Mètodes Numèrics per Equacions Diferencials Ordinàries

Part II: Problemes de valors a la fontera

Lluís Alsedà
Departament de Matemàtiques
Universitat Autònoma de Barcelona
http://www.mat.uab.cat/∼alseda

23 de desembre de 2025 (versió 2.0)

UAB Universitat Autònoma de Barcelona — Departament de Matemàtiques

Continguts

Introducció: Problemes de Valors a la Frontera

Un Problema de Valors a la Frontera té la forma[1]:

$$\begin{cases} \dot{x}(t) = f(t, x(t)), \text{ amb} \\ \psi(x(a), x(b)) = 0, \end{cases}$$

on f és una funció $f\colon \Omega \longrightarrow \mathbb{R}^n$, Ω és un domini d'$\mathbb{R} \times \mathbb{R}^n$, i ψ és una funció $\psi\colon \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$.

La condició $\psi(x(a), x(b)) = 0$ s'anomena *condició de frontera* o de *contorn*.

[1] Recordem que tota EDO o sistema d'EDO's d'ordre més gran que 1 es pot reescriure com un sistema d'EDO's d'ordre 1 en dimensió superior.

Introducció: Problemes de Valors a la Frontera

El tipus més habitual (i més senzill) de condicions de frontera són les *lineals*. Una condició de frontera s'anomena *lineal* si és de la forma

$$B_a x(a) + B_b x(b) = c$$

on $a, b \in \mathbb{R}$ amb $a \neq b$, B_a i B_b són matrius $n \times n$ i $c \in \mathbb{R}^n$. Es diu que les condicions de frontera estan *separades* si es poden escriure com

$$B_a x(a) = c_a, \qquad B_b x(b) = c_b$$

on, com abans, $a, b \in \mathbb{R}$ amb $a \neq b$, B_a i B_b són matrius $n \times n$ i $c_a, c_b \in \mathbb{R}^n$. Observem que si unes condicions de frontera estan separades, aleshores són lineals.

Introducció: Cap al Problema de Tir Simple

Un Problema de Valors a la Frontera es pot reformular en un Problema de Valor Inicial

Considerem el problema de Cauchy[2]

$$\begin{cases} \dot{x}(t) = f(t, x(t)), \text{ amb} \\ x(a) = s_a \in \mathbb{R}^n, \end{cases}$$

i denotem $x(t; a; s_a)$ la seva solució (és a dir, $x(a; a; s_a) = s_a$).

Llavors, el Problema de Valors a la Frontera es pot reescriure com el Problema de Valor Inicial anterior afegint-hi una condició addicional de contorn:

$$\psi(s_a, x(b; a; s_a)) = \psi(x(a; a; s_a), x(b; a; s_a)) = 0.$$

Observem que la condició de contorn anterior és, de fet, un sistema d'equacions en dimensió n i incògnita s_a.

[2]Recordem que si l'EDO té paràmetres, aquests es poden convertir en variables augmentant la dimensió del sistema i afegint condicions inicials ad-hoc. En aquest context, estem assumint que tenim un sistema d'EDO's "parameters free" perquè estem suposant que ja s'ha fet aquest procés en cas que hagi estat necessari.

Finalment, el Problema de Tir Simple

Un Problema de Valors a la Frontera es pot reformular com un problema de zeros de funcions

En relació a la resolució del sistema

$$\psi(s, x(b; a; s)) = \psi(x(a; a; s), x(b; a; s)) = 0,$$

és útil definir la funció

$$\begin{aligned} \mathrm{T} : \mathbb{R}^n &\longrightarrow \mathbb{R}^n \\ s &\longmapsto \psi(s, x(b; a; s)), \end{aligned}$$

i reformular-lo com un problema de zeros de funcions:

$$\mathrm{T}(s) = \psi(s, x(b; a; s)) = 0.$$

Definició

S'anomena *Problema de Tir Simple* a trobar un zero del sistema (en general no-lineal) d'equacions

$$\mathrm{T}(s) = \psi(s, x(b; a; s)) = 0,$$

o a determinar que aquest zero no existeix.

Introducció al Tir Simple (inspirat en la Wikipedia)

El mètode de tir és un procediment per resoldre un problema de valors a la frontera reduint-lo a trobar zeros d'un problema de valor inicial.

Consisteix a resoldre el problema de valor inicial per a diferents condicions inicials fins que es trobi la solució que compleix les condicions de contorn.

El terme "mètode de tir" té el seu origen en l'artilleria amb la següent analogia:

- es col·loca un canó a la posició $x(t_0) = x_0$, llavors
- es varia la inclinació $x'(t_0)$ del canó, i aleshores
- es dispara el canó fins que encerti la diana (és a dir el valor de contorn $x(t_1) = x_1$).

A cada tret s'ajusta la inclinació del canó en funció del tret anterior, de manera que caigui més a prop de l'objectiu.

La trajectòria que encerta el valor de contorn $x(t_1) = x_1$ és la solució del problema de valors a la frontera.

Un comentari important mitjançant un exemple

Considerem el problema de valors a la frontera

$$\ddot{w} + w = 0 \qquad \text{amb} \qquad w(0) = 0 \text{ i } w(\pi) = 1.$$

La solució general és $w(t) = c_1 \sin(t) + c_2 \cos(t)$ però *clarament no hi ha cap solució que satisfaci les condicions de frontera*.

En conclusió: Els PVF's no necessàriament tenen solució

Els problemes de valors a la frontera són més generals que els problemes de valor inicial i, a diferència d'ells, no tenen perquè tenir solució.

Tanmateix, si canviem la condició $w(\pi) = 1$ per, per exemple, $w\left(\frac{\pi}{2}\right) = 1$, llavors tenim *existència i unicitat de solució pel problema de valors a la frontera*.

Per altra banda, si canviem la condició $w(\pi) = 1$ per $w(\pi) = 0$, llavors tenim *infinites solucions de l'EDO* (totes les de la forma $c \sin(t)$) *que verifiquen les condicions de frontera*.

Resolent el Problema del Tir Simple

Com hem dit, el problema de tir consisteix a trobar un zero de

$$\mathsf{T}(s) = \psi(s, x(b; a; s)) = 0$$

que, en general, és un sistema d'equacions no-lineal $n-$dimensional.

La millor manera de resoldre aquest sistema és usant el mètode de Newton o *mètodes Newton-like*, amb estratègies convenients per a tractar el problema de la llavor.

A la pàgina següent es recorda el mètode de Newton en varies variables, en la notació de l'Equació de Tir.

Més avall (pàgines 33 i 18) veurem (i discutirem) l'ús del mètode de Newton en dimensió 2 per Problemes de Tir.

Resolent el Problema del Tir Simple

El mètode de Newton per problemes de tir

0. Es tria una llavor $\mathbf{s}^{(0)} \in \mathbb{R}^n$,

i per $k = 0, 1, 2, \ldots$ fins que es compleixi alguna condició de parada s'itera:

- k-a Es resol el PVI
$$\dot{x}(t) = f(t, x(t)) \quad \text{amb} \quad x(a) = \mathbf{s}^{(k)},$$
per a obtenir $x\left(b; a; \mathbf{s}^{(k)}\right)$ i, llavors, es calcula $\mathsf{T}\left(\mathbf{s}^{(k)}\right) = \psi\left(\mathbf{s}^{(k)}, x\left(b; a; \mathbf{s}^{(k)}\right)\right)$.
- k-b Es calcula (una aproximació de) $J\,\mathsf{T}\left(\mathbf{s}^{(k)}\right)$, la matriu Jacobiana de T al punt $\mathbf{s}^{(k)}$.
- k-c Per a calcular $\mathbf{s}^{(k+1)} := \mathbf{s}^{(k)} - \left(J\,\mathsf{T}\left(\mathbf{s}^{(k)}\right)\right)^{-1} \mathsf{T}\left(\mathbf{s}^{(k)}\right)$, es resol el sistema $J\,\mathsf{T}\left(\mathbf{s}^{(k)}\right) \cdot \mathbf{r}^{(k)} = \mathsf{T}\left(\mathbf{s}^{(k)}\right)$.
- k-d Es defineix $\mathbf{s}^{(k+1)} := \mathbf{s}^{(k)} - \mathbf{r}^{(k)}$

Resolent el Problema del Tir Simple

Tenim dos problemes

El primer és el ja famós problema de la llavor. En aquest cas és especialment greu perquè, de fet, no sabem ni si el problema té solució ni quantes en té. Dit d'una altra manera, el més normal és que l'algorisme de la pàgina anterior es perdi (sobretot quan no hi ha solució 😇).

