Κεφάλαιο 9ο: Γραφικές Παραστάσεις

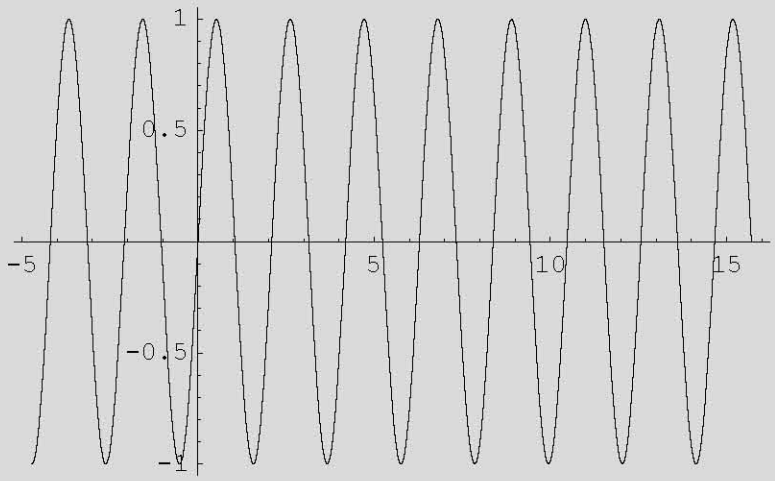
9.1 Διδιάστατες Γραφικές Παραστάσεις

9.1.1 Γραφικές παραστάσεις καμπυλών.

Όταν θέλουμε να κάνουμε την γραφική παράσταση μιας καμπύλης c πρέπει κατ' αρχήν να δούμε με ποιό τρόπο περιγράφεται η c. Περιπτώσεις:

1. Η καμπύλη δίνεται ως γράφημα μιας συνάρτησης f του x. Π.χ f[x]=Sin[3 x]. Τότε μπορούμε να χρησιμοποιήσουμε την Plot:

```
Plot[Sin[3 x], {x, -3 Pi / 2, 5 Pi}];
```



Σχόλια: Πατώντας πάνω στο γραφικό με το ποντίκι ο δείκτης μετατρέπεται σε ρόμβο που έχει ενα σταυρό μέσα και εμφανίζεται ένα πλαίσιο με λαβές. Με το αριστερό πλήκτρο του ποντικιού μπορούμε να μετακινήσουμε το γραφικό. Με τις "λαβές" μπορούμε να αλλάξουμε τις διαστάσεις του γραφικού, συροντάς της σε διάφορες κατευθύνσεις.

Αν θέλουμε να "ζουμάρουμε" σε κάποιο κομμάτι του γραφήματος θα μπορούσαμε να σύρουμε κατάλληλα τις λαβές πατώντας ταυτόχρονα το Ctrl.

Πατώντας πάνω στο γράφημα και στη συνέχεια το Ctrl(συνεχόμενα)και πάλι κλίκ στο γράφημα μπορούμε να δούμε να σχηματίζεται ένα σταυρός και τις συντεταγμένες οποιοδήποτε σημείου (εκείνου που σημαδεύει το κέντρο του σταυρού) κάτω αριστερά στο παράθυρο.

