Κεφάλαιο 15

Διαφορικές Εξισώσεις

Στο κεφάλαιο αυτό θα εξετάσουμε τις εντολές που μας επιτρέπουν να λύσουμε διαφορικές εξισώσεις συνήθεις ή με μερικές παραγώγους, αναλυτικά αλλά και αριθμητικά. Θα παρουσιάσουμε τον τρόπο με τον οποίο μπορούμε να παραστήσουμε γραφικά το σύνολο των λύσεων, καθώς και το πεδίο διευθύνσεων μιας διαφορικής εξίσωσης. Στο τέλος του κεφαλαίου παρουσιάζουμε ορισμένες λυμένες ασκήσεις.

15.1 Συνήθεις Διαφορικές Εξισώσεις

Για την επίλυση μιας διαφορικής εξίσωσης ή ενός συστήματος διαφορικών εξισώσεων το Maple είναι εφοδιασμένο με την εντολή `dsolve`. Η ίδια εντολή εφαρμόζεται είτε έχουμε αρχικές είτε συνοριακές συνθήκες.

Περιγραφή:	***Επιλύει μια διαφορική εξίσωση, ένα πρόβλημα αρχικών τιμών-συνοριακών τιμών.***	
Γενική σύνταξη:	***dsolve(ode, y(x), args)*** ***dsolve({ode,ics}, y(x), args)***	
Παράμετροι:	ode	Η διαφορική εξίσωση (Σύστημα διαφορικών εξισώσεων {ode1,ode2,...}).
	ics	Αρχικές, συνοριακές συνθήκες.
	y(x)	Η άγνωστη συνάρτηση.
	args	▪ **'implicit'** αν θέλουμε τη λύση σε πεπλεγμένη μορφή. ▪ **'explicit'** – αν θέλουμε τη λύση σε μη πεπλεγμένη μορφή (η solve, αν δεν ορισθεί διαφορετικά ,προσπαθεί να δώσει τη λύση σε μη πεπλεγμένη μορφή). ▪ **'parametric'** – για την επίλυση μόνο με παραμετρικό scheme (μόνο για πρώτης τάξης διαφορικές εξισώσεις).

Στα επόμενα παραδείγματα θα εξετάσουμε διαφορικές εξισώσεις και συστήματα τόσο με αρχικές συνθήκες όσο και με συνοριακές.

Παράδειγμα: 15-1

Να βρεθεί η γενική λύση της διαφορικής εξίσωσης $y' = y^2 + \frac{1}{x}y - \frac{3}{x^2}$.

Λύση:

Αρχικά, για ευκολία θα ονομάσουμε τη διαφορική εξίσωση ως ode.

```
> ode:=diff(y(x),x)=y(x)^2+(1/x)*y(x)+3/x^2;
```

$$ode := \frac{d}{dx}y(x) = y(x)^2 + \frac{y(x)}{x} + \frac{3}{x^2}$$

Χρησιμοποιώντας την εντολή `dsolve` θα αναζητήσουμε μια αναλυτική γενική λύση:

```
> dsolve(ode,y(x));
```

$$\{y(x) = \frac{-1 - \sqrt{2}\,\tan(-\sqrt{2}\,\ln(x) + \sqrt{2}\,\_C1)}{x}\}$$

Το Maple μας επιστρέφει τη γενική λύση της διαφορικής εξίσωσης χωρίς όμως να γνωρίζουμε ποια από τις μεθόδους επίλυσης ακολούθησε. Τη δυνατότητα να γνωρίσουμε με ποια μέθοδο το πρόγραμμα λύνει μια διαφορική εξίσωση μας την δίνει η εντολή `infolevel`. Έτσι, αν ορίσουμε το επίπεδο πληροφορίας στο 3 (υπάρχουν 4 επίπεδα πληροφοριών που μπορεί να μας δώσει το πρόγραμμα),

```
>infolevel[dsolve]:= 3;
```

$$infolevel_{dsolve} := 3$$

και ζητήσουμε ξανά τη γενική λύση της διαφορικής εξίσωσης, τότε θα μας πληροφορήσει το Maple ότι η διαφορική είναι ομογενής.

```
>dsolve(ode,y(x));
Methods for first order ODEs:
--- Trying classification methods ---
trying a quadrature
trying 1st order linear
trying Bernoulli
trying separable
trying inverse linear
trying homogeneous types:
trying homogeneous G
<- homogeneous successful
```

$$y(x) = \frac{-1 - \sqrt{2}\,\tan(-\sqrt{2}\,\ln(x) + \sqrt{2}\,\_C1)}{x}$$

Αν θέλουμε τη λύση σε πεπλεγμένη μορφή, μπορούμε να προσθέσουμε την παράμετρο `implicit`. Έτσι, θα έχουμε:

```
> dsolve(ode,y(x),'implicit');
```

$$\ln(x) - \_C1 - \frac{1}{2}\sqrt{2}\,\arctan\left(\left(\frac{1}{2}\,y(x)\,x + \frac{1}{2}\right)\sqrt{2}\right) = 0$$

□

Παράδειγμα: 15-2

Να λυθεί η διαφορική εξίσωση $xy' - 4y = x^2\sqrt{y}$.

Λύση:

Δίνουμε την εντολή `restart`, για να μηδενίσουμε όλες τις παραμέτρους του προγράμματος και εξετάζουμε την κατηγορία της διαφορικής εξίσωσης που μας δίνεται. Έτσι, ξεκινάμε ορίζοντας το επίπεδο πληροφοριών για την εντολή `dsolve` στο 3.

```
> infolevel[dsolve]:= 3;
```

$$infolevel_{dsolve} := 3$$

Στη συνέχεια ορίζουμε την εξίσωση.

```
> ode:=x*diff(y(x),x)-4*y(x)=x^2*sqrt(y(x));
```

$$ode := x\left(\frac{d}{dx}\,y(x)\right) - 4\,y(x) = x^2\sqrt{y(x)}$$

Τη λύση της διαφορικής εξίσωσης θα την ονομάσουμε με τη μεταβλητή sol, για να μπορέσουμε στη συνέχεια να κάνουμε ένα είδος επαλήθευσης. Έτσι, έχουμε την εντολή:

```
> sol:=dsolve(ode,y(x));
Methods for first order ODEs:
--- Trying classification methods ---
trying a quadrature
trying 1st order linear
trying Bernoulli
<- Bernoulli successful
```

$$sol := \sqrt{y(x)} - \left(\frac{1}{2}\ln(x) + \_C1\right)x^2 = 0$$

Η διαφορική εξίσωση αναγνωρίστηκε ως Bernoulli και μας δόθηκε η λύση της σε πεπλεγμένη μορφή.

Με την εντολή `odetest` μπορούμε να ελέγξουμε το αποτέλεσμα της εντολής `dsolve`. Έτσι, αν καλέσουμε την εντολή για τη λύση sol, βλέπουμε ότι το πρόγραμμα επιστρέφει την τιμή 0, πράγμα που σημαίνει ότι η λύση sol ικανοποιεί τη διαφορική εξίσωση.

```
> odetest(sol,ode,y(x));
```

$$0$$

Περιγραφή:	***Ελέγχει το αποτέλεσμα της εντολής dsolve για μια διαφορική εξίσωση ή ένα σύστημα.***	
Γενική σύνταξη:	***odetest(sol, ode, y(x))***	
Παράμετροι:	ode	Η διαφορική εξίσωση.
	sol	Το όνομα της λύσης.
	y(x)	Η άγνωστη συνάρτηση .

Στο επόμενο παράδειγμα θα εξετάσουμε ένα πρόβλημα αρχικών τιμών.

Παράδειγμα: 15-3

Να λυθεί το πρόβλημα αρχικών τιμών $\begin{cases} xy' + xy + y - 1 = 0 \\ y(1) = 1 \end{cases}$.

Λύση:

Ορίζουμε τη διαφορική εξίσωση.

```
> ode:=x*diff(y(x),x)+x*y(x)+y(x)-1=0;
```

$$ode := x\left(\frac{d}{dx}\mathrm{y}(x)\right) + x\,\mathrm{y}(x) + \mathrm{y}(x) - 1 = 0$$

Η γενική λύση της διαφορικής εξίσωσης βρίσκεται ως εξής:

```
> dsolve(ode,y(x));
```

$$\mathrm{y}(x) = \frac{1}{x} + \frac{\mathrm{e}^{(-x)}\_C1}{x}$$

Καλούμε την εντολή `dsolve` προσθέτοντας ως παράμετρο την αρχική συνθήκη και έτσι βρίσκουμε μια μερική λύση της διαφορικής εξίσωσης.

```
> dsolve({ode,y(1)=1},y(x));
```

$$y(x) = \frac{1}{x}$$

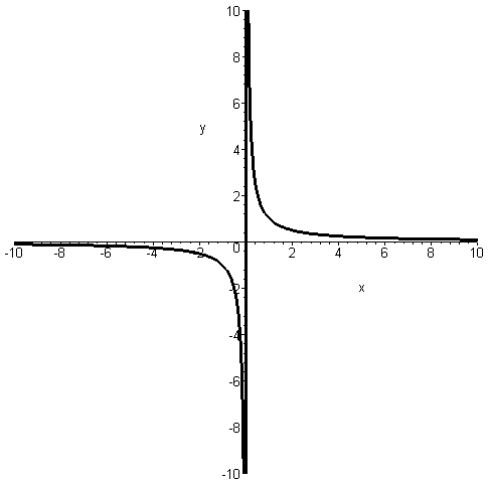
Αν θελήσουμε να σχεδιάσουμε τη γραφική παράσταση της λύσης, δηλαδή να σχεδιάσουμε την ολοκληρωτική καμπύλη του προβλήματος, πρέπει αρχικά να μετατρέψουμε τη λύση σε συνάρτηση και στη συνέχεια με την εντολή `plot` να δημιουργήσουμε τη γραφική της παράσταση, όπως ήδη γνωρίζουμε από το κεφάλαιο 10.

Έτσι, ονομάζουμε με p τη συνάρτηση της λύσης (το δεξιό μέρος).

```
> p:=unapply(rhs(%),x);
```

$$p := x \to \frac{1}{x}$$

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



15.2 Οικογένεια Ολοκληρωτικών Καμπυλών

Αν έχουμε τη γενική λύση μιας διαφορικής εξίσωσης και θέλουμε να αναπαραστήσουμε τις ολοκληρωτικές καμπύλες για διάφορες τιμές των παραμέτρων, μπορούμε να ακολουθήσουμε τη διαδικασία που περιγράφεται στο παρακάτω παράδειγμα.

Παράδειγμα: 15-4

Να λυθεί η διαφορική εξίσωση $y'(x) = x \cdot y(x)$ και να παρασταθούν σε ένα σύστημα αξόνων οι ολοκληρωτικές καμπύλες.

Λύση:

Ορίζουμε ως ode τη διαφορική εξίσωση.

```
> ode:=diff(y(x),x)=x*y(x);
```

$$ode := \frac{d}{dx}\mathrm{y}(x) = x\,\mathrm{y}(x)$$

Ορίζουμε ώστε η συνάρτηση που ονομάζουμε f να είναι το δεξιό μέρος της λύσης της διαφορικής εξίσωσης.

```
> f:=unapply(rhs(dsolve(ode)),x);
```

$$f := x \to \_C1\ \mathrm{e}^{(1/2\,x^2)}$$

Δημιουργούμε τις εκφράσεις f1,f2,f3,f4 όπου κάθε μια είναι η συνάρτηση f για διάφορες τιμές της παραμέτρου_C1 (ενδεικτικά θέτουμε τις τιμές -2,-1,1,2).

```
> f1:=subs(_C1=1,f(x));
```

$$f1 := \mathrm{e}^{\left(\frac{x^2}{2}\right)}$$

```
> f2:=subs(_C1=2,f(x));
```

$$f2 := 2\,\mathrm{e}^{\left(\frac{x^2}{2}\right)}$$

```
> f3:=subs(_C1=-1,f(x));
```

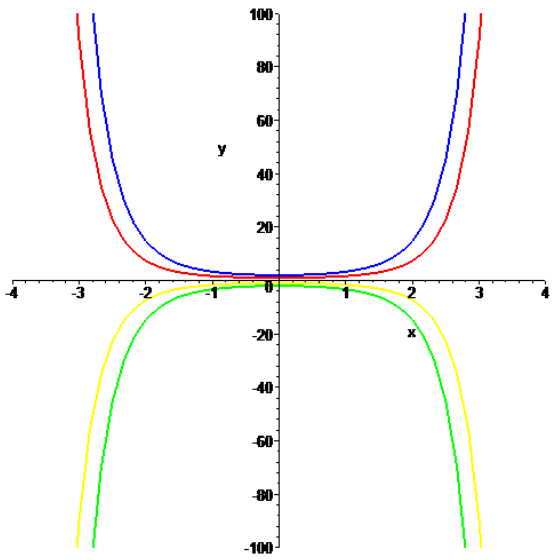
$$f3 := -\mathrm{e}^{\left(\frac{x^2}{2}\right)}$$

```
> f4:=subs(_C1=-2,f(x));
```

$$f4 := -2\, e^{\left(\frac{x^2}{2}\right)}$$

Δημιουργούμε στο ίδιο σύστημα αξόνων τις γραφικές παραστάσεις των p1,p2,p3,p4.

```
>plot({f1,f2,f3,f4},x=-4..4,y=-100..100,thickness=2);
```



Ένας άλλος τρόπος για να δημιουργήσουμε μια ακολουθία λύσεων για διάφορες τιμές των παραμέτρων είναι χρησιμοποιώντας την εντολή `seq` να δημιουργήσουμε μια ακολουθία παραμέτρων c και στη συνέχεια με την εντολή map να δημιουργήσουμε μια λίστα λύσεων της διαφορικής εξίσωσης :

```
>restart;
```

Ορίζουμε ως ode την διαφορική εξίσωση.

```
>ode:=diff(y(x),x)=x*y(x);
```

$$ode := \frac{d}{dx}\,y(x) = x\,y(x)$$

Ορίζουμε ώστε η συνάρτηση f να είναι το δεξιό μέρος της λύσης της διαφορικής εξίσωσης.

```
> f:=unapply(rhs(dsolve(ode)),x);
```

$$f := x \to \_C1\, e^{(1/2\, x^2)}$$

Κατασκευάζουμε μια ακολουθία τιμών για την παράμετρο _C1.

```
> c:=[seq(_C1=i,i=-6..6)];
```

$$c := [\_C1 = -6, \_C1 = -5, \_C1 = -4, \_C1 = -3, \_C1 = -2, \_C1 = -1, \_C1 = 0, \_C1 = 1, \_C1 = 2, \_C1 = 3, \_C1 = 4, \_C1 = 5, \_C1 = 6]$$

