POLYTECH SORBONNE

SCIENCES SORBONNE UNIVERSITÉ

Rapport TP 1

encadré par

Stéphane Graillat

Arithmétique en virgule Flottante et Analyse des Erreurs

réalisé par

Clément Apavou

Arthur Zucker

Mathématiques Appliquées et Informatique

3e année de cycle ingénieur

23 octobre 2020

Paris, France

Exercice 1 : Compensated Horner-Scheme

Question 1 : Schéma de Horner classique

1.1.a Le schéma de `Horner` pour l'évaluation d'un polynôme est donnée dans le listing suivant :

```
function res = Horner(p,x)
p = fliplr(p);
n = length(p);
a = p;
s(n) = a(n);
for i = (n-1) : (-1) : 1
    p(i) = s(i+1) * x;
    s(i) = p(i) + a(i);
end
res = s(1);
end
```

La fonction `fliplr` permet d'inverser l'ordre des coefficients du vecteur afin d'obtenir les mêmes résultats que pour la fonction `polyval`. La représentation d'un polynôme dans matlab est probablement inverse.

1.1.b Effectuons une comparaison des performance de l'évaluation d'Horner sur le polynôme $p = 1 + 2x^2 + 3x^3 + 4x^4 + 5x^5 + 6x^6 + 7x^7 + 8x^8 + 9x^9$ évalué en $x = \pi$

```
format long
p = [1 2 3 4 5 6 7 8 9];
x = pi;
res2 = Horner (p,x)
```

```
res2 = 2.041366687877934e+04
```

La fonction `polyval` fournie par matlab donne un résultat similaire :

```
res1 = polyval(p,x)
```

```
res1 = 2.041366687877934e+04
```

Question 2 : Error-Free Transformations (EFT)

Les algorithmes de EFT permettent d'effectuer des opérations entre deux flottants en prenant en compte l'erreur. L'implémentation des algorithmes dit "EFT" est la suivante :

```
function [x,y] = FastTwoSum(a,b)
x =  a+b;
y = (a-x)+b;
end
```

```
function [x,y] = TwoSum(a,b)
x = a+b;
z = x-a;
y = (a-(x-z))+(b-z);
end
```

```
function [x, y] = Split(a)
factor = 2^(27) +1;
c = factor * a;
x = c - (c - a);
y = a-x;
end
```

```
function [x,y] = TwoProduct(a,b)
x = (a*b);
[a1,a2]=Split(a);
[b1,b2]=Split(b);
y = (a2*b2 - (((x-(a1*b1))-(a2*b1))-(a1*b2)));
end
```

Question 3 : Schéma de Horner compensé

La version compensé du schéma de Horner permet d'effectuer une évaluation de Horner en prenant compte l'erreur grâce aux algorithmes de EFT. Une version compensée de l'évaluation d'Horner est implémentée dans le listing suivant :

```
function res = CompHorner(p,x)
p = fliplr(p);
n = length(p);
a = p;
s(n) = a(n);
r(n) = 0;
for i = (n-1) :(-1) : 1
    [p(i),pi(i)] = TwoProduct(s(i+1),x);
    [s(i),si(i)] = TwoSum(p(i),a(i));
    r(i) = (r(i+1)*x+(pi(i)+si(i)));
end
res = s(1) + r(1);
end
```

En reprenant l'exemple précédent, on trouve l'évaluation suivante :

```
res3 = CompHorner(p,x)
```

```
res3 = 2.041366687877935e+04
```

Question 4 : Schéma de Horner en valeur exacte

En utilisant la fonction sym de la bibliothèque Symbolic Math Toolbox, la version exacte de l'évaluation de Horner est donnée par le code suivant :

```
function res = sHorner(p,x)
p = sym(p,'f')
x = sym(x,'f')
p = fliplr(p);
n = length(p);
a = p;
s(n) = a(n);
for i = (n-1) : (-1) : 1
    p(i) = s(i+1) * x;
    s(i) = p(i) + a(i);
end
res = s(1);
end
```

Question 5 : Conditionnement

La fonction condp calcul le conditionnement de p pour une évaluation en x. On utilise ici la fonction sym de la bibiliothèque Symbolic Math Toolbox pour avoir une valeur exacte.

```
function condp = condp(p,x)
p=sym(p,'f');
x= sym(x,'f');
s1 = 0;
s2 = 0;
for i=1:length(p)
    s1 = s1 + abs(p(i))*(abs(x)^i);
    s2 = s2 + p(i)*(x^i);
end
s2 = abs(s2);
condp = s1/s2;
end
```

Question 6 : Comparaison de la précision des méthodes

Afin de tester les fonctions précédemment implémenté, nous les avons testé sur les polynômes $p_n(x) = (x-1)^n$ pour $n = [3, ..., 42]$ en prenant le flottant $x = 1.333$, et avons obtenu les résultats suivants :

```
n = 3:42;
x = 1.333;
for i=1:length(n)
    a = poly(ones(1,n(i)));
    res1 = Horner    (a,x);      % Classic horner
    res2 = sHorner   (a,x);      % Thorical value
    res3 = CompHorner(a,x);      % Compensated Horner
    er1(i) = abs(res2-res1)/abs(res2);
    er2(i) = abs(res2-res3)/abs(res2);
    c(i) = condp(a,x);
    %c(i) = abs((x+1)/(x-1))^i;
    cond1(i) = 2*n(i)*eps*c(i);
    cond2(i) = eps + 4*n(i)*n(i)*eps*eps*c(i);
end
```

Sur le graphe 1, les majorations théorique pour les deux schéma sont bien respectées.

