Programmation

Les instructions Scilab sont sauvegardées dans deux types de fichiers :

• les fichiers **exécutables**, qui se terminent en .sce
on y écrit les lignes de commande telles quelles ;
on peut exécuter *une partie* ou *toutes* les commandes contenues dans le fichier ;

• les fichiers **d'instructions**, qui se terminent en .sci
on y définit *uniquement des fonctions* grâce à "`function`" ;
il faut d'abord charger les fonctions dans **Scilab** ;
si la syntaxe du fichier est correcte, rien n'est affiché,
mais les fonctions seront désormais définies et disponibles.

• le meilleur outil pour *créer* et *modifier* les fichiers **.sce** et **.sci** est l'éditeur de Scilab.
À partir du menu $\boxed{\text{Exécuter}}$ de l'éditeur, on peut **charger** ou **évaluer** le contenu du fichier édité.

Syntaxe de `function`

Variables en entrée : `x1,...,xm`
Variables en sortie : `y1,...,yn`

```
function [y1,...,yn] = nom_de_la_fonction(x1,...,xm)
        ...
     instructions
        avec
     variables internes z1,...,zp
        ...
endfunction
```

Remarque : il est utile d'enregistrer plusieurs fonctions dans un même fichier `mes_fonctions.sci` :
ceci permet de les charger toutes en même temps !

Fonctions : variables internes et externes

- une *variable en sortie* doit être définie dans la fonction ;
- les opérations sur les *variables internes* ne sont visibles qu'à l'intérieur de la fonction ;
- les *variables externes* sont visibles/utilisables dans une fonction ;
- sauf demande explicite (`return`, `resume`), les modifications sur *variables internes* ne sont pas visibles hors la fonction ;
- il peut y avoir aucune variable en entrée ou en sortie ;
- sauf demande explicite (`disp`, `printf`, `plot2d`,. . .), l'exécution d'une commande ne produit aucun affichage ;

Fonctions : exemple 1

```
function [S,D] = SOM_DIF(A,B)
   S = A+B
   D = A-B
   A = zeros(B), M= S/2
endfunction
```

```
--> U = [1,2;3,4]; V=U';      | --> [a,b] = SOM_DIF(U,V)
--> SOM_DIF(U,V)              |  b=
 ans=                         |    0.   - 1.
      2.    5.                |    1.     0.
      5.    8.                |  a=
--> U                         |    2.     5
 U  =                         |    5.     8.
      1.    2.                | --> M
      3.    4.                |  !--error 4
--> A                         |    Variable non définie : M
 !--error 4                   |
   Variable non définie : A   |
```

Fonctions : exemple 2

```
function affiche()
  x = linspace(0,1,10); y = x.^2
  scf(); plot2d(x,y)
  disp(x([$, 1]))
  [y_o, u] = resume(y([1,$]), 2)
endfunction
```

```
--> affiche()  // produit le graphe de x --> x.^2 et
    1.0   0.0
--> y
   !--error 4
     Variable non définie : y
--> y_o
y_o =
      0.0   1.0
--> u
u =
      2.0
```

Programmation : branchements et boucles

1. Les commandes `if`, `then`,. . . ,`end`
   permettent d'exécuter une partie de code en cas de test vrai ;
2. les commandes `select` et `case` permettent de sélectionner le code à exécuter en fonction des valeurs prises par une variable ou expression ;
3. la commande `for` permet d'obtenir une boucle **pour** toutes les valeurs de la variable de contrôle ;
4. la commande `while` exécute des instructions **tant que** le test est vrai.

Attention à la syntaxe précise de ces commandes :
virgules, passage à la ligne, . . .
Ne pas oublier les `end` : Scilab restera en attente s'il en manque un !

Évaluation de polynômes

Soit une fonction polynomiale $P(x)$ définie par :

$$P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \ .$$

Pour évaluer P au point $x = \alpha$ il faut :

- de façon naïve :
  n additions, n multiplications $a_k * \alpha^k, 1 \leq k \leq n$
  et $n-1$ appels à la fonction puissance $\alpha \mapsto \alpha^k$, $2 \leq k \leq n$,
  avec à chaque fois $k-1$ multiplications.
  Ce qui fait $O(n^2/2)$ multiplications.
- En calculant α^k à partir de α^{k-1}, $2 \leq k \leq n$,
  on réduit à $O(2n)$ multiplications.
- Est-ce que l'on peut mieux faire ?

Évaluation de polynômes : méthode de HORNER

On écrit $P(x) = Q(x)(x-\alpha) + P(\alpha)$

et l'on obtient les coefficients de $Q(x) = \sum_{i=0}^{n-1} q_i x^i$ par récurrence :

$$\begin{cases} q_{n-1} & = & a_n \\ q_{i-1} & = & \alpha q_i + a_i \ , \ i = n-1, \ldots, 0\,; \end{cases}$$

où $q_{-1} = P(\alpha)$.

Ce qui revient à une factorisation :

$$P(\alpha) = \underbrace{\alpha\,\underbrace{(\alpha\,(\alpha\,(\cdots \underbrace{(\alpha\,\underbrace{(\alpha\,\underbrace{a_n}_{q_{n-1}} + a_{n-1})}_{q_{n-2}} + a_{n-2})}_{q_{n-3}} \cdots) + a_2) + a_1)}_{q_0} + a_0}_{q_{-1}}$$

Évaluation de polynômes : méthode de HORNER

- En utilisant l'algorithme de HORNER, on obtient la valeur de $P(\alpha)$ en n multiplications.
- En **Scilab** la fonction `horner(P,x)` permet d'évaluer le polynôme P en x.

