Curs pràctic de Maple

Pràctica 5

5 Funcions i eines bàsiques de programació

5.1 Definir i treballar amb funcions

En aquesta secció aprendreu com definir una funció f(x) en Maple. La resta de la secció tracta de com avaluar funcions, resoldre equacions on hi ha funcions, i de com fer gràfiques de funcions.

5.1.1 Definir i esborrar una funció en Maple

Per a distingir entre una funció i una expressió, Maple necessita una notació especial quan es defineix una funció. Per exemple, la funció f(x) = cos($\pi\, x$) + 3 s'introdueix en Maple com:

```
> f:=x->cos(Pi*x)+3;
```

Preneu nota de la sintaxi que utilitzem. És **absolutament necessari** introduir la fletxa " ->" construïda amb el signe "menys" i un símbol "més gran que". **Maple no definirà una funció** si introduïu f(x) := cos($\pi\, x$) + 3.

A continuació podeu veure una comparació entre una expressió i una funció. Fixeu-vos amb la diferència de la sintaxi i en com dóna el resultat Maple per a cada un dels dos casos.

```
> y:=(x + 2)/(x^3 + 5*x + 2);
```

```
> f:=x->(x + 2)/(x^3 + 5*x + 2);
```

Les funcions necessiten sempre una fletxa quan s'han d'introduir; en el resultat que dóna Maple també hi ha d'haver una fletxa. Verifiqueu sempre que en el resultat hi ha la fletxa per a confirmar que heu definit una funció.

Exercici 5.1

Definiu la funció h(x) = $x^3 \sin(2\, x + 1)$.

Quan ja heu definit una funció, Maple recorda aquesta funció durant tota la sessió de treball. Si voleu substituir la funció per una nova definició, simplement reescriviu la definició. Per exemple, si voleu substituir la funció anterior f(x) per ln(cos(5 x)), escriviu:

```
> f:=x->ln(cos(5*x));
```

Podeu confirmar el valor actual de la funció f(x):

```
> f(x);
```

Si voleu esborrar la funció f(x) sense definir-la de nou, escriviu:

```
> f:='f';
```

Sempre és una bona idea esborrar les funcions que tingueu quan comenceu un problema nou. De forma alternativa també podeu utilitzar la comanda `restart` per a netejar tot el que hi hagi en la memòria de la sessió.

5.1.2 Avaluar una funció

Quan heu definit una funció, podeu avaluar-la per a diferents valors o per a expressions literals utilitzant la notació funcional. Abans de definir la funció netegem el valor de f.

```
> f:='f';
> f:=x->3*x+x^2;
> f(-1);
> f(2+sqrt(5));
> evalf(f(2+sqrt(5)));
> f(x+4);
> simplify(%);
> (f(x+h)-f(x))/h;
> simplify(%);
```

Si intervenen més d'una funció, la composició de funcions és fàcil de fer.

```
> g:=x->cos(x)+1;
> f(g(Pi/3));
> j:=x->g(f(x));
> j(x);
```

Exercici 5.2

Definiu la funció $\mathrm{s}(t) = \frac{3+t^2}{\sqrt{3\,t+1}}$ i feu que Maple calculi $\mathrm{s}(2)$, $\mathrm{s}(t-3)$ i $\mathrm{s}(t) - \mathrm{s}(3)$ simplificant els resultats. No oblideu la notació de la fletxa!

5.1.3 Resoldre equacions en les que intervenen funcions

Quan heu definit una funció, podeu resoldre equacions amb aquesta funció de manera exacta o aproximada:

```
> g:='g';
> g:=t->t^3-6*t^2+6*t+8;
> solve(g(t)=0,t);
> fsolve(g(t)=0,t);
```

5.1.4 Gràfics de funcions

La comanda `plot` funciona igualment per a funcions:

```
> h:='h';   y:='y';  x:='x';
> h:=x->x*exp(-x);
> plot(h(x),x=-1..4,y=-2..1);
```

Es poden dibuixar gràfics de diferents funcions simultàniament de la mateixa manera que es fa per a les expressions.
Considereu la funció $\mathrm{f}(x) = \frac{2}{x^2+1}$. A continuació fem el gràfic d'aquesta funció i dels seus desplaçaments horitzontals $\mathrm{f}(x+1)$, $\mathrm{f}(x-3)$ i $\mathrm{f}(x-6)$. Podeu identificar cada un d'aquests gràfics?

```
> f:=x->2/(x^2+1);
> plot([f(x),f(x+1),f(x-3),f(x-6)],x=-5..10,y=-1..3);
```

Exercici 5.3

Definiu la funció $\mathrm{f}(x) = 2\,x - \left|x^2 - 5\right|$ després responeu a les qüestions següents:
a) Quin és el valor de f(6.5)
b) Simplifiqueu el valor de $\mathrm{f}(z-4)$, on z és una variable.
c) Feu el gràfic de $\mathrm{f}(x)$.
d) Determineu els valors de x per als que $\mathrm{f}(x) = 0$.

