Matemàtiques amb ordinador.
Curs pràctic de Maple

Presentació de Maple

Curs 2003–04

Índex

Part I
Presentació de Maple

Maple és un sistema de computació algebraica molt potent. En aquesta primera secció veurem com realitzar algunes de les funciones aritmètiques bàsiques. Com veureu, les comandes s'introdueixen després del símbol > que apareix per pantalla i s'acaben amb punt i coma. La línia s'executa quan premeu ⏎ i el resultat apareix al centre de la línia següent en color blau.

1 Càlculs Numèrics

En aquesta secció utilitzareu Maple per a fer alguns càlculs numèrics estàndard. L'habilitat de Maple per a donar resultats exactes a més de les aproximacions numèriques us donarà més opcions a l'hora de resoldre problemes.

1.1 Realitzant càlculs aritmètics exactes amb Maple

Utilitzar Maple per a realitzar càlculs numèrics és molt fàcil. Només s'ha d'introduir l'expressió numèrica i acabar la línia amb un **punt i coma (;)**. Prement ⏎ (retorn) executareu la línia i el resultat apareixerà en color blau en el centre de la pantalla.

Exemple 1.1

Introduïu els següents càlculs simples i premeu ⏎ al final de cada línia.

```
> 2+4;

> 12*34567890;

> 34+(-27);
```

Cada línia vermella és "viva" i es pot modificar en qualsevol moment.
Canvieu el "4" de la línia anterior per un "8" i premeu ⏎ .
Noteu com el resultat de color blau s'actualitza automàticament per a mostrar el nou resultat.

Exemple 1.2

Pel nostre exemple següent calculem 134^{39}.

```
> 134^39;
```

A diferència de la vostra calculadora, Maple us donarà la resposta exacte d'aquest problema, les 83 xifres que té.

Exemple 1.3

Maple pot fer càlculs amb fraccions sense convertir-les en expressions decimals:

```
> 3/5 + 5/9 + 7/12;
```

Exemple 1.4

Per a introduir l'arrel quadrada d'un nombre utilitzeu `sqrt( )` :

```
> sqrt(24);
```

Noteu que Maple ha simplificat $\sqrt{24}$ però que ha deixat el resultat en forma exacta. En el proper apartat aprendreu com obtenir una aproximació decimal per a aquest nombre.

Exemple 1.5

Maple té introduïdes totes les constants matemàtiques importants. Per a introduir π escriviu `Pi`. Fixeu-vos que es necessita un asterisc `*` per a indicar la multiplicació.

```
> 4*(3+Pi);
```

Una altra vegada Maple fa el càlcul però deixa el resultat en una forma exacta.

Exemple 1.6

A diferència de la vostra calculadora, Maple dóna la solució exacta quan s'aplica a les funcions trigonomètriques.

```
> sin(5*Pi/3);
> sec(Pi/4);
```

Per a obtenir l'invers del sinus d'un nombre utilitzeu la funció `arcsin( )`:

```
> arcsin(-1);
```

Si demaneu a Maple que calculi un valor *indeterminat* respondrà amb un missatge d'error:

```
> tan(Pi/2);
```

Exemple 1.7

Per a introduir la funció exponencial e^x en Maple escriviu: `exp(x)` .

```
> exp(x);
```

I per a obtenir el nombre e escriviu: `exp(1)` .

```
> exp(1);
```

Exemple 1.8

Per a introduir la funció valor absolut $|x|$ en Maple escriviu: `abs(x)`.
Noteu que Maple dóna la solució correcta i exacta per a la tercera línia ja que: $e - \pi < 0$

```
> abs(x);
> abs(-3);
> abs(exp(1)-Pi);
```

Exemple 1.9

Maple té moltes comandes per a realitzar càlculs específics amb nombres. Les podreu aprendre així que es vagin necessitant. Aquí presentem un últim exemple per ara: si teniu un nombre enter i voleu obtenir la seva descomposició en factors primers podeu utilitzar la comanda de Maple `ifactor( )`. Podeu practicar tot el que vulgueu canviant el nombre que hi ha aquí sota.

```
> ifactor(31722722304);
```

Exemple 1.10

Pot haver hi molts moments en els que vulgueu introduir més d'una comanda en la mateixa línia. Això es pot fer en Maple, només cal assegurar-se d'acabar **cada una** de les comandes amb un punt i coma (;). També ajudarà posar espais en blanc entre les comandes. Quan premeu  totes les expressions s'executaran i els resultats apareixeran, en ordre, en un únic camp de resultats.

```
> sin(Pi/3);   cos(Pi/3);   tan(Pi/3);
```

Exemple 1.11

Per a calcular i mostrar una successió de nombres utilitzeu la comanda `seq( )`. Aquí mostrem els 100 primers nombres naturals, els seus quadrats i els cent primers nombres imparells.

```
> seq(n,n=1..100);
> seq(k^2,k=1..100);
> seq(2*k+1,k=1..100);
```

1.2 Aproximacions numèriques utilitzant la comanda `evalf( )`

Recordeu que en l'apartat anterior hem demanat a Maple que fes la suma de tres fraccions i que ha donat com a resultat una altra fracció. Aquest tipus d'aritmètica exacta és molt útil però hi ha vegades que preferirem una resposta en forma d'expressió decimal. La comanda de Maple `evalf( )` realitza aquesta tasca per nosaltres.

Exemple 1.12

Compareu els resultats de les dues línies següents.

```
> 3/5+5/9+7/12;
> evalf(3/5+5/9+7/12);
```

Exemple 1.13

Donar un nom al resultat d'un determinat càlcul fa que sigui més fàcil utilitzar-lo en un càlcul posterior. Per assignar un nom utilitzem uns **dos punts** seguit del **signe igual** (és a dir `nom := resultat`). En la línia que ve a continuació hem assignat a la lletra `k` el resultat del càlcul anterior. Després apliquem `evalf( )` a `k`.

```
> k:=3/5+5/9+7/12;
> evalf(k);
```

Nota important de Maple: Maple distingeix entre majúscules i minúscules. Per tant Maple considera que k i K són variables diferents.

```
> k;
> K;
```

Per cert també es poden utilitzar paraules com a nom de variables.

```
> josep:=2^5;
> sqrt(josep);
```

Exemple 1.14

Si volem més o menys dígits de precisió en comptes dels 10 que s'utilitzen per defecte podem afegir un argument extra a la comanda `evalf( )` tal i com es mostra a continuació.

```
> w:=4*(3+Pi);
> evalf(w);
> evalf(w,4);
> evalf(w,45);
```

Exemple 1.15

Si introduïu nombres amb un punt decimal Maple donarà de forma automàtica un resultat decimal. Compareu els dos resultats que apareixen a continuació.

```
> sqrt(34);
> sqrt(34.0);
```

Aquí hi ha un altre exemple:

```
> 4-1/3;
> 4.0-1/3;
```

Exemple 1.16

Podem aplicar la comanda `evalf( )` a una successió de nombres: aquí sota generem primer les arrels quadrades exactes dels 10 primers nombres naturals, després apliquem `evalf( )` per a obtenir les aproximacions decimals. Recordeu la comanda per crear successions de números que hem après a l'Exemple 1.11.

```
> result:=seq(sqrt(k),k=1..10);
> evalf(result);
```

Drecera de Maple: referir-se de manera ràpida a l'últim resultat
Quan utilitzeu Maple hi ha moltes vegades que voldreu encadenar una sèrie de càlculs. En comptes de donar un nom a cada un dels resultats que aneu obtenint, podeu utilitzar el signe tant per cent (`%`) per referir-vos a **l'última expressió que Maple ha calculat.** Aquí hi ha alguns exemples de com funciona.

```
> 3/5+5/9+7/12;
> evalf(%);
> Pi;
> evalf(%);
> %+5;
```

Per a més informació de com fer servir aquest símbol mireu l'Exercici 1.4 que ve tot seguit.

Exercici 1.1

Utilitzeu Maple per a calcular el nombre 37^{43} .

Exercici 1.2

Calculeu $\sqrt{34}$ amb 18 xifres.

Exercici 1.3

Obteniu una aproximació numèrica per a l'expressió : $\dfrac{3+\pi}{7-\sqrt{13}}$

Exercici 1.4

El símbol tant per cent (`%`) és una drecera molt útil però pot donar en alguns casos resultats inesperats.
Aquí hi ha un exemple.
Primer executeu cada una de les tres línies següents. Heu de poder dir quin serà el resultat per endavant.

```
> 4+Pi;
> evalf(%);
> %+10;
```

Ara torneu enrera i torneu a executar l'última línia (i.e. `> %+10;`). Fixeu-vos que el resultat canvia de 17.14159265 a 27.14159265
Podeu explicar el perquè?

1.3 "Netejant" Variables

Des del moment en que definiu una variable, Maple recordarà quin és el seu valor durant tota la sessió de treball. Si voleu introduir un nou valor a la variable, només heu de fer la nova assignació. Per exemple cada una de les assignacions que hi ha aquí baix redefineix el valor de la variable `h`. (Nota: per a verificar el valor actual d'una variable només cal escriure el seu nom seguit d'un punt i coma).

```
> h;
> h:=56;
> h;
> h:=sqrt(Pi);
> h;
```

A vegades voldreu "netejar" una variable de la memòria per a poder utilitzar-la en una situació nova.
Aquí hi ha un exemple. Primer introduïm en `x` el valor 65.

```
> x:=65;
```

Ara suposem que comencem un problema nou i que volem introduir l'expressió algebraica genèrica (depenent d'una variable indetermeinada x) $x^2 - 4\,x + 7$ i assignar-li el nom `w`. Si només fem això, Maple substitueix automàticament el valor anterior de `x`.

```
> w:=x^2-4*x+7;
```

Per a fer que `x` sigui una variable genèrica un altre cop hem de "netejar" (és a dir esborrar la memòria de Maple) el nostre antic valor per a `x`. Això s'aconsegueix introduint

```
> x := 'x';
```

Fixeu-vos que aquí utilitzem una cometa (apòstrof).

Executeu les dues línies següents per a veure com funciona.

```
> x:='x';
> w:=x^2-4*x+7;
```

Netejant totes les variables de cop: la comanda `restart`.

La comanda `restart` neteja la memòria de Maple de totes les definicions que hagueu fet. És com si es comencés una nova sessió de Maple. Si heu de començar un problema totalment nou podeu utilitzar la comanda `restart` per assegurar que no queden definicions del vostre treball anterior. Abans d'executar la segona línia d'aquí sota, endevineu ràpidament el resultat.

```
> p:=4;
> p;  x;  h;
```

Probablement recordareu que `p` era 4 i que `x` s'havia reassignat a la variable genèrica `x`, però poder no haureu recordat que `h` s'havia assignat al valor $\sqrt{\pi}$. És per això que és una bona idea utilitzar `restart` per a eliminar totes les definicions d'un sol cop. (Si aneu seguint aquestes pràctiques veureu que començarem moltes seccions noves amb una comanda `restart`).

```
> restart;
> p;  x;  h;
```

2 Càlculs Algebraics

Maple és un "C.A.S" , això vol dir un **"Computer Algebra System" (que en català s'anomena normalment un *Manipulador Algebraic*)**. Això significa que Maple coneix totes les regles algebraiques que vosaltres sabeu. Al mateix temps que aneu progressant en el Càlcul, l'Àlgebra lineal i les altres especialitats matemàtiques veureu que Maple també conté les operacions essencials d'aquests temes introduïdes en el seu gran conjunt de comandes.

