TRAVAUX PRATIQUES DE l'UE MNBmater

Matériaux 3A

MÉTHODES NUMÉRIQUES DE BASE

2024-2025, Automne

Jérôme Bastien

Document compilé le 6 octobre 2024

Le lien original de ce document est le suivant :
http://utbmjb.chez-alice.fr/Polytech/MNBmater/MNBmater.pdf

Ce document est mis à disposition selon les termes de la licence Creative Commons : Paternité - Pas d'Utilisation Commerciale - Pas de Modification ; 3.0

http://creativecommons.org/licenses/by-nc-nd/3.0/
ou en français
http://creativecommons.org/licenses/by-nc-nd/3.0/deed.fr

Liste des Travaux Pratiques

Avant-propos ii

Travaux Pratiques 1. Connaissances de base de matlab 1

Travaux Pratiques 2. Interpolation 2
2.1. Interpolation polynômiale 2
2.2. Interpolation au sens des moindres carrés 6

Travaux Pratiques 3. Intégration 7
3.1. Écriture des différentes méthodes d'intégration 7
3.2. Quelques exercices 7
3.3. Corrigé 7

Travaux Pratiques 4. Équations non-linéaires 8

Travaux Pratiques 5. Équations différentielles 9

Annexe A. Matlab/Octave à distance 10
A.1. Matlab à distance 10
A.2. Octave sur votre machine 10

Bibliographie 12

Avant-propos

Ce polycopié constitue les TP de Méthodes Numériques de Base du département Matériaux 3A (2024-2025, Automne).
Ce polycopié de TP et les fichiers matlab utilisés sont normalement disponibles à la fois

- en ligne sur `http://utbmjb.chez-alice.fr/Polytech/index.html` à la rubrique habituelle ;
- en cas de problème internet, sur le réseau de l'université Lyon I : il faut aller sur :
  — 'Poste de travail',
  — puis sur le répertoire '`P:`' (appelé aussi '\\`teraetu`\`Enseignants`'),
  — puis '`jerome.bastien`',
  — puis '`Polytech`',
  — puis '`Matériaux 3A`'.
  — enfin sur '`MNBmater`'.

Pour l'utilisation de Matlab/Octave à distance, on pourra consulter l'annexe A page 10.

TRAVAUX PRATIQUES 1

Connaissances de base de matlab

On poourra consulter les [Bas22b, Travaux Pratiques 1], extraites [1] de [BM03].

1. Elles en constituent l'annexe B.

TRAVAUX PRATIQUES 2

Interpolation

2.1. Interpolation polynômiale

2.1.1. Détermination du polynôme d'interpolation Π_n

Ce TP correspond au [BM03, TP 2.A p. 61]. La plupart des fonctions sont issues (parfois adaptées) de `https://www.dunod.com/sciences-techniques/introduction-analyse-numerique-applications-sous-matlab`

2.1.1.1. Détermination des différences divisées.

Dans un premier temps, il nous faut calculer les différences divisées $(f[x_i])_{0\leq i\leq n}$ définies données par l'équation (2.38) du cours rappelée ici

$$\forall i \in \{0, ..., n\}, \quad f[x_i] = f(x_i),$$

puis, pour tout $k \in \{1, ..., n\}$, les quantités $(f[x_i, ..., x_{i+k}])_{0\leq i\leq n-k}$ définies par l'équation (2.41) du cours rappelée ici

$$\forall k \in \{1, ..., n\}, \quad \forall i \in \{0, ..., n-k\}, \quad f[x_i, ..., x_{i+k}] = \frac{f[x_{i+1}, ..., x_{i+k}] - f[x_i, ..., x_{i+k-1}]}{x_{i+k} - x_i}.$$

Algorithme 2.1 Détermination des différences divisées : *diff_div_dist*$(n, x, y \longrightarrow d)$

Entrée :

n : entier naturel tel que $n+1$ désigne le nombre de points d'interpolation ;

x : vecteur des $n+1$ abscisses réelles des points d'interpolation : pour tout i de $\{0, ..., n\}$, $x(i) = x_i$;

y : vecteur des $n+1$ valeurs réelles prises par f en $(x_i)_{0\leq i\leq n}$: pour tout i de $\{0, ..., n\}$, $y_i = f(x_i)$.

Sortie :

d : vecteur des différences divisées. Pour tout i de $\{0, ..., n\}$, $d(i) = f[x_0, ..., x_i]$.

pour $i = 0$ à n **faire**
  $d(i) \longleftarrow y(i)$
fin pour
pour $i = 1$ à n **faire**
  pour $j = n$ à i **faire**
    $d(j) \leftarrow \dfrac{d(j) - d(j-1)}{x(j) - x(j-i)}$
  fin pour
fin pour

On remplit le tableau des différences divisées grâce à chacune de ces formules. On utilisera l'algorithme 2.1. On prendra garde au fait que dans cet algorithme, les vecteurs x, y et d sont indicés par des entiers décrivant l'ensemble $\{0, ..., n\}$ alors que, ci-dessous, les variables correspondates de matlab, X, Y et D sont indicés par des entiers décrivant l'ensemble $\{1, ..., n+1\}$.