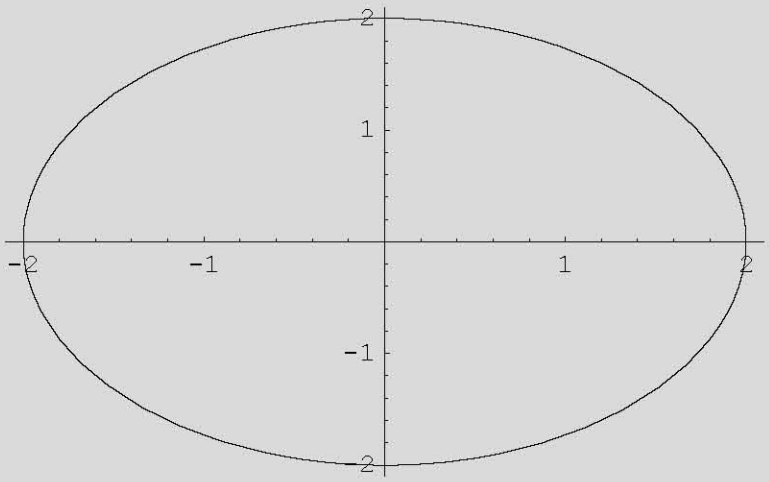
Η Plot παίρνει κάποια σημεία του x στο δοθέν διάστημα (π.χ {x,-3 Pi/2,5 Pi})και χρησιμοποιώντας τα, υπολογίζει την συνάρτηση y (εδώ Sin[3 x]) και με τα ζεύγη (x,y) κάνει την γραφική παράσταση. Τα σημεία x που επιλέγονται λέγονται PlotPoints και είναι λίγα. Για αυτό υπάρχουν και οι ατέλεις στον σχεδιασμό της καμπύλης. Προσέξτε τις ατέλειες για παράδειγμα στον σχεδιασμό της Sin[3 x] όταν μεγενθύνετε τραβώντας απο τις λαβές. Όμως με PlotPoints->40 για παράδειγμα, μπορείται να έχετε ένα καλύτερο αποτέλεσμα.

Άσκηση: Μάθετε περισσότερα για τις επιλογές Compiled, MaxBend, PlotPoints και PlotDivision και

προσπαθείστε να τις εφαρμόσετε στο σχεδιασμό της Sin[3 x] με {x,0,4 Pi}. Για να δείτε καλύτερα το προβλημα στο σχεδιασμό θέστε στην Plot το AspectRatio->Automatic και κοιτάξτε την καμπύλη π.χ στα βαθουλώματα . Δεν είναι τόσο ομαλή όσο σε άλλα σημεία της. Για την AspectRatio θα μιλήσουμε αμέσως παρακάτω.

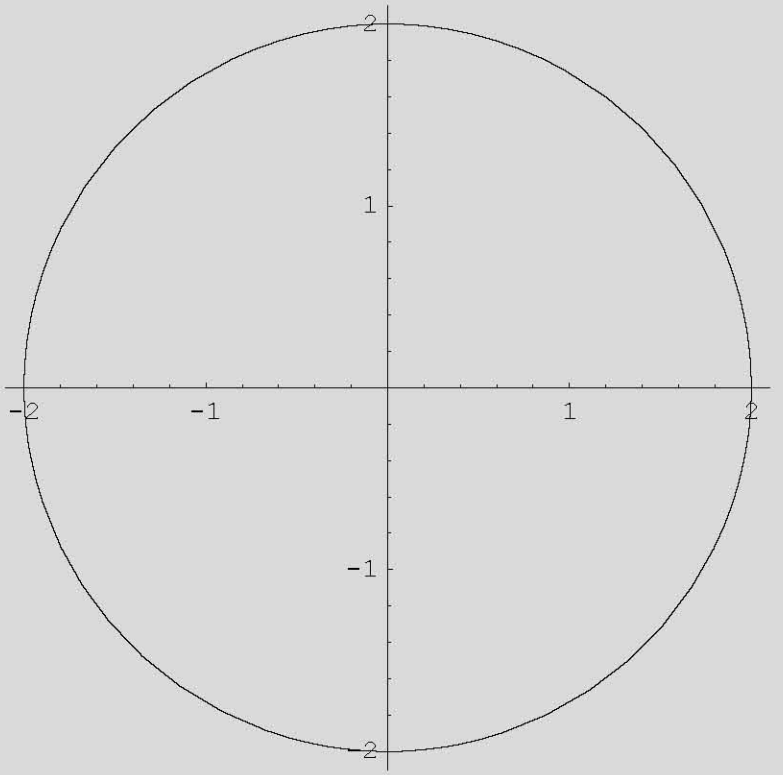
2. Όταν η καμπύλη δεν είναι γράφημα συνάρτησης(π.χ όταν η καμπύλη είναι ένας κύκλος) τότε δεν μπορούμε να χρησιμοποιήσουμε την Plot. Ας υποθέσουμε λοιπόν ότι η καμπύλη ορίζεται με παραμετρικές εξισώσεις. Π. χ οι παραμετρικές εξισώσεις ενός κύκλου με ακτίνα ίση με r είναι x=r Cos[t], y=r Sin[t], με παράμετρο t στο διάστημα [0,2 π]. Σε τέτοιες περιπτώσεις χρησιμοποιούμε την εντολή ParametricPlot

```
ParametricPlot[{2 Cos[t], 2 Sin[t]}, {t, 0, 2 Pi}];
```



