DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

BIBLIOTHÈQUE NUMÉRIQUE
MTH2210

Automne 2001

Ce guide comprend une description sommaire de la bibliothèque numérique qui est mise à votre disposition pour vous aider à compléter vos devoirs dans le cadre du cours MTH2210. Les algorithmes du livre de référence ont été programmés, ce qui vous permet de vous concentrer sur les problèmes à résoudre. Vous *devez* utiliser ces routines et MATLAB pour compléter vos devoirs. Ces routines sont disponibles sur le site Internet du cours. Pour ce qui est du fonctionnement de MATLAB, vous êtes invités à consulter le *Guide* MATLAB et les références qui y sont citées.

Liste des fonctions disponibles

- Chapitre 2: `bissect`, `ptfixe`, `newton`, `secante`
- Chapitre 3: `cond1`, `cond2`, `condinf`, `sysnl`
- Chapitre 5: `polynew`, `splinc`
- Chapitre 6: `trapeze`, `simp13`, `simp38`, `boole`, `romberg`, `intgauss`, `intspl`
- Chapitre 7: `euler`, `eulmod`, `ptmilieu`, `rk4`, `taylor`, `tir`, `traceqd`

Chapitre 2: Équations non-linéaires

2.2 La méthode de bissection: `bissect.m`

Appel de la fonction: `[x,err]=bissect(f,a,b,maxiter,arret,resul)`
Exemple d'appel: `[x,err]=bissect('fonc',1,2,20,1.0e-5,'resul.dat')`

Arguments en entrée:

`f`: Le nom, entre apostrophes, de la fonction stockée dans un fichier-M.

`a`, `b`: Les bornes inférieure et supérieure de l'intervalle initial.

`maxiter`: Le nombre maximal d'itérations.

`arret`: Le critère d'arrêt.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

Arguments en sortie:

`x`: Le vecteur des itérations.

`err`: S'il y a convergence, le vecteur $|x_n - x^*|$, où x^* est l'approximation calculée de la racine r.

2.3 La méthode des points fixes: `ptfixe.m`

Appel de la fonction: `[x,err]=ptfixe(g,x0,maxiter,arret,resul)`
Exemple d'appel: `[x,err]=ptfixe('g',2.0,30,1.0e-6,'resul.dat')`

Arguments en entrée:

`g`: Le nom, entre apostrophes, de la fonction stockée dans un fichier-M.

`x0`: L'approximation initiale.

`maxiter`: Le nombre maximal d'itérations.

`arret`: Le critère d'arrêt.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

Arguments en sortie:

`x`: Le vecteur des itérations.

`err`: S'il y a convergence, le vecteur $|x_n - x^*|$, où x^* est l'approximation calculée de la racine r.

2.4 La méthode de Newton: `newton.m`

Appel de la fonction: `[x,err]=newton(f,df,x0,maxiter,arret,resul)`
Exemple d'appel: `[x,err]=newton('fonc','dfonc',2.0,10,1.0e-6,'resul.dat')`

Arguments en entrée:

`f`: Le nom, entre apostrophes, de la fonction stockée dans un fichier-M.

`df`: Le nom, entre apostrophes, de la dérivée de la fonction stockée dans un fichier-M.

`x0`: L'approximation initiale.

`maxiter`: Le nombre maximal d'itérations.

`arret`: Le critère d'arrêt.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

Arguments en sortie:

`x`: Le vecteur des itérations.

`err`: S'il y a convergence, le vecteur $|x_n - x^*|$, où x^* est l'approximation calculée de la racine r.

2.5 La méthode de la sécante: `secante.m`

Appel de la fonction: `[x,err]=secante(f,x0,x1,maxiter,arret,resul)`
Exemple d'appel: `[x,err]=secante('fonc',2.0,3.0,10,1.0e-6,'resul.dat')`

Arguments en entrée:

`f`: Le nom, entre apostrophes, de la fonction stockée dans un fichier-M.

`x0`: Une première approximation initiale.

`x1`: Une deuxième approximation initiale.

`maxiter`: Le nombre maximal d'itérations.

`arret`: Le critère d'arrêt.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

Arguments en sortie:

`x`: Le vecteur des itérations.

`err`: S'il y a convergence, le vecteur $|x_n - x^*|$, où x^* est l'approximation calculée de la racine r.

Chapitre 3: Systèmes d'équations algébriques

3.7 Le conditionnement d'une matrice: `cond1.m`, `cond2.m`, `condinf.m`

Appel des fonctions:
```
y = cond1(A)
y = cond2(A)
y = condinf(A)
```

Argument en entrée:

`A`: Une matrice carrée.

Argument en sortie:

`y`: Le conditionnement cond(A), selon les normes l_1, l_2 ou l_∞.

