255
  4.3685876   0.4373343   1.2800585   1.5312167   8.7841258
  2.6931248   4.8185089   7.7831286   6.9708506   1.1383597
```

Puis on exécute notre fonction.

```
    column 6 to 10
 1.        0.   0.   0.   0.
-1.786754   1.   0.   0.   0.
 0.        0.   1.   0.   0.
 0.        0.   0.   1.   0.
 0.        0.   0.   0.   1.
    column 1 to 5
 3.3217189   6.3257449   2.639556    2.1190304   8.4155184
 0.         -7.2505962  -0.5681337  -2.6648316  -10.974437
 0.         -0.3663024  -1.1788708   3.6574496  -8.6114572
 4.3685876   0.4373343   1.2800585   1.5312167   8.7841258
 2.6931248   4.8185089   7.7831286   6.9708506   1.1383597
    column 6 to 10
 1.        0.   0.   0.   0.
-1.786754   1.   0.   0.   0.
-1.5098633  0.   1.   0.   0.
 0.        0.   0.   1.   0.
 0.        0.   0.   0.   1.
    column 1 to 5
 3.3217189   6.3257449   2.639556    2.1190304   8.4155184
 0.         -7.2505962  -0.5681337  -2.6648316  -10.974437
 0.         -0.3663024  -1.1788708   3.6574496  -8.6114572
 0.         -7.8820242  -2.1913766  -1.2556447  -2.2836166
 2.6931248   4.8185089   7.7831286   6.9708506   1.1383597
    column 6 to 10
 1.        0.   0.   0.   0.
-1.786754   1.   0.   0.   0.
-1.5098633  0.   1.   0.   0.
-1.3151587  0.   0.   1.   0.
 0.        0.   0.   0.   1.
    column 1 to 5
 3.3217189   6.3257449   2.639556    2.1190304   8.4155184
 0.         -7.2505962  -0.5681337  -2.6648316  -10.974437
 0.         -0.3663024  -1.1788708   3.6574496  -8.6114572
 0.         -7.8820242  -2.1913766  -1.2556447  -2.2836166
 0.         -0.3101672   5.6430758   5.2528204  -5.6846263
    column 6 to 10
 1.        0.   0.   0.   0.
-1.786754   1.   0.   0.   0.
-1.5098633  0.   1.   0.   0.
-1.3151587  0.   0.   1.   0.
-0.8107624  0.   0.   0.   1.
    column 1 to 5
 3.3217189   0.          2.1438908  -0.205888   -1.1590718
 0.         -7.2505962  -0.5681337  -2.6648316  -10.974437
 0.         -0.3663024  -1.1788708   3.6574496  -8.6114572
 0.         -7.8820242  -2.1913766  -1.2556447  -2.2836166
 0.         -0.3101672   5.6430758   5.2528204  -5.6846263
    column 6 to 10
-0.5588443   0.8724448   0.   0.   0.
-1.786754    1.          0.   0.   0.
```

Chaque étape de la résolution affiche une « grande » matrice 5x10 (Cf : méthode magique vue en cours).

```
-1.5098633   0.          1.   0.   0.
-1.3151587   0.          0.   1.   0.
-0.8107624   0.          0.   0.   1.
      column 1 to 5
 3.3217189   0.          2.1438908  -0.205888   -1.1590718
 0.         -7.2505962  -0.5681337  -2.6648316  -10.974437
 0.          0.         -1.1501685   3.7920777  -8.0570252
 0.         -7.8820242  -2.1913766  -1.2556447  -2.2836166
 0.         -0.3101672   5.6430758   5.2528204  -5.6846263
      column 6 to 10
-0.5588443   0.8724448   0.   0.   0.
-1.786754    1.          0.   0.   0.
-1.419596   -0.0505203   1.   0.   0.
-1.3151587   0.          0.   1.   0.
-0.8107624   0.          0.   0.   1.
      column 1 to 5
 3.3217189   0.          2.1438908  -0.205888   -1.1590718
 0.         -7.2505962  -0.5681337  -2.6648316  -10.974437
 0.          0.         -1.1501685   3.7920777  -8.0570252
 0.         -8.882D-16  -1.5737663   1.6412573   9.6465439
 0.         -0.3101672   5.6430758   5.2528204  -5.6846263
      column 6 to 10
-0.5588443   0.8724448   0.   0.   0.