En aquesta secció aprendreu com introduir una expressió algebraica i a donar valors a les seves variables. Després aprendreu les comandes que us permetran expandir, factoritzar i simplificar expressions.

És recomanable iniciar les sessions executant la comanda `restart` que ja hem vist a la secció anterior, recordeu quin és el seu efecte?

```
> restart;
```

2.1 La comanda `subs( )`

La primera comanda que analitzarem ens permetrà substituir valors en les variables o paràmetres d'una expressió algebraica que haguem definit.

Exemple 2.1

Com a primer exemple comencem amb l'expressió `3*x^2+8` i assignem-li com a nom `W`.

```
> W:=3*x^2+8;
> W;
```

Ara suposem que volem substituir la variable `x` de l'expressió `3*x^2+8` pel valor 4. La forma més ràpida de fer-ho és utilitzar la comanda de Maple `subs( )`. Tot seguit veiem com fer-ho.

```
> subs(x=4,3*x^2+8);
```

De forma alternativa podem aplicar la comanda `subs( )` a `W`.

```
> subs(x=4,W);
```

Exemple 2.2

La comanda `subs( )` també funciona bé amb valors simbòlics. Per exemple, per a substituir x per $5+2u$ en l'expressió $3x^2+8$ executem la línia següent (en aquest cas posem l'etiqueta `M` al resultat).

```
> W:=3*x^2+8;
> M:=subs(x=5+2*u,W);
```

I ara per a fer que Maple "multipliqui" aquesta expressió utilitzem la comanda `expand( )` .

```
> expand(M);
```

Exemple 2.3

La comanda `subs( )` és molt versàtil. Podem utilitzar-la per a avaluar expressions en les que intervenen més d'una variable. Aquí substituïm la x per 7 i la y per 12 en l'expressió $U = \frac{2\,x^2}{5} + 3\,y$.

```
> U:=(2/5)*x^2+3*y;
> subs(x=7,y=12,U);
> evalf(%);
```

Exemple 2.4

També podem utilitzar la comanda `subs( )` per a substituir un valor en una equació. Això és el tipus de coses que cal fer per a verificar si un determinat valor "satisfà" l'equació. En les línies que venen a continuació substituirem diferents valors en l'equació $x^3 - 5\,x^2 + 7\,x - 12 = 0$. És algun d'aquests valors una solució de l'equació?

Noteu que utilitzem "`:=`" per assignar el nom i només "`=`" per a l'equació mateixa.

```
> eqn:=x^3-5*x^2+7*x-12=0;
> subs(x=3,eqn);
> subs(x=4,eqn);
> subs(x=5,eqn);
```

Exercici 2.1

Assigneu el nom k a l'expressió $x^2 + 4\,x - 3$. Un cop fet això dieu M a l'expressió $k^2 - 9$. Finalment feu que Maple calculi $3\,M + 6$.

Nota: per fer que Maple multipliqui les expressions utilitzeu la comanda `expand( )`. És a dir, introduïu

```
> expand(3*M+6);
```

Aprendreu més coses sobre la comanda `expand( )` en l'apartat següent.

Exercici 2.2

Desenvolupeu $(1 + x)^4$ utilitzant la comanda `expand( )`.

Exercici 2.3

Sigui $P = ax^3 + bx^2 + c\,x + d$. Calculeu P si $x = 0.01$, $a = -\dfrac{1}{5}$, $b = \dfrac{2}{5}$, $c = 0$, i $d = \dfrac{13}{15}$

Exercici 2.4

Utilitzeu la comanda `subs( )` per a verificar si algun dels nombres 1, 2 o 3 és una solució de l'equació $x^3 - 16x^2 + 51x - 36 = 0$.

2.2 La comanda `expand( )`

L'ús principal de la comanda `expand( )` és el de "fer les multiplicacions" en els productes d'expressions polinòmiques. També es pot utilitzar per a desenvolupar funcions trigonomètriques i altres tipus de funcions més generals.

Exemple 2.5

Utilitzeu la comanda `expand( )` per efectuar les multiplicacions en $(x+2)^2\,(3\,x-3)\,(x+5)$.

```
> k:=(x+2)^2*(3*x-3)*(x+5);
> expand(k);
```

Exemple 2.6

Maple aplica algunes de les identitats trigonomètriques més comuns per a desenvolupar $\sin(2\,x)$ i $\cos(2\,x)$.

```
> expand(sin(2*x));
> expand(cos(2*x));
```

Feu alguna prova desenvolupant el sinus i el cosinus d'alguns altres múltiples enters de x. Per exemple: $\sin(3\,x)$, $\cos(6\,x)$, etc.

Exemple 2.7

Finalment feu que Maple multipliqui l'expressió $x^{(\frac{1}{2})}\,(x^{(\frac{3}{2})} + x^{(-\frac{1}{2})})$

```
> h:=x^(1/2)*(x^(3/2)+x^(-1/2));
> expand(h);
```

Exercici 2.5

Desenvolupeu $(x+1)^n$ per a $n = 2, 3$ i 4.

2.3 La comanda `factor( )`

La comanda `factor` agrupa un polinomi en producte de factors util·lizant nombres enters i fraccions. Aprendrem el seu funcionament amb una serie d'exemples.

Exemple 2.8

Factoritzem l'expressió: $3\,x^2 - 10\,x - 8$

```
> w:=3*x^2-10*x-8;
> factor(w);
```

O podem fer el mateix en una sola línia:

```
> factor(3*x^2-10*x-8);
```

Exemple 2.9

Primer desenvolupem l'expressió $2\,(x-2)\,(2\,x^2 + 5\,x + 2)\,(x+4)$. Després, per a tornar a descomposar l'expressió, apliquem la comanda `factor( )` al resultat. Podeu explicar per què el resultat final sembla diferent de l'expressió original?

```
> H:=2*(x-2)*(2*x^2+5*x+2)*(x+4);
> ans:=expand(H);
> factor(ans);
```

Exemple 2.10

Maple pot factoritzar expressions amb més d'una variable. Per exemple, factoritzem l'expressió, donada en funció de les variables x i y, $x^2\,y + 2\,xy + y$

```
> h:=x^2*y+2*x*y+y;
> factor(h);
```

Exemple 2.11

Si Maple no pot factoritzar una expressió util·litzant nombres racionals (és a dir, enters i fraccions) donarà com a resultat el mateix que heu introduït sense cap canvi.

```
> factor(3*x^2-10*x-9);
```

Exemple 2.12

La comanda `factor( )` no està limitada als polinomis. Es pot utilitzar per a factoritzar altres formes d'expressions, per exemple, les que contenen funcions trigonomètriques. Factoritzeu $(\sin x)^2 - (\cos x)^2$.

```
> factor((sin(x))^2-(cos(x)^2));
```

Exemple 2.13

Si la comanda `factor( )` s'utilitza amb una expressió racional, es factoritza el numerador i el denominador i els factors comuns es cancel·len per a simplificar l'expressió:

```
> A:=(x^3-7*x^2+15*x-9)/(x^2+4*x+4);
> factor(A);
> B:=(x^3-7*x^2+15*x-9)/(x^2-4*x+3);
> factor(B);
```

El següent exemple us permet veure la forma factoritzada sense les simplificacions.

Exemple 2.14

Les comandes de Maple `numer( )` i `denom( )` us permeten aïllar tant el numerador com el denominador d'una fracció. Aquí utilitzem aquestes comandes per examinar els factors del numerador i el denominador per separat (és a dir, abans de les simplificacions dels factors comuns).

```
> B:=(x^3-7*x^2+15*x-9)/(x^2-4*x+3);
> numer(B);  denom(B);
> factor(numer(B));  factor(denom(B));
```

Exercici 2.6

Factoritzeu l'expressió $3\,x^4 - 2\,x^3 + 22\,x^2 - 18\,x - 45$.

Exercici 2.7

Factoritzeu l'expressió $x^{(\frac{1}{2})} - x^{(\frac{3}{2})}$ i després utilitzeu la comanda `expand( )` per a comprovar el resultat.

2.4 La comanda `simplify( )`

La comanda `simplify( )` és la que ens permet simplificar diferents tipus d'expressions. Comencem veient el seu funcionament amb un seguit d'exemples.

Exemple 2.15

Considereu l'expressió $(\cos x)^5 + (\sin x)^4 + 2\,(\cos x)^2 - 2\,(\sin x)^2 - \cos(2\,x)$. Maple pot aplicar identitats per a simplificar expressions matemàtiques llargues, com ara expressions trigonomètriques.

```
> V:=cos(x)^5 + sin(x)^4 + 2*cos(x)^2 - 2*sin(x)^2 - cos(2*x);
> simplify(V);
```

Exemple 2.16

Les expressions trigonomètriques amb arguments donats com a múltiples d'algun angle queden simplificades en termes de funcions trigonomètriques d'aquest angle si és possible:

```
> simplify(sin(5*t)+sin(3*t));
```

Exemple 2.17

La comanda `simplify( )` es pot utilitzar per a sumar expressions racionals. A continuació reescrivim la suma $\dfrac{1}{x+1}+\dfrac{x}{x-1}$ com una única fracció.

```
> M:=(1/(x+1))+(x/(x-1));
> simplify(M);
```

Exercici 2.8

Simplifiqueu l'expressió $\dfrac{7}{x+2}+\dfrac{3\,x}{(x+2)^2}$

Exercici 2.9

Com simplifica Maple l'expressió $\sin(3\,t)-\sin(7\,t)$? Si aquesta expressió "simplificada" és o no és útil dependrà del que es tingui planejat fer amb ella.

2.4.1 La comanda `collect( )`

Hi ha ocasions en les que el que es necessita no és una simplificació com les que dóna la comanda `simplify()` si no que el que interessa és *agrupar* els termes d'una expressió segons les potències (enteres o fraccionaries) d'una variable en concret. Per fer aquest tipus de manipulació es pot utilitzar la comanda `collect( )` com en l'exemple següent:

Exemple 2.18

Si considereu l'expressió polinòmica $x^3 - x^2\,y + x^2 - 2\,x\,y - x - y^2\,x + y^3 + y^2 - y - 1$ (respecte les variables x i y) podeu agrupar respecte x amb

```
> collect(x^3-x^2*y+ x^2-2*x*y-x-y^2*x+y^3+y^2-y-1,x);
```

Com seria l'expressió agrupant respecte de la variable y?

Si mireu en l'ajuda de Maple veureu que la comanda `collect( )` admet que la *variable* respecte de la que es fa l'agrupament pugui ser una expressió gairebé arbitrària.