3.8 La méthode de Newton pour les systèmes non-linéaires: `sysnl.m`

Appel de la fonction: `x=sysnl(residu,x0,maxiter,h,arret,resul)`
Exemple d'appel: `x=sysnl('sys',[1 2],10,1.0e-3,1.0e-6,'resul.dat')`

Arguments en entrée:

`residu`: Le nom, entre apostrophes, du vecteur résidu $\vec{R}(\vec{x})$ stocké dans un fichier-M.

`x0`: Une approximation initiale sous la forme d'un vecteur ligne.

`maxiter`: Le nombre maximal d'itérations.

`h`: Le paramètre h utilisé dans le calcul des différences finies pour le calcul de l'approximation de la matrice jacobienne.

`arret`: Le critère d'arrêt.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

Argument en sortie:

`x`: Matrice contenant les itérations, où chaque ligne correspond à une itération.

Chapitre 5: Interpolation

5.4 Le polynôme d'interpolation de Newton: `polynew.m`

Appel de la fonction:
```
[c,a,D]=polynew(xi,fxi,[],resul)
[c,a,D,fx]=polynew(xi,fxi,x,resul)
```
Exemple d'appel: `[c,a,D,fx]=polynew([0 1 2 3],[1 2 9 28],[2.2 8.4],'r.dat')`

Arguments en entrée:

`xi`: Les abscisses x_i sous la forme d'un vecteur ligne.

`fxi`: La valeur de la fonction à interpoler $f(x_i)$, aux points d'interpolation x_i, sous la forme d'un vecteur ligne.

`x`: Les valeurs de x où vous voulez interpoler la fonction, sous la forme d'un vecteur ligne. Vous pouvez entrer un vecteur vide `[]` si vous ne voulez pas utiliser cette fonctionnalité.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

Arguments en sortie:

`c`: Vecteur ligne de la forme $\vec{c} = (c_3, c_2, c_1, c_0)$ par exemple, contenant les coefficients du polynôme d'interpolation, soit $p_3(x) = c_3x^3 + c_2x^2 + c_1x + c_0$ dans cet exemple.

`a`: Vecteur ligne de la forme $\vec{a} = (a_0, a_1, a_2, a_3)$ par exemple, contenant les coefficients du polynôme de Newton, soit $p_3(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2)$ dans cet exemple.

`D`: Matrice contenant la table des différences divisées.

`fx`: Vecteur ligne contenant l'interpolation de la fonction aux points donnés dans le vecteur `x`.

Graphique d'un polynôme d'interpolation:

```
% on a préalablement initialisé les vecteurs xi et fxi
x=0:0.001:1;
% on construit le polynôme d'interpolation
[c,a,D,fx]=polynew(xi,fxi,x,'resul.dat');
% on évalue un polynôme, dont les coeffs sont dans c, en x
f=polyval(c,x);
% on fait le graphe du polynôme d'interpolation
plot(x,f);
```

5.6 L'interpolation par splines cubiques: `splinc.m`

Appel de la fonction: `[S,fpp]=splinc(xi,fxi,type,[],resul)`
`[S,fpp,Sx]=splinc(xi,fxi,type,x,resul)`

Exemple d'appel: `[S,fpp,Sx]=splinc([1 2 4],[1 4 9],'n',[2.2 3.1],'r.dat')`

Arguments en entrée:

`xi`: Les abscisses x_i sous la forme d'un vecteur ligne.

`fxi`: La valeur de la fonction à interpoler $f(x_i)$, aux points d'interpolation x_i, sous la forme d'un vecteur ligne.

`type`: Le type de spline cubique, sous la forme d'une chaîne de caractères:

- `'n'`: $f_0'' = f_n'' = 0$ (spline naturelle)
- `'dp'`: $f_a' = a$ et $f_n' = b$
- `'ds'`: $f_a'' = a$ et $f_n'' = b$
- `'cc'`: $f_0'' = f_1''$ et $f_{n-1}'' = f_n''$

`x`: Les valeurs de x où vous voulez interpoler la fonction, sous la forme d'un vecteur ligne. Vous pouvez entrer un vecteur vide `[]` si vous ne voulez pas utiliser cette fonctionnalité.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

Arguments en sortie:

`S`: Matrice de dimension (*nombre d'intervalles*) $\times$ 4, des coefficients des splines cubiques. Chaque ligne de la matrice (`S(i,:)`) contient les coefficients (a_i, b_i, c_i, d_i) de la spline sur l'intervalle $[x_{i-1}, x_i]$, soit $p_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$

`fpp`: Vecteur ligne contenant les dérivées secondes (f_i'') de la spline cubique

`Sx`: Vecteur ligne contenant l'interpolation de la fonction aux points donnés dans le vecteur `x`