El segon problema és el punt (k-b): el càlcul de la matriu Jacobiana de T al punt $\mathbf{s}^{(k)}$, $J\,\mathsf{T}\left(\mathbf{s}^{(k)}\right)$. Aquí hi ha dues estratègies de càlcul possibles:

- l'*analítica* recomanada quan hi ha expressions analítiques senzilles de les derivades parcials de la la funció ψ, i
- l'*aproximació numèrica mitjançant diferències* de la matriu $J\,\mathsf{T}\left(\mathbf{s}^{(k)}\right)$.

A continuació discutirem els cóms, pros i contres de les solucions d'aquests problemes, i en donarem algun exemple.

Resolent el Problema del Tir Simple

Algunes possibles estratègies de bypass del problema de la llavor

i. **Difícil:** Usar arguments analítics (si es pot) per determinar un entorn el més petit possible de la solució del PVF i agafar la llavor en aquest entorn.

ii. Usar un mètode de minimització com un *quasi-Newton de descens amb backtracking per a controlar el salt*, o *Levenberg-Marquardt* (**lent però segur**)[3]. Aquests mètodes són molt poc dependents de la llavor sempre que es compleixin dos requisits: El primer i fonamental és que el PVF tingui solució. El segon és que *la norma de la funció $T(s)$ es porti bé (és dir sigui quadratic-like)*.

iii. **Problemàtic:** Usar un mètode Newton modificat com, per exemple, Broyden. El problema d'aquesta estratègia és que arregla poc el problema de la llavor.

[3] No és gens usual fer servir mètodes de minimització per a calcular zeros de funcions. Una bona comprensió de la geometria d'aquests problemes permet aquest tipus de reformulacions.

(iv) **Lent però segur:** Usar un mètode de continuació de la llavor des d'un problema similar del qual coneguem la solució. En veurem i discutirem un exemple més avall (pàgina 46).

(v) **Bullet Proof (solament vàlid en dimensió 1):** Trobar un interval de canvi de signe de la Funció de Tir (si és que existeix) i usar un mètode que mantingui i exploti aquest fet (**més lent però segur**) com bisecció o millor Regula Falsi, que és més eficient. Veurem i discutirem un exemple d'ús del Mètode de Regula Falsi més avall (pàgina 22).

Tenim

$$\begin{aligned} DT(s) &= D_s\psi(s, x(b;a;s)) \\ &= D_1\psi(s, x(b;a;s)) + D_2\psi(s, x(b;a;s)) \cdot D_s x(b;a;s), \end{aligned}$$

on $D_1\psi$ denota la diferencial de ψ respecte de la seva primera variable i $D_2\psi$ denota la diferencial de ψ respecte de la seva segona variable.

Les matrius de $D_1\psi$ i $D_2\psi$ òbviament estan determinades per la funció ψ i es poden considerar dades del problema.

En conseqüència, per a calcular $DT(s)$ es requereix el càlcul previ de $D_s x(b;a;s)$. Si denotem

$$x(b;a;s) := \big(x_1(b;a;s), x_2(b;a;s), \ldots, x_n(b;a;s)\big)^\top$$

i $s := (s_1, s_2, \ldots, s_n)^\top$ tenim

$$D_s x(b;a;s) = \left(\frac{\partial x_i(b;a;s)}{\partial s_j}\right),$$

que és la matriu de les derivades parcials de la solució respecte les condicions inicials al punt $t = b$.

Recordem que si f està definida a un obert d'$\mathbb{R} \times \mathbb{R}^n$ i és contínua i derivable amb continuïtat respecte de la segona variable, llavors $z_j(t) := \dfrac{\partial x(t;a;s)}{\partial s_j}$ existeix per tot $j \in \{1, 2, \ldots, n\}$ i per tot t a l'interval maximal de definició de la solució $x(t;a;s)$. A més satisfà el següent PVI:

$$\dot{z}_j = D_2 f\big(t, x(t;a;s)\big) \cdot z_j \quad \text{amb} \quad z_j(a) = e_j,$$

on e_j denota el $j-$èssim vector de la base canònica d'$\mathbb{R}^n$.

En conseqüència,

$$\frac{\partial x(b;a;s)}{\partial s_j} = D_s x(b;a;s) \cdot e_j,$$

la columna $j-$èssima de la matriu $D_s x(b;a;s)$, és la solució $z_j(t)\big|_{t=b}$ del PVI anterior.

Si $f(t,x)$ és prou diferenciable, llavors $x(t;a;s)$ també és prou diferenciable, i

$$\frac{\partial}{\partial t} z_j(t) = \frac{\partial}{\partial t}\frac{\partial x(t;a;s)}{\partial s_j} = \frac{\partial}{\partial s_j}\frac{\partial x(t;a;s)}{\partial t} =$$

$$\frac{\partial}{\partial s_j} f\big(t, x(t;a;s)\big) = D_2 f\big(t, x(t;a;s)\big)\frac{\partial x(t;a;s)}{\partial s_j} =$$

$$D_2 f\big(t, x(t;a;s)\big) \cdot z_j(t).$$

A més, donat que $x(a;a;s) = s$,

$$z_j(a) = \frac{\partial x(a;a;s)}{\partial s_j} = \frac{\partial s}{\partial s_j} = e_j.$$

Resolent el Problema del Tir Simple

Resumint: Càlcul conjunt de $\mathsf{T}(s)$ i $D\mathsf{T}(s)$

Per a calcular $\mathsf{T}(s)$, cal $x(b;a;s)$, que és solució del PVI

$$\dot{x} = f(t,x) \qquad \text{amb} \qquad x(a) = s.$$

Per a calcular $D\mathsf{T}(s)$, prèviament calcularem $D_s x(b;a;s)$ per columnes. Per $j = 1, 2, \ldots, n$,

$$D_s x(b;a;s) \cdot e_j = \frac{\partial x(b;a;s)}{\partial s_j}$$

és solució del PVI

$$\dot{z}_j = D_2 f\big(t, x(t;a;s)\big) \cdot z_j \qquad \text{amb} \qquad z_j(a) = e_j.$$

Això dóna el *"PVI Total"*

$$\begin{cases} \dot{x} = f(t,x), \\ \dot{z}_1 = D_2 f(t,x) \cdot z_1 \\ \dot{z}_2 = D_2 f(t,x) \cdot z_2 \\ \quad \vdots \\ \dot{z}_n = D_2 f(t,x) \cdot z_n \end{cases} \quad \text{amb} \quad \begin{cases} x(a) = s, \\ z_1(a) = e_1, \\ z_2(a) = e_2, \\ \quad \vdots \\ z_n(a) = e_n, \end{cases}$$

en dimensió $n + n \cdot n = n(n+1)$.

Resolent el Problema del Tir Simple

Estratègia d'aproximació de la matriu $D\mathsf{T}(s)$ mitjançant diferències

Ara, en comptes de descompondre la matriu $D\mathsf{T}(s)$ i calcular-la per factors, usarem avaluacions addicionals de la funció T per a aproximar directament

$$D\mathsf{T}(s) = \left(\frac{\partial \mathsf{T}_i(s)}{\partial s_j}\right)$$

on, com abans, hem denotat $s := (s_1, s_2, \ldots, s_n)^\top$ i

$$\mathsf{T}(s) := \big(\mathsf{T}_1(s), \mathsf{T}_2(s), \ldots, \mathsf{T}_n(s)\big)^\top.$$

Per a aproximar les derivades parcials $\dfrac{\partial \mathsf{T}_i(s)}{\partial s_j}$ usarem diferències finites. Recordem que les avaluacions de la funció T son "cares" (cal resoldre el PVI des de $t = a$ fins a $t = b$). Per tant, per no augmentar massa la complexitat del càlcul, ens restringirem a diferències de primer ordre (és a dir, una única avaluació addicional de la funció T per derivada parcial).

Concretament usarem *diferències finites endavant de primer ordre*.

Resolent el Problema del Tir Simple

Estratègia d'aproximació de la matriu $D\mathsf{T}(s)$ mitjançant diferències

Les derivades parcials en termes de les *diferències finites endavant de primer ordre* s'escriuen com:

$$\frac{\partial \mathsf{T}_i(s)}{\partial s_j} = \frac{\mathsf{T}_i(s + h e_j) - \mathsf{T}_i(s)}{h} - \frac{\left.\frac{d^2 \mathsf{T}_i(s + r e_j)}{dr^2}\right|_{r=\xi}}{2} h,$$

on $0 < |\xi| < h$ i e_j denota el $j-$èssim vector de la base canònica d'$\mathbb{R}^n$.