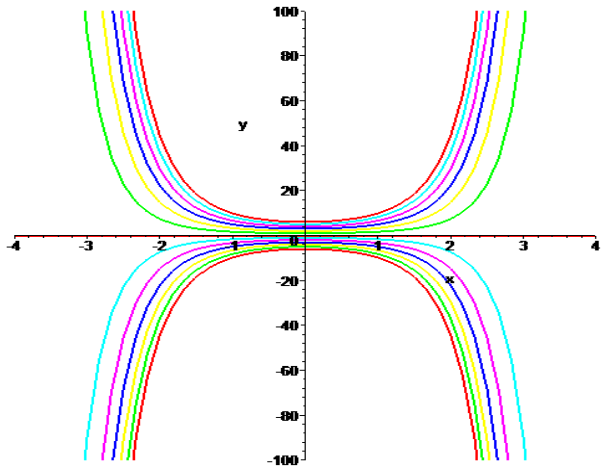
Δημιουργούμε μια λίστα από ολοκληρωτικές καμπύλες για τις διάφορες τιμές της παραμέτρου.

```
> oloklirotikes:=map(subs,c,f(x));
```

$$oloklirotikes := \left[-6\, e^{\left(\frac{x^2}{2}\right)}, -5\, e^{\left(\frac{x^2}{2}\right)}, -4\, e^{\left(\frac{x^2}{2}\right)}, -3\, e^{\left(\frac{x^2}{2}\right)}, -2\, e^{\left(\frac{x^2}{2}\right)}, -e^{\left(\frac{x^2}{2}\right)}, 0, e^{\left(\frac{x^2}{2}\right)}, 2\, e^{\left(\frac{x^2}{2}\right)}, 3\, e^{\left(\frac{x^2}{2}\right)}, 4\, e^{\left(\frac{x^2}{2}\right)}, 5\, e^{\left(\frac{x^2}{2}\right)}, 6\, e^{\left(\frac{x^2}{2}\right)}\right]$$

Σχηματίζουμε τη γραφική παράσταση.

```
> plot(oloklirotikes,x=-4..4,y=-100..100, thickness=2);
```



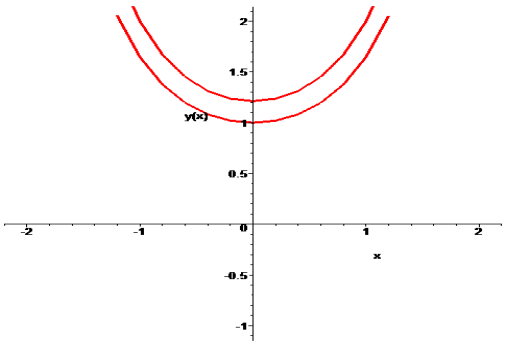
Ωστόσο, το Maple μας δίνει την εντολή `DEplot` του πακέτου `Detools` για τη δημιουργία ολοκληρωτικών καμπυλών. Η γενική σύνταξη της εντολής είναι:

Περιγραφή:	***Δημιουργεί τη γραφική αναπαράσταση των ολοκληρωτικών καμπυλών για ένα σύνολο αρχικών συνθηκών.***	
Γενική σύνταξη:	***DEplot(ode,y(x),x=a..b,y=c..d,[[ic]])***	
Παράμετροι:	ode	Η διαφορική εξίσωση.
	y(x)	Η άγνωστη συνάρτηση.
	x=a..b, y=c..d	Τα διαστήματα μεταβολή του x και y.
	ic	Αρχικές συνθήκες.
Πακέτο:	***DEtools***	

Έτσι, αν φορτώσουμε αρχικά το πακέτο `Detools` και καλέσουμε την εντολή `Deplot` για αρχικές συνθήκες $y(0)=1$ και $y(1)=2$, θα έχουμε δυο ολοκληρωτικές καμπύλες που αντιστοιχούν στα προβλήματα αρχικών τιμών σχεδιασμένες μέσα στο πεδίο διευθύνσεων της διαφορικής εξίσωσης.

```
> with(DEtools):
```

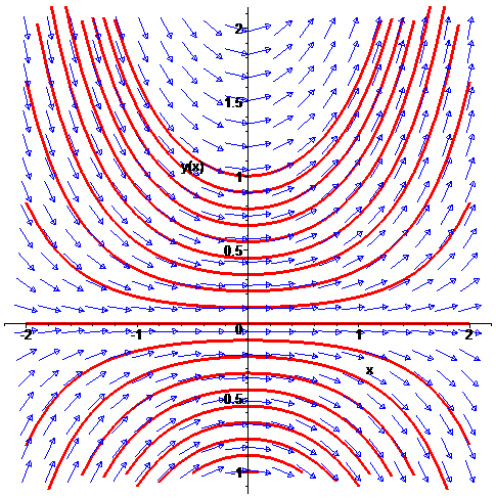
```
> DEplot(ode,y(x),x=-2..2,y=-1..2, [[y(0)=1], [y(1)=2]],
linecolour=red,arrows=NONE);
```



Για το σχεδιασμό μπορούμε να χρησιμοποιήσουμε παραμέτρους γνωστές και από την εντολή plot, για να ορίσουμε το χρώμα των γραμμών σε κόκκινο. Επίσης, ορίσαμε, με την παράμετρο `arrow=NONE`, να μην εμφανίζονται τα βέλη του πεδίου διευθύνσεων. Οι δυνατές τιμές της παραμέτρου arrows είναι `SMALL,MEDIUM, LARGE,LINE,NONE.`

Έχουμε, επίσης, τη δυνατότητα να ορίσουμε αρχικές συνθήκες με μια παράμετρο έτσι ώστε να σχεδιάζονται περισσότερες από μια ολοκληρωτικές καμπύλες. Εδώ χρησιμοποιούμε την παράμετρο k με τιμές από -9 έως 9. Επίσης, σχεδιάζουμε συγχρόνως μεσαίου μεγέθους βέλη για το πεδίο διευθύνσεων του προβλήματος.

```
> with(DEtools):
>DEplot(ode,y(x),x=-2..2,y=-1..2, [[ y(0) = k/9] $ k = -
9..9 ],linecolour=red, color = blue,
stepsize=.1,arrows=MEDIUM);
```

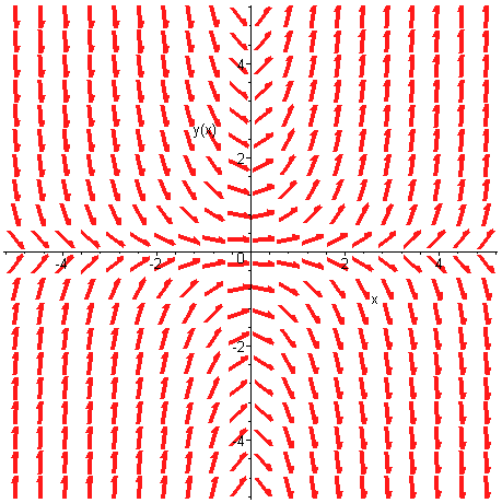


Αν θελήσουμε να σχεδιάσουμε το πεδίο διευθύνσεων για τη διαφορική εξίσωση πρέπει να χρησιμοποιήσουμε την εντολή `dfieldplot` του πακέτου εντολών `DEtools`.

Περιγραφή:	***Δημιουργεί το πεδίο διευθύνσεων της λύσης μιας διαφορικής εξίσωσης.***	
Γενική σύνταξη:	***dfieldplot(ode,y(x),x=a..b,y=c..d)***	
Παράμετροι:	ode	Η διαφορική εξίσωση.
	y(x)	Η άγνωστη συνάρτηση.
	x=a..b, y=c..d	Τα διαστήματα μεταβολής του x και y.
Πακέτο:	***DEtools***	

Η εντολή `dfieldplot` δέχεται γενικά όποιες παραμέτρους δέχεται και η εντολή plot, για να ορίσουμε τίτλο, χρώμα κ.ά.

```
> dfieldplot(ode, y(x), x=-5..5,y=-5..5,thickness=4);
```



Παράδειγμα: 15-5

Να λυθεί η δευτέρας τάξης διαφορική εξίσωση $y''(x)-y(x)=0$ και να παρασταθούν σε ένα σύστημα αξόνων οι ολοκληρωτικές καμπύλες.

Λύση:

Ορίζουμε τη διαφορική εξίσωση:

```
> ode:=diff(y(x),x$2)-y(x)=0;
```

$$ode := \left(\frac{d^2}{dx^2}\,y(x)\right) - y(x) = 0$$

Βρίσκουμε τη γενική της λύση.

```
> dsolve(ode,y(x));
```

$$y(x) = \_C1\, e^{x} + \_C2\, e^{(-x)}$$

Αν θεωρήσουμε τώρα το αντίστοιχο πρόβλημα αρχικών τιμών με αρχικές συνθήκες $\begin{matrix} y(0)=1 \\ y'(0)=1 \end{matrix}$, θα βρούμε τη μερική λύση του προβλήματος και θα σχεδιάσουμε την ολοκληρωτική καμπύλη που προκύπτει.

Ορίζουμε ως ic τις αρχικές συνθήκες. Εδώ οι αρχικές συνθήκες είναι δύο. Έτσι, τις ορίζουμε ως ακολουθία εκφράσεων χωρισμένες με κόμμα.

```
> ic:=y(0)=0,D(y)(0)=1;
```

$$ic := y(0) = 0, D(y)(0) = 1$$

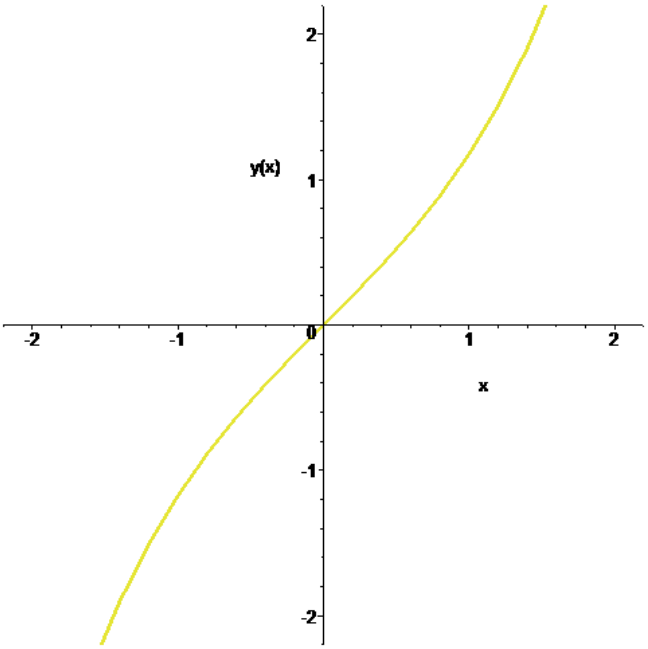
Χρησιμοποιώντας την εντολή `dsolve` λύνουμε το πρόβλημα αρχικών τιμών.

```
> dsolve({ode,ic},y(x));
```

$$y(x) = \frac{1}{2}e^{x} - \frac{1}{2}e^{(-x)}$$

Καλούμε την εντολή `DEplot` του πακέτου `Detools` για τη γραφική παράσταση της ολοκληρωτικής καμπύλης.

```
> with(DEtools):
>DEplot(ode,y(x),x=-2..2,y=-2..2,[[ic]]);
```

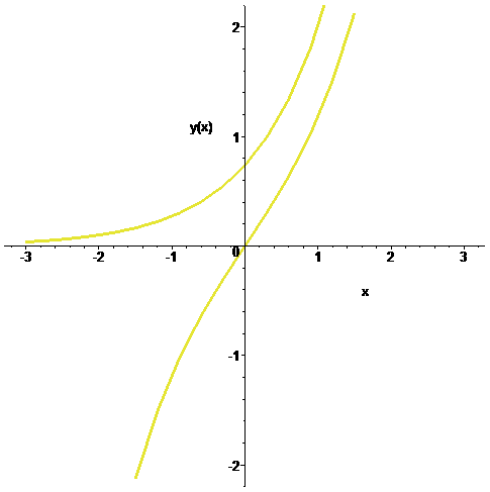


Αν τώρα θεωρήσουμε άλλες αρχικές συνθήκες, της ic1 μπορούμε να δούμε και τις δύο ολοκληρωτικές καμπύλες στο ίδιο σύστημα αξόνων.

```
> ic1:=y(1)=2,D(y)(1)=2;
```

$$ic1 := y(1) = 2, D(y)(1) = 2$$

```
> DEplot(ode,y(x),x=-3..3,y=-2..2,[[ic],[ic1]]);
```



Παράδειγμα: 15-6

Να λυθεί το σύστημα των διαφορικών εξισώσεων $\begin{cases} y'(x) = -y(x) - x \\ y'(x) = y(x) \end{cases}$.

Λύση:

Ορίζουμε τις δύο διαφορικές εξισώσεις:

```
> ode1 := diff(y(t),t)=-y(t)-x(t);
```

$$ode1 := \frac{d}{dt} y(t) = -y(t) - x(t)$$

```
> ode2 := diff(x(t),t)=y(t);
```

$$ode2 := \frac{d}{dt} x(t) = y(t)$$

Ορίζουμε τις αρχικές συνθήκες.

```
> ic1:= x(0)=0, y(0)=1;
```

$$ic1 := x(0) = 0, y(0) = 1$$

Η γενική λύση του συστήματος δίνεται από την εντολή:

```
> dsolve({ode1,ode2}, {x(t),y(t)});
```

$$\{x(t) = e^{\left(-\frac{t}{2}\right)}\left(\_C1 \sin\left(\frac{\sqrt{3}\,t}{2}\right) + \_C2 \cos\left(\frac{\sqrt{3}\,t}{2}\right)\right), y(t) = \frac{1}{2} e^{\left(-\frac{t}{2}\right)}$$
$$\left(-\_C1 \sin\left(\frac{\sqrt{3}\,t}{2}\right) + \_C1 \cos\left(\frac{\sqrt{3}\,t}{2}\right)\sqrt{3} - \_C2 \cos\left(\frac{\sqrt{3}\,t}{2}\right) - \_C2 \sin\left(\frac{\sqrt{3}\,t}{2}\right)\sqrt{3}\right)\}$$

Προσθέτοντας και τις αρχικές συνθήκες έχουμε:

```
> dsolve({ode1,ode2,ic1}, {x(t),y(t)});
```

$$\{y(t) = \frac{1}{2} e^{\left(-\frac{t}{2}\right)}\left(-\frac{2}{3}\sqrt{3} \sin\left(\frac{\sqrt{3}\,t}{2}\right) + 2 \cos\left(\frac{\sqrt{3}\,t}{2}\right)\right), x(t) = \frac{2}{3} e^{\left(-\frac{t}{2}\right)} \sqrt{3} \sin\left(\frac{\sqrt{3}\,t}{2}\right)\}$$

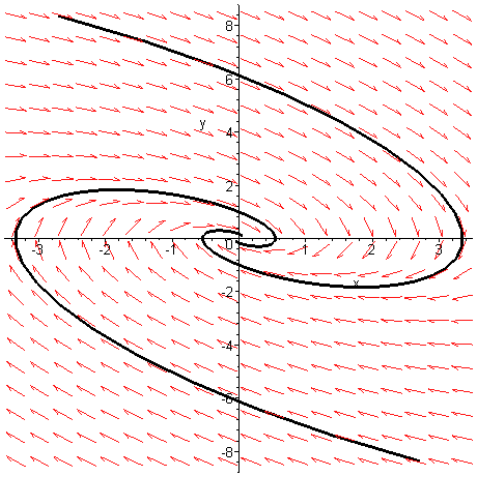
```
> ic2 := x(0)=0, y(0)=-1;
```

$$ic2 := x(0) = 0, y(0) = -1$$

Σχεδιάζουμε τις ολοκληρωτικές καμπύλες.

```
> with(DEtools):
```

```
>DEplot({ode1,ode2}, [x(t),y(t)], -4..4,[[ic1], [ic2]],
linecolor=black);
```



□

Παράδειγμα: 15-7

Να λυθεί το πρόβλημα συνοριακών τιμών $y''+2y'-10y=e^{-x}$ με $y(0)=2, y(1)=-5$.

