Analyse Numérique

Semestre de Printemps 2022 - Section MA

Prof. Annalisa Buffa

Corrigé séance 2 - 4 mars 2022

Introduction à Matlab, Conditionnement et Stabilité

Exercice 1

1. Entrez la matrice

```
>> A=[1 2 3; 2 3 1; 3 1 2]
```

   Quels sont les résultats des commandes suivantes ?

```
>> A([2 3],[1 3])
>> A([2 3],1:2)
>> A([2 3],:)
>> A([2 3],end)
>> A(:)
>> A(5)
>> reshape(A(:),size(A))
```

2. Écrire les instructions MATLAB pour construire une matrice triangulaire supérieure (resp. inférieure) de dimension 10 ayant des 2 sur la diagonale principale et des 3 sur la seconde sur-diagonale (resp. sous-diagonale).
3. Écrire la matrice carrée M d'ordre 12 contenant les entiers de 1 à 144 rangés par ligne. Extraire de cette matrice les matrices suivantes :
   - la sous-matrice formée par les coefficients a_{ij} pour $i = 1, \dots, 6$ et $j = 7, \dots, 12$;
   - celles des coefficients a_{ij} pour $(i, j) \in \{1, 2, 3, 4, 5, 6, 9, 10\}^2$;
   - celles des coefficients a_{ij} pour $i + j$ pair. (Utilisez la command `rem(n,2)` pour controller si n est pair ou impair).

Solution

1.
```
>> A=[1 2 3; 2 3 1; 3 1 2 ]
A =

1     2     3
2     3     1
```

```
3     1     2

>> A([2 3],[1 3])

ans =

2     1
3     2

>> A([2 3],1:2)

ans =

2     3
3     1
>> A([2 3],:)

ans =

2     3     1
3     1     2

>> A([2 3],end)

ans =

1
2

>> A(:)

ans =

1
2
3
2
3
1
3
1
2

>> A(5)

ans =

3

>> reshape(A(:),size(A))

ans =

1     2     3
2     3     1
3     1     2
```

2. Pour la matrice triangulaire supérieure :

```
>> M=diag(2*ones(10,1))+diag(3*ones(9,1),1)
```

```
M =

2    3    0    0    0    0    0    0    0    0
0    2    3    0    0    0    0    0    0    0
0    0    2    3    0    0    0    0    0    0
0    0    0    2    3    0    0    0    0    0
0    0    0    0    2    3    0    0    0    0
0    0    0    0    0    2    3    0    0    0
0    0    0    0    0    0    2    3    0    0
0    0    0    0    0    0    0    2    3    0
0    0    0    0    0    0    0    0    2    3
0    0    0    0    0    0    0    0    0    2
```

Pour la matrice triangulaire inférieure :

```
>> M=diag(2*ones(10,1))+diag(3*ones(9,1),-1)

M =

2    0    0    0    0    0    0    0    0    0
3    2    0    0    0    0    0    0    0    0
0    3    2    0    0    0    0    0    0    0
0    0    3    2    0    0    0    0    0    0
0    0    0    3    2    0    0    0    0    0
0    0    0    0    3    2    0    0    0    0
0    0    0    0    0    3    2    0    0    0
0    0    0    0    0    0    3    2    0    0
0    0    0    0    0    0    0    3    2    0
0    0    0    0    0    0    0    0    3    2
```

