

CÀLCUL INFINITESIMAL I
Pràctiques de Laboratori. Curs 2005-06

Antoni Susin i Sànchez
Departament de Matemàtica Aplicada I
Universitat Politècnica de Catalunya (UPC)

Part I

Familiarització amb **Matlab**

Capítol 1

Introducció a `Matlab`

1.1 Què és `Matlab`?

`Matlab` és l'acrònim anglès de *Matrix Laboratory*. Però què és `Matlab`? Realment és un entorn de computació tècnica i matemàtica, altament especialitzat en manejar matrius i vectors de forma ràpida i eficient.

Per tant, `Matlab` és una plataforma ideal per càlcul numèric, raó de l'existència d'aquest document.

1.2 Primer contacte amb l'entorn matemàtic

Quan iniciem el programa `Matlab`, ens apareix en pantalla una interfície gràfica com la que es mostra a la figura 1.1. Allà es poden observar tres zones clarament diferenciades:

- **Finestra de comandes.** La pantalla realment principal del programa. És la finestra on s'interactua amb el programa, on es declaren les variables a usar i es criden a les funcions a usar.
- **Historial de comandes.** Mostra la línia temporal d'ús del programa, totes les comandes entrades per ordre cronològic
- **Workspace.** Aquesta és la part més avançada. Allà apareixen els mòduls programats de `Matlab` carregats en memòria (a la secció ??, a la pàgina ??, farem èmfasi en aquest apartat; de moment no hi entrarem).

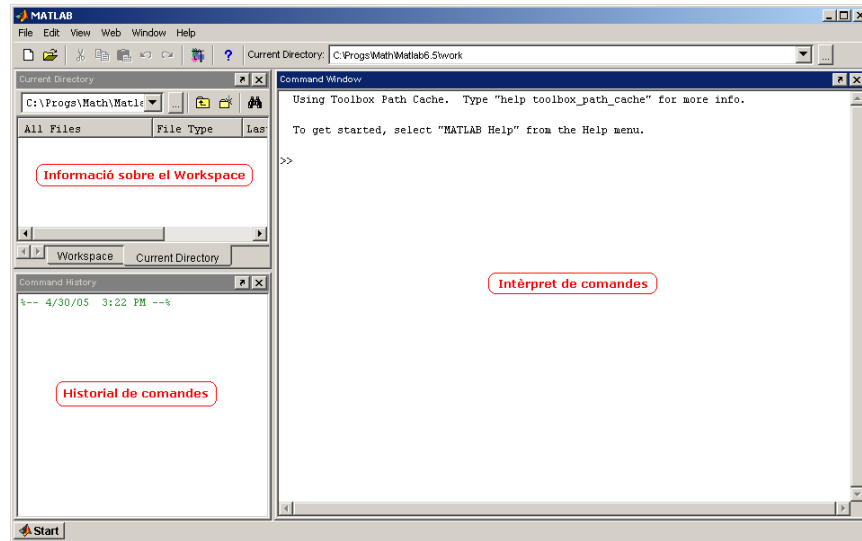


Figura 1.1: Finestra principal de l'entorn Matlab, amb les tres zones principals clarament diferenciades: a) la finestra de comandes, b) l'historial de comandes, i c) la finestra de la zona de treball (*Workspace*)

1.3 Primers passos amb Matlab

1.3.1 Càlcul d'expressions

Matlab funciona exactament com una calculadora: computa tant funcions lineals, com no lineals.

Exemple 1.1. Càlcul d'una expressió lineal

Expressió Matemàtica

$$\frac{(1 + 2 * 3 + 4^5 - 6)}{(7 * 8)} \simeq 18.3036$$

Expressió en Matlab

```
>> ( 1 + 2*3 + 4^5 - 6 ) / ( 7 * 8 )
```

```
ans =  
18.3036
```

Tanmateix, realitza càlculs immediats de funcions no lineals, com per exemple les funcions trigonomètriques, com les calculadores científiques.

Càlcul d'una expressió no lineal **Exemple 1.2.***Expressió Matemàtica*

$$\left(\sin\left(\frac{\pi}{3}\right) + 2\cos\pi - \ln(e^2) + \sqrt{10} \right) * \log_2(4) \simeq 0.0566$$

Expressió en Matlab

```
>> ( sin(pi/3) + 2*cos(pi) - log(exp(2)) + sqrt(10) ) * log2(4)

ans =
    0.0566
```

1.3.2 Definició i ús de variables

Matlab permet definir variables, a les quals se'ls pot modificar el valor amb total llibertat, i usar-les a la nostra conveniència.