En particular,

$$\left| \frac{\partial \mathsf{T}_i(s)}{\partial s_j} - \frac{\mathsf{T}_i(s + h e_j) - \mathsf{T}_i(s)}{h} \right| = \mathcal{O}(h).$$

Resolent el Problema del Tir Simple

Estratègia d'aproximació de la matriu $D\mathsf{T}(s)$ mitjançant diferències

De fet, com hem fet a l'estratègia analítica, en comptes de calcular $\dfrac{\partial \mathsf{T}_i(s)}{\partial s_j}$ d'un en un, per eficiència computacional estem interessats a aproximar la matriu $D\mathsf{T}(s)$ per columnes:

$$D\mathsf{T}(s) \cdot e_j = \frac{\partial \mathsf{T}(s)}{\partial s_j} = \left(\frac{\partial \mathsf{T}_1(s)}{\partial s_j}, \frac{\partial \mathsf{T}_2(s)}{\partial s_j}, \ldots, \frac{\partial \mathsf{T}_n(s)}{\partial s_j}\right)^\top \approx \frac{\mathsf{T}(s + h e_j) - \mathsf{T}(s)}{h}.$$

Llavors, en notació vectorial,

$$\left\| \frac{\partial \mathsf{T}(s)}{\partial s_j} - \frac{\mathsf{T}(s + h e_j) - \mathsf{T}(s)}{h} \right\|_\infty = \mathcal{O}(h).$$

Altrament dit, per funcions *suaus* per les quals $\left|\frac{d^2 \mathsf{T}_i(s + r e_j)}{dr^2}\right|$ no és molt gran en un domini, com a norma general h es pot prendre de l'ordre de la precisió desitjada a l'aproximació.

Resolent el Problema del Tir Simple

Pros i contres de les dues estratègies d'aproximació de la matriu $DT(s)$

Recordem que, cada avaluació de la funció T requereix integrar un PVI en dimensió n i una avaluació de la funció ψ.

Estratègia analítica: S'integren $n+1$ PVI's en dimensió n, que donen $x(b;a;s)$ i $D_s x(b;a;s)$. Amb $x(b;a;s)$ i una avaluació de la funció ψ es calcula la funció de tir $T(s)$, i amb $D_1\psi(s,x(b;a;s))$ i $D_2\psi(s,x(b;a;s))$ operat amb $D_s x(b;a;s)$ s'obté una aproximació de $DT(s)$.

Estratègia d'aproximació de la matriu $DT(s)$ mitjançant diferències: Calen $n+1$ avaluacions de la funció T. Per tant, en total, cal integrar $n+1$ PVI's en dimensió n i $n+1$ avaluacions de la funció ψ.

L'eficiència computacional relativa de les dues estratègies

serà funció de la complexitat comparativa d'n avaluacions de ψ respecte d'una avaluació de $D_1\psi(s,x(b;a;s))$ i $D_2\psi(s,x(b;a;s))$ i el càlcul de $D_1\psi(s,x(b;a;s)) + D_2\psi(s,x(b;a;s)) \cdot D_s x(b;a;s)$, en dimensió n.

Resolent el Problema del Tir Simple

Pros i contres de les dues estratègies d'aproximació de la matriu $DT(s)$

En precisió guanya clarament l'estratègia analítica

Recordem que, en aproximar $DT(s)$ mitjançant diferències, es perd la convergència quadràtica de Newton per a passar a tenir un mètode només lineal. La bona notícia és que la convergència millora amb la qualitat de les aproximacions.

Un exemple simple de Tir Simple

Considerem el següent Problema de Valors a la Frontera

$$\ddot{w}(t) = \frac{3}{2}w(t)^2 \qquad \text{amb} \qquad w(0) = 4 \text{ i } w(1) = 1.$$

Introduction to Numerical Analysis, J. Stoer and R. Bulirsch, Springer-Verlag, New York, 1980. http://dx.doi.org/10.1007/978-1-4757-5592-3

El Problema de Valor Inicial associat és

$$\begin{pmatrix} \dot{w}(t) \\ \dot{s}(t) \end{pmatrix} = \begin{pmatrix} s(t) \\ \frac{3}{2}w(t)^2 \end{pmatrix} \qquad \text{amb} \qquad w(0) = 4 \text{ i } s(0) = \xi,$$

i denotem per

$$\Big(w(t;0;4;\xi),\ s(t;0;4;\xi)\Big)$$

la seva solució.

I la funció de tir associada és:

$$T(\xi) = w(1;0;4;\xi) - 1.$$

Un exemple simple de Tir Simple

Per trobar ξ, cal resoldre l'equació no-lineal $T(\xi) = 0$

Observació

Com que no sabem si, de fet, $T(\xi) = 0$ té solució, el "problema de la llavor" és realment seriós en aquest cas.

Aprofitant que, de fet, la funció T és $1-$dimensional, una manera de resoldre el problema de la llavor és localitzar un interval on tingui un canvi de signe. Això ja s'ha fet al document *Mètodes Numèrics per Equacions Diferencials Ordinàries; Part I: Mètodes explícits d'un pas* a l'apartat *Un exemple complet i comentat d'integració d'un PVI en* **C**. La taula de la Pàgina 89 del document citat diu que[4] $T(-5) = 11.05\cdots$ i $T(-10) = -3.400\cdots$.

En particular, per la continuïtat de solucions respecte de condicions inicials i el Teorema de Bolzano, el Problema de Tir (equivalentment el PVF) té solució $\xi \in (-5,-10)$.

[4] En realitat aquesta taula dóna els valors de $w(1;0;4;\xi)$ però $T(\xi) = w(1;0;4;\xi) - 1$.

Un exemple simple de Tir Simple

Per trobar ξ, cal resoldre l'equació no-lineal $T(\xi) = 0$

En aquesta situació, en comptes d'usar el Mètode de Newton 1−dimensional, és molt més robust (però una mica més lent) usar el *Mètode de Regula Falsi*.

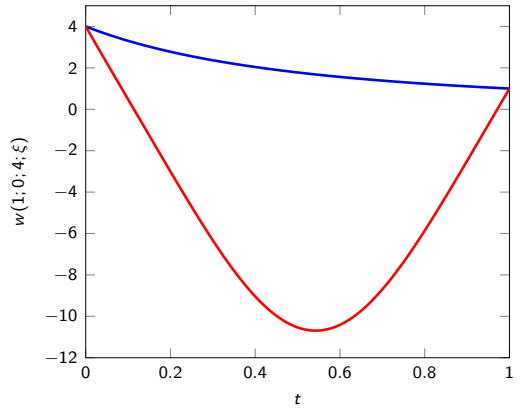
Fent això trobem la següent solució amb precisió[5] $1.e-12$:

ξ = -8.000000000000; T(ξ) = -6.92668145063635e-13;

[5] Tot i que $6.92668145063635e-13 < 1.e-12$ cal reflexionar sobre el significat de la paraula *precisió* en problemes de zeros de funcions i sobre que, de fet, *es necessiten dues precisions*.
Veure, mes avall, el codi del procediment **RegulaFalsi**.

Un exemple simple de Tir Simple

Les solucions



La corba blava de l'esquerra és la solució amb $\xi_0 = -8$.

La corba vermella mostra que, de fet, hi ha una altra solució d'aquest Problema de Valors a la Frontera amb ξ_0 bastant més petit que -10.0.

Exercici

Trobar la "solució vermella" d'aquest Problema de Valors a la Frontera.

Un exemple simple de Tir Simple

El codi principal

Tir Simple en un exemple senzill

```
#include <stdio.h>
#include "RKF78.h"

double Integra2tfin (double *, double[], double, void *,
          void (*) (double, double *, unsigned, double *, void *));
double RegulaFalsi (double, double, double, double (*)(double));

void CV (double t, double w[], unsigned n, double f[], void *pars) {
         f[0] = w[1]; f[1] = 1.5e0 * w[0] * w[0];
}

double TIR (double xi) { double t = 0.0, w[2] = { 4.0, xi };
       double max_error = Integra2tfin (&t, w, 1.0, NULL, CV);
       if (max_error > 1.e-14 || t < 1.0) return MAXDOUBLE;
       return (w[0] - 1.0);
}

int main () {
    double s = RegulaFalsi (-5.0, -10.0, 1.e-12, TIR);
    printf ("s_0 = %.15lf; F(s_0) = %.15g\n", s, TIR (s));
    return 0;
}
```


Un exemple simple de Tir Simple

El codi auxiliar

La funció **Integra2tfin**, vista i comentada anteriorment

```
double
Integra2tfin (double *t, double w[], double tfin, void *Parms,
   void (*CV) (double, double *, unsigned, double *, void *))
{
  double hmin = 1.e-8, hmax = 1.0, h = 1.e-6, err, maxerr = -1.0;

  while (*t + h < tfin)
    {
      if (RKF78_VF_forward (t, w, 2, &h, &err,
                            hmin, hmax, 1.e-15,
                            Parms, CV)) return MAXDOUBLE;
      maxerr = MAX (maxerr, err);
    }

  double tt = *t;
  h = tfin - *t;
  *t = tfin;
  maxerr = MAX (maxerr, RKF78_VF_1step (tt, w, 2, h, Parms, CV));
  return maxerr;
}
```

Un exemple simple de Tir Simple

El codi auxiliar II: **RegulaFalsi**

Una implementació del procediment RegulaFalsi

```
double RegulaFalsi (double a, double b, double prec,
                    double (*F) (double))
{ double fa = F (a);
  if (fabs (fa) < prec) return a;
  else if (fa == MAXDOUBLE) return MAXDOUBLE;

  double fb = F (b);
  if (fabs (fb) < prec) return b;
  else if (fb == MAXDOUBLE) return MAXDOUBLE;

  if (fa * fb > 0) return MAXDOUBLE;

  if (a > b) { double swap = a; a = b; b = swap;
               swap = fa; fa = fb; fb = swap; }

  while (b - a >= prec) { double m, fm;
      m = a - fa * (b - a) / (fb - fa);
      fm = F (m);
      if (fabs (fm) < prec) return m;
      else if (fm == MAXDOUBLE) return MAXDOUBLE;
      if (fa * fm < 0) { b = m; fb = fm; }
      else { a = m; fa = fm; }
  }
  return (a - fa * (b - a) / (fb - fa));
}
```


Un exemple no escalar de Tir Simple

Numerical methods for boundary value problems, Jeffrey Wong, Math 563 Lecture Notes, Duke University, Durham, 2020.

