Departament de Matemàtica Aplicada 4 — EPSEVG
Universitat Politècnica de Catalunya

EQUACIONS DIFERENCIALS - ENGINYERIA INDUSTRIAL - EPSEVG

# Manual de pràctiques amb MATLAB

Copyright 2011 Carles Batlle (carles.batlle@upc.edu)
This work is licensed under a Creative Commons Attribution-Share Alike 3.0 License. A copy of the license can be found at http://creativecommons.org/licenses/by-sa/3.0/

# Índex

**I Pràctica 1 — Introducció a MATLAB 5**

**1 Introducció 5**

**2 L'entorn de treball 5**

**3 MATLAB com a calculadora avançada 8**

**4 Vectors i matrius 10**

**5 *Scripts* i funcions 13**

**6 Exercicis proposats 16**

**II Pràctica 2 — Solució numèrica d'EDO 19**

**7 Integradors numèrics 19**

**8 Com utilitzar `ode45` 20**

**9 Exercicis proposats 25**

**III Part 3 — Anàlisi de Fourier (opcional) 29**

**10 Referències a funcions 29**

**11 Exemple: polinomi de Taylor d'una funció 31**

**12 Exemple: suma de Riemann d'una funció 33**

**13 Coeficients de Fourier i sumes parcials de la sèrie de Fourier d'una funció 35**

**14 Algunes eines més per a funcions periòdiques 38**

**15 Representació de l'espectre d'un senyal periòdic 44**

**16 La transformada discreta de Fourier 45**

**17 Exercicis proposats 49**

# Part I
# Pràctica 1 — Introducció a MATLAB

## 1 Introducció

MATLAB®, de la companyia *The MathWorks*, és un entorn de càlcul numèric i alhora un llenguatge de programació. El nom MATLAB prové de *matrix laboratory*, és a dir, laboratori de matrius, ja que el programari ho representa tot mitjançant matrius.

MATLAB proporciona un llenguatge d'alt nivell i un entorn interactiu que permeten solucionar problemes tècnics de manera molt més ràpida que amb llenguatges tradicionals com C, C++ o Fortran, i a més incorpora algorismes numèrics de màxima qualitat i nombroses llibreries específiques que fan que sigui extraordinàriament útil en tots els camps de l'enginyeria i, de fet, en totes les ciències que requereixin computació numèrica.

MATLAB incorpora també l'entorn de programació gràfica SIMULINK, amb nombroses funcions i llibreries per al modelatge de sistemes i de processos.

MATLAB és un programari comercial propietari. Existeixen alternatives de software lliure com ara GNU Octave, Scilab i Sage (basat en Python), però no tenen l'abast ni, sobretot, la implantació en la comunitat d'enginyers i la indústria que té MATLAB.

## 2 L'entorn de treball

La Figura 1 mostra l'entorn de treball de MATLAB.

Hom pot distingir 4 espais principals, que es poden re-configurar segons convingui:

- la finestra de comandes (*Command Window*), on hom escriu les instruccions que s'executen de forma interactiva.
- l'historial de comandes (*Command History*), on es guarden les ordres executades a la sessió actual i a les sessions anteriors.
- l'espai de variables (*Workspace*), on apareixen les variables definides en la sessió, així com les seves característiques.
- el directori de treball (*Current Folder*), amb la llista dels fitxers que conté. Aquí és on es guarden per defecte els fitxers creats per MATLAB, i d'on MATLAB llegeix els fitxers creats per l'usuari.

Si escriviu `a=4` a la finestra de comandes

```
» a=4
```

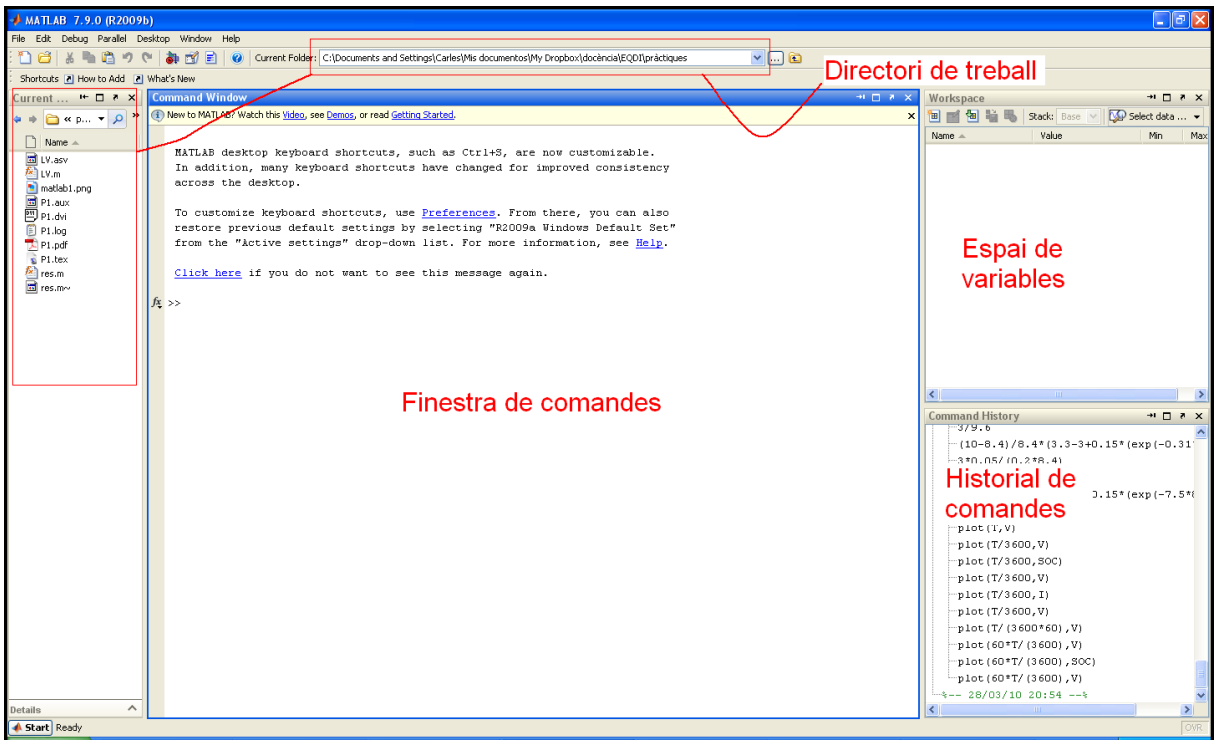



Figura 1: L'entorn de treball de MATLAB

veureu que a l'historial de comandes hi apareix aquesta instrucció, i que a l'espai de variables hi ha ara un objecte. Proveu les comandes següents, i observeu els canvis:

```
» a^2
» a+7
» b=sqrt(a)
» c=b+a/pi
```

En particular, hi ha una variable `ans` que conté el resultat de la darrera operació que heu fet. Podeu suprimir la visualització d'un resultat a la finestra de comandes si acabeu la instrucció amb `;`

```
» a=23/3.24;
```

Si voleu veure el resultat però no que surti `ans=`, podeu emprar la funció `disp()`

```
» disp(23/3.24)
```

i llavors, a més, `ans` no canvia de valor. Es poden recuperar i editar comandes amb la tecla de fletxa amunt, i també podeu anar a la finestra d'historial i executar qualsevol comanda prèvia (prement F9 o amb el botó dret del ratolí).

Podeu donar qualsevol nom a una variable, sempre que no correspongui a una paraula reservada, tenint en compte, però, que

- el primer caràcter ha de ser una lletra,
- els caràcters següents poden ser lletres, dígits o _,
- minúscules i majúscules es consideren diferents.

A més de variables de caràcter numèric, podeu emprar també cadenes de caràcters:

```
» la_de_sempre='Hola mon!'
```

La comanda `save` permet salvar variables en un fitxer:

```
» save elmeufitxer a b
```

Això salva les variables `a` i `b` en el fitxer `elmeufitxer.mat` del directori de treball (fixeu-vos que el fitxer apareix a la finestra del directori de treball). Ara podeu esborrar les variables `a` i `b` de l'espai de variables (i de la sessió) amb

```
» clear a b
```

Si feu

```
» a
```

obtindreu com a resposta `??? Undefined function or variable 'a'.` Podeu recuperar les variables, amb els valors salvats, amb

```
» load elmeufitxer
```

Si no escriviu cap variable, la instrucció `save` ho salva tot en el fitxer que indiqueu; llavors ho podeu esborrar tot i recuperar-ho més endavant:

```
» save totes
```

```
» clear all
```

```
» load totes
```

Podeu obtenir ajuda escrivint `help`. Això retorna una enorme llista amb enllaços a totes les funcions i paraules clau de MATLAB. Si sabeu el nom del que cerqueu, podeu escriure, per exemple

```
» help sin
```

La instrucció

```
» doc sin
```

dóna una ajuda mes amigable, amb exemples.

# 3 MATLAB com a calculadora avançada

A més de les comandes elementals que ja han aparegut, MATLAB es pot emprar com una calculadora avançada. MATLAB utilitza `i` per a la unitat imaginària, però podeu emprar també `j`. Proveu:

```
» exp(i*pi)
» exp(j*pi/2)
» c=1+2*j
» c/(1+i)
» c/(1+i)^5
```

Observeu que MATLAB sempre retorna el resultat en forma cartesiana. Podeu obtenir els paràmetres de la representació polar amb `abs` i `angle`

```
» rho=abs(1+2*i)
» theta=angle(1+2*i)
» rho*exp(i*theta)
```

Es pot obtenir la part real i la part imaginària d'un complex, i calcular-ne el complex conjugat:

```
» ar=real((1+2*i)/(3+4*i))
» ai=imag((1+2*i)/(3+4*i))
» z=(1+2*i)/(3+4*i)
» bz=conj(z)
» z+bz-2*ar
```

MATLAB fa els càlculs interns amb 15 dígits de precisió, però per defecte mostra els resultats amb 5 dígits. Això es pot canviar amb la instrucció `format long e`:

```
» 2/pi
» format long e
» 2/pi
```

Es pot tornar als 5 dígits amb `format short e`.

MATLAB pot calcular límits si es declaren prèviament amb `syms` les variables i paràmetres implicats:

```
» syms x a h
» limit(sin(a*x)/x,x,0)
» limit(sin(a*x)/x,a,0)
» limit((2*x^2+x)/(3*x^2+x+1),x,Inf)
» limit(1/x,x,0,'right')
» limit((sin(x+h)-sin(x))/h,h,0)
```

De la mateixa manera, es poden calcular derivades (no cal tornar a declarar les variables si encara existeixen d'abans)

```
» syms x a
» diff(sin(a*x)/x,x)
» diff(sin(a*x)/x,a)
```

i integrals

```
» syms x a
» int(x^2*sin(a*x),x)
» int(x*log(1+x),x,0,1)
» int(x/(x^4+1),x,0,Inf)
```

Per representar una funció cal definir primer els punts on es calcula, i després dir que representi els punts i els valors:

```
» x=linspace(-2,2,1000);
» plot(x,sin(x))
```

Això calcula 1000 valors equidistants (això és el que fa `linspace`; hi ha altres opcions) entre $-2$ i 2 i ho posa dins el vector `x` (escrivim `;` al final per a que no ensenyi els 1000 valors). Després calcula $\sin x$ en aquests 1000 punts i ho representa. Si volem representar funcions polinòmiques o racionals, com ara $f(x) = \frac{1+x}{x^2+1}$, cal utilitzar una notació especial si volem que les operacions amb el vector `x` es facin component a component (tal com cal):

```
» x=linspace(-2,2,1000);
» plot(x,(1+x)./(x.^2+1))
```

El punt davant de `^` indica que els elements de `x` s'han d'elevar cada un d'ells al quadrat. A cada component del resultat se li suma 1 (això ja ho entén directament), i cal posar `./` per dividir cada component del vector `1+x` per la component corresponent del resultat.

# 4 Vectors i matrius

Fins ara, exceptuant l'aparició de `linspace`, hem treballat sols amb *escalars*, és a dir, elements de $\mathbb{R}$ o de $\mathbb{C}$. Hom pot definir vectors o matrius amb elements reals o complexos, i fer tota mena d'operacions. MATLAB considera els vectors com a casos particulars de matrius. Les matrius es donen fila a fila, amb els elements d'una fila separats per espais o per comes; per canviar de fila cal posar un punt i coma. Per exemple, la matriu

$$A = \begin{pmatrix} 1 & 2 & 4 \\ -1 & 0 & 2 \\ 3 & 1 & 2 \end{pmatrix}$$

és

```
» A=[1,2,4;-1 0 2;3,1,2]
```

on hem posat comes per separar els elements de la primera i tercera files i espais pels de la segona. El vector (columna)

$$b = \begin{pmatrix} 4 \\ 2 \\ -2 \end{pmatrix}$$

és

```
» b=[4;2;-2]
```

i el vector fila

$$c = \begin{pmatrix} 0 & -1 & 5 \end{pmatrix}$$

es pot obtenir com

```
» c=[0 -1 5]
```

L'espai de variables mostra els objectes que hem definit tal com es veu a la Figura 2. Aquesta finestra té menús amb diverses opcions per manipular i mostrar les dades. Si, per defecte, no teniu l'opció `Size` activada, cliqueu amb el botó dret sobre alguna de les pestanyes que sí que surten i activeu-la.

Les operacions de l'àlgebra lineal amb matrius i vectors són immediates:

```
» det(A)
```

```
» inv(A)
```

```
» A*b
```

```
» B=ones(3)
```
(matriu $3 \times 3$ tot amb 1)

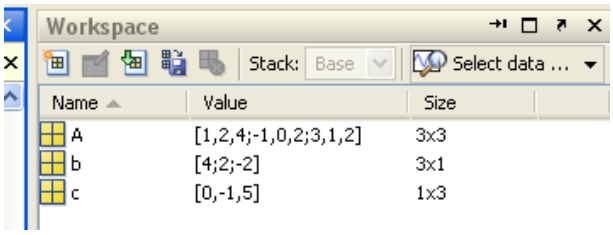



Figura 2: L'espai de variables amb les matrius i vectors introduïts, amb les seves dimensions (files $\times$ columnes).

» `C=A+B`

» `C'` (la matriu transposada de $C$; de fet calcula a més el complex conjugat dels elements)

» `Z=zeros(3)` (matriu $3 \times 3$ tot amb $0$)

» `y=Z*b`

» `D=eye(3)` (matriu identitat $3 \times 3$)

» `A*inv(A)-D`

(la darrera operació dóna la matriu zero amb la precisió numèrica amb que es treballa).

L'operació de transposar amb `'` en realitat transposa i a més calcula el complex conjugat dels elements (això s'anomena calcular la matriu hermítica de la donada). Això és irrellevant si la matriu té sols elements reals, però és important si hi ha elements complexos:

» `A=[1-j, 1; j, 3+j]`

» `A'`

Si sols voleu transposar, sense calcular els complexos conjugats, s'ha de fer amb `.'`:

» `A.'`

Es poden seleccionar elements individuals d'una matriu, files o columnes senceres, o grups de files o columnes. Com exemple, primer creem una matriu aleatòria amb 5 files i 4 columnes, i després en seleccionem diverses parts:

» `H=rand(5,4)`

» `H(4,3)` (l'element de la fila 4 columna 3)

» `H(:,3)` (la tercera columna)

» `H(2,:)` (la segona fila)

» `H(1:2,3)` (les files 1 i 2 de la tercera columna)

» `H(1:2,3:4)` (les files 1 i 2 de les columnes 3 i 4)

» `H([1 4 5],:)` (les files 1, 4 i 5)

» `H([1 4 5],[2 4])` (les files 1, 4 i 5 de les columnes 2 i 4)

És possible concatenar vectors o matrius vertical o horitzontalment amb la funció `cat`:

» `x=[1 4 6]`

» `y=[-2 0 8]`

» `cat(2,x,y)` (el `2` indica que es concatena al llarg de columnes)

» `cat(1,x,y)` (es concatena al llarg de files)

» `cat(2,x,[4])` (afegeix l'element 4 al vector fila `x`)

S'ha de tenir en compte, però, que `cat` no modifica el vector o matriu que li passeu. Si és això el que voleu, cal fer l'assignació explícitament:

» `x=[1 4 6]`

» `y=[-2 0 8]`

» `cat(2,x,y)`

» `x`

» `x=cat(2,x,y)`

» `x`

La solució de $Ax = b$, si existeix i és única, s'obté amb `A\b`:

» `x=A\b`

» `A*x`

Podem calcular la base del nucli de l'aplicació lineal associada a una matriu (quadrada o no) amb `null`:

» `E=[ 1 2 0 5; -2 -1 4 5]`

» `base_nucli=null(E)`

Per seleccionar els vectors individuals de la base, cal obtenir les columnes per separat:

```
» v1=base_nucli(:,1)

» v2=base_nucli(:,2)

» E*v1, E*v2
```

De nou, els resultats de la darrera comanda són equivalents a zero, tal com ha de ser. Per poder calcular una base de la imatge, cal declarar primer la matriu com a simbòlica amb `sym` i després emprar `colspace`:

```
» F=sym([ 1  2; -3 4; 0 1])

» colspace(F)
```

# 5 *Scripts* i funcions

Un *script* és un conjunt d'instruccions de MATLAB que es guarden en fitxers amb extensió `.m`, o *M-files*, i que s'editen amb l'editor de MATLAB. Per crear-ne un, amb nom `moneda.m`, podeu escriure

```
» edit moneda
```

o invocar l'editor des del menú (Figura 3).