Σχόλιο: Το σχήμα που προέκυψε μοιάζει με έλλειψη και όχι με κύκλο! Αυτό ωφείλεται στο γεγονός ότι αυτόματα καθορίζεται (απο την Plot) ο λόγος του ύψους του πλαισίου(που περιβάλλει "αόρατα" το γράφημα, πατείστε πάνω στο γράφημα για να το δείτε) προς το πλάτος του πλαισίου να είναι 1 αλλά 1/ Χρυσή Τομή δηλ. περίπου 61.8034 % είναι μικρότερο το ύψος απο το πλάτος! Αν θέλουμε να εμφανίσουμε το σχήμα όπως πραγματικά είναι δεν έχουμε παρα να θέσουμε AspectRatio->1(υψος δια πλάτος=1 δηλ. τετράγωνο πλαίσιο) ή καλύτερα AspectRatio->Automatic (δηλ. η μονάδα μέτρησης μήκους στον Ox =μονάδα μέτρησης στον Oy):

```
ParametricPlot[{2 Cos[t], 2 Sin[t]},
  {t, 0, 2 Pi}, AspectRatio → Automatic];
```

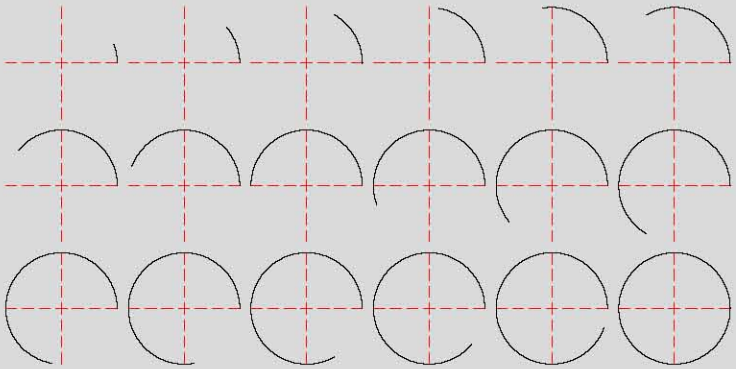


Το AspectRatio→Automatic *είναι στις περισσότερες περιπτώσεις αρκετά καλό διότι δεν παραμορφώνει την καμπύλη. Σε μερικές όμως περιτώσεις όταν για παράδειγμα όταν έχουμε μεγάλει δυσαναλογία του μήκους του πεδίου του* x(*πεδίο ορισμού*) *προς το μήκος του πεδίου του* y(*πεδίο τιμών*) *χρησιμοποιούμε μια συγκεκριμμένη τιμή του* AspectRatio *ώστε να έχουμε ένα κατανοητό γράφημα*.

Άσκηση. Σχεδιάστε την καμπύλη 1/x *για* {x,-30,30} *και* AspectRatio-> Automatic. *Τι παρατηρείτε*;. *Αλλάξτε το* AspectRatio *π.χ σε* 1/2 *ή* 1 *ή κάτι άλλο ή αφαιρέστε το* AspectRatio *για να βελτιώσετε το γράφημα*.