Definició i ús d'una variable anomenada x **Exemple 1.3.***Expressió Matemàtica*

$$x^2 + x - 3 = \begin{cases} x = 3 & \rightarrow \text{resultat} = 9 \\ x = 5 & \rightarrow \text{resultat} = 27 \end{cases}$$

Expressió en Matlab

```
>> x = 3; % amb ';' al final de comanda, no surt el resultat.
>> x^2 + x - 3

ans =
     9

>> x = 5;
>> var = x^2 + x - 3 % assignem a 'var' el resultat amb x=5

var =
    27
```

Els nombres complexos també tenen cabuda a Matlab. El programa usa les variables ja definides i i j amb el valor $\sqrt{-1}$.

★ **Nota:** És important notar que tota variable definida a Matlab és subjecte de ser modificat el seu valor. Per tant, si escrivim $i = 2$, i deixarà de valdre $\sqrt{-1}$ per valdre 2.

Vegem com Matlab proporciona recursos per, a partir d'un nombre complex, trobar les seves coordenades, tant rectangulars com polars.

Exemple 1.4. *Coordenades rectangulars i polars d'un nombre $comp \in \mathbb{C}$*

Expressió Matemàtica

$$comp \in \mathbb{C} = 1 - 2i \equiv [2.2361_{-1.1071rad} = 2.2361_{-63.4349^\circ}]$$

Expressió en Matlab

```
>> comp = 1-2i

comp =
    1.0000 - 2.0000i

>> modC = abs(comp), angR = angle(comp), angG = angR*180/pi

modC =
    2.2361

angR =
   -1.1071

angG =

   -63.4349

>> % les comandes es poden separar en una mateixa linia amb comes
>> r1 = modC*cos(angR), real(comp), im = modC*sin(angR), imag(comp)

r1 =
    1.0000

ans =
     1

im =
    -2

ans =
    -2
```

Com s'ha vist, amb **abs** trobem el mòdul d'un nombre complex. A més, la funció serveix per trobar el valor absolut d'un nombre: **abs**(-3) = 3

1.4 Vectors i matrius

1.4.1 Vectors. Polinomis com a vectors

A l'entorn matemàtic de `Matlab`, els vectors a \mathbb{R}^n s'expressen com un array de n elements, separats per espais o per comes:

$$(v_1, v_2, \dots, v_n) \in \mathbb{R}^n \implies \begin{cases} [v_1 \ v_2 \ \dots \ v_n] \\ [v_1, v_2, \dots, v_n] \end{cases}$$

Amb ells, es poden realitzar les operacions típiques que es fan amb els vectors: sumes i multiplicacions de vectors amb constants, i sumes i multiplicacions entre vectors; o el producte vectorial a \mathbb{R}^3 .

Operacions de vectors amb constants Exemple 1.5.

Expressió Matemàtica

$$vec = (1, 2, 3, 4, 5) \in \mathbb{R}^5 \implies \begin{cases} vecp2 \equiv vec + 2 & = (3, 4, 5, 6, 7) \\ vecx2 \equiv vec \cdot 2 & = (2, 4, 6, 8, 10) \end{cases}$$

Expressió en Matlab

```
>> vec = [ 1 2 3 4 5 ]

vec =
     1     2     3     4     5

>> vecp2 = vec + 2, vecx2 = vec * 2

vecp2 =
     3     4     5     6     7

vecx2 =
     2     4     6     8    10
```

Operacions entre vectors Exemple 1.6.

Expressió Matemàtica

(seguint amb els vectors de l'exemple anterior:)

$$\begin{aligned} sumaVecs &\rightarrow vecp2 + vecx2 &= (5, 8, 11, 14, 17) \\ prodEsc &\rightarrow \langle vecp2, vecx2 \rangle &= 170 \\ prodVect &\rightarrow (1, 2, 3) \times (1, 0, 1) &= (2, 2, -2) \end{aligned}$$

Expressió en Matlab

```
>> sumaVecs = vecp2 + vecx2

sumaVecs =
     5     8    11    14    17

>> prodEsc = dot( vecp2, vecx2 )

prodEsc =
    170

>> prodVect = cross( [1 2 3], [1 0 1] )

prodVect =
     2     2    -2
```

Matlab, a part, ofereix una interpretació alternativa als vectors: els **polinomis**. Un polinomi de grau n es pot expressar com un vector de n elements:

$$k_n x^n + k_{n-1} x^{n-1} + \dots + k_0 \implies [k_n \ k_{n-1} \ \dots \ k_0]$$

Al ser expressats com a vectors (de fet, per Matlab ho són), els polinomis poden rebre totes les operacions descrites fins ara pels vectors (per exemple, a Matlab, sumar i restar vectors equival a sumar i restar polinomis).

I a més, té operacions més pròpies dels polinomis, com la multiplicació i divisió de polinomis, o trobar les seves arrels.