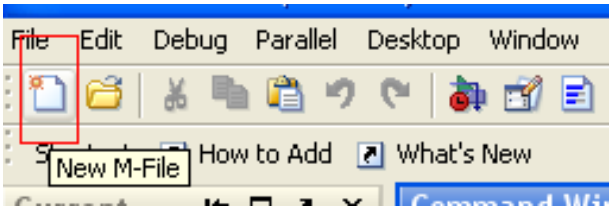



Figura 3: L'editor de fitxers M des del menú de MATLAB.

En aquest fitxer hi posarem codi que calcula un nombre aleatori distribuït uniformement entre 0 i 1, mitjançant la instrucció `rand`, i que escriu `CARA` o `CREU` segons doni més petit o més gran o igual que 0.5. El contingut del fitxer, que heu d'escriure, apareix a la Figura 4.

El codi del fitxer s'explica ell mateix. Cal notar el següent:

- Podeu executar el codi des de l'editor o des de la finestra de comandes, cridant el fitxer pel seu nom amb

  ```
  » moneda
  ```

  Cal, però, que el fitxer estigui en el directori de treball.

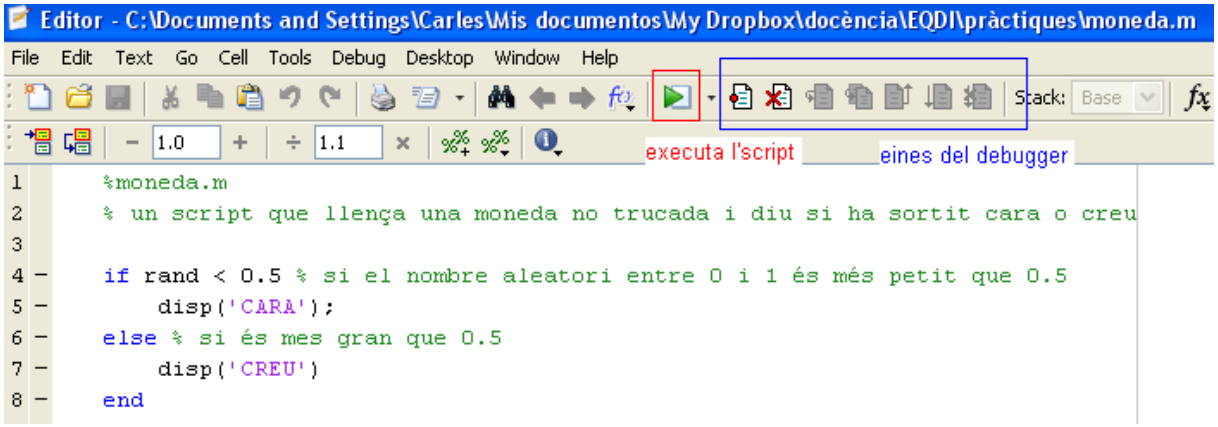



Figura 4: El fitxer `moneda.m`.

- El codi es va indentant automàticament segons aneu escrivint.
- Els comentaris es precedeixen amb %. Les primeres línies de comentaris abans del codi són l'ajuda del codi per a l'usuari. Proveu

  ```
  » help moneda
  ```

- L'editor té eines de depuració, que es poden cridar des del menú.

Els *scripts* utilitzen el llenguatge de programació de MATLAB. Com sempre, podeu cercar ajuda sobre les construccions del llenguatge amb `help` o `doc`:

```
» help if
```

```
» doc while
```

```
» doc for
```

Els *scripts* són útils si voleu executar un conjunt complex d'instruccions diverses vegades, o si voleu corregir errors en el conjunt d'instruccions. Per exemple, el *script* `matriu.m` que apareix a la Figura 5 construeix una matriu quadrada $n \times n$ que a la fila $i$, columna $j$ té per element $i+j+3$, i després calcula el seu determinant i el seu polinomi característic. Com a variables d'iteració emprem `k,l` enlloc de `i,j` per evitar un possible conflicte amb els noms de la unitat imaginària. Canviant l'assignació de $n$ podeu executar el conjunt d'instruccions per al valor que vulgueu sense haver de canviar res més.

Les sentencies que calculen quelcom acaben amb `;` per evitar que mostrin el resultat a l'executar-se.

Quan executeu el *script* veureu que a l'espai de treball apareixen els valors de les variables calculades o definides (`n,A,d,p`), a més dels valors finals de les variables `k,l` d'iteració dels llaços. Podeu veure el valor de la variable que interessi cridant-la explícitament:

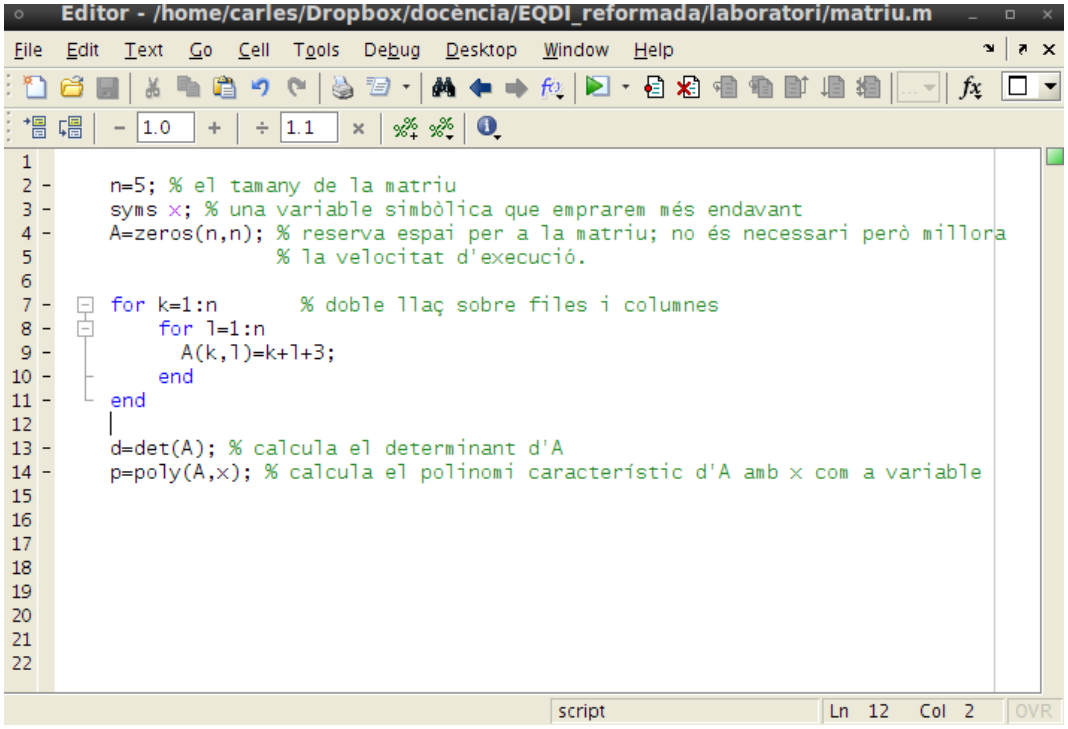



Figura 5: El fitxer `matriu.m`.

```
» matriu

» p
```

Les funcions són com els *scripts*, però amb la diferència que han de contenir necessàriament la declaració de la funció, tal com mostra la Figura 6.

Feu

```
» help estats

» x=[1 2 -4 5 sqrt(2) 0 -15 7]

» [max,min,mitj]=estats(x)
```

La Figura 7 conté el codi d'una funció que calcula les $n$ arrels $n$-èsimes d'un nombre complex $z$, i a més les representa, juntament amb $z$, en el pla complex.

Si voleu calcular, per exemple, les arrels cinquenes de $z = 1 + j$ heu de fer

```
» r=arresl(1+i,5)
```

i podeu després comprovar el resultat:

```
» for k=1:5 display(r(k)^5); end
```

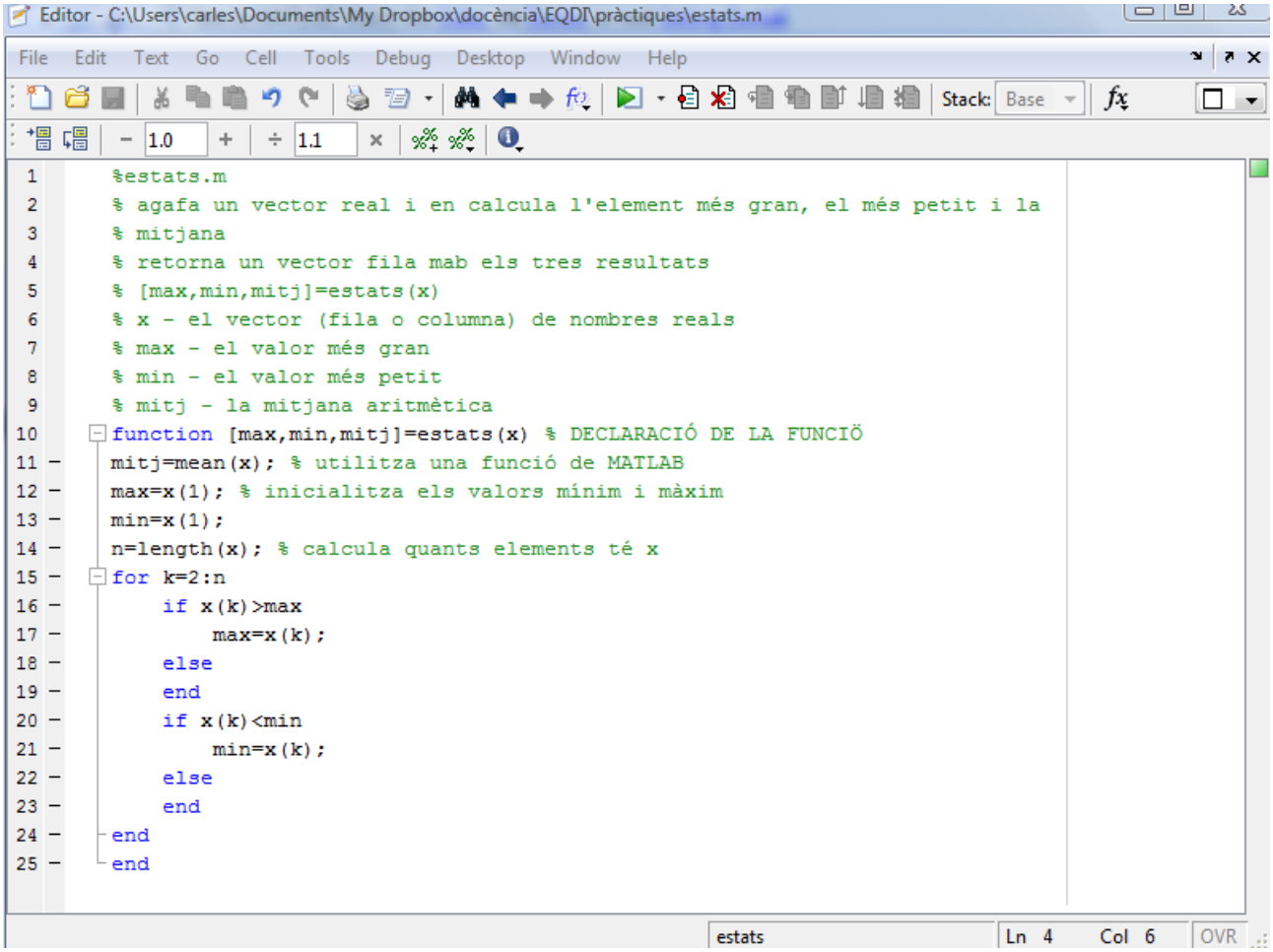



Figura 6: El fitxer `estats.m`.

# 6 Exercicis proposats

1. Calcula els següents límits amb MATLAB. Comprova el resultat fent el càlcul a ma i representa la funció en un interval que corrobori els càlculs.

   (a) $\lim_{x\to 0} \frac{x \sin x}{1-\cos 7x}$.

   (b) $\lim_{x\to 2} \frac{x-2}{x^2-4}$.

   (c) $\lim_{x\to +\infty} \frac{x^2+x+1}{3x^2-x-2}$.

   (d) $\lim_{x\to -\infty} \sin \arctan x$.

   (e) $\lim_{x\to 2^+} \frac{2}{2-x}$.

   (f) $\lim_{x\to 1-} \frac{\log x}{\sqrt{1-x}}$.

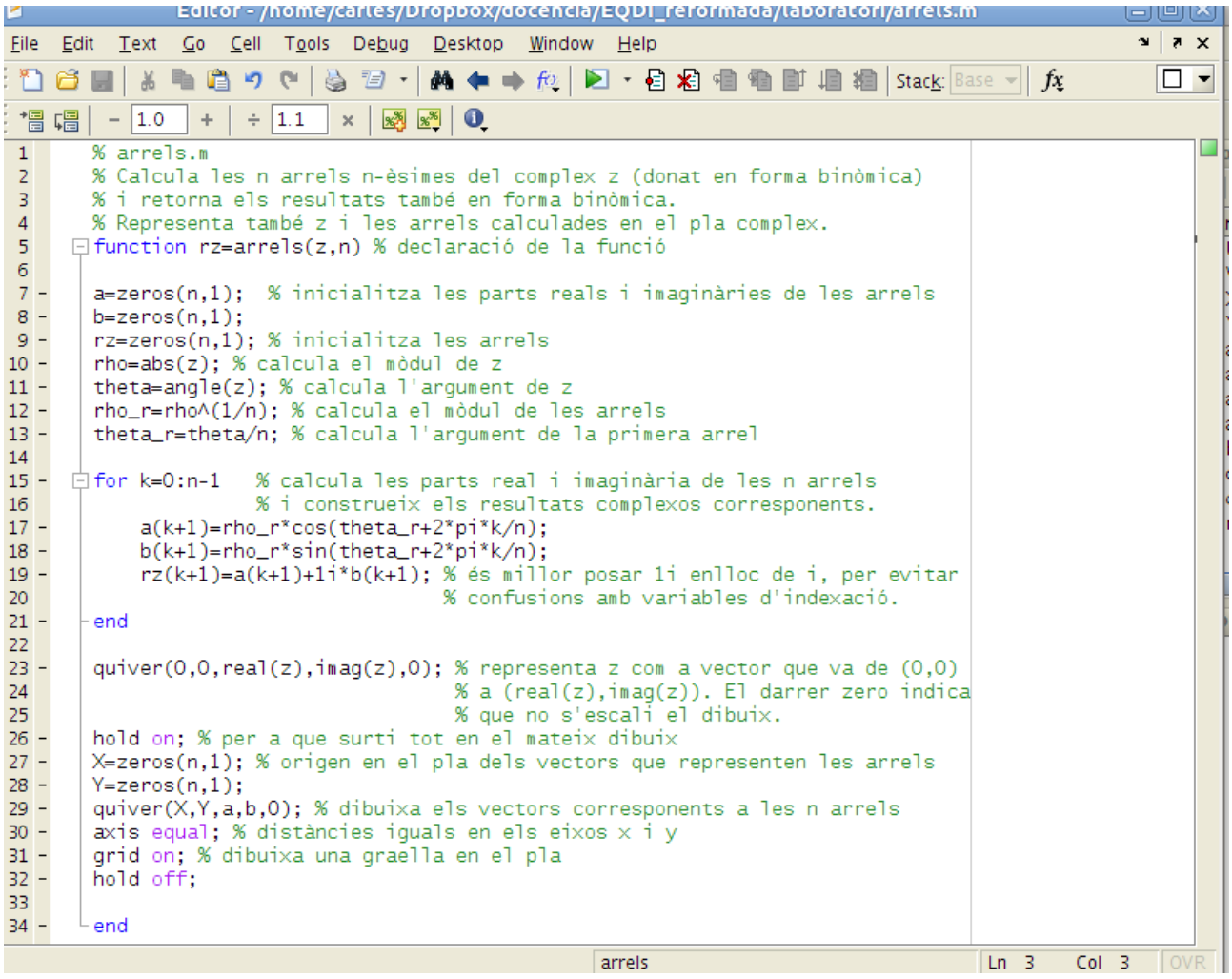



Figura 7: El fitxer `arrels.m`.