-1.786754    1.          0.   0.   0.
-1.419596   -0.0505203   1.   0.   0.
 0.6271972  -1.0870864   0.   1.   0.
-0.8107624   0.          0.   0.   1.
      column 1 to 5
 3.3217189   0.          2.1438908  -0.205888   -1.1590718
 0.         -7.2505962  -0.5681337  -2.6648316  -10.974437
 0.          0.         -1.1501685   3.7920777  -8.0570252
 0.         -8.882D-16  -1.5737663   1.6412573   9.6465439
 0.          0.          5.6673795   5.366817   -5.21516
      column 6 to 10
-0.5588443   0.8724448   0.   0.   0.
-1.786754    1.          0.   0.   0.
-1.419596   -0.0505203   1.   0.   0.
 0.6271972  -1.0870864   0.   1.   0.
-0.7343283  -0.0427782   0.   0.   1.
      column 1 to 5
 3.3217189   0.          0.          6.8624679  -16.177203
 0.         -7.2505962  -0.5681337  -2.6648316  -10.974437
 0.          0.         -1.1501685   3.7920777  -8.0570252
 0.         -8.882D-16  -1.5737663   1.6412573   9.6465439
 0.          0.          5.6673795   5.366817   -5.21516
      column 6 to 10
-3.2049423   0.7782759   1.8639797   0.   0.
-1.786754    1.          0.          0.   0.
-1.419596   -0.0505203   1.          0.   0.
 0.6271972  -1.0870864   0.          1.   0.
-0.7343283  -0.0427782   0.          0.   1.
```

```
    column 1 to 5
 3.3217189   0.         0.         6.8624679  -16.177203
 0.         -7.2505962   0.        -4.5379544  -6.9946138
 0.          0.        -1.1501685   3.7920777  -8.0570252
 0.         -8.882D-16  -1.5737663   1.6412573   9.6465439
 0.          0.         5.6673795   5.366817   -5.21516
    column 6 to 10
-3.2049423   0.7782759   1.8639797   0.   0.
-1.0855349   1.0249549  -0.4939569   0.   0.
-1.419596   -0.0505203   1.          0.   0.
 0.6271972  -1.0870864   0.          1.   0.
-0.7343283  -0.0427782   0.          0.   1.
    column 1 to 5
 3.3217189   0.         0.         6.8624679  -16.177203
 0.         -7.2505962   0.        -4.5379544  -6.9946138
 0.          0.        -1.1501685   3.7920777  -8.0570252
 0.         -8.882D-16   2.220D-16  -3.5474119   20.670906
 0.          0.         5.6673795   5.366817   -5.21516
    column 6 to 10
-3.2049423   0.7782759   1.8639797   0.   0.
-1.0855349   1.0249549  -0.4939569   0.   0.
-1.419596   -0.0505203   1.          0.   0.
 2.5696189  -1.0179598  -1.3682919   1.   0.
-0.7343283  -0.0427782   0.          0.   1.
    column 1 to 5
 3.3217189   0.         0.         6.8624679  -16.177203
 0.         -7.2505962   0.        -4.5379544  -6.9946138
 0.          0.        -1.1501685   3.7920777  -8.0570252
 0.         -8.882D-16   2.220D-16  -3.5474119   20.670906
 0.          0.         0.          24.05203   -44.915621
    column 6 to 10
-3.2049423   0.7782759   1.8639797   0.   0.
-1.0855349   1.0249549  -0.4939569   0.   0.
-1.419596   -0.0505203   1.          0.   0.
 2.5696189  -1.0179598  -1.3682919   1.   0.
-7.7292939  -0.2917137   4.9274341   0.   1.
    column 1 to 5
 3.3217189  -1.718D-15   4.295D-16   0.         23.810662
 0.         -7.2505962   0.        -4.5379544  -6.9946138
 0.          0.        -1.1501685   3.7920777  -8.0570252
 0.         -8.882D-16   2.220D-16  -3.5474119   20.670906
 0.          0.         0.          24.05203   -44.915621
    column 6 to 10
 1.7659851  -1.1909672  -0.7829808   1.9344999   0.
-1.0855349   1.0249549  -0.4939569   0.          0.