Un exemple no escalar de Tir Simple

L'equació de Van der Pol és (en forma de sistema)

$$\begin{cases} \dot{x} = y, \\ \dot{y} = \mu y(1 - x^2) - x, \end{cases}$$

que descriu el moviment d'un oscil·lador no lineal amb amortiment negatiu quan la seva amplitud és petita.

Quan $\mu > 0$ aquest sistema té un *cicle límit*; és a dir, *una orbita periòdica isolada*.

Aquest fet es pot formular com un Problema de Valor a la Frontera de la manera següent.

Un exemple no escalar de Tir Simple

No és difícil demostrar que el cicle límit ha de creuar l'eix de les y's. Per tant, com que el sistema és autònom, podem escollir aquest punt d'encreuament com origen de temps, fixant la condició inicial $x(0) = 0$.

Això dóna les següents condicions de frontera pel cicle límit:

$$\begin{cases} x(0) = 0, \\ x(0) = x(p), \\ y(0) = y(p), \end{cases}$$

on p denota el període del cicle límit, que és una incògnita. Observem que p ha de ser positiu.

Aquesta situació no és gaire còmoda, ja que tenim dues equacions al sistema d'EDO's mentre que tenim tres condicions de frontera.

Un exemple no escalar de Tir Simple

Podem "arreglar" el PVF amb el canvi de variable en el temps $\tau = \frac{t}{p}$, de manera que el període del cicle limit en les noves coordenades serà 1. El nou PVF en les noves variables que, abusant de notació també seran (x, y), en el paràmetre desconegut p i en el temps τ és[6]:

$$\begin{cases} \frac{dx}{d\tau} = py, \\ \frac{dy}{d\tau} = \mu py(1 - x^2) - px, \end{cases} \quad \text{amb} \quad \begin{cases} x(0) = 0, \\ x(0) = x(1), \\ y(0) = y(1). \end{cases}$$

[6] Queda clar que p de cap manera pot ser zero.

Un exemple no escalar de Tir Simple

El PVI associat és

$$\begin{cases} \frac{dx}{d\tau} = py, \\ \frac{dy}{d\tau} = \mu py\left(1-x^2\right) - px, \end{cases} \quad \text{amb} \quad \begin{cases} x(0) = 0, \\ y(0) = s; \end{cases}$$

quina solució denotarem per

$$\Big(x(\tau;s;p),\, y(\tau;s;p)\Big).$$

Observem que, aquest és un PVI en dimensió 2, amb una condició inicial indeterminada s i un paràmetre p, especificats explícitament a les dues components de la solució.

A partir de les condicions de frontera $x(1;s;p) = x(0;s;p) = 0$ i $y(1;s;p) = y(0;s;p) = s$, s'obté la funció de tir, que depèn de la condició inicial s i del paràmetre p:

$$\mathrm{T}(s,p) = \big(x(1;s;p),\, y(1;s;p) - s\big).$$

Un exemple no escalar de Tir Simple

Observació: Com augmentar la ineficiència dels càlculs

Alternativament es podria haver tractat el paràmetre p com una variable més del PVI. Això implicaria afegir una tercera equació (trivial i independent) a l'EDO: $\frac{dp}{d\tau} = 0$ i una tercera condició inicial al PVI: $p(0) = r$. Si ara denotem la solució del PVI per

$$\big(x(\tau;s;r),\, y(\tau;s;r),\, p(\tau;s;r)\big),$$

obtenim la funció de tir

$$\mathrm{T}(s,r) = \big(x(1;s;r),\, y(1;s;r) - s\big).$$

Observem que, en resoldre el PVI, calculem la tercera component de la solució $p(\tau;s;r)$ però no l'utilitzem a la funció de tir (ja que l'equació $\frac{dp}{d\tau} = 0$ és independent de les altres). Acabem doncs de descobrir una manera magnífica de complicar i retardar innecessàriament els càlculs.

Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton, amb l'estratègia analítica

D'ara endavant resoldrem l'equació de tir

$$\mathrm{T}(s,p) = \big(x(1;s;p),\, y(1;s;p) - s\big),$$

mitjançant el Mètode de Newton; usant el mètode desenvolupat a la Pàgina 12.

Per fer això, necessitem la matriu Jacobiana de T a un punt $\big(\mathbf{s}^{(k)},\, \mathbf{p}^{(k)}\big)$:

$$J\,\mathrm{T}(s,p)\Big|_{\left(\mathbf{s}^{(k)},\,\mathbf{p}^{(k)}\right)} = \left(\begin{array}{cc} \frac{\partial x(1;s;p)}{\partial s} & \frac{\partial x(1;s;p)}{\partial p} \\ \frac{\partial y(1;s;p)}{\partial s} - 1 & \frac{\partial y(1;s;p)}{\partial p} \end{array} \right)\Bigg|_{(s,p)=\left(\mathbf{s}^{(k)},\,\mathbf{p}^{(k)}\right)}.$$

Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton:
Càlcul de la matriu Jacobiana de T mitjançant l'estratègia analítica

Recordem que $\begin{pmatrix} \frac{\partial x(\tau;s;p)}{\partial s} \\ \frac{\partial y(\tau;s;p)}{\partial s} \end{pmatrix}$ satisfà el PVI:

$$\begin{cases} \begin{pmatrix} \frac{du}{d\tau} \\ \frac{dv}{d\tau} \end{pmatrix} & = \boldsymbol{D}_{(x,y)} \begin{pmatrix} py \\ \mu py\left(1-x^2\right) - px \end{pmatrix}\Big|_{(x,y)=(x(\tau;s;p),\,y(\tau;s;p))} \begin{pmatrix} u \\ v \end{pmatrix} \\ & = \begin{pmatrix} 0 & p \\ -2\mu pxy - p & \mu p\left(1-x^2\right) \end{pmatrix}\Big|_{(x,y)=(x(\tau;s;p),\,y(\tau;s;p))} \begin{pmatrix} u \\ v \end{pmatrix}, \\ \begin{pmatrix} u(0) \\ v(0) \end{pmatrix} & = \begin{pmatrix} 0 \\ 1 \end{pmatrix}; \end{cases}$$

Notem que: s és la segona component del vector de condicions inicials i per tant, $(u(0), v(0)) = e_2 = (0,1)$.

Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton:
Càlcul de la matriu Jacobiana de T mitjançant l'estratègia analítica

El PVI anterior, en forma explícita, és:

$$\begin{cases} \frac{du(\tau)}{d\tau} = pv(\tau), \\ \frac{dv(\tau)}{d\tau} = \mu p\big(1 - x(\tau;s;p)^2\big)v(\tau) - p\big(2\mu x(\tau;s;p)y(\tau;s;p) + 1\big)u(\tau), \\ u(0) = 0, \\ v(0) = 1. \end{cases}$$

Per tant,

$$\left(\frac{\partial x\big(1;\ \mathbf{s}^{(k)};\ \mathbf{p}^{(k)}\big)}{\partial s}, \frac{\partial y\big(1;\ \mathbf{s}^{(k)};\ \mathbf{p}^{(k)}\big)}{\partial s}\right)^\top$$

es pot calcular a partir de la solució $\big(x(\tau;s;p),\ y(\tau;s;p)\big)$ del PVI de la Pàgina 31 a $\tau = 1$, amb $(s,p) = \big(\mathbf{s}^{(k)},\ \mathbf{p}^{(k)}\big)$.

Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton:
Càlcul de la matriu Jacobiana de T mitjançant l'estratègia analítica

Per altra banda, $\left(\frac{\partial x(\tau;s;p)}{\partial p}, \frac{\partial y(\tau;s;p)}{\partial p}\right)^\top$ satisfà el PVI següent[7]:

$$\begin{cases} \begin{pmatrix} \frac{dz}{d\tau} \\ \frac{dw}{d\tau} \end{pmatrix} = D_{(x,y)} \begin{pmatrix} py \\ \mu py\big(1-x^2\big) - px \end{pmatrix}\Big|_{(x,y)=(x(\tau;s;p),\ y(\tau;s;p))} \begin{pmatrix} z \\ w \end{pmatrix} + \\ \qquad D_p \begin{pmatrix} py \\ \mu py\big(1-x^2\big) - px \end{pmatrix}\Big|_{(x,y)=(x(\tau;s;p),\ y(\tau;s;p))} \\ \qquad = \begin{pmatrix} 0 & p \\ -2\mu px(\tau;s;p)y(\tau;s;p) - p & \mu p\big(1 - x(\tau;s;p)^2\big) \end{pmatrix} \begin{pmatrix} z \\ w \end{pmatrix} + \\ \qquad \begin{pmatrix} y(\tau;s;p) \\ \mu y(\tau;s;p)\big(1 - x(\tau;s;p)^2\big) - x(\tau;s;p) \end{pmatrix}, \\ \begin{pmatrix} z(0) \\ w(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \end{cases}$$

[7] Atenció!!: p és un paràmetre (no una variable); les equacions variacionals són diferents.

Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton:
Càlcul de la matriu Jacobiana de T mitjançant l'estratègia analítica

Aquest darrer PVI és equivalent a

$$\begin{cases} \frac{dz(\tau)}{d\tau} = pw(\tau) + y(\tau;s;p), \\ \frac{dw(\tau)}{d\tau} = \mu\big(1 - x(\tau;s;p)^2\big)\big(pw(\tau) + y(\tau;s;p)\big) - \\ \qquad 2\mu p\, x(\tau;s;p)\, y(\tau;s;p)\, z(\tau) - \big(pz(\tau) + x(\tau;s;p)\big), \\ z(0) = 0, \\ w(0) = 0; \end{cases}$$

i per tant,

$$\left(\frac{\partial x\big(1;\ \mathbf{s}^{(k)};\ \mathbf{p}^{(k)}\big)}{\partial p}, \frac{\partial y\big(1;\ \mathbf{s}^{(k)};\ \mathbf{p}^{(k)}\big)}{\partial p}\right)^\top$$

també es pot calcular a partir de la solució $\big(x(\tau;s;p),\ y(\tau;s;p)\big)$ del PVI Pàgina 31 a $\tau = 1$, amb $(s,p) = \big(\mathbf{s}^{(k)},\ \mathbf{p}^{(k)}\big)$.

Un exemple no escalar de Tir Simple

En conclusió: Resolent l'equació del tir amb Newton i l'estratègia analítica

A cada iterat de Newton cal integrar el *PVI total* fins a $\tau = 1$.
Aquest PVI és (en forma compacta):

$$\begin{cases} \frac{dx}{d\tau} = py, \\ \frac{dy}{d\tau} = p\gamma, \\ \frac{du}{d\tau} = pv, \\ \frac{dv}{d\tau} = \alpha v - \beta u, \\ \frac{dz}{d\tau} = pw + y, \\ \frac{dw}{d\tau} = \alpha w + \gamma - \beta z, \\ x(0) = 0, \\ y(0) = s, \\ u(0) = 0, \\ v(0) = 1, \\ z(0) = 0, \\ w(0) = 0. \end{cases}$$

on:
$\alpha = \mu p\big(1 - x^2\big)$,
$\gamma = \mu\big(1 - x^2\big)y - x$,
$\beta = p(2\mu xy + 1)$.

Un exemple no escalar de Tir Simple

El càlcul numèric: Resultats

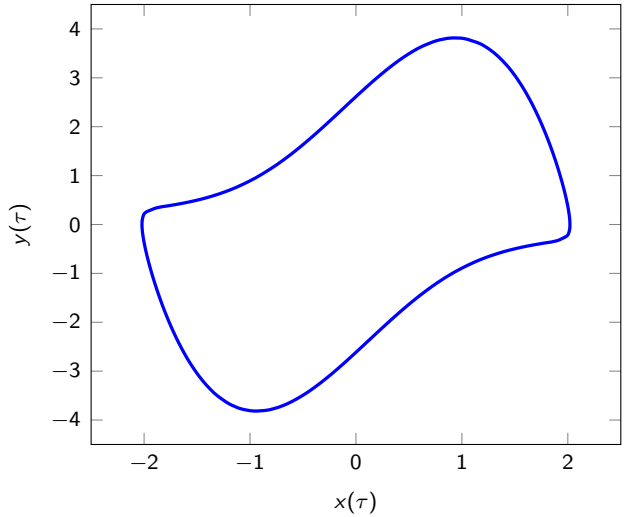
Taula: Resultats obtinguts en calcular el cicle límit amb el mètode anterior.
Observem que aquest mètode, de fet, no funciona sense "enginyeria"".

Llavor		Solució		criteri de parada	Comentaris
s_0	p_0	s	p		
2	1	-2.614972625631902	15.259748959349684	**1**	Sorpresa!!: convergeix a la primera. Problema: p no és correcte; es donen dues voltes al cicle límit. Perquè passa això? Com arreglar-ho?
2.6	0	2.614972625631902	0.0	**0**	$p = 0$ està prohibit. En aquest cas, l'algorisme fa el que ha de fer malgrat intentem prendre-li el pèl. Es queda quiet a la llavor.
2.6	3	Error: no s'ha pogut integrar el camp fins a t_{final}			Aquí veiem el problema de la llavor en directe.
2	5	2.614972625631902	114.448117195122492	**1**	Exactament igual que la primera fila però amb el problema del període agreujat enormement. Repetim, perquè passa això?
2.6	7.62	2.614972625631903	7.629874479674842	**0**	Ara "l'hem clavat" però és perquè, de fet, hem entrat la solució com a llavor. Per cert com savíem el període?

Criteri de parada: **0** $\leftarrow \|T(s_k, p_k)\| < \texttt{tolf} = 1.e-14$
1 $\leftarrow \|s_{k+1} - s_k, p_{k+1} - p_k\| < \texttt{tolx} = 1.e-12$

Un exemple no escalar de Tir Simple

El cicle límit en dibuix. Ja es comença a veure el "canard"



Un exemple no escalar de Tir Simple

Una implementació de la solució de l'Equació del Tir pel mètode de Newton, en C

El Camp Vectorial aprofitant l'expressió compacta per eficiencia

```
typedef struct { double mu; double p; } ODE_Parameters;

void CV (double t, double x[], unsigned n, double f[], void *pars)
{ double a = ((ODE_Parameters *) pars)->mu * (1.e0 - x[0] * x[0]),
         b = ((ODE_Parameters *) pars)->mu * x[0] * x[1];
  b = ((ODE_Parameters *) pars)->p * (b + b + 1.e0);
  f[5] = a * x[1] - x[0];
  a *= ((ODE_Parameters *) pars)->p;

  f[0] = ((ODE_Parameters *) pars)->p * x[1];
  f[1] = ((ODE_Parameters *) pars)->p * f[5];
  f[2] = ((ODE_Parameters *) pars)->p * x[3];
  f[3] = a * x[3] - b * x[2];
  f[4] = ((ODE_Parameters *) pars)->p * x[5] + x[1];
  f[5] += (a * x[5] - b * x[4]);
}
```


Un exemple no escalar de Tir Simple

Una implementació de la solució de l'Equació del Tir pel mètode de Newton, en C

El main

```
int main ()
{ double sp[2] = { 2.6, 7.62 };

  int cntrl = TIRSol (2.0e0, sp);

  if (cntrl < 2)
    { ODE_Parameters parms = { 2.e0, sp[1] };
      double tfin = 1.0, t = 0.0, h = 1.e-6, err;
      double x[6] = { 0.e0, sp[0], 0.e0, 1.e0, 0.e0, 0.e0 };

      printf ("## s = %.15lf; p = %.15lf; stop = %d\n",
                                          sp[0], sp[1], cntrl);

      while (t + h < tfin)
        { RKF78_VF_forward (&t, x, 6, &h, &err, 1.e-8, 1.0, 1.e-15,
                            &parms, CV);
          printf ("%.15lf %.15lf %.15lf %.15g\n", t, x[0], x[1], err);
        }
      err = RKF78_VF_1step (t, x, 6, tfin - t, &parms, CV);
      printf ("%.15lf %.15lf %.15lf %.15g\n", tfin, x[0], x[1], err);
    }
  else printf ("Acabant amb codi d'error %d\n", cntrl);

  return cntrl;
}
```

La idea és que sp[2] contindrà s i p.
Més precisament, sp[0]$= s$ i sp[1]$= p$.