2. Expressa els següents nombres complexos en forma cartesiana. Calcula el seu mòdul i argument, i la seva part real i imaginària.

   (a) $\frac{(0.2+5j)^5}{1+3.72j}$.

   (b) $e^{-1-j}$.

   (c) $(1+3j)(1+4j)^8$.

   (d) $\sin(1-j)$.

   (e) $\frac{e^{j(1-j)}-e^{-j(1-j)}}{2j}$.

   (f) $\bar{a}+a$, on $a=(1-j)/(2+j)$.

3. Siguin
$$A = \begin{pmatrix} -1 & 3 & 4 & 1 \\ -2 & 0 & 2 & 5 \\ 0 & 3 & 3 & 6 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 3 \\ -1 \\ 4 \end{pmatrix}.$$
   (a) Calcula el determinant d'$A$, $A^{-1}$, la matriu trasposta d'$A$ i $A^6$.
   (b) Troba la solució de $Ax = b$.
   (c) Construeix la matriu que s'obté afegint a $A$ la columna formada pels elements de $b$, i la que s'obté afegint la fila formada pels elements de $b$.
   (d) Construeix la matriu que s'obté amb les dues primeres columnes d'$A$ i el vector columna $b$.
   (e) Construeix la matriu que s'obté amb la primera i quarta files d'$A$ i el vector fila $b$.

4. Sigui la matriu
$$A = \begin{pmatrix} -1 & 3 & 4 \\ -2 & 0 & 2 \\ 0 & 3 & 3 \end{pmatrix}.$$
   (a) Calcula el seu determinant.
   (b) Calcula el nucli d'$A$, i comprova el resultat.
   (c) Construeix una matriu $B$ amb les dues primeres columnes d'$A$. Calcula el seu nucli.
   (d) Sigui $v$ el vector que s'obté de sumar la primera columna d'$A$ amb 3 vegades la segona columna d'$A$. Soluciona el sistema $Bx = v$. Com és que això té solució i és única?

5. Escriu una funció que prengui com argument dos valors reals i retorni la seva suma i el seu producte.

6. Escriu una funció que prengui com argument un nombre complex i en retorni el seu mòdul, el seu argument, la seva part real i la seva part imaginària.

7. Escriu una funció que prengui com argument un vector de valors reals o complexos i retorni un vector amb els valors absoluts o mòduls ordenats de més petit a més gran (pots emprar la funció `sort` de MATLAB).

8. Escriu una funció que, donats dos vectors de nombres reals, en retorni la seva suma, el seu producte escalar i un vector amb els productes de les components del primer vector pel sinus de les components del segon vector.

9. Escriu una funció que prengui com argument un enter positiu $N$ i retorni una matriu $N \times N$ que a la posició $(k, l)$ tingui $k + l$.

10. Escriu una funció que prengui com argument una matriu i en retorni una altra amb els vectors columna de la primera que siguin linealment independents.

11. Escriu una funció que, a partir d'un vector de nombres reals o complexos, calculi aquell (o aquells si n'hi ha més d'un) que tingui el mòdul més proper a 1. La funció ha de retornar tots els elements que compleixin la condició, i el valor (comú, si n'hi ha més d'un) del mòdul. Documenta la funció i comprova-la amb diversos vectors, però en particular amb

    ```
    » x=[1-j, 1+j, 2, -3, 2+j, 0, sqrt(2)]
    ```

12. Escriu una funció que, a partir d'una matriu qualsevol, en retorni una altra igual a la primera, però amb tots els valors 1 i 2, si n'hi ha, canviats per $-1$ i $-2$, respectivament.

13. Escriu una funció que, donada una matriu $A$ i un valor real $a$, retorni una matriu com $A$ però sense les columnes la suma dels valors de les quals sigui menor que $a$.

14. Escriu una funció que prengui com arguments dues matrius $A$ i $B$ de les mateixes dimensions, i en retorni una altra amb els elements d'$A$, però canviant per zero aquells que coincideixen amb els de $B$.

# Part II
# Pràctica 2 — Solució numèrica d'EDO

## 7 Integradors numèrics

L'elecció del mètode d'integració d'una equació diferencial, és a dir, com es converteix l'equació diferencial en una equació en diferències, pot tenir resultats dramàtics respecte a la qualitat del resultat final. Matlab proporciona fins a 7 integradors diferents, i cadascun d'ells permet diverses opcions. Amb aquests mètodes, Matlab pot solucionar sistemes d'equacions de la forma

$$M(t, y)y' = f(t, y)$$

on $y \in \mathbb{R}^n$ i $M$ és una matriu $n \times n$, anomenada la *matriu de massa* del sistema. Si $M$ no té el rang màxim, el sistema és de fet una DAE (differential-algebraic equation), i llavors s'ha d'anar amb cura perquè no qualsevol condició inicial $y(0)$ és acceptable.

Seguint el consell de Matlab, utilitzarem `ode45`. Aquest mètode és el que s'anomena un Runge-Kutta(4,5) explícit, i és un mètode d'un sol pas: per calcular $y(t_n)$ sols necessita $y(t_{n-1})$. A diferència del Runge-Kutta habitual, és un mètode de pas variable: escull el pas que ha de fer en funció de les toleràncies que li marca l'usuari.

# 8 Com utilitzar `ode45`

El primer que cal fer per usar `ode45` (o qualsevol altre) és definir una funció que ens descrigui el sistema. Això es fa mitjançant un M-file. Sigui per exemple el sistema de dimensió 3, que representa l'evolució de les tres components de la velocitat angular d'un sòlid rígid a l'espai sense cap mena de força externa,

$$\begin{aligned} \dot{w}_1 &= w_2\, w_3 \\ \dot{w}_2 &= -w_1\, w_3 \\ \dot{w}_3 &= -0.51\, w_1\, w_2 \end{aligned}$$

amb les condicions inicials $w_1(0) = 2$, $w_2(0) = -1$, $w_3(0) = 0$. Escrivim llavors el fitxer M que apareix a la Figura 8 (els comentaris són opcionals, si no teniu ganes d'escriure), i ho guardem com a `rigid.m` en el directori de treball.

```
1     % rigid(t,w)
2     % Aquesta funció representa el sistema d'equacions per a les tres components de
3     % la velocitat angular d'un sòlid rígid, per a certs valors dels moments
4     % principals d'inèrcia. Es tracta d'un sistema de 3 EDO de primer ordre
5     % no lineals i autònom.
6     % w(1), w(2), w(3) (les components de la velocitat angular) són les variables
7     % dependents, i t (el temps) és la variable independent.
8
9     function dw=rigid(t,w) % DECLARACIÓ DE LA FUNCIÓ
10    dw=zeros(3,1); % inicialitzem les derivades
11    dw(1) = w(2)*w(3); % donem el sistema d'equacions
12    dw(2) = -w(1)*w(3);
13    dw(3) = -0.51*w(1)*w(2);
14    end
```

Figura 8: El fitxer `rigid.m`. El cuc vermell sota la `t` és degut a que l'argument `t` no s'utilitza en la funció, ja que el nostre sistema no depèn explícitament del temps. Això no és un error, però MATLAB avisa sempre que un argument no es fa servir.

En qualsevol cas, el codi mínim que s'ha de posar és

```
function dw=rigid(t,w)
    dw=zeros(3,1);
    dw(1)=w(2)*w(3);
    dw(2)=-w(1)*w(3);
    dw(3)=-0.51*w(1)*w(2);
end
```

La funció `rigid` retorna els valors de $\dot{w}_1$, $\dot{w}_2$ i $\dot{w}_3$ segons el que valguin $t$, $w_1$, $w_2$ i $w_3$ (de fet és independent de $t$). Per exemple

```
» rigid(2,[1 3 4])

» rigid(14,[1 3 4])
```

retornen totes dues $12.0000$, $-4.0000$, $-1.5300$, però la funció no està pensada per ser cridada directament pel l'usuari, sinó per les funcions d'integració numèrica, tal com veurem.

Ara podem ja cridar el mètode numèric d'integració, i és aquest el moment de dir-li quines són les condicions inicials i l'interval d'integració:

```
» [T,W]=ode45(@rigid,[0,12],[2,-1,0]);
```

Els arguments d'`ode45` són

- `@rigid`, crida la funció on està definit el sistema d'EDO.
- `[0,12]`, és l'interval d'integració, entre $t = 0$, on tenim les condicions inicials, i $t = 12$.
- `[2,-1,0]`, és el vector de condicions inicials, $w_1(0) = 2$, $w_2(0) = -1$, $w_3(0) = 0$.

Els tres primers arguments (el nom de la funció amb `@` al davant, l'interval d'integració i les condicions inicials) són obligatoris; si no hi poseu l'argument `options` el mètode agafa valors per defecte.

La funció `ode45` es pot cridar amb més arguments, però aquests són els fonamentals. La funció `ode45` retorna els resultats de la integració i els posa en el vector `T` i la matriu `W`:

- `T` és el vector de temps on s'ha calculat els valors de la solució. Podeu observar a la finestra de variables que és un vector amb 157 files i una columna. Això vol dir que, amb el sistema definit per `rigid.m` i les opcions d'error d'`options`, `ode45` ha necessitat 157 passos de temps entre $t = 0$ i $t = 12$.
- `W` és una matriu amb 157 files i 3 columnes. Cada columna conté els valors d'una de les variables $w_i$, $i = 1, 2, 3$, per als 157 instants de temps.

Podem representar les tres components de la solució en funció del temps amb la instrucció

```
» plot(T,W(:,1),'b',T,W(:,2),'g',T,W(:,3),'r')
```

Això dibuixa $w_1(t)$ en blau (el color per defecte), $w_2(t)$ en verd i $w_3(t)$ en vermell. Hauria de sortir quelcom semblant al resultat de la Figura 9. Fixem-nos que les corbes comencen en les condicions inicials correctes i que la solució sembla ser periòdica.

Podeu dibuixar les tres components de la solució en finestres diferents amb la successió d'instruccions

```
» plot(T,W(:,1))

» figure
```

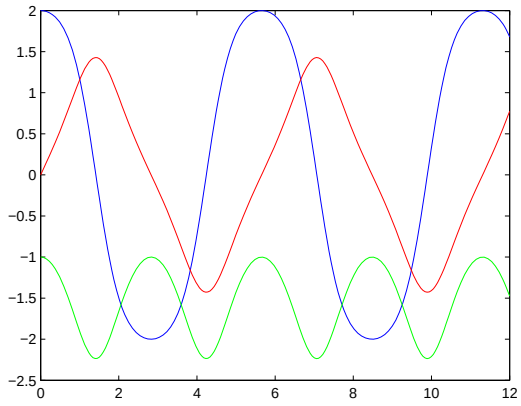



Figura 9: Solució del sistema donat per `rigid.m`.

```
» plot(T,W(:,2))

» figure

» plot(T,W(:,3))
```

Cada vegada que crideu `figure` es crea una finestra on es representa el següent `plot`. També podeu representar les tres components en una mateixa finestra, però successivament en lloc de totes alhora, si utilitzeu la instrucció `hold on`, que manté en la finestra el que ja està representat:

```
» close all

» plot(T,W(:,1))

» hold on

» plot(T,W(:,2),'g')

» plot(T,W(:,3),'r')

» hold off
```

Primer cridem `close all` per tancar totes les finestres (això no és necessari), i al final cridem `hold off` per tornar al comportament normal. Si voleu tancar la finestra $i-$èsima de les que teniu obertes, podeu fer `close(i)`.

També és possible representar una variable dependent en termes d'una altra, en lloc de les variables en funció del temps. Per exemple

```
» plot(W(:,1),W(:,3))
```

representa $w_3$ en funció de $w_1$.

Ja hem dit que els tres primers arguments d'`ode45` són obligatoris, però n'hi ha d'opcionals. És possible, per exemple, especificar el grau de precisió amb que volem la solució. Per fer-ho, a la finestra de comandes de Matlab definim el conjunt d'opcions de control de l'error que passarem a l'integrador:

```
» options=odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
```

Això estableix un error relatiu de $10^{-4}$ per a totes les components de la solució i uns errors absoluts $10^{-4}$, $10^{-4}$ i $10^{-5}$, respectivament, per a cadascuna de les components. Si tots els errors absoluts que demanem fossin iguals, per exemple $10^-4$, es pot escriure també

```
» options=odeset('RelTol',1e-4,'AbsTol',1e-4);
```

A la finestra de variables hi apareix l'objecte `options`, que és un tipus de dades amb estructura. Per cridar l'integrador amb aquestes opcions fem

```
» [T,W]=ode45(@rigid,[0,12],[2,-1,0],options);
```

i el tractament posterior és el mateix.

Com a segon exemple, considerem

$$\ddot{x} = -\arctan\frac{x}{10} - \dot{x} + \sin t, \quad x(0) = 20,\ \dot{x}(0) = 0.$$

Per a $|x| \ll 10$, $\arctan\frac{x}{10} \approx \frac{x}{10}$ i tenim un oscil·lador de massa $m = 1$, amb fregament proporcional a la velocitat amb $\gamma = 1$ i amb una molla de constant $k = \frac{1}{10}$, i sotmès a una força $F(t) = \sin t$.

Per poder integrar numèricament aquesta EDO cal, en primer lloc, escriure-la com un sistema de dues EDO de primer ordre. Si definim $y_1 = x$ i $y_2 = \dot{x}$, queda

$$\begin{aligned} \dot{y}_1 &= y_2, \\ \dot{y}_2 &= -\arctan\frac{y_1}{10} - y_2 + \sin t, \end{aligned}$$

amb condicions inicials $y_1(0) = 20$, $y_2(0) = 0$.

La part essencial del corresponent fitxer M, que anomenem `oscil.m`, és

```
function dy=oscil(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-atan(y(1)/10)-y(2)+sin(t);
end
```

Ara podem executar les instruccions ja conegudes:

```
» [T,Y]=ode45(@oscil,[0,100],[20,0]);
```

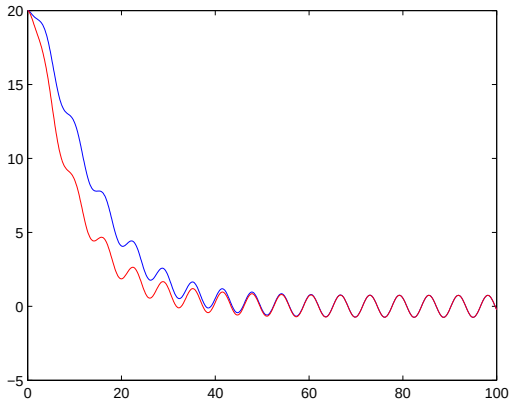



Figura 10: Solucions $x(t)$ del sistema amb $\arctan(x/10)$ (blau) i amb $x/10$ (vermell).

```
» plot(T,Y(:,1))
```

```
» plot(Y(:,1),Y(:,2))
```

La darrera instrucció mostra la velocitat en funció de la posició. Després del transitori, la solució va a parar a una òrbita tancada periòdica. De fet, es pot veure que, en règim permanent, els valors de $x$ són molt més petits que 10 i per tant no hi hauria d'haver molta diferència entre el sistema amb $\arctan(x/10)$ i el sistema amb $x/10$. Si fem el canvi al fitxer M (i anomenem el fitxer `oscil1.m`)

```
function dy=oscil1(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-y(1)/10-y(2)+sin(t);
end
```

i fem de nou la integració numèrica (guardant els resultats en `[T1,Y1]` per no esborrar el resultat de la simulació amb l'arctan)

```
» [T1,Y1]=ode45(@oscil1,[0,100],[20,0],options);
```

```
» plot(T,Y(:,1),'b',T1,Y1(:,1),'r')
```

podem veure que, tal com mostra la Figura 10, en règim permanent les dues solucions són quasi la mateixa. La solució corresponent a $x/10$ cau cap a l'origen més de pressa, ja que, per a $x$ per sobre de 10, la força de recuperació $\arctan(x/10)$ està ja molt a la vora del seu valor asimptòtic i en canvi $x/10$ segueix creixent.

# 9 Exercicis proposats

1. Sigui el sistema

$$\begin{aligned} \dot{x}_1 &= -x_1 + 2x_2, \\ \dot{x}_2 &= \sin x_1 - 3x_2 + \sin t, \end{aligned}$$

per a les variables $x_1(t)$, $x_2(t)$, amb $x_1(0) = 1$, $x_2(0) = -1$. Representa gràficament la solució i determina el valor màxim $M$ de $x_2(t)$ en $[0, 20]$, i el valor de $x_1$ en $t = 20$.

2. Dibuixa la solució de

$$y' + e^{-x}y = \frac{1}{1+x^2},\ y'(0) = 1$$

en $[0, 20]$.

3. Dibuixa la solució de

$$y'' + xy' + \cos y = 0,\ y(0) = 0, y'(0) = 1,$$

en l'interval $[0, 10]$.