Exercici 2.10

Agrupeu en funció de $\ln(x)$ l'expressió $a\,\ln(x) - x\,\ln(x) - x$. Feu el mateix, respecte e^x, amb l'expressió $x^2\,e^x - 2\,x\,e^x + 2\,e^x - \dfrac{x^2}{e^x} - 2\dfrac{x}{e^x} - \dfrac{2}{e^x}$.

3 Gràfics

En aquesta secció aprendreu a dibuixar el gràfic d'una funció definida per una expressió. A més, altres temes que hi podeu trobar inclouen: combinar gràfics de diferents expressions en un únic dibuix, representar punts, i combinar diferents estructures gràfiques en un únic dibuix.

```
> restart;
```

3.1 Representar una expressió: la comanda `plot( )`

La comanda bàsica per representar gràficament una expressió o funció d'una variable és `plot`. Aprendrem el seu funcionament a partir dels exemples següents.

Exemple 3.1

Utilitzem la comanda `plot( )` per a dibuixar el gràfic de $3\,x^2 - 8$ per a x entre -5 i 5.

```
> plot(3*x^2-8,x=-5..5);
```

Observeu que Maple posa l'escala de l'eix de les y de manera automàtica, triant l'escala de les y que mostra tot el gràfic que correspon al domini que s'ha especificat.

Podem eliminar l'elecció automàtica de l'escala de les y especificant a part del rang de les x un rang per a les y. En la línia següent hem limitat el rang de les y a l'interval $[-20, 40]$.

```
> plot(3*x^2-8,x=-5..5,-20..40);
```

Si feu clic amb el botó esquerre del ratolí, el gràfic queda seleccionat i la barra d'opcions inferior es modifica. Ara quan feu clic en el gràfic, les coordenades del punt del lloc on ho feu es veuran en la finestra de l'esquerra. El botó 1:1 fa que les escales de les x i de les y siguin iguals.

Podeu experimentar sobre el gràfic anterior les diferents opcions possibles.

Exemple 3.2

L'escala automàtica és una característica útil però a vegades necessitarem especificar manualment el rang de les y. Per exemple l'escalat automàtic no és apropiat per a gràfics amb asímptotes verticals.

Compareu els dos gràfics següents. Noteu que hem fixat els límits per a les y a l'interval $[-20, 20]$ en la segona comanda `plot( )`.

```
> plot(x/(x-2),x=-5..5);
> plot(x/(x-2),x=-5..5,-20..20);
```

Exemple 3.3

Dibuixarem el gràfic de $y = x^3 + 1 - e^x$ en el domini $[-8, 8]$. Triarem un rang per a les y que permeti veure els quatre talls amb l'eix de les x.

Primer doneu un cop d'ull al gràfic amb l'escala automàtica per a les y.

```
> plot(x^3+1-exp(x),x=-8..8);
```

Com que els valors de les y (valors de l'expressió) a prop de 8 són negatius i molt grans en valor absolut l'escala vertical ha hagut de ser massa gran per a veure clarament els talls amb l'eix de les x.

S'obté una visió molt millor restringint els límits en el rang de les y.

```
> plot(x^3+1-exp(x),x=-8..8,-5..15);
```

Exercici 3.1

Dibuixeu $y = \sin(x)$ per a dos períodes complets.

Exercici 3.2

Dibuixeu $y = 3\,x^4 - 6\,x^2$ per al domini $[-10, 10]$ amb escala automàtica per a les y. Després d'observar el gràfic, editeu el domini i el recorregut per tal de veure els talls amb l'eix de les x clarament. Feu una estimació dels talls amb l'eix x amb el cursor.

3.2 Representar diferents expressions

Per a mostrar més d'un gràfic en el mateix dibuix feu-ne la llista dins de claudàtors `[ ]` separant-los per comes. Per exemple,

```
> plot([cos(x),x^2],x=-1..4,-4..4);
```

Noteu que cada un dels gràfics es mostra utilitzant un color diferent. Podeu especificar els colors per a cada funció afegint una opció de color al final de la comanda. Els colors s'assignaran en el mateix ordre que el de les funcions. Fixeu-vos que la llista dels colors també es fa amb uns claudàtors `[ ]`. Aquí hi ha un exemple.

```
> plot([cos(x),x^2],x=-1..5,-4..4, color=[blue,black]);
```

Aquests són els colors disponibles en Maple (s'han d'escriure en anglès, no val posar `groc`):

`aquamarine`, `black`, `blue`, `navy`, `coral`, `cyan`, `brown`, `gold`, `green`, `gray`, `grey`, `khaki`, `magenta`, `maroon`, `orange`, `pink`, `plum`, `red`, `sienna`, `tan`, `turquoise`, `violet`, `wheat`, `white` i `yellow`.

Exercici 3.3

Feu junts els gràfics de les funcions $y = x^2 - 5\,x + 6$ i $y = \dfrac{1}{(x-2)^2}$. Experimenteu amb diferents rangs per a les y de forma que es puguin veure dibuixos complets dels dos gràfics.

3.3 Representar punts

La comanda `plot( )` pot dibuixar també un o més punts.

Exemple 3.4

Dibuixem el punt $(2, 3)$. Noteu en la línia següent que utilitzem dos jocs de claudàtors.

```
> plot([ [2,3] ],style=point);
```

Podem controlar la mida dels rangs per a les x i per a les y afegint aquesta informació a la comanda com en la línia següent.

```
> plot([ [2,3] ],-7..7,-7..7,style=point);
```

Exemple 3.5

Per a dibuixar més d'un punt fem una llista dins la comanda `plot( )` (observeu les comes). Recordeu que s'ha de posar un parell de claudàtors per cada punt i un parell extra envoltant la llista.

```
> plot([ [2,3],[-2,5],[1,-4] ],-7..7,-7..7, style=point);
```

Exemple 3.6

Canviant l'estil a `line` es connecten els punts conservant l'ordre de la llista.

```
> plot([ [2,3],[-2,5],[1,-4] ],-7..7,-7..7, style=line);
```

Què passa si canviem l'ordre dels punts de la llista?

Exemple 3.7

Es poden utilitzar extensions opcionals per a especificar el color dels punts i el símbol que es fa servir (per exemple `diamond`, `circle` i `cross`, que és el que hi ha per defecte) per a representar-los. Utilitzant l'ajuda de Maple, esbrineu les diferents possibilitats que admet l'opció `symbol`.

```
> plot([[3,2],[-2,3],[2,-1]],style=point,color=blue,symbol=circle);
```

Exercici 3.4

Dibuixeu els punts següents utilitzant el color vermell i el símbol `diamond`: $[1, 4]$, $[-2, -3]$, $[4, -5]$ i $[-6, 5]$. Després connecteu els punts amb línies rectes amb una comanda `plot( )` a part.

3.4 Combinar gràfics d'expressions i punts: la comanda `display( )`

Hi ha comandes de Maple que no estan disponibles fins que no es *carrega* un *paquet* que les activa. Un d'aquests paquets és el **plots** que estén les possibilitats gràfiques. Per accedir a les comandes d'un paquet (carregar el paquet) s'utilitza la comanda `with( )`, la línia següent carrega el paquet **plots**

```
> with(plots);
```

Després d'executar aquesta comanda veureu que Maple ha mostrat, com a resultat, el nom de les comandes noves que proporciona el paquet. Si no volem que el resultat d'una comanda es mostri per pantalla però sí que s'executi, podem acabar la línia amb dos punts enlloc de punt i coma. Els dos punts al final de la línia fa que la línia es pugui executar sense mostrar cap resultat i serveix per a qualsevol comanda.

```
> with(plots):
```

De les comandes noves que apareixen quan carreguem **plots** mencionarem (de moment) únicament la comanda `display( )` que permet combinar gràfics d'expressions i de punts i línies en el mateix dibuix. Per utilitzar `display( )` el primer pas consisteix en nomenar individualment cada un dels components del dibuix.

Important: Ens hem d'assegurar que utilitzem uns **dos punts** al final de cada línia per a suprimir la presentació dels resultats ja que si no es fa el que s'obté com a resultat és força desagradable (mireu les tres primeres línies que hi ha a sota).

Després la comanda `display( )` s'utilitza per a fer el dibuix que volem (acaba amb un punt i coma).

```
> pict1:=plot([-3*x+5,9-x^2],x=-3..5,color=[green,red]):
> pict2:=plot([[-1,8],[4,-7]],style=point,color=blue,symbol=circle):
> display([pict1,pict2]);
```

De forma alternativa podem fer la llista d'aquestes tres comandes `plot( )` relacionades en un sol grup d'execució. Com que sovint sortiran línies bastant llargues és útil canviar de línia sense executar la comanda prement les tecles Shift ⇑ + ↵ a l'hora en el punt on volem que acabi cada una de les línies.

```
> pict1:=plot([-3*x+5,9-x^2],x=-3..5,color=[green,red]):
  pict2:=plot([[-1,8],[4,-7]],style=point,color=blue,symbol=circle):
  display([pict1,pict2]);
```

Exercici 3.5

Feu un gràfic que contingui a l'hora el gràfic de la funció $y = x^2 + x - 6$ i les seves interseccions amb l'eix de les x i el de les y (per a $x = -3$ i $x = 2$) marcades amb un cercle. (Podeu consultar en l'ajuda de Maple com la comanda `solve( )`, que introduirem en una secció propera, us pot donar aquests valors de la x).

4 Resoldre equacions

En aquesta secció aprendrem a aplicar la comanda de Maple `solve( )` per a obtenir les solucions **exactes** d'equacions (quan això sigui possible). Recordeu que en nombrosos casos no és possible obtenir solucions exactes de les equacions i que s'ha de deixar la feina als *solucionadors numèrics* per a poder calcular solucions **aproximades**. Més endavant en aquesta mateixa secció utilitzarem la comanda de Maple `fsolve( )` per a obtenir aproximacions decimals de solucions. També es discutirà la resolució de sistemes d'equacions lineals.

Comencem reinicialitzant les variables i carregant el paquet **plots**.

```
> restart:
> with(plots):
```

4.1 Introduir i manipular equacions: les comandes `lhs( )` i `rhs( )`

En aquest apartat veurem unes comandes bàsiques que ens permeten manipular equacions i extreure'n informació.

Exemple 4.1

Es pot donar un nom a una equació igual que es fa amb qualsevol altre expressió. En la línia següent introduïm l'equació $x^3 - 5\,x^2 + 23 = 2\,x^2 + 4\,x - 8$ i li posem com a nom `eqn1`.

```
> eqn1:=x^3-5*x^2+23=2*x^2+4*x-8;
> eqn1;
```

Exemple 4.2

Podem aïllar la part esquerra i la part dreta de l'equació utilitzant les comandes `lhs( )` i `rhs( )`.

```
> lhs(eqn1);
> rhs(eqn1);
```

Exemple 4.3

Vegeu com podem utilitzar les comandes `lhs( )` i `rhs( )` per a obtenir una equació que és equivalent a l'equació original `eqn1` però que té com a part dreta un 0. Posem-li com etiqueta `eqn2`.

```
> eqn2:=lhs(eqn1)-rhs(eqn1)=0;
> eqn2;
```

4.2 Obtenir solucions exactes: la comanda `solve( )`

En aquest apartat veurem com utilitzar la comanda `solve` en diferents situacions i el seu funcionament. Per començar considerarem equacions polinòmiques. Existeixen algoritmes per a calcular les solucions **exactes** per a polinomis que tenen fins a **grau 4**. La comanda de Maple `solve( )` coneix aquests algoritmes i els aplica. Aquesta comanda necessita dos arguments: l'equació i el nom de la variable.

