Calculer avec des intervalles : une activité symbolique

Ecole des Journées Doctorales MACS 2009 - Angers

Yves.Papegay@sophia.inria.fr
INRIA Sophia Antipolis Méditerranée

Le calcul avec des intervalles permet la certification de résultats mais souvent au prix d'une surestimation des incertitudes sur ces résultats. Ce problème est essentiellement d'origine symbolique. Une idée originale permet - au moins partiellement - de le contourner : elle consiste à implémenter les algorithmes de calcul par intervalles de sorte que les calculs restent formels le plus longtemps possible et permettent un préconditionnement des expressions générées en cours d'exécution. Dans cette intervention, nous montrerons l'importance des gains d'efficacité apportés au calcul par intervalles par ces approches symboliques, mais aussi nous étudierons en détails l'ensemble des apports du calcul symbolique au calcul par intervalles.

---

1. Quelques considérations arithmétiques

Habitués que nous sommes à calculer dans des arithmétiques exactes - symboliques, entières ou rationelles - nous avons tendance à oublier que les ordinateurs et autres calculateurs electroniques, malgré leur omnipotence et omniscience affichée ne savent représenter - et encore moins manipuler - qu'une infime partie des nombres que l'on dit réels malgré leur abstraction : les **nombres flottants**.

● Nombres flottants

Les nombres flottants sont caractérisés par deux entiers qui définissent la **base** et la **précision** de leur représentation.

Ainsi, en base 10 et avec une précision de 6, π est représenté par 3.14159.

```
In[7]:= Pi // N
Out[7]= 3.14159
```

La plupart des machines représentent les nombres en base 2 - avec 52 bits pour la mantisse, 1 bit pour le signe et 11 bits pour l'exposant - ce qui a pour conséquence directe que seule une partie des nombres décimaux est représentable.

○ exemple de 0.1

Par exemple, en base 2, le nombre décimal 0.1 est représenté par

```
In[8]:= BaseForm[0.1`16, 2]
```

Out[8]//BaseForm= $0.0001100110011001100110011001100110011001100110011001100110011010_2$

or, si l'on calcule exactement ce que cette représentation vaut exactement

```
In[9]:= RealDigits[
         0.0001100110011001100110011001100110011001100110011001100110-
           10]
        ToNumber[%, 2]
        N[%, 60]
```

```
Out[9]= {{1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
           0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
           1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0}, -3}
```

$$\text{Out[10]=}\quad \frac{3\,602\,879\,701\,896\,397}{36\,028\,797\,018\,963\,968}$$

```
Out[11]= 0.1000000000000000055511151231257827021181583404541015625-
          00000
```

ce qui diffère légèrement de 0.1.

○ plus petit nombres

Cette représentation est par essence discrète et sa granularité dépends du nombre total de bits utilisés pour représenter un nombre (32 ou 64) et de la répartition des bits entre le signe, la mantisse et l'exposant.

```
In[12]:= $MachineEpsilon
```

Out[12]= 2.22045×10^{-16}

```
In[13]:= N[2^-52]
```

Out[13]= 2.22045×10^{-16}

Elle est habituellement définie comme le plus petit nombre qui ajouté à 1 produit un résultat strictement supérieur à 1.

```
In[14]:= x = 1. + $MachineEpsilon
         y = 1. + ($MachineEpsilon / 2)
```

Out[14]= 1.

Out[15]= 1.

```
In[16]:= x - 1.
         y - 1.
```

Out[16]= 2.22045×10^{-16}

Out[17]= 0.

Elle n'est pas absolue, mais relative au nombre au voisinage duquel on la calcule, aussi peut-on représenter des nombres plus petit que cette granularité et notamment, elle ne doit pas être confondue avec le plus petit nombre représentable.

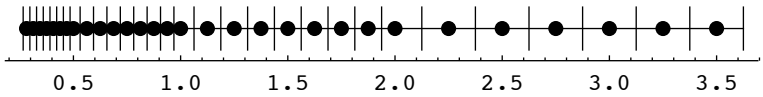
```
In[18]:= $MinMachineNumber
         RealDigits[%, 2]
```

Out[18]= 2.22507×10^{-308}

```
Out[19]= {{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, -1021}
```

Intervalles et arrondis

De fait, cette répartition discrète des nombres représentables en machine le long de la droite des réels conduit à associer à chaque nombre un intervalle, et à systématiquement arrondir tout réel au nombre représentable le plus proche.



Et l'implémentation de l'arithmétique et des fonctions numériques usuelles doit (ou devrait !) tenir compte de cette association, tant au niveau des arrondis de calculs que de la signification des résultats - ce qui n'est pas en général le cas.

○ arrondis

Voici deux exemples où l'arithmétique est prise en flagrant délit de ne pas considérer 1 comme le petit intervalle de la droite réelle qu'il repreésente.

```
In[20]:= x = 3 (1 + $MachineEpsilon / 2)
Out[20]= 3.
In[21]:= x - 3
Out[21]= 0.
In[22]:= x = (1 + $MachineEpsilon / 2) ^ 2
Out[22]= 1.
In[23]:= x - 1
Out[23]= 0.
```

○ signification

Dans cette exemple, la largeur de l'intervalle associé à 2.22×10^{25} est de l'ordre de 10^{9}, ce qui enlève tout sens au calcul du sinus.

```
In[24]:= Sin[2.22 × 10^25]
         Sin[2.22 × 10^25 + Pi / 2]
         Sin[2.22 × 10^25 + Pi]
Out[24]= 0.34318
Out[25]= 0.34318
Out[26]= 0.34318
```

● Arithmétique par intervalles

L'arithmétique par intervalle est mathématiquement bien définie comme une extension ensembliste de l'arithmétique sur les réels.
Ainsi, par exemple, la somme de deux intervalles est le plus petit (au sens de l'inclusion) intervalle qui contient la somme de deux nombres quelconques de ces deux intervalles.