En dérivant la relation $P(x) = Q(x)(x - \alpha) + P(\alpha)$ on obtient :

$$P'(x) = Q'(x)(x - \alpha) + Q(x) \quad \text{et} \quad P'(\alpha) = Q(\alpha)$$

- Il suffit donc de rappliquer la méthode de Horner à Q pour obtenir $P'(\alpha)$.
- On peut généraliser et évaluer la valeur de $P^{(k)}(\alpha)$, la valeur de la dérivée d'ordre k de P en α, grâce à l'algorithme de Horner.

Évaluation de polynômes : précision

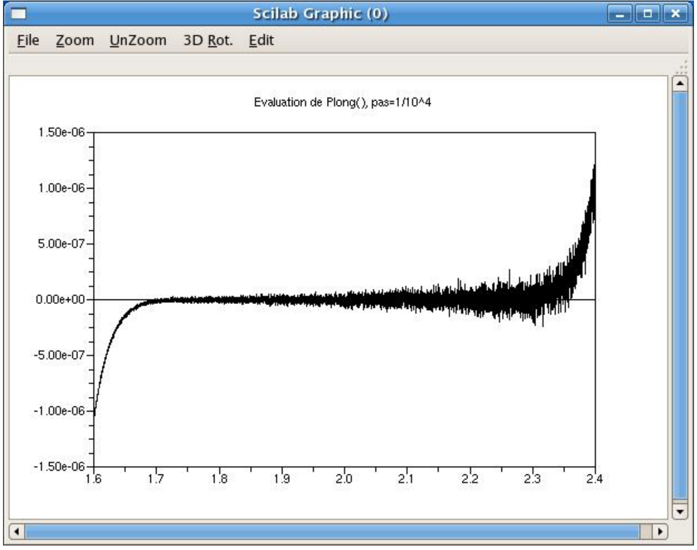
On s'intéresse à l'évaluation et la représentation de la fonction polynomiale $P(x) = (x - 2)^{15}$ sur $[1.6, 2.4]$ avec un pas de 10^{-4}. On procèder de trois façons différentes :

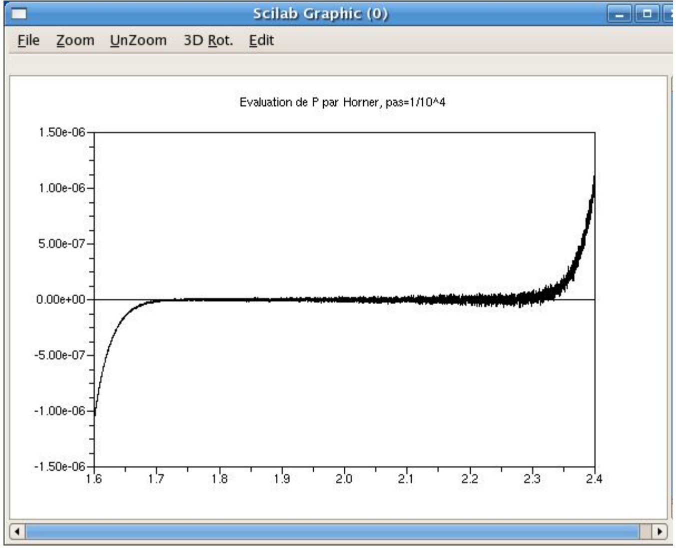
- on utilise une définition "explicite" de P :

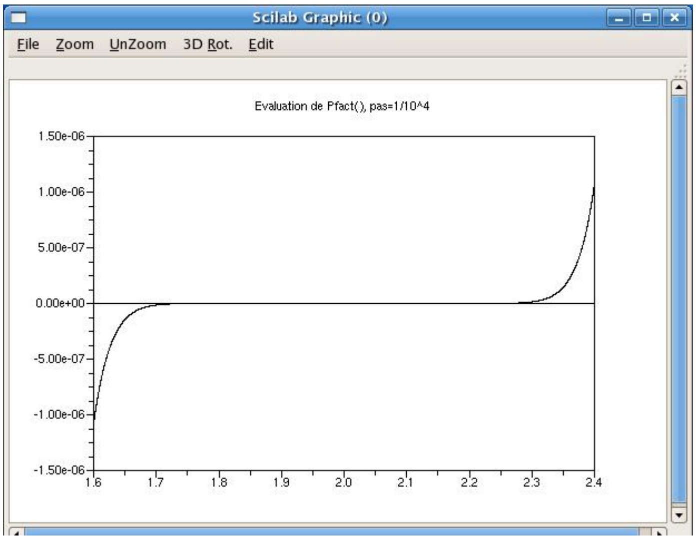
```
--> z = 2.*ones(1,15);
--> P = poly(z,"x")

                                    2           3           4
P =   - 32768 + 245760x - 860160x + 1863680x - 2795520x
                5           6            7          8          9
 + 3075072x - 2562560x  + 1647360x - 823680x + 320320x
              10        11        12       13      14    15
 - 96096x  + 21840x  - 3640x  + 420x  - 30x  + x
--> deff('[y]=Plong(x)','y='+pol2str(P))
```

- on applique la méthode de Horner à P ;
- on utilise la factorisation, de P :

```
--> deff('[y]=Pfact(x)','y=(x-2).^15')
```







Instabilité des racines d'un polynôme

On se propose de déterminer les racines des deux polynômes

$$P(x) = \prod_{k=1}^{20}(x-k) \qquad PW(x) = P(x) - 2^{-23}x^{19} ,$$

Le polynôme P est *perturbé* dans le coefficient de x^{19} de la quantité $2^{-23} \sim 10^{-7}$.

```
--> format("v",15)
--> P = poly([1:20],"x");
--> x = poly(0,"x")
--> PW = P - 2^(-23)*x^(19)
```