Exemple 4.4

Per a obtenir les solucions exactes de l'equació polinòmica $3\,x^3 - 4\,x^2 - 43\,x + 84 = 0$ utilitzem la comanda `solve( )`. Noteu que el segon argument de la comanda li diu a Maple que x és la incògnita respecte de la qual volem obtenir la solució.

```
> solve(3*x^3-4*x^2-43*x+84=0,x);
```

Aquí Maple ha trobat les tres solucions i les ha ensenyades. Les solucions són retornades agrupades en una estructura de dades amb les diferents solucions separades per comes.

Exemple 4.5

A vegades voldrem seleccionar una solució de la llista de solucions obtinguda i utilitzar-la posteriorment en un altre càlcul. Podem fer això assignant primer un nom (en aquest cas utilitzem la lletra `N`) al resultat obtingut de la comanda `solve()`. Aleshores podem utilitzar un índex numèric per aïllar elements de la llista. Per exemple `N[1]` és el primer número de la llista, `N[2]` és el segon i així successivament (fixeu-vos que utilitzem claudàtors).

```
> N:=solve(x^2-5*x+3=0,x);
> N[1];
```

Exemple 4.6

Quan es treballa amb la comanda `solve( )` sovint és convenient començar donant un nom a l'equació. Recordeu que utilitzem "`:=`" per assignar el nom i només "`=`" per a l'equació pròpiament dita.

```
> eqn1:=7*x^3-11*x^2-27*x-9=0;
```

Seguidament resolem l'equació utilitzant la comanda `solve( )` i assignem el nom `H` al resultat.

```
> H:=solve(eqn1,x);
```

Per a practicar comproveu que cada un d'aquests valors satisfà l'equació. Això es pot fer fàcilment utilitzant la comanda `subs( )` que ja ha estat analitzada en una secció anterior. Repaseu-ne la síntaxi i el seu funcionament.

```
> subs(x=H[1],eqn1);
> subs(x=H[2],eqn1);
> subs(x=H[3],eqn1);
```

Exemple 4.7

A vegades les solucions "exactes" són massa complicades per a poder ser realment útils. En les dues línies següents resolem l'equació $x^3 - 34\,x^2 + 4 = 0$.

```
> eqn1:=x^3-34*x^2+4=0;
> H:=solve(eqn1,x);
```

Com podeu veure, llegir aquestes solucions exactes és realment un repte! Noteu que I representa $\sqrt{-1}$. Quan una solució és així de complicada és més útil mirar una solució aproximada utilitzant la comanda `evalf( )` que ja coneixeu.

```
> evalf(H);
```

Una bona alternativa a la comanda `solve( )` en una situació com la de l'exemple que acabem de veure és la comada `fsolve( )` que discutirem en la secció següent.

La comanda `solve( )` també es pot utilitzar per a determinar solucions exactes d'equacions **no polinòmiques**. Tot seguit veurem una llista d'alguns exemples simples d'aquest tipus. En tot cas, si les equacions són complicades, per exemple si combinen exponencials, polinomis i expressions trigonomètriques, normalment no es podrà disposar de les solucions exactes. Un altre cop la comanda `fsolve( )` serà una alternativa.

Exemple 4.8

Resoleu l'equació $5\,e^{(\frac{x}{4})} = 43$.

```
> solve(5*exp(x/4)=43,x);
```

Exemple 4.9

A vegades Maple no mostra **totes** les solucions possibles.

```
> solve(sin(x)=1/2,x);
```

Com podeu utilitzar el resultat que acabeu d'obtenir amb la comanda `solve` per a poder escriure el conjunt de totes les solucions de l'equació?

Exercici 4.1

Resoleu l'equació $x^3 - 11\,x^2 + 7\,x + 147 = 0$. Per què Maple produeix només dues solucions diferents per a aquesta equació cúbica? Per què una d'elles està escrita dues vegades? (**Indicació:** Factoritzeu la part esquerra de l'equació).

El fet que $(x - 7)$ és un **factor repetit** porta com a conseqüència que l'equació cúbica tingui només dues solucions diferents -3 i 7. Diem que l'arrel 7 té **multiplicitat 2**, això significa que hi ha dos factors de la forma $(x - 7)$ en la factorització del polinomi.

4.3 Obtenir solucions aproximades: la comanda `fsolve( )`

La comanda `fsolve()` es pot utilitzar per a aproximar solucions de qualsevol tipus d'equació i funciona de manera anàloga a la comanda `solve()`. Per a equacions polinòmiques `fsolve()` produeix una **llista completa** de totes les solucions reals en un sol pas (mireu l'Exemple 4.10). Per a altres tipus d'equacions, `fsolve( )` es pot utilitzar per a obtenir les solucions **d'una en una** (mireu els Exemples 4.11 i 4.12).

Exemple 4.10

La comanda de Maple `fsolve( )` calcularà una aproximació numèrica per a cada una de les solucions reals d'una equació polinòmica. En aquest exemple l'utilitzarem per aproximar totes les solucions reals de l'equació: $x^4 - x^3 - 17\,x^2 - 6\,x + 2 = 0$.

```
> eqn:=x^4-x^3-17*x^2-6*x+2=0;
> fsolve(eqn,x);
```

Les quatre solucions que es mostren ens donen una llista completa de les aproximacions de les solucions de l'equació polinòmica.

Exemple 4.11

En aquest exemple trobarem **totes** les solucions reals de l'equació $x^3 + 1 - e^x = 0$ utilitzant la comanda `fsolve( )`.

```
> eqn:=x^3+1-exp(x)=0;
> fsolve(eqn,x);
```

Maple ens dóna una solució real però aquest cop Maple no ens ha explicat la historia completa. Hi ha altres solucions? Com es poden trobar? Tot seguit en l'Exemple 4.12 es presenta un procediment sistemàtic per a determinar les solucions que resten.

Exemple 4.12

Trobem les altres solucions reals de l'equació $x^3 + 1 - e^x = 0$. El primer pas per a trobar les altres solucions és fer un dibuix del gràfic de la part esquerra de l'equació. Recordeu que els talls amb l'eix de les x de $y = x^3 + 1 - e^x$ es corresponen exactament amb les solucions de l'equació $x^3 + 1 - e^x = 0$.

```
> plot(x^3+1-exp(x),x=-3..5,-5..15);
```

El gràfic mostra **quatre** interseccions amb l'eix de les x. Una d'elles correspon a la solució que hem obtingut en l'Exemple 4.11. Sabrieu dir quina? Com trobaríeu les altres que falten?

Podem estendre la comanda `fsolve( )` per a que miri de trobar una solució en un interval particular. Per exemple per a trobar la solució negativa li demanem a Maple que busqui en l'interval $[-1, -0.2]$ ja que podem veure a partir del gràfic que hi ha exactament una solució en aquest interval. La manera de fer-ho és escriure l'interval corresponent juntament amb el nom de la variable a la comanda `fsolve`.

```
> fsolve(eqn,x=-1..-.2);
```

Per a determinar les altres dues solucions utilitzem `fsolve( )` un altre cop, però aquesta vegada buscant a l'interval $[1, 2]$ i a l'interval $[4, 5]$.

```
> fsolve(eqn,x=1..2);
> fsolve(eqn,x=4..5);
```

Què passa si demaneu a Maple que busqui una solució en un interval on no hi ha solucions? Proveu-ho.

A partir del gràfic és clar que no hi ha talls amb l'eix de les x entre 2 i 4 (i per tant no hi ha solucions).

```
> fsolve(eqn,x=2..4);
```

Noteu que Maple simplement respon amb la línia que originalment heu introduït sense cap canvi quan no pot trobar una solució a l'interval donat.

Hi ha altres solucions? Per exemple, hi ha alguna solució per a x més gran que 5? Podem comprovar això fent més gran l'interval sobre el que fem el gràfic. En la línia següent allarguem

l'interval fins al $[-3, 50]$.

```
> plot(x^3+1-exp(x),x=-3..50,-10..15);
```

No apareixen nous talls amb l'eix de les x. El gràfic confirma el que podem esperar mirant els termes de l'equació, és a dir el terme exponencial domina i fa que el gràfic vagi baixant a la llarga.

Alternativament podem utilitzar la comanda `fsolve( )`, ara buscant en aquest interval més gran.

```
> fsolve(eqn,x=5..50);
```

Tal com esperàvem Maple no haurà trobat solucions.

D'una forma semblant podem comprovar si hi ha solucions **cap a l'esquerra**. Aquí busquem si hi ha solucions a l'interval $[-50, -1]$.

```
> fsolve(eqn,x=-50..-1);
```

Cap ni una tampoc!

Finalment, hem pogut fer una llista completa de les solucions aproximades de la nostra equació original $x^3 + 1 - e^x = 0$. Són: -0.8251554597, 0, 1.545007279 i 4.567036837.

Exemple 4.13

Utilitzem `fsolve( )` per a calcular les solucions aproximades de l'equació: $\dfrac{x^2}{20} - 10\,x = 15\,\cos(x + 15)$.

Com ja hem vist en l'últim exemple utilitzarem un gràfic per ajudar-nos a determinar el nombre i la situació aproximada de les solucions. La nostra feina es simplifica si comencem convertint l'equació que tenim en una d'equivalent que té com a part dreta un 0. Així buscarem les solucions de l'equació equivalent: $\dfrac{x^2}{20} - 10\,x - 15\,\cos(x + 15) = 0$.

Si ara dibuixem el gràfic de la part esquerra d'aquesta equació obtindrem un altre cop solucions en cada un dels talls amb l'eix de les x.

```
> eqn:=x^2/20-10*x-15*cos(x+15)=0;
```

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

A partir del gràfic sembla que hi ha una solució a l'interval $[1, 2]$. Així doncs ara dirigirem Maple cap a trobar una solució en aquest interval.

```
> fsolve(eqn,x=1..2);
```

Hem trobat totes les solucions d'aquesta equació? De fet hi ha una altra solució! Per a trobar-la comenceu estirant l'interval sobre el que heu fet el gràfic. Aleshores utilitzeu `fsolve( )` per a obtenir una aproximació numèrica d'aquesta segona solució.

Exercici 4.2

Determineu totes les solucions de l'equació $x^5 - 4\,x^3 + 3\,x^2 + 7\,x - 1 = 0$. Comenceu mirant un gràfic significatiu.