Σχόλιο: *Καθώς το* t *μεγαλώνει απο* 0 *προς το* 2 *π όλο και μεγαλύτερο μέρος του κύκλου σχεδιάζεται. Για να δούμε τις διαδοχικές "φάσεις" μπορούμε να χρησιμοποιήσουμε την εντολή* Show *σε συνδυασμό με την εντολή* GraphicsArray *π.χ*

```
akoloythia = Table[ParametricPlot[{2 Cos[t], 2 Sin[t]}, {t, 0, n Pi},
    DisplayFunction → Identity, Ticks → None, AspectRatio → 1,
    AxesOrigin → {0, 0}, PlotRange → {{-2, 2}, {-2, 2}}, AxesStyle →
     {RGBColor[1, 0, 0], Dashing[{.1, .05}]}], {n, 1 / 9, 2, 1 / 9}];
kommatia = Partition[akoloythia, 6];
Show[GraphicsArray[kommatia]];
```



Η *επιλογή* DisplayFunction→Identity *αναγκάζει την* ParametricPlot *να μην σχεδιάσει*. Η akoloythia *περιέχει* 18 *γραφικές παραστάσεις στη σειρά και η* Partition *κόβει την* akoloythia *σε κομμάτια των* 6 *γραφημάτων*. Οπότε *στα* kommatia *θα έχουμε* 3 *γραμμές επι* 6 *γραφήματα το καθένα*. Η GraphicsArray *τοποθετεί τα γραφήματα του πίνακα στη σειρά(καρέ - καρέ) σε όμοια πλαίσια και η* Show *εμφανίζει τελικά το αποτέλεσμα*! *Για τις επιλογές που υπάρχουν μέσα στο* ParametricPlot *όπως για παράδειγμα* PlotRange→{{-2,2},{-2,2}} *έχουν βασικό σκοπό να κάνουν πιο κατανοητά και ελκυστικά τα γραφικά*. Με *την* PlotRange *λέμε ποιά σημεία επιθυμούμε να εμφανίζονται στην εικόνα*. Με PlotRange-> All *εμφανίζονται όλα τα σημεία της καμπύλης*.

Άσκηση: Μάθετε *περισσότερα για τις παραπάνω επιλογές της* ParametricPlot *μεσω του* Help *και προσπαθήσετε να πειραματιστείτε με αυτές*.

Άσκηση: *Σχεδιάσετε την* 3^{-x^2} *για* {x,-5,5} *διαλέγοντας διαφορετικά* PlotRange *και* AspectRatio. *Π.χ θέστε* PlorRange->All, PlotRange->{{-1,1},All}, PlotRange->{All,{-1,.02}},PlotRange->{-4,1}, PlotRange->{{2,10},{0,.02}} *κ.ο.κ. και διάφορες επιλογές για* AspectRatio *π.χ* Automatic, 1/2 *κ.ο.κ*

Πρέπει να έχουμε υπόψη ότι η ParemetricPlot είναι γενικότερη της Plot διότι μπορεί να σχεδιάσει και καμπύλες που τέμνουν τον ευατό τους(όπως π.χ ένα κύκλο). Φυσικά κάθε συνάρτηση με εξίσωση y=f[x] για κάποια f, μπορεί να θεωρηθεί και παραμετρικά θέτοντας για παράμετρο t το ίδιο το x. Έτσι για παράδειγμα η y=Sin[3x] μπορεί να σχεδιαστεί και ως

```
ParametricPlot[{t, Sin[3 t]}, {t, -3 Pi / 2, 5 Pi}]
```

3. Η *καμπύλη δίνεται ως γράφημα μιας συνάρτησης, που ορίζεται πεπλεγμένα δηλ. μέσω εξίσωσης*.
Π.*χ για να κάνουμε την γραφική παράσταση της παραβολής* x^2+3 y^2==5 *στο διάστημα* [-3,3] *θα πρέπει να λύσουμε ως προς* y (y=Sqrt[(5- x^2)/3)]ήy=-Sqrt[(5- x^2)/3)]) *και μετά να χρησιμοποιήσουμε την* Plot. *Καλύτερα όμως για θα ήταν να καλέσουμε την εντολή* ImplicitPlot *χωρίς να χρειαστεί να κάνουμε παραπάνω δουλειά*:

```
ImplicitPlot[x^2 + 3 y^2 == 5, {x, -3, 3}]
```

```
ImplicitPlot[x^2 + 3 y^2 == 5, {x, -3, 3}]
```