4. Sigui el sistema d'EDO

$$\begin{aligned} \dot{x}_1 &= -x_1 + x_2^2 \sin t, \\ \dot{x}_2 &= x_1 - x_2 + \cos x_1, \end{aligned}$$

amb condicions inicials $x_1(0) = 1$, $x_2(0) = 1$. Calcula la solució entre $t = 0$ i $t = 50$ amb `ode45`, i representa $x_1$ i $x_2$ en funció del temps. Repeteix el problema amb

$$\begin{aligned} \dot{x}_1 &= -x_1 + x_2^2 \sin t, \\ \dot{x}_2 &= x_1 - x_2 + \sin x_1. \end{aligned}$$

Segons la gràfica, quin és el règim permanent, és a dir, quan val $x_1(t)$ i $x_2(t)$ per a $t$ gran? Demostra, substituint a l'EDO, que això és realment una solució.

5. Sigui l'EDO de primer ordre

$$\dot{y} = (1 + y^2) \sin t, \quad y(0) = 1.$$

   (a) Calcula la solució exacta (és una EDO de variables separades) i representa-la entre $t = 0$ i $t = 1$.

   (b) Soluciona el mateix problema numèricament amb `ode45` i amb un error relatiu i absolut de $10^{-5}$, i representa també la solució fins a $t = 1$.

(c) Escriu una funció que implementi el mètode d'Euler per a aquest problema. La funció ha d'acceptar com arguments els temps inicial ($t_0$ )i final ($t_f$), la condició inicial ($y_0$), i el nombre de punts ($n$), i ha de retornar el vector de temps $t$ i de solucions $y$ (en els instants de temps). La interfície de la funció ha de ser per tant

```
» [T,Y]=el_nom_que_sigui(t0,tf,y0,n)
```

(d) Crida la funció amb temps inicial igual a zero i temps final igual a 1, amb condició inicial $y(0) = 1$ i amb 10, 50 i 100 punts, respectivament. Representa els resultats i compara'ls entre ells i amb la solució exacta (representada sobre els mateixos punts).

6. El model de Lotka-Volterra representa la interacció entre dues espècies, amb poblacions $x$ (les preses) i $y$ (els depredadors). És un sistema no lineal donat per

$$\begin{aligned} \dot{x} &= \alpha x - \beta xy, \\ \dot{y} &= -\gamma y + \delta xy, \end{aligned}$$

on $\alpha$, $\beta$, $\gamma$ i $\delta$ són paràmetres reals positius. La interpretació dels diferents termes és la següent:

- $\alpha x$ representa el creixement natural de la població de preses, que es suposa que tenen una font inexhaurible d'aliments.
- $-\beta xy$ representa el decreixement de la població de preses degut a la seva interacció amb els depredadors, i es proporcional al producte de les dues poblacions (com més preses i més depredadors, més encontres entre les dues espècies, amb resultats fatídics per a les preses).
- $-\gamma y$ representa la dinàmica de la població de depredadors si no hi ha preses: els depredadors disminueixen degut a la manca d'aliments.
- Finalment, $\delta xy$ té en compte el creixement de la població de depredadors per les seves trobades amb les preses. El coeficients $\beta$ i $\delta$ poden ser diferents degut a que quan un depredador consumeix una presa és produeix una disminució immediata de $x$, però l'efecte sobre $y$ és indirecte (en forma, per exemple, de més descendència). Si penseu que $x$, $y$ representen les masses de les poblacions en lloc de les quantitats d'individus, llavors la diferència entre $\beta$ i $\delta$ té la seva explicació en el fet que un kilogram d'aliment no es tradueix en un augment permanent de massa d'un kilogram.

Posa $\alpha = 2$, $\beta = 0.02$, $\gamma = 1$, $\delta = 0.01$, i considera unes poblacions inicials $x(0) = 80$, $y(0) = 20$. Calcula numèricament la solució en un interval prou gran per poder observar el comportament periòdic resultant. Juga amb els valors dels paràmetres per veure com canvien les solucions.

7. Sigui l'EDO de segon ordre

$$\ddot{x} + 0.002\dot{x} + x = F(t), \quad x(0) = 0,\ \dot{x}(0) = 0.$$

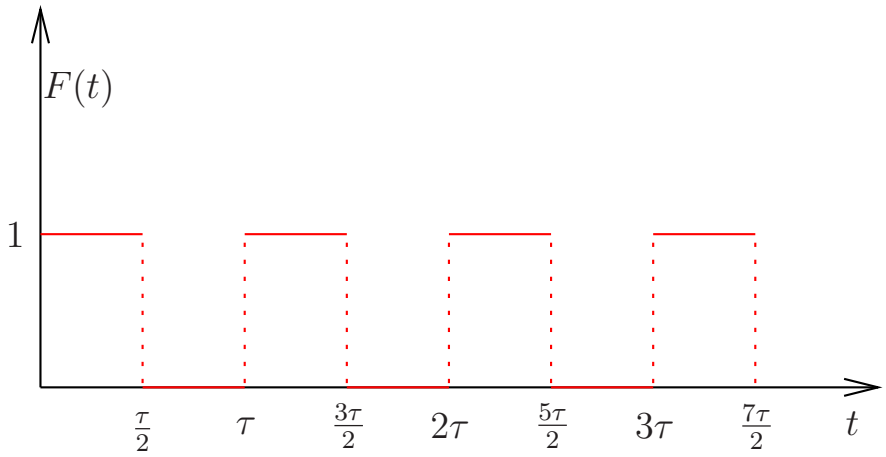



Figura 11: Pols quadrat de període $\tau$.

Això correspon a un oscil·lador molt feblement esmorteït, amb $\omega_n = 1$ i $\xi = 0.001$, sotmès a la força externa $F(t)$. Com a $F(t)$ escollim el pols quadrat de període $\tau$ que apareix a la Figura 11.

(a) Demostra que

$$F(t) = \frac{1 + \text{sign}\left(\sin\frac{2\pi t}{\tau}\right)}{2},$$

on sign és la funció signe, que val $-1$ per a argument negatiu i $+1$ per a argument positiu. Ajuda: calcula el membre de la dreta per a $t \in (0, \tau/2)$ i per a $t \in (\tau/2, \tau)$.

(b) Per a un sistema de segon ordre amb $\xi$ tant petit, la freqüència de ressonància en amplitud es pràcticament igual a la freqüència natural, $\omega_A \approx \omega_n = 1$. Escull $\tau$ de manera que la freqüència fonamental de $F(t)$ sigui $\omega_n = 1$ (recorda que la relació entre període i freqüència angular és $\omega = \frac{2\pi}{T}$).

(c) Escriu una funció que implementi l'equació diferencial amb el valor de $\tau$ obtingut, i calcula la solució numèrica del problema amb `ode45` entre $t = 0$ i $t = 4000$, amb errors relatiu i absoluts iguals a $10^{-5}$. Representa $x(t)$. Cal notar que el temps característic del sistema és $1/0,002 = 500$, i que per tant cal anar a temps de l'ordre de 2000 per a que comenci el règim permanent. Per veure les oscil·lacions, caldrà que amplifiquis la figura en un rectangle molt petit al voltant de $t = 3500$. Anota el valor de la magnitud de les oscil·lacions.

(d) Repeteix el problema amb valors de $\tau$ lleugerament per sota i per sobre (per exemple, a distància $\pm 0.1$). Quina conclusió en treus?

8. **Justificació numèrica de la resposta impulsiva.** Sigui el sistema entrada/sortida

$$\ddot{x} + 3\dot{x} + 2x = u(t), \quad y = x.$$

(a) Calcula la funció de transferència $H(s)$ de $u$ a $y$, i la seva antitransformada de Laplace $h(t)$, la resposta impulsiva.

(b) Sigui $\epsilon > 0$. Agafa $u(t)$ de la forma

$$\delta_\epsilon(t) = \begin{cases} 1/\epsilon & \text{si } t < \epsilon, \\ 0 & \text{si } t > \epsilon. \end{cases}$$

Això és una funció que, per a tot valor de $\epsilon$, té àrea 1, i com més petit es fa $\epsilon$ més *concentrada* està en $t = 0$. Es pot representar, emprant la funció de Heaviside, com

$$\delta_\epsilon(t) = \frac{1}{\epsilon}\theta(\epsilon - t).$$

Calcula la solució numèrica de l'EDO amb condicions inicials nul·les, entre $t = 0$ i $t = 5$, amb $\epsilon = 0.5$, $\epsilon = 0.1$ i $\epsilon = 0.01$, i vegeu que el resultat s'acosta cada vegada més a $h(t)$. Representa les quatre corbes a la mateixa gràfica (la de $h(t)$ amb 100 punts; caldrà que utilitzis `linspace(0,5,100);` per obtenir els valors de l'eix de temps). La funció de Heaviside en MATLABés `heaviside`.

9. **Un problema amb rigidesa numèrica.** Sigui l'EDO no lineal

$$\dot{y} = y^2 - y^3,$$

amb condició inicial $y(0) = \delta > 0$, amb $\delta < 1$, i considereu el càlcul numèric de la seva solució en $t \in [0, 2/\delta]$. La solució exacta es pot calcular (l'EDO és de variables separades), però sols es pot obtenir implícitament (no es pot aïllar $y(t)$) i no aporta massa informació. En canvi, es poden obtenir resultats qualitatius sobre la solució sense calcular-la. Tenint en compte el signe de $y^2 - y^3$ per a $y < 1$ i $y > 1$, es veu immediatament que la solució creix si està per sota de $y = 1$ i decreix si està per sobre de $y = 1$. Això vol dir que si la solució comença per sota de $y = 1$ mai podrà superar aquest valor.

(a) Posa $\delta = 0.01$ i calcula la solució en $[0, 200]$ emprant `ode45`.

(b) Posa ara $\delta = 0.00001$ i repeteix el càlcul, obtenint la solució en $[0, 2 \cdot 10^5]$. Inspecciona en detall (amb un *zoom*) el comportament al voltant de $t = 1/\delta = 10^5$. Observaràs que en aquest punt el càlcul es fa molt lent i apareixen unes oscil·lacions per sobre i per sota de $y = 1$ que no corresponen a la solució real. Es diu que el problema numèric associat a l'equació diferencial és rígid (*stiff*). Per solucionar això, cal utilitzar mètodes numèrics especials, en lloc de `ode45`. Prova `ode15s`, `ode23s` i `ode23tb`, i compara els resultats.

# Part III

# Part 3 — Anàlisi de Fourier (opcional)

## 10 Referències a funcions

Fins ara hem vist com crear funcions mitjançant fitxers M. Es poden crear també funcions sense crear un fitxer, en la pròpia finestra de comandes, mitjançant un mecanisme anomenat creació d'una referència a funció (*function handle*). Les funcions creades d'aquesta manera poden a més passar-se com arguments a altres funcions, definides per exemple en fitxers M, i això és especialment útil si volem escriure una funció que calculi els coeficients de Fourier d'una funció donada, sense haver de fixar la funció dins el fitxer M.

El mecanisme de creació de referències a funcions utilitza el caràcter `@`, tal com mostra el següent exemple:

```
» f=@(t) t*sin(t)
```

Això defineix la funció $f(t) = t \sin t$. La funció es pot cridar de la manera usual:

```
» f(pi/2)
```

Fixem-nos que a la finestra de variables hi ha aparegut un objecte associat a la funció. La funció es pot passar com a referència a altres funcions predefinides, com ara `ezplot`, que permet representar-la sense haver d'utilitzar `linspace`:

```
» ezplot(f,[-4*pi,4*pi])
```

(poden sortir alguns *warnings* però apareix igualment la gràfica). Si pensem representar la funció directament construint el vector de punts, cal substituir el producte normal pel producte element a element:

```
» f=@(t) t.*sin(t)
```

```
» f(pi/2)
```

```
» T=linspace(-4*pi,4*pi,100);
```

```
» plot(T,f(T))
```

Les funcions definides d'aquesta manera poden també tenir un nombre qualsevol d'arguments. Per exemple, la funció $g : \mathbb{R}^2 \to \mathbb{R}$ definida per

$$g(x, y) = \frac{\sin(x^2 + y^2)}{x^2 + y^2}$$

seria

```
» g=@(x,y) sin(x^2+y^2)/(x^2+y^2)

» g(0,pi)
```

Es poden també definir funcions vectorials. Per exemple, la corba $h : \mathbb{R} \to \mathbb{R}^3$ donada per

$$h(t) = (\cos t, \sin t, \log(1+t))$$

es pot definir i avaluar en un punt com

```
» h=@(t) [cos(t) sin(t) log(1+t)]

» h(pi)
```

Recordem de la pràctica anterior que quan cridàvem `ode45` ho fèiem posant `@` davant el nom de la funció definida en un fitxer M . Això és degut a que `ode45` és una funció que té com un dels arguments una altra funció, i utilitza per tant el pas per referència. Tenim però la possibilitat de definir la referència a la funció directament a la línia de comandes, i això pot estalviar temps si l'EDO és senzilla. Per exemple, sigui l'EDO d'ordre 2

$$y'' + xy' + y = \sin x, \quad y(0) = 1,\ y'(0) = 0,$$

que com a sistema de primer ordre és

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= -xy_2 - y_1 + \sin x, \end{aligned}$$

amb $y_1(0) = 1$, $y_2(0) = 0$. La podem definir a la finestra de comandes com

```
» dy=@(x,y) [y(2); -x*y(2)-y(1)+sin(x)]
```

Si volem trobar la solució entre $x = 0$ i $x = 50$ farem llavors

```
» [X,Y]=ode45(dy,[0,50],[1,0]);
```

Fixem-nos que hem posat `dy` i no `@dy` a `ode45`, ja que `dy` ja és una funció definida per referència. Si dibuixeu la solució amb

```
» plot(X,Y(:,1))
```

podreu observar un comportament remarcable.

# 11 Exemple: polinomi de Taylor d'una funció

Volem construir primer una funció `taylor1` en un fitxer M que accepti com arguments una funció $f(x)$, un punt $x_0$ i un natural $n$ i retorni un vector amb el valor de la funció i de les seves primeres $n$ derivades en el punt $x_0$. El fitxer M de la funció apareix a la Figura 12, i el codi bàsic és

```
function a=taylor1(f,x0,n)
    a=zeros(n+1,1);
    syms x;
    a(1)=subs(f(x),x0);
    for k=2:n+1
        g=@(x) diff(f(x),x);
        a(k)=subs(g(x),x0);
        f=@(x) g(x);
    end
end
```

Fixem-nos que utilitzem definicions de funcions per referència dins del fitxer M. Cal assenyalar a més que, com que `x` és una variable simbòlica, s'ha d'utilitzar la substitució `subs(g(x),x0)` per avaluar en un punt les funcions que depenen d'ella.