Λύση:

Ορίζουμε και σε αυτό το παράδειγμα ως ode τη διαφορική εξίσωση.

```
> ode:=diff(y(x),x$2)+2*diff(y(x),x)-10*y(x)=exp(-x) ;
```

$$ode := \left(\frac{d^2}{dx^2}\,y(x)\right) + 2\left(\frac{d}{dx}\,y(x)\right) - 10\,y(x) = e^{(-x)}$$

Ορίζουμε τις συνοριακές συνθήκες.

```
> bc:=y(0)=2,y(1)=-5;
```

$$bc := y(0) = 2,\ y(1) = -5$$

Η λύση της διαφορικής εξίσωσης είναι:

```
> dsolve({ode,bc},y(x));
```

$$y(x) = -\frac{1}{11}\,\frac{e^{((-1+\sqrt{11})x)}\,(55\,e^{(\sqrt{11})}\,e + 23 - e^{(\sqrt{11})})}{(e^{(\sqrt{11})})^2 - 1} + \frac{1}{11}\,\frac{e^{(-(1+\sqrt{11})x)}\,e^{(\sqrt{11})}\,(55\,e - 1 + 23\,e^{(\sqrt{11})})}{(e^{(\sqrt{11})})^2 - 1} - \frac{1}{11}\,e^{(-x)}$$

□

Παράδειγμα: 15-8

Να λυθεί το σύστημα των διαφορικών εξισώσεων $\begin{cases} x'(t) = -10x(t) + 10y(t) \\ y'(x) = 28x(t) - y(t) - x(t)z(t) \\ z'(t) = -\frac{8}{3}z(t) + x(t)y(t) \end{cases}$.

Λύση:

Ορίζουμε τις διαφορικές εξισώσεις:

```
> ode1:=diff(x(t),t)=-10*x(t)+10*y(t);
```

$$ode1 := \frac{d}{dt}\,x(t) = 10\,y(t) - 10\,x(t)$$

```
> ode2:=diff(y(t),t)=28*x(t)- y(t) -x(t)*z(t);
```

$$ode2 := \frac{d}{dt}\,y(t) = 28\,x(t) - y(t) - x(t)\,z(t)$$

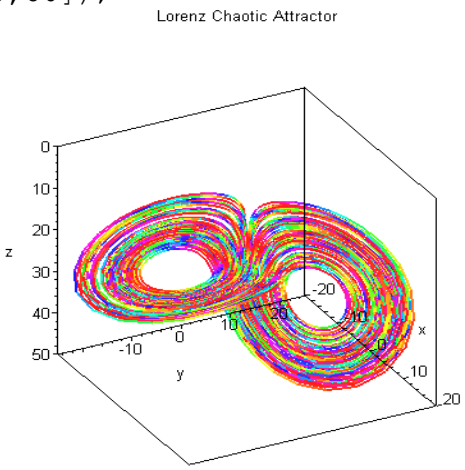
```
> ode3:=diff(z(t),t)=x(t)*y(t)-(8/3)*z(t);
```

$$ode3 := \frac{d}{dt}\,z(t) = x(t)\,y(t) - \frac{8}{3}\,z(t)$$

Περιγραφή:	***Δημιουργεί την γραφική αναπαράσταση της λύσης ενός συστήματος διαφορικών εξισώσεων.***	
Γενική σύνταξη:	***DEplot3d({ode1,ode2,...},{x(t),y(t),..},t=a..b,y=c..d,[[ic]])***	
Παράμετροι:	ode1,ode2	Οι διαφορικές εξισώσεις
	x(t),y(t)	Οι άγνωστες συναρτήσεις
	t=a..b	Το διάστημα μεταβολής του t
	ic	Αρχικές συνθήκες.
Πακέτο:	***DEtools***	

Ενεργοποιούμε το πακέτο DEtools

```
> with(DEtools):
>DEplot3d({ode1,ode2,ode3}, {x(t),y(t),z(t)}, t=0..100,
[[x(0) = 10, y(0)= 10,z(0)= 10]], stepsize=.02, x = -
20..20, y=-25..25,z= 0..50, linecolour=sin(10*t),
thickness = 2,title=`Lorenz Chaotic Attractor`,
orientation=[-30,80]);
```



15.3 Αριθμητική Επίλυση Διαφορικών Εξισώσεων

Χρησιμοποιώντας την εντολή `dsolve` μπορούμε να λύσουμε ένα πρόβλημα αρχικών τιμών ή συνοριακών τιμών αριθμητικά, θέτοντας την παράμετρο `numeric`. Το Maple χρησιμοποιεί από μόνο του εκείνη τη μέθοδο που θεωρεί κατάλληλη. Βέβαια, μας δίνεται η δυνατότητα, αν θέλουμε, να προσδιορίσουμε τη μέθοδο που θα χρησιμοποιηθεί ορίζοντας την παράμετρο `method=`, με δυνατές τιμές αυτές που φαίνονται στον παρακάτω πίνακα.

rkf45	Βρίσκει την αριθμητική λύση χρησιμοποιώντας την Fehlberg τέταρτης τάξης Runge-μέθοδο με τέταρτου βαθμού παρεμβολή.
rosenbrock	Βρίσκει την αριθμητική λύση χρησιμοποιώντας την πεπλεγμένη Rosenbrock τρίτης τάξης Runge-Kutta μέθοδο με τρίτου βαθμού παρεμβολή.
dverk78	Βρίσκει την αριθμητική λύση χρησιμοποιώντας την έβδομης τάξης Runge-Kutta μέθοδο.
lsode	Βρίσκει την αριθμητική λύση χρησιμοποιώντας τη Livermore Stiff ODE solver.
gear	Η μέθοδος bstoer με επιλογή τη Burlirsch-Stoer μέθοδο.
taylorseries	Βρίσκει την αριθμητική λύση χρησιμοποιώντας τη μέθοδο σειρών του Taylor.
classical	Βρίσκει την αριθμητική λύση χρησιμοποιώντας την κλασική για το Maple μέθοδο. Περισσότερα για την μέθοδο μπορούμε να βρούμε στο on line εγχειρίδιο του Maple..

Πίνακας 22. *Δυνατές τιμές της παραμέτρου method για την αριθμητική επίλυση μιας Δ.Ε.*

Παράδειγμα: 15-9

Να λυθεί αριθμητικά με την μέθοδο Rkf45 το πρόβλημα αρχικών τιμών

$$\begin{cases} y' = 2x \cdot e^{-x \cdot y} \\ y(0) = 1 \end{cases}.$$

Λύση:

Ορίζουμε τη διαφορική εξίσωση:

```
> ode:=diff(y(x),x)=2*x+exp(-x*y(x));
```

$$ode := \frac{d}{dx}\,y(x) = 2\,x + e^{(-x\,y(x))}$$

Ζητάμε, χρησιμοποιώντας την εντολή `dsolve` και την παράμετρο numeric, την προσεγγιστική λύση του προβλήματος αρχικών τιμών με την μέθοδο Rkf45.

```
> sol:=dsolve({ode,y(0)=1},numeric,method=rkf45);
```

$$sol := \mathbf{proc}(x\_rkf45)\ \ldots\ \mathbf{end\ proc}$$

Το Maple ως λύση μιας διαφορικής εξίσωσης με αριθμητικές μεθόδους επιστρέφει μια διαδικασία με την οποία μπορούμε να προσδιορίσουμε τιμές της λύσης της άγνωστης συνάρτησης (μια διαδικασία η οποία προκύπτει από παρεμβολή στα σημεία της αριθμητικής λύσης).

Έτσι, για παράδειγμα μπορούμε να ζητήσουμε τις τιμές:

```
> sol(0);
```

$$[x = 0.,\ y(x) = 1.]$$

(επαληθεύσαμε της αρχικές συνθήκες)

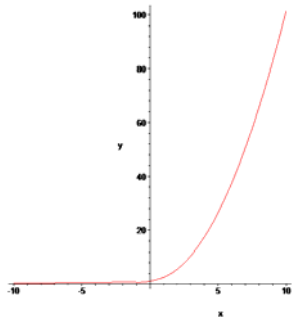
```
> sol(1);
```

$$[x = 1.,\ y(x) = 2.48090978007924168]$$

Αν θέλουμε να σχεδιάσουμε την ολοκληρωτική καμπύλη που προκύπτει, μπορούμε να χρησιμοποιήσουμε την εντολή του πακέτου plots, `odeplot`.

Περιγραφή:	***Σχεδιάζει τη λύση μιας διαφορικής εξίσωσης που έχει λυθεί αριθμητικά.***	
Γενική σύνταξη:	***odeplot(sol, vars, range, options)***	
Παράμετροι:	sol	Η λύση της διαφορικής εξίσωσης.
	range	Το διάστημα της ανεξάρτητης μεταβλητής.
	options	Όλες οι δυνατές παράμετροι της εντολής plot (title,color κ.α.).
Πακέτο:	***plots***	

```
> with(plots):
> odeplot(sol);
```



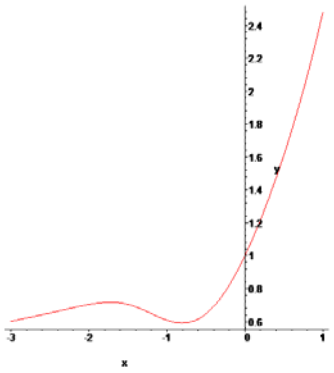
Μπορούμε να χρησιμοποιήσουμε την παράμετρο `range` για να ορίσουμε το διάστημα στο οποίο θα προσδιορίσουμε τη λύση .

```
> sol:=dsolve({ode,y(0)=1},numeric,method=rkf45,range=-
3..1);
```

$$sol := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc}$$

Ζητάμε πάλι την ολοκληρωτική καμπύλη.

```
> odeplot(sol);
```



□

Παράδειγμα: 15-10

Να λυθεί αριθμητικά με την μέθοδο taylor το πρόβλημα αρχικών τιμών

$$\begin{cases} y''(x) + \sin x \cdot y(x) = 0 \\ y(0) = 2 \\ y''(0) = -2 \end{cases} .$$

Λύση:

Ορίζουμε τη διαφορική εξίσωση.

```
> ode:=diff(y(x),x$2) + sin(x)*y(x) = 0;
```

$$ode := \left(\frac{d^2}{dx^2}\,y(x)\right) + \sin(x)\,y(x) = 0$$

Η ονομάζουμε με sol τη λύση της.

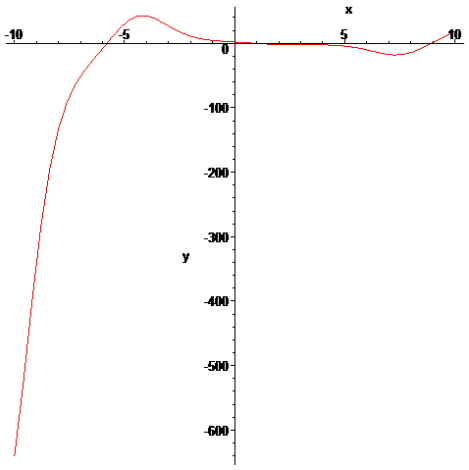
```
> sol:=dsolve({ode, y(0)=2, D(y)(0)=-2},y(x),
numeric,method=taylorseries);
```

$$sol := \mathbf{proc}(x\_taylorseries)\ \ldots\ \mathbf{end\ proc}$$

Η ολοκληρωτική καμπύλη με χρήση της εντολής `odeplot` είναι:

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

```
> odeplot(sol);
```



Αν εξετάσουμε τις τιμές για τις αρχικές συνθήκες, βλέπουμε ότι πράγματι ισχύουν.

```
> sol(0);
```

$$\left[x = 0.,\ y(x) = 2.,\ \frac{d}{dx}\,y(x) = -2.\right]$$

Αν θέλουμε η λύση να δίνεται σε διαφορετική μορφή (η προεπιλογή είναι διαδικασία) μπορούμε να ορίσουμε την παράμετρο output. (δυνατές τιμές: `procedurelist, listprocedure, operator, piecewise, array`).

Έτσι, για το παραπάνω πρόβλημα ζητάμε λύση σε μορφή πολύκλαδης συνάρτησης χωρίς να προσδιορίζουμε την προσεγγιστική μέθοδο.

```
> sol:=dsolve({ode, y(0)=2, D(y)(0)=-2},y(x),numeric,
output=piecewise,range=0..0.1);
```

$$
\begin{aligned}
sol := \Big[\; & x = x,\ y(x) = \{\ undefined\ ,\ x < 0. \\
& 2.0000121592890 - 2.0006898196426\,x \\
& - 0.025794375751724\,(x - 0.026500208468168)^2 \\
& - 0.31552710075910\,(x - 0.026500208468168)^3 \\
& + 0.16853560055255\,(x - 0.026500208468168)^4\ ,\ x \le 0.053004169363349972 \\
& 2.0002811686365 - 2.0055509709194\,x \\
& - 0.070573272873034\,(x - 0.076500208468168)^2 \\
& - 0.28134591142900\,(x - 0.076500208468168)^3 \\
& + 0.17271348372933\,(x - 0.076500208468168)^4\ ,\ x \le 0.10000000000000004 \\
& undefined\ ,\ otherwise,\ \frac{d}{dx}\,y(x) = \{\ undefined\ ,\ x < 0. \\
& -1.9993226750195 - 0.051589691904584\,x \\
& - 0.94665051886709\,(x - 0.026500208468168)^2 \\
& + 0.67548178067061\,(x - 0.026500208468168)^3 \\
& + 0.084281385839285\,(x - 0.026500208468168)^4\ ,\ x \le 0.053004169363349972 \\
& -1.9947531835187 - 0.14114710626717\,x \\
& - 0.84409061288620\,(x - 0.076500208468168)^2 \\
& + 0.69186782542891\,(x - 0.076500208468168)^3 \\
& + 0.081463953344954\,(x - 0.076500208468168)^4\ ,\ x \le 0.10000000000000004 \\
& undefined\ ,\ otherwise\ \Big]
\end{aligned}
$$

□

15.4 Διαφορικές Εξισώσεις με Μερικές Παραγώγους

Για την επίλυση διαφορικών εξισώσεων με μερικές παραγώγους το Maple παρέχει την εντολή `pdsolve` η οποία επιτυγχάνει να λύση ορισμένες μερικές διαφορικές εξισώσεις αναλυτικά. Επίσης ορίζοντας την παράμετρο numeric μπορούμε να έχουμε την αριθμητική επίλυσης της μ.δ.ε..

Περιγραφή:	***Επιλύει μια διαφορική εξίσωση ή ένα σύστημα διαφορικών εξισώσεων με μερικές παραγώγους.***	
Γενική σύνταξη:	***pdsolve(pde ,conds, f, build)***	
Παράμετροι:	pde	Η διαφορική εξίσωση.
	conds	Αρχικές, συνοριακές συνθήκες.
	f	Η άγνωστη συνάρτηση.
	build	Προσπαθεί να κατασκευάσει την λύσης σε μη πεπλεγμένη μορφή.
	numeric	Ορίζει το να αναζητηθούν αριθμητικές λύσεις.

Παράδειγμα: 15-11

Δίνεται η εξίσωση κύματος σε μια διάσταση $u_{xx} = \frac{1}{c^2} u_{tt}$.