-1.419596   -0.0505203   1.          0.          0.
 2.5696189  -1.0179598  -1.3682919   1.          0.
-7.7292939  -0.2917137   4.9274341   0.          1.
    column 1 to 5
 3.3217189  -1.718D-15   4.295D-16   0.         23.810662
 0.         -7.2505962  -2.840D-16   0.        -33.437449
```

```
 0.         0.        -1.1501685   3.7920777  -8.0570252
 0.        -8.882D-16   2.220D-16  -3.5474119   20.670906
 0.         0.         0.          24.05203   -44.915621
     column 6 to 10
 1.7659851  -1.1909672  -0.7829808   1.9344999   0.
-4.3726675   2.3271592   1.2564027  -1.2792296   0.
-1.419596   -0.0505203   1.          0.         0.
 2.5696189  -1.0179598  -1.3682919   1.          0.
-7.7292939  -0.2917137   4.9274341   0.          1.
     column 1 to 5
 3.3217189  -1.718D-15   4.295D-16   0.         23.810662
 0.        -7.2505962  -2.840D-16   0.        -33.437449
 0.        -9.494D-16  -1.1501685  -4.441D-16   14.039558
 0.        -8.882D-16   2.220D-16  -3.5474119   20.670906
 0.         0.         0.          24.05203   -44.915621
     column 6 to 10
 1.7659851  -1.1909672  -0.7829808   1.9344999   0.
-4.3726675   2.3271592   1.2564027  -1.2792296   0.
 1.3272501  -1.138689   -0.4626633   1.0689702   0.
 2.5696189  -1.0179598  -1.3682919   1.          0.
-7.7292939  -0.2917137   4.9274341   0.          1.
     column 1 to 5
 3.3217189  -1.718D-15   4.295D-16   0.         23.810662
 0.        -7.2505962  -2.840D-16   0.        -33.437449
 0.        -9.494D-16  -1.1501685  -4.441D-16   14.039558
 0.        -8.882D-16   2.220D-16  -3.5474119   20.670906
 0.        -6.022D-15   1.505D-15   0.         95.236486
     column 6 to 10
 1.7659851  -1.1909672  -0.7829808   1.9344999   0.
-4.3726675   2.3271592   1.2564027  -1.2792296   0.
 1.3272501  -1.138689   -0.4626633   1.0689702   0.
 2.5696189  -1.0179598  -1.3682919   1.          0.
 9.6931401  -7.1936468  -4.3498077   6.7801626   1.
     column 1 to 5
 3.3217189  -2.126D-16   5.315D-17   0.          0.
 0.        -7.2505962  -2.840D-16   0.        -33.437449
 0.        -9.494D-16  -1.1501685  -4.441D-16   14.039558
 0.        -8.882D-16   2.220D-16  -3.5474119   20.670906
 0.        -6.022D-15   1.505D-15   0.         95.236486
     column 6 to 10
-0.6574568   0.6075609   0.3045415   0.2393495  -0.2500162
-4.3726675   2.3271592   1.2564027  -1.2792296   0.
 1.3272501  -1.138689   -0.4626633   1.0689702   0.
 2.5696189  -1.0179598  -1.3682919   1.          0.
 9.6931401  -7.1936468  -4.3498077   6.7801626   1.
     column 1 to 5
 3.3217189  -2.126D-16   5.315D-17   0.          0.
 0.        -7.2505962   2.445D-16   0.          0.
 0.        -9.494D-16  -1.1501685  -4.441D-16   14.039558
 0.        -8.882D-16   2.220D-16  -3.5474119   20.670906
 0.        -6.022D-15   1.505D-15   0.         95.236486
```

```
      column 6 to 10
 -0.6574568   0.6075609   0.3045415   0.2393495  -0.2500162
 -0.9694142  -0.1985241  -0.2708111   1.1012797   0.3510992
  1.3272501  -1.138689   -0.4626633   1.0689702   0.
  2.5696189  -1.0179598  -1.3682919   1.          0.
  9.6931401  -7.1936468  -4.3498077   6.7801626   1.
      column 1 to 5
  3.3217189  -2.126D-16   5.315D-17   0.          0.
  0.         -7.2505962   2.445D-16   0.          0.
  0.         -6.169D-17  -1.1501685  -4.441D-16   0.