Volem ara escriure una funció que calculi el polinomi de Taylor simbòlic (és a dir, en funció de la variable $x$), a partir de la informació computada per `taylor1`. Recordem que aquest polinomi ve donat per

$$P_{f,n,x_0}(x) = f(x_0) + \sum_{k=1}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k.$$

El fitxer M apareix a la Figura 13, i el codi essencial és

```
function s=taylor2(a,x,x0)
    n=size(a,1);
    s=a(1);
    fact=1;
    for k=1:n-1
        fact=fact*k;
        s=s+a(k+1)/fact*(x-x0)^k;
    end
end
```

Les següents comandes il·lustren la utilització d'aquestes funcions. En primer lloc, es defineix la funció $f(x) = \sin(x)/(x-1)$ mitjançant una referència:

```
» f=@(x) sin(x)./(x-1)
```

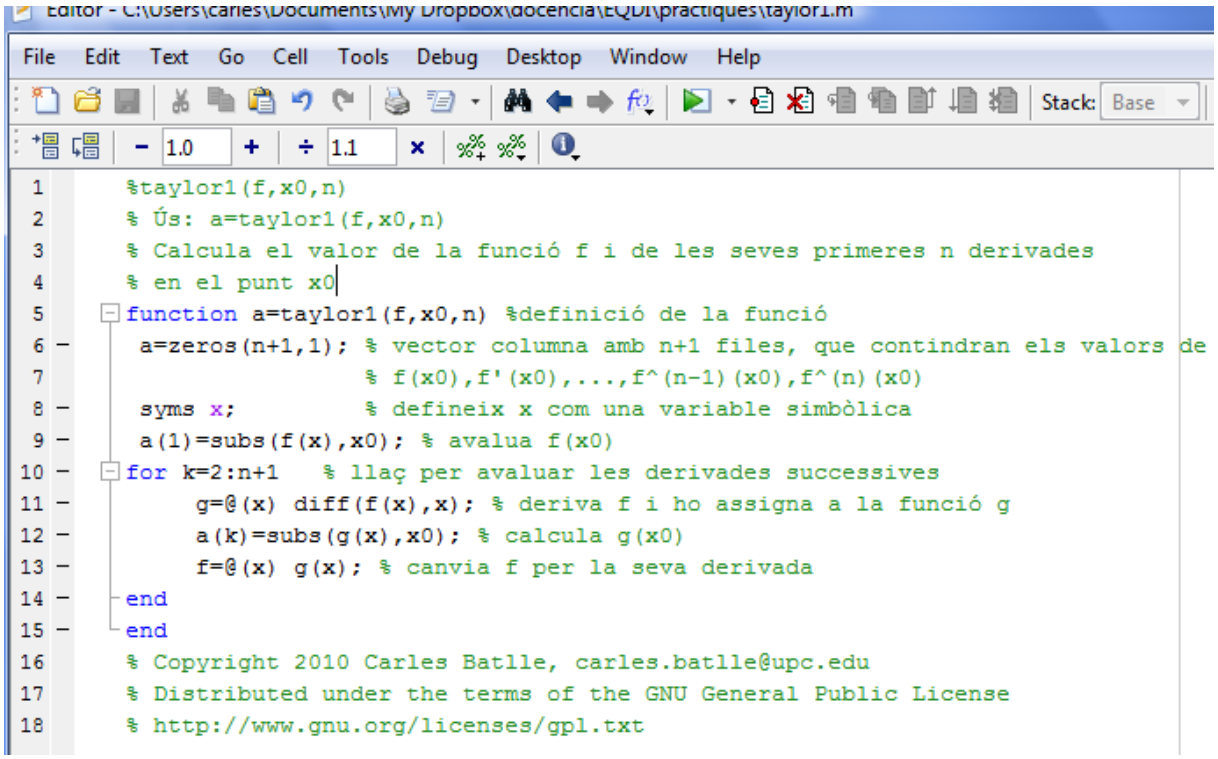



Figura 12: La funció `taylor1.m`.

on hem utilitzat la notació de divisió vectorial ja que després dibuixarem la gràfica de la funció (no cal si sols hem d'usar $f$ dins de `taylor1`). Ara cridem `taylor1` (amb $x_0 = 0$ i $n = 6$):

```
» a=taylor1(f,0,6)
```

Això retorna el valors $f(0) = 0$, $f'(0) = -1$, $f''(0) = -2$, $f'''(0) = -5$, $f^{(iv)}(0) = -20$, $f^{(v)}(0) = -101$ i $f^{(vi)}(0) = -606$ en el vector `a`, que ara passem a `taylor2`, definint primer `x` com a variable simbòlica:

```
» syms x;

» s=taylor2(a,x,0)
```

El resultat a la finestra de comandes és

```
s =

- (101*x^6)/120 - (101*x^5)/120 - (5*x^4)/6 - (5*x^3)/6 - x^2 - x
```

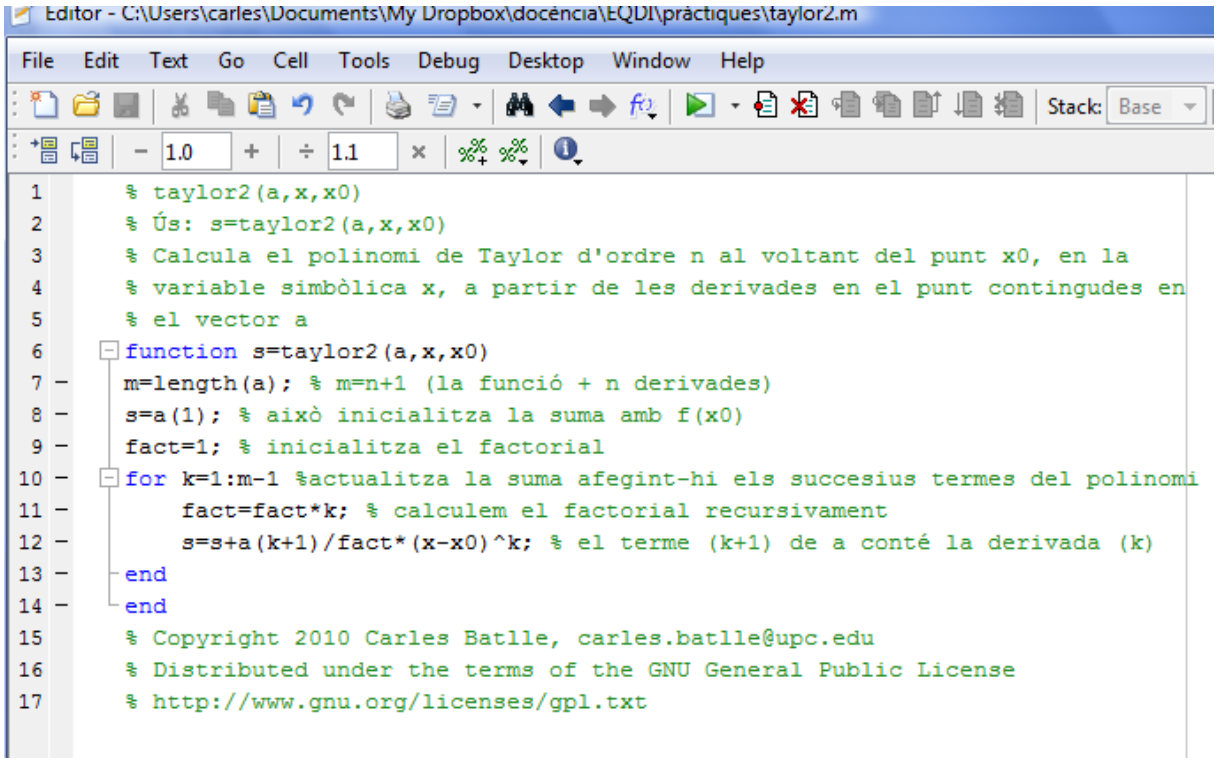



Figura 13: La funció `taylor2.m`.

Les següents instruccions permeten comparar el polinomi d'ordre 6 amb la funció, en l'interval $(-1.6, 0.9)$:

```
» X=linspace(-1.6,0.9,100);
```

```
» plot(X,f(X),'b',X,subs(s,X),'r')
```

De nou, com que `x` és una variable simbòlica, cal usar `subs` per avaluar-la en el conjunt de punts `X`. La Figura 14 mostra els resultats. A l'acostar-se a 1 per l'esquerra la funció tendeix cap a $-\infty$, mentre que si ens allunyem molt de 0 per la l'esquerra el polinomi agafa valors molts negatius.

# 12 Exemple: suma de Riemann d'una funció

Com a segon exemple de funció definida en un fitxer M que admet funcions com arguments, considerarem el problema d'avaluar una integral com a suma de les àrees de $n$ rectangles,

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{k=0}^{n-1} f(\xi_k)\Delta_k$$

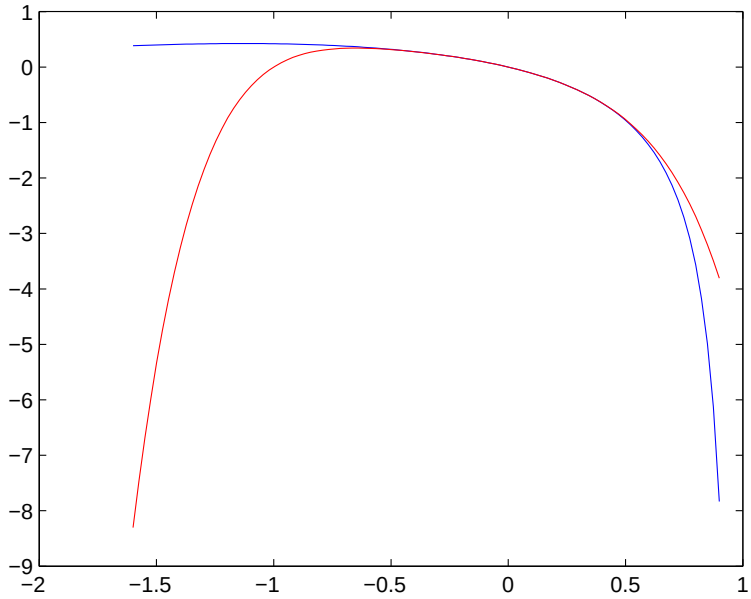



Figura 14: La funció $f(x) = \sin x/(x-1)$ (blau) i el seu polinomi de Taylor d'ordre 6 (vermell) al voltant de $x = 0$.

on $\xi_k$ és un punt de la base del rectangle $k$-ésim i $\Delta_k$ és la seva base. Això s'anomena una *suma de Riemann* de la funció en $[a, b]$, i per simplificar posarem $\Delta_k = \Delta$ i agafarem $\xi_k$ com el punt més a l'esquerra de cada rectangle, tal com mostra la Figura 15. D'aquesta manera la suma de Riemann queda com

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{k=0}^{n-1} f(a+k\Delta)\Delta.$$

El que volem ara és implementar una funció que proporcioni aquest resultat, i a més, per comparar, també el resultat exacte.

La Figura 16 mostra el codi de `riemann`, que avalua la integral exacta simbòlicament i després utilitza la regla de Barrow.

Podem ara comparar l'efecte de l'augment del nombre de rectangles a la suma de Riemann en el càlcul de

$$\int_1^5 x \sin x\,\mathrm{d}x$$

```
» f=@(x) x*sin(x)

» [R,I]=riemann(f,1,5,20)
```

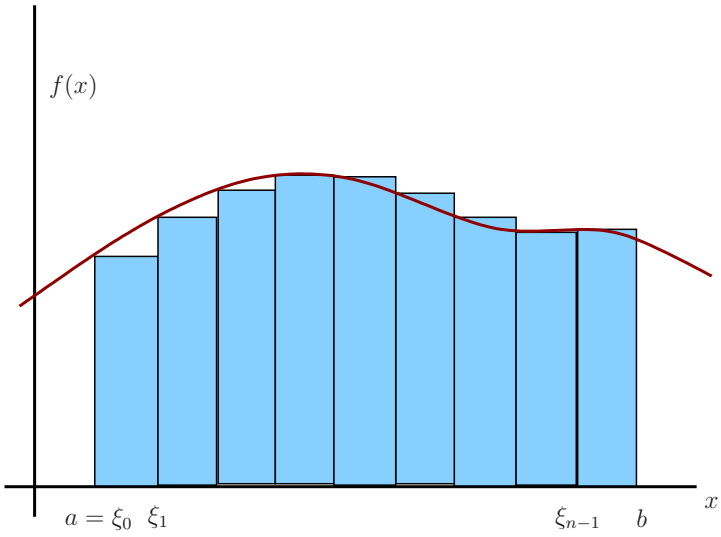



Figura 15: Suma de Riemann per aproximar la integral de $f(x)$ entre $a$ i $b$.

```
» [R,I]=riemann(f,1,5,1000)
```

# 13 Coeficients de Fourier i sumes parcials de la sèrie de Fourier d'una funció

Les funcions `fourier1` (Figura 17) i `fourier2` (Figura 18), que teniu en fitxers M, porten a terme el càlcul dels coeficients i sumes parcials de la sèrie de Fourier d'una funció:

**`fourier1`** Pren com arguments la referència a una funció $f$, el període $T$ i el nombre de termes de Fourier $n$, i retorna el vector de coeficients $\{a_k\}_{k=0\ldots n}$ i el vector de coeficients $\{b_k\}_{k=0\ldots n}$:

$$\begin{aligned} a_k &= \frac{2}{T}\int_0^T f(t)\cos\frac{2\pi kt}{T}\,\mathrm{d}t,\ k=0,\ldots,n,\\ b_k &= \frac{2}{T}\int_0^T f(t)\sin\frac{2\pi kt}{T}\,\mathrm{d}t,\ k=0,\ldots,n, \end{aligned}$$

on $b_0 = 0$ apareix per simplicitat.

**`fourier2`** Pren com arguments els vectors de coeficients $a$ i $b$, el període i la variable $t$ i retorna la suma parcial dels $n$ primers harmònics de la sèrie de Fourier avaluada en $t$:

$$SF_n(f(t)) = \frac{a_0}{2} + \sum_{k=1}^{n}\left(a_k\cos\frac{2\pi kt}{T} + b_k\sin\frac{2\pi kt}{T}\right).$$

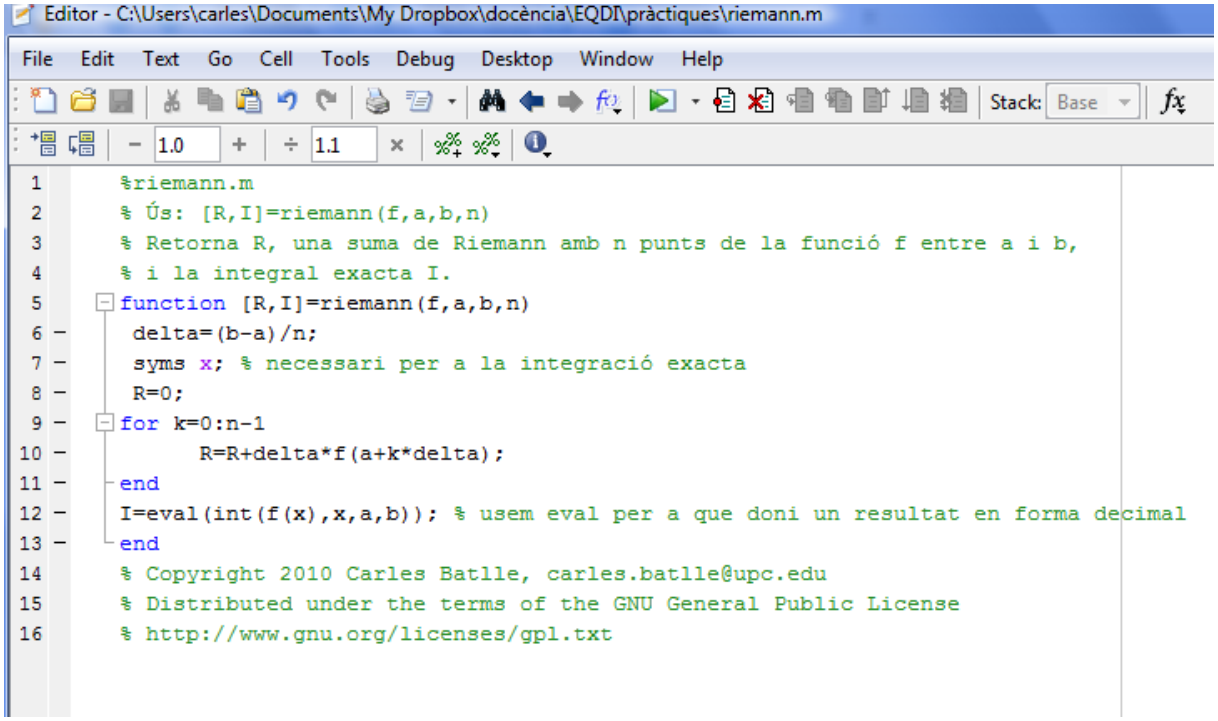



Figura 16: Codi de `riemann.m`, amb un càlcul simbòlic de la integral exacte.

Llegeix el codi de les dues funcions i tracta d'entendre'l. Com que MATLAB utilitza índexs que comencen per 1 en lloc de per 0, l'element `a(1)` conté $a_0$, i l'element `b(3)` dóna el valor de $b_2$, per exemple.

Com exemple d'utilització d'aquestes funcions, sigui

$$f(t) = te^{-t}, \quad t \in [0, 3),$$

amb període $T = 3$.