Écrire une fonction matlab

[D]=**diff_div_dist**(X,Y)

- X et Y sont deux vecteurs de taille $n+1$ contenant respectivement les valeurs $(x_i)_{0\leq i\leq n}$ et $(f(x_i))_{0\leq i\leq n}$
- D est un vecteur de taille $n+1$, contenant les valeurs $\big(f[x_0], ..., f[x_0,...,x_n]\big)$

On pourra modifier la fonction écrite de la façon suivante afin qu'elle calcule l'ensemble des différences divisées :

```
function [D,Dt]=diff_div_dist(X,Y)

% Corps d'algorithme
nout=nargout;
if nout>=2
    if isnumeric(X)&&isnumeric(Y)
        Dt=zeros(n+1,n+1);
    else
        Dt=sym(zeros(n+1,n+1));
    end
end
D=Y;
if nout>=2
    Dt(:,1)=Y.';
end
for i=1:n;
    j=i:n;
    D(j+1)=(D(j+1)-D(j))./(X(j+1)-X(j-i+1));
    if nout>=2
        Dt(1:n+1-i,i+1)=(D(i+1:n+1)).';
    end
end
```

Notons que la seconde boucle a été "vectorizée". La double boucle peut en effet s'écrire (mais c'est plus long), conformément à l'algorithme 2.1 :

```
for i=1:n;
    for j=n:-1:i;
        D(j+1)=(D(j+1)-D(j))/(X(j+1)-X(j-i+1));
    end
end
```

On notera aussi le retournement du tableau d'écrit par la variable `j`.

Télécharger la fonction `diff_div_dist`, disponible à l'adresse habituelle.

2.1.1.2. Exemples de calcul de différences divisées.

Reprendre quelques exemples traités en TD. On pourra utiliser le format `rat`, le calcul symbolique, voire utiliser la troncature pour revoir certains calculs faits en TD.

Exemple 2.1. Par exemple, traiter le calcul des différences divisées de l'exercice de TD 2.2, en tapant

```
[D,Dt]=diff_div_dist([1 2 6],[-3 1 2]);
disp(Dt);
format rat;
```

```
disp(Dt);
format;
[D,Dt]=diff_div_dist(sym([1 2 6]),sym([-3 1 2]));
disp(Dt);
```

Exemple 2.2. Par exemple, traiter le calcul des différences divisées de l'exercice de TD 2.3, en tapant

```
A=2;
B=3;
a=6;
b=7;
n=3;
No=-cos(pi*(0:n)/n);
Nob=(a+b)/2+(b-a)/2*No;
f=@(x) log(A*x+B);
[D,Dt]=diff_div_dist(Nob,f(Nob));
disp(Dt);

format long;
disp(Dt);
format;
for p=1:7
    [D,Dt]=diff_div_dist(Nob,round(f(Nob)*10^p)/10^p);disp(round(Dt*10^p)/10^p);
end

No=-cos(sym(pi)*(0:n)/n);
Nob=(a+b)/2+(b-a)/2*No;
[D,Dt]=diff_div_dist(Nob,f(Nob));
disp(Dt);
disp(double(Dt));
```

2.1.1.3. Évalution grâce à la méthode de Hörner.

Rappelons l'équation (2.37a) du cours : nous allons chercher Π_n sous la forme

$$\Pi_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + ... + a_n(x - x_0)(x - x_1)...(x - x_{n-1}),$$

où les $(a_i)_{0\leq i\leq n}$ sont des réels connus.

On se convaincra que cet algorithme fournit bien les valeurs de Π_n en consultant les équations (2.33) de la remarque à la fin du corrigé de l'exercice 2.4 de TD, rappelées ci-dessous

$p_1(x) = a_0 + (x - x_0)a_1,$

si $n = 1$,

$p_2(x) = a_0 + (x - x_0)(a_1 + (x - x_1)a_2),$

si $n = 2$ et de façon plus générale :

$p_n(x) = a_0 + (x - x_0)(a_1 + (x - x_1)(a_2 + (x - x_2)(a_3 + (x - x_3)(...(a_{n-2} + (x - x_{n-2})(a_{n-1} + (x - x_{n-1})a_n)))...)))$

et en le faisant tourner, de façon "débranchée" pour de petites valeurs de n.