Graphique d'une spline cubique:

```
% on a préalablement initialisé les vecteurs xi et fxi
x=0:0.001:1;
% on construit la spline cubique
[S,fpp,Sx]=splinc(xi,fxi,'n',x,'resul.dat');
% on fait le graphe de la spline
plot(x,Sx);
```

Chapitre 6: Différentiation et intégration numérique

6.4 Les formules de Newton-Cotes composées: `trapeze.m`, `simp13`, `simp38.m`, `boole.m`

Appel des fonctions:
```
I=trapeze(f,a,b,N)
I=simp13(f,a,b,N)
I=simp38(f,a,b,N)
I=boole(f,a,b,N)
```
Exemple d'appel:
```
I=trapeze('fonc',0,2,5)
```

Arguments en entrée:

`f`: Soit le nom, entre apostrophes, de la fonction stockée dans un fichier-M, soit les valeurs numériques de la fonction aux points d'intégration sous la forme d'un vecteur ligne, correspondant au nombre de sous-intervalles utilisés (si on a N sous-intervalles, on doit avoir $N+1$ valeurs de la fonction dans le vecteur.

`a,b`: Les bornes d'intégration.

`N`: Le nombre de sous-intervalles, qui doit être un multiple de 2 pour `simp13`, un multiple de 3 pour `simp38` et un multiple de 4 pour `boole`.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

Argument en sortie:

`I`: L'approximation de l'intégrale définie.

6.4 La méthode de Romberg: `romberg.m`

Appel de la fonction:
```
T=romberg(f,a,b,N,resul)
```
Exemple d'appel:
```
T=romberg('fonc',0,2,16,'resul.dat')
```

Arguments en entrée:

`f`: Soit le nom, entre apostrophes, de la fonction stockée dans un fichier-M, soit les valeurs numériques de la fonction aux points d'intégration sous la forme d'un vecteur ligne, correspondant au nombre de sous-intervalles utilisés (si on a N sous-intervalles, on doit avoir $N+1$ valeurs de la fonction dans le vecteur.

`a,b`: Les bornes d'intégration.

`N`: Le nombre de sous-intervalles, qui doit être une puissance de 2.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

Argument en sortie:

`T`: Matrice contenant les approximations par extrapolation de Richardson T_{ij} de l'intégrale définie.

6.4 La quadrature de Gauss: `intgauss.m`

Appel de la fonction:
```
[I,ti,wi] = intgauss(f,a,b,n,resul)
```
Exemple d'appel:
```
[I,ti,wi] = intgauss('fonc',0,2,5,'resul.dat')
```

Arguments en entrée:

`f`: Le nom, entre apostrophes, de la fonction stockée dans un fichier-M.

`a,b`: Les bornes d'intégration.

`n`: Le nombre de points de Gauss.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

Arguments en sortie:

`I`: L'approximation de l'intégrale définie.

`ti`: Les points de Gauss, dans l'intervalle $[-1, 1]$, sous la forme d'un vecteur ligne.

`wi`: Les poids d'intégration, sous la forme d'un vecteur ligne.

6.4 Intégration à l'aide de splines cubiques: `intspl.m`

Appel de la fonction: `I=intspl(xi,fxi,fpp)`

Exemple d'appel:
```
[S,fpp,Sx]=splinc([1 2 4],[1 4 9],'n',[],'r.dat')
I=intspl([1 2 4],[1 4 9],fpp)
```

Arguments en entrée:

`xi`: Les abscisses x_i sous la forme d'un vecteur ligne.

`fxi`: La valeur de la fonction à interpoler $f(x_i)$, aux points d'interpolation x_i, sous la forme d'un vecteur ligne.

`fpp`: Vecteur ligne contenant les dérivées secondes (f_i'') de la spline cubique, calculées à l'aide de la fonction `splinc`.

Argument en sortie:

`I`: L'approximation de l'intégrale définie.

Chapitre 7: Équations différentielles

7.2–7.6 La résolution des équations différentielles: `euler.m`, `eulmod`, `ptmilieu.m`, `rk4.m`

Appel des fonctions:
```
[t,y]=euler(f,t0,y0,h,pastemps,resul,impres)
[t,y]=eulmod(f,t0,y0,h,pastemps,resul,impres)
[t,y]=ptmilieu(f,t0,y0,h,pastemps,resul,impres)
[t,y]=rk4(f,t0,y0,h,pastemps,resul,impres)
```

Exemple d'appel:
```
[t,y]=euler('eqdif',0.0,[1 1],1.0e-1,10,'resul.dat',5)
```

Arguments en entrée:

`f`: Le nom, entre apostrophes, de la fonction $\vec{f}(t, \vec{y})$ stockée dans un fichier-M.