Une définition similaire permet d'étendre les foncions numériques usuelles aux intervalles.

○ réels et intervalles

On peut associer chaque réel au plus petit intervalle dont les bornes sont des nombres représentables en machine qui le contient.

```
In[27]:= Interval[1.] // InputForm
Out[27]//InputForm= Interval[{0.9999999999999998, 1.0000000000000002}]
```

Et si l'implémentation de l'arithmétique par intervalle est correcte - c'est à dire prends en compte les arrondis, et c'est en général le cas - alors on obtient des résultats qui retrouvent tout leur sens et sont intrinsèquement justes.

```
In[28]:= 3 Interval[1.] - 3
Out[28]= Interval[{-1.33227 × 10^-15, 1.33227 × 10^-15}]

In[29]:= Sin[Interval[2.22 × 10^25]]
Out[29]= Interval[{-1, 1}]
```

● Un calcul pour frémir

L'instabilité numérique de certains calculs est parfois intrinsèque aux expressions que l'on évalue, parfois relié à la manière dont on conduit cette évaluation, mais rarement prévisible simplement ou évidente. En voici un exemple qui illustre bien la nécessité de certifier les résultats des calculs.

Le polynome ci-dessous est dû a Rump.

```
In[30]:= RumpFunc[x_, y_] := (1335 / 4 - x^2) y^6 +
           x^2 (11 x^2 y^2 - 121 y^4 - 2) + (11 / 2) y^8 + x / (2 y)
         RumpFunc[77 617, 33 096]
         % // N
Out[31]= -54 767 / 66 192
Out[32]= -0.827396
```

Ce premier calcul est exacte puisqu'il est réalisé avec des nombre entiers et rationnels. Voici le même calcul avec des nombres flottants.

```
In[33]:= RumpFunc[77 617., 33 096.]
Out[33]= 0.
```

Plus troublant encore est le résultat obtenu si l'on remplace 11/2 par 5.5 dans la définition du polynome.

```
In[34]:= RumpFuncN[x_, y_] := (1335 / 4 - x^2) y^6 +
           x^2 (11 x^2 y^2 - 121 y^4 - 2) + (5.5) y^8 + x / (2 y)
         RumpFuncN[77 617, 33 096]
Out[35]= 1.18059 × 10^21
```

Le recours a une arithmétique d'intervalles permet d'obtenir un résultat juste et de comprendre le phénomène ... même si il n'est pas très intéressant d'un point de vue calculatoire.

```
In[36]:= RumpFunc[Interval[77 617.], Interval[33 096.]]
Out[36]= Interval[{-3.89595 × 10^22, 3.65983 × 10^22}]
```

2. Arithmétique par intervalles et évaluation des expressions

Propriétés de l'arithmétique

En étant étendues aux intervalles, la soustraction et la division perdent leurs rapports priviligiés respectifs à l'addition et à la multiplication qui, du coup ne sont plus des lois de groupe.

élément symétrique

Notamment, il n'existe pas d'élément symétrique pour l'addition et la propriété $x - x = 0$ ne tient plus pour x intervalle.

```
In[37]:= -Interval[{1, 2}]
Out[37]= Interval[{-2, -1}]
In[38]:= Interval[{1, 2}] - Interval[{1, 2}]
Out[38]= Interval[{-1, 1}]
```

Un phénomène similaire se produit avec la division.

```
In[39]:= Interval[{2, 5}] / Interval[{2, 5}]
Out[39]= Interval[{2/5, 5/2}]
```

distributivité

La multiplication des intervalles n'est pas distributive par rapport à l'addition.

```
In[40]:= Interval[{-1, 2}] (Interval[{-3, 2}] + Interval[{4, 5}])
Out[40]= Interval[{-7, 14}]
In[41]:= (Interval[{-1, 2}] Interval[{-3, 2}]) +
          (Interval[{-1, 2}] Interval[{4, 5}])
Out[41]= Interval[{-11, 14}]
```

Evaluation des expressions

A toute fonction numérique, définie par une expression symbolique - par exemple un polynome - on peut associer une extension aux intervalles comme on l'a fait précédemment avec les opérateurs arithmétiques ou avec les fonctions usuelles.

Toutefois, deux expressions symboliquement équivalentes - et qui donc définissent la même fonction numérique produisent en général deux fonctions d'évaluation par intervalles différentes.

```
In[45]:= Clear[x]
         p[x_] := (x - 2) (x + 3);
         Expand[p[x]]
         HornerForm[p[x]]
Out[47]= -6 + x + x^2
Out[48]= -6 + x (1 + x)
```