Exercici 4.3

Determineu totes les solucions de l'equació $x^2 - 2 = \ln(x + 5)$. Utilitzeu el gràfic **d'una** expressió per a localitzar les solucions. Comproveu cada una de les solucions substituint-la en l'equació original.

Exercici 4.4

Els gràfics de $y = 10 - x^2$ i de $y = 4\sin(2\,x) + 5$ s'intersequen dues vegades sobre l'interval $[-5,\, 5]$.

a) Feu el gràfic de les dues equacions juntes i estimeu amb el ratolí els punts d'intersecció.

b) Escriviu una equació que es pugui resoldre per a determinar les coordenades x dels puns d'intersecció.

c) Utilitzeu `fsolve( )` per a resoldre aquesta equació.

d) Utilitzeu els resultats de la part c) per a estimar les coordenades y dels punts d'intersecció.

e) Sembla que les corbes es poden intersectar en un tercer punt a prop de $(1,\, 9)$. Utilitzeu `fsolve( )` i/o un gràfic significatiu per a demostrar que no hi ha cap punt d'intersecció en aquest lloc.

4.4 Resoldre equacions formals

En moltes ocasions Maple també pot resoldre equacions de manera formal per a qualsevol de les variables que hi aparegui. Suposeu que volem obtenir la solució per a la variable g de l'equació: $4 - v = 2\,T - k\,g$. És a dir, volem aïllar g en funció de les demés variables. La comanda `solve( )` funciona bé en aquest cas.

```
> solve(4-v=2*T-k*g,g);
```

Aquí hi ha una manera més maca per a mostrar el mateix resultat per pantalla:

```
> g=solve(4-v=2*T-k*g,g);
```

Exercici 4.5

Editeu l'última comanda per a trobar solucions per a les altres lletres T, k i v.

Exercici 4.6

Resoleu l'equació $x^2 + y^2 = 9$ per a la variable y. Assigneu el conjunt de solucions a una variable que es digui S. Quina relació hi ha entre les solucions `S[1]` i `S[2]`?

4.5 Resoldre sistemes d'equacions lineals utilitzant la comanda `solve( )`

Recordeu que cal reinicialitzar les variables abans de continuar. Carreguem també el paquet **plots**:

```
> restart:
> with(plots):
```

La comanda `solve( )` també es pot utilitzar per a resoldre un sistema de m equacions lineals amb n incògnites. En direm un **sistema lineal** m **per** n perquè sigui més curt.

Exemple 4.14

Començarem per resoldre el sistema 2 per 2: $\begin{cases} 3\,x + 2\,y = 3 \\ x - y = -4 \end{cases}$

```
> solve({3*x+2*y=3,x-y=-4});
```

Un gràfic de les dues funcions subjacents mostrarà que la solució correspon al punt d'intersecció en $(-1, 3)$. Però primer necessitem obtenir la forma explícita de cada una de les dues funcions lineals abans de poder fer-ne el dibuix. Per tant resolem cada una de les equacions respecte y.

```
> y1:=solve(3*x+2*y=3,y);
> y2:=solve(x-y=-4,y);
```

Ara construim un gràfic format de dues parts: la "`part1`" conté els gràfics de les dues equacions i la "`part2`" dibuixa el punt que és la solució que hem trobat. Aquest punt ha de ser el punt d'intersecció de les dues línies. Ho és efectivament?

```
> part1:=plot([y1,y2],x=-5..5):
> part2:=plot([[-1,3]],style=point,color=blue,symbol=circle):
> display([part1,part2]);
```

Exemple 4.15

En aquest exemple solucionarem un sistema **3 per 3** amb incògnites x, y i z.

Resolem el sistema 3 per 3: $\begin{cases} x+y+z=1 \\ 3\,x+y=3 \\ x-2\,y-z=0 \end{cases}$

```
> solve({x+y+z=1, 3*x+y=3, x-2*y-z=0});
```

Exercici 4.7

Determineu la solució del sistema: $\begin{cases} 4\,x+3\,y=12 \\ 5\,x-7\,y=35 \end{cases}$

Comproveu la solució substituint els valors que s'obtenen en les dues equacions del sistema.

Sistemes lineals amb un nombre de solucions infinit

Quan un sistema té més **incògnites** que **equacions** sovint trobem no una sinó un nombre infinit de solucions. Tot seguit en veurem un exemple.

Exemple 4.16

Resolem el sistema: $\begin{cases} x+y+z=1 \\ 3\,x+y=3 \end{cases}$

```
> solns:=solve({x+y+z=1, 3*x+y=3});
```

Noteu que aquest cop no obtenim un únic conjunt de valors numèrics per a x, y i z. En lloc d'això Maple ens diu com han d'estar relacionats els diferents valors de x, y i z per a construir una solució típica.

En particular l'expressió $y=y$ en la resposta anterior indica que la incògnita y pot ser **qualsevol** número. Ens referirem a aquesta incògnita com la *variable lliure* de la solució. Per a obtenir qualsevol **solució particular** (entre les infinites solucions que hi ha) trieu un valor per a la y i utilitzeu-lo per a calcular els valors corresponents de la x i de la z. Per exemple si $y=-9$.

```
> subs(y=-9,solns);
```

Així una solució particular és: $x=4$, $y=-9$ i $z=6$.

Preneu-vos un minut de temps i verifiqueu a mà que aquests tres nombres satisfan realment les equacions originals: $x+y+z=1$ i $3\,x+y=3$.

Ara mirem la solució generada quan prenem $y=-3$.

```
> subs(y=-3,solns);
```

Així doncs dues de les infinites solucions que obtenim per a les incògnites (x, y, z) són $(4, -9, 6)$ i $(2, -3, 2)$.

Exercici 4.8

Resoleu el sistema: $\{x + 2\,y + z = 2,\ 3\,x + y = 1\}$ i doneu com a mínim tres solucions particulars.

5 Funcions: definir, avaluar i fer-ne gràfics

L'objectiu bàsic d'aquesta secció és el d'aprendre primer de tot a definir funcions $f(x)$. Després també veurem com avaluar funcions, resoldre equacions on hi intervenen funcions, i com fer-ne gràfiques.

```
> restart;
```

5.1 Definir i esborrar una funció en Maple

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

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

És important que preneu nota de la sintaxi que utilitzem. És **absolutament necessari** introduir la fletxa "->" construïda amb el signe "menys" i el símbol "més gran que". **Maple no definirà una funció** si introduïu `f(x) := cos(Pi*x)+3;`.

A continuació podeu veure la diferència entre una expressió i una funció en Maple. 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

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

Evidentment, no cal que les funcions depenguin d'un sol argument. Podem considerar per exemple

Exemple 5.1

```
> f:= (x,y)->x^2-x*y+ln(x^2+y^2);
```

Aquí obtenim una funció `f` que admet dues variables `(x,y)`.

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, y) per ln(cos(5 x)), escriviu:

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

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

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

És important que noteu que mentre f es una funció, el valor de $f(x)$ és una expressió i es pot tractar com a tal. És a dir, podem aplicar-li qualsevol de les comandes que hem vist per expressions. Per exemple, avaluant,

```
> subs(x=0,f(x));
> subs(x=2*t+1,f(x));
```

Ara bé, aquesta no és la millor manera d'avaluar una funció com veurem més endavant.

Si voleu esborrar la funció f sense definir-la de nou utilitzem el mateix procediment que el que vam veure a la secció 1 per esborrar variables, és a dir, 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 memoria de la sessió.

5.2 Avaluar una funció

Quan heu definit una funció, podeu avaluar-la per a diferents valors o per a expressions literals utilitzant la notació funcional. És a dir, com veurem en l'exemple següent podem substituir x per l'expressió que volguem. Abans de definir la funció però netegem el valor de f.

Exemple 5.2

```
> 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. Simplement, consisteix en aplicar la definició de composició de funcions.

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

Exercici 5.2

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

5.3 Resoldre equacions en les que intervenen funcions

Un cop definida una funció, també podem resoldre equacions amb aquesta funció de manera exacta o aproximada utilitzant les comandes vistes a la secció 4.

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

Com que g ha estat definida com una funció i no una expressió, el nom de la variable no és important. Això no passava quan utilitzàvem expressions en la secció anterior.

```
> fsolve(g(x)=0,x);
```

5.4 Gràfics de funcions

La comanda `plot` funciona igualment per a funcions f. Sempre podem utilitzar la comanda `plot` amb l'expressió obtinguda avaluant a la variable x, $f(x)$. Vegem-ho en el següent exemple.

Exemple 5.3

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

Es poden dibuixar gràfics de diferents funcions simultàniament de la mateixa manera que es fa per a les expressions.

Considerem la funció $\mathrm{f}(x) = \dfrac{2}{x^2+1}$. 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);
> f(x+1);
> f(x-3);
> f(x-6);
> plot([f(x),f(x+1),f(x-3),f(x-6)],x=-5..10,-1..3);
```

La comanda `plot` admet també que l'entrada sigui una funció directament f. Aleshores no s'ha d'especificar la variable x en el rang.

```
> plot(f,-5..10,-1..3);
```

Exercici 5.3

Definiu la funció $\mathrm{f}(x) = 2\,x - |x^2 - 5|$ i tot seguit 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 $y = 4$. Utilizeu 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 $\mathrm{k}(x) = 4$.

5.5 Com convertir expressions en funcions

Fins ara hem vist com a partir d'una funció f definida en Maple, la podem convertir en una expressió mitjançant avaluació en una variable $f(x)$. Maple també conté la comanda que ens permet resoldre el problema invers, és a dir, donada una expressió ens pot ser molt útil convertir-la en una funció. Aquesta situació es resol amb la comanda `unapply( )`. Observeu el seu funcionament en l'exemple següent.

Exemple 5.4

```
> f:='f';
> expr:=ln(3*x^3+sqrt(x-1));
> f:=unapply(expr,x);
> f(x);
> evalf(f(3));
```

Podeu veure que la comanda `unapply( )` té dos arguments: el primer és l'expressió que volem convertir en funció i el segon és la variable que surt en l'expressió respecte de la que volem que depengui la funció resultant.

Exercici 5.6

Considereu l'expressió $x^2 - y^2 + \sin(x\,y)$. Mireu en l'ajuda sobre `unapply` la manera de convertir l'expressió anterior en una funció depenent de les dues variables x i y.