`t0`: Le temps initial.

`y0`: Un vecteur ligne contenant la condition initiale $\vec{y}(t_0)$.

`h`: Le pas de temps.

`pastemps`: Le nombre de pas de temps.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

`impres`: Le pas des itérations qui seront écrites dans le fichier de résultats (impression à tout les 5 pas de temps, par exemple).

Arguments en sortie:

`t`: Vecteur ligne contenant les t_i.

`y`: Matrice de dimension (*nombre de pas de temps*) $\times$ (*nombre d'équations*) dont la i-ième colonne contient la solution de la i-ième équation.

7.3 La méthode de Taylor d'ordre 2: `taylor.m`

Appel de la fonction: `[t,y]=taylor(f,dft,dfy,t0,y0,h,pastemps,resul,impres)`
Exemple d'appel: `[t,y]=taylor('ftay','dtftay','dyftay',0.0,1,1.0e-1,10,'resul.dat',5)`

Arguments en entrée:

`f`: Le nom, entre apostrophes, de la fonction $\vec{f}(t,\vec{y})$ stockée dans un fichier-M.

`dft`: Le nom, entre apostrophes, de la dérivée $\frac{\partial \vec{f}(t,\vec{y})}{\partial t}$ stockée dans un fichier-M.

`dfy`: Le nom, entre apostrophes, de la dérivée $\frac{\partial \vec{f}(t,\vec{y})}{\partial y}$ stockée dans un fichier-M.

`t0`: Le temps initial.

`y0`: Un vecteur ligne contenant la condition initiale $\vec{y}(t_0)$.

`h`: Le pas de temps.

`pastemps`: Le nombre de pas de temps.

`resul`: Le nom, entre apostrophes, du fichier de résultats.

`impres`: Le pas des itérations qui seront écrites dans le fichier de résultats (impression à tout les 5 pas de temps, par exemple).

Arguments en sortie:

`t`: Vecteur ligne contenant les t_i.

`y`: Vecteur ligne contenant les solutions y_i.

7.8 La méthode de tir: `tir.m`

Appel de la fonction: `[t,Y]=tir(a2,a1,a0,ab,yayb,pastemps,methode,resul,impres)`
Exemple d'appel: `[t,Y]=tir('a2','a1','a0',[1 2],[0 0.693147],10,'rk4','resul.dat',5)`

Arguments en entrée:

`a2`: Le nom, entre apostrophes, de la fonction $a_2(x)$ stockée dans un fichier-M.

`a1`: Le nom, entre apostrophes, de la fonction $a_1(x)$ stockée dans un fichier-M.

`a0`: Le nom, entre apostrophes, de la fonction $a_0(x)$ stockée dans un fichier-M.

`ab`: Un vecteur ligne contenant l'intervalle de définition du problème aux limites.

`y0`: Un vecteur ligne contenant les conditions aux limites.

`pastemps`: Le nombre de pas de temps.

`methode`: La méthode numérique à utiliser, entre apostrophes, pour résoudre les systèmes d'équations différentielles intermédiaires:

- `'euler'`
- `'eulmod'`
- `'ptmilieu'`
- `'rk4'`

`resul`: Le nom, entre apostrophes, du fichier de résultats.

`impres`: Le pas des itérations qui seront écrites dans le fichier de résultats (impression à tout les 5 pas de temps, par exemple).

Arguments en sortie:

`t`: Vecteur ligne contenant les t_i.

`y`: Matrice dont les colonnes sont formées de $\vec{y}$, $\vec{y}_1$, $\vec{y}_1'$, $\vec{y}_2$, $\vec{y}_2'$.

Graphiques de solutions d'équations différentielles: `traceqd.m`

Appel de la fonction: `traceqd(t,y,methode,nosol,impres,sol)`
Exemple d'appel: `traceqd(t,y,'euler',[1 2],5,'solex')`

Arguments en entrée:

`t`: Vecteur ligne contenant les pas de temps.

`y`: Matrice solution obtenue des méthodes de résolution des équations différentielles.

`methode`: La méthode numérique utilisée, entre apostrophes, pour résoudre les équations différentielles:

- `'euler'`
- `'eulmod'`
- `'ptmilieu'`
- `'rk4'`
- `'taylor'`
- `'tir'`

`nosol`: Un vecteur ligne dont les entrées indiquent les indices des colonnes de la matrice solution `y` à afficher.

`impres`: Le pas des itérations qui seront écrites dans le fichier de résultats (impression à tout les 5 pas de temps, par exemple).

`sol`: Le nom, entre apostrophes, de la solution analytique de l'équation différentielle stockée dans un fichier-M. Cet argument est facultatif.

Steven Dufour
11 novembre 2001