Definim en primer lloc (la vectorització del producte és necessària si volem representar la funció)

```
» f=@(t) t.*exp(-t)
```

Això no incorpora la definició del període, i per tant cal dir a la funció que si $t$ està fora de $[0, 3)$ ha de buscar el valor corresponent a $[0, 3)$. Això es fa amb la funció `mod`, que és tal que $\mod(t, T)$ retorna la resta de dividir $t$ entre $T$:

```
» mod(5.2,3)

» f(mod(5.2,3))
```

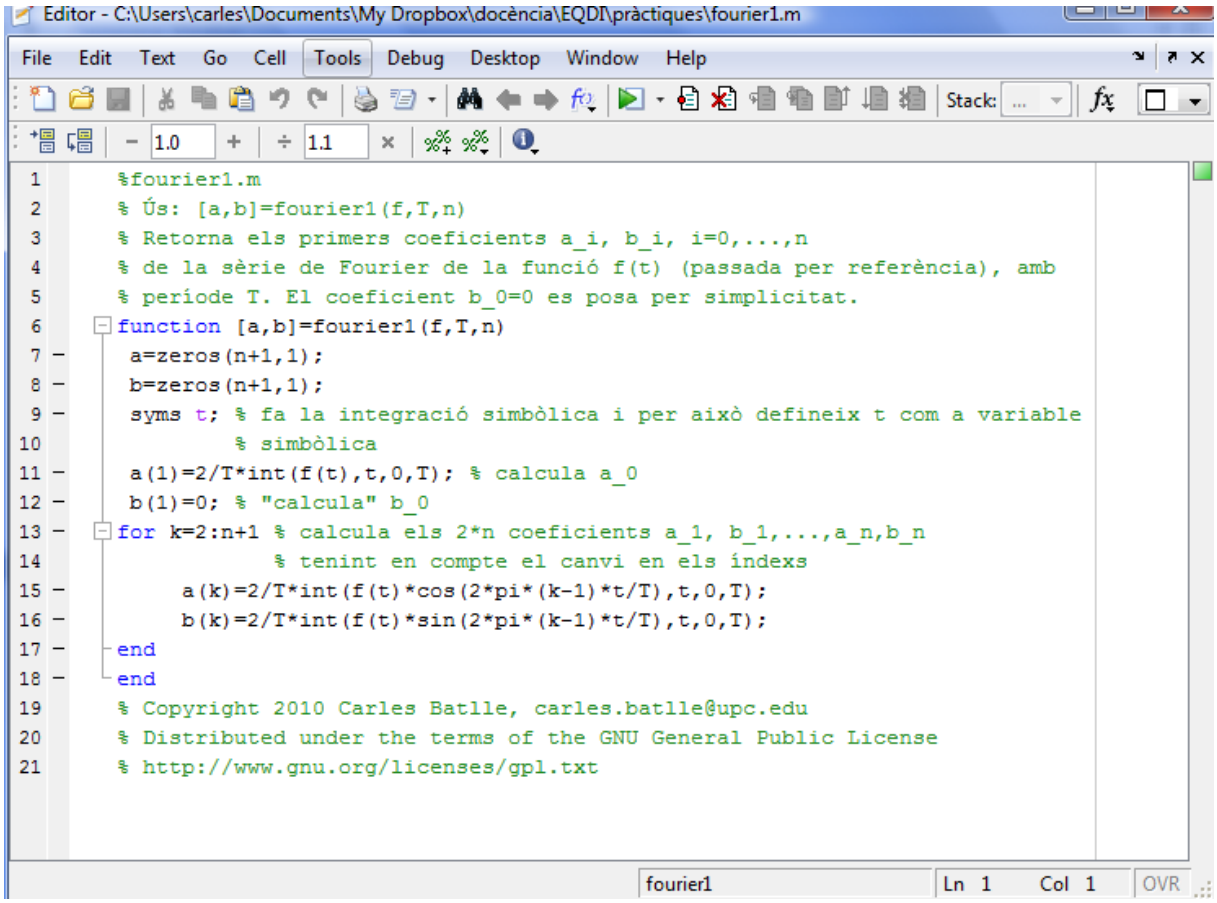



Figura 17: La funció `fourier1.m`.

Per representar $f(t)$ entre $0$ i $8$ amb 300 punts, per exemple, caldrà per tant fer

```
» I=linspace(0,8,300)
» plot(I,f(mod(I,3)))
```

Si volem calcular els coeficients fins a $n = 5$ de $f(t)$ farem

```
» [a,b]=fourier1(f,3,5)
```

i obtindrem $a_0 = 0.5339$, $a_1 = -0.0924$, ..., $a_5 = -0.0065$, $b_0 = 0$, $b_1 = 0.0527$, ..., $b_5 = -0.0083$. Aquest resultat es pot passar ara a la funció `fourier2`, que dóna la suma parcial de la sèrie de Fourier calculada en un punt específic. Per exemple

```
» fourier2(a,b,3,1.15)
```

calcula $SF_5(f(t))$ en $t = 1.15$ (el valor $n = 5$ l'obté directament de la longitud d'`a`). Si volem representar aquesta suma parcial en l'interval $I = [0, 8]$ farem (suposant que volem mantenir la gràfica de $f(t)$ que ja tenim)

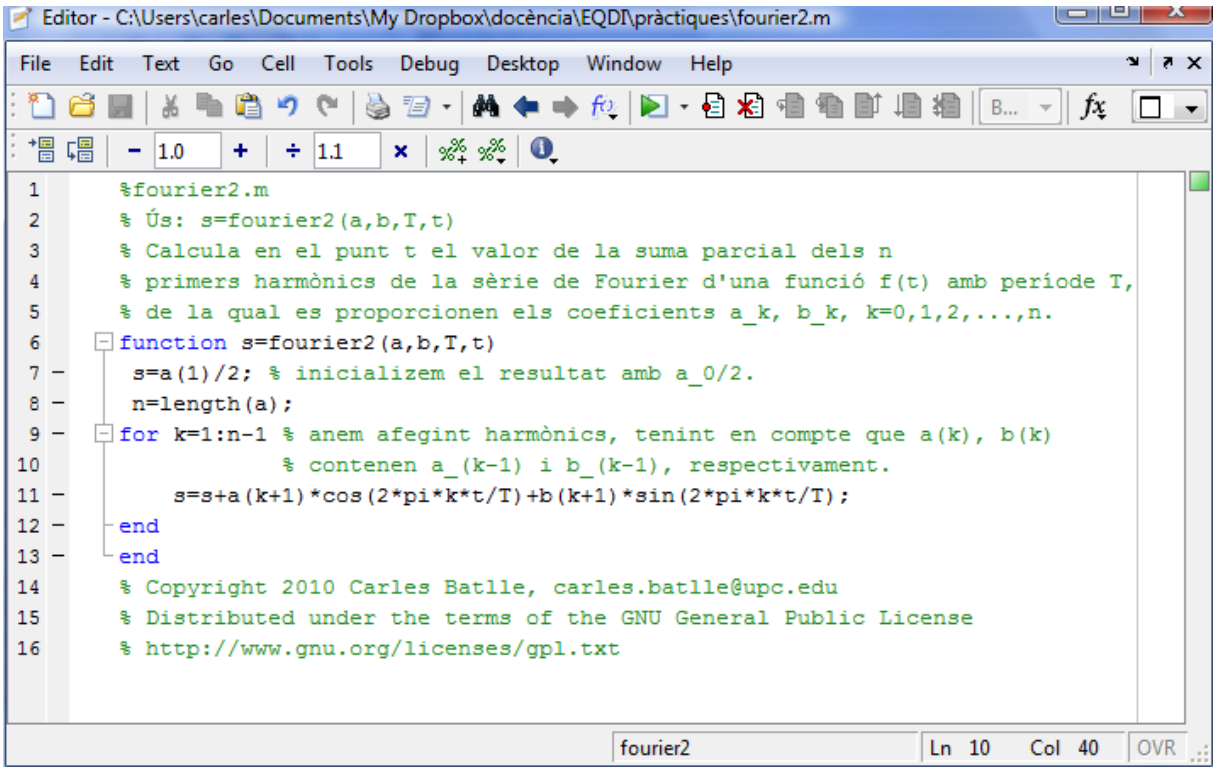



Figura 18: La funció `fourier2.m`.

```
» hold on

» plot(I,fourier2(a,b,3,I),'r')
```

# 14 Algunes eines més per a funcions periòdiques

Hem vist que per representar en un interval $I = [a, b)$ qualsevol una funció periòdica $f(t)$ definida a $[0, T)$ hom pot utilitzar `f(mod(I,T))`, ja que la funció `mod(t,T)` retorna el valor en $[0, T)$ que difereix de $t$ en un nombre enter de períodes. Sovint, però, la funció $f(t)$ no es defineix en $[0, T)$, sinó en un interval $[a, b)$ qualsevol, per exemple $[-T/2, T/2)$. Emprar llavors la funció `mod` dóna el resultat incorrecte, ja que retorna igualment un valor entre 0 i $T$, i no un valor entre $-T/2$ i $T/2$, que és on coneixem la funció.

La Figura 19 mostra el codi de `Tab.m`, que permet representar una funció periòdica amb període $T = b - a$ en un interval qualsevol a partir de la seva definició en $[a, b)$.

Sigui per exemple la funció

$$f(t) = |t|, \quad t \in [-1, 1)$$

amb període $T = 2$. Si volguéssim definir la mateixa funció donant els seus valors en $[0, 2)$,

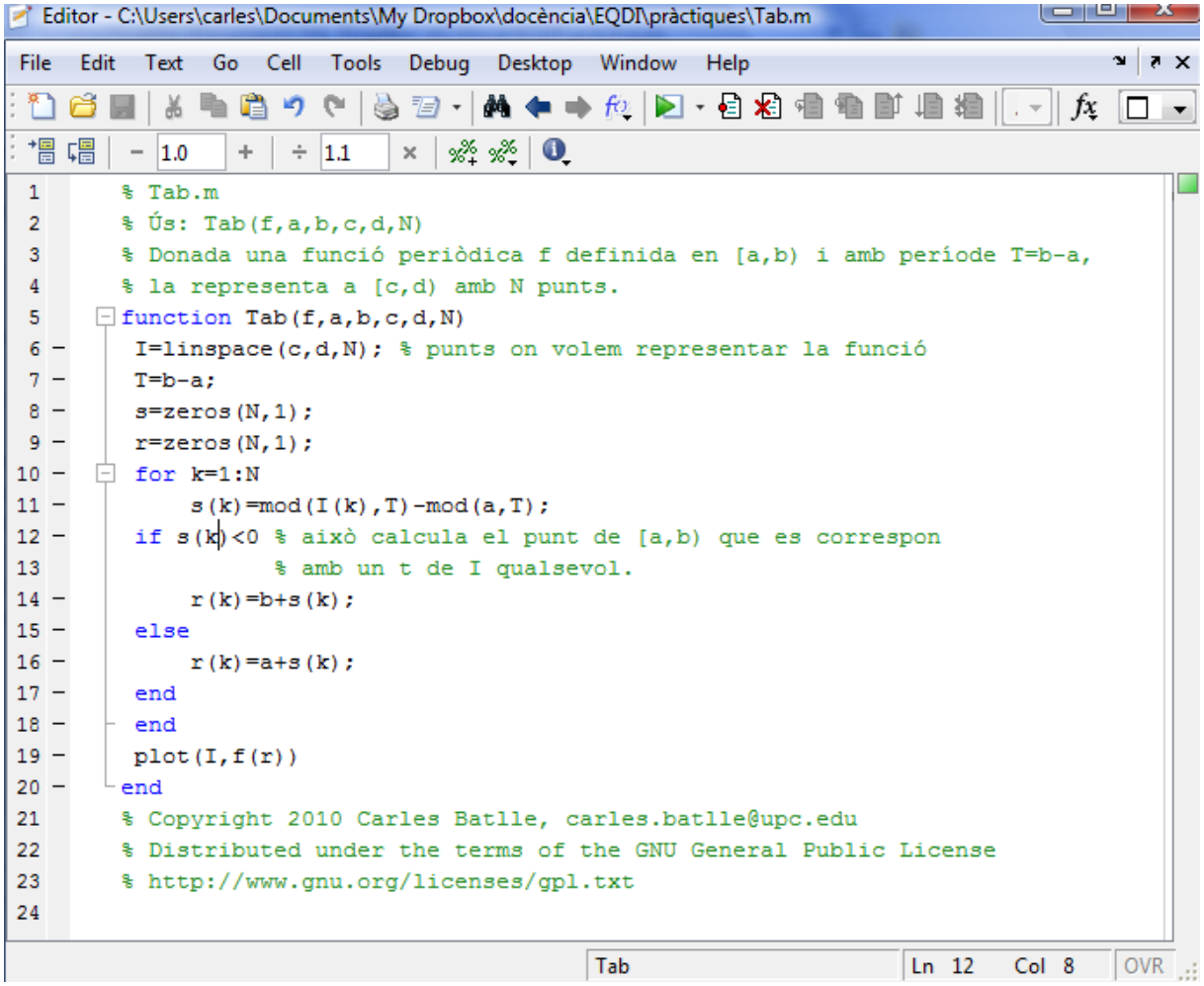



Figura 19: La funció `Tab.m`.

hauríem de fer-ho necessàriament a trossos. De fet, obtindríem

$$f(t) = \begin{cases} t & \text{si } t \in [0,1], \\ 2-t & \text{si } t \in (1,2), \end{cases}$$

i és millor treballar amb la definició original. Per representar-la en $[-3,4]$ amb 700 punts emprant `Tab` farem

```
» f=@(t) abs(t)

» Tab(f,-1,1,-3,4,700)
```

i el resultat apareix a la Figura 20. Fixem-nos que, efectivament, per a $t \in [1,2)$ la funció val $2-t$.

El codi de la funció `fourier` apareix a la Figura 21. Aquesta és una variació de la funció `fourier1`, que permet calcular els coeficients però amb la integral sobre qualsevol interval

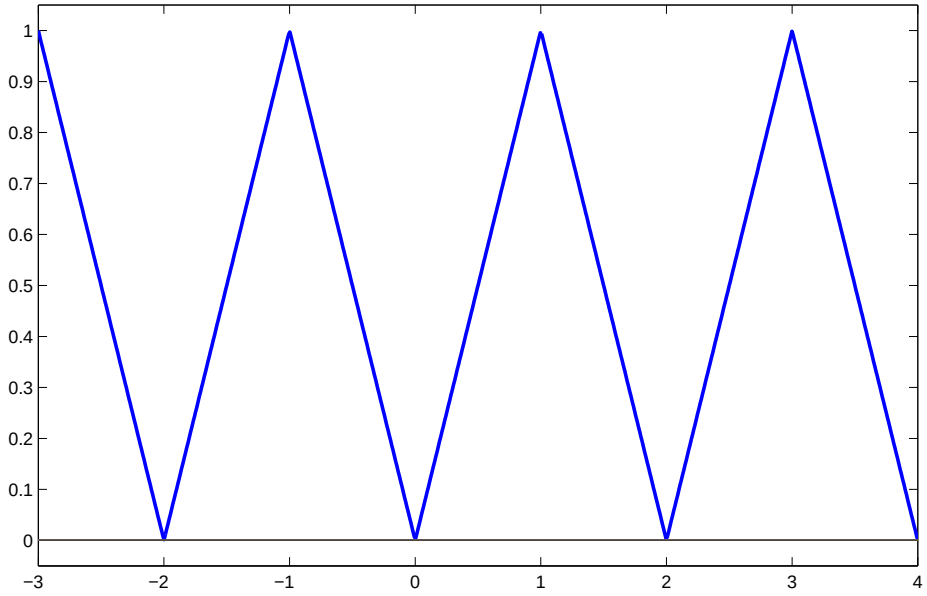



Figura 20: El senyal $f(t) = |t|$ per a $t \in [-1, 1)$, estès amb període $T = 2$, representat a $[-3, 4]$.

$[t_1, t_2)$ de longitud $T$. De nou, això facilita el càlcul si la funció no està definida originalment en $[0, T)$. Com ja sabem, la integral d'una funció periòdica no depèn de quin interval es seleccioni, i aquesta funció implementa la forma

$$\begin{aligned} a_n &= \frac{2}{T}\int_{t_1}^{t_2} f(t)\cos\frac{2\pi nt}{T}\,\mathrm{d}t, \quad n = 0, 1, 2, \ldots, \\ b_n &= \frac{2}{T}\int_{t_1}^{t_2} f(t)\sin\frac{2\pi nt}{T}\,\mathrm{d}t, \quad n = 1, 2, \ldots \end{aligned}$$

amb $t_2 = t_1 + T$.

Per exemple, per calcular els 20 primers coeficients de la funció que estem considerant, farem

```
» [a,b]=fourier(f,-1,1,20)
```

Observareu que el càlcul tarda una mica, igual que passava amb `fourier1`, degut a que el càlcul exacte de les integrals és molt costós. El temps de càlcul es pot reduir enormement si s'utilitza integració numèrica en lloc d'exacta. Això és el que fa la funció `fourierQ`, el codi de la qual apareix a la Figura 22. El mètode numèric que utilitza és una quadratura de Simpson adaptativa, mitjançant una funció anomenada `quad`. Cal notar que `quad` requereix que la funció li sigui passada per referència, i per això s'han de crear, per a cada harmònic, funcions que defineixin els productes de $f(t)$ amb els cosinus i sinus corresponents.