Un exemple no escalar de Tir Simple

Una implementació de la solució de l'Equació del Tir pel mètode de Newton, en **C**

El procediment clau **TirSol**, que implementa el mètode de Newton

```
#define MAXNewtonIter 100
double supnorm (double x, double y)
{ x = fabs (x); y = fabs (y); return MAX (x, y); }

int TIRSol (double mu, double *sp)
{ double tolx = 1.e-12, tolf = 1.e-14, t = 0.0, maxerr;
  ODE_Parameters parms = { mu, 0.0 };

  for (int i = 0; i < MAXNewtonIter; i++)
    { double x[6] = { 0.e0, sp[0], 0.e0, 1.e0, 0.e0, 0.e0 };

      parms.p = sp[1]; t = 0.0;
      maxerr = Integra2tfin (&t, x, 6, 1.e0, &parms, CV);
      if (maxerr > tolf || t < 1.0) return 66;
      if (supnorm (x[0], x[1] - sp[0]) < tolf) return 0;

      double detJT = x[2] * x[5] - x[4] * (x[3] - 1);
      if (fabs (detJT) < tolf) return 6;
      double ds = (x[0] * x[5] - x[4] * (x[1] - sp[0])) / detJT,
             dp = ((x[1] - sp[0]) * x[2] - x[0] * (x[3] - 1)) / detJT;

      sp[0] -= ds; sp[1] -= dp;
      if (supnorm (ds, dp) < tolx) return 1;
    }
  return 4;
}
```

sp[0] conté s i sp[1] conté p.

Un exemple no escalar de Tir Simple

Una implementació de la solució de l'Equació del Tir pel mètode de Newton, en **C**

Comentaris al procediment **TirSol**

```
maxerr = Integra2tfin (&t, x, 6, 1.e0, &parms, CV);
```

Amb aquesta instrucció s'integra el *PVI total* fins $\tau = 1$, de manera que

- $\mathsf{T}(s,p) = (x(1;s;p),\, y(1;s;p) - s) = (\texttt{x[0]},\, \texttt{x[1]} - \texttt{sp[0]})$, i
- $J\mathsf{T}(s,p) = \begin{pmatrix} \frac{\partial x(1;s;p)}{\partial s} & \frac{\partial x(1;s;p)}{\partial p} \\ \frac{\partial y(1;s;p)}{\partial s} - 1 & \frac{\partial y(1;s;p)}{\partial p} \end{pmatrix} = \begin{pmatrix} u(1;s;p) & z(1;s;p) \\ v(1;s;p) - 1 & w(1;s;p) \end{pmatrix} = \begin{pmatrix} \texttt{x[2]} & \texttt{x[4]} \\ \texttt{x[3]} - 1 & \texttt{x[5]} \end{pmatrix}$.

```
double detJT = x[2] * x[5] - x[4] * (x[3] - 1);
if (fabs (detJT) < tolf) return 6;
double ds = (x[0] * x[5] - x[4] * (x[1] - sp[0])) / detJT,
       dp = ((x[1] - sp[0]) * x[2] - x[0] * (x[3] - 1)) / detJT;
```

És la resolució del sistema d'equacions

$$\begin{pmatrix} \texttt{x[2]} & \texttt{x[4]} \\ \texttt{x[3]} - 1 & \texttt{x[5]} \end{pmatrix} \cdot \begin{pmatrix} \texttt{ds} \\ \texttt{dp} \end{pmatrix} = J\mathsf{T}(s^{(k)}, p^{(k)}) \cdot \begin{pmatrix} \texttt{ds} \\ \texttt{dp} \end{pmatrix} = \mathsf{T}(s^{(k)}, p^{(k)}) = \begin{pmatrix} \texttt{x[0]} \\ \texttt{x[1]} - \texttt{sp[0]} \end{pmatrix},$$

per a cada iterat de Newton.

Un exemple no escalar de Tir Simple

Una implementació de la solució de l'Equació del Tir pel mètode de Newton, en **C**

La funció ja clàssica **Integra2tfin** adaptada a aquest problema

```
#include <stdio.h>
#include "RKF78.h"

double
Integra2tfin (double *t, double x[], unsigned n,
   double tfin, void *parms,
   void (*CV) (double, double *, unsigned, double *, void *))
{
  double h = 1.e-6, err, maxerr = -1.0;

  while (*t + h < tfin)
    {
      if (RKF78_VF_forward (t, x, n, &h, &err,
                  1.e-8, 1.0, 1.e-15, parms, CV)) return MAXDOUBLE;
      maxerr = MAX (maxerr, err);
    }

  double tt = *t;
  maxerr = MAX (maxerr,
                RKF78_VF_1step (tt, x, n, tfin - tt, parms, CV));

  *t = tfin;
  return maxerr;
}
```


Un exemple no escalar de Tir Simple

Ara provem continuació, amb el mètode de Newton, amb l'estratègia analítica a veure si s'arregla el problema de la llavor

Per $\mu = 0$ el sistema (a l'esquerra) és l'equació d'un centre que té la solució general de la dreta:

$$\begin{cases} \frac{dx}{d\tau} = py, \\ \frac{dy}{d\tau} = -px, \end{cases} \qquad \begin{cases} x(\tau) &= c_1 \sin(p\tau) + c_2 \cos(p\tau), \\ y(\tau) &= c_1 \cos(p\tau) - c_2 \sin(p\tau), \end{cases}$$

Llavors, $\mathbb{R}^2$ està foliat de solucions de període 2π, que són cercles amb centre l'origen. Els cicles límit de l'equació de Van der Pol per $\mu \gtrapprox 0$ bifurquen del cercle que talla l'eix de les y's als punts $y = \pm 2.0$

Per tant, per a trobar la bona llavor per $\mu = 2$, farem continuació a partir de $\mu = 0$ amb llavor inicial $(s,p) \approx (2, 2\pi)$.

Els resultats d'aquests càlculs es mostren a la pàgina següent.

Un exemple no escalar de Tir Simple

Continuació: Els resultats

Taula: Solucions de l'Equació del Tir per diversos valors de μ. La llavor per a cada μ és la solució del problema anterior. La de $\mu = 0.1$, com ja s'ha dit a la pàgina anterior, és $(2, 6.283185307179586)$. El càlcul de la solució per $\mu = 2$ ha tornat **0** com a criteri de parada (és a dir, $\|\mathrm{T}(s_k, p_k)\| < \texttt{tolf} = 1.e-14$).

μ	s	p
0.1	2.001770546296870	6.287111272288729
0.2	2.007078652833994	6.298876713852454
0.3	2.015913073377634	6.318443203454112
0.4	2.028252964767562	6.345743276798672
0.5	2.044064968368357	6.380675801773585
0.6	2.063299598757797	6.423100483284693
0.7	2.085887430260811	6.472832473367605
0.8	2.111735698355819	6.529638239834604
0.9	2.140725995687918	6.593233878696322
1.0	2.172713692622547	6.663286859323128
1.1	2.207529516339591	6.739421733280076
1.2	2.244983387915799	6.821229667821767
1.3	2.284870207603620	6.908280930343552
1.4	2.326976903832062	7.000138858596775
1.5	2.371089835575797	7.096373589684782
1.6	2.417001628658980	7.196573966960774
1.7	2.464516724267375	7.300356529964430
1.8	2.513455242272576	7.407371140449970
1.9	2.563655103747034	7.517303405212407
2.0	2.614972625631901	7.629874479674839

Un exemple no escalar de Tir Simple

Reconvertint una implementació del mètode de Newton en **C** en continuació aprofitant l'avantatge que s'ha programat el procediment **TirSol** amb paràmetres minimals

Simplement, a la implementació del mètode de Newton en **C**, cal canviar

```
int main ()
{ double sp[2] = { 2.6, 7.62 };

  int cntrl = TIRSol (2.0e0, sp);
```

per

```
#define mustep 0.1
int main () { int cntrl;
    double sp[2] = { 2.0, 6.283185307179586 };

    for (double mu = mustep ; mu < 2.0 ; mu += mustep) {
        printf("##-mu-=-%lf;-Llavor-=-(%.15lf,-%.15lf);-",
                                                mu, sp[0], sp[1]);
        cntrl = TIRSol (mu, sp);
        if (cntrl > 1) return cntrl ;
        printf ("Solucio:-(%.15lf,-%.15lf)\n", sp[0], sp[1]);
    }

    cntrl = TIRSol (2.0e0, sp);
```

Aquesta és la part de continuació. Interposada entre la llavor inicial i el càlcul final amb la bona llavor.

Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton:
Càlcul aproximat de la matriu Jacobiana de T mitjançant diferències

Usarem l'aproximació numèrica de la Pàgina 18 per a modificar la implementació de la Pàgina 41 i següents. Únicament cal fer *dos* canvis (apart dels que cal fer al llaç del `main` que escriu la solució calculada per RKF7(8)).