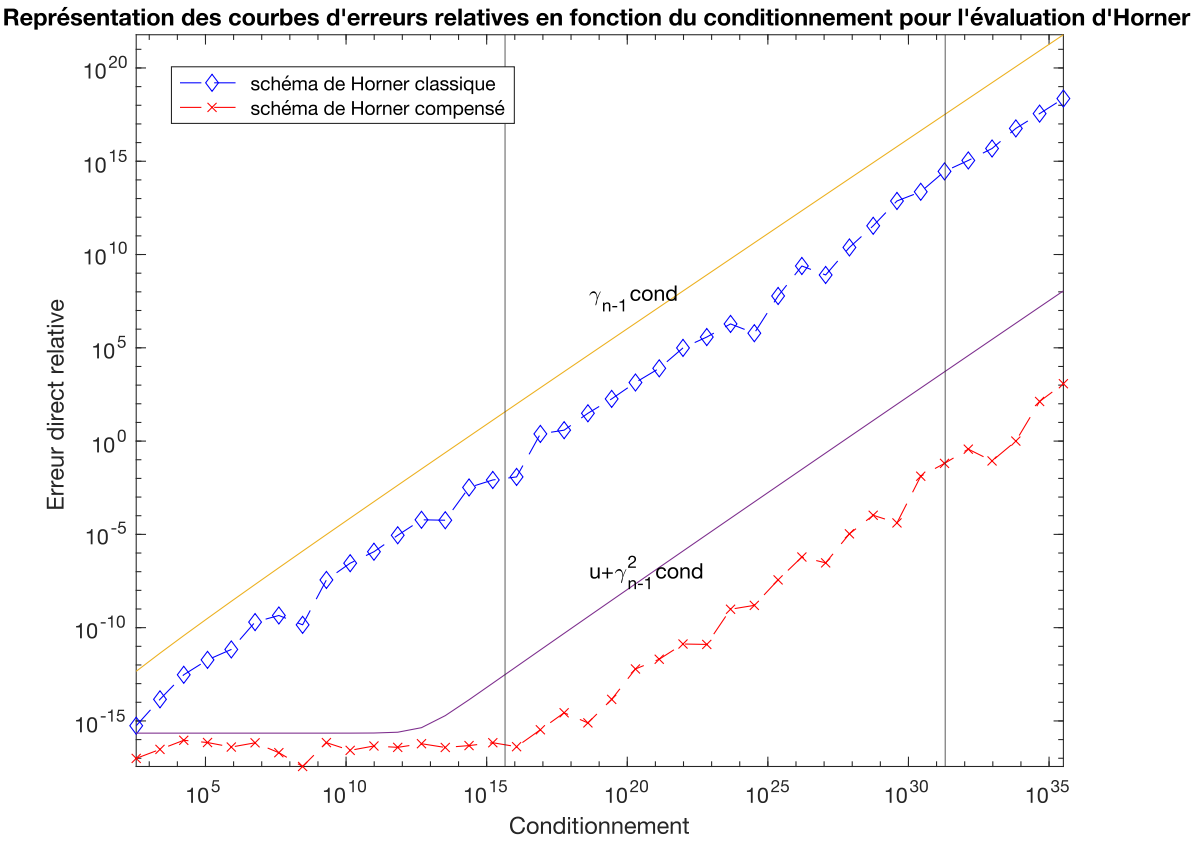

FIGURE 1 – Comparaison des erreurs pour l'évaluation d'Horner

Exercice 2 : Précisions des algorithmes de sommation

Question 1 : Les différents algorithmes de sommation

2.1.a L'algorithme de sommation récursif classique donne le code suivant :

```
function res = Sum(p)
n = length(p);
o = 0;
for i = 1:n
    o = o + p(i);
end
res = o;
end
```

2.1.b L'algorithme de sommation de Kahan correspond au code qui suit :

```
function res = SCompSum(p)
n=length(p);
o = 0;
e = 0;
for i = 1:n
    y = p(i) + e;
    [o, e] = FastTwoSum(o, y);
end
res = o;
end
```