  0.         -8.882D-16   2.220D-16  -3.5474119   20.670906
  0.         -6.022D-15   1.505D-15   0.          95.236486
      column 6 to 10
 -0.6574568   0.6075609   0.3045415   0.2393495  -0.2500162
 -0.9694142  -0.1985241  -0.2708111   1.1012797   0.3510992
 -0.1016918  -0.0782171   0.178576    0.0694532  -0.1474178
  2.5696189  -1.0179598  -1.3682919   1.          0.
  9.6931401  -7.1936468  -4.3498077   6.7801626   1.
      column 1 to 5
  3.3217189  -2.126D-16   5.315D-17   0.          0.
  0.         -7.2505962   2.445D-16   0.          0.
  0.         -6.169D-17  -1.1501685  -4.441D-16   0.
  0.          4.189D-16  -1.047D-16  -3.5474119   0.
  0.         -6.022D-15   1.505D-15   0.          95.236486
      column 6 to 10
 -0.6574568   0.6075609   0.3045415   0.2393495  -0.2500162
 -0.9694142  -0.1985241  -0.2708111   1.1012797   0.3510992
 -0.1016918  -0.0782171   0.178576    0.0694532  -0.1474178
  0.4657404   0.5434082  -0.4241741  -0.471622   -0.2170482
  9.6931401  -7.1936468  -4.3498077   6.7801626   1.
```

On récupère la matrice C...

```
--> // Affichage de C arrondi au millième
--> disp(round(1000*C)/1000)

 -0.198   0.183   0.092   0.072  -0.075
  0.134   0.027   0.037  -0.152  -0.048
  0.088   0.068  -0.155  -0.06    0.128
 -0.131  -0.153   0.12    0.133   0.061
  0.102  -0.076  -0.046   0.071   0.011
```

... enfin on l'affiche arrondie au millième.

```
--> // Calcul de l'inverse de A par scilab
--> A=inv(A);
--> // Comparaison des résultats obtenus
--> disp(round(1000*A)/1000)

 -0.198   0.183   0.092   0.072  -0.075
  0.134   0.027   0.037  -0.152  -0.048
  0.088   0.068  -0.155  -0.06    0.128
 -0.131  -0.153   0.12    0.133   0.061
  0.102  -0.076  -0.046   0.071   0.011
```

On affiche l'inverse théorique, calculé par la fonction inv() de Scilab, afin de la comparer au résultat obtenu précédemment.

IV. METHODE D'APPROXIMATION

1. HYPOTHESES

Nous disposons d'un ensemble de quatre fonctions (Cf : IV.2). Pour chaque jeu de deux fonctions, la fonction f permet d'approcher les valeurs prises par la fonction u.

Il s'agit de résoudre l'équation *MY=D*, pour M une matrice tridiagonale. D est exprimé en fonction du temps, noté T, où chaque valeur D_i est obtenue par :

$$D_i = h^2 f(T_i)$$

Ainsi, les points (T_i, Y_i) seront censés approximer ceux de la courbe C_u, représentative de u(t).

NB : on résout en fait une équation différentielle.

2. FONCTIONS A DISPOSITION

```
function y = Ua(t)
    y = t*sin(7*(%pi)*t)
endfunction
79
function y = Ub(t)
    y = (t-t^2)*(2+sin(9*%pi*t))
endfunction
83
function y = Fa(t)
    y = 49*(%pi^2)*t*sin(7*%pi*t) - 14*%pi*cos(7*%pi*t)
endfunction
87
function y = Fb(t)
    y = 4 + 2*sin(9*%pi*t) + (2*t-1)*18*%pi*cos(9*%pi*t) + (t-t^2)*((9*%pi)^2)*sin(9*%pi*t
)
endfunction
```

3. DETAILS DU SCRIPT

Le script de l'exercice 3 se décline en deux étapes (a et b, voir sujet).