3. Nous utilisons le code suivant :

```
>> v = [1:144];
>> A = vec2mat(v,12);

A =

1      2      3      4      5      6      7      8      9     10     11     12
13     14     15     16     17     18     19     20     21     22     23     24
25     26     27     28     29     30     31     32     33     34     35     36
37     38     39     40     41     42     43     44     45     46     47     48
49     50     51     52     53     54     55     56     57     58     59     60
61     62     63     64     65     66     67     68     69     70     71     72
73     74     75     76     77     78     79     80     81     82     83     84
85     86     87     88     89     90     91     92     93     94     95     96
97     98     99    100    101    102    103    104    105    106    107    108
109    110    111    112    113    114    115    116    117    118    119    120
121    122    123    124    125    126    127    128    129    130    131    132
133    134    135    136    137    138    139    140    141    142    143    144

>> A(1:6,7:12)

ans =

7      8      9     10     11     12
19     20     21     22     23     24
31     32     33     34     35     36
43     44     45     46     47     48
55     56     57     58     59     60
```

```
67    68    69    70    71    72

>> A(1:10,1:10)

ans =

1     2     3     4     5     6     7     8     9    10
13    14    15    16    17    18    19    20    21    22
25    26    27    28    29    30    31    32    33    34
37    38    39    40    41    42    43    44    45    46
49    50    51    52    53    54    55    56    57    58
61    62    63    64    65    66    67    68    69    70
73    74    75    76    77    78    79    80    81    82
85    86    87    88    89    90    91    92    93    94
97    98    99   100   101   102   103   104   105   106
109   110   111   112   113   114   115   116   117   118

>> for i = 1:size(A)
B(i,:)=A(i,2-rem(i,2):2:end);
end

>> B

B =

1     3     5     7     9    11
14    16    18    20    22    24
25    27    29    31    33    35
38    40    42    44    46    48
49    51    53    55    57    59
62    64    66    68    70    72
73    75    77    79    81    83
86    88    90    92    94    96
97    99   101   103   105   107
110   112   114   116   118   120
121   123   125   127   129   131
134   136   138   140   142   144
```

Exercice 2

On considère les vecteurs $x = [1\ 4\ 7\ 2\ 1\ 2]$ et $y = [0\ 9\ 1\ 4\ 3\ 0]$. Calculer :

1. le produit, composante par composante, entre deux vecteurs x et y ;
2. le produit scalaire entre les mêmes vecteurs x et y ;
3. un vecteur dont les éléments sont définis par :
   $v_1 = x_1 y_n$, $v_2 = x_2 y_{n-1}$, . . ., $v_{n-1} = x_{n-1} y_2$, $v_n = x_n y_1$.

Suggestion : utiliser une boucle for pour calculer les composants de v.

Solution

Dans un script, on a :

```
clc
clear
close all
```

```
x=[1 4 7 2 1 2];
y=[0 9 1 4 3 0];

% 1) produit composante pour composante
ElByElProd = x.*y

% 2a) produit scalaire
ScalProd = x*y'

% 2b) on peut aussi utiliser la fonction built-in "dot"
dot(x,y)

% 3) On a 3 manieres

% 3a) une boucle for

n=size(x,2); % c'est a dire le nombre de colonnes de x.

% On peut aussi utiliser n=length(x)
v1 = zeros(1,n);
for i = 1:n
v1(i)=x(i)*y(n-i+1);
end
v1

% 3b) inverser l'ordre du vecteur y
v2=x.*y(end:-1:1)

% 3c) inverser l'ordre du vecteur y avec la fonction built-in "fliplr"
v3=x.*fliplr(y)
```

Exercice 3

1. Soit
$$f(x) = \frac{x^2}{2}\sin(x),\ x \in [1, 20]$$
une fonction qu'on veut représenter graphiquement en choisissant 10 points, 20 points et 100 points dans l'intervalle de définition. Réaliser les trois graphiques sur la même figure et avec trois couleurs différentes. Quelle est la meilleure représentation ?
2. Faire la même chose pour les fonctions :
$$g(x) = \frac{x^3}{6}\cos(\sin(x))\exp(-x) + \left(\frac{1}{1+x}\right)^2,\ x \in [1, 20];$$
$$h(x) = x(1-x) + \frac{\sin(x)\cos(x)}{x^3}, x \in [1, 20].$$