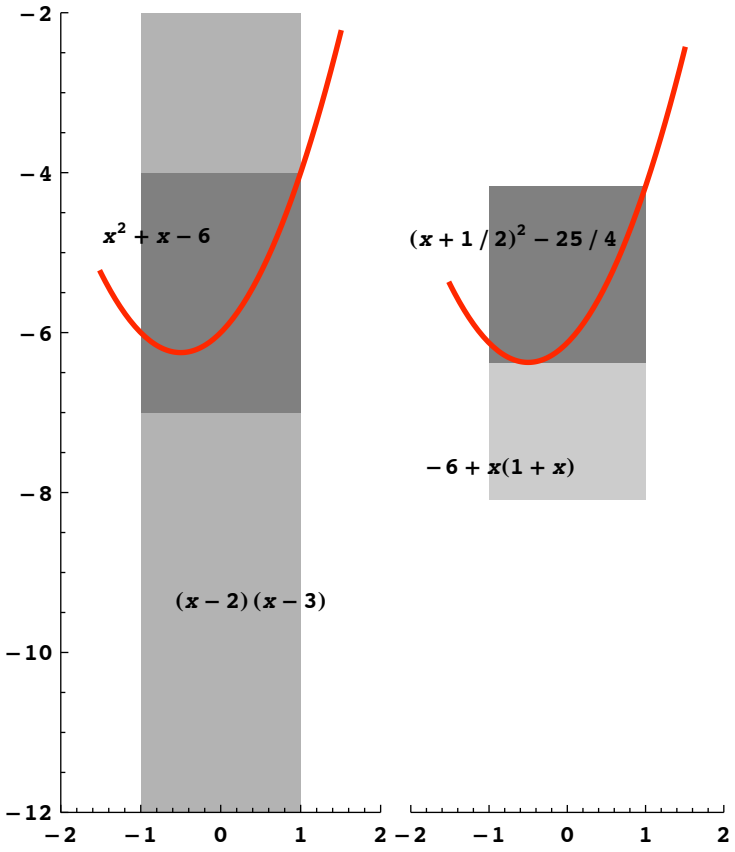


Fig. 1: évaluations comparées d'un polynome

Remarquons que l'évaluation de p[x] par l'expression $(x+1/2)^2 - 25/4$ est minimale car les bornes sont prises par des valeurs de la fonction. Un résultat classique (de l'analyse par intervalles) établit qe c'est le cas si chacune des variables de l'expression qui prends des valeurs intervalles apparaît une fois seulement dans l'expression. Ce résultat assez intuitif est le seul dont on dispose.

3. Prétraitement symbolique pour l'évaluation

Ce que l'on vient de voir nous conduit à essayer de trouver, parmi l'ensemble (infini) des expressions symboliques équivalentes associé à une fonction numérique, celle qui à la meilleure évaluation, c'est à dire aux intervalles résultats les plus étroits.

● Nombre d'occurences des variables

Une première piste à explorer concerne le nombre d'occurences de chaque variable. On considère l'exemple suivant

```
In[58]:= Clear[y]
         p1 = -6 x^3 + 4 x^3 z^3 + 15 x y^3 z^3 -
             3 y^5 z^3 - 12 x^2 y z^3 - 3 x y z^3 + y^3 z^3;
         b = {Interval[{-1., 2.}], Interval[{-2., 1.}],
             Interval[{-1.5, 1.5}]};
         IEval[p1, {x, y, z}, b]
Out[61]= Interval[{-1681.5, 1639.5}]

In[62]:= NbOcc[p1, {x, y, z}]

Out[62]= {5, 5, 6}
```

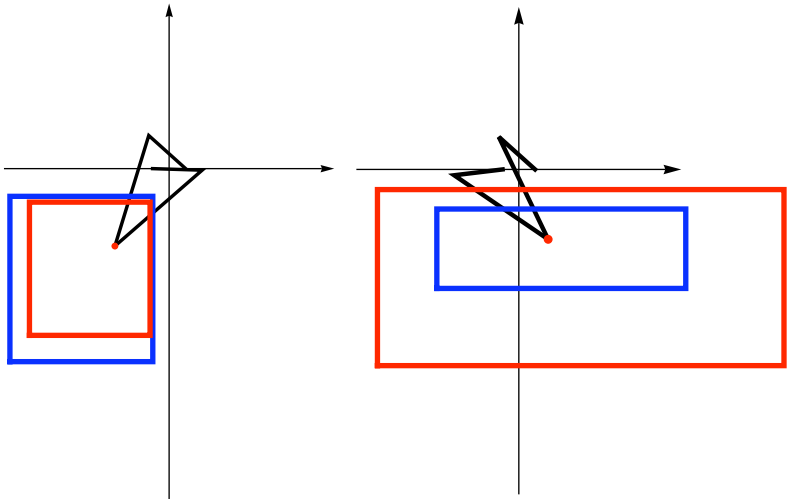
On peut essayer de minimiser ces nombres d'occurences - ou leur total.

```
In[63]:= p1s = OccSimplify[p1]
         NbOcc[p1s, {x, y, z}]
         IEval[p1s, {x, y, z}, b]
Out[63]= -6 x^3 + (4 x^3 - 3 x (1 + 4 x) y + (1 + 15 x) y^3 - 3 y^5) z^3

Out[64]= {5, 3, 1}

Out[65]= Interval[{-1222.5, 1180.5}]
```

Dans ce cas le résultat est intéressant, mais ce n'est pas systématique.