Exemple 1.7. *Definició i ús de polinomis**Expressió Matemàtica*

$$\begin{aligned} \text{arrels del polinomi } x^2 - 1 = 0 &\rightarrow x = \begin{cases} +1 \\ -1 \end{cases} \\ \left. \begin{aligned} polyA &= 2x^2 + 3x + 1 \\ polyB &= 1x^2 + 2x + 3 \end{aligned} \right\} \implies \begin{cases} polyA \times polyB &= 2x^4 + 7x^3 + 13x^2 + 11x + 3 \\ polyA / polyB &= \underbrace{\begin{cases} q = 2 \\ r = 0x^2 - x - 5 \end{cases}}_{q \times polyB + r = polyA} \end{cases} \end{aligned}$$

Expressió en Matlab

```

>> polinomi = [ 1 0 -1 ];
>> arrels = roots( polinomi )           % arrels del polinomi

arrels =
    -1
     1

>> polyA = [ 2 3 1 ];
>> polyB = [ 1 2 3 ];
>> polyMult = conv( polyA, polyB )     % multiplicacio (convolucio) de polinomis

polyMult =
     2     7    13    11     3

>> [q, r] = deconv( polyA, polyB )     % divisio (deconvolucio) de polinomis

q =
     2

r =
     0    -1    -5

```

1.4.2 Matrius. Operacions matricials bàsiques

Les matrius $\mathcal{M}_{m \times n} \in \mathbb{R}$ són, de fet, l'estructura de dades fonamental de Matlab. L'entorn matemàtic d'aquest programa està preparat per l'eficiència en càlculs matricials, i sempre que es pugui extrapolar un conjunt de dades a una matriu, és recomanable fer-ho.

Les matrius, per la forma d'expressar-les en Matlab, es podrien entendre com un conjunt de vectors, posats en fila, separades per punts i coma ';'. Així, una matriu es declararia seguint aquesta sintaxi:

$$\begin{pmatrix} 3 & 6 \\ 2 & 4 \end{pmatrix} \implies [3, 6; 2, 4]$$

Òbviament, Matlab proporciona de per si funcions que calculen propietats característiques bàsiques de les matrius, tals com transposades i inverses de matrius, o els seus determinants i rangs; seguidament proporcionarem exemples d'ús d'elles.

★ **Nota:** És important notar que, per la seva naturalesa, un vector de n elements, tal i com l'hem definit a la secció 1.4.1, es pot considerar una matriu $\mathcal{M}_{1 \times n} \in \mathbb{R}$. Per tant, és lògic pensar que tota funció que es pugui aplicar a una matriu NO quadrada es podrà aplicar a un vector.

Exemple 1.8. Propietats bàsiques de les matrius

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix} \Rightarrow \begin{cases} M^t = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 0 \end{pmatrix} & \det(M) = 27 \\ M^{-1} = \begin{pmatrix} -1.\hat{7} & 0.\hat{8} & -0.\hat{1} \\ 1.\hat{5} & -0.\hat{7} & 0.\hat{2} \\ -0.\hat{1} & 0.\hat{2} & -0.\hat{1} \end{pmatrix} & r(M) = 3 \end{cases}$$

Expressió Matemàtica

Expressió en Matlab

```
>> M = [ 1 2 3;
         4 5 6;
         7 8 0 ]

M =
     1     2     3
     4     5     6
     7     8     0

>> detM = det( M ), rangM = rank( M )

detM =
     27

rangM =
     3

>> transpM = M'

transpM =
     1     4     7
     2     5     8
     3     6     0

>> invM = inv( M )

invM =
    -1.7778    0.8889   -0.1111
     1.5556   -0.7778    0.2222
    -0.1111    0.2222   -0.1111
```

1.4.3 Rang de valors. Construcció de polinomis seqüencials i obtenció de subestructures matricials

Un recurs molt bo que ofereix `Matlab` és la possibilitat d'assignar rangs de valors a una variable. Això és molt útil quan es vol, per exemple, representar una funció $y = f(x)$, i volem que x tingui uns valors inicial i final concret amb un punt de pas determinats.

Aquest punt de pas és únicament per qüestions computacionals. Un ordinador ha de disposar d'un nombre tangible, no és vàlid dir-li que consideri "tots els valors entre x_0 i x_1 ", li hem de dir la resolució de les dades.

La sintaxi per generar un rang de valors és la següent:

$$\underbrace{x \in [3, 10]}_{\text{amb resolució de } 0.01} \implies x = 3 : 0.01 : 10$$

El valor retornat a la variable x seria, en aquest cas, un vector (o un polinomi) amb tots els valors entre 3 i 10 amb punt de pas 0.01:

$$x = 3.00, 3.01, 3.02, \dots, 10.00$$

Successió dels nombres parells fins a 20 **Exemple 1.9.**

Expressió Matemàtica

$$(2k)_{k=0,\dots,10} = 0, 2, 4, 6, \dots, 20$$

Expressió en Matlab

```
>> parellsFins20 = 0:2:20
parellsFins20 =
    0     2     4     6     8    10    12    14    16    18    20
```

L'ús de rangs de valors va més enllà d'això, però. Per obtenir **subestructures matricials** d'una matriu són imprescindibles els rangs de valors.