Solution

1. Dans un script, on a :

```
clc
clear
close all

% On a 2 manieres. La premiere c'est avec les operations standard
```

```
x1 = linspace(1,20,10);
f1=x1.^2/2.*sin(x1);
x2 = linspace(1,20,20);
f2=x2.^2/2.*sin(x2);
x3 = linspace(1,20,100);
f3=x3.^2/2.*sin(x3);

figure
plot(x1,f1,'r','LineWidth',2)
hold on
plot(x2,f2,'g','LineWidth',2)
plot(x3,f3,'m','LineWidth',2)
legend('10 points','20 points','100 points')
xlabel('x')
ylabel('f(x)')

% la deuxieme maniere c'est avec les @-functions
f=@(x) x.^2/2.*sin(x);
x1 = linspace(1,20,10);
x2 = linspace(1,20,20);
x3 = linspace(1,20,100);

figure
plot(x1,f(x1),'r','LineWidth',2)
hold on
plot(x2,f(x2),'g','LineWidth',2)
plot(x3,f(x3),'m','LineWidth',2)
legend('10 points','20 points','100 points')
xlabel('x')
ylabel('f(x)')
```

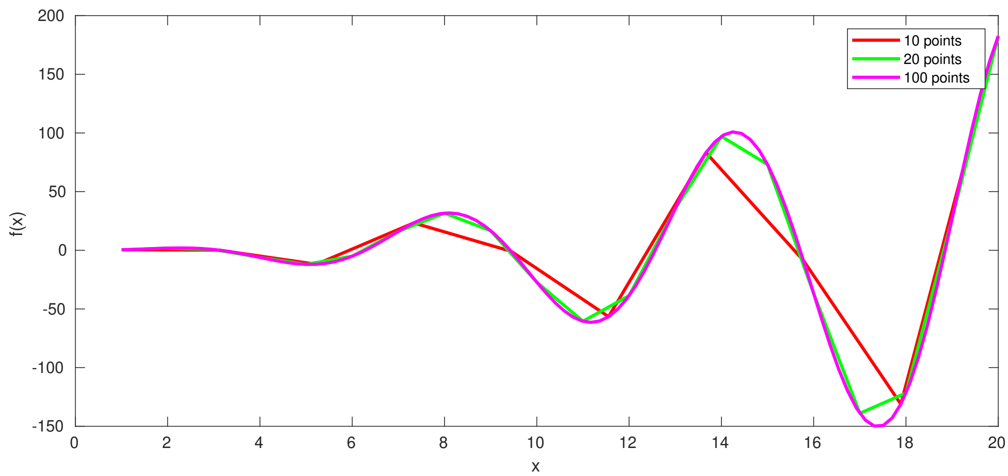

FIGURE 1 – Graphe de la fonction f.

2. On remplace la fonction f par les fonctions g et h :

```
g=@(x) x.^3/6.*cos(sin(x)).*exp(-x)+(1./(1+x)).^2;
h=@(x) x.*(1-x)+sin(x).*cos(x)./(x.^3);
```

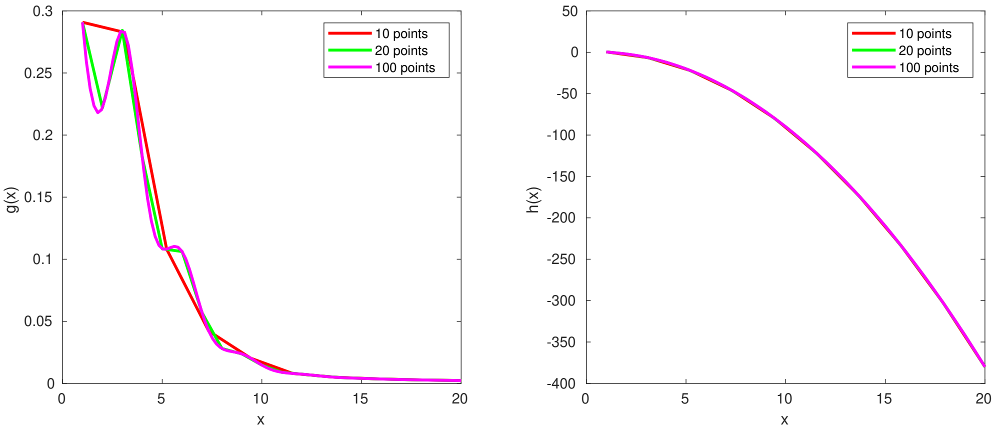

FIGURE 2 – Graphes des fonctions g (gauche) et h (droite).