Exercici 5.4

Definiu les funcions $\mathrm{g}(x) = 5\,e^{x/2}$ i $\mathrm{h}(x) = x + 10$ i feu el següent:
a) Dibuixeu un gràfic que mostri les dues funcions $\mathrm{g}(x)$ i $\mathrm{h}(x)$. Experimenteu amb diferents valors per al domini i per al recorregut.
b) Feu una estimació del punt d'intersecció d'aquests dos gràfics utilitzant el botó esquerra del ratolí.
c) Utilitzeu la comanda `fsolve( )` per a resoldre l'equació $\mathrm{g}(x) = \mathrm{h}(x)$. Com es relaciona la solució d'aquesta equació amb el que heu obtingut en l'apartat (b)?

Exercici 5.5

Definiu la funció $\mathrm{k}(x) = x + 3\sin(2\,x)$, després feu el següent:
a) Dibuixeu el gràfic d'aquesta funció en el domini $[-1, 8]$.
b) Modifiqueu el dibuix de l'apartat (a) per tal que inclogui la línia horitzontal $y = 4$. Utilitzeu aquest nou gràfic per a estimar el nombre i els valors aproximats dels x tals que $\mathrm{k}(x) = 4$.
c) Quina funció única hauríeu de dibuixar per a obtenir la mateixa informació que en l'apartat (b)?
d) Utilitzeu la comanda de Maple `fsolve( )` per a aproximar totes les solucions de l'equació $\mathrm{k}(x) = 4$.

5.2 Programació: lògica, iteracions i procediments

Alguns cops el Maple no tindrà les comandes que necessitem per a algun treball concret. En aquests casos necessitarem definir les nostres pròpies comandes i necessitarem saber com incloure funcions lògiques, iteracions i procediments. Comencem amb funcions lògiques:

5.2.1 Lògica: `if-then-else-end if`

La comanda bàsica és `if` i podem veure el funcionament al següent exemple:
Exemple 5.1 Suposem que volem definir una funció `f(x)` que valgui -1 si $x < 0$, 0 si $x = 0$ i 1 si $x > 0$. Podríem fer-ho de la següent manera:

```
> restart;
> f:=x-> if x<0 then -1 elif x=0 then 0 else 1 end if;
```

comproveu que funciona executant:

```
> f(-.5);f(0);f(.5);
```

Nota: el que hem definit ja estava definit al Maple amb el nom de `signum( )`.
El que hem fet és definir la funció `f` on hauríem de traduir `if` per "*si*", `elif` per "*altrament si*" i `end if` per "*hem acabat*". Llavors podem pensar la comanda `if` com una comanda amb tres elements:

- El primer que hem de veure és la sintaxi de la funció. Hem d'entrar els arguments `if ... then`, `elif ... then`, `else`, i `end if` sempre i en aquest ordre.
- El segon és la introducció de condicions lògiques. Això són expressions com $x > 0$, ... i que en Maple s'han d'introduir com:

`<`	més petit que,
`<=`	més petit o igual que,
`>`	més gran que,
`>=`	més gran o igual que,
`=`	igual,
`<>`	diferent.

  Quan Maple es troba amb alguna d'aquestes expressions ho avalua com una "*expressió Booleana*", o sigui, com una expressió que pot ser "*certa*" o "*falsa*". Això també es pot avaluar mitjançant la funció `evalb( )`:

```
> evalb(7>5);evalb(3=4);evalb(3<>4);
```

  Podem fer construccions més complicades, combinant amb les comandes lògiques `and`, `or`, `xor` i `not`. Podem consultar la ajuda per a més detalls: `?boolean`.