El primer canvi és conseqüència que, en aquesta estratègia, solament cal fer avaluacions de la funció de tir T a punts $(s; p)$ diferents. Per tant, ara, el camp vectorial és el del PVI inicial sense els dos parells d'equacions variacionals:

El camp vectorial base, de dimensió 2

```
void CV (double t, double x[], unsigned n, double f[], void *pars) {
  f[0] = ((ODE_Parameters *) pars)->p * x[1];
  f[1] = ((ODE_Parameters *) pars)->mu * f[0] * (1.e0 - x[0] * x[0]
          - ((ODE_Parameters *) pars)->p * x[0];
}
```


Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton:
Càlcul aproximat de la matriu Jacobiana de T mitjançant diferències

El segon i principal canvi afecta bastant globalment al procediment **TirSol** i és conseqüència dels canvis al camp vectorial, que afecten a les avaluacions de la funció de tir T.

En primer lloc cal calcular (tenint en compte la dimensió d'`x`):

1. $x(1; s; p)$ i $y(1; s; p)$ per a obtenir (pàgina 33)

$$\mathrm{T}(s, p) = \left(x(1; s; p),\ y(1; s; p) - s\right)^{\top}.$$

Cal canviar: la integració del PVI i les equacions variacionals

```
double x[6] = { 0.e0, sp[0], 0.e0, 1.e0, 0.e0, 0.e0 };
parms.p = sp[1]; t = 0.0;
maxerr = Integra2tfin (&t, x, 6, 1.e0, &parms, CV);
```

per la integració del PVI *sense* les equacions variacionals

1.
```
double x[2] = { 0.e0, sp[0] };
parms.p = sp[1]; t = 0.0;
maxerr = Integra2tfin (&t, x, 2, 1.e0, &parms, CV);
```

Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton:
Càlcul aproximat de la matriu Jacobiana de T mitjançant diferències

Seguidament, cal aproximar la matriu JT amb diferències de primer ordre. Usarem les fórmules de la Pàgina 18.

El que cal calcular és:

2. $x(1; s+h; p)$ i $y(1; s+h; p)$ per a obtenir

$$\mathrm{T}(s+h, p) = \left(x(1; s+h; p),\ y(1; s+h; p) - (s+h)\right)^\top.$$

Llavors, la *primera* columna de $D\mathrm{T}(s,p)$ és:

$$\frac{\partial \mathrm{T}(s,p)}{\partial s} \approx \frac{\mathrm{T}(s+h,p) - \mathrm{T}(s,p)}{h} =$$

$$\frac{\left(x(1; s+h; p) - x(1; s; p),\ y(1; s+h; p) - y(1; s; p) - h\right)^\top}{h} =$$

$$\left(\frac{x(1; s+h; p) - x(1; s; p)}{h},\ \frac{y(1; s+h; p) - y(1; s; p)}{h}\right)^\top - (0,1)^\top.$$

Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton:
Càlcul aproximat de la matriu Jacobiana de T mitjançant diferències

El canvi important: Newton amb JT a partir de les variacionals

```
double detJT = x[2] * x[5] - x[4] * (x[3] - 1);
if (fabs (detJT) < tolf) return 6;
double ds = (x[0] * x[5] - x[4] * (x[1] - sp[0])) / detJT,
       dp = ((x[1] - sp[0]) * x[2] - x[0] * (x[3] - 1)) / detJT;
```

per Newton amb JT calculada amb diferències de primer ordre

2
```
t = 0.0;  double JT_L[2] = { 0.e0, sp[0] + tolx };
maxerr = Integra2tfin (&t, JT_L, 2, 1.e0, &parms, CV);
if (maxerr > tolf || t < 1.0) return 66;
JT_L[0] = (JT_L[0] - x[0]) / tolx;
JT_L[1] = (JT_L[1] - x[1]) / tolx - 1.e0;
```

Aquí tenim `parms.p = sp[1];` que s'havia fixat i no modificat a 1

3
```
double JT_R[2] = { 0.e0, sp[0]};
t = 0.0;  parms.p = sp[1] + tolx;
maxerr = Integra2tfin (&t, JT_R, 2, 1.e0, &parms, CV);
if (maxerr > tolf || t < 1.0) return 66;
JT_R[0] = (JT_R[0] - x[0]) / tolx;
JT_R[1] = (JT_R[1] - x[1]) / tolx;
```

Iteració de newton en les noves variables JT_L i JT_R.

```
double detJTsp = JT_L[0] * JT_R[1] - JT_L[1] * JT_R[0];
if (fabs (detJTsp) < tolf) return 6;
double ds = (x[0] * JT_R[1] - JT_R[0] * (x[1] - sp[0])) / detJTsp,
       dp = ((x[1] - sp[0]) * JT_L[0] - x[0] * JT_L[1]) / detJTsp;
```


Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton:
Càlcul aproximat de la matriu Jacobiana de T mitjançant diferències

A més, cal calcular:

3. $x(1; s; p+h)$ i $y(1; s; p+h)$ per a obtenir

$$\mathrm{T}(s, p+h) = \left(x(1; s; p+h),\ y(1; s; p+h) - s\right)^\top.$$

Llavors, la *segona* columna de $D\mathrm{T}(s,p)$ és:

$$\frac{\partial \mathrm{T}(s,p)}{\partial p} \approx \frac{\mathrm{T}(s,p+h) - \mathrm{T}(s,p)}{h} =$$

$$\left(\frac{x(1; s; p+h) - x(1; s; p)}{h},\ \frac{y(1; s; p+h) - y(1; s; p)}{h}\right)^\top.$$

Observació

A la implementació dels càlculs anteriors hem pres $h = \texttt{tolx}$ seguint la recomanació de la Pàgina 18.

Un exemple no escalar de Tir Simple

Resolent l'equació del tir amb el mètode de Newton:
Conclusions al càlcul aproximat de la matriu Jacobiana de T mitjançant diferències

Tal com ja s'havia dit, el mètode de Newton amb el càlcul aproximat de la matriu JT mitjançant diferències és més lent (per un factor aproximadament 2) i s'incrementa una mica l'error però *sembla* més estable i menys sensible al problema de la llavor.

De tota manera tot això depèn de que aquest problema és, de fet, analític i té les solucions fàcils de calcular (per valors de μ raonables).

Tir Múltiple: Motivació

En general, poden haver-hi seriosos problemes en avaluar la funció de tir amb precisió.

Aquests problemes ocorren molt sovint en EDO's on la solució varia molt ràpidament respecte del temps (per exemple les EDO's "stiff"). També hi ha cassos en els que hi ha dependència sensible respecte de condicions inicials, degut a l'expressió de la solució general de l'EDO (amb funcions "complicades" com exponencials, tangents hiperbòliques, etc).

Un exemple il·lustratiu d'aquest darrer fenomen és l'Exemple 1 de la Secció 7.3.4 de

Introduction to Numerical Analysis, J. Stoer and R. Bulirsch, Springer-Verlag, New York, 1980. http://dx.doi.org/10.1007/978-1-4757-5592-3

Òbviament, tots aquests problemes s'agreugen com més gran és la separació entre a i b o, equivalentment, la mida de $b-a$.

Tir Múltiple

El Tir Múltiple pretén resoldre aquest problema atacant i modificant la llargada de l'interval $[a,b]$ de manera que tractem amb intervals petits on les solucions siguin difícils de seguir (integrar), i intervals grans on les solucions siguin suaus.

Observem que això no es pot fer gratis. L'interval $[a,b]$ està fixat. L'estratègia del *Tir Múltiple* és una de les usuals en Mètodes Numèrics: *divideix i venceràs*. Dit d'una altra manera, canviem (com el Robert i les cabres) un problema difícil per uns quants de fàcils (o més fàcils).

L'estratègia del Tir Múltiple consisteix a dividir l'interval $[a,b]$ en m subintervals $[\tau_i,\tau_{i+1}]$, amb $i=0,1,2,\ldots,m-1$, de manera que $\tau_0=a$ i $\tau_m=b$. Com ja s'ha dit la mida de cada interval està determinada per la dificultat de seguir la solució de l'EDO en aquell interval: més gran quan les solucions són suaus en aquella zona de l'eix del temps; més petit on les solucions siguin difícils d'integrar.

Tir Múltiple: el multi-PVI

Ara considerem el *Problema de Valor Inicial Múltiple* (o el *multi-problema de valor inicial*):

$$\begin{cases} \dot{x}(t)=f(t,x(t)), \text{ amb} \\ x(\tau_0)=\xi_0, \\ x(\tau_1)=\xi_1, \\ \qquad\vdots \\ x(\tau_{m-1})=\xi_{m-1}, \end{cases}$$

i denotem per $x(t;\tau_i;\xi_i)$ les seves solucions. És a dir, $x(\tau_i;\tau_i;\xi_i)=\xi_i$.