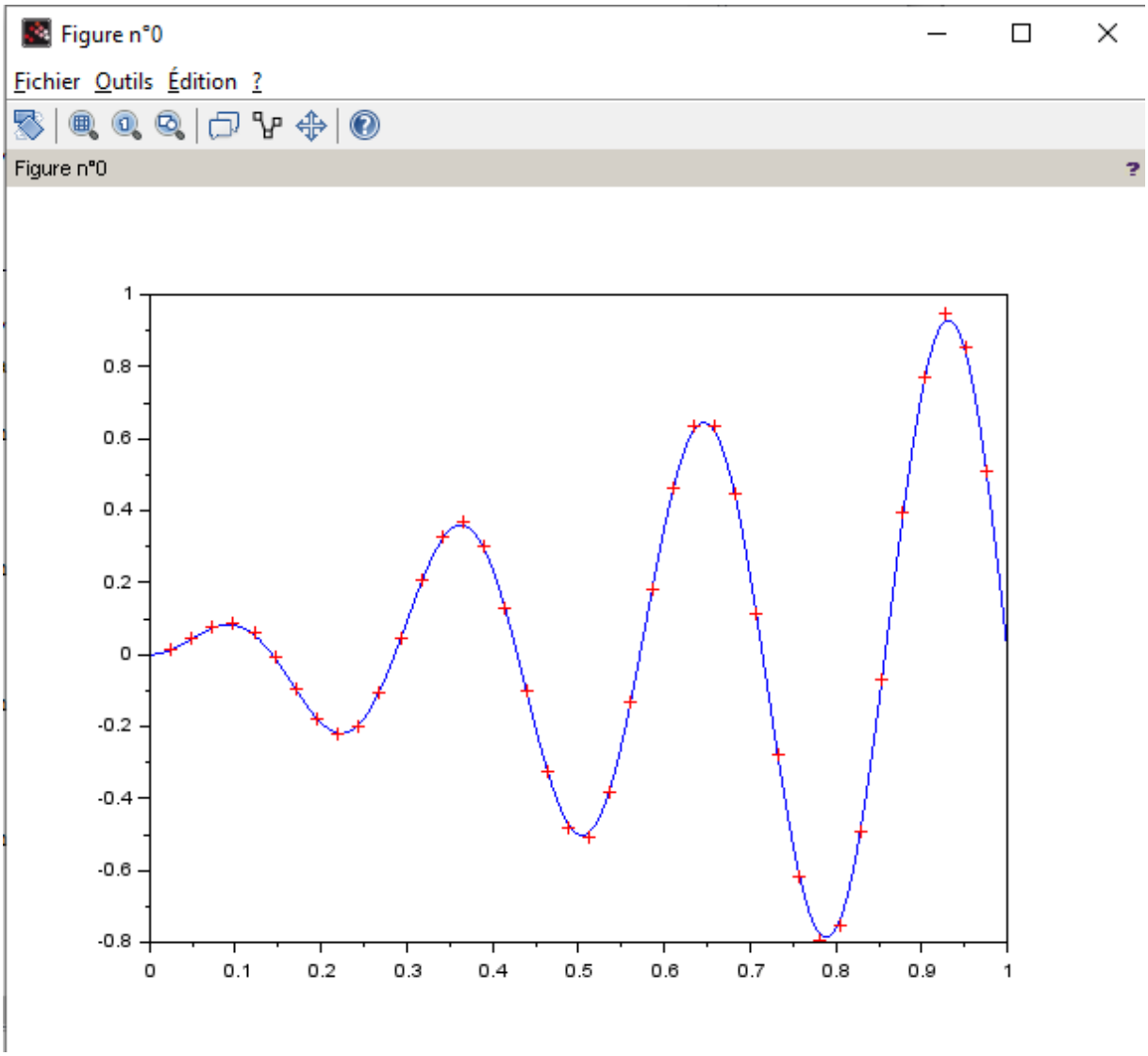
Sur le graphique, nous réalisons d'abord l'affichage de la courbe décrite par u(t) pour un très grand nombre de valeur. Pour le déroulement de cet exercice, nous fixons t à 2000 valeurs compris entre 0 et 1.

Il s'agit ensuite d'approximer cette courbe par f(t) selon un nombre connu de points (ici, 10, 20, 30, 40 puis 50). Nous initialisons les variables dont nous avons besoin : h = 1/N+1, M (qui est en fait l'ensemble des matrices A, B et C), T et D qui sont définies comme ci-avant.