La forma d'obtenir l'element $a_{i,j}$ d'una matriu A és $A(i, j)$. Però si el que es vol es obtenir un conjunt d'elements de la matriu, el que s'ha d'usar és, en comptes de i i j , un rang de valors.

Obtenció de subestructures matricials

Exemple 1

Expressió Matemàtica

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{pmatrix}$$

↓

$$a_{23} = 8, \quad \begin{pmatrix} a_{13} & a_{14} & a_{15} \\ a_{23} & a_{24} & a_{25} \\ a_{33} & a_{34} & a_{35} \end{pmatrix} = \begin{pmatrix} 3 & 4 & 5 \\ 8 & 9 & 10 \\ 13 & 14 & 15 \end{pmatrix}, \quad \begin{pmatrix} a_{32} & a_{31} \\ a_{42} & a_{41} \\ a_{12} & a_{11} \end{pmatrix} = \begin{pmatrix} 12 & 11 \\ 17 & 16 \\ 2 & 1 \end{pmatrix}$$

Expressió en Matlab

```
>> A = [ 1 2 3 4 5;
        6 7 8 9 10;
        11 12 13 14 15;
        16 17 18 19 20 ]
```

```
A =
     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15
    16    17    18    19    20
```

```
>> A(2,3)
```

```
ans =
     8
```

```
>> A( 1:3, 3:5 )
```

```
ans =
     3     4     5
     8     9    10
    13    14    15
```

```
>> A( [3 4 1], [2 1] )
```

```
ans =
    12    11
    17    16
     2     1
```

1.4.4 El producte matricial a Matlab

Com bé se sap, la forma del producte entre matrius (i per tant vectors) és un tant peculiar, i Matlab l'ofereix. No obstant, també posa a la nostra disposició un càlcul sense massa connotació matemàtica, el producte (i divisió) terme a terme.

Tipologia de productes matricials a Matlab **Exemple 1.11.**

Expressió Matemàtica

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \quad B = \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$\nexists A \times B \implies \exists A \times B^t = \mathcal{M}_{2 \times 3} \times \mathcal{M}_{3 \times 2} = \mathcal{M}_{2 \times 2} = \begin{pmatrix} 32 & 50 \\ 77 & 122 \end{pmatrix}$$

$$\exists A \cdot B \equiv \{a_{ij} \cdot b_{ij}, i = 1..m, j = 1..n\} = \begin{pmatrix} 4 & 10 & 18 \\ 28 & 40 & 54 \end{pmatrix}$$

Expressió en Matlab

```
>> A = [ 1 2 3;
        4 5 6 ];
>> B = [ 4 5 6;
        7 8 9 ];
>> A * B    % dim(A) <> dim(B) -> falla
??? Error using ==> *
Inner matrix dimensions must agree.

>> prodMatricial = A * B'

prodMatricial =

    32    50
    77   122

>> prodTermeATerme = A .* B

prodTermeATerme =

     4    10    18
    28    40    54
```


Capítol 2

Càlcul simbòlic amb `Matlab`

2.1 Introducció al càlcul simbòlic

El càlcul simbòlic és una utilitat que ofereix `Matlab` per poder realitzar càlculs de forma semblant a com es fa a la pissarra (es diu que opera com un *manipulador algebraic*)

Aquest és un recurs molt potent, i força útil per realitzar tasques i càlculs tals com:

- *límits*, entenent-se com una expressió del tipus $\lim_{x \rightarrow a} g(x)$;
- *derivades*, com per exemple $f(x) = x^2 + 3 \implies f'(x) = 2x$;
- *primitives*, com amb : $f(x) = x^2 + 3 \implies \int f(x) = x^3/3 + 3x$;
- o simplement funcions on hi ha un cert paràmetre, com per exemple la funció $f(x) = e^{tx}$, $t \in \mathbb{R}$, on t seria una variable simbòlica, i donant valors a x s'obtidria $f(x)$ en funció del paràmetre t .

2.1.1 Com declarar una variable simbòlica?

Per `Matlab` totes les variables son *numèriques*, és a dir que prenen un valor determinat. Per poder declarar una variable com a *simbòlica* dins l'entorn de `Matlab` s'usa la següent crida:

```
var = sym( 'valorsimbolic' );
```

o de manera més compacta

```
syms var;
```

Des d'aquest moment, cada cop que aparegui la variable `var` es substituirà per `valorsimbolic`; cal notar que no és necessari que `valorsimbolic` sigui igual al nom de la variable `var`.