2.1.c L'algorithme de double compensation de la somme de Priest est implémenté avec le code suivant :

```
function res = DCompSum(p)
n=length(p);
[out,idx] = sort(abs(p),'descend');
p = p(idx);
s = 0;
c = 0;
for i = 1:n
    [y, u] = FastTwoSum(c, p(i));
    [t, v] = FastTwoSum(y, s);
    z = u + v;
    [s, c] = FastTwoSum(t, z);
end
res = s;
end
```

2.1.d La méthode de somme compensée de Rump est présentée ci-après :

```
function res = CompSum(p)
n=length(p);
pi = zeros(1,n);
o = zeros(1,n);
pi(1) = p(1);
o(1)= 0;
q = zeros(1,n);
for i = 2:n
    [pi(i),q(i)] = TwoSum(pi(i-1), p(i));
    o(i) = o(i-1) + q(i);
```

```
11 end
12 res = pi(n)+o(n);
13 end
```

Question 2 : Comparaison de la précision

Étudions la précision des différentes méthodes présentées ci-dessus en fonction du conditionnement de la somme.

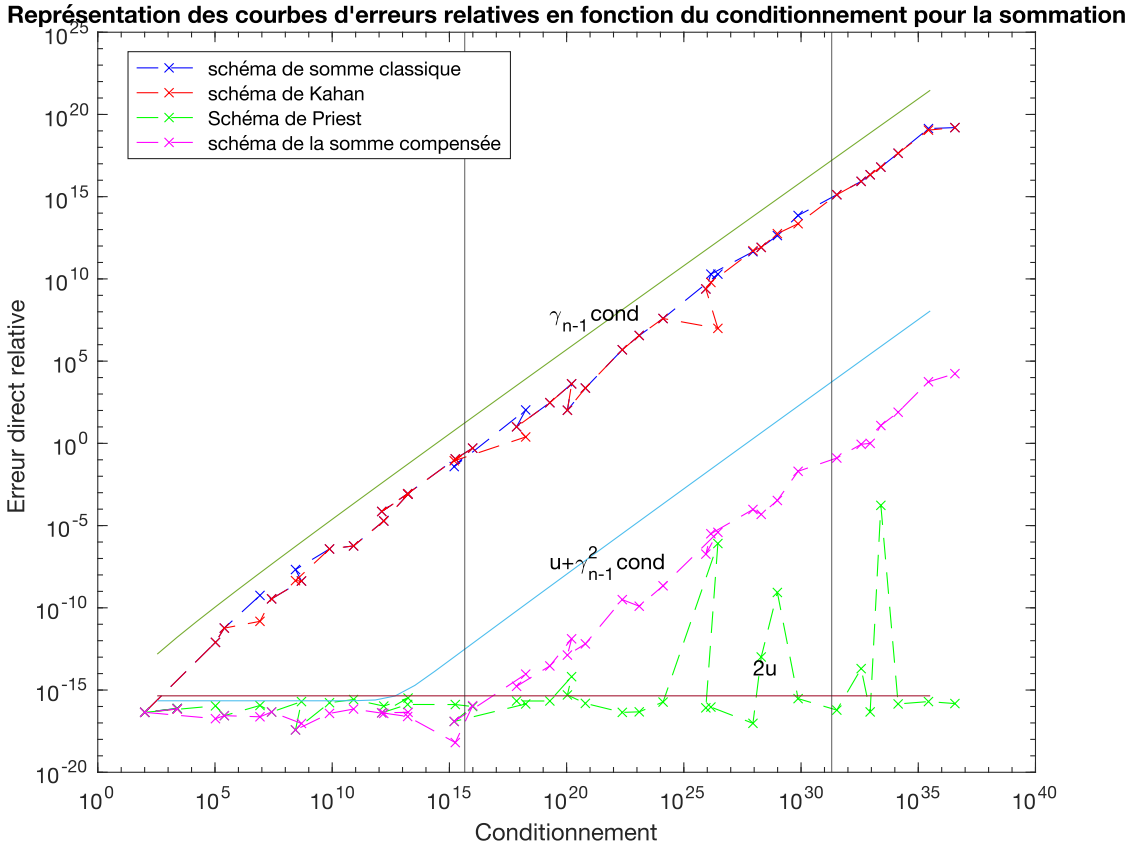

FIGURE 2 – Comparaison des erreurs sur les méthodes de sommations

Comme pour l'évaluation d'Horner, la méthode de sommation compensée est la plus intéressante (sans prendre en compte la méthode de Priest qui requière de trier le tableau d'entrée), on voit que son erreur commence à réellement perdre en précision lorsque le conditionnement dépasse $\frac{1}{u}$ avec u la double précision de l'ordinateur utilisé pour faire les calculs, ici $u = 2^{-52}$.

On observe juste quelques valeurs d'erreur un peu anormal pour `DCompSum`, qui est censée

rester en dessous de la droite d'équation $2u$ selon le théorème du cours. Ces valeurs sont probablement liées à un mauvais calcul au sein de la fonction.

L'algorithme de sommation de Kahan n'est pas très intéressant puisqu'il donne à peu près les mêmes résultats de performances que l'algorithme de sommation classique.