Exemple 5.5

Sigui $y = f(x)$ una funció tal que satisfà la relació $y\,x = -y^2 + 7\,x + 2\,y$. Volem definir la corresponent funció f depenent de x. Primer utilitzem la comanda `solve` per aïllar y en funció de x i després convertim l'expressió resultant (o expressions resultants) en funcions.

```
> expre:=y*x=-y^2+7*x+2*y;
> yexpre:=solve(expre,y);
> f1:=unapply(yexpre[1],x);
> f2:=unapply(yexpre[2],x);
> plot([f1,f2],-1..2);
```

Exemple 5.6

En algunes ocasions és millor conservar les expressions mitjançant funcions per evitar perdre la informació que conté l'expressió quan es manipulen les variables. Per exemple, considerem l'expressió següent i la seva corresponent funció,

```
> ex:=(ln(x+1)-x^2)/(x+1);
> func:=unapply(ex,x);
```

Durant la sessió de Maple podem assignar un valor a la variable `x`. Aleshores observeu que passa amb l'expressió i la funció.

```
> x:=3;
> ex;
> func(x);
```

Ja no podem recuperar l'expressió inicial utilitzant `ex` (sense haver de *buidar* la variable `x`) que ha passat a tenir un valor numèric però si que podem recuperar-la mitjançant la funció que hem definit.

```
> func(t);
```

Exercici 5.7

Sigui $y = f(x)$ una funció que satisfà la igualtat $y\,x + y^2 - 6\,x + x^2 = 0$. Mitjançant la comanda `solve` doneu les expressions de y en funció de x i convertiu-les en funció. Feu-ne les corresponents representacions gràfiques.

6 Llistes, conjunts i seqüències

El programa Maple permet treballar amb diferents *conjunts* o estructures de dades. Aquests *conjunts* es poden definir de diverses maneres i a més en podem canviar l'estructura d'un tipus a un altre segons ens interessi. Les estructures de dades que analitzarem en aquesta pràctica són les llistes, conjunts, seqüències i taules.

6.1 Llistes

Una llista és un conjunt ordenat d'expressions on es permeten repeticions. L'entrada al Maple es fa entre claudàtors (`[ ]`) i separant els elements que conté amb comes.

Exemple 6.1

Definim la variable a com la llista següent $[1, 2, 3, 2, 1, 2]$:

```
> a:=[1,2,3,2,1,2];
```

Els elements d'una llista es poden cridar o recuperar un a un mitjançant la posició que ocupen escrivint aquesta entre claudàtors. Una particularitat d'aquest mètode és que quan utilitzem nombres negatius retorna l'element corresponent començant per la cua.

Exemple 6.2

Si volem recuperar o extreure el tercer element de la llista `a` escriurem:

```
> a[3];
```

Mentre que si volem l'últim podem fer-ho amb la comanda següent:

```
> a[-1];
```

De la manera semblant també podem extreure una subllista de la llista original. Si volem una subllista amb els elements entre les posicions i i j cal introduir l'argument `i..j` dins els claudàtors.

Exemple 6.3

En aquest exemple recuperarem la subllista que conté els elements des de la posició 2 fins a la 4.

```
> a[2..4];
```

Una manera de construir llistes a partir d'altres és mitjançant la unió, és a dir, també podem

construir una llista com la unió de vàries. Per a això necessitem abans "*treure*" els claudàtors de les llistes inicials i afegir-los després a la unió o concatenació obtinguda. La comanda que ens permet treure els claudàtors és la comanda `op( )`. Un cop hem tret els claudàtors podem concatenar dues llistes. Per exemple, observeu el funcionament de la comanda `op` executant la següent línia de Maple.

```
> op(a);
```

Exercici 6.1

Definiu la funció `concatenar`, depenent de dues variables, que retorni la unió de dues llistes donades. Comproveu que la definició que heu fet és correcta fent la prova amb les llistes $a = [1, 2, 3, 2, 1]$ i $b = [3, 2, 4]$. El resultat ha de ser la llista $c = [1, 2, 3, 2, 1, 3, 2, 4]$.

6.2 Conjunts

En un conjunt de dades no hi ha repeticions i les dades no tenen un ordre prefixat (és a dir, Maple pot variar l'ordre, en funció del context). L'entrada a Maple es fa entre claus ({ }) i separant els elements que conté per comes. La majoria d'opcions que hem vist per les llistes són també vàlides per als conjunts, però tenint en compte les possibles diferències.

Exemple 6.4

Definiu b com el conjunt $\{1, 2, 5, 1, 3\}$ i observeu la sortida que us torna.

```
> b:={1,2,5,1,3};
```

Els elements d'un conjunt també es poden cridar mitjançant els claudàtors, tenint en compte que, a priori, no coneixem l'ordre en que Maple guarda els elements ja que són estructures no ordenades de dades.

Exemple 6.5

Per a cridar el tercer element del conjunt b fem:

```
> b[3];
```

Observeu que el tercer element no coincideix amb el que hem introduït inicialment en la posició 3 quan hem definit b.

Exercici 6.2

Definiu a i b com els conjunts $\{x, y, z\}$ i $\{t, u\}$. Definiu la unió d'aquests dos conjunts $c = a \cup b$. Construïu una funció que, a partir de dos conjunts, doni com a resultat la seva unió. Proveu-la amb els conjunts $A = \{1, 4, 9, 16\}$ i $B = \{2, 4, 6, 8, 10\}$.

6.3 Seqüències

Una seqüència d'elements és un conjunt ordenat però que s'interpreta com n arguments (i no una sola llista), on n és el nombre d'elements de la seqüència. L'entrada al Maple es realitza amb les dades separades entre comes i sense cap delimitador.

Exemple 6.6

Si volem definir com a la seqüència $1, 4, 9, 16$ posem

```
> a:=1,4,9,16;
```

Una altra manera d'aconseguir seqüències és mitjançant la comanda `seq` que ja va ser introduïda en el primer bloc i vàreu utilitzar en alguns exemples.

Exemple 6.7

La seqüència c amb els primers 10 quadrats enters es construeix amb la comanda `seq` de la manera següent.

```
> c:=seq(i^2,i=1..10);
```

Igual que fèiem en el cas de les llistes o conjunts podem extreure o recuperar un element d'una seqüència, unir seqüències, ...

Exercici 6.3

Definiu la funció de dues variables $f(x, y) = x^2 - y^2$. Considereu $a = (5, 7)$. Com s'ha d'introduir a per aconseguir que en escriure $f(a)$ Maple doni com a resultat el valor -24?

6.4 Taules

Totes les estructures per emmagatzemar dades que hem vist fins ara admeten només una indexació mitjançant nombres enters. Maple té una estructura de dades més sofisticada que admet com a conjunt de indexació pràcticament qualsevol cosa. És l'estructura anomenada `table`. Com veurem més endavant fins i tot admet cadenes de caràcters com a índexs. La millor manera de veure com funciona es mitjançant uns exemples. Ja sabeu que podeu aprendre més de les seves propietats utilitzant l'ajuda de Maple.

Exemple 6.8

En aquest exemple veurem com es defineix aquesta estructura i el seu comportament respecte assignació de valors.

```
> T:=table([1=1,2=3,3=5,x=u,y=v,z=w,t=sin(2*t)]);
> T[1];
> T[2];
> T[x];
> T[5];
> t:=5;
> T[t];
> T[5];
> subs(u=3,T[x]);
```

I podem afegir-hi nous elements,

```
> T[5]:=m;
> T[5];
```

Com heu pogut observar el conjunt de índexs és qualsevol cosa i per tant, no és immediat saber el que s'ha de posar entre claudàtors per a recuperar cada un dels elements d'una estructura `table`. De la feina de recuperar el conjunt d'índexs per als que s'ha introduït algun valor en una taula se n'encarrega la comanda `indices`.

Exemple 6.9

```
> indices(T);
```

Exemple 6.10

En aquest exemple utilitzarem el format `table` d'estructura de dades per crear un petit diccionari.

```
> Traductor:=table();
> Traductor[dilluns]:=Monday;
> Traductor[dimarts]:=Tuesday;
> Traductor[dimecres]:=Wednesday;
> Traductor[dijous]:=Thursday;
> Traductor[divendres]:=Friday;
> Traductor[dissabte]:=Saturday;
> Traductor[diumenge]:=Sunday;
> Traductor[dimecres];
> Traductor[divendres];
```

Fixeu-vos que la taula `Traductor` s'ha definit inicialment sense cap assignació de valors i que cada un dels valors s'ha afegit en les comandes posteriors.

6.5 Canvis de format entre llistes, seqüències i conjunts. La comanda `convert`

Maple permet transformar una expressió que té guardada com una llista, seqüència o conjunt als altres formats.

Un d'aquests canvis ja l'hem utilitzant. A l'apartat corresponent a les llistes ja hem vist com podem canviar una llista de dades en una seqüència, mitjançant la comanda `op( )`.

Exemple 6.11

Convertirem la llista $a = [1, 2, 3, 2, 1]$ en una seqüència b.

```
> a:=[1,2,3,2,1];
> b:=op(a);
```

Un cop tenim una seqüència podem transformar-la un altre cop en una llista simplement afegint els delimitadors que caracteritzen les llistes: els claudàtors.

Exemple 6.12

Si volem transformar la seqüència b en una llista només cal afegir els claudàtors:

```
> [b];
```

El procediment anàleg funciona amb els conjunts i les seqüències. Recordeu que els delimitadors que caracteritzen els conjunts són les claus. és molt important que tingueu en compte que quan

passem una seqüència o una llista a format conjunt perdem les repeticions i l'ordre que tenien originalment.

Exercici 6.4

Convertiu la llista $a = [3, 2, 1, 4, 2, 2, 2]$ en un conjunt b i feu una nova llista c amb els elements de b.

Una altra manera de fer conversions entre llistes, seqüències i conjunts és amb la comanda `convert( )`. Aquesta comanda és molt versàtil i s'utilitza també per a canvis en altres contexts dins de Maple (taules a matrius, ...).

Per a passar d'un format a l'altre tan sols cal conèixer la terminologia de Maple corresponent a cada un d'ells: `set` per a conjunts i `list` per a llistes. Per seguretat escriurem aquestes paraules entre comes ja que si haguessin estat definides com a variables serien substituïdes pel seu valor i així només són una paraula.

Exemple 6.13

Considerem la llista a formada pels elements $a = [3, 2, 1, 4, 3]$. Podem passar-la a format conjunt mitjançant:

```
> a:=[3,2,1,4,3];
> b:=convert(a,'set');
```

I podem tornar al format llista mitjançant:

```
> convert(b,'list');
```

6.6 Comptar elements de llistes i conjunts

La funció que permet comptar el nombre d'elements d'una llista o conjunt és la funció `nops( )`.

Exemple 6.14

Volem saber quants elements diferents té la llista $a = [3, 4, 2, 1, 3, 5, 3, 4, 3, 4, 5, 3, 2, 2, 4]$. La funció `nops( )` aplicada directament a la llista

```
> a:=[3,4,2,1,3,5,3,4,3,4,5,3,2,2,4];
> nops(a);
```

retorna el nombre d'elements de la llista (amb repeticions incloses). Una manera de saber quants elements diferents té és convertir primer la llista a conjunt b i seguidament comptar el nombre d'elements del conjunt:

```
> b:=convert(a,'set'); nops(b);
```

Exercici 6.5

Quants elements té la llista `a:=[[1,2,3],[x,y],{2, 3,4}, x, 1, 2]`? Raoneu la resposta.

