

Memòria Pràctica 3 - Mètodes Numèrics

Arnau del Río

Sunday 31st May, 2026

1 Problema

L'objectiu d'aquesta pràctica és utilitzar mètodes numèrics per calcular la trajectòria de flyby lunar de la missió Artemis II.

Ho farem a través del mètode de Newton per trobar zeros de funcions i el mètode de Nevill d'interpolació numèrica.

A més a més, utilitzarem certes funcions donades per a propagar la trajectòria de la missió.

En particular, utilitzarem el model del problema restringit de tres cossos, en el qual considerem que la missió es propaga en un pla, on dos cossos, la terra i la lluna, amb massa significativa es troben a l'espai i un cos amb massa insignificant es troba en una òrbita generada per l'atracció gravitatòria de la terra i la lluna.

Els mètodes de Newton i de Neville estan implementats en un fitxer de biblioteca anomenat `nwtnev.c`.

Les utilitats que desenvoluparem són les següents:

- `detcsig`: Propaga la trajectòria i troba les posicions i velocitats en cada moment.
- `trobalf`: Resol el problema trobant el moment del flyby i la distància a la lluna.
- `interp`: Interpola un conjunt de punts per a obtenir una funció que aproxima la trajectòria.

2 Implementació

2.1 Mètode de Newton

El mètode de Newton permet aproximar arrels d'equacions de la forma $f(x) = 0$.

Suposem que $f : [a, b] \rightarrow \mathbb{R}$ és de classe \mathcal{C}^2 i que existeix $\alpha \in (a, b)$ amb $f(\alpha) = 0$. Aleshores, el mètode és el següent algorisme iteratiu:

Mètode de Newton

Escollim $x_0 \in [a, b]$ aproximació inicial, $\epsilon > 0$ tolerància i $n_{max} > 0$ el nombre màxim d'iteracions.

$\forall i = 0, 1, \dots, n_{max} - 1$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

si $|x_{i+1} - x_i| < \epsilon$, STOP

2.2 Algoritme de Neville

L'algoritme de Neville ens permet trobar el polinomi interpolador de Lagrange a partir d'un conjunt de punts de suport donats.

Considerem que tenim un conjunt de $n + 1$ punts de suport $\{(x_i, f_i)\}_{i=0}^n$. Aleshores denotem per P_{i_0, i_1, \dots, i_k} el polinomi interpolador de Lagrange resol $P_{i_0, i_1, \dots, i_k}(x_{i_j}) = f_{i_j}$, per $j = 0, 1, \dots, k$.

Algoritme de Neville

Siguin $\{(x_i, f_i)\}_{i=0}^n$ el conjunt de punts de suport. Aleshores se satisfà:

- $P_i(x) = f_i$
- $P_{i_0, i_1, \dots, i_k}(x) = \frac{(x - x_{i_0})P_{i_1, \dots, i_k}(x) - (x - x_{i_k})P_{i_0, \dots, i_{k-1}}(x)}{x_{i_k} - x_{i_0}}$

Podem organitzar els càlculs de l'algoritme de Neville en una taula:

	$k = 0$	$k = 1$	$k = 2$	\dots
x_0	$f_0 = P_0(x)$			
		$P_{0,1}(x)$		
x_1	$f_1 = P_1(x)$		$P_{0,1,2}(x)$	
		$P_{1,2}(x)$		\vdots
x_2	$f_2 = P_2(x)$		\vdots	
\vdots	\vdots	\vdots		

Finalment, el polinomi interpolador de Lagrange que busquem serà $P_{0,1,\dots,n}(x)$.

3 Manual de software

3.1 Funció newton

La funció `newton` té el prototipus següent:

Funció newton

```
int newton (double *x, double tolf, double tolx, int
maxit, void (*fdf)(double,double*,double*,void*), void
*prm, int ixrr);
```

Aquesta funció troba el valor de x que, dins la tolerància donada, troba el zero de una funció. Els arguments són:

- `x`: Punter al valor inicial de x .
- `tolf`: Tolerància per a la imatge de x .
- `tolx`: Tolerància per a la diferència entre valors consecutius de x .
- `maxit`: Nombre màxim d'iteracions.
- `fdf`: Punter a la funció que calcula la funció i la seva derivada.
- `prm`: Punter a paràmetres addicionals a la funció `fdf`.
- `ixrr`: Índex de xarrera, indica si imprimeix valors sobre les iteracions per `stderr`.