Exercice 4

Rappel : Représentation des grands nombres dans Matlab
Soit $\mathcal{F}$ l'ensemble fini de nombres rationnels écrit en format normalisé. Cet ensemble dépend de 4 paramètres :

- la base β dans lequel on représente les nombres ($\beta = 2$ base binaire et $\beta = 10$ base décimale) ;
- le nombre t de chiffres significatifs ;
- une borne inférieure L et une borne supérieure U.

On peut alors écrire les nombres sous la forme $(\pm 0.\alpha_a \cdots \alpha_t)_\beta \cdot \beta^e$.
De nos jours, dans les calculateurs (standard IEC 559) $\beta = 2$, $t = 53$ (53 chiffres de mantisse, chiffres après la virgule) et $L = -1021$, $U = 1024$.
Une méthode pour observer ce qu'il se passe dans la mémoire est d'utiliser le format de représentation `format rat` (on cherche à exprimer un nombre comme fraction d'entiers les plus petits possibles). Dans cette exercice, vous utiliserez le `format rat`.

1. Calculer 10^{22} puis 10^{23}, et enfin $100000000000000000000000 - 99999999999999991611392$ (c'est-à-dire la valeur exacte de 10^{23} moins sa valeur approchée, qui peut dépendre de l'ordinateur utilisé). Qu'observez-vous ?
2. En utilisant la fonction `linspace`, diviser l'intervalle $[2^{52}, 2^{52}+1]$ en 11 parties, grâce à la fonction :

```
linspace(2^52,2^52+1,11)'
```

   Qu'observez-vous ? À partir de quand ?
3. On sait que
$$\sqrt{x+1} - \sqrt{x} = \frac{1}{\sqrt{x+1} + \sqrt{x}}.$$
   Pour $x = 10, 1000, \cdots, 10^{19}$, comparer sur Matlab $\sqrt{x+1} - \sqrt{x}$ et $\dfrac{1}{\sqrt{x+1} + \sqrt{x}}$. Que constatez-vous ?

Solution

1. On utilise les commandes suivantes :

```
clc
clear all
close all

format rat

10^22
10^23
100000000000000000000000-99999999999999991611392
```

On obtient :

```
ans =
100000000000000000000000

ans =
99999999999999991611392

ans =
       0
```

On observe que $10^{22} \in \mathcal{F}(2, 53, -1021, +1024)$ et $10^{23} \notin \mathcal{F}$. En effet, on a $10^n = 5^n \cdot 2^n$. L'entier 5^n joue le rôle de mantisse, or on a $5^n \leq 2^{53} - 1 < 2^{53}$, *i.e.* $n < 22.83$. Cependant, on observe que la différence est nulle car l'approxiation de 100000000000000000000000 en virgule flottante, c'est-à-dire l'approximation de 10^{23} est égale à 99999999999999991611392.

2. On obtient :

```
ans =
4503599627370496
4503599627370496
4503599627370496
4503599627370496
4503599627370496
4503599627370496
4503599627370497
4503599627370497
4503599627370497
4503599627370497
4503599627370497
```

On observe que l'intervalle $[2^{52}, 2^{52} + 1]$ ne contient que les entiers et qu'à partir de $2^{52} + 0.6$, on arrondit au chiffre supérieur.