Λύση:

Ορίζουμε την εξίσωση του κύματος ως wave.

```
> wave:=diff(u(x,t),x$2)=1/c^2*diff(u(x,t),t$2);
```

$$wave := \frac{\partial^2}{\partial x^2} u(x, t) = \frac{\frac{\partial^2}{\partial t^2} u(x, t)}{c^2}$$

Η λύση της μονοδιάστατης κυματικής εξίσωσης είναι:

```
> sol:=pdsolve(wave,u(x,t));
```

$$sol := u(x, t) = \_F1(c\,t + x) + \_F2(c\,t - x)$$

Όπου _F1, _F2 είναι αυθαίρετες συναρτήσεις.

Αν αυξήσουμε το επίπεδο πληροφοριών του συστήματα για την εντολή `pdsolve` θα έχουμε πληροφορίες για τον τρόπο επίλυσης της διαφορικής εξίσωσης.

```
> infolevel[pdsolve]:=5;
```

$$infolevel_{pdsolve} := 5$$

```
> sol:=pdsolve(wave,u(x,t));
Checking arguments ...
First set of solution methods (general or quase general
solution)
Trying differential factorization for linear PDEs ...
differential factorization successful.
First set of solution methods successful
```

$$sol := \mathrm{u}(x,t) = \_\mathrm{F1}(c\,t + x) + \_\mathrm{F2}(c\,t - x)$$

Με την εντολή `pdetest` μπορούμε να κάνουμε επαλήθευση του αποτελέσματος.

```
> pdetest(sol,wave);
```

$$0$$

Ας εφαρμόσουμε τώρα την μέθοδο του χωρισμού των μεταβλητών για να προσδιορίσουμε την λύση της

Αντικαθιστούμε στην διαφορική εξίσωση $u(x,t) = X(x)T(t)$

```
> wave:=subs(u(x,t)=X(x)*T(t),wave);
```

$$wave := \frac{\partial^2}{\partial x^2}(\mathrm{X}(x)\,\mathrm{T}(t)) = \frac{\frac{\partial^2}{\partial t^2}(\mathrm{X}(x)\,\mathrm{T}(t))}{c^2}$$

Διαιρούμε με την διαφορική εξίσωση με $X(x)T(t)$

```
> sep := wave/X(x)/T(t);
```

$$sep := \frac{\frac{d^2}{dx^2}\mathrm{X}(x)}{\mathrm{X}(x)} = \frac{\frac{d^2}{dt^2}\mathrm{T}(t)}{\mathrm{T}(t)\,c^2}$$

Θέτουμε το δεξή μέλος της παραπάνω εξίσωσης με $-k^2$. Επιλύουμε την συνήθη διαφορική εξίσωση που προκύπτει χρησιμοποιώντας την εντολή `dsolve`.

```
> rhs(sep)= -k^2;
```

$$\frac{\frac{d^2}{dt^2}\mathrm{T}(t)}{\mathrm{T}(t)\,c^2} = -k^2$$

```
> tsolution := dsolve(%,T(t));
```

$$tsolution := \mathrm{T}(t) = \_C1\,\sin(c\,k\,t) + \_C2\,\cos(c\,k\,t)$$

Αντικαθιστούμε τις σταθερές με C1=1/2, C2=1 k=1 και c=4.

```
> tsolution:=subs(_C1=1/2,_C2=1,k=1,c=4,tsolution);
```

$$tsolution := \mathrm{T}(t) = \frac{1}{2}\sin(4\,t) + \cos(4\,t)$$

Θέτουμε το αριστερό μέλος της παραπάνω εξίσωσης με επιλύουμε την συνήθη διαφορική εξίσωση που προκύπτει χρησιμοποιώντας την εντολή `dsolve`

```
.> lhs(sep)= -k^2;
```

$$\frac{\frac{d^2}{dx^2}\mathrm{X}(x)}{\mathrm{X}(x)} = -k^2$$

```
> xsolution := dsolve(%,X(x));
```

$$xsolution := \mathrm{X}(x) = \_C1\,\sin(k\,x) + \_C2\,\cos(k\,x)$$

Αντικαθιστούμε τις σταθερές με C1=1/2, C2=1 k=1 και c=4.

```
> xsolution:=subs(_C1=1/2,_C2=1,k=1,c=4,xsolution);
```

$$xsolution := \mathrm{X}(x) = \frac{1}{2}\sin(x) + \cos(x)$$

Η λύση $u(x,t)$ ια είναι:

```
> usolution:=rhs(tsolution)*rhs(xsolution);
```

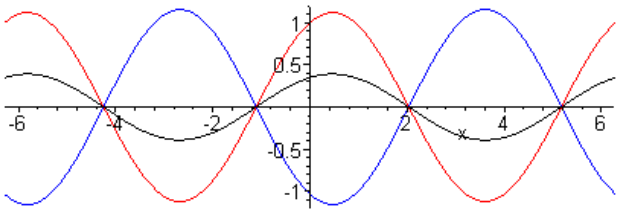
$$usolution := \left(\frac{1}{2}\sin(4\,t) + \cos(4\,t)\right)\left(\frac{1}{2}\sin(x) + \cos(x)\right)$$

Δημιουργούμε τις γραφικές παραστάσεις της λύσης για t=0,t=1 και t=2.

```
> p1:=plot(subs(t=0,usolution),x=-2*Pi..2*Pi):
> p2:=plot(subs(t=1,usolution),x=-2*Pi..2*Pi,color=blue):
> p3:=plot(subs(t=2,usolution),x=-2*Pi..2*Pi,color=black):
```

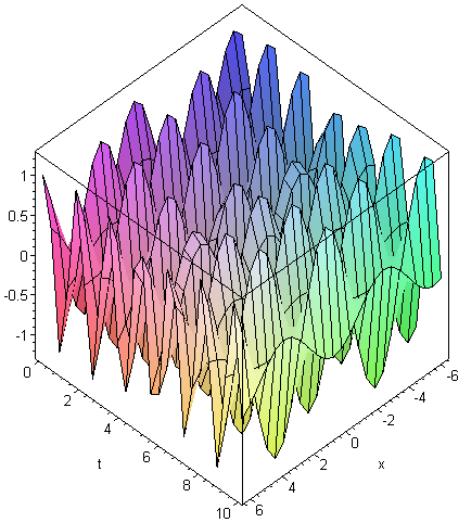
Ενεργοποιούμε το πακέτο εντολών `plots` και με την εντολή `display` εμφανίζουμε τις τρεις γραφικές παραστάσεις μαζί.

```
> with(plots):display(p1,p2,p3);
```



Αν χρησιμοποιήσουμε την εντολή `plot3d` έχουμε την γραφική αναπαράσταση της λύσης.

```
> plot3d(usolution,x=-2*Pi..2*Pi,t=0..10,axes=boxed);
```



Παράδειγμα: 15-12

Στο παράδειγμα αυτό θα μελετήσουμε την εξίσωση θερμότητας σε μια διάσταση. Θεωρούμε μια ράβδο μήκους 1m και υποθέτουμε ότι στις άκρες της ράβδου υπάρχει σταθερή θερμοκρασία ίση με 0 ενώ η αρχική κατανομή της θερμοκρασίας είναι $f(x)=\left(1-x\right)\left(1-e^{-50x}\right)$

Λύση:

Η εξίσωση διάδοσης της θερμότητας σε μια διάσταση είναι: $u_t = au_{xx}$ (με $a = \frac{1}{10}$), έτσι ορίζουμε την εξίσωση heat_eqn.

```
> heat_eqn:={diff(u(x,t),t)=1/10*diff(u(x,t),x,x)};
```

$$heat\_eqn := \{\frac{\partial}{\partial t}\mathrm{u}(x,t) = \frac{1}{10}\left(\frac{\partial^2}{\partial x^2}\mathrm{u}(x,t)\right)\}$$

Η λύση που προκύπτει με χρήση της εντολής `pdsolve` είναι:

```
> pdsolve(heat_eqn);
```

$$(\mathrm{u}(x,t) = \_\mathrm{F1}(x)\ \_\mathrm{F2}(t))\ \&\mathrm{where}$$

$$\left[\{\frac{d}{dt}\_\mathrm{F2}(t) = \frac{1}{10}\_c_1\ \_\mathrm{F2}(t), \frac{d^2}{dx^2}\_\mathrm{F1}(x) = \_c_1\ \_\mathrm{F1}(x)\}\right]$$

Οι συνοριακές, αρχικές συνθήκες που θα χρησιμοποιήσουμε είναι: $\begin{cases} u(0,t) = 0 \\ u(1,t) = 0 \\ u(x,0) = f(x) \end{cases}$

Ορίζουμε τις συνοριακές, αρχικές συνθήκες.

```
> conditions:={u(x,0)=(1-x)*(1-exp(-50*x)),u(0,t)=0,
u(1,t)=0};
```

$$conditions := \{\mathrm{u}(x,0) = (1-x)\,(1-\mathbf{e}^{(-50\,x)}), \mathrm{u}(0,t) = 0, \mathrm{u}(1,t) = 0\}$$

Τώρα χρησιμοποιώντας την εντολή `pdsolve` θα επιλύσουμε το πρόβλημα αριθμητικά.

```
> solution := pdsolve(heat_eqn,conditions, numeric,
timestep=1/100, spacestep=1/100);
```

$$solution := \mathbf{module}()\ \mathbf{export}\ plot, plot3d, animate, value, settings;\ \ldots\ \mathbf{end\ module}$$

Η εντολή `pdsolve` όταν χρησιμοποιηθεί για αριθμητική επιλύσει μιας διαφορικής εξίσωσης με μερικές παραγώγους δίνει την δυνατότητα να υπολογίσουμε ή να αναπαραστήσουμε την λύσεις της διαφορικής εξισώσεις με διάφορες μεθόδους. Οι μέθοδοι που μπορούμε να χρησιμοποιήσουμε είναι η `plot`, η `plot3d`, η `animate`, και η `value`.

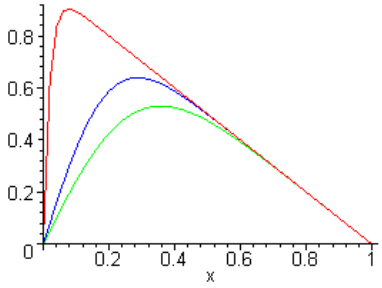
Έτσι αν θέλουμε να παραστήσουμε γραφικά την λύση για διάφορες χρονικές στιγμές χρησιμοποιώντας στην solution την εντολή `plot`.

```
> p1:=solution:-plot(t=0,numpoints=50):
```

```
> p2:=solution:-plot(t=1/8,numpoints=50,color=blue):
> p3:=solution:-plot(t=1/4,numpoints=50,color=green):
```

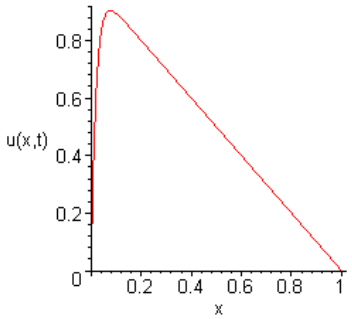
Ενεργοποιούμε το πακέτο `plots` και εμφανίζουμε τις τρεις γραφικές παραστάσεις μαζί.

```
> with(plots):
> display(p1,p2,p3);
```



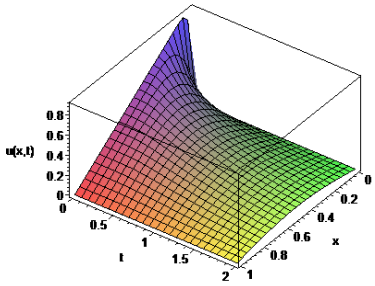
Μπορούμε να δούμε την κατανομή της θερμότητας με κίνηση ως προς t, χρησιμοποιώντας στην solution την εντολή `animate`.

```
> solution :-animate(t=0..1.5,frames=80,labels=["x",
"u(x,t)"]);
```



Ακόμα μπορούμε να παραστήσουμε την λύση γραφικά χρησιμοποιώντας στην solution την εντολή `plot3d`.

```
> solution:-plot3d(t=0..2,axes=boxed,labels=["x","t",
"u(x,t)"]);
```



Παράδειγμα: 15-13

Να λυθεί η διαφορική εξίσωση $u_x + \cos(2x)u_y = -\sin y$.

Λύση:

Ορίζουμε την εξίσωση ως pde.

```
> pde:=diff(u(x,y),x)+cos(2*x)*diff(u(x,y),y)=-sin(y);
```

$$pde := \left(\frac{\partial}{\partial x}\,u(x,y)\right) + \cos(2x)\left(\frac{\partial}{\partial y}\,u(x,y)\right) = -\sin(y)$$

Η λύση της θα είναι:

```
> sol:=pdsolve(pde,u(x,y));
```

$$sol := u(x,y) = -\int^{x} \sin\left(\frac{1}{2}\sin(2\_a) + y - \frac{1}{2}\sin(2x)\right) d\_a + \_F1\left(y - \frac{1}{2}\sin(2x)\right)$$

Αν θέλουμε να σχεδιάσουμε τη λύση της για δεδομένες αρχικές συνθήκες, μπορούμε να χρησιμοποιήσουμε την εντολή `PDEplot`

Περιγραφή:	***Σχεδιάζει τη λύση μιας πρώτης τάξης ημιγραμμικής μερικής διαφορικής εξίσωσης.***	
Γενική σύνταξη:	***PDEplot(pde, conds, sranges)***	
Παράμετροι:	pde	Η διαφορική εξίσωση.
	conds	Αρχικές συνθήκες.
	srange	Το εύρος των παραμέτρων για τις αρχικές συνθήκες.
Πακέτο:	***PDEtools***	

Ορίζουμε τις αρχικές συνθήκες

```
> incont:=[0,s,1+s^2];
```

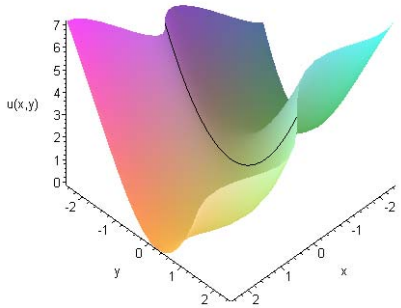
$$incont := [0, s, 1 + s^2]$$

Ενεργοποιούμε το πακέτο `PDEtools`.

```
> with(PDEtools):
```

Έτσι, η επιφάνεια της λύσης μαζί με τις αρχικές συνθήκες είναι:

```
> PDEplot(pde,u(x,y),incont,s=-2..2 );
```



Κεφάλαιο 16

Προσεγγιστικές Μέθοδοι

Στο κεφάλαιο αυτό θα δούμε εντολές που μας βοηθούν να αντιμετωπίσουμε θέματα προσεγγιστικά. Θα παρουσιάσουμε κατάλληλες εντολές για την εύρεση του βέλτιστου μοντέλου με χρήση της μεθόδου των ελαχίστων τετραγώνων. Στη συνέχεια θα παρουσιάσουμε διάφορες μεθόδους παρεμβολής και αριθμητικής ολοκλήρωσης. Εδώ πρέπει να θυμίσουμε ότι το θέμα της αριθμητικής επίλυσης αλγεβρικών εξισώσεων το εξετάσαμε αρκετά αναλυτικά στην παράγραφο 2 του κεφαλαίου 7 και την επίλυση διαφορικών εξισώσεων με αριθμητικές μεθόδους την είδαμε στο κεφαλαίου15.