3.2 Funció neville

La funció `neville` té el prototipus següent:

Funció `neville`

```
double neville (int n, double xi[], double yi[], double  
x);
```

Aquesta funció calcula l'interpolació de Newton per un conjunt de punts donats. Els arguments són:

- `n`: Hi ha $n + 1$ punts.
- `xi`: Vector de valors de x .
- `yi`: Vector de valors de y .
- `x`: Valor de x per el qual calcular l'interpolació.

3.3 Utilitat detcsig

La utilitat `detcsig` reb unes condicions inicials i propaga la trajectòria, imprimint certs valors a cada pas. No reb cap valor per `stdin`.

```
$ ./detcsig r0 v0 alf0 alff nalf nt pmax
```

on `r0` és el valor inicial de r , `v0` és el valor inicial de v , `alf0` és el valor inicial de α , `alff` és el valor final de α i `nalf` és el nombre de valors de α a considerar. Els valors de `nt` i de `pmax` serveixen per passar-los a `proptraj`.

Exemple d'ús:

```
$ ./detcsig 0.01701377045867686=r0 10.67764174790536=v0  
0.95=alf0 1.25=alff 100=nalf 2=nt .05=pmax  
-----  
0.95 0.07049004991490235 7385.357327340641  
0.953030303030303 0.05298628767590897 7803.10624193974  
...  
1.246969696969697 0.05226682892839966 35245.90328074239  
1.25 0.05497146275142845 35339.34197112153
```

En particular, si podem redirigir aquesta sortida al programa `gnuplot` per obtenir:

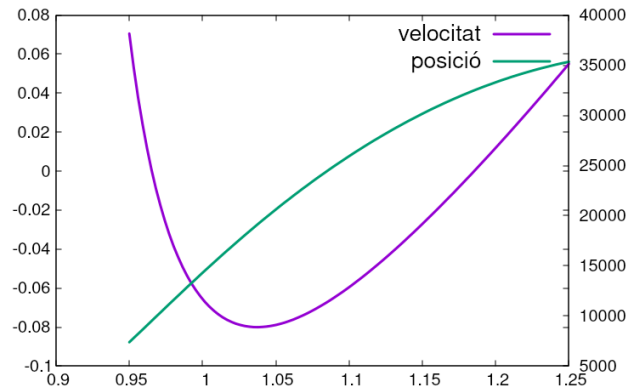


Figure 1: Gràfica de la funció `detcsig`

Observem que en aquest cas, el flyby es produeix en una alçada on la velocitat s'anula, és a dir, en una alçada aproximada de 6047.70 km (ja que el valor al voltant de 30000 no és el que ens interessa).

3.4 Utilitat trobalf

La utilitat `trobalf` reb unes condicions inicials de forma iterada, i per a cada iteració, propaga la trajectòria i calcula el punt allà on la trajectòria passa a prop de la lluna i talla l'eix d'ordenades.

```
$ ./trobalf tolu maxit pmax ixrr
```

on `tolu` és la tolerància, `maxit` el nombre màxim d'iteracions, i `ixrr` el valor de `ixrr`. El valor de `pmax` serveix per passar-lo a `proptraj`.

A més a més, per `stdin` reb quatre valors per a cada iteració: `r0`, `v0`, α_0 , `nt`, on `r0` és la distància inicial, `v0` és la velocitat inicial, α_0 és l'angle inicial, i `nt` és el nombre de talls amb l'eix d'ordenades.

Exemple d'ús (seguint l'exemple de la utilitat anterior):

```
$ echo 0.01701377045867686 10.67764174790536  
0.9651515151515151 2 | ./trobalf 1e-10 10 .05 1  
-----  
newton:  it=0 x=0.9651515151515151  
f(x)=0.0007233922573357029 f'(x)=-3.387472267594433  
newton:  it=0 x=0.9653650644296172  
|dx|=0.000213549278102132  
newton:  it=1 x=0.9653650644296172  
f(x)=2.880323103687877e-06 f'(x)=-3.360536680934217  
newton:  it=1 x=0.9653659215317343  
|dx|=8.571021170578064e-07  
newton:  it=2 x=0.9653659215317343  
f(x)=4.648310961316571e-11 f'(x)=-3.360429056389705  
0.01701377045867686 10.67764174790536 0.9653659215317343  
6046.557984050551
```

Obtenim un resultat coherent que indica que l'angle que busquem és $\alpha = 0.9653659215317343$ per a aconseguir una velocitat horitzontal nul·la al creuar l'eix d'ordenades.