3. On utilise les commandes suivantes :

```
linspace(2^52,2^52+1,11)'

for e=1:2:19
    x=10^e;
    a= sqrt(x+1)-sqrt(x);
    b= 1/(sqrt(x+1)+sqrt(x));

    fprintf('10^%d \t %e \t %e \n',e,a,b )
end
```

On obtient :

```
10^1     1.543471e-01    1.543471e-01
10^3     1.580744e-02    1.580744e-02
10^5     1.581135e-03    1.581135e-03
10^7     1.581139e-04    1.581139e-04
10^9     1.581139e-05    1.581139e-05
10^11    1.581153e-06    1.581139e-06
10^13    1.578592e-07    1.581139e-07
10^15    1.862645e-08    1.581139e-08
10^17    0.000000e+00    1.581139e-09
10^19    0.000000e+00    1.581139e-10
```

On observe que pour $x = 10^1$ à 10^9, les résultats semblent similaires puis ils diffèrent. Quelle est la bonne réponse ? Y en a t-il une ? La seule chose que l'on peut dire est qu'elles ne peuvent pas être justes en même temps. Cette question montre que deux formules algébriquement équivalentes peuvent donner des résultats différents.

Exercice 5

Soit $x \in \mathbb{R}$, on définit par $fl(x)$ (float de x) le nombre appartenant à l'ensemble $\mathcal{F}$ le plus proche de x. Ainsi, on définit la fonction d'approximation :

$$fl : \mathbb{R} \to \mathcal{F}.$$

On peut montrer que l'erreur relative d'approximation de x vérifie

$$\frac{|x - fl(x)|}{|x|} \leq \frac{1}{2}\beta^{1-t}.$$

On note $u = \dfrac{1}{2}\beta^{1-t}$ l'erreur d'arrondissement, *roundoff*.
On définit par ε_x epsilon machine, erreur machine, la quantité suivante :

$$\varepsilon_x = \frac{fl(x) - x}{x} \qquad \text{avec } |\varepsilon_x| \leq u,$$

on en déduit que $fl(x) = x(1 + \varepsilon_x)$.
Enfin, on définit par ε_p (resp. ε_d) l'erreur relative du produit (resp. de la division) signée.

1. Calculer l'erreur relative du produit pour $x, y \in \mathbb{R}$, montrer qu'elle est égale à $|\varepsilon_x + \varepsilon_y + \varepsilon_p + \varepsilon_x\varepsilon_y + \varepsilon_x\varepsilon_p + \varepsilon_y\varepsilon_p + \varepsilon_x\varepsilon_y\varepsilon_p|$.
2. On suppose que $|\varepsilon_x|, |\varepsilon_y|, |\varepsilon_p| \leq u$, majorer cette erreur par u.
3. Que pouvez-vous en conclure ?
4. Refaire de même avec la division.

Solution

Multiplication en virgule flottante

On a :

$$\begin{aligned} fl(fl(x)fl(y)) &= (x(1+\varepsilon_x))(y(1+\varepsilon_y))(1+\varepsilon_p) \\ &= xy(1+\varepsilon_x)(1+\varepsilon_y)(1+\varepsilon_p) \\ &= xy(1+\varepsilon_x+\varepsilon_y+\varepsilon_p+\varepsilon_x\varepsilon_y+\varepsilon_x\varepsilon_p+\varepsilon_y\varepsilon_p+\varepsilon_x\varepsilon_y\varepsilon_p). \end{aligned}$$

L'erreur relative du produit est égale à :

$$\begin{aligned}\frac{|fl(fl(x)fl(y)) - xy|}{|xy|} &= \frac{|xy(\varepsilon_x + \varepsilon_y + \varepsilon_p + \varepsilon_x\varepsilon_y + \varepsilon_x\varepsilon_p + \varepsilon_y\varepsilon_p + \varepsilon_x\varepsilon_y\varepsilon_p)|}{|xy|} \\ &= |\varepsilon_x + \varepsilon_y + \varepsilon_p + \varepsilon_x\varepsilon_y + \varepsilon_x\varepsilon_p + \varepsilon_y\varepsilon_p + \varepsilon_x\varepsilon_y\varepsilon_p|.\end{aligned}$$

Or on a $|\varepsilon_x|,\ |\varepsilon_y|,\ |\varepsilon_p| \leq u << 1$, d'où

$$\frac{|fl(fl(x)fl(y)) - xy|}{|xy|} \approx |\varepsilon_x + \varepsilon_y + \varepsilon_p| \leq 3u.$$

On constate que le produit d'élements de $\mathbb{R}$ semble stable.