Monotonie

Une autre piste consiste à évaluer les dérivées de l'expression pour analyser sa monotonie, afin de pouvoir appliquer le cas échéant la propriété suivante :

> si f est croissante - resp. décroissante sur l'intervalle $[a, b]$, alors
> $f([a, b])=[f(a), f(b)]$ - resp. f([a, b]) = [f(b), f(a)]

Essayons avec l'exemple ci-dessous :

```
In[66]:= p2 = -6 x^3 - 4 x^3 z^3 + 15 x y^3 z^3 +
            3 y^5 z^3 + 12 x^2 y z^3 + 3 x y z^3 + y^3 z^3;
          b = {Interval[{0.5, 1}], Interval[{0.5, 1.}],
            Interval[{0.5, 1.}]}
          IEval[p2, {x, y, z}, b]

Out[66]= {Interval[{0.5, 1}],
           Interval[{0.5, 1.}], Interval[{0.5, 1.}]}

Out[67]= Interval[{-9.57422, 33.1875}]
```

On dérive l'expression par rapport à chacune des variables concernées, et on évalue les dérivées.

```
In[68]:= IEval[D[p2, x], {x, y, z}, b]
         IEval[D[p2, y], {x, y, z}, b]
         IEval[D[p2, z], {x, y, z}, b]

Out[68]= Interval[{-28.8281, 37.125}]

Out[69]= Interval[{1.47656, 78.}]

Out[70]= Interval[{-9.44531, 101.625}]
```

Comme l'intervalle résultat de l'évaluation de la dérivée de l'expression par rapport à y est strictement positif, on peut conclure à la croissance du polynome sur l'intervalle considéré. Ainsi le minimum (resp. le maximum) sera atteint pour la valeur minimale (resp. maximale) de y.

```
In[71]:= p2m = p2 /. y → 0.5
         p2M = p2 /. y → 1.

Out[71]= -6 x^3 + 0.21875 z^3 + 3.375 x z^3 + 6. x^2 z^3 - 4 x^3 z^3

Out[72]= -6 x^3 + 4. z^3 + 18. x z^3 + 12. x^2 z^3 - 4 x^3 z^3
```

○ borne inférieure

On considère maintenant le polynome en deux variables qui va nous fournir le minimum,

```
In[73]:= p2ms = OccSimplify[p2m]

Out[73]= -6 x^3 - 4. (-1.94763 + x) (0.0754419 + x) (0.372193 + x) z^3
```

après simplification, ce polynome s'avère croissant

```
In[74]:= D[p2ms, z] /. z -> b[[3]] /. x -> b[[1]]

Out[74]= Interval[{1.42684, 25.6355}]
```

on peut donc donner à z la valeur minimale

```
In[75]:= p2mm = p2ms /. z -> 0.5

Out[75]= -6 x^3 - 0.5 (-1.94763 + x) (0.0754419 + x) (0.372193 + x)
```

puis recommencer le même processus

```
In[76]:= D[p2mm, x] /. x -> b[[1]]

Out[76]= Interval[{-18.0519, -2.97931}]
```

et obtenir la borne minimale de la fonction de départ.

```
In[77]:= p2mmm = p2mm /. x -> 1.

Out[77]= -5.30078
```

○ borne supérieure

C'est ensuite au tour du polynome en deux variables qui va nous fournir le maximum,

```
In[78]:= p2Ms = OccSimplify[p2M]

Out[78]= -6 x^3 - 4. (-4.14411 + x) (0.2789 + x) (0.865208 + x) z^3
```

un processus similaire - à une simplification près - permet de venir à bout des calculs.

```
In[79]:= D[p2Ms, z] /. z -> b[[3]] /. x -> b[[1]]
         p2MM = p2Ms /. z -> 1.
         p2MMs = OccSimplify[p2MM]
         D[p2MMs, x] /. x -> b[[1]]
         p2MMM = p2MMs /. x -> 1.

Out[79]= Interval[{10.03, 104.312}]

Out[80]= -6 x^3 - 4. (-4.14411 + x) (0.2789 + x) (0.865208 + x)

Out[81]= -10. (-2.13219 + x) (0.293923 + x) (0.638266 + x)

Out[82]= Interval[{0.678118, 38.8219}]

Out[83]= 24.
```

Le résultat final est un interval environ 2 fois moins large que celui obtenu avec l'évaluation "naturelle".

```
In[84]:= Interval[{p2mmm, p2MMM}]

Out[84]= Interval[{-5.30078, 24.}]
```

● Formules de Taylor

Lorsque l'on sait dériver, on peut aussi utiliser une version adaptée aux intervalles du développement de Taylor pour évaluer une fonction.

○ premier ordre

$$[f]([x]) = f(m) + \left[\frac{\partial f}{\partial_x}\right]([x])([x] - m) \qquad (1)$$

Dans la formule ci-dessus, le premier terme est l'évaluation de la fonction au point milieu de l'intervalle, le deuxième le produit de l'intervalle recentré par l'extension de sa dérivée aux intervalles évaluée sur l'intervalle tout entier (!).

Pour traiter un exemple à plusieurs variables, on itère le procesus sur chacune des variables :

```
In[85]:= p1 /. x -> 0.75
         D[p1, x]
         D[p1, x] /. x -> b[[1]]
         b[[1]] - 0.75
         p1x =
          (p1 /. x -> 0.75) + (D[p1, x] /. x -> b[[1]]) (b[[1]] - 0.75)

Out[85]= -2.53125 + 1.6875 z^3 - 9. y z^3 + 12.25 y^3 z^3 - 3 y^5 z^3

Out[86]= -18 x^2 + 12 x^2 z^3 - 3 y z^3 - 24 x y z^3 + 15 y^3 z^3

Out[87]= -3 y z^3 + 15 y^3 z^3 + y z^3 Interval[{-24, -12.}] +
          Interval[{-18, -4.5}] + z^3 Interval[{3., 12}]

Out[88]= Interval[{-0.25, 0.25}]

Out[89]= -2.53125 + 1.6875 z^3 - 9. y z^3 +
          12.25 y^3 z^3 - 3 y^5 z^3 + Interval[{-0.25, 0.25}]
           (-3 y z^3 + 15 y^3 z^3 + y z^3 Interval[{-24, -12.}] +
             Interval[{-18, -4.5}] + z^3 Interval[{3., 12}])

In[90]:= p1xy =
          (p1x /. y -> 0.75) + (D[p1x, y] /. y -> b[[2]]) (b[[2]] - 0.75)

Out[90]= -2.53125 - 0.606445 z^3 + Interval[{-0.25, 0.25}]
           (4.07813 z^3 + z^3 Interval[{-18., -9.}] +
             Interval[{-18, -4.5}] + z^3 Interval[{3., 12}]) +
          Interval[{-0.25, 0.25}]
           (-9. z^3 + z^3 Interval[{-15., -0.9375}] +
             z^3 Interval[{9.1875, 36.75}] + Interval[{-0.25, 0.25}]
              (-3 z^3 + z^3 Interval[{-24, -12.}] +
                Interval[{0, 0}] + z^3 Interval[{11.25, 45.}]))

In[91]:= p1xyz = (p1xy /. z -> 0.75) +
            (D[p1xy, z] /. z -> b[[3]]) (b[[3]] - 0.75)

Out[91]= Interval[{-23.9144, 18.3402}]
```