Amb la utilitat donada **shtmn** (shoot the moon) podem obtenir un conjunt de punts, que, després de graficar-los, ens mostren la òrbita que busquem. En el nostre cas, si executem (noteu que hem utilitzat els valors obtinguts en executar **trobalf**):

```
$ echo 0.01701377045867686 10.67764174790536  
0.9653659215317343 3 | ./shtmn .05 trajsex.txt
```

```
$ gnuplot  
gnuplot> plot "trajsex.txt" u 2:3 w l
```

Obtindrem el gràfic que volíem, amb els tres talls a l'eix horitzontal:

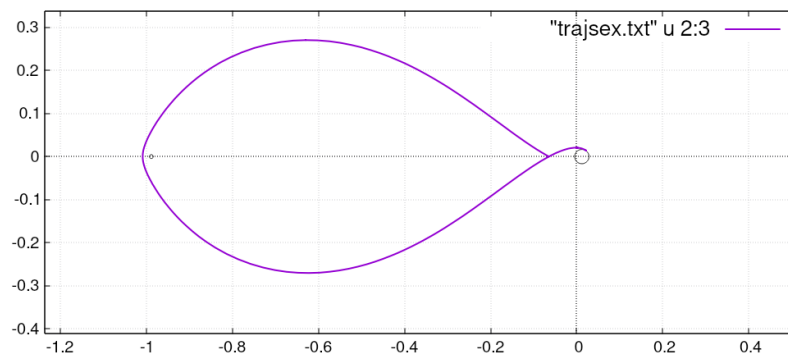


Figure 2: Órbita de la nau espacial amb els valors trobats

3.5 Utilitat interp

La utilitat `interp` calcula el polinomi interpolador a partir d'uns punts de suport i avalua el polinomi en uns altres punts donats.

```
$ ./interp n
```

on n és el nombre de punts de suport. Per altra banda, reb per `stdin` $n + 1$ punts de suport, i a continuació un nombre indeterminat de punts a avaluar.

Per exemple, suposem que tenim els punts de suport $(0, 1)$, $(1, 3)$, $(3, 2)$ i volem avaluar el polinomi en el punt $x = 2$. Aleshores, executem:

```
$ echo 0 1 1 3 3 2 2 | ./interp n
```

```
2 3.3333333333333333
```

Podem comprovar que això és correcte fent la taula de l'algoritme de Neville:

	$k = 0$	$k = 1$	$k = 2$
$x_0 = 0$	$f_0 = P_0(2) = 1$		
		$P_{0,1}(2) = 5$	
$x_1 = 1$	$f_1 = P_1(2) = 3$		$P_{0,1,2}(2) = 10/3$
		$P_{1,2}(2) = 5/2$	
$x_2 = 3$	$f_2 = P_2(2) = 2$		

Per tant el resultat és correcte.

4 Resolució del problema

Ara, ens volem plantejar aconseguir una alçada de persileni de 6545 km.

Per fer-ho, modificarem els valors de v_0 per a obtenir una alçada pròpia a 6545 km, i quan la tinguem, aplicarem interpolació polinòmica amb l'algoritme de Neville utilitzant la utilitat `interp` per a interpolar la alçada que busquem i trobar v_0 .

Provem alguns valors i creem la següent taula:

v_0	h
10.67764174790536	6046.557984050551
10.67754174790536	6201.149491899968
10.67744174790536	6364.005404339322
10.67735174790536	6518.393064334263
10.67734174790536	6536.038703405597
10.67733174790536	6553.786761598189
10.67732174790536	6571.638426123778
10.67724174790536	6718.331514385751
10.67714174790536	6912.181634825405

Aplicuem `interp` a la taula anterior per a interpolar la alçada que busquem i trobar v_0 . Guardem la taula en un fitxer que anomenarem `trobalf.txt` i executem `interp` amb el següent comandament:

```
$ (cat trobalf.txt | awk 'print $2,$1' ; echo 6545) |  
./interp 8  
-----  
6545 10.67733669148103
```

Comprovem aquest resultat executant de nou la utilitat `trobalf` amb el valor interpolat v_0 per veure si obtenim l'alçada que busquem:

```
$ echo 0.01701377045867686 10.67733669148103  
0.9653659215317343 2 | ./trobalf 1e-10 10 .05 0  
-----  
0.01701377045867686 10.67733669148103 0.9731833801332418  
6545.000000012091
```

Efectivament, hem comprovat que, amb la velocitat inicial $v_0 = 10.67733669148103$, i amb l'angle inicial $\alpha_0 = 0.9653659215317343$, l'alçada que obtenim és 6545 km.