- El tercer element són les comandes que s'han d'executar a cada un dels casos. En l'exemple tant sols havíem d'introduir -1, 0 i 1, però es poden introduir expressions més complexes: volem definir una funció `g` que prengui els següents valors:

```
> g:=x-> if x<0 then 1/(1+x^2) else cos(x) end if;
```

  Podem comprovar que funciona

```
> g(-1.5);g(2.5);
```

Tot i això, què passa quan fem:

```
> g(Pi);
```

El missatge d'error diu que no ha pogut avaluar la condició booleana que hem introduït. Ens a passat degut a que `Pi` no és exactament un número. Obtenim exactament el mateix error quan avaluem `g` a una indeterminada a:

```
> g(a);
```

Una manera de arreglar el problema seria utilitzant la comanda `evalf()` i avaluar `g` en un valor aproximat de `Pi`.

```
> g(evalf(Pi));
```

Intentem ara dibuixar la funció `g` que hem definit:

```
> plot(g(x),x=-10..10);
```

Tornem a obtenir errors en les expressions booleanes. El problema és que quan substitueix x a la comanda `g` encara no és un número i per això retorna l'error. En aquest cas el que podem fer és introduir la "*notació d'operadors*", o sigui, no escriure la variable `x`:

```
> plot(g,-10..10);
```

Però si volem utilitzar la funció `g` per a definir altres funcions, per exemple, si volem calcular la seva derivada, també tindrem problemes:

```
> diff(g(x),x);
```

La funció `if` és molt útil en segons quins contexts, però no és la més apropiada per a definir funcions a trossos. Per a fer això és més versàtil la funció `piecewise( )`. Podem redefinir la funció `g` com:

```
> g:=x->piecewise(x<0, 1/(1+x^2),x>=0, cos(x));
```

Ara podem treballar amb `g` com amb qualsevol altre funció, per exemple:

```
> plot(g(x),x=-5..5);
```

i calcular la seva derivada:

```
> diff(g(x),x);
```

Per tant per a definir funcions a trossos utilitzarem la funció `piecewise`, mentre que reservarem `if-then-elif-end if` per a utilitzar-la dins de iteracions, ...

Exercici 5.6

Utilitzeu la funció `piecewise` per a definir amb Maple la funció que avalua la següent funció i feu la gràfica a l'interval $(-4, 4)$.

$$f(t) = \begin{cases} \frac{(2+t)^3}{6} & \text{si } t \in [-2,-1) \\ \frac{-3t^3-6t^2+4}{6} & \text{si } t \in [-1,0) \\ \frac{3t^3-6t^2+4}{6} & \text{si } t \in [0,1) \\ \frac{-(t-2)^3}{6} & \text{si } t \in [1,2) \\ 0 & \text{si } t \notin [-2,2) \end{cases}$$

Segons com feu la definició haureu d'introduir rangs amb límit inferior i límit superior. Per a això hem d'introduir dues desigualtats, i això ho hem de fer mitjançant la comanda `and`.

5.2.2 Iteracions: `for`, `do`, `end do` i `while`

Moltes comandes a Maple contenen iteracions: per exemple, les comandes `sum`, `fsolve`, ... Tot i això algun cop necessitarem definir les nostres pròpies iteracions. Com a exemple, calculem la suma dels enters entre 1 i 100 elevats al quadrat:

```
> sum(i^2,i=1..100);
```

$$338350$$

En alguna part del procés Maple ha aplicat una iteració per a fer el càlcul. Fet "a mà" hauríem fet:

Iteració suma:

Decidim primer el nom de la variable, i la inicialitzem a zero.

```
> resp:=0;
```

Llavors cal introduir la iteració. Per a això utilitzem les comandes `for..from..by..to..while..do..end do`. Les comandes que s'han d'executar s'han d'introduir entre `do` i `end do`.

```
> for i from 1 to 100 do
  resp:=resp + i^2;
  end do;
```

Per a evitar la sortida de tots els passos, podem canviar el punt i coma (`;`) per uns dos punts (`:`) i demanar el valor de la variable `resp` al final del procediment.

```
> resp:=0;

> for i from 1 to 100 do
  resp:=resp + i^2;
  end do:
> resp;
```

Exercici 5.7

Calculeu la suma de tots els nombre senars entre 1 i 99 mitjançant la comanda `for` (consulteu l'ajuda per veure la opció `by` dins de la comanda `for`).

