TP5-6: Tir simple et *hampath*

GERGAUD Joseph

1 Introduction

L'objectif de cette partie est de résoudre les problèmes de contrôle optimal avec le logiciel *hampath*. Ce logiciel, développé au sein de l'équipe APO de l'ENSEEIHT-IRIT par J.B. Caillau [1], O. Cots and J. Gergaud [2], est un code qui implémente en autre la méthode du tir simple en *Fortran*90 pour les problèmes de contrôle optimal lisses, c'est-à-dire que toutes les fonctions qui interviennent dans le problème sont C^∞ et que la minimisation de l'hamiltonien donne une fonction $u(x,p)$ elle aussi C^∞. Ce logiciel a aussi une interface avec *Matlab*. L'utilisateur n'a à écrire en *Fortran*90 que ce qui code l'hamiltonien vraie (voir l'équation (2) pour la définition du hamiltonien vraie) et ce qui code les conditions au deux bouts. La différentiation automatique (*tapenade*) permet ensuite de générer automatiquement les équations d'états et les équations adjointes ! En appliquant une nouvelle fois la différentiation automatique, les équations variationnelles sont générées, ce qui permet de fournir au solveur `ssolve` non seulement la fonction de tir, mais aussi sa dérivée.

ATTENTION La fonction de tir dans *hampath* est, pour les problèmes à temps final fixé, une fonction de $\mathbf{R}^{2n}$ à valeurs dans $\mathbf{R}^{2n}$ (n étant la dimension de l'état) :

$$\begin{array}{rcl} S : \mathbf{R}^{2n} & \longrightarrow & \mathbf{R}^{2n} \\ y_0 & \longmapsto & S(y_0) = \begin{pmatrix} b_0(y_0) \\ b_f(y(t_f, y_0)). \end{pmatrix} \end{array}$$