C'est l'algorithme implémenté dans la fonction de bibliothèque **CenteredEval**.

```
In[92]:= NaturalEval[p1, {x, y, z}, b]
         CenteredEval[p1, {x, y, z}, b]

Out[92]= Interval[{-23.8047, 18.957}]

Out[93]= Interval[{-23.9144, 18.3402}]
```

Notons que le gain n'est pas très important , mais qu'il s'améliore grandement avec le rétrécissement de l'intervalle de départ - un phénomène de convergence bien connu des amateurs de formules de Taylor de tous poils.

```
In[94]:= b1 = {Interval[{1.9, 2}],
              Interval[{0.9, 1}], Interval[{1.4, 1.5}]};
         NaturalEval[p1, {x, y, z}, b1]
         CenteredEval[p1, {x, y, z}, b1]

Out[95]= Interval[{-106.08, 45.5503}]

Out[96]= Interval[{-40.6544, -21.6928}]
```

```
In[97]:= b2 = {Interval[{1.9, 1.91}],
              Interval[{0.9, 0.91}], Interval[{1.49, 1.5}]};
          NaturalEval[p1, {x, y, z}, b2]
          CenteredEval[p1, {x, y, z}, b2]
Out[98]= Interval[{-38.2786, -23.4789}]
Out[99]= Interval[{-31.3969, -30.3772}]
```

○ second ordre

$$[f]([x]) = f(m) + (x-m)\frac{\partial f}{\partial_x}(x) + \frac{([x]-m)}{2}\left[\frac{\partial^2 f}{\partial_x \partial_x}\right]([x]) \qquad (2)$$

On peut raffiner l'approximation de la fonction par une formule de taylor en augmentant l'ordre de cette formule. La fonction de bibliothèque **TaylorEval** implémente ce calcul avec la même approche que précedemment - variable après variable.

```
In[100]:= TaylorEval[p1, {x, y, z}, b]
          TaylorEval[p1, {x, y, z}, b1]
          TaylorEval[p1, {x, y, z}, b2]
Out[100]= Interval[{-11.4277, 5.22482}]
Out[101]= Interval[{-37.6537, -24.0756}]
Out[102]= Interval[{-31.3713, -30.3974}]
```

L'amélioration des résultats en terme de largeur d'intervalles est plus sensible lorsque l'intervalle d'évaluation est grand, et s'attenue au fur et à mesure de sa diminution.

4. Filtrage symbolique

L'évaluation est une étape cruciale qui intervient dans toutes les méthodes de l'analyse par intervalles. Mais des fonctionnalités de calcul symboliques - savoir manipuler, dériver des formules, faire des changements de variables - peuvent aussi aider à développer certaines de ces méthodes.

● Résolution de systèmes et filtrage

Un domaine de prédilection de l'analyse par intervalles est la résolution de systèmes d'équations. Il s'agit de déterminer, à l'intérieur d'un intervalle - resp. d'une boîte dans le cas d'un système à plusieurs variables - de départ, un ensemble d'intervalles (resp. de boites) suffisament petit(e)s où le système peut-être satisfait.

La méthode d'analyse d'intervalles la plus populaire pour attaquer ce problème est une méthode itérative de recherche globale basée sur une boucle de bissection et d'évaluation éventuellement associée à un filtrage.