També es pot fer una declaració múltiple a partir de la instrucció

```
syms var1 var2 ... varN,
```

Vegem-ne un exemple per veure de forma pràctica els efectes:

Exemple 2.1. Càlcul numèric i simbòlic d'una expressió algebraica*Expressió Matemàtica*

$$f(x) = x + x + x \times x = 2x + x^2, x \in \mathbb{R} \implies f(3) = 15$$

Expressió en Matlab

```
>> x = 3; f = x + x + x * x
f =
    15

>> x = sym('x'); f = x + x + x * x
f =
2*x+x^2
```

2.1.2 Avaluació i representació d'una funció simbòlica

Una funció simbòlica i, de fet, tota expressió simbòlica, la podem avaluar en un punt o en tot un vector amb la comanda **subs**

Exemple 2.2. Avaluació d'una funció simbòlica*Expressió Matemàtica*

$$f(x) = x + x + x \times x = 2x + x^2, x \in \mathbb{R} \implies f(3) = 15$$

Expressió en Matlab

```
>> x = sym('x'); f = x + x + x * x
f =
2*x+x^2
>> subs(f,x,3)
ans =
15
>>%Substitucio multiple
>> v=1:4; y=subs(f,x,v)
y =
3      8      15      24
>>
```

★ **Nota:** Amb la comanda **pretty** la presentació del resultat és una mica més comprensible:

```
>> pretty(f) 2x + x2
```

Per representar una funció s'utilitza la comanda **ezplot**, la seva utilització és molt simple i s'obre una finestra gràfica en la que es dibuixa la funció.

```
>> f=acos(x);
>> ezplot(f);
```

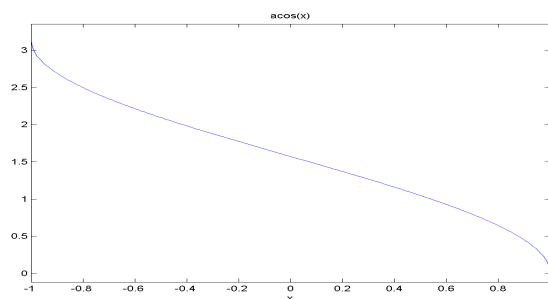


Figura 2.1: Gràfica de la funció $\arccos(x)$

2.2 Límits de funcions d'una variable

Matlab incorpora la possibilitat de calcular límits de funcions. Per això incorpora la comanda **limit**

```
valor = limit ( func,      - funció a calcular
               varsimb,   - variable simbòlica respecte a la que es fa el límit
               a          - valor al que tendeix la variable
               )
```

Veiem un exemple:

Exemple 2.3. Càlcul del límit les dues primeres derivades de $\arccos(x)$

Expressió Matemàtica

$$\lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x$$

Expressió en Matlab

```
>> syms x n;
>> f=(1+x/n)^n;
>> valorLim= limit( f, n, inf )

valorLim =
exp(x)
```

★ **Nota:** Matlab també permet fer límits laterals per l'esquerra i per la dreta. N'hi ha prou en incorporar la paraula corresponent:

```
>> val=limit(1/x,x,0,'left')
>> val= -inf
```

2.3 Derivació de funcions d'una variable

Per derivar una funció d'una variable simbòlica només cal utilitzar la comanda **diff**, que fa derivades de la funció fins al grau desitjat:

```
derivada = diff ( func,      - funció a integrar
                 varsimb,   - variable simbòlica d'integració
                 n          - grau d'integració
                 )
```

★ **Nota:** Si no s'especifica cap paràmetre `n`, `Matlab` entendrà que la derivada és de grau 1.

Vegem un exemple on es calculen la primera i la segona derivada de la funció $\arccos(x)$.

Càlcul de les dues primeres derivades de $\arccos(x)$

Exemple 2.4.

Expressió Matemàtica

- Primera derivada: $f'(x) = -\frac{1}{\sqrt{1-x^2}}$
- Segona derivada: $f''(x) = -\frac{x}{(1-x^2)\sqrt{1-x^2}}$

Expressió en Matlab

```
>> syms x;
>> f = acos(x);
>> d1f = diff(f, x, 1)

d1f =
-1/(1-x^2)^(1/2)

>> d2f = diff(f, x, 2) % equival a d2f = diff(d1f, x, 1)

d2f =
-1/(1-x^2)^(3/2)*x
```

2.4 Integració de funcions d'una variable

A l'igual que passava amb la derivació, `Matlab` també ofereix una comanda `int` amb la que es permet calcular la primitiva d'una funció respecte una variable simbòlica.

La seva sintaxi és la següent:

```
primitiva = int ( func, - funció a integrar
                 varsimb - variable simbòlica d'integració
                 )
```

on `primitiva` serà la funció que resulta d'integrar `func` respecte la variable `varsimb`:

$$\text{primitiva} = \int \text{func} \, d(\text{varsimb})$$

★ **Nota:** Si no s'especifica `varsimb Matlab` farà una cerca de la variable especificada com a simbòlica, i integrarà respecte ella. I si n'hi ha més d'una, ho farà respecte `x`.