2 Présentation de `hampath`

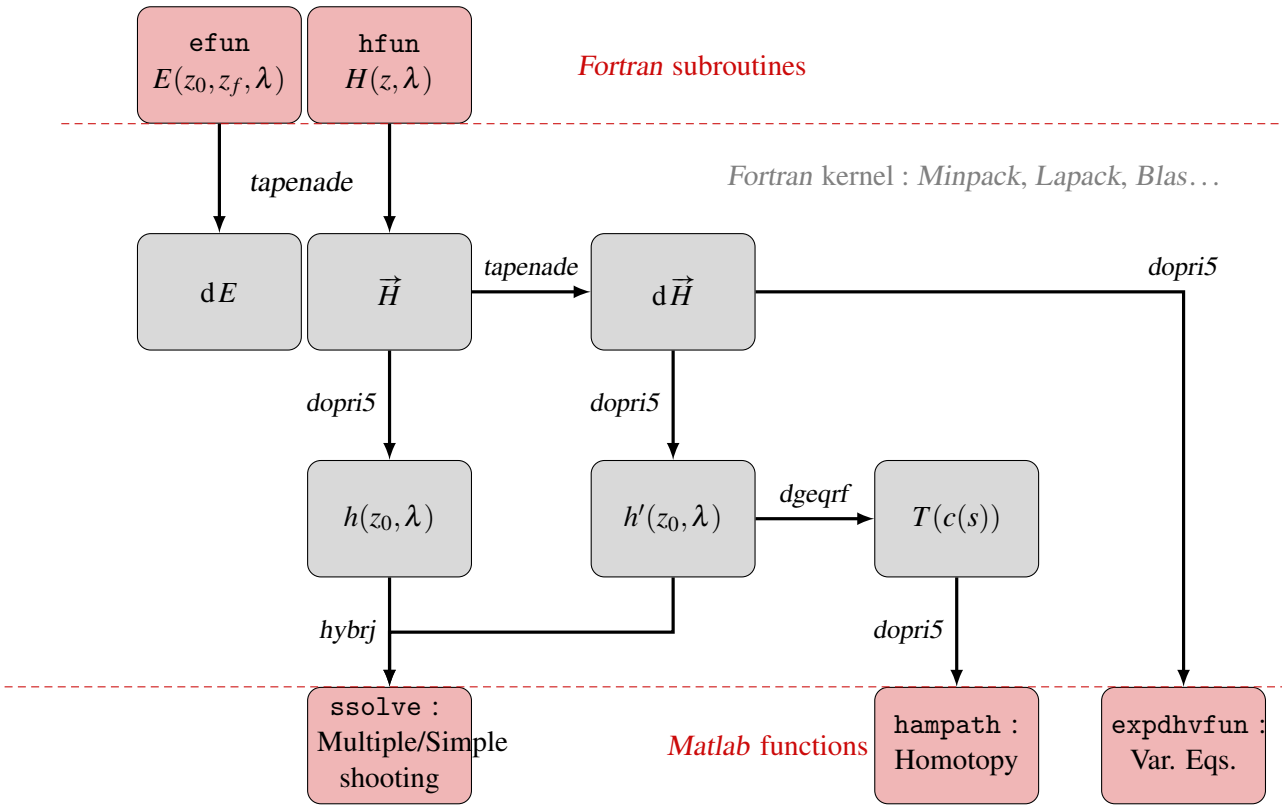
2.1 Introduction

Le logiciel *hampath* permet :
- de résoudre les problèmes de contrôle optimal lisse via les méthodes indirectes et la méthode du tir simple ;
- de suivre le chemin de zéros en fonction d'un paramètre ;
- de calculer les conditions du deuxième ordre via les points conjugués (hors programme).

Remarque 2.1. *Si on connaît la structure du contrôle optimal alors on peut aussi utiliser ce logiciel en utilisant le tir multiple.*

1. Math. Institute, Bourgogne Univ. & CNRS, {`jean-baptiste.caillau, olivier.cots`}`@u-bourgogne.fr`
2. INPT-ENSEEIHT-IRIT, joseph.gergaud@enseeiht.fr

2.2 Schéma



2.3 Exemple d'utilisation

Le problème résolu ici est le suivant :

$$(P_{demo}) \begin{cases} Min \int_0^1 u^2(t)dt \\ \dot{x}(t) = v(t) \\ \dot{v}(t) = -\lambda v^2(t) + u(t) \\ x(0) = -1; v(0) = -1 \\ x(1) = 0; v(1) = 0, \end{cases}$$

pour λ allant de 0 à 1.

La **maximisation de l'hamiltonien** donne

$$u(x, v, p_x, p_v) = p_v/2, \tag{1}$$

et l'hamiltonien vrai est

$$H_r(t, x, v, p_x, p_v) = H(t, x, p, u(x, p)) = \frac{p_v^2}{4} + v p_x - \lambda v^2 p_v. \tag{2}$$

1. Créer un répertoire de travail pour votre problème et aller dans ce répertoire
2. taper `hampath -nss` (new simple shooting)
   Le répertoire contient alors les fichiers :

– `hfun.f90` fonction en *Fortran*90 qui code l'hamiltonien vrai, c'est-à-dire la fonction $H_r(t,x,p) = H(t,x,u(t,x,p),p)$ où $u(t,x,p)$ est le contrôle solution de la maximisation (ou de la minimisation suivant la formulation choisie) de l'hamiltonien. Ce sous-programme *Fortran*90 code aussi la fonction $u(t,x,p)$;

```
Subroutine hfun(t,n,z,nbarc,iarc,lpar,par,H)
     ...
     !Local declarations
     double precision :: x,v,px,pv,lambda,u

     IF (nbarc.NE.1) THEN
         call printandstop('Error: wrong number of arcs.')
     END IF
     IF (n.NE.2) THEN
         call printandstop('Error: wrong state dimension.')
     END IF
     IF (lpar.NE.5) THEN
         call printandstop('Error: wrong par dimension.')
     END IF
     ...
     call control(t,n,z,nbarc,iarc,lpar,par,u)

     H       = -u**2 + px * v + pv * (-lambda * v**2 + u)

 end subroutine hfun

 Subroutine control(t,n,z,nbarc,iarc,lpar,par,u)
     ....
     pv      = z(n+2)
     u       = pv/2

 end subroutine control
```