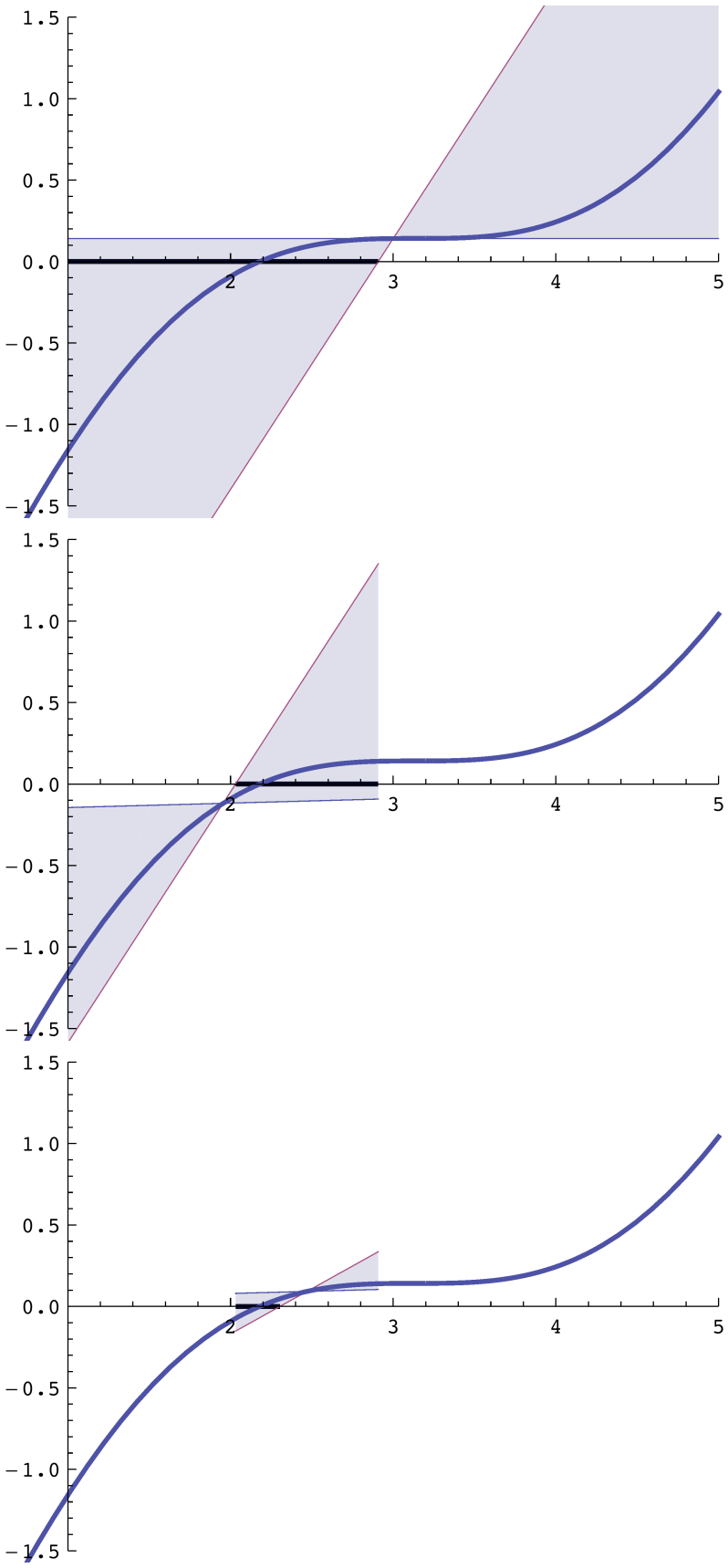
Un filtrage peut se définir comme l'opération suivante :

> Etant donné une boite b_i et un système à résoudre $f(X)=0$, trouver une boîte b_f incluse dans b_i telle que si il existe un élément x de b_i vérifiant $f(x)=0$, alors nécessairement, x est un élément de b_f.

Il existe des filtrages purement numériques, comme le filtrage de Newton. Prenons

un exemple,

```
In[103]:= f[x_] := Sin[x] + x - 3
```



Filtrage 2-B

Ce filtrage est par essence symbolique, même si souvent cet aspect est caché dans son implémentation qui s'appuie directement sur la représentation des systèmes d'équations.

L'idée est simple, il s'agit d'abord de décoreller les instances de chaque variables, en en introduisant de nouvelles pour obtenir des équations supplémentaires. Ensuite, on utilise ces équations supplémentaires pour filtrer.

Reprenons le même exemple,

```
In[104]:= f[x] == 0

Out[104]= -3 + x + Sin[x] == 0
```

On peut considérer la première instance de x comme indépendante de la deuxième, la remplacer par une autre variable, y.

```
In[105]:= {Solve[Replace[f[x], x -> y, {1}] == 0, y][[1, 1]] /.
            Rule -> Equal, x == y}

Out[105]= {y == 3 - Sin[x], x == y}
```

L'évaluation de la première équation donne une borne sur y en fonction de x, tandis que l'évaluation de la deuxième donne une borne - trivialement - sur x en fonction de y. Il suffit donc d'itérer.

```
In[106]:= Function[i, N[3 - Sin[x] /. x -> i]][Interval[{-10., 10.}]]

Out[106]= Interval[{2., 4.}]
```

Sur cet exemple, la convergence est longue.

```
In[107]:= NestList[Function[i, N[3 - Sin[x] /. x -> i]],
           Interval[{-10., 10.}], 25]

Out[107]= {Interval[{-10., 10.}], Interval[{2., 4.}],
           Interval[{2.0907, 3.7568}], Interval[{2.13213, 3.57713}],
           Interval[{2.15346, 3.4219}], Interval[{2.165, 3.27665}],
           Interval[{2.1714, 3.13465}], Interval[{2.17501, 2.99305}],
           Interval[{2.17705, 2.85201}],
           Interval[{2.17821, 2.71444}],
           Interval[{2.17887, 2.58572}],
           Interval[{2.17925, 2.47232}],
           Interval[{2.17947, 2.37958}],
           Interval[{2.17959, 2.30962}],
           Interval[{2.17966, 2.26074}],
           Interval[{2.1797, 2.22872}], Interval[{2.17973, 2.20874}],
           Interval[{2.17974, 2.19668}],
           Interval[{2.17975, 2.18955}], Interval[{2.17975, 2.1854}],
           Interval[{2.17975, 2.183}], Interval[{2.17976, 2.18161}],
           Interval[{2.17976, 2.18082}],
           Interval[{2.17976, 2.18037}],
           Interval[{2.17976, 2.18011}],
           Interval[{2.17976, 2.17996}]}
```

Remarquons qu'on aurait pu faire d'autre choix de variables, par exemple,

```
In[108]:= {Solve[Replace[f[x], Sin[x] -> z, {1}] == 0, x][[1, 1]] /.
            Rule -> Equal, z == Sin[x]}

Out[108]= {x == 3 - z, z == Sin[x]}
```