16.1 Μέθοδος Ελαχίστων Τετραγώνων

Το Maple διαθέτει την εντολή `LeastSquares` με την οποία μπορούμε να βρούμε το βέλτιστο μοντέλο γραμμικό (και όχι μόνο) με τη μέθοδο των ελαχίστων τετραγώνων.

Η γενική σύνταξη της εντολής είναι:

Περιγραφή:	**Βέλτιστο Μοντέλο με τη Μέθοδο των Ελαχίστων Τετραγώνων**		
Γενική σύνταξη:	***LeastSquares(xydata, v, opts);*** ***LeastSquares(xdata, ydata, v, opts);***		
Παράμετροι:	xydata	Το σύνολο των δεδομένων στη μορφή: [[x1,y1],[x2,y2],...,[xn,yn]]	
	xdata	Το σύνολο των δεδομένων του x στη μορφή: [x1,x2,...,xn]	
	ydata	Το σύνολο των δεδομένων του x στη μορφή: [y1,y2,...,yn]	
	v	Το όνομα της ανεξάρτητης μεταβλητής.	
	opts	curve=	Το μοντέλο που θα χρησιμοποιήσει η μέθοδος (προεπιλογή: $ax+b$)
		weight=	Τα Βάρη.
Πακέτο:	**CurveFitting**		

Παράδειγμα: 16-1

Δίνονται οι αριθμητικές μετρήσεις:

x	0.2	0.5	0.6	0.9	1	1.1
y	0.52	0.9	1	1.3	1.5	1.7

Να εφαρμοστεί η μέθοδος των ελαχίστων τετραγώνων για τον προσδιορισμό:

a. Του βέλτιστου γραμμικού μοντέλου.

b. Του βέλτιστου μοντέλου της μορφής $y = ax^2 + bx + c$

Λύση:

Ενεργοποιούμε το πακέτο `CurveFitting`.

```
>with(CurveFitting):
```

Ορίζουμε τα δεδομένα ως δύο διανύσματα και τα καλούμε Xvalues, YValues.

```
>Xvalues:=[0.2,0.5,0.6,0.9,1,1.1];
```

$$Xvalues := [0.2, 0.5, 0.6, 0.9, 1, 1.1]$$

```
>Yvalues:=[0.52,0.9,1,1.3,1.5,1.7];
```

$$Yvalues := [0.52, 0.9, 1, 1.3, 1.5, 1.7]$$

- Καλούμε την εντολή `LeastSquares` χωρίς άλλες παραμέτρους. Είναι άλλωστε προεπιλογή το γραμμικό μοντέλο που αναζητούμε. Το αποτέλεσμα της εντολής το ονομάζουμε lin ώστε να το χρησιμοποιήσουμε στη συνέχεια.

```
> lin:=LeastSquares(Xvalues,Yvalues, x);
```

$$lin := 0.2584135977 + 1.24872521246458868\,x$$

- Καλούμε την εντολή `LeastSquares` με παράμετρο curve=a*x^2+b*x+c. Το πολυώνυμο 2ας τάξης που προκύπτει το ονομάζουμε nonlin.

```
> nonlin:=LeastSquares(Xvalues,Yvalues,x,curve=
a*x^2+b*x+c);
```

$$nonlin := 0.3515413467 + 0.882961243079119939\,x + 0.276299339167709102\,x^2$$

Στη συνέχεια θα παρουσιάσουμε πάνω στο ίδιο σύστημα αξόνων τα δεδομένα

x	0.2	0.5	0.6	0.9	1	1.1
y	0.52	0.9	1	1.3	1.5	1.7

και τις καμπύλες που προκύψαν.

(ενεργοποιούμε το πακέτο plots.)

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

Ορίζουμε ως c το γράφημα των σημείων που ορίζονται από τα δεδομένα.

```
>c:=plot([[0.2,0.52],[0.5,0.9],[0.6,1],[0.9,1.3],[1,1.5],
[1.1,1.7]], style=point,symbol=diamond,symbolsize=20);
```

Ορίζουμε ως d τη γραφική παράσταση του γραμμικού μοντέλου που βρήκαμε.

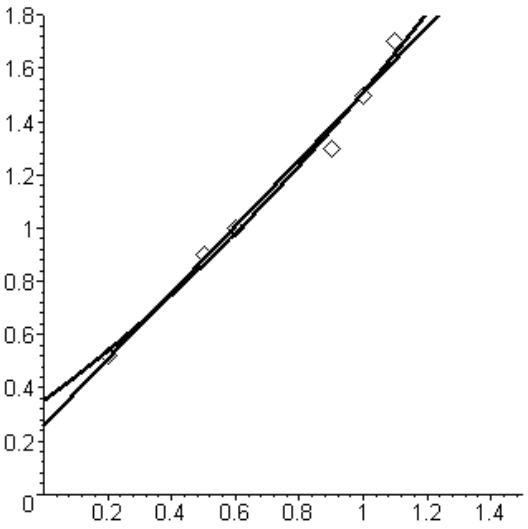
```
> d:=plot(lin,x=0..2):
```

Ορίζουμε ως e τη γραφική παράσταση της καμπύλης $y = ax^2 + bx + c$.

```
> e:=plot(nonlin,x=0..2):
```

Παρουσιάζουμε στο ίδιο σύστημα αξόνων τα c,d,e.

```
>display(c,d,e,view=[0..1.5,0..1.8]);
```



Παράδειγμα: 16-2

Να βρεθεί με τη μέθοδο των ελαχίστων τετραγώνων το βέλτιστο μοντέλο της μορφής $y = a\sin(x) + b\cos(x)$: για τα δεδομένα

x	-1	-0.5	0	0.5	1
y	-1	0	1	2	1

Λύση:

Ενεργοποιούμε το πακέτο `CurveFitting`.

```
> with(CurveFitting):
```

Ορίζουμε τα δεδομένα ως δύο διανύσματα και τα καλούμε Xvalues,YValues.

```
> Xvalues:=[-1,-0.5,0,0.5,1];
```

$$Xvalues := [-1, -0.5, 0, 0.5, 1]$$

```
> Yvalues:=[-1,0,1,2,1];
```

$$Yvalues := [-1, 0, 1, 2, 1]$$

Καλούμε την εντολή `LeastSquares` με παράμετρο curve=a*sin(x)+b*cos(x).

```
>mondelo:=LeastSquares(Xvalues,Yvalues, x, curve=
a*sin(x)+b*cos(x));
```

$$mondelo := 1.408321960 \sin(x) + 0.8818911704 \cos(x)$$

Ενεργοποιούμε το πακέτο plots.

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

Ορίζουμε ως c το γράφημα των σημείων που ορίζονται από τα δεδομένα.

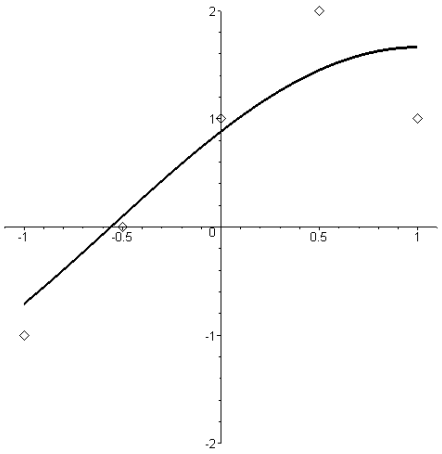
```
>c:=plot([[-1,-1],[-0.5,0],[0,1],[0.5,2],[1,1]],   style=
point,symbol=diamond,colour=black,symbolsize=20):
```

Ορίζουμε ως d τη γραφική παράσταση του μοντέλου που υπολογίσαμε

```
> d:=plot(mondelo,x=-1..1):
```

Με την εντολή display εμφανίζουμε στο ίδιο σύστημα αξόνων τις παραπάνω γραφικές παραστάσεις.

```
>display(c,d,view=[-1.1..1.1,-2..2]);
```



□

16.2 Παρεμβολή

- **Πολυωνυμική Παρεμβολή**

Με το Maple μπορούμε να κάνουμε πολυωνυμική παρεμβολή με μια από τις μεθόδους **Lagrange**, **monomial**, **Newton**, ή **power.**

	Πολυωνυμική Παρεμβολή	
Γενική σύνταξη:	***PolynomialInterpolation(xydata, v, opts);*** ***PolynomialInterpolation(xdata, ydata, v, opts);***	
Παράμετροι:	xydata	Το σύνολο των δεδομένων στη μορφή: [[x1,y1],[x2,y2],...,[xn,yn]].
	xdata	Το σύνολο των δεδομένων του x στη μορφή: [x1,x2,...,xn].
	ydata	Το σύνολο των δεδομένων του y στη μορφή: [y1,y2,...,yn].
	v	Το όνομα της ανεξάρτητης μεταβλητής.
	form=	Lagrange, monomial, Newton, power
Πακέτο:	**CurveFitting**	

Η παρεμβολή αυτή επιστρέφει ένα πολυώνυμο βαθμού n-1, εάν το πλήθος το δεδομένων είναι n.

Παράδειγμα: 16-3

Να βρεθεί το πολυώνυμο που παρεμβάλλει τα δεδομένα

x	0.2	0.5	0.6	0.9	1	1.1
y	0.52	0.9	1	1.3	1.5	1.7

με την μεθόδους Lagrange, και να βρεθεί η τιμή του πολυωνύμου στο x=0.625.

Λύση:

Ενεργοποιούμε το πακέτο `CurveFitting`.

```
>with(CurveFitting):
```

Ορίζουμε ως p1 το πολυώνυμο παρεμβολής του Lagrange.

```
>p1:=x->PolynomialInterpolation ([0.2,0.5,0.6,0.9,1,1.1],
[0.52,0.9,1,1.3,1.5,1.7],x,form=Lagrange);
```

$$p1 := x \to CurveFitting:-PolynomialInterpolation([0.2, 0.5, 0.6, 0.9, 1, 1.1], [0.52, 0.9, 1, 1.3, 1.5, 1.7], x, form = Lagrange)$$

(υπολογίζουμε την τιμή στο 0.625)

```
>p1(0.625);
```

$$1.019027322$$

Σχεδιάζουμε στο ίδιο σύστημα αξόνων τα σημεία (x_i, y_i), $i = 0,...,5$. και το πολυώνυμο παρεμβολής του Lagrange.

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

Ορίζουμε ως c την γραφική αναπαράσταση των σημείων παρεμβολής.

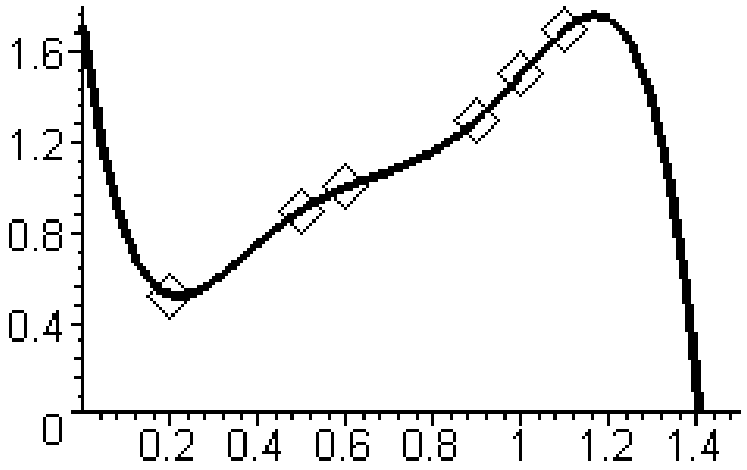
```
>c:=plot([[0.2,0.52],[0.5,0.9],[0.6,1],[0.9,1.3],[1,1.5],
[1.1,1.7]],style=point,symbol=diamond,symbolsize=30);
```

Ορίζουμε ως d την γραφική παράσταση του πολυωνύμου παρεμβολής.

```
>d:=plot(p1(x),x=0..2);
```

Εμφανίζουμε και τις δύο γραφικές παραστάσεις στο ίδιο σύστημα αξόνων.

```
>display(c,d,view=[0..1.5,0..1.8]);
```



Αν θέλουμε να εμφανιστεί ο τύπος του πολυωνύμου , τότε:

```
> PolynomialInterpolation([0.2,0.5,0.6,0.9,1,1.1], [0.52,
0.9,1,1.3,1.5,1.7],x,form=Lagrange);
```

$$\begin{aligned}&-8.597883598\,(x-0.5)\,(x-0.6)\,(x-0.9)\,(x-1)\,(x-1.1)\\&\quad+250.0000000\,(x-0.2)\,(x-0.6)\,(x-0.9)\,(x-1)\,(x-1.1)\\&\quad-416.6666667\,(x-0.2)\,(x-0.5)\,(x-0.9)\,(x-1)\,(x-1.1)\\&\quad+773.8095238\,(x-0.2)\,(x-0.5)\,(x-0.6)\,(x-1)\,(x-1.1)\\&\quad-937.5000000\,(x-0.2)\,(x-0.5)\,(x-0.6)\,(x-0.9)\,(x-1.1)\\&\quad+314.8148148\,(x-0.2)\,(x-0.5)\,(x-0.6)\,(x-0.9)\,(x-1)\end{aligned}$$

Ζητάμε το παραπάνω αποτέλεσμα σε πιο απλή μορφή.

```
> simplify(%);
```

$$-14.49246037x+60.18882292x^2-104.0839950x^3-24.14021170x^5+82.30820126x^4+1.719642862$$

Υπολογίζουμε το πολυώνυμο παρεμβολής με τη μέθοδο Newton.

```
> PolynomialInterpolation([0.2,0.5,0.6,0.9,1,1.1], [0.52,
0.9,1,1.3,1.5,1.7], x,form=Newton);
```

$$\begin{aligned}&((((-24.14021148\,x+29.19973523)\,(x-0.9)+0.9523809758)\,(x-0.6)\\&\quad-0.6666666750)\,(x-0.5)+1.266666667)\,(x-0.2)+0.52\end{aligned}$$

Ζητάμε το παραπάνω αποτέλεσμα σε πιο απλή μορφή.

```
> simplify(%);
```

$$-24.14021148x^5+82.30820049x^4-104.0839939x^3+60.18882226x^2-14.49246017x+1.719642843$$

□

▪ Γραμμικές και Κυβικές Spline

Μια πολύ αποδοτική μέθοδος παρεμβολής είναι εκείνη που προκύπτει από παρεμβολή τμηματικά πολυωνυμικών συναρτήσεων.

Γενική σύνταξη:	***Spline(xydata, v, degree);***	
Παράμετροι:	xydata	Το σύνολο των δεδομένων στη μορφή: [[x1,y1],[x2,y2],...,[xn,yn]]
	xdata	Το σύνολο των δεδομένων του x στη μορφή: [x1,x2,...,xn]
	ydata	Το σύνολο των δεδομένων του y στη μορφή: [y1,y2,...,yn]
	v	Το όνομα της ανεξάρτητης μεταβλητής.
	degree=d	Ο βαθμός των πολυωνύμων.
Πακέτο:	CurveFitting	

Για να γίνει χρήση της εντολής αυτής πρέπει να έχει ενεργοποιηθεί πρώτα το πακέτο `CurveFitting`. Αυτό γίνεται με την εντολή: `with(CurveFitting):`

Παράδειγμα: 16-4

Να βρεθεί η συνάρτηση που παρεμβάλει τα δεδομένα $\begin{array}{c|ccccc} x & 0 & 1 & 2 & 3 & 4 \\ \hline y & 0 & 1 & 4 & 5 & -2 \end{array}$ με τη μέθοδο Splines.