La funció de Tir Múltiple: una definició

La *Funció de Tir Múltiple* es defineix fàcilment 😇 com[8]:

$$\mathrm{T}:\ \mathbb{R}^{m\cdot n}\longrightarrow\mathbb{R}^{m\cdot n}$$

$$\begin{pmatrix}\xi_0\\ \xi_1\\ \vdots\\ \xi_{m-2}\\ \xi_{m-1}\end{pmatrix}\longmapsto\begin{pmatrix} x(\tau_1;\tau_0;\xi_0)-\xi_1\\ x(\tau_2;\tau_1;\xi_1)-\xi_2\\ \vdots\\ x(\tau_{m-1};\tau_{m-2};\xi_{m-2})-\xi_{m-1}\\ \psi\big(\xi_0,x(b;\tau_{m-1};\xi_{m-1})\big)\end{pmatrix}$$

Com abans, un problema de Tir Múltiple consisteix a trobar un zero de la Funció de Tir Múltiple

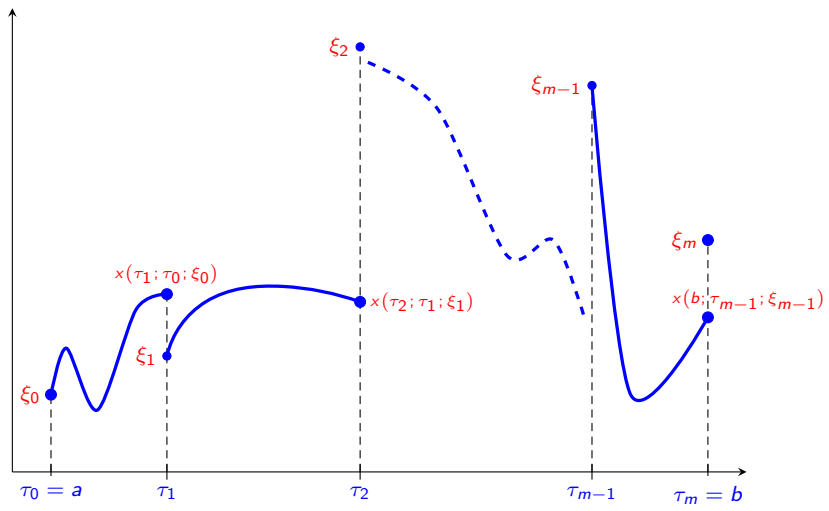
$$\mathrm{T}(\xi_0,\xi_1,\ldots,\xi_{m-1})=\mathbf{0}$$

que, en general, és un sistema d'equacions no-lineal $m\cdot n-$dimensional.

[8] Observem que hem comès un abús de notació: les variables independents de la funció T *no* són $\xi_0,\xi_1,\ldots,\xi_{m-1}$; en realitat són (la unió de) les n components de cada un d'aquests vectors. Anàlogament amb les components de la funció T que són $m\cdot n$ (no n).

Tir Múltiple: un exemple gràfic en dimensió (simulada) 1

aka val més una imatge que mil paraules



Tir Múltiple

Exercici / Observació

Dissenyar el pseudocodi de la Funció de Tir Múltiple.

Observació: No caure a la temptació de "fabricar" un camp de vectors de dimensió $m \cdot n$ que sigui m còpies del camp de vectors original, que té dimensió n.

Perquè?: Simplement el "*temps de vol*" a la $i-$èssima "*variable de dimensió n*" de la Funció de Tir és $\tau_{i+1} - \tau_i$ que, en principi, és diferent per a cada i.

Tir Múltiple: notes d'implementació

Què cal calcular i què no s'ha de fer

En general tots el comentaris al problema de la llavor (a les dues estratègies de càlcul de la matriu Jacobiana de T, en usar continuació, etc), s'apliquen al cas de Tir Múltiple amb la diferència bàsica (però no irrellevant) que, ara, *tot és més gran*. Hem passat de dimensió n a dimensió $m \cdot n$.

Per exemple, un exercici com el de Van der Pol partint l'interval $[0, 1]$ en 10 subintervals de temps implica que la Funció de Tir té dimensió 20 i la matriu JT té dimensió 20×20 (encara que moltes — com a mínim 180 — de les seves entrades seran zero — veure la Pàgina 63).

Recordem però que *el temps de vol de la $i-$èssima variable independent de dimensió n de la Funció de Tir és* $\tau_{i+1} - \tau_i$ que, en principi, és diferent per a cada i[9].

[9] La utilitat del tir múltiple ve de l'adaptació dels intervals de temps a les zones de la solució amb dificultats d'integració. En particular no és raonable pensar o forçar que els temps de vol de cada interval siguin els mateixos.

Tir Múltiple: notes d'implementació (II)

Què i com cal calcular??

En primer lloc la pròpia funció de tir múltiple.

Com hem dit abans (degut a que els temps de vol de cada interval de tir poden ser diferents), no s'hauria de caure a la temptació d'usar un camp de vectors de dimensió $m \cdot n$ que sigui m còpies del camp de vectors original (de dimensió n).

Hi ha diverses maneres d'implementar la funció de tir múltiple, depenent del problema a resoldre.

Tir Múltiple: notes d'implementació (III)

Què i com cal calcular: La matriu $JT(\xi_0, \xi_1, \ldots, \xi_{m-1})$ té una estructura molt particular

$$JT(\xi_0, \xi_1, \ldots, \xi_{m-1}) = \begin{pmatrix} \mathbf{J}_0 & -\mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n & \cdots & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{J}_1 & -\mathbf{I}_n & \mathbf{0}_n & \cdots & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{0}_n & \mathbf{J}_2 & -\mathbf{I}_n & \cdots & \mathbf{0}_n \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{0}_n & \mathbf{0}_n & \mathbf{0}_n & \cdots & \mathbf{J}_{m-2} & -\mathbf{I}_n \\ \boldsymbol{D_1\psi} & \mathbf{0}_n & \mathbf{0}_n & \cdots & \mathbf{0}_n & \boldsymbol{D_2\psi} \end{pmatrix}$$

on:

$\mathbf{0}_n$ és la matriu zero de dimensió $n \times n$,

$\mathbf{I}_n$ és la matriu identitat de dimensió $n \times n$,

$\mathbf{J}_i := D_{\xi_i} x(\tau_{i+1}; \tau_i; \xi_i)$ per $i = 0, 1, 2 \ldots, m-2$,

$\boldsymbol{D_1\psi} := D_1\psi\Big(\xi_0, x(b; \tau_{m-1}; \xi_{m-1})\Big)$,

$\boldsymbol{D_2\psi} := D_2\psi\Big(\xi_0, x(b; \tau_{m-1}; \xi_{m-1})\Big) \cdot D_{\xi_{m-1}} x(b; \tau_{m-1}; \xi_{m-1})$.

Tir Múltiple: notes d'implementació (i IV)

Resumint: Què cal calcular de la matriu $JT(\xi_0, \xi_1, \ldots, \xi_{m-1})$

- Precalcular $D_1\psi(\cdot,\cdot)$ i $D_2\psi(\cdot,\cdot)$.

A més, a cada iteració,

- usant les equacions variacionals del sistema calcular[10] $\mathbf{J}_i = D_{\xi_i} x(\tau_{i+1}; \tau_i; \xi_i)$ per $i = 0, 1, 2 \ldots, m-2$, i $D_{\xi_{m-1}} x(b; \tau_{m-1}; \xi_{m-1})$;
- i avaluar $D_1\psi\Big(\xi_0, x(b; \tau_{m-1}; \xi_{m-1})\Big)$ i $D_2\psi\Big(\xi_0, x(b; \tau_{m-1}; \xi_{m-1})\Big) \cdot D_{\xi_{m-1}} x(b; \tau_{m-1}; \xi_{m-1})$.

Addicionalment, és fonamental tunejar la resolució del sistema $JT(\mathbf{s}^{(k)}) \cdot \mathbf{r}^{(k)} = T(\mathbf{s}^{(k)})$ (on $\mathbf{s}^{(k)}$ ara denota $(\xi_0^{(k)}, \xi_1^{(k)}, \ldots, \xi_{m-1}^{(k)})$ — veure el pas (k-c) de la Pàgina 8) per a incrementar l'eficiència del mètode a cada pas.

[10] Observeu que, aquest càlcul és pot fer de manera simultània al càlcul de $x(\tau_{i+1}; \tau_i; \xi_i)$ i $x(b; \tau_{m-1}; \xi_{m-1})$, ampliant la dimensió del sistema com ja s'ha fet anteriorment (veure la Pàgina 15).

Aquí si que ens farà falta