L' itération est un tout petit peu moins simple,

```
In[109]:= Function[i, N[3 - z /. z :> (Sin[x] /. x -> i)]][
           Interval[{-10., 10.}]]
Out[109]= Interval[{2., 4.}]
```

et sur cet exemple la convergence est identique ...

```
In[110]:= NestList[Function[i, N[3 - z /. z :> (Sin[x] /. x -> i)]],
           Interval[{-10., 10.}], 25]
Out[110]= {Interval[{-10., 10.}], Interval[{2., 4.}],
           Interval[{2.0907, 3.7568}], Interval[{2.13213, 3.57713}],
           Interval[{2.15346, 3.4219}], Interval[{2.165, 3.27665}],
           Interval[{2.1714, 3.13465}], Interval[{2.17501, 2.99305}],
           Interval[{2.17705, 2.85201}],
           Interval[{2.17821, 2.71444}],
           Interval[{2.17887, 2.58572}],
           Interval[{2.17925, 2.47232}],
           Interval[{2.17947, 2.37958}],
           Interval[{2.17959, 2.30962}],
           Interval[{2.17966, 2.26074}],
           Interval[{2.1797, 2.22872}], Interval[{2.17973, 2.20874}],
           Interval[{2.17974, 2.19668}],
           Interval[{2.17975, 2.18955}], Interval[{2.17975, 2.1854}],
           Interval[{2.17975, 2.183}], Interval[{2.17976, 2.18161}],
           Interval[{2.17976, 2.18082}],
           Interval[{2.17976, 2.18037}],
           Interval[{2.17976, 2.18011}],
           Interval[{2.17976, 2.17996}]}
```

... mais ce n'est pas toujours le cas.

Sous-expressions

Un autre moyen symbolique de générer des équations redondantes supplémentaires qui vont permettre de faire du filtrage est de s'appuyer sur la représentation des expressions et des sous-expressions.

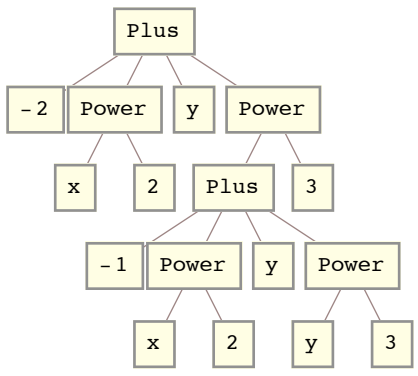
```
In[111]:= Eq = {x^2 + y + (y + x^2 + y^3 - 1)^3 - 2,
            ((x^2 + y^3) (x^2 + Cos[y]) + 14) / (x^2 + Cos[y]) - 8}
```

$$\text{Out[111]= } \left\{-2 + x^2 + y + \left(-1 + x^2 + y + y^3\right)^3,\ -8 + \frac{14 + \left(x^2 + y^3\right)\ \left(x^2 + \text{Cos}[y]\right)}{x^2 + \text{Cos}[y]}\right\}$$

Regardons la structure de la première équation.

```
In[112]:= Eq[[1]] // TreeForm
```

Out[112]//TreeForm=



On peut introduire, comme le ferait un compilateur, des variables intermédiaires pour représenter les sous-expressions.

```
In[113]:= Experimental`OptimizeExpression[Eq,
           OptimizationSymbol -> X, OptimizationLevel -> 2]

Out[113]= Experimental`OptimizedExpression[
           Block[{X$1183, X$1184, X$1185, X$1186, X$1187, X$1188,
             X$1190, X$1191, X$1192}, X$1183 = x^2; X$1184 = y^2;
            X$1185 = y X$1184; X$1186 = -1 + X$1183 + y + X$1185;
            X$1187 = X$1186^2; X$1188 = X$1186 X$1187;
            X$1190 = Cos[y]; X$1191 = X$1183 + X$1190;
            X$1192 = 1/X$1191; {-2 + X$1183 + y + X$1188,
             -8 + X$1192 (14 + (X$1183 + X$1185) X$1191)}]]
```

L'introduction de certaines de ces nouvelles variables fournit de nouvelles équations qui peuvent rendre le filtrage du système plus efficace. Le choix de ces variables et la possibilité de prévoir le gain induit est un problème de recherche en cours.

Méthodes algébriques

Dans le cas particulier de systèmes algébriques, on peut aussi faire appel à des méthodes algébriques complexes telles que les *bases de Gröbner* qui enrichissent les systèmes polynomiaux par des équations supplémentaires particulièrement adaptées à leur résolution.

```
In[114]:= GroebnerBasis[{x^25 + 2 x y + 1, x y - x}, {x, y}]

Out[114]= {-1 + y, 1 + 2 x + x^25}

In[115]:= GroebnerBasis[
           {x^2 + y^2 + z^2 - 1, x y - z + 2, z^2 - 2 x + 3 y}, {x, y, z}]

Out[115]= {1024 - 832 z - 215 z^2 + 156 z^3 - 25 z^4 + 24 z^5 + 13 z^6 + z^8,
           -11 552 + 2560 y + 2197 z + 2764 z^2 + 443 z^3 + 728 z^4 +
            169 z^5 + 32 z^6 + 13 z^7, -34 656 + 5120 x + 6591 z +
            5732 z^2 + 1329 z^3 + 2184 z^4 + 507 z^5 + 96 z^6 + 39 z^7}

In[116]:= GroebnerBasis[{x^2 + y^2 + z^2 - 1,
            x y - z + 2, z^2 - 3 + x, x - y^2 + 1}, {x, y, z}]

Out[116]= {1}

In[117]:= GroebnerBasis[{x^2 + y^2 + z^2 - 1, x y - z + 2}, {x, y, z}]

Out[117]= {4 - y^2 + y^4 - 4 z + z^2 + y^2 z^2,
           -2 x - y + y^3 + x z + y z^2, 2 + x y - z, -1 + x^2 + y^2 + z^2}
```

Une description, même intuitive de la théorie des bases de Gröbner dépasse le cadre de ce cours.