Més iteracions:

A vegades no coneixerem el nombre d'iteracions que volem fer. La informació que tindrem serà que hem d'iterar fins que alguna condició fixada es doni. Per exemple, què passa si apretem indefinidament la tecla "*cosinus*" de la vostra calculadora, començant pel valor 5? A l'exemple ho farem 20 vegades:

```
> x:=5.;

> for i from 1 to 20 do
  x:=cos(x);
  end do;
```

Podeu observar que sembla que la successió es va acostant a un valor determinat. De fet, podeu comprovar que aquest valor ha de complir l'equació $\cos(x) = x$. Tot i això, podeu veure que amb 20 cops no n'hi ha prou per a obtenir una successió que no canvia de valor (8 dígits correctes), per tant, el que volem fer és iterar el procediment fins que la diferència entre dos passos sigui inferior a un número fixat, per exemple, 10^{-8}.

Necessitem dues variables, una serà `x` i la segona serà `xold`, que en tot moment serà el valor de la iteració anterior (haurem de donar-li un valor inicial diferent del que donem a `x`).

```
> x:=5.;xold:=0;
```

Llavors hem d'iterar mentre es compleixi que $|\texttt{x} - \texttt{xold}| > 10^{-8}$. La comanda la podríem entrar com:

```
> for i from 1 while abs(x-xold)>1e-8 do
  tempx:=cos(x):
  xold:=x:
  x:=tempx:
  end do;
```

Observem que la "`i`" que hi ha dins del `for` no s'ha utilitzat al procediment. De fet en qualsevol iteració l'única part obligatòria és el `do..end do`. Les parts `for` i `while` són opcionals. Com a exemple podríem escriure:

```
> x:=0.; xold:=1;
  while abs(x-xold)>1e-8 do
  tempx:=cos(x):
  xold:=x:
  x:=tempx:
  end do;
```

5.2.3 Procediments

Quan volem definir una funció poder necessitarem incloure iteracions, condicions lògiques, altres funcions... i en general potser necessitarem dues o més instruccions de Maple encadenades per a fer els càlculs. També és bastant comú que ens aquests casos utilitzem variables que només tenen sentit dins el càlcul de la funció que estem definint i no a la resta del full de treball. Això és el que anomenarem **variables locals**, mentre que les que són vàlides a tot el full de treball les anomenarem **variables globals**.

Per a definir funcions que inclouen variables locals i globals hem d'utilitzar la funció `proc`.

Exemple 5.2 Mireu l'estructura de la següent definició:

```
> f:=proc(x,y)
  local a,b,z;
  global d;
  a:=3.;b:=17.;
  z:=a*b*x*y*d;
  end proc;
```

Podem veure com s'executa aquesta definició provant:

```
> f(5,5);
> d:=2.;
> f(5,5);
> d:='d';
> f(5,5);
> a;
> b;
> z;
```

Exemple 5.3 La funció següent avalua el cosinus d'un angle en graus:

```
> cosdeg:=proc(theta)
  local resultat,phi;
  phi:=theta*Pi/180;
  resultat:=cos(phi);
  end proc;
```

Proveu calculant el cosinus de 45 graus.

Si voleu veure els passos que segueix un procediment, ho podeu fer amb la funció `trace`. Aquesta funció permet "*activar*" per pantalla els passos que hi ha dins de la funció `proc`.

Exemple 5.4 Per a activar la funció `cosdeg` hem de fer:

```
> trace(cosdeg);
```

llavors l'execució fa:

```
> cosdeg(45);
```

Si volem desactivar aquesta la visualització podem fer:

```
> untrace(cosdeg);
```

5.2.4 Exercicis

Els següents exercicis són perquè practiqueu els diferents conceptes de "*programació*" que hem introduït, però també pretenen il·lustrar alguns conceptes matemàtics. No us oblideu d'interpretar els resultats que aneu obtenint.

Exercici 5.8

Considereu la funció:

$$\texttt{aprcos}(x,n) = \sum_{i=0}^{n} (-1)^i \frac{x^{2i}}{(2i)!}$$

on x és un paràmetre real i n és un enter positiu.

Dibuixeu en on sol gràfic les funcions `cos(x)`, `aprcos(x,1)`, `aprcos(x,2)` i `aprcos(x,4)` per a `x` entre -2π i 2π.

Exercici 5.9

Definiu una funció que depengui de tres paràmetres `aprox(funcio,x,iteracions)` que retorni una llista amb els valors `[x+1/i,funcio(x+1/i)]` per a `i=1..iteracions`.
Avalueu `aprox` a la funció $\frac{\sin(x)}{x}$, per a $x = 0$ amb 50 iteracions.
Modifiqueu la funció `aprox` per a que a més retorni un dibuix dels punts calculats.