Λύση:

Ενεργοποιούμε το πακέτο εντολών `CurveFitting`.

```
> with(CurveFitting):
```

Η γραμμική παρεμβολή `Splines` είναι (degree=1):

```
> Spline([0,1,2,3,4],[0,1,4,5,-2], x, degree=1);
```

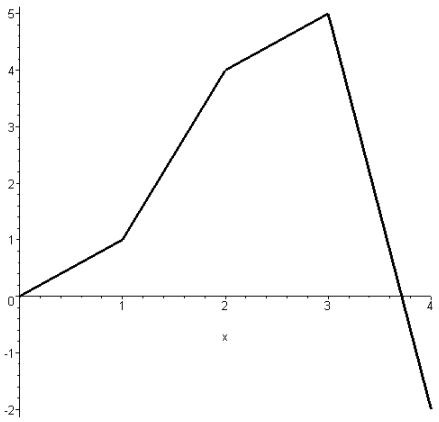
$$\begin{cases} x & x < 1 \\ -2 + 3\,x & x < 2 \\ 2 + x & x < 3 \\ 26 - 7\,x & otherwise \end{cases}$$

Αν ορίσουμε το παραπάνω αποτέλεσμα ως συνάρτηση, μπορούμε να δούμε την γραφική της παράσταση.

```
> f:=x->Spline([0,1,2,3,4],[0,1,4,5,-2], x,degree=1);
```

$$f := x \to CurveFitting:\text{-}Spline([0, 1, 2, 3, 4], [0, 1, 4, 5, -2], x, degree = 1)$$

```
> plot(f(x),x=0..4);
```



Η κυβική παρεμβολή `Splines` είναι (degree=3).

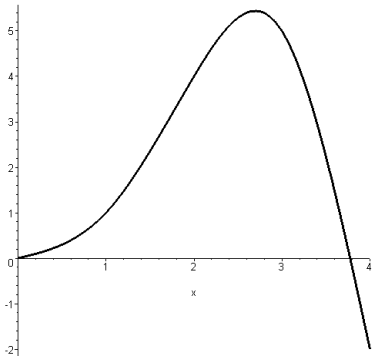
```
> with(CurveFitting):
> Spline([0,1,2,3,4],[0,1,4,5,-2], x,degree=3);
```

$$\begin{cases} \frac{13}{28}x + \frac{15}{28}x^3 & x < 1 \\ \frac{17}{14} - \frac{89}{28}x + \frac{51}{14}x^2 - \frac{19}{28}x^3 & x < 2 \\ \frac{145}{14} - \frac{473}{28}x + \frac{21}{2}x^2 - \frac{51}{28}x^3 & x < 3 \\ -\frac{643}{7} + \frac{2389}{28}x - \frac{165}{7}x^2 + \frac{55}{28}x^3 & otherwise \end{cases}$$

```
> g:=x->Spline([0,1,2,3,4],[0,1,4,5,-2], x,degree=3);
```

$$g := x \to CurveFitting:\text{-}Spline([0, 1, 2, 3, 4], [0, 1, 4, 5, -2], x, degree = 3)$$

```
> plot(g(x),x=0..4);
```



Ας δούμε τώρα τις γραφικές παραστάσεις των δύο αυτών προσεγγίσεων και το σύνολο των δεδομένων στο ίδιο σύστημα αξόνων.

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

Ορίζουμε ως c το γράφημα των σημείων που ορίζονται από τα δεδομένα

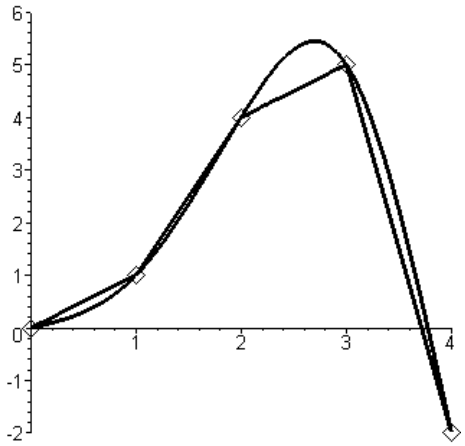
```
> c:=plot([[0,0],[1,1],[2,4],[3,5],[4,-2]],style=point,
symbol=diamond,colour=black,symbolsize=25);
```

Ορίζουμε ως d τις γραφικές παραστάσεις των f,g.

```
>d:=plot([f(x),g(x)],x=0..4);
```

Παρουσιάζουμε τις γραφικές παραστάσεις c και d στο ίδιο σύστημα αξόνων.

```
>display(c,d,view=[0..4,-2..6]);
```



16.3 Αριθμητική Ολοκλήρωση

Στην παράγραφο 5 του κεφαλαίου 9 εξετάσαμε την εντολή `int` η οποία υπολογίζει το ολοκλήρωμα συνάρτησης μιας μεταβλητής. Στις περιπτώσεις όπου δεν είναι δυνατόν να υπολογιστεί ένα ολοκλήρωμα είδαμε ότι μπορούμε να χρησιμοποιήσουμε την εντολή `int` σε συνδυασμό με την εντολή `evalf` για να έχουμε την προσέγγιση ενός ορισμένου ολοκληρώματος,

Στην παράγραφο αυτή θα δούμε πώς προσεγγίζουμε με το Maple ένα ολοκλήρωμα χρησιμοποιώντας τα αθροίσματα Riemman. Επίσης, θα αναφέρουμε για εκπαιδευτικούς λόγους δυο κλασικές μεθόδους αριθμητικής ολοκλήρωσης όπως αυτή του τραπεζίου και του Simpson, που υποστηρίζονται από εντολές του πακέτου `student`.

Παράδειγμα: 16-4

Στο παράδειγμα αυτό θα υπολογίσουμε αλλά και θα παρουσιάσουμε γραφικά διάφορες προσεγγίσεις του ορισμένου ολοκληρώματος $\int_0^{\pi} \sin x dx$.

Λύση:

```
> restart;
```

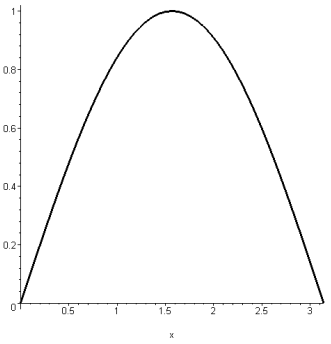
Ορίζουμε τη συνάρτηση $f(x) = \sin x$.

```
> f:=x->sin(x);
```

$$f := x \to \sin(x)$$

Η γραφική της παράσταση είναι:

```
> plot(f(x), x=0..Pi);
```



Για να υπολογίσουμε τις διάφορες προσεγγίσεις του ολοκληρώματος θα χρησιμοποιήσουμε τις ακόλουθες εντολές του πακέτου `student`.

rightbox(συνάρτηση, x=a..b,n);	Παριστάνει γραφικά n το πλήθος ορθογώνια που προσεγγίζουν από δεξιά το εμβαδό που περικλείεται από τη συνάρτηση από a έως b.
rightsum(συνάρτηση, x=a..b,n);	Υπολογίζει το άθροισμα των εμβαδών των ορθογωνίων.
leftbox(συνάρτηση, x=a..b,n);	Παριστάνει γραφικά n το πλήθος ορθογώνια που προσεγγίζουν από αριστερά το εμβαδό που περικλείεται από τη συνάρτηση από a έως b.
leftsum(συνάρτηση, x=a..b,n);	Υπολογίζει το άθροισμα των εμβαδών των ορθογωνίων.

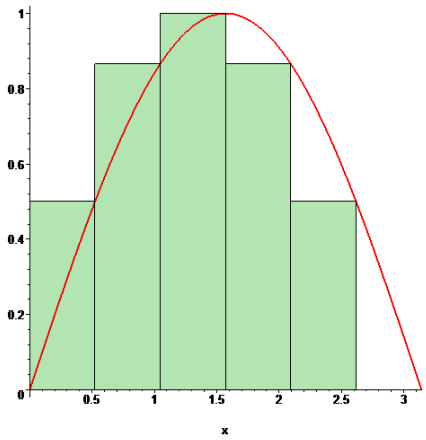
Θα προσεγγίσουμε την περιοχή με παραλληλόγραμμα.

Ενεργοποιούμε το πακέτο.

```
> with(student):
```

Αρχικά θα προσεγγίσουμε το εμβαδό με 6 ορθογώνια (δεξιά), οπότε θα έχουμε:

```
> rightbox(f(x), x=0..Pi, 6);
```



Το άθροισμα των εμβαδών των ορθογωνίων είναι:

```
> rightsum(f(x),x=0..Pi,6);
```

$$\frac{1}{6}\pi\left(\sum_{i=1}^{6}\sin\left(\frac{i\,\pi}{6}\right)\right)$$

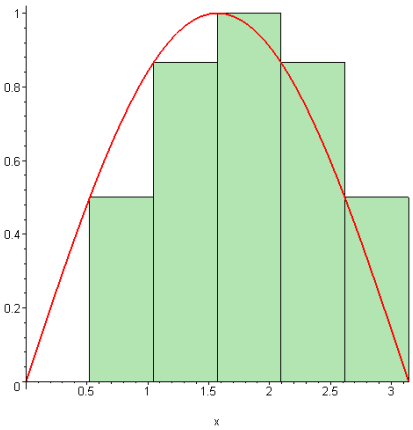
που προσεγγιστικά δίνει την τιμή:

```
> evalf(%);
```

$$1.954097234$$

Αρχικά θα προσεγγίσουμε το εμβαδό με 6 ορθογώνια (αριστερά), οπότε θα έχουμε:

```
> leftbox(f(x), x=0..Pi, 6);
```



Το άθροισμα των εμβαδών των ορθογωνίων είναι:

```
> leftsum(f(x),x=0..Pi,6);
```

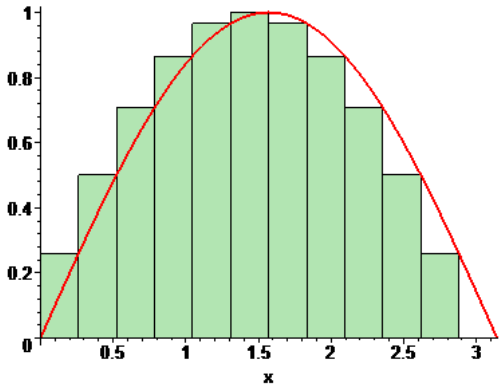
$$\frac{1}{6}\pi\left(\sum_{i=0}^{5}\sin\left(\frac{i\,\pi}{6}\right)\right)$$

```
> evalf(%);
```

$$1.954097234$$

Τώρα θα αυξήσουμε το πλήθος των ορθογωνίων σε 12.

```
> rightbox(f, x=0..Pi, 12);
```



Το άθροισμα των εμβαδών των ορθογωνίων είναι:

```
> rightsum(f(x),x=0..Pi,12);
```

$$\frac{1}{12}\pi\left(\sum_{i=1}^{12}\sin\left(\frac{i\,\pi}{12}\right)\right)$$

```
> evalf(%);
```

$$1.988563777$$

Το συγκεκριμένο ολοκλήρωμα μπορούμε να το υπολογίσουμε ακριβώς με την εντολή int.

```
> int(f(x),x=0..Pi);
```

$$2$$

Οι δύο κλασικές μέθοδοι αριθμητικής ολοκλήρωσης που διδάσκονται είναι αυτή του τραπεζίου και του Simpson, οι οποίες επίσης συμπεριλαμβάνονται στο πακέτο εντολών `student`.

▪ Μέθοδος του Τραπέζιου

Υπενθυμίζουμε ότι αν διαμερίσουμε το διάστημα [a,b] σε n υποδιαστήματα μήκους $h=\frac{b-a}{n}$, τότε το ολοκλήρωμα με την μέθοδο του τραπεζιου δίνεται απο το άθροισμα: $\int_a^b f(x)dx=\left(\frac{f_0}{2}+f_1+\ldots+f_{n-1}+\frac{f_n}{2}\right)h$

Η εντολή με την οποία εκτελείται παραπάνω διαδικασία είναι:

Γενική σύνταξη:	*trapezoid(expr, x=a..b, n);*	
Παράμετροι:	expr:	Η προς ολοκλήρωση συνάρτηση
	x:	Η μεταβλητή ως προς την οποία ολοκληρώνουμε.
	a,b:	Τα άκρα της ολοκλήρωσης.
	n	Ο αριθμός των υποδιαστημάτων της διαμέρισης.

▪ Μέθοδος του Simpson

Αν διαμερίσουμε το διάστημα [a,b] σε n υποδιαστήματα μήκους $h = \frac{b-a}{n}$

τότε το ολοκήρωμα με την μέθοδο του Simpson δίνεται απο το άθροισμα: $\int_a^b f(x)dx = \frac{h}{3}\left(f_0 + f_n + 4(f_1 + f_3 ... + f_{n-1}) + 4\left(f_2 + f_4 + ... + f_{n-2}\right)\right)$

Η εντολή με την οποία εκτελείται η παραπάνω διαδικασία είναι:

Γενική σύνταξη:	*simpson(expr, x=a..b, n);*	
Παράμετροι:	expr:	Η προς ολοκλήρωση συνάρτηση
	x:	Η μεταβλητή ως προς την οποία ολοκληρώνουμε.
	a,b:	Τα άκρα της ολοκλήρωσης.
	n	Ο αριθμός των υποδιαστημάτων της διαμέρισης.

Παράδειγμα: 16-5

Να υπολογιστεί με τη μέθοδο του τραπεζίου και του Simpson το ολοκλήρωμα $\int_0^1 \frac{e^{-x^3}}{x^2+1}dx$.

Λύση:

Ενεργοποιούμε το πακέτο.

```
> with(student):
```

```
> trapezoid(exp(-x^3)/(x^2+1), x = 0..1,10);
```

$$\frac{1}{20}+\frac{1}{10}\left(\sum_{i=1}^{9}\frac{\mathbf{e}^{\left(-\frac{i^3}{1000}\right)}}{\frac{i^2}{100}+1}\right)+\frac{1}{40}\,\mathbf{e}^{(-1)}$$

```
> evalf(%);
```

$$0.6643228980$$

```
> simpson(exp(-x^3)/(x^2+1),x=0..1,10);
```

$$\frac{1}{30}+\frac{1}{60}\,\mathbf{e}^{(-1)}+\frac{2}{15}\left(\sum_{i=1}^{5}\frac{\mathbf{e}^{\left(-\left(\frac{i}{5}-1/10\right)^3\right)}}{\left(\frac{i}{5}-\frac{1}{10}\right)^2+1}\right)+\frac{1}{15}\left(\sum_{i=1}^{4}\frac{\mathbf{e}^{\left(-\frac{i^3}{125}\right)}}{\frac{i^2}{25}+1}\right)$$

```
> evalf(%);
```

$$0.6649406381$$

□

Κεφάλαιο 17

Στοιχεία Προγραμματισμού

Στο κεφάλαιο αυτό θα δούμε κάποια βασικά στοιχεία για το π;νς μπορούμε να χρησιμοποιήσουμε το Maple ως μια γλώσσα προγραμματισμού. Θα παρουσιάσουμε τις βασικές μορφές προγραμματισμού όπως είναι οι εντολές επανάληψης και ελέγχου και ο ορισμός διαδικασιών. Αυτά μαζί με τα όσα έχουμε αναφέρει στα προηγούμενα κεφάλαια για μεταβλητές και δομές θα μας επιτρέψουν να γράψουμε τις πρώτες γραμμές κώδικα.