5. Exécution symbolique des algorithmes

C'est à la fois l'utilisation du calcul symbolique qui semble la plus prometteuse pour améliorer les algorithmes d'analyse par intervalles ... et celle qui est le moins connue - pour l'instant.

La plupart des algorithmes mettenent en oeuvre des calculs - notamment des calculs matriciels - qui sont actuellement menés numériquement soit avec des nombres soit avec des intervalles.

Par éxécution symbolique des algorithmes, on entends mener - tant que faire se peux - les calculs de façon symbolique, pour retarder les évaluations par intervalles et ainsi améliorer leur qualité.

Illustrons cette idée sur un exemple simple.

● Filtrage de Newton en dimension 2

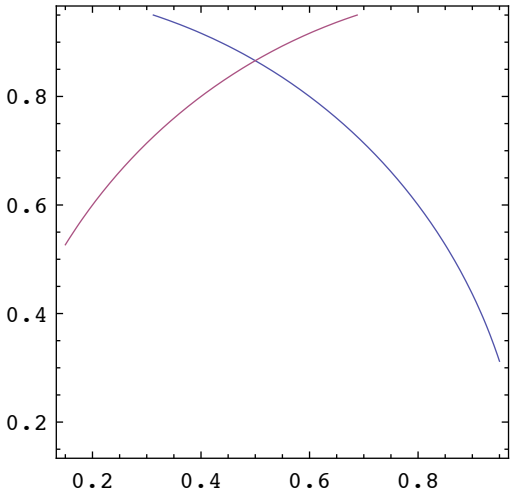
On considère le problème suivant,

```
In[118]:= g1[{x_, y_}] := x^2 + y^2 - 1
          g2[{x_, y_}] := (x - 1)^2 + y^2 - 1
          b = {Interval[{0.45, 0.65}], Interval[{0.75, 0.95}]};
```

qui consiste à localiser le point d'intersection ci-dessous.

```
In[121]:= ContourPlot[{g1[{x, y}] == 0, g2[{x, y}] == 0},
           {x, 0.15, 0.95}, {y, 0.15, 0.95}]
```

Out[121]=



Pour résoudre ce système, on va utiliser un filtrage de Newton bi-dimensionnel, en appliquant la formule suivante

$$b_{n+1} = \bigcap \left(b_n,\ \text{milieu}(b_n) - J(b_n)^{-1}\, g(\text{milieu}(b_n)) \right) \qquad (3)$$

où g représente le système et Jsa jacobienne.

```
In[122]:= (J = D[{g1[{x, y}], g2[{x, y}]}, {{x, y}}]) // MatrixForm
```

Out[122]//MatrixForm=

$$\begin{pmatrix} 2x & 2y \\ 2(-1+x) & 2y \end{pmatrix}$$

Un essai classique fournit un résultat catastrophique.

```
In[128]:= m = MidPoint[b]
          J /. {x -> b[[1]], y -> b[[2]]} // MatrixForm
          Inverse[J /. {x -> b[[1]], y -> b[[2]]}] // MatrixForm
          {g1[b], g2[b]}
          m - Inverse[J /. {x -> b[[1]], y -> b[[2]]}].{g1[b], g2[b]}

Out[128]= {0.55, 0.85}

Out[129]//MatrixForm=
( Interval[{0.9, 1.3}]   Interval[{1.5, 1.9}] )
( Interval[{-1.1, -0.7}] Interval[{1.5, 1.9}] )

Out[130]//MatrixForm=
( Interval[{0.328947, 0.791667}] Interval[{-0.791667, -0.328
( Interval[{0.153509, 0.458333}]  Interval[{0.197368, 0.5416

Out[131]= {Interval[{-0.235, 0.325}], Interval[{-0.315, 0.205}]}

Out[132]= {Interval[{0.0433333, 0.898333}],
           Interval[{0.59, 1.12833}]}
```

Si l'on mène les calculs symboliques plus loin, et notamment dans le calcul de l'inverse de la jacobienne - bien regarder les détails du code ci-dessus et ci-dessous - on obtient de meilleurs résultats qui ne sont , malheureusement sur cet exemple, pas suffisament bons pour assurer la convergence.

```
In[133]:= J /. {x -> b[[1]], y -> b[[2]]} // MatrixForm
          Inverse[J] /. {x -> b[[1]], y -> b[[2]]} // MatrixForm
          {g1[b], g2[b]}
          m - (Inverse[J] /. {x -> b[[1]], y -> b[[2]]}).{g1[b], g2[b]}

Out[133]//MatrixForm=
( Interval[{0.9, 1.3}]   Interval[{1.5, 1.9}] )
( Interval[{-1.1, -0.7}] Interval[{1.5, 1.9}] )

Out[134]//MatrixForm=
(                  1/2                                  -1/2
( Interval[{0.184211, 0.366667}] Interval[{0.236842, 0.43333

Out[135]= {Interval[{-0.235, 0.325}], Interval[{-0.315, 0.205}]}

Out[136]= {Interval[{0.23, 0.77}], Interval[{0.642, 1.07267}]}
```