Seguint amb l'exemple usat a la secció 2.3, veurem un exemple on s'integra la funció

$$f(x) = \arccos(x);$$

seguidament, i agafant el rang de valors $-1 < x < 1$, $x \in \mathbb{R}$, es poden veure a la figura 2.2 les gràfiques de cadascuna de les funcions, $f(x)$ i $F(x) = \int f(x)$.

Exemple 2.5. Primitivització de la funció $\arccos(x)$

Expressió Matemàtica

$$F(x) = \int \arccos(x) dx = x \arccos(x) - \sqrt{1-x^2}$$

Expressió en Matlab

```
>> syms x;
>> f = acos(x)

f =
acos(x)

>> F = int(f, x) %no cal posar la x

F =
x*acos(x) - (1-x^2)^(1/2)
```

A la figura 2.2 es pot veure una gràfica de les 4 funcions calculades per Matlab: $\arccos(x)$, la seva primitiva i les seves dues primeres derivades.

2.4.1 Aplicació de la regla de Barrow

La forma clàssica de calcular el valor numèric de l'àrea que forma una funció delimitada per l'eix d'abscisses i les rectes verticals $x = a$ i $x = b$, essent a i b els extrems d'integració, és la denominada **Regla de Barrow**:

$$F'(x) = f(x) \implies \int_a^b f(x) dx = F(b) - F(a), \text{ essent } f \text{ cont nua en } [a, b]$$

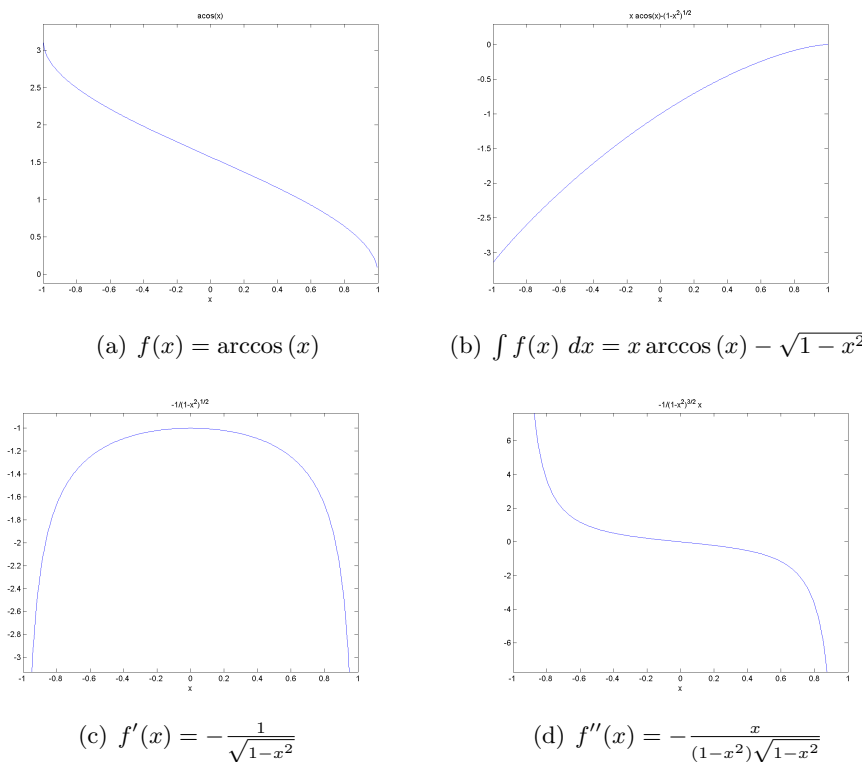


Figura 2.2: Gràfica de la funció $\arccos(x)$, la primitiva i les seves dues primeres derivades (per crear la gràfica s'utilitza la funció `ezplot(f)`)

Matlab ofereix una variació de la comanda `int` que fa ús de la Regla de Barrow i retorna el valor de la integral entre els extrems a i b :

```
valorArea = int ( func,      - funció a integrar
                  varsimb,   - variable simbòlica d'integració
                  a,         - extrem inferior d'integració
                  b          - extrem superior d'integració
                  )
```

★ **Nota:** +s sabut que no és necessari, i per tant, Matlab tampoc requereix que a sigui estrictament menor que b :

- $a < b \implies \int_a^b f(x) dx = I \in \mathbb{R}$.
- $a = b \implies \int_a^b f(x) dx = 0$.
- $a > b \implies \int_a^b f(x) dx = -I \in \mathbb{R}$.