Επίσης θα παρουσιάσουμε τον τρόπο με τον οποίον μπορούμε να εντάξουμε τους υπολογισμούς μας σε μικρές εφαρμογές που παρουσιάζονται σε διαλογικό γραφικό περιβάλλον. Αυτές οι εφαρμογές στο Maple ονομάζονται Maplets.

17.1 Εντολές Επανάληψης

Για την επαναληπτική εκτέλεση ορισμένων εντολών το Maple είναι εφοδιασμένο με τις εντολές `For` και `While`. Την εντολή For τη χρησιμοποιούμε όταν γνωρίζουμε ακριβώς το πλήθος των επαναλήψεων, ενώ η εντολή `While` χρησιμοποιείται όταν θέλουμε την επανάληψη κάποιων εντολών όσο ισχύει μια συνθήκη.

Η εντολή FOR

Περιγραφή:	***Ορίζει μια επαναληπτική διαδικασία η οποία πραγματοποιείται όσες φορές ορίζει η αρχική τιμή, η τελική τιμή και το βήμα της επανάληψης.***
Γενική σύνταξη:	***for*** ***δείκτης from αρχική τιμή to τελική τιμή by βήμα do*** ***εντολές Maple*** ***od*** ή **end do**

Παράδειγμα: 17-1

Να υπολογιστούν οι 8 πρώτοι αριθμοί .

Λύση:

Όπως είδαμε στο κεφάλαιο 2 η εντολή που δίνει τον i-στο πρώτο αριθμό είναι η `ithprime`. Έτσι δημιουργούμε την επαναληπτική διαδικασία:

```
> for i from 1 to 8 do
    ithprime(i);
 od;
```

2

3

5

7

11

13

17

19 □

Παράδειγμα: 17-2

Να κατασκευαστεί το τρίγωνο του Pascal, δηλαδή το τρίγωνο που δημιουργείται από τους συντελεστές (διωνυμηκή συντελεστές) των πολυωνύμων $(x+y)^n$.

Λύση:

Θα δημιουργήσουμε όλα τα αναπτύγματα της μορφής $(x+y)^i$ με το δείκτη i να παίρνει τιμές από 1 έως 8.

```
> for i from 0 to 8 do
    expand((x+y)^i );
  od;
```

$$1$$

$$x+y$$

$$x^2+2\,x\,y+y^2$$

$$x^3+3\,x^2\,y+3\,x\,y^2+y^3$$

$$x^4+4\,x^3\,y+6\,x^2\,y^2+4\,x\,y^3+y^4$$

$$x^5+5\,x^4\,y+10\,x^3\,y^2+10\,x^2\,y^3+5\,x\,y^4+y^5$$

$$x^6+6\,x^5\,y+15\,x^4\,y^2+20\,x^3\,y^3+15\,x^2\,y^4+6\,x\,y^5+y^6$$

$$x^7+7\,x^6\,y+21\,x^5\,y^2+35\,x^4\,y^3+35\,x^3\,y^4+21\,x^2\,y^5+7\,x\,y^6+y^7$$

$$x^8+8\,x^7\,y+28\,x^6\,y^2+56\,x^5\,y^3+70\,x^4\,y^4+56\,x^3\,y^5+28\,x^2\,y^6+8\,x\,y^7+y^8$$

Αν τώρα χρησιμοποιήσουμε την εντολή `coeffs` που είδαμε στο κεφάλαιο 4, μπορούμε να παρουσιάσουμε το τρίγωνο των δυωνυμικών συντελεστών.

```
> for i from 0 to 8 do
   coeffs( expand( (x+y)^i ) );
  od;
```

$$1$$

$$1,\ 1$$

$$1,\ 1,\ 2$$

3, 3, 1, 1

4, 6, 1, 1, 4

1, 1, 10, 10, 5, 5

1, 1, 6, 15, 20, 15, 6

1, 1, 7, 21, 35, 35, 21, 7

28, 56, 70, 56, 28, 8, 8, 1, 1

Η εντολή WHILE

Περιγραφή:	*Ορίζει μια επαναληπτική διαδικασία η οποία πραγματοποιείται όσο ισχύει η λογική έκφραση.*
Γενική σύνταξη:	*while λογική έκφραση do* *εντολές _Maple* *od*

Παράδειγμα: 17-3

Να δημιουργηθεί μια επαναληπτική διαδικασία η οποία θα εξετάζει ποιοι αριθμοί από το 1 έως το 4 είναι πρώτοι.

Λύση:

Θα χρησιμοποιήσουμε την εντολή `isprime`, για να εξετάσουμε εάν ένας ακέραιος αριθμός είναι πρώτος, και την εντολή `while`, για να επαναλάβουμε τη διαδικασία αυτή για όλους τους αριθμούς από το 1 έως το 4.

```
> i := 1;
```

$$i := 1$$

```
>while i <=4 do
    "Ο",i, "Είναι πρώτος αριθμός;", isprime(i);
    i := i+1;
 od;
```

```
"Ο",1, "Είναι πρώτος αριθμός;", false
```

$$i := 2$$

```
"Ο",2, "Είναι πρώτος αριθμός;", true
```

$i := 3$

```
Ο",3, "Είναι πρώτος αριθμός;", true
```

$i := 4$

```
"Ο",4, "Είναι πρώτος αριθμός;", false
```

17.2 Εντολές Ελέγχου

Όταν σε ένα πρόγραμμα απαιτείται η λήψη αποφάσεων ανάλογα με κάποιες συνθήκες, τότε γίνεται χρήση των εντολών ελέγχου. Η πιο απλή μορφή της εντολής ελέγχου είναι:

Περιγραφή:	***Ορίζει έναν έλεγχο.***

Γενική σύνταξη:	***if λογική παράσταση then***
	εντολές_Maple
	else
	εντολές_Maple
	fi;

Μια πιο σύνθετη μορφή της εντολής if χρησιμοποιείται όταν έχουμε προς έλεγχο δύο λογικές παραστάσεις. Στην περίπτωση αυτή η δομή της εντολής είναι:

if λογική παράσταση then

εντολές_Maple

elif λογική παράσταση then

εντολές_Maple

else

εντολές_Maple

fi;

Παράδειγμα: 17-4

Να ορισθεί η δίκλαδη συνάρτηση $f(x) = \begin{cases} x^2+1 & \text{αν } x < 0 \\ \sin(\pi x) & \text{αν } x \geq 0 \end{cases}$

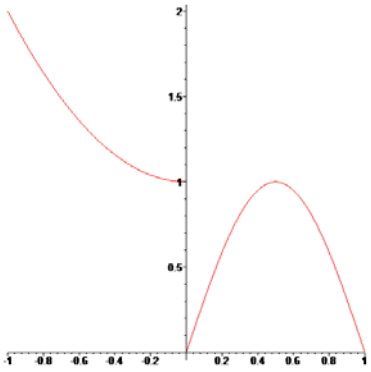
Λύση:

```
> f := proc(x)
>   if x < 0 then x^2+1 else
>     if x >= 0 then sin(Pi*x) fi;
>   fi;
> end;
```

$$f := \mathbf{proc}(x)\ \mathbf{if}\ x < 0\ \mathbf{then}\ x\text{^}2 + 1\ \mathbf{else}\ \mathbf{if}\ 0 \le x\ \mathbf{then}\ \sin(\pi \times x)\ \mathbf{end\ if}\ \mathbf{end\ if}\ \mathbf{end\ proc}$$

Ας δούμε και τη γραφική της παράσταση.

```
> plot( f, -1..1, discont=true, color=red );
```



17.3 Ορίζοντας Διαδικασίες

Οι διαδικασίες, βασική μορφή προγραμματισμού σε όλες τις γλώσσες, επιτρέπουν τον ορισμό νέων εντολών και την αυτοματοποίηση σειράς εντολών για πιο αποδοτικό προγραμματισμό.

Περιγραφή:	***Ορισμός διαδικασίας.***	
Γενική σύνταξη:	***proc (argseq)*** ***local nseq;*** ***global nseq;*** ***options nseq;*** ***description stringseq;*** ***statseq*** ***end proc***	
Παράμετροι:	argseq	Η λίστα των τυπικών παραμέτρων.
	nseq	Ορισμός των τοπικών και γενικών μεταβλητών.
	stringseq	Περιγραφή της διαδικασίας.
	statseq	Οι εντολές της διαδικασίας.

Παράδειγμα: 17-5

Στο παράδειγμα αυτό ορίζουμε μια διαδικασία με δυο παραμέτρους εισόδου η οποία προσθέτει δύο αριθμούς.

```
> plus:=proc(x,y) x+y;  end;
```

$$plus := \mathbf{proc}(x, y)\ x + y\ \mathbf{end\ proc}$$

Καλούμε τη διαδικασία plus.

```
> plus(2,3);
```

$$5$$

Επίσης, η εντολή proc μπορεί να χρησιμοποιηθεί για τον ορισμό του τύπου μιας μαθηματικής συνάρτησης. Έτσι, ορίζουμε τη συνάρτηση $f\left(x\right) = x^2 - 2x + 1$.

```
> f := proc(x)
x^2-2*x+1
 end;
```

$$f := \mathbf{proc}(x)\ x\text{^}2 - 2 \times x + 1\ \mathbf{end\ proc}$$

Καλούμε τη συνάρτηση f για τον υπολογισμό της τιμής της στο x=2.

```
> f(2);
```

$$1$$

□

Παράδειγμα: 17-6

Να κατασκευασθεί μια αναδρομική διαδικασία που να υπολογίσει το παραγοντικό ενός ακεραίου.

Λύση:

Γνωρίζουμε ότι το $n!$ ορίζεται από την αναδρομική σχέση $\begin{matrix} n! = n(n-1)! \\ 0! = 1 \end{matrix}$

Για τον υπολογισμό του θα δημιουργήσουμε μια αναδρομική διαδικασία.

```
> paragontiko:= proc(n)
   if n=0 then
     1
   else
     n*paragontiko(n-1)
 fi;
 end;
```

$$paragontiko := \mathbf{proc}(n)\ \mathbf{if}\ n = 0\ \mathbf{then}\ 1\ \mathbf{else}\ n \times \text{paragontiko}(n-1)\ \mathbf{end\ if}\ \mathbf{end\ proc}$$

Για να υπολογίσουμε το $10!$ καλούμε τη διαδικασία που κατασκευάσαμε:

```
> paragontiko(10);
```

$$3628800$$

Βέβαια το Maple διαθέτει διαδικασία για τον υπολογισμό του παραγοντικού.

```
> 10!;
```

3628800

□

Παράδειγμα: 17-7

Να κατασκευαστεί μια διαδικασία που να υπολογίζει το τρίγωνο του Pascal.

Λύση:

Σε προηγούμενο παράδειγμα είδαμε πώς με τη χρήση μιας επαναληπτικής διαδικασίας μπορούμε να κατασκευάσουμε το τρίγωνο του Pascal. Αυτή την επαναληπτική διαδικασία θα την εντάξουμε σε ένα υποπρόγραμμα.

```
> pascal := proc(m, n)
  local i, j, x, y;
  for i from m to n do
       coeffs(expand( (x+y)^i ));
       print( % );
  od;
end;
```

pascal := **proc** (*m*, *n*)
local *i*, *j*, *x*, *y*;
for *i* **from** *m* **to** *n* **do** coeffs(expand((*x* + *y*)^*i*)); print(%) **end do**
end proc

```
> pascal(0,10);
```

1

1, 1

1, 2, 1

3, 3, 1, 1

4, 6, 4, 1, 1

10, 10, 5, 1, 1, 5

1, 1, 6, 15, 20, 15, 6

7, 21, 35, 35, 21, 7, 1, 1

1, 1, 8, 28, 56, 70, 56, 28, 8

1, 1, 9, 36, 84, 126, 126, 84, 36, 9

1, 1, 10, 45, 120, 210, 252, 210, 120, 45, 10

□

Παράδειγμα: 17-8

Να κατασκευαστεί μια διαδικασία η να υπολογίζει το βέλτιστο γραμμικό μοντέλο με τη μέθοδο των ελαχίστων τετραγώνων.

Λύση:

Ορίζουμε μιας διαδικασία την οποία ονομάζουμε myleastsquare. Η διαδικασία αυτή υπολογίζει τους συντελεστές a,b του βέλτιστου γραμμικού μοντέλου $y=ax+b$ για δεδομένο πλήθος δεδομένων (x_i,y_i). Υπενθυμίζουμε ότι τα α,b δίνονται από τις σχέσεις:

$$a=\frac{n\sum_{i=1}^{n}x_iy_i-\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i}{n\sum_{i=1}^{n}x_i^{\,2}-(\sum_{i=1}^{n}x_i)^2},\ b=\frac{\sum_{i=1}^{n}x_i^{\,2}\sum_{i=1}^{n}y_i-\sum_{i=1}^{n}x_iy_i\sum_{i=1}^{n}x_i}{n\sum_{i=1}^{n}x_i^{\,2}-(\sum_{i=1}^{n}x_i)^2}$$

```
> myleastsquare:=proc(x,y)
> local sx,sy,sxy,sx2,i,n; global a,b,c:
> sx:=0;
> sy:=0;
> sx2:=0;
> sxy:=0;
> n:=nops(x):  #nops(x) Αριθμός στοιχείων του x
> for i from 1 to n  do
>   sx:=sx+x[i];
>   sy:=sy+y[i];
>   sxy:=sxy+x[i]*y[i];
>   sx2:=sx2+x[i]*x[i];
> od;
> a:=((n*sxy-sx*sy)/(n*sx2-sx*sx));
> b:=((sx2*sy-sxy*sx)/(n*sx2-sx*sx));
> print("The linear model is:",evalf(a), "x+",evalf(b));
> end proc;
```

Αν εκτελέσουμε την εντολή maplemint, το Maple ελέγχει αν πιθανόν υπάρχουνε σημασιολογικά λάθη. Πράγματι εδώ βρίσκει ότι έχουμε δηλώσει μια μεταβλητή την c την οποία δε χρησιμοποιούμε πουθενά.

```
> maplemint( myleastsquare );
   These global variables were declared, but never used:c
```

Τώρα θα δούμε ένα παράδειγμα εφαρμογής της παραπάνω διαδικασίας που δημιουργήσαμε. Αρχικά ορίζουμε τον σύνολο δεδομένων.

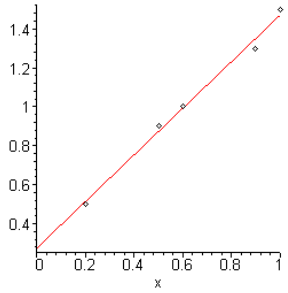
```
> xvalues:=[0.2,0.5,0.6,0.9,1]:
> yvalues:=[0.5,0.9,1,1.3,1.5]:
```

Ας εκτελέσουμε το παραπάνω υποπρόγραμμα για το παραπάνω σύνολο δεδομένων, τότε:

```
> myleastsquare(xvalues,yvalues);
```

"The linear model is:", 1.194174757, "x+", 0.2757281553

```
> with(plots):
> lmodel:=plot(a*x+b,x=0..1):
> points:=pointplot(zip((x,y) -> <x,y>,xvalues, yvalues)):
> display(points,lmodel);
```



17.4 Χρόνος υπολογισμών

Για την μέτρηση του χρόνου που χρειάζεται το Maple για τον υπολογισμό μιας διαδικασίας μπορούμε να κάνουμε χρήση της εντολής time().

```
> settime:=time():
> int(cos(x),x=0..1):
> cpu_time:=(time()-settime)*sec;
```

$$cpu\_time := 4.266 \text{ sec}$$

17.5 Ανάγνωση-Εγγραφή σε Αρχεία

ΕΝΤΟΛΗ ΑΝΑΓΝΩΣΗΣ.