Division en virgule flottante

On fait de même pour la division est on a :

$$\begin{aligned} fl\left(\frac{fl(x)}{fl(y)}\right) &= \frac{(x(1+\varepsilon_x))(1+\varepsilon_d)}{y(1+\varepsilon_y)} \\ &\approx \frac{x}{y}(1+\varepsilon_x)(1-\varepsilon_y)(1+\varepsilon_d).\end{aligned}$$

On a donc

$$\frac{|fl\left(\frac{fl(x)}{fl(y)}\right) - \frac{x}{y}|}{|\frac{x}{y}|} \approx |(1+\varepsilon_x)(1-\varepsilon_y)(1+\varepsilon_d) - 1| \leq 3u.$$

Exercice 6

1. Vérifier numériquement que $\varepsilon = 2^{-52}$, où ε est l'erreur machine, c'est-à-dire le plus petit x tel que $fl(1+x) \neq 1$.

   Suggestion : Afin de visualiser les résultats, il est plus facile de regarder la différence entre 1 et $fl(1+x)$.

2. On définit par ε_s l'erreur relative de la somme signée. On suppose que $|\varepsilon_x|, |\varepsilon_y|, |\varepsilon_s| \leq u$, calculer (**à la main**) l'erreur relative de la somme pour $x, y \in \mathbb{R}$.

3. En utilisant la question précédente, les deux racines $\{x_+, x_-\}$ de l'équation $x^2 - 2px + 1 = 0$ pour $p \geq 1$, sont-elles bien stables ? Si non, trouver un autre moyen pour la (ou les) calculer.

   En utilisant Matlab, sachant que le conditionnement du système précédent est $K(p) \simeq \dfrac{p}{\sqrt{p^2-1}}$ pour $p > 1$, tracer ce conditionnement en fonction de p. Qu'observe-t-on ?

   Tracer les différentes courbes $y = x_+x_- - 1$ en fonction de p, pour p variant de 1 et 100. Que constatez-vous ?

   Remarque : On sait que $x_+x_- = 1$.

Solution

1. On utilise les commandes suivantes :

```
 clc
clear all
close all

format rat

for e=50:1:54
    x=2^-e;
    a=(1+x)-1;
    fprintf('2^%d \t %e \n',e,a)
end
```

On obtient :

```
2^-50    8.881784e-16
2^-51    4.440892e-16
2^-52    2.220446e-16
2^-53    0.000000e+00
2^-54    0.000000e+00
```

2. On a :

$$\begin{aligned} fl(fl(x)+fl(y)) &= (x(1+\varepsilon_x)+y(1+\varepsilon_y))(1+\varepsilon_s) \\ &= x+y+x\varepsilon_x+y\varepsilon_y+(x+y)\varepsilon_s+x\varepsilon_x\varepsilon_s+y\varepsilon_y\varepsilon_s. \end{aligned}$$

On en déduit que :

$$\begin{aligned} \frac{|fl(fl(x)+fl(y))-(x+y)|}{|x+y|} &\leq \frac{|x|}{|x+y|}\varepsilon_x+\frac{|y|}{|x+y|}\varepsilon_y+\varepsilon_s+\frac{|x|}{|x+y|}\varepsilon_x\varepsilon_s+\frac{|y|}{|x+y|}\varepsilon_y\varepsilon_s \\ &\leq u+u(1+u)\frac{|x|+|y|}{|x+y|} \end{aligned}$$

On en déduit que la somme est stable si $x+y$ n'est pas "trop" petit. Si $x+y$ est petit, on peut avoir des *erreurs d'annulation*.