Exemple 2.6. Aplicació de la Regla de Barrow a $\int_{-1/2}^{+1/2} \arccos(x) dx$

Expressió Matemàtica

$$\int_{-1/2}^{+1/2} \arccos(x) dx = \left[x \arccos(x) - \sqrt{1-x^2} \right]_{-1/2}^{+1/2} =$$

$$\left[\underbrace{\left(\frac{1}{2}\right) \arccos\left(\frac{1}{2}\right)}_{=\frac{\pi}{3}} - \underbrace{\sqrt{1-\left(\frac{1}{2}\right)^2}}_{=\sqrt{\frac{3}{4}}} \right] - \left[\underbrace{\left(-\frac{1}{2}\right) \arccos\left(-\frac{1}{2}\right)}_{=\frac{2\pi}{3}} - \underbrace{\sqrt{1-\left(-\frac{1}{2}\right)^2}}_{=\sqrt{\frac{3}{4}}} \right]$$

$$= \left[\left(\frac{\pi}{6}\right) - \sqrt{\frac{3}{4}} \right] - \left[-\left(\frac{\pi}{3}\right) - \sqrt{\frac{3}{4}} \right] = \left(\frac{\pi}{6}\right) + \left(\frac{\pi}{3}\right) - \sqrt{\frac{3}{4}} + \sqrt{\frac{3}{4}} = \boxed{\frac{\pi}{2}}$$

Expressió en Matlab

```
>> syms x;
>> f = acos(x);
>> valorInt = int(f, x, -0.5, +0.5)

valorInt =
1/2*pi

>> subs(valorInt)

ans =
1.57079632679490
```

Part II

Pràctiques de Laboratori

Capítol 3

Desenvolupament de Taylor d'una funció

Veiem com aplicació pràctica com es pot fer el desenvolupament de Taylor d'una funció real de variable real $f : \mathbb{R} \rightarrow \mathbb{R}$ al voltant del punt 0 i fins a un grau determinat.

Recordem que es defineix el desenvolupament de Taylor d'una funció $f(x)$ al voltant del punt a i fins a ordre $n + 1$ com

$$T_a(f) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x - a)^n.$$

En el cas de que el desenvolupament es faci en l'origen, és a dir $a = 0$, el desenvolupament de Taylor també s'anomena desenvolupament de McLaurin.

3.1 Càlcul del desenvolupament de Taylor

Com exercici del que s'ha introduït en el capítol anterior podem construir el polinomi de Taylor fins a un grau petit d'una determinada funció i veurem com en augmentar el grau de l'aproximació la gràfica de la funció i la gràfica del seu desenvolupament son cada vegada més semblants. Ho farem fins a grau 3 i es deixa com exercici fer-ho fins a grau 5.

Obs. Matlab incorpora la comanda **factorial** per calcular el factorial d'un nombre natural.

Càlcul del desenvolupament de Taylor fins a ordre 4 de la funció
 $f(x) = \sin(x)$

Exemple 3.1.

Expressió Matemàtica

$$f(x) = \sin(x) \Rightarrow T_4(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 = x - \frac{1}{6}x^3$$

Expressió en Matlab

```

>> syms x
>> f=sin(x);
>> d1=diff(f)
d1 =
cos(x)
>> d2=diff(f,2)
d2 =
-sin(x)
>> d3=diff(f,3)
d3 =
-cos(x)
>> T3=subs(f,0)+subs(d1,0)*x+subs(d2,0)*x*x/factorial(2)
+subs(d3,0)*x*x*x/factorial(3)
T3 =
x-1/6*x^3

```

3.2 Taylor directament per Matlab

Matlab incorpora una comanda per calcular directament el desenvolupament de Taylor en el punt zero **taylor** de manera que l'exemple anterior es pot simplificar i només utilitzar la comanda

\mathcal{M} – file 3.1.:
taylor1.m

Taylor directament de matlab

Especificació del M-file

Codi del M-file

```

syms x;
f=sin(x);
>> T4=taylor(f,4)
T4 =
x-1/6*x^3

```

Una bona manera d'entendre com el desenvolupament de Taylor d'una funció l'aproxima al voltant del punt en el que es fa el desenvolupament podria ser fer la gràfica de la funció i les dels seus desenvolupaments amb diferents ordres. Així, en el següent exemple veiem com la funció $f(x) = \sin(x)$ és aproximada pels seus desenvolupaments.