Την εντολή αυτή στο Maple την αντικαθιστούμε θέτοντας τις μεταβλητές στην αρχή του φύλλου εργασίας.
Ωστόσο, αν πρέπει να εισαγάγουμε δεδομένα από ένα αρχείο, μπορούμε να χρησιμοποιήσουμε την εντολή `readdata`.

fopen(«όνομα αρχείου»,read, τύπος) - ανοίγει ένα αρχείο για ανάγνωση ή εγγραφή.
readdata («όνομα αρχείου», αριθμός στοιχείου) - διαβάζει από ένα αρχείο.

```
> data := fopen("C:\\px.txt",READ,BINARY);
> b:=readdata(data,3);
> fclose(data);
```

$$data := 2$$

$$b := [[1.512345, 2.2, 3.4], [4.], [2.7, 3.4, 5.6], [1.8, 3.1, 6.7]]$$

Η ΕΝΤΟΛΗ ΕΚΤΥΠΩΣΗΣ

print("Σχόλιο", μεταβλητές)

```
> s:=100;
```

$$s := 100$$

```
> print("Το άθροισμα είναι:", s);
```

```
Το άθροισμα είναι: 100
```

fopen("όνομα αρχείου", write, τύπος); - ανοίγει ένα αρχείο για εγγραφή.
writeddata ("όνομα αρχείου", αριθμός στοιχείου); - εγγραφή σε ένα αρχείο.

```
>data := fopen("C:\\px.txt",WRITE,BINARY);
>A := array( [[1.512345,2.2,3.4],[2.7,3.4,5.6],
[1.8,3.1,6.7]] );
```

```
>writedata(data,A,float);
>fclose(data);
```

$$data := 2$$

$$A := \begin{bmatrix} 1.512345 & 2.2 & 3.4 \\ 2.7 & 3.4 & 5.6 \\ 1.8 & 3.1 & 6.7 \end{bmatrix}$$

17.6 Εισαγωγή στα Maplets

Τα Maplets μας επιτρέπουν να δημιουργήσουμε γραφικά περιβάλλοντα επικοινωνίας με τον υπολογιστή και να τα συνδυάσουμε με τις εντολές του Maple. Είναι εφαρμογές που βασίζονται στην τεχνολογία Java και μπορούν να περιέχουν buttons, toolbars, menu bars, radio buttons, pop-up menus, drop down buttons, plotter windows κ.ά.

Η βασική σύνταξη ενός maplet είναι:

Περιγραφή:	***Δημιουργεί ένα maplet.***
Γενική σύνταξη:	***όνομα:=maplet(εντολές _Maple,Maplet)***

Για να εκτελέσουμε ένα maplet το οποίο δημιουργήσαμε, πρέπει να χρησιμοποιήσουμε την εντολή `Display:` ως εξής

`Maplets[Display](όνομα);`

Το πακέτο Maplets περιέχει τρία υποπακέτα, τα :

1. ***Examples*** είναι το υποπακέτο στο οποίο περιέχονται ορισμένα χρήσιμα maplets, όπως τα:
   - Alert
   - GetColor
   - GetExpression
   - GetInput
   - Question
   - Confirm
   - GetEquation
   - GetFile
   - Integration Message
   - Selection ShowTable

Παράδειγμα: 17-9

Στο παράδειγμα αυτό θα ανοίξουμε ένα πλαίσιο διαλόγου στο οποίο θα μπορούμε να εισάγουμε μια έκφραση της οποίας στη συνέχεια θα υπολογίζεται η παράγωγος ως προς x. Το αποτέλεσμα θα το επιστρέφει μέσα στο φύλλο εργασίας.

Λύση:

Ενεργοποιούμε το πακέτο `Maplets[Examples]`.

```
> with(Maplets[Examples]):
```

Ορίζουμε ως expr την έκφραση που θα εισαγάγουμε από το πλαίσιο διαλόγου.

```
> expr:=GetExpression();
```

$$expr := x$$

Παραγωγίζουμε την έκφραση expr.

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

$$\cos(x)$$

□

Παράδειγμα: 17-10

Στο παράδειγμα αυτό θα ενεργοποιήσουμε το έτοιμο Maplet για τον υπολογισμό του ολοκληρώματος $\int_0^{\pi} \sin x dx$.

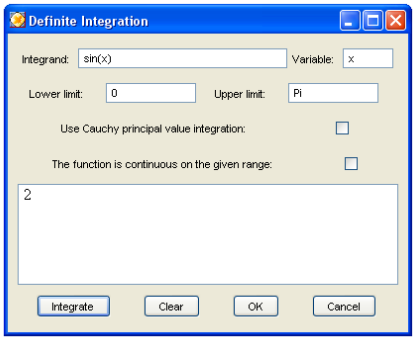
Λύση:

Ενεργοποιούμε το πακέτο `Maplets[Examples]`.

```
> with(Maplets[Examples]):
```

Καλούμε το Maplet integration.

```
>Integration();
```



□

2. ***Elements***, είναι το υποπακέτο στο οποίο περιέχονται όλες εκείνες οι εντολές με τις οποίες μπορούμε να δημιουργήσουμε ένα maplet. Αυτές διακρίνονται στις κατηγορίες:
   - ***Window***- Ορίζει ένα παράθυρο διαλόγου

Περιγραφή:	***Ορίζει ένα παράθυρο διαλόγου.***	
Γενική σύνταξη:	***Window(παράμετροι)***	
Παράμετροι:	reference=´´	Το όνομα αναφοράς.
	title=´´	Ο τίτλος του παραθύρου.
	width =	Το πλάτος του παραθύρου σε pixels.
	height =	Το ύψος του παραθύρου σε pixels.
	menubar = layout = toolbar =	Ορίζει το αντίστοιχο στοιχείο.

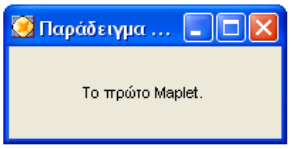
Παράδειγμα: 17-11

Ενεργοποιούμε το πακέτο `Maplets[Elements]`.

```
> with(Maplets[Elements]):
```

Ορίζουμε ως maplet1 το maplet που δημιουργεί ένα παράθυρο.

```
maplet1:= Maplet(
   Window(reference='wind1','title' = "Παράδειγμα
παραθύρου",width=200,height=100,["Το πρώτο Maplet."]
            )
                  ):
Maplets[Display](maplet1);
```



□

- ***Window Body Elements***, είναι κατηγορία στοιχείων που ορίζει ορατά στοιχεία σε ένα παράθυρο και μπορεί να περιέχει:
    - Button
    - ComboBox
    - Label
    - MathMlViewer
    - RadioButton
    - Table
    - Note
    - ToggleButton
    - CheckBox
    - DropDownBox
    - ListBox
    - Plotter
    - Slider
    - TextBox
    - TextField
    - MathMLEditor
- ***Command Elements***
    - CloseWindow- Κλίνει ένα ανοικτό παράθυρο διαλόγου.
    - Evaluate- Υπολογίζει μια διαδικασία του Maple.
    - RunDialog- Εμφανίζει ένα πλαίσιο διαλόγου.
    - RunWindows-Εμφανίζει ένα παράθυρο.
    - ShutDown - Κλίνει μια maplet εφαρμογή.
- ***Layout Elements***- Περιγράφει τον τρόπο με τον οποίο θα εμφανίζονται τα στοιχεία του maplet
    - BoxLayout- Ορίζει μια διάταξη κουτιού.
        - BoxRow- Ορίζει την οριζόντια θέση σχετικά με ένα άλλο στοιχείο.
        - BoxColumn- Ορίζει την κάθετη θέση σχετικά με ένα άλλο στοιχείο.
        - BoxCell- Ορίζει ένα κελί.
    - GridLayout- Ορίζει μια διάταξη πλέγματος.
        - GridCell- Ορίζει ένα κελί σε πλέγμα.
        - GridRow- Ορίζει μια γραμμή σε πλέγμα.
    - Nested Lists –Είναι λίστα αντικειμένων που ορίζεται ως εξής:
      [[αντικείμενο1], [αντικείμενο2],...]

Παράδειγμα: 17-12

Στο παράδειγμα αυτό θα δημιουργήσουμε ένα maplet, στο οποίο θα μπορούμε να εισάγουμε τον τύπο μιας συνάρτησης και το πεδίο στο οποίο θέλουμε την γραφική της παράστασης και εκείνο θα την δημιουργεί.

Λύση:

Ενεργοποιούμε το πακέτο `Maplets[Elements]`.

```
> with(Maplets[Elements]):
```

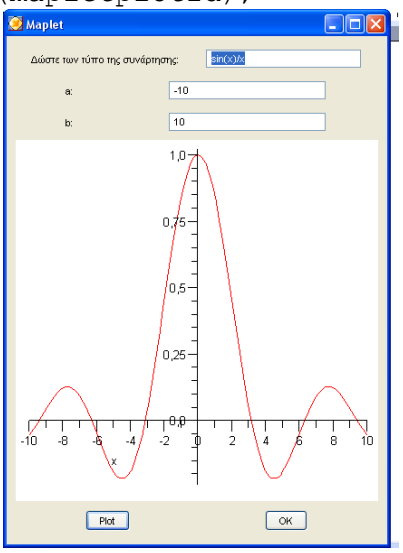
Ορίζουμε το maplet με όνομα mapletplot2d.

```
> mapletplot2d := Maplet(
      [
```

```
["Δώστε τον τύπο της συνάρτησης:",
                TextField['TF2d']("sin(x)/x")],
["a:", TextField['a']("-10")],
["b:", TextField['b']("10")],
Plotter['PL1']( plot(sin(x)/x, x = -10..10) ),
[Button("Plot", Evaluate('PL1' = 'plot(TF2d, x =
     a..b)') ),Button("OK", Shutdown(['TF2d']))]
]
):
```

Εκτελούμετο mapletplot2d

```
>Maplets[Display](mapletplot2d);
```



- ***Dialog Elements***- Στοιχεία διαλόγου τα οποία έχουν συγκεκριμένη διάταξη.
  - AlertDialog
  - ColorDialog
  - ConfirmDialog
  - FileDialog
  - InputDialog
  - MassageDialog
  - QuestionDialog
- ***MenuBar Elements***- Ορίζει την εμφάνιση και τη συμπεριφορά των μενού.
  - MenuBar
  - Menu

- MenuItem
- MenuSeparator
- PopupMenu

- ***Toolbar Elements***- Ορίζει την εμφάνιση και τη συμπεριφορά των tool bars.
  - ToolBar
  - ToolBarButton
  - ToolBarSeparator
- ***Other Elements***
  - Action- Ορίζει μια ενέργεια.
  - Argument- Ορίζει μια υπόθεση για κάποιον υπολογισμό.
  - Font- Ορίζει τη γραμματοσειρά.
  - Image- Εισάγει μια εικόνα σε μια εφαρμογή.
  - Item- Ορίζει ένα στοιχείο σε ένα box element.

Το τελευταίο υποπακέτο του πακέτου εντολών Maplets είναι τα Tools

3. ***Tools***- είναι χρήσιμα εργαλεία για τη δημιουργία και τον έλεγχο ενός maplet.

- AddAttribute
- AddContent Get
- ListBoxSplit
- Print
- Set
- SetTimeout
- StartEngine
- StopEngine

Παράδειγμα: 17-13

Στο παράδειγμα αυτό θα δημιουργήσουμε ένα maplet το οποίο θα δημιουργεί ένα παράθυρο και ένα «κλασικό» μενού, από το οποίο θα μπορούμε να βρούμε το ολοκλήρωμα, την παράγωγο, τη σειρά Maclaurin, και να λύσουμε την αντίστοιχη εξίσωση μιας έκφρασης.

Λύση:

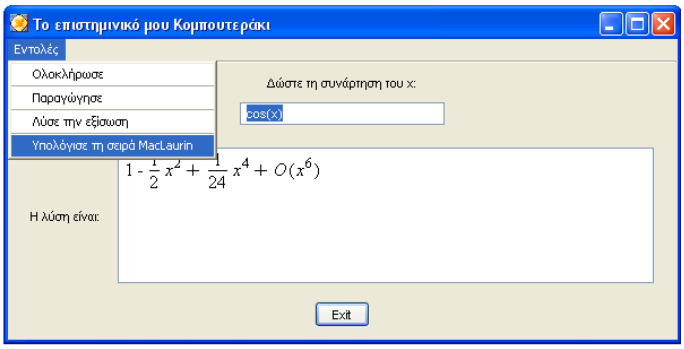
Ενεργοποιούμε το πακέτο `Maplets[Elements]`.

```
> with(Maplets[Elements]):
```

Ορίζουμε το maplet με όνομα minicalc.

```
>minicalc:=Maplet(
      Window(height='300',width='600',
      'title'="Το επιστημινικό μου Κομπουτεράκι",
      'menubar'='menu1',
      ["Δώστε τη συνάρτηση του x:",
```

```
                [TextField(reference='tf1')],
      [[BoxCell("Η λύση είναι:")],
                [MathMLViewer(reference='viewfield')]],
      Button("Exit", Shutdown())]),
      MenuBar(reference='menu1',Menu("Εντολές",
            MenuItem("Ολοκλήρωσε",Evaluate('viewfield' =
                  'int(tf1, x)') ),
            MenuSeparator(),
            MenuItem("Παραγώγησε", Evaluate('viewfield' =
                  'diff(tf1, x)')),
            MenuSeparator(),
            MenuItem("Λύσε την εξίσωση",
                  Evaluate('viewfield' = 'solve(tf1, x)')),
            MenuSeparator(),
            MenuItem("Υπολόγισε τη σειρά MacLaurin",
                  Evaluate('viewfield' = 'taylor(tf1,
            x=0)'))
            )
      )
):
>Maplets[Display](minicalc);
```



Ευρετήριο Εντολών

—E—

—F—

—G—

—H—

—I—

—J—

—L—

—M—

—N—

—O—

—P—

—Q—

—R—

—S—

—T—

—U—

—V—

—W—

—Z—

Βιβλιογραφία

- Char W., (2003). *Maple 9 Learning Guide, Maplesoft*, a division of Waterloo Maple Inc.
- Heck A., (2003). *Introduction to Maple, 3rd edition*, Springer-Verlag-New York.
- Kraft R. (2007). *Maple For Math Majors*, http://ems.calumet.purdue.edu/mcss/kraftrl/mfmm/ *(20/6/2007)*
- Monagan M, Geddes K., Heal K., Labahn G., Vorkoetter S., McCarron J., DeMarco P. (2003). *Maple 9 Introductory Programming Guide*, - Waterloo Maple Inc.
- *Maple Group (2001). Maplets Beginner's Guide*, Waterloo Maple Inc, Canada.
- *Maple Group (2007). Maple 11 User Manual*, Waterloo Maple Inc, Canada.
- *Soukup R. (2001). Engineering According To Maple, International Conference on Engineering Education.*
- Tocci C., Adams S. (1996). *Applied Maple for Engineers and Scientists*, Artch House Boston-London
- Wright, F. (2001). *Computing with Maple*, CRC Press.

\