3. On a $\Delta = 4p^2-4 \geq 0$ et les racines sont $x_\pm = p \pm \sqrt{p^2-1}$.
D'après la question précédente, si p est proche de 1, les deux racines sont instables et il n'est pas possible de les rendre stables. De plus, si p est grand, x_+ est stable mais $x_- = p-\sqrt{p^2-1}$ est instable. Cette formule est en effet sujette aux erreurs dues à l'*annulation numérique*. Dans ce cas, une possibilité est d'utiliser le fait que $x_+x_- = 1$ *i.e.* $x_- = \dfrac{1}{x_+}$. En effet, on sait que x_+ est stable et que la division est une opération stable, donc x_- calculé de cette manière est stable quand p est grand.

```
format long
% Solution de l'equation du second ordre x^2-2px+1=0 p >=1
figure(1)
f=@(p) p/sqrt(p^2-1);
fplot(f,[1, 10])

% Stabilite
p=linspace(1,100,101)';
xp=p+sqrt(p.^2-1);
xm=p-sqrt(p.^2-1);
xm2=1./xp;
figure(2)
plot(p,xp.*xm-1,p,xp.*xm2-1)
legend('xp*xm-1','xp*xm2-1')
```

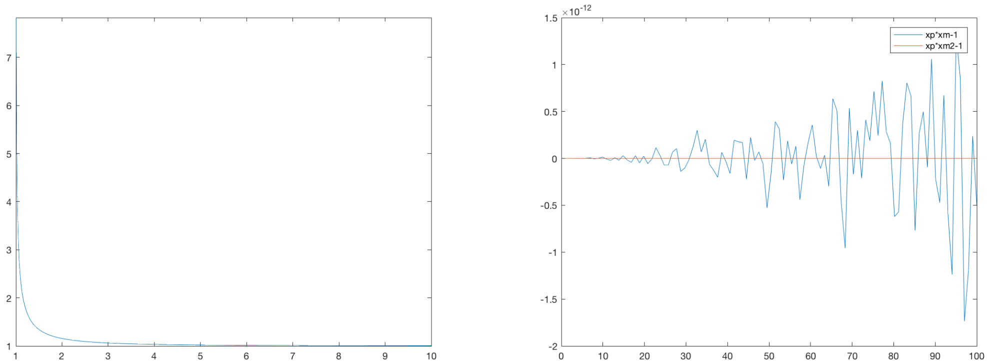

FIGURE 3 – Conditionnement $K(p) \simeq \dfrac{p}{\sqrt{p^2-1}}$ (gauche) et courbes $y = x_+ * x_- - 1$ (droite).

On observe, tout d'abord, que le conditionnement est très grand lorsque p est au voisinage de 1, i.e. lorsque le problème admet deux racines "proches" l'une de l'autre (la racine est multiple si $p = 1$). Dans ce cas, le problème est mal conditionné/instable.

De plus, le conditionnement est petit lorsque $p >> 1$, car dans ce cas, $K(p) \simeq 1$. Donc dans ce cas, le problème est bien conditionné. En effet, nous avons trouvé une manière stable de calculer les deux racines quand p est grand. Il faut donc bien faire la différence entre la stabilité du problème et la stabilité de la méthode numérique, c'est-à-dire ici de la méthode de calcul des racines.

Remarquez également que sur le graphe de droite de la Figure 3, l'instabilité du problème autour de $p = 1$ n'est pas visible, alors que l'instabilité numérique pour p grand est très nette. Les erreurs introduites par l'annulation numérique à cause du mauvais choix de calcul de la racines sont donc beaucoup plus grandes que les erreurs introduites à cause du mauvais conditionnement du problème autour de 1.