Càlcul del desenvolupament de Taylor fins a ordre 4 de la funció

Exemple 3.2. $f(x) = \sin(x)$

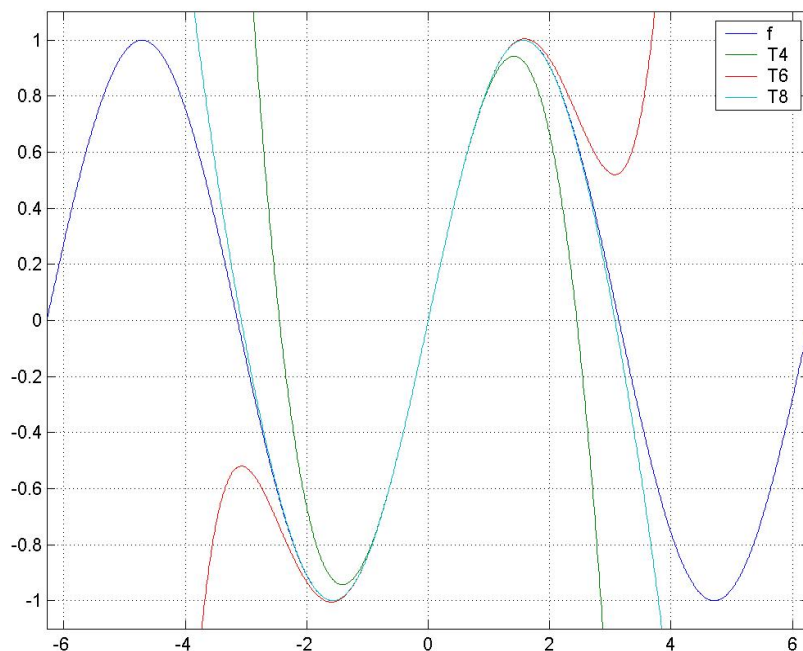


Figura 3.1: Gràfiques de la funció $\sin(x)$ i els seus desenvolupaments d'ordres 4, 6 i 8.

Expressió Matemàtica

$$\begin{aligned} f(x) &= \sin(x) \\ T_4(x) &= x - \frac{x^3}{3!} \\ T_6(x) &= x - \frac{x^3}{3!} + \frac{x^5}{5!} \\ T_8(x) &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \end{aligned} \tag{3.1}$$

Expressió en Matlab

```

>> syms x
>> f=sin(x);
>> T4=taylor(f,4)
T4 =
x-1/6*x^3

>> T6=taylor(f,6)
T6 =
x-1/6*x^3+1/120*x^5

>> ezplot(f);
>> grid;
>> hold on; %per mantenir la mateixa finestra grafica activada
>> ezplot(T4);
>> axis([-2*pi 2*pi -1.1 1.1])
>> ezplot(T6);
>> axis([-2*pi 2*pi -1.1 1.1])

```

En la figura 3.1 es pot apreciar com les gràfiques de la funció $f(x) = \sin(x)$ i les dels seus desenvolupaments a ordre 4,6 i 8 són cada vegada més semblants i com coincideixen al voltant de l'origen.

3.3 Taylor en el càlcul de límits

El desenvolupament de Taylor d'una funció també es pot utilitzar en el càlcul de límits de funcions per mitjà dels infinitèsims. De fet, l'infinitèsim d'una funció no és res més que el desenvolupament de Taylor fins a un ordre determinat (en molts casos ordre 1 o 2).

Veiem per exemple el càlcul del límit corresponent al problema 20 del tema 8 de la llista d'exercicis de l'assignatura de Càlcul 1.

En aquests exemples s'han d'aproximar les funcions pels seus desenvolupaments de Taylor d'ordre 3 o 4 per tal d'obtenir una substitució vàlida i que resolgui la indeterminació en el límit.

\mathfrak{M} – file 3.2.: Exemple 1 de càlcul del límits per infinitèsims
Taylor2.m

Especificació del M-file

$$\lim_{x \rightarrow 0} \frac{(x + \sin x)^2 + \sin^3 x}{\sin x^2 + \tan^2 x \arctan x}$$

Codi del M-file

```

>> syms x
>> num=(x+sin(x))^2+(sin(x))^3
>> den=sin(x^2)+tan(x)^2*atan(x)
>> f=num/den
f =
((x+sin(x))^2+sin(x)^3)/(sin(x^2)+tan(x)^2*atan(x))
>> taylor(num,4)
ans = 4*x^2+x^3
>> taylor(den,4)
ans = x^2+x^3

```

El límit final seria:

$$\lim_{x \rightarrow 0} \frac{(x + \sin x)^2 + \sin^3 x}{\sin x^2 + \tan^2 x \arctan x} = \lim_{x \rightarrow 0} \frac{4x^2 + x^3}{x^2 + x^3} = 4$$

Exemple 2 de càlcul del límits per infinitéssims

\mathfrak{M} – file 3.3.:
Taylor3.m

Especificació del M-file

$$\lim_{x \rightarrow 0} \frac{\sin^3 x - \sin(x^3)}{\ln(1 + x^5)}$$

Codi del M-file

```

>> syms x
>> num=(sin(x))^3-sin(x^3)
>> den=log(1+x^5)
>> f=num/den
f =
((sin(x))^3-sin(x^3))/log(1+x^5)
>> taylor(num,8)
ans = -1/2*x^5+13/120*x^7
>> taylor(den,8)
ans = x^5

```

El límit final seria:

$$\lim_{x \rightarrow 0} \frac{\sin^3 x - \sin(x^3)}{\ln(1 + x^5)} = \lim_{x \rightarrow 0} \frac{-\frac{1}{2}x^5 + \frac{13}{120}x^7}{x^5} = -\frac{1}{2}$$