6.7 Exemple: grups de permutacions

Recordeu que una permutació d'un conjunt de n elements $\{1, 2, \ldots, n\}$ és una aplicació bijectiva:

$$\begin{array}{rccc} \sigma: & \{1,2,\ldots,n\} & \longrightarrow & \{1,2,\ldots,n\} \\ & i & \mapsto & \sigma(i) \end{array}$$

El conjunt de totes aquestes permutacions es designa per S_n.

Les permutacions s'acostumen a escriure mitjançant la següent notació:

$$\begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ \sigma(1) & \sigma(2) & \sigma(3) & \cdots & \sigma(n) \end{pmatrix}$$

o bé mitjançant un producte de cicles disjunts.

Per a treballar amb permutacions utilitzant Maple necessitem carregar el paquet **group** que conté comandes específiques per a la seva manipulació.

```
> with(group);
```

6.7.1 Entrar elements

Inicialment, la manera més immediata d'introduir una permutació en Maple és mitjançant les imatges de cada posició posades en una llista de la forma `[`$\sigma(1), \ldots, \sigma(n)$`]`. Ara bé, per la majoria de funcions és necessari escriure-la en la seva descomposició en cicles disjunts i això es fa introduint una *llista de llistes* de la forma `[`$c_1, \ldots, c_k$`]`, on cada c_i és una llista que representa un cicle. En particular l'element neutre (la permutació identitat) es representa en aquest context per una llista buida (`[ ]`).

Exemple 6.15

Per a entrar la permutació $\sigma = (1, 2, 3)(4, 5)$ (producte dels cicles $(1, 2, 3)$ i $(4, 5)$) del grup simètric S_5 definim

```
> sigma:=[[1,2,3],[4,5]];
```

En determinats casos la manera com tindrem definida una permutació serà com una llista ordenada amb les imatges de cada posició, o sigui, `[`$\sigma(1), \sigma(2), \ldots, \sigma(n)$`]`. Tot i això, si volem operar amb ella haurem de transformar-la a un producte de cicles disjunts mitjançant la comanda `convert`. En el següent exemple veurem com passar d'un format a un altre.

Exemple 6.16

Si considerem la permutació $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 1 & 3 & 5 \end{pmatrix}$ i la volem introduir amb aquesta notació escriurem,

```
> sigma:=[2,4,1,3,5];
```

Si la volem descriure com a producte de cicles disjunts utilitzarem la comanda `convert` de la manera següent,

```
> convert(sigma,'disjcyc');
```

Tenint en compte que sempre podem recuperar la notació inicial utilitzant també la comanda `convert`.

```
> convert(%,'permlist',5);
```

on l'argument 5 és per a concretar que és un element de S_5.

Exercici 6.6

Donada la permutació `tau` expressada com a producte dels cicles disjunts `[[1,3,5],[2,7,6]]`, podeu preveure quina diferència hi ha entre els resultats que s'obtenen quan executeu les comandes `convert(tau, 'permlist', 7)` i `convert(tau, 'permlist', 10)`? I si intenteu fer `convert(tau, 'permlist', 5)`?

6.7.2 Operacions amb permutacions

Les permutacions es poden multiplicar (composar) i invertir. En aquest apartat veurem com realitzar aquestes operacions usant comandes del paquet **group** que hem carregat.

Per a multiplicar elements utilitzem la funció `mulperms( )`.

Exemple 6.17

Per a multiplicar les permutacions $\sigma_1 = (1,2,3)(4,5)$ amb $\sigma_2 = (3,4)$ escriurem:

```
> sigma1:=[[1,2,3],[4,5]];
> sigma2:=[[3,4]];
> mulperms(sigma1,sigma2);
```

Què ha calculat: $\sigma_1 \circ \sigma_2$ o bé $\sigma_2 \circ \sigma_1$?

Per a calcular la inversa d'una permutació s'utilitza la funció `invperm( )`.

Exemple 6.18

Calculem la inversa de la permutació $(1,2,3)(5,6,4)$:

```
> sigma:=[[1,2,3],[6,5,4]];
> beta:=invperm(sigma);
```

Podem comprovar que `beta` és la inversa de `sigma`.

```
> mulperms(sigma,beta);
> mulperms(beta,sigma);
```

Exercici 6.7

Considereu les permutacions següents de S_8: $\sigma_1 = (1,2,3)(6,7,8)$, $\sigma_2 = (5,4,3,1)$ i $\sigma_3 = (2,3,4,5,6,7)$. Calculeu les composicions $\sigma_1 \circ \sigma_2 \circ \sigma_3$, $\sigma_3 \circ \sigma_2 \circ \sigma_1$, $\sigma_1^{-1} \circ \sigma_2^{-1} \circ \sigma_3^{-1}$, $\sigma_3^{-1} \circ \sigma_2^{-1} \circ \sigma_1^{-1}$, $(\sigma_1 \circ \sigma_2 \circ \sigma_3)^{-1}$ i $(\sigma_3 \circ \sigma_2 \circ \sigma_1)^{-1}$.

Quines relacions veieu entre els resultats que heu obtingut? Recordeu que el producte $\sigma_2 \circ \sigma_1$ correspon a la comanda de Maple `multperms(sigma1,sigma2)`.

6.7.3 Signe d'una permutació

Maple té implementada la funció signe d'una permutació amb el nom de `parity( )`. L'argument és una permutació que ha d'estar definida com a producte de cicles disjunts i aleshores Maple ens retorna els valors 1 o -1 depenent del signe de la permutació.

Exemple 6.19

Per calcular el signe de les permutacions $(2,1,3,5)(4,9,10)$ i $(3,4,1)(5,6,7)$ amb Maple ho faríem de la manera següent:

```
> parity([[2,1,3,5],[4,9,10]]);
> parity([[3,4,1],[5,6,7]]);
```

Exercici 6.8

Considereu la permutació de S_{10} donada per $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 3 & 6 & 9 & 10 & 4 & 7 & 8 & 2 & 1 & 5 \end{pmatrix}$. Determineu el seu signe.

6.7.4 Ordre d'una permutació

Recordeu que l'ordre d'una permutació σ és el més petit nombre natural n tal que $\sigma^n = \sigma \circ \overset{(n)}{\ldots} \circ \sigma$ és la permutació identitat. Per tant, si es vol calcular aquest ordre es pot anar calculant les potències successives de σ fins a obtenir per primer cop la identitat. Naturalment, anar fent `mulperms(mulperms(...(`σ,σ`)),`σ`)` no és gaire pràctic. Aquest procés es pot simplificar si es defineix prèviament una funció `ms( )` que multipliqui qualsevol permutació per σ, d'aquesta manera només s'ha de fer `ms(ms(ms...(`σ`)))`. Maple preveu poder fer aquest tipus d'operació d'una manera ràpida utilitzant l'operador d'iteració `@@` que permet escriure `(ms@@n)(`σ`)`, on `n` és el nombre de cops que apliquem `ms`, per a abreujar l'operació anterior.

Exercici 6.9

Donada la permutació $\sigma = (2, 3, 5, 4, 6, 1, 8, 7)$ de S_8, definiu la funció `ms` que aplicada sobre una permutació qualsevol τ doni com a resultat la permutació producte de σ per τ.

Utilitzant la funció `ms` i l'operador `@@` determineu l'ordre (i totes les potències diferents) de σ.

7 Programació: lògica, iteracions i procediments

No sempre Maple tindrà les comandes que necessitem per a realitzar algun treball o càlcul concret, o bé ens agradaria iterar l'execució d'un paquet de comandes vàries vegades. En aquests casos necessitarem definir les nostres pròpies comandes (o funcions). Per això hem d'aprendre com crear procediments (comandes pròpies o blocs de programa) on podrem incloure funcions lògiques, iteracions i altres procediments.

Si ja heu programat en algun llenguatge informàtic fàcilment podreu reconèixer els operadors i funcions lògiques que apareixeran al llarg de la pràctica.

7.1 Lògica: `if-then-elif-else-end if`

La comanda bàsica que ens permetrà implementar raonaments lògics és la comanda `if` amb totes les seves variants. Farà que Maple executi diferents seqüències de comandes en funció del compliment d'unes condicions (que Maple avaluarà com certes o falses). El cas més simple correspon a `if...then...end if` que executarà una seqüència de comandes si es compleix una condició.

```
> restart;
> a:=3;
> if a=4 then b:=a+1 end if;
> if a=3 then b:=a end if;
```

Ara bé, aquest operador lògic `if` admet moltes variants que ens permetran combinar diferents condicions. Per exemple l'operador `else` que precedeix el codi que caldrà executar en el cas que no es compleixi la condició. També admet l'operador `elif` que ens permetrà considerar més d'una condició. En l'exemple següent podem veure el seu funcionament.

Exemple 7.1

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 utilitzant `if`.

```
> 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);
```

De fet, la funció que acabem de definir a l'exemple anterior ja existeix en Maple amb el nom de `signum()`.

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

El que hem fet és definir la funció `f` on a la seva definició hauríem de traduir `if` per "*si*", `elif` per "*altrament si*", `else` per "*altrament*" i `end if` per "*hem acabat*". Llavors podem pensar la comanda `if` com una comanda amb tres elements:

- El primer element que hem de tenir en compte és la sintaxi de la funció. Hem d'entrar els arguments `if ... then, elif ... then, else,` i `end if` sempre en aquest ordre. Els operadors `if..then..end if` són necessaris, la resta són opcionals.

- El segon és la introducció de condicions lògiques després dels corresponents `if`. Això són expressions com $x > 0$, $y \leq 3$, ... i que en Maple s'han d'introduir mitjançant els operadors relacionals següents:

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

  Quan Maple troba alguna d'aquestes expressions l'avalua com una "*expressió Booleana*", o sigui, com una expressió que pot ser "*certa*" o "*falsa*". Les avaluacions booleanes d'expressions també es poden realitzar directament mitjançant la funció `evalb( )`. Analitzeu el resultat obtingut a l'executar la següent línia de comanda:

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

  Podem fer construccions més complicades, combinant els operadors anteriors amb els operadors lògics `and`, `or`, `xor` i `not`. Podeu consultar l'ajuda per a obtenir més detalls sobre els operadors lògics escrivint `?boolean;`.

- El tercer element que formarà part d'una comanda lògica són les comandes que s'han d'executar en cada un dels casos que es poden produir. En l'exemple tan sols havíem d'introduir -1, 0 i 1, però es poden introduir expressions més complexes, Per exemple, volem definir una funció `g` que prengui valors segons la següent definició:

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

  Podem comprovar que funciona amb alguns exemples,

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

  Tot i això, què passa en el següent exemple?

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

  El missatge d'error diu que no ha pogut avaluar la condició booleana que hem introduït. Això és a causa del fet que `Pi` no és exactament un número. De fet, obtenim exactament el mateix error quan avaluem `g` a una variable indeterminada x:

  ```
  > g(x);
  ```

  Una manera de solucionar el problema és utilitzant la comanda `evalf()` i avaluar `g` en una aproximació numèrica de `Pi`.