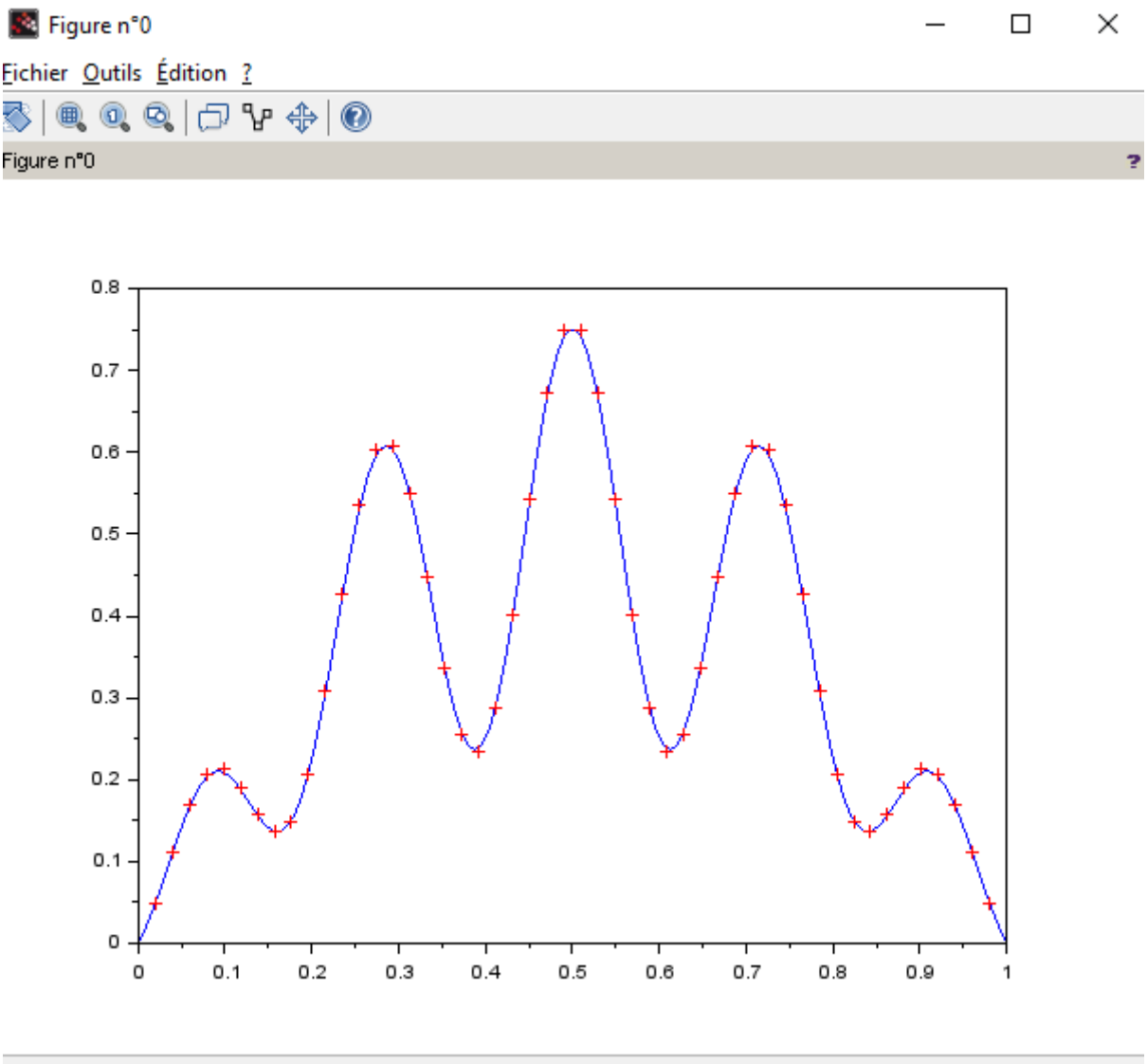
Pour résoudre le problème, nous disposons déjà de la fonction RESOUTRI du premier exercice. Il ne reste plus qu'a afficher les valeurs trouvées pour T et pour Y telles que (T_i, Y_i) les points approximant u(t).

On réitère bien sûr l'opération pour les fonctions de l'étape b.

4. RESULTATS



Approximation de Ua(t) en bleue par Fa(t) en rouge



Approximation de Ub(t) en bleue par Fb(t) en rouge