Algorithme 2.2 Algorithme d'évaluation de p en t $eval_horner(n, a, c, t \longrightarrow val)$

Entrée :
n : entier naturel représentant le degré de p ;
a : vecteur de $n+1$ réels désignant les coefficients de p dans la base de Newton considérée, donc $a(i) = a_i$ pour tout i de $\{0, ..., n\}$;
c : vecteur de n réels désignant les centres considérés, donc $c(i) = c_i$ pour tout i de $\{1, ..., n\}$;
t : réel en lequel on évalue p ;
Sortie :
val : réel défini par $val = p(t)$.

$val \longleftarrow a(n)$
pour $i = n-1$ à 0 **faire**
  $val(i) \longleftarrow a(i) + (t - c(i+1))\, val$
fin pour

Télécharger la fonction `eval_horner`, disponible à l'adresse habituelle, décrite ci-dessous :

[Z]=**eval_horner**(T,C,D)

- T est un tableau de réels (notés t), C est un vecteur de taille n contenant $(c_i)_{1 \leq i \leq n}$, D est un vecteur de taille $n+1$ contenant $(d_i)_{0 \leq i \leq n}$
- Z est égal au tableau des images de T par p_n, notés z vérifiant :

$$z = p_n(t) = \sum_{i=0}^{n} d_i \left(\prod_{j=1}^{i} (t - c_j) \right). \tag{2.1}$$

2.1.2. Mise en place finale de Π_n

Montrer, qu'à partir d'un tableau T et des valeurs $(x_i)_{0 \leq i \leq n}$ et $(f(x_i))_{0 \leq i \leq n}$ stockées dans les vecteurs X et Y, l'appel des fonctions **diff_div_dist**(X,Y) et **eval_horner**(T,C,D) permet de calculer l'image par p_n du tableau T. Pour quelle valeur de C ?

Télécharger la fonction `test_diff_div_dist`, disponible à l'adresse habituelle.

2.1.3. Exemples complets

Quelle quantité peut-on déterminer et calculer sous matlab pour être certain que les programmes sont bien écrits ?

Exemple 2.3. Reprenons l'exemple 2.1 (et les données l'exercice de TD 2.2). Taper, comprendre et commenter ce qui suit

```
X=[1 2 6];
Y=[-3 1 2];
disp(norm(test_diff_div_dist(X,X,Y)-Y));
disp(test_diff_div_dist(1.8,X,Y));
syms x;
disp(expand(test_diff_div_dist(x,X,Y)));
```

Exemple 2.4. Reprenons l'exemple 2.2 (et les données l'exercice de TD 2.3). Taper, comprendre et commenter ce qui suit :

```
A=2;
B=3;
a=6;
b=7;
n=3;
No=-cos(pi*(0:n)/n);
Nob=(a+b)/2+(b-a)/2*No;
f=@(x) log(A*x+B);
x=linspace(a,b,1000);
y=test_diff_div_dist(x,Nob,f(Nob));
plot(x,y,x,f(x),Nob,f(Nob),'o');
figure;
plot(x,abs(y-f(x)));

z=linspace(a,b,1e6);
disp(max(abs(test_diff_div_dist(z,Nob,f(Nob))-f(z))));
```

On pourra consulter la figure 2.5 page 14 ainsi que l'équation (2.26) des corrigés de TD.

2.1.4. Corrigé

Télécharger le script `corrige_TP1`, disponible à l'adresse habituelle et qui reprend tout ce qui été vu.

2.2. Interpolation au sens des moindres carrés

2.2.1. Détermination de coefficients de correction

En cours de rédaction.

Faire en direct en TP en utilisant les fichiers envoyé par mail ....

Voir aussi [Bas22a, Chapitre "Systèmes linéaires et matrices", section "Étude d'un exemple concret"] disponible sur `http://utbmjb.chez-alice.fr/Polytech/MFI/coursMFI.pdf`.

2.2.2. Ajustement affine

En cours de rédaction.

Voir l'exemple D.5 de l'annexe D du cours.

On se donne $p \in \mathbb{N}$ et p couples du plan $(x_i, y_i)_{1 \leq i \leq n}$ et l'on cherche à déterminer la droite aux moindres carrés, qui passe donc "le plus proche possible de ce nuage de points".

En utilisant la fonction `polyfit` de matlab (et éventuellemet en comparant avec les systèmes linéaires donnés !), reprendre par exemple les simulations 1 page 159 du cours, 2 page 160 du cours et éventuellement 3 page 161 du cours données dans l'exemple D.5 de l'annexe D du cours.

Voir la solution dans les script .... **Faire en direct en TP**

TRAVAUX PRATIQUES 3

Intégration

3.1. Écriture des différentes méthodes d'intégration

On fait appel au tableau 3.4 page 59 du cours.

Pour évaluer la somme

$$S = h \sum_{i=0}^{N-1} f(x_i),$$

on peut utiliser la boucle suivante

```
h=(B-A)/N;
S=0;
for i=0:N-1
    S=S+feval(fcn,A+h*i);
end
```

Écrire cette formule et testez-la.