Per cridar-la, hom fa

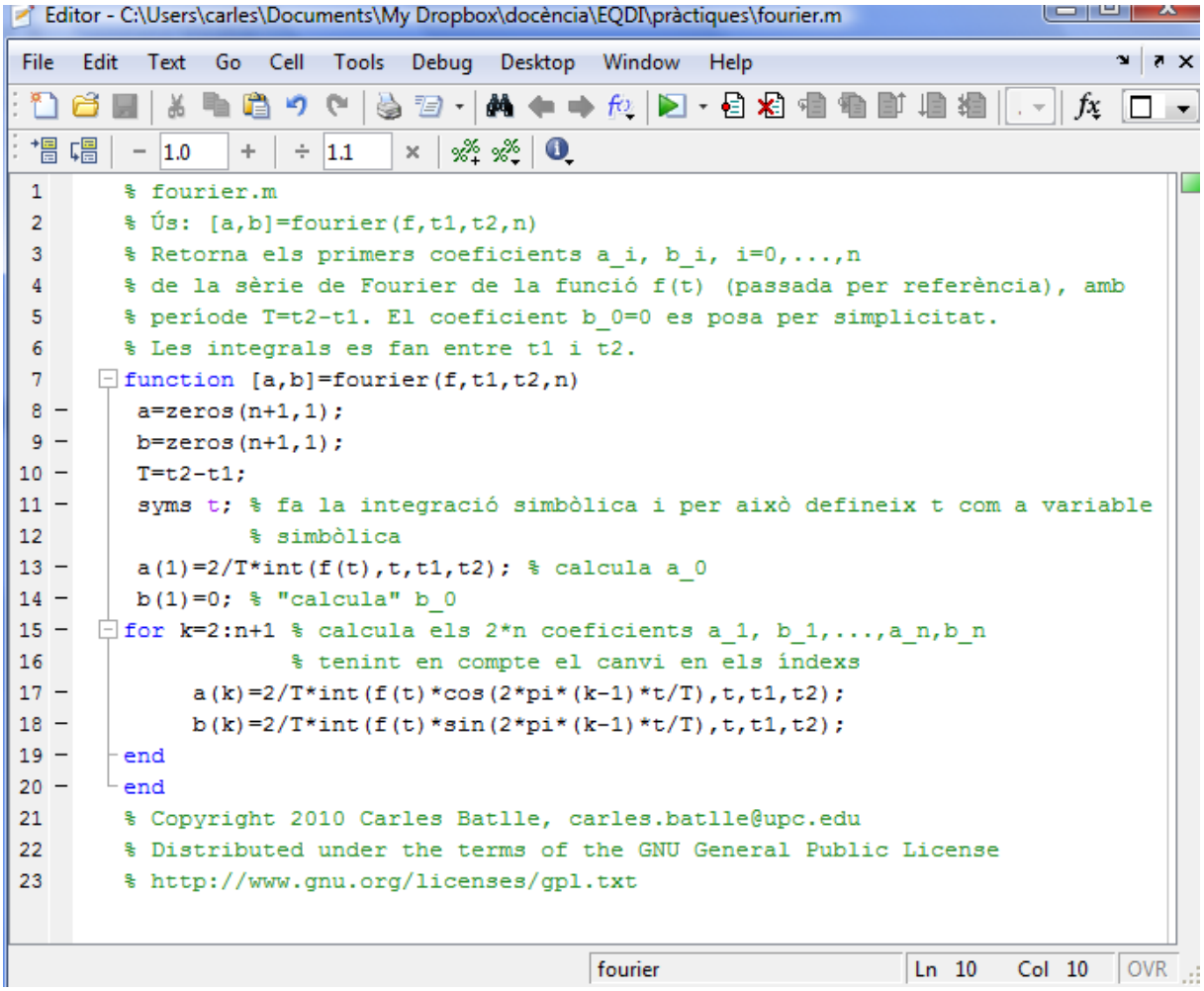



Figura 21: La funció `fourier.m`.

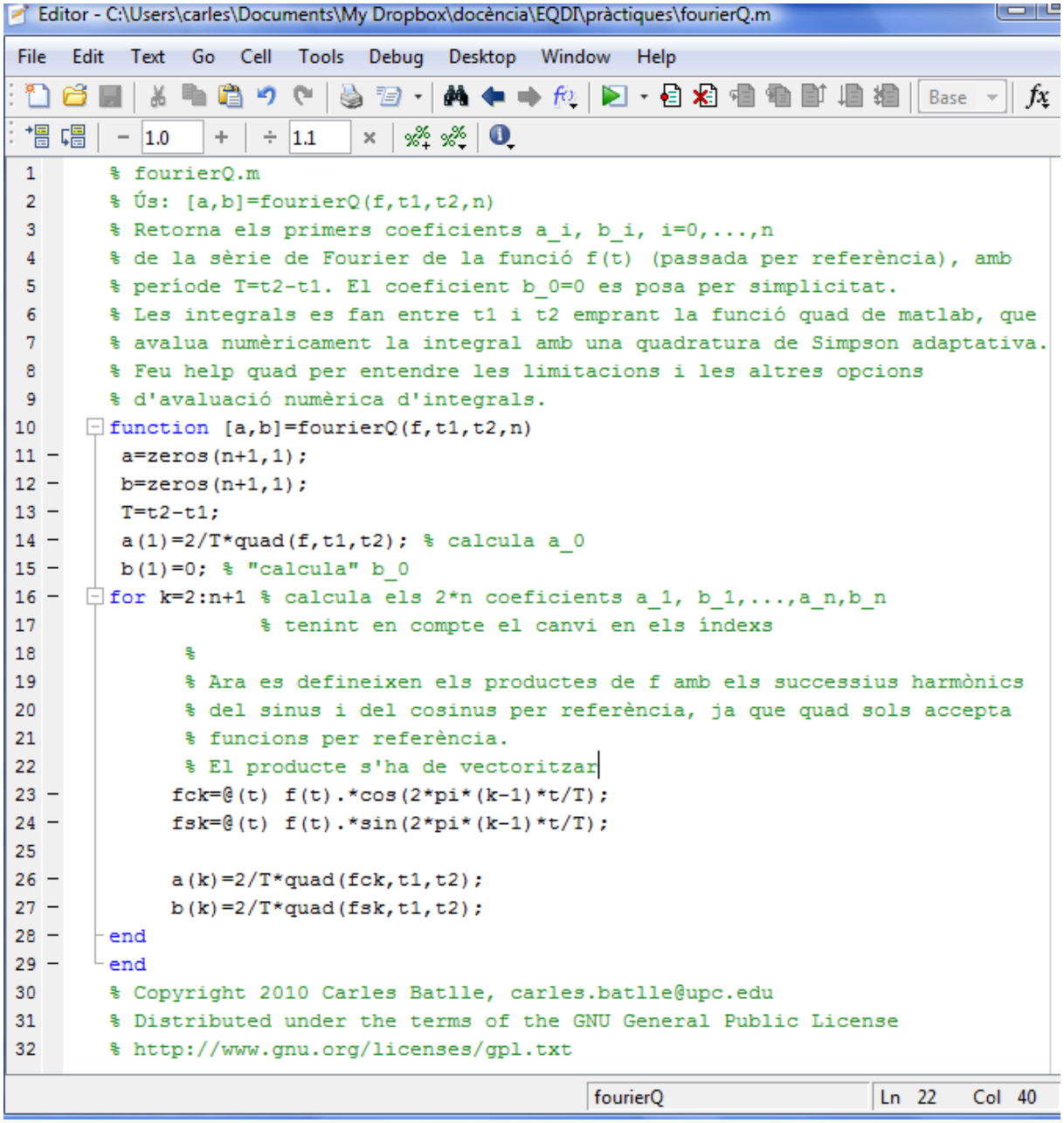



Figura 22: La funció `fourierQ.m`.

```
» [aQ,bQ]=fourierQ(f,-1,1,20)
```

El resultat és instantani, i la diferència amb el càlcul *exacte* és menyspreable. Podem calcular la màxima diferència entre les amplituds dels harmònics per un i altre mètode amb

```
» max((a.^2+b.^2)-(aQ.^2+bQ.^2))
```

que dóna de l'ordre de $10^{-9}$. De tota manera, l'error relatiu amb `fourierQ` és més important com més alt és l'harmònic, ja que per a funcions ràpidament oscil·lants els mètodes numèrics tenen, en general, problemes. En alguns casos això pot donar lloc a magnituds d'harmònics elevats molt més grans que les exactes, i cal llavors assegurar-se que no és un error d'integració numèrica, comparant per exemple amb el resultat que dóna `fourier`.

Podem mesurar més acuradament la diferència de velocitat entre el càlcul exacte i el numèric emprant les funcions `tic` i `toc`:

```
» clear all; f=@(t) abs(t)

» tic; [a,b]=fourier(f,-1,1,20);  toc;

» tic; [aQ,bQ]=fourierQ(f,-1,1,20); toc;
```

on hem netejat la memòria i tornat a definir $f(t)$ per evitar que el programa utilitzi els valors que ja ha calculat.

A més de la velocitat, `fourierQ` també té l'avantatge de poder-se utilitzar amb funcions que contenen expressions per a les quals no es pot calcular les primitives que apareixen en els coeficients de Fourier. Això pot passar també si la funció té expressions que no estan definides per a variables simbòliques, que és el que passa amb la funció signe:

$$f(t) = \text{sign}(t-1) = \begin{cases} -1 & \text{si } t \in [0,1), \\ 1 & \text{si } t \in (1,2). \end{cases}$$

Hom té

```
» f=@(t) sign(t-1)
```

i `fourier` dóna un error:

```
» [a,b]=fourier(f,0,2,20)
```

que és degut a que la funció `sign` no està definida, en aquesta versió de MATLAB, per a variables simbòliques. En canvi `fourierQ` no es queixa:

```
» [aQ,bQ]=fourierQ(f,0,2,20)
```

No hi cap problema, en canvi, amb la funció de Heaviside. Per exemple, per a la funció graó de període 1

$$f(t) = \theta(t-1/2),\ t \in [0,1)$$

es poden calcular els coeficients tant amb `fourier` com amb `fourierQ`:

```
» f=@(t) heaviside(t-1/2)
» [a,b]=fourier(f,0,1,20)
» [aQ,bQ]=fourierQ(f,0,1,20)
```

# 15 Representació de l'espectre d'un senyal periòdic

Sigui la funció de període $T = 4\pi$ definida per

$$f(t) = |t| \sin t, \quad t \in [-2\pi, 2\pi).$$

Per representar-la en $[-6\pi, 6\pi]$ fem

```
» f=@(t) abs(t).*sin(t)
» Tab(f,-2*pi,2*pi,-6*pi,6*pi,800)
```

i el resultat, amb algun tractament posterior, apareix a la Figura 23.

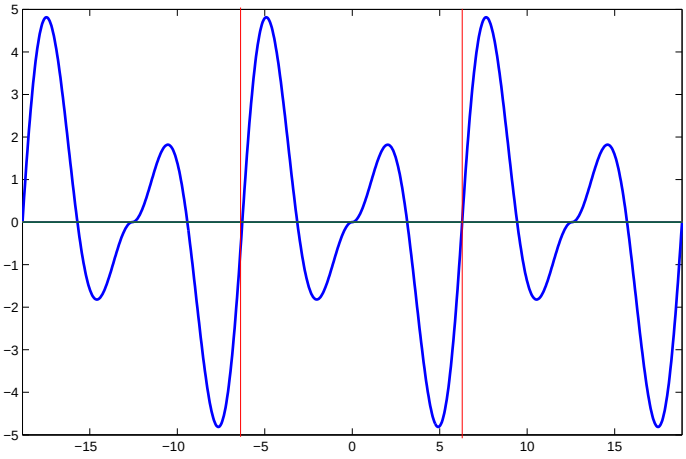



Figura 23: La funció $f(t) = |t| \sin t$, $t \in [-2\pi, 2\pi)$.

Calculem els primers 20 coeficients de la seva sèrie de Fourier:

```
» [a,b]=fourierQ(f,-2*pi,2*pi,20)
```

Els coeficients complexos corresponents són

$$c_n = \frac{a_n - jb_n}{2}, \quad n = 0, 1, 2, \ldots, 20,$$

i podem representar el seu mòdul, obtenint l'espectre en amplitud, mitjançant

```
» stem(0:1:length(a)-1,abs((a-j*b)/2))
```

La funció `stem` és com `plot` però, en lloc d'unir els punts, a cada punt dibuixa un cercle i l'uneix amb l'eix d'abscisses. El codi `0:1:length(a)-1` crea un eix d'abscisses amb els $N+1$ valors $0, 1, 2, \ldots, N$, on $N$ és el nombre d'elements del vector `a` menys un. El resultat, amb una mica de post-producció, el mostra la Figura 24.

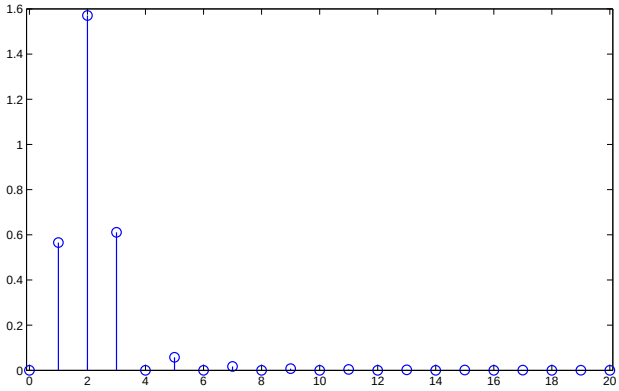



Figura 24: Els primers 21 punts de l'espectre en amplitud de $f(t) = |t| \sin t$, $t \in [-2\pi, 2\pi)$.

S'observa que l'amplitud més gran correspon al segon harmònic, de període $4\pi/2 = 2\pi$, amb contribucions notables del primer i tercer harmònics, mentre que la resta, excepte el cinquè harmònic, té molt poca importància. La predominança del segon harmònic s'explica pel fet que el senyal és bàsicament un sinus, amb període $2\pi$, però deformat per $|t|$ i amb període $4\pi$.

# 16 La transformada discreta de Fourier

Sigui la funció de període $T = 2$

$$f(t) = t^2, \quad t \in [-1, 1).$$

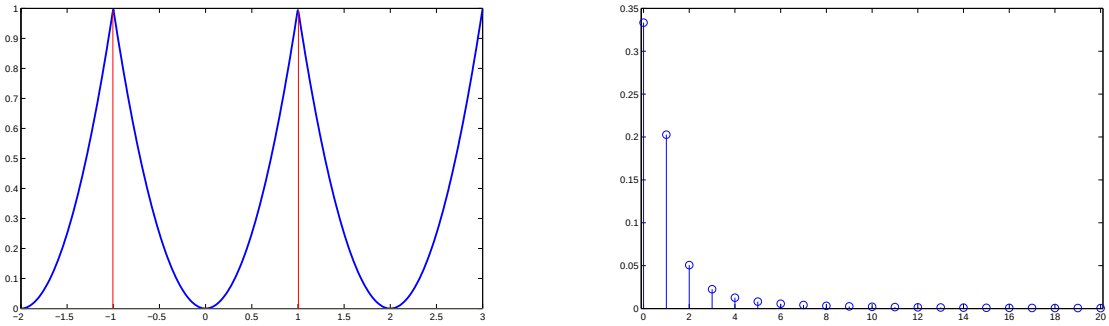
La funció i el seu espectre en amplitud fins l'harmònic $n = 20$ apareixen a la Figura 25.

Encara que no és un senyal de banda limitada, els coeficients de la sèrie de Fourier poden considerar-se menyspreables a partir de $n > M = 6$ i, a efectes pràctics, podem considerar que la freqüència màxima present en el senyal és

$$\nu_{\text{màx}} = \frac{M}{T} = 3.$$

El nombre mínim $N$ de punts per període que ha de tenir un mostratge per poder recuperar el senyal a partir d'una DFT ha de complir

$$N > 2M = 12$$



(a) (b)

Figura 25: La funció $f(t) = t^2$, $t \in [-1, 1)$, i el seu espectre en amplitud.

i com que $N$ ha de ser parell tindrem $N = 14$. Per a un mostratge amb $N$ punts que verifiqui la condició de Nyquist tenim, per a un senyal de banda limitada, que els coeficients complexos es poden obtenir amb

$$c_k = \frac{1}{N} X_k, \quad \text{per a } 0 \leq k \leq \frac{N}{2}, \tag{1}$$

$$c_{-N+k} = \frac{1}{N} X_k, \quad \text{per a } \frac{N}{2} < k \leq N - 1. \tag{2}$$

Com que el nostre senyal és tan sols aproximadament de banda limitada, aquestes relacions seran aproximades. El que sí que és cert, de tota manera, és que aquestes relacions pressuposen que el mostratge es fa en un interval de la forma $[0, T)$. Si això no és així, apareixen uns factors que s'han de tenir en compte. Com que la nostra funció està definida de manera natural en $[-1, 1)$, haurem de redefinir-la en $[0, 2)$ si volem evitar aquests factors.

Per a $t \in (1, 2)$, el nostre senyal ve donat per $(t-2)^2$, que és la mateixa paràbola que $t^2$ però amb vèrtex en $t = 2$ en lloc de $t = 0$. Per tant, en termes de la funció de Heaviside,

$$f(t) = t^2 + \theta(t-1)\left((t-2)^2 - t^2\right), \ t \in [0, 2).$$

Per definir-la en MATLAB fem

```
» f=@(t) t.^2+heaviside(t-1).*((t-2).^2-t.^2)
```

Això dóna la mateixa funció que la definició original de $f(t)$, tal com es pot veure amb la gràfica,

```
» I=linspace(-2,3,600)
```

```
» plot(I,f(mod(I,2))
```

on no cal, ara, utilitzar `Tab`, ja que `mod` fa correctament la feina.