– `efun.f90` fonction en *Fortran*90 qui code les conditions aux deux bouts ;

```
Subroutine efun(nbarc,n,ti,zi,expzi,lpar,par,ne,E)
     implicit none
     ....
     x0    =    zi(1,1      )
     v0    =    zi(2,1      )
     xf    = expzi(1,nbarc  )
     vf    = expzi(2,nbarc  )

     E(1)   = x0 - par(1)
     E(2)   = v0 - par(2)
     E(3)   = xf - par(3)
     E(4)   = vf - par(4)
 end subroutine efun
```

  – `zi(:,1)` $= z_0$
  – `expzi(:,1 )` $= z(t_1,t_0,z_0)$
  – `expzi(:,i+1 )` $= z(t_{i+1},t_i,z_i)$

– `main.m` script *Matlab* qui

  - initialise les paramètres du problème ;
  - appelle la fonction `ssolve` qui résout l'équation de tir ;
  - appelle la fonction `hampath` qui réalise le suivi de chemin ;
  - trace le chemin de zéros ;
  - appelle la fonction `exphvfun` qui intègre le problème à condition initiale ;
  - trace la solution pour $\lambda = 1$;
  - test les conditions du deuxième ordre
- taper `hampath`
  Le code est alors compilé. Il crée entre autre les fonctions *Fortran*90 et les interfaces *Matlab* qui codent
  - la fonction qui code le deuxième membre du problème aux deux bouts (merci à la différentiation automatique *tapenade*) ;
  - la fonction qui code la fonction de tir ;
  - la fonction qui code le calcul de la dérivée de la fonction de tir via les équations variationnelles (merci à la différentiation automatique *tapenade*) ;
  - ...
- sous *Matlab*
  - taper `help hfun`

  - Exécuter le script `main`.

Les fonctions disponibles sous *Matlab* sont données à la table 1

Functions	Descriptions
`hfun` (*)	calcul de l'hamiltonien
`hvfun` (*)	calcule le deuxième membre du problème aux deux bouts
`sfun` (*)	calcule la fonction de tir
`sjac`	calcule la dérivée de la fonction de tir
`ssolve` (*)	calcule un zéro de la fonction de tir
`hampathget` (*)	récupère les paramètres numériques (tolérances pour l'intégration numérique, ...)
`hampathset` (*)	initialise les paramètres numériques (tolérances pour l'intégration numérique, ...)
`hampath`	réalise le suivi de chemin de zéros
`exphvfun` (*)	réalise une intégration numérique
`expdhvfun`	réalise une intégration numérique des équations variationnelles
`control` (*)	calcule le contrôle

TABLE 1 – *Liste des fonctions Matlab ,(*) indique les fonctions utiles pour ce TP.*

3 Travail demandé

En utilisant le logiciel `hampath`

1. Résoudre le problème (TP1)

$$(P_1)\begin{cases} Min \int_0^2 u^2(t)dt \\ \dot{x_1}(t) = x_2(t) \\ \dot{x_2}(t) = u(t) \\ x_1(0) = 0; x_2(0) = 0 \\ x_1(2) = 0.5; x_2(2) = 0. \end{cases}$$

2. Résoudre pour $\varepsilon = 1, 0.1, 0.01$ et 0.001 le problème (TP2)

$$(P_\varepsilon) \begin{cases} Min \int_0^2 (|u(t)| - \varepsilon(\ln|u(t)| + \ln(1-|u(t)|)))dt \\ \dot{x}(t) = -x(t) + u(t) \\ |u(t)| < 1 \\ x(0) = x^0 = 0 \\ x(2) = x^f = 0.5 \end{cases}$$

3. Résoudre le problème de transfert d'orbite (TP 2-3)
4. Résoudre le même problème mais en faisant décroître γ_{max}. On donnera les résultats numériques sous la forme d'un tableau :

γ_{max}	$t_{f_{min}}$	$\|\|S(z)\|\|$	`ifail`
388.8	3.9111	1.988e+00	1
380	3.950416	1.816e-01	1
⋮	⋮	⋮	⋮

TABLE 2 – *Résultats numériques pour le problème de transfert d'orbite avec minimisation du temps final.*