Testez la version suivante :

```
h=(B-A)/N;
res=h*sum(feval(fcn,A:h:B-h,varargin{:}));
```

Télécharger la fonction `int_fcn` et testez-la.

3.2. Quelques exercices

Testez cette fonction sur l'exemple de la section 3.1 du cours.

Testez cette fonction sur l'exemple 3.28 page 66 du cours.

Tester cette fonction en symbolique en tapant par exemple

```
int_fcn(1,3,sym(0),sym(1),'cos')
```

3.3. Corrigé

Télécharger le script `corrige_TP3`, disponible à l'adresse habituelle et qui reprend tout ce qui été vu (qui appelle `logerreur`).

TRAVAUX PRATIQUES 4

Équations non-linéaires

En cours de rédaction

TRAVAUX PRATIQUES 5

Équations différentielles

En cours de rédaction

Annexe A

Matlab/Octave à distance

Vous avez deux les possibilités suivantes pour utiliser Matlab (section A.1) et son clone, libre et gratuit, Octave (section A.2).

A.1. Matlab à distance

Utilisez une machine virtuelle en consultant :
`https://etu.univ-lyon1.fr/outils/acces-distant-aux-fichiers-et-aux-applications-pedagogiques`
Il faut donc faire (pour windows, pour les autres systèmes d'exploitation, voir l'url donnée ci-dessous)

— Ouvrez le menu Démarrer -> Tous les programmes -> Accessoires -> Connexion bureau à distance (ou parfois Accessoires -> Communication -> ...) ;
— La boîte de dialogue "Connexion bureau à distance" apparaît ;
— Tapez `tseetu.univ-lyon1.fr` dans le champ "Ordinateur", puis cliquez sur le bouton "Connexion".

Attention, cette solution a des inconvénients :

- Le réseau de la fac est trop aléatoire ! On peut avoir un bon débit puis dans l'heure, il devient catastrophique. De plus, pour qu'un TP ait officiellement lieu avec cette solution, une réservation de salle virtuelle doit être faite. Donc, sauf dans le cas où cette réservation est faite et annoncée, cette solution est dédiée aux utilisations individuelles.
- Vous aurez, accès *via* une machine viruelle à votre disque réseau (commençant par `U:`) et il faudra gérer vos fichiers et répertoires sur ce disque et pointer sur ce disque depuis Matlab.

D'autres logiciels utilisés à Lyon I sont disponible sur cette machine virtuelle (comme Maple).

A.2. Octave sur votre machine

(1) Installer Octave. Voir `https://www.gnu.org/software/octave/download`

(2) Installer le symbolique d'Octave

(a) Voir par exemple
`https://sites.google.com/site/lm3tpoptimisation/guide-octaveinstall-config`, qui présente une installation sans Python (d'autres installations utilisant des bibliothèque de Python sont possibles).

(b) Regarder l'exemple pour le "Symbolic package" et suivre pas-pas l'installation.

(c) N'oubliez pas, à chaque utilisation de la partie sybmbolique d'Octave, de taper

```
pkg load symbolic
```

Attention, la première ou les première fois il affiche `Symbolic pkg v2.7.1:` et puis, il faut attendre un peu ...

(d) Faites le test final suivant : tapez (et interprétez !)

```
syms x
int((cos(x))^2)
```

Quelques liens (certains sont contextuels et peuvent changer selon la version d'Octave).
`https://octave.org/doc/v5.2.0/`
`https://octave.org/octave.pdf`
`https://octave.sourceforge.io/list_functions.php?sort=alphabetic`

Bibliographie

[Bas22a] J. Bastien. *Mathématiques Fondamentales pour l'Informatique.* Notes de cours de l'UV MFI (Département Informatique) de Polytech Lyon, disponible sur le web : `http://utbmjb.chez-alice.fr/Polytech/index.html`, rubrique "Informatique 3A : Mathématiques Fondamentales pour l'Informatique". 2022. 270 pages.

[Bas22b] J. Bastien. *Mathématiques Fondamentales pour l'Informatique (apprentis).* Travaux Pratiques de l'UV MFIappro (Département Informatique) de Polytech Lyon, disponible sur le web : `http://utbmjb.chez-alice.fr/Polytech/index.html`, rubrique "Informatique 3A : Mathématiques Fondamentales pour l'Informatique (apprentissage)". 2022. 25 pages.

[BM03] J. Bastien et J.-N. Martin. *Introduction à l'analyse numérique. Applications sous Matlab.* Ouvrage disponible à la bibliothèque Sciences de Lyon 1 (cote : 519.4 BAS, 4e étage). Voir `https://www.dunod.com/sciences-techniques/introduction-analyse-numerique-applications-sous-matlab`. Paris : Dunod, 2003. 392 pages.