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

  Finalment intentem ara representar gràficament 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` aquesta 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`. Recordeu

que no era necessària a l'hora d'usar la comanda `plot` per representar gràficament una funció (no una expressió).

```
> 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. Vegeu-ho en el següent exemple.

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

La funció `if` és molt útil en segons quins contexts, però si el que volem és definir una funció a trossos, no és la més apropiada com acabem de veure. Per a fer això és més versàtil la funció `piecewise( )`. Podem redefinir la funció `g` utilitzant aquesta nova comanda:

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

Fixeu-vos en els paràmetres que necessita, és a dir, hem d'introduir entre els parèntesis les corresponents seqüències de "condició, funció" separades per comes. Si escrivim una última funció sense cap condició prèvia, aquesta és la que executarà en cas que no es compleixin cap de les condicions anteriors en la definició.

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

```
> g(Pi); g(a);
> 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 comanda `piecewise`, mentre que reservarem `if-then-elif-else-end if` per a utilitzar-la en altres ocasions.

Exercici 7.1

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

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

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 amb l'ajuda de l'operador lògic `and`. (Encara que si s'ordenen adequadament les diferents condicions es pot escriure una comanda sense `and`).

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

Una iteració consisteix en executar una secció de codi vàries vegades. Moltes comandes a Maple contenen iteracions: les comandes `sum`, `fsolve`, ... La comanda `sum` és semblant a la comanda `seq` que ja coneixeu amb la diferència que retorna la suma total de la seqüència de valors obtinguts. Com a exemple, calculem la suma dels enters entre 1 i 100 elevats al quadrat:

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

En alguna part del procés Maple ha aplicat una iteració per a fer el càlcul. Tot i això algun cop necessitarem definir les nostres pròpies iteracions. Fet "a mà" hauríem fet:

Iteració suma:

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

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

Seguidament cal introduir la iteració. Per a això utilitzem les comandes `for... from... by... to... while... do... end do`. El bloc de comandes a iterar s'ha d'introduir sempre 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 grup de comandes.

```
> resp:=0; for i from 1 to 100 do
> resp:=resp + i^2; end do:
> resp;
```

Exercici 7.2

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

Més iteracions:

A vegades no sabrem el nombre exacte d'iteracions que volem fer sinó que la informació que tindrem serà que hem d'iterar un bloc de comandes fins que alguna condició fixada es compleixi. Per exemple, què passa si pitgeu indefinidament la tecla "*cosinus*" de la vostra calculadora, començant pel valor 5? A l'exemple següent 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 apropant a un valor determinat. De fet, podeu comprovar que aquest valor ha de complir l'equació $\cos(x) = x$. Tot i això, veureu que amb 20 cops no n'hi ha prou per a obtenir una successió que no variï (fixant 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 de la manera següent.

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

En qualsevol iteració l'única part obligatòria és el `do..end do`. Les parts `for` i `while` són opcionals i serveixen per controlar el número d'iteracions que cal fer per obtenir el resultat desitjat.

Les diferents maneres de realitzar iteracions de blocs de comandes es poden combinar, és a dir, podem tenir un `while` dins d'un `for`.

Exercici 7.3

Llegiu el següent bloc de comandes i abans d'executar-lo escriviu el que vosaltres creieu que serà la sortida. La comanda `isprime( )` és una funció Booleana que retorna el valor `true` o `false` segons si el número és primer o no.

```
> for p from 1000 by -1 while not isprime(p) do
  end do;
  p;
```

Hi ha una comanda que permet aturar una iteració en qualsevol punt del bloc de comandes que iterem: `break`. Observeu el següent exemple,

```
> for p from 1000 by -1 do
  if isprime(p) then break end if;
  end do;
  p;
```

7.3 Procediments

Al definir una funció pot passar que necessitem incloure-hi vàries iteracions, condicions lògiques, altres funcions, ... 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 (mentre aquesta s'executi), per buidar-se en acabar l'execució, 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`. El procediment o bloc de programa està definit per `proc ... end proc`. Al costat de `proc` hem de definir els paràmetres que el nostre procediment utilitzarà. Després de fixar els paràmetres, cal declarar les variables que utilitzarem, això es realitza amb la comanda `local` si són locals (es destruiran en acabar l'execució del programa) i/o amb la comanda `global` si són globals.

La millor manera d'entendre la sintaxi de definició d'un procediment és analitzant un exemple concret.

Exemple 7.2

Mireu l'estructura de la definició següent d'un procediment:

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

Exemple 7.3

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 tots els passos que segueix un procediment des dels paràmetres que li passem fins al resultat final, ho podeu fer amb la funció `trace`. Aquesta funció permet "*activar*" per pantalla els passos que hi ha dins de la funció `proc`.

Exemple 7.4

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);
```

7.4 Exercicis

Els exercicis següents s'han pensat per a que 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 7.4

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 un sol gràfic les funcions `cos(x)`, `aprcos(x,1)`, `aprcos(x,2)` i `aprcos(x,4)` per a `x` entre els valors -2π i 2π.

Exercici 7.5

Definiu una funció que depengui de tres paràmetres `aprox(fn,x,iteracions)`, on `fn` és el nom d'una funció qualsevol, `x` un valor i iteracions un enter, i que retorni una llista amb els valors de la forma `[x + (1/i),fn(x + (1/i) )]` per a `i=1..iteracions`.

Avalueu `aprox` a la funció $\dfrac{\sin(x)}{x}$, per a $x = 0$ amb 50 iteracions.

Modifiqueu la funció `aprox` per a que a més retorni un dibuix dels punts calculats.

Exercici 7.6

Definiu una funció `ordre` tal que donada una permutació σ en calculi el seu ordre determinant el primer nombre natural n tal que σ^n és la permutació identitat.

Calculeu l'ordre de les permutacions $\sigma_1 = (1,3,5)(2,4)$ i $\sigma_2 = (1,2,3,4,5)(6,7,8)$ utilitzant la funció `ordre` que acabeu de definir.

Exercici 7.7

Feu un procediment que donat n faci la llista de totes les permutacions de n elements, cada una en format de llista (és a dir, la permutació σ es representa per la llista `[`σ`(1), ...`σ`(n)]`).

8 Programació recursiva. La successió de Fibonacci

8.1 Programació recursiva

Un procediment es diu recursiu si es crida a ell mateix d'una manera directa o indirecta. En aquesta mena de programació heu d'anar molt en compte ja que podríeu crear un programa que es cridés a ell mateix indefinidament per això heu de tenir clares dues coses: l'algorisme recursiu i les condicions que fan que la recursió s'acabi.

L'exemple més clàssic és el de crear un procediment que calculi el factorial d'un número natural, $n!$. Ja sabeu que Maple té una comanda que el calcula directament però tot i això crearem la nostra comanda pròpia `Factorial( )`.

El factorial d'un número enter positiu es defineix com

1. $0! = 1$.
2. $n! = n \cdot (n-1)!$ si $n > 0$.

Observeu que aquesta és una fórmula recursiva amb una condició inicial.

Exercici 8.1

Construïu un procediment `Factorial` que calculi el factorial segons la fórmula anterior. A més, feu que aparegui un error per pantalla quan $n < 0$. Per escriure missatges per pantalla podeu fer servir les comandes `printf` o `print`, amb l'ajuda de Maple observeu les seves diferències.

Amb la comanda `trace` observeu què passa quan calculeu el factorial de -1, 0 i 4. En l'últim cas veureu com el procediment es crida diferents vegades.

8.2 La successió de Fibonacci

La successió dels nombres de Fibonacci f_n és la que s'obté a partir de les condicions $f_1 = f_2 = 1$ i $f_n = f_{n-1} + f_{n-2}$ per a n més gran o igual a 3.

Exercici 8.2

Definiu una funció `fibon(n)` que doni com a resultat el `n`-èssim nombre de Fibonacci. Comproveu que funciona correctament calculant `fibon(-1)`, `fibon(0)`, `fibon(1)`, `fibon(2)`, `fibon(3)` i `fibon(4)`.

Fibonacci va descobrir aquesta successió de números l'any 1202 quan se li va proposar el següent problema que consistia en analitzar la velocitat amb que es reproduïen els conills en condicions ideals. Més concretament, el problema era el següent. Suposem que en un camp tancat s'hi

introdueix una parella de conills (mascle i femella) acabats de néixer. Els conills s'aparellen per primer cop al cap d'un mes d'haver nascut i neix una nova parella al final del següent mes. A partir de llavors les parelles s'aparellen i crien cada mes. Suposem que els nostres conills no es moren i que a cada cria obtenim una nova parella (un mascle i una femella). La pregunta que se li va fer a Fibonacci és: quantes parelles tindrem al final del primer any? Anem a comptar.

Al final del primer mes només tinc una parella que s'han aparellat.

Al final del segon mes la femella produeix una nova parella així doncs en tenim dues.

Al final del tercer mes, tenim les dues parelles anteriors més una nova parella que ha produït la femella original (la nova femella encara és al seu primer més i no ha criat), així en portem tres.

Al final del quart mes tenim les tres parelles anteriors més dues noves parelles que han nascut de les femelles del segon més (les femelles nascudes al mes anterior encara no han criat).

I així, successivament. Podeu observar la relació d'aquest problema amb la successió de números que hem definit a l'exercici 8.2?

Exercici 8.3

Quina és la resposta al problema de Fibonacci?

Exercici 8.4

Feu un gràfic dels nombres `fibon(n)` per a `n=3..20`. és a dir, representeu els punts de la forma `[n,fibon(n)]`.

Exercici 8.5

Feu un gràfic de la funció `ln(fibon(n))` pels mateixos valors de l'exercici anterior. Què observeu? A partir d'aquesta gràfica, què podeu deduir del tipus de creixement dels valors de l'exercici anterior?

Exercici 8.6

Que podem deduir del gràfic del quocient `fibon(n+1)/fibon(n)` per a `n=1..50`? Aquests nombres representen la proporció entre dos números consecutius a la successió de Fibonacci. Feu una llista amb aquests valors. Què observeu?

A l'exercici anterior haureu observat que la proporció entre dos números consecutius a la successió de Fibonacci estabilitza cap a un número. Aquest número s'anomena el número auri o la raó aurea. Si teniu curiositat podeu buscar informació sobre aquest número i veureu que apareix a molts fenòmens naturals.

Exercici 8.7

Analitzant les gràfiques vistes fins ara, que podríem deduir dels valors de la successió `fibon(n)`?

Exercici 8.8

La successió de Fibonacci ha estat definida de forma recursiva però també té una fórmula pel terme general que ens calcula el terme n-éssim directament. Utilitzeu la funció `rsolve` (mireu l'ajuda de Maple per veure com funciona) per a trobar el terme general de la successió de Fibonacci.

Observeu que passa quan es substitueixen diferents valors de n a l'expressió que retorna la comanda `rsolve` i en calculeu una aproximació numèrica.

Calculeu una aproximació numèrica de $\dfrac{1+\sqrt{5}}{2}$ i compareu-la amb el valor obtingut a l'exercici 8.6.

Exercici 8.9

Què tenen a veure aquests valors amb el quocient de les longituds dels costats del DNI o d'una tarja de crèdit?