Per obtenir un mostratge amb $N = 14$ punts en $[0, 2)$ cal fer

```
» x=zeros(1,14);for k=1:14 x(k)=f(0+2/14*(k-1)); end;
```

Això proporciona els valors

| $n$ | 0 | 1 | 2 | $\cdots$ | 12 | 13 |
|---|---|---|---|---|---|---|
| $x_n$ | 0 | 0.0204 | 0.0816 | $\cdots$ | 0.0816 | 0.0204 |

La DFT de la seqüència de $N$ valors $\{x_n\}_{n=0,1,\ldots,N-1}$ és la seqüència de $N$ valors (complexos) $\{X_n\}_{n=0,1,\ldots,N-1}$ calculada com

$$X_k = \sum_{l=0}^{N-1} e^{-j\frac{2\pi kl}{N}} x_l, \quad k = 0, \ldots, N-1.$$

Aquest càlcul es pot posar equivalentment en forma matricial,

$$\begin{pmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-2} \\ X_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & f & f^2 & \cdots & f^{N-2} & f^{N-1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & f^{N-2} & f^{2(N-2)} & \cdots & f^{(N-2)(N-2)} & f^{(N-1)(N-2)} \\ 1 & f^{N-1} & f^{2(N-1)} & \cdots & f^{(N-2)(N-1)} & f^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{pmatrix},$$

on

$$f = e^{-j\frac{2\pi}{N}}.$$

La Figura 26 mostra el codi de `dft.m`, que efectua aquest càlcul.

El càlcul de la DFT per a $N$ gran (en aplicacions no massa espectaculars $N$ pot ser de l'ordre de $10^5$) és força costós des del punt de vista computacional. Cada terme de la DFT requereix $N$ multiplicacions de nombres complexos, i el càlcul d'una DFT complerta implica per tant $N^2$ multiplicacions. Cooley i Tukey van crear el 1965 un algorisme que calcula la DFT (si $N$ és una potència de 2) de manera recurrent i que sols necessita $\frac{1}{2}N\log_2 N$ productes. Aquest algorisme s'anomena FFT (*Fast Fourier Transform*), i tant la versió original com les desenvolupades posteriorment per casos quan $N$ no és una potència de 2 s'utilitzen industrialment i en tecnologies de la informació i les telecomunicacions per a processar immenses quantitats de dades en temps real. Per exemple, si $N = 10^5$, l'algorisme DFT que hem presentat requereix $10^{10}$ multiplicacions, mentre que la FFT original de Cooley i Tukey sols en necessita

$$\frac{1}{2} \cdot 10^5 \cdot \log_2 10^5 = 5 \cdot 10^4 \cdot 5 \cdot \log_2 10 \approx 8.6 \cdot 10^4$$

és a dir un guany d'un factor de més de $10^5$. Això vol dir que si un ordinador fa el càlcul de la FFT en un segon, per fer el mateix càlcul amb la DFT li caldria un dia!

Amb un processador Intel Core2 Duo T7500 @2.20 GHz, fer la DFT de $10^3$ punts amb MATLAB mitjançant la multiplicació matricial costa una mica menys de mig segon, mentre que tarda menys de $0.0005$ segons emprant la FFT que MATLAB mateix té implementada.

Per emprar l'algorisme FFT de MATLAB, que s'anomena `fft`, sobre la seqüència $x$, cal fer

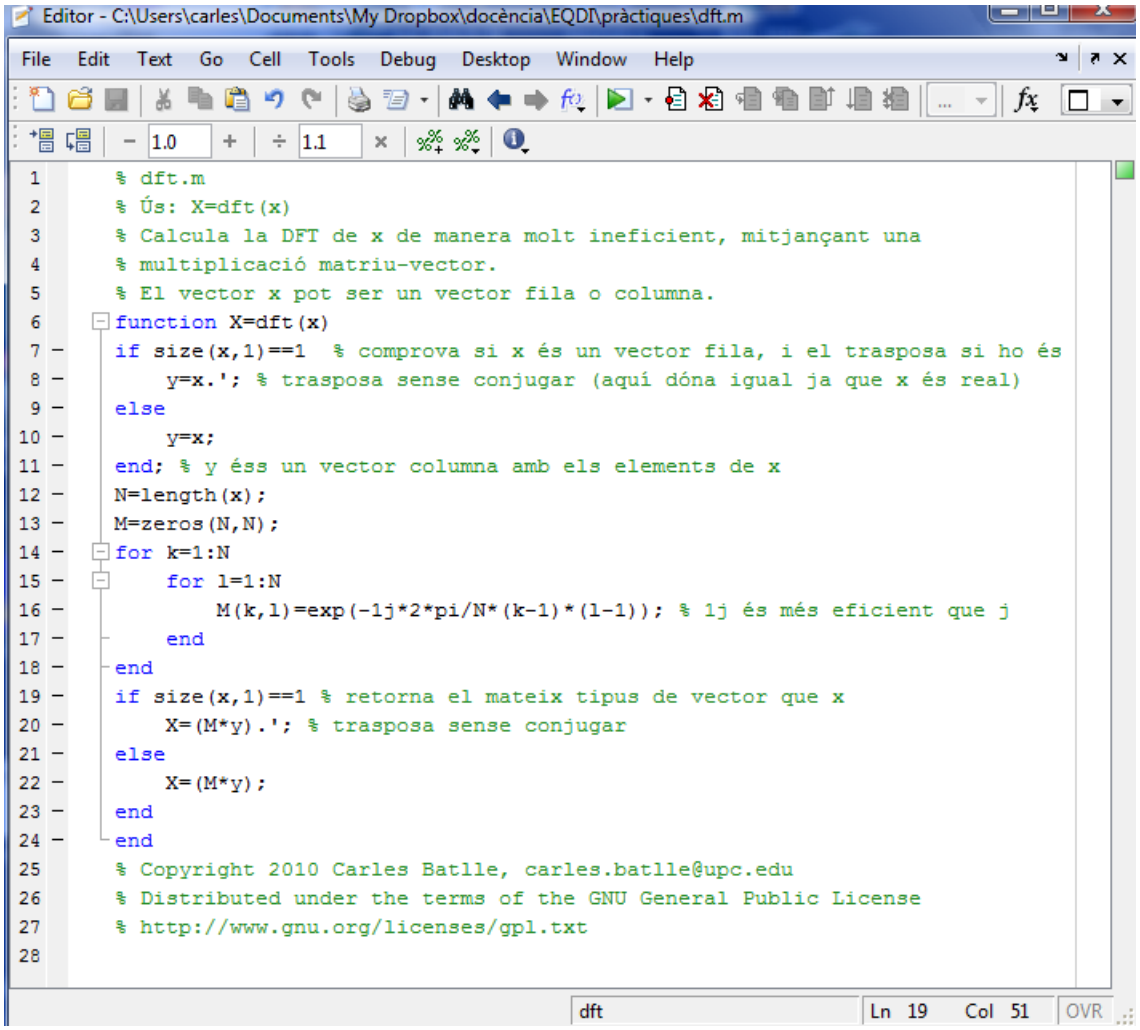



Figura 26: Codi de la funció `dft`, que calcula la DFT a partir de la definició original.

```
» X=fft(x)
```

amb resultat

| $n$ | 0 | 1 | 2 | $\cdots$ | 12 | 13 |
|---|---|---|---|---|---|---|
| $X_n$ | 4.7143 | $-2.8851 + 0.0j$ | $0.7588 - 0.0j$ | $\cdots$ | $0.7588 + 0.0j$ | $-2.8851 - 0.0j$ |

Tenint en compte (1) podem calcular els $N/2 + 1 = 8$ primers coeficients $c_n$:

```
» cDFT=1/14*X(1:8)
```

Els coeficients $a_n$, $b_n$ s'obtenen dels $c_n$ mitjançant

$$a_n = c_n + \bar{c}_n, \tag{3}$$
$$b_n = j(c_n - \bar{c}_n). \tag{4}$$

En el nostre cas, com que els $c_n$ són reals, els $b_n$ són tots zero (la funció és simètrica), i els primers 8 coeficients $a_n$ calculats per la DFT els obtenim amb

```
» aDFT=cDFT+conj(cDFT)
```

Amb

```
» [a,b]=fourier(f,0,2,9);
```

```
» stem(0:1:length(aDFT)-1,abs(a(1:8)-aDFT'))
```

podem veure la diferència entre els $a_k$, $k = 0, 1, 2, \ldots, 7$, calculats *exactament* i amb la DFT del mostratge. El tros de codi `aDFT'` converteix el vector fila `aDFT` en un vector columna, per poder restar-lo de `a(1:8)`. El resultat apareix a la Figura 27, i es pot veure que la diferència, que no és nul·la degut a que el senyal no és realment de banda limitada, és petita. La diferència va disminuint a mesura que es fan mostratges amb un nombre de punts cada vegada més elevat. Per exemple, els mateixos primers 8 coeficients calculats amb una DFT a partir d'un mostratge amb $N = 50$ s'obtenen amb un error menor que $6 \cdot 10^{-4}$.

# 17 Exercicis proposats

1. Defineix per referència la funció

$$f(x) = (x + 1)^4 \sin x.$$

   (a) Calcula el seu valor en $x = \pi/4$.

   (b) Utilitza la funció `taylor1` per calcular el valor de $f(x)$ i de les seves set primeres derivades en $x = 0$.

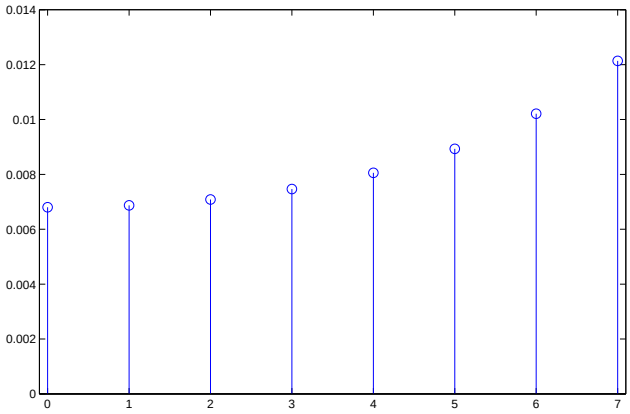



Figura 27: Diferència entre els coeficients $a_k$, $k = 0, 1, 2, \ldots, 7$, calculats exactament i mitjançant una DFT.

(c) Utilitza la funció `taylor2` per calcular el polinomi de Taylor de $f(x)$ d'ordre 7 en el punt $x = 0$, que anomenarem $P_7(x)$.

(d) Representa la funció $f(x)$ i el polinomi de Taylor $P_7(x)$ en $[-2.5, 2.2]$.

(e) Representa el valor absolut de la diferència entre la funció $f(x)$ i el polinomi de Taylor $P_7(x)$ en $[-2.5, 2.2]$.

2. Calcula la solució de

$$y' = x + \sin y, \ y(0) = 1,$$

definint l'EDO en la línia de comandes mitjançant una referència.

3. Escriu una funció de nom `FOMAp1` en un fitxer M, que prengui com arguments la referència a una funció $f$ i un parell de valors reals $x_0$ i $x_1$ i retorni $f'(x_0)$, $f'(x_1)$ i

$$\int_{x_0}^{x_1} f(x)\,\mathrm{d}x.$$

Aplica la funció a $f(x) = (x + 1)^{10} \sin 3x$, amb $x_0 = 0$ i $x_1 = 1$.

4. Escriu una funció que prengui com arguments una funció $f(x)$ definida per referència, un interval $[a, b]$ i un neter $N$, i que representi $f(x)$ en blau, $f'(x)$ en vermell i $f''(x)$ en verd, en l'interval $[a, b]$ i amb $N$ punts.

5. (a) Sigui $f(t) = t$ en $[0, 2\pi]$, amb període $2\pi$.

i. Usant la funció `fourier1`, calcula els coeficients de la seva sèrie de Fourier fins a $n = 3$, $n = 8$ i $n = 20$ (anomena els vectors de coeficients amb noms diferents per a diferent $n$).

ii. Calcula l'amplitud del setè harmònic de $f(t)$.

iii. Emprant `fourier2`, representa, en una mateixa gràfica i al llarg de **dos períodes**, la funció $f(t)$ i les sumes parcials $SF_n(f(t))$ de la seva sèrie de Fourier, amb $n = 3$, $n = 8$ i $n = 20$.

iv. Digues si el fenomen de Gibbs hi és present (utilitza l'eina de *zoom* de la finestra de gràfiques) i justifica el resultat a partir de la teoria.

(b) Ídem per a $g(t) = \sin t$ en $[0, \pi]$ amb període $\pi$.

(c) Ídem per a $h(t) = t^2$ en $[0, 1]$ amb període 1.

Fent

```
» help fourier1
```

o

```
» help fourier2
```

pots recordar la interfície de les dues funcions en qualsevol moment.

6. Escriu una funció de nom `fourierc` amb la mateixa interfície que `fourier1`, però que retorni els coeficients complexos de Fourier d'una funció periòdica.

7. Siguin les funcions periòdiques

$$\begin{aligned} f(t) &= t + 2(1-t)\theta(t-1) + 2(t-3)\theta(t-3),\ t \in [0, 4), \\ g(t) &= \sin 2t + 3\cos 4t - \sin 7t,\ t \in [0, 2\pi) \\ h(t) &= \theta(t)e^{-t},\ t \in [-1, 2), \end{aligned}$$

Defineix-les per referència. Para a cadascuna de les funcions:

(a) Representa la funció en l'interval $[0, 2T)$, on $T$ és el període corresponent.

(b) Usant la funció `fourier` o la funció `fourierQ`, calcula els coeficients de la sèrie de Fourier fins al dotzè harmònic.

(c) Calcula els coeficients complexos $c_n$, $n = 0, 1, \ldots, 12$ de la sèrie de Fourier.

(d) Representa l'espectre en amplitud fins al dotzè harmònic.

(e) Raona si el senyal és de banda limitada.

8. Dels dos senyals de l'exercici anterior que no són de banda limitada, n'hi ha un per al qual els valors de l'espectre en amplitud es poden considerar menyspreables abans que els de l'altre. Les següents preguntes es refereixen a ell.

(a) Determina una freqüència a partir de la qual els valors de l'espectre en amplitud siguin menyspreables. Pren aquesta freqüència com a màxima.

(b) Fes un mostratge del senyal amb un nombre de punts parell de manera que la freqüència del mostratge estigui per sobre de la de Nyquist, però el més proper possible.

(c) Utilitzant la funció `fft`, calcula la DFT d'aquest mostratge.

(d) Compara el resultat de la DFT amb el càlcul exacte dels coeficients de Fourier del senyal. Pots, per exemple, representar els mòduls de la diferència entre els coeficients $c_n$ calculats a partir dels $a_n$ i $b_n$ i els coeficients $c_n$ obtinguts de la DFT. Quina conclusió en treus?

9. (a) Construeix la seqüència de valors $x_k = \sin k$, $k = 1, 2, \ldots, 100$. Quant val $x_{73}$?

(b) Repassa el codi de `dft.m`.

(c) Fes la DFT de $\{x_k\}$ emprant la funció `dft` i la funció pròpia de MATLAB `fft`. Representa el mòdul de la diferència dels dos resultats. Quina conclusió en treus?

(d) Usant les funcions `tic` i `toc`, compara els temps d'execució de 1000 vegades el càlcul de la DFT emprant `dft` i `fft`.

10. Escriu una funció, `idft`, que calculi la iDFT.

11. Sigui l'equació diferencial

$$\ddot{x} = 0.1 \sin x + \cos t$$

amb $x(0) = 0$, $\dot{x}(0) = 0$.

(a) Escriu-la com un sistema de dues equacions de primer ordre.

(b) Integra-la entre $t = 0$ i $t = 100$, i dibuixa la primera component de la solució en aquest interval.

(c) L'entrada del sistema és $\cos t$, que té freqüència $\omega_0 = 1$. Porteu a terme una DFT de la solució. Quines freqüències s'observen? Hauries de detectar un subharmònic; això passa degut a que el sistema és no-lineal; en sistemes lineals, la solució sols pot mostrar les freqüències d'entrada.

12. **El Niño**. El fenomen climatològic de *El Niño* és conseqüència de canvis de la pressió atmosfèrica al oceà Pacífic sud. El *Southern oscillation Index* és la diferència entre la pressió atmosfèrica a l'illa de Pàscua i la de Darwin (al nord-est d'Austràlia), mesurada a nivell del mar al mateix temps. En el fitxer `elnino.dat` hi trobaràs dades mensuals d'aquest índex entre 1962 i 1975. Analitza la periodicitat d'aquestes dades. Hauries de veure un cicle dominant de 12 mesos, i un segon cicle menys important de període més llarg.