Δυστυχώς δεν μας έβγαλε τίποτα! Ας δούμε το λόγο. Αν πατήσουμε F1 και δώσουμε την λέξη ImplicitPlot θα διαπιστώσουμε ότι αυτή βρίσκεται στο πακέτο Graphics`ImplicitPlot`. Το Graphics είναι η διεύθυνση (folder)μέσα στο οποίο βρίσκεται το "πακέτο"(πρόγραμμα) ImplicitPlot. Μέσα στο πακέτο αυτό έχει ορισθεί η συνάρτηση ImplicitPlot. Πακέτο λέμε ένα αρχείο με τους ορισμούς κάποιων συναρτήσεων που έχουν σκοπό να κάνουν μια ειδική εργασία π.χ να σχεδιάσουν μια συνάρτηση όταν αυτή δίνεται σε πολικές συντεταγμένες κ.ο.κ. Απο την στιγμή που καλούμε ένα πακέτο δεν χρειάζεται να το ξανακαλέσουμε παρακάτω για δεύτερη φορά! Ένα πακέτο το καλούμε με την εντολή Needs Π.χ Needs["Graphics`ImplicitPlot`"]. Πιο απλά συνήθως γράφουμε το διπλό << στην θέση του Needs π.χ.

```
<< Graphics`ImplicitPlot`
```

```
- ImplicitPlot::shdw : Symbol ImplicitPlot appears in multiple
    contexts {Graphics`ImplicitPlot`, Global`}; definitions in context
    Graphics`ImplicitPlot` may shadow or be shadowed by other definitions.
```

Σχόλια: 1. Προσέξτε αμέσως μας έβγαλε ένα μήνυμα που λέει ότι συνάντησε την λέξη ImplicitPlot σε δύο διαφορετικά context στο Global και στο Graphics`ImplicitPlot`. Το **context** είναι το όνομα του πακέτου ή ενός προγράμματος του. Με το όνομα αυτό καθοδηγούμε το *Mathematica* που θα πάει να ψάξει το σωστό ορισμό. Το Global είναι ένα γενικό πακέτο και ανοίγει αυτόματα κάθε φορά που ανοίγουμε το *Mathematica*. Είναι "ο χώρος υποδοχής" σε όποιον μπαίνει στο *Mathematica*. Για παράδειγμα κάθε φορά που γράφομε μια λέξη(π.χ ImplicitPlot) ή ένα σύμβολο μιας μεταβλητής, αποθηκεύεται στο Global εκτός και αν εμείς του ζητήσουμε να το αποθηκεύσει σε άλλο αρχείο. Αυτή η λέξη ή το σύμβολο θα παραμένει στο Global έως ότου κλείσουμε το *Mathematica* ή την/το σβήσουμε με την εντολή Remove. Δυστυχώς εδώ η λέξη ImplicitPlot δεν εχει ορισθεί στο Global με κάποιο τρόπο. Δεν υπάρχει ούτε στο πακετο του συστήματος δηλ. το **System** (στο System βρίσκονται αποθηκευμένοι οι ορισμοί των βασικών συναρτήσεων όπως π.χ της πρόσθεσης, της Plot κ.ο.κ). Έτσι το *Mathematica* θεωρεί την ImplicitPlot σαν κάτι νέο και δεν έχει ιδέα περι τίνος πρόκειται ή περί του τι κάνει ή ποια συνάρτηση παριστάνει! Όταν καλέσουμε το πακέτο <<Graphics`ImplicitPlot` τότε το *Mathematica* ανακαλύπτει μέσα σ' αυτό ξανά τη συνάρτηση ImplicitPlot γι'αυτό παραπονιέται!!! Η ImplicitPlot παραμένει μια συνάρτηση του Global επειδή το Global επισκιάζει το (ειδικό) πακέτο Graphics`ImplicitPlot` Έτσι λοιπόν ούτε τώρα θα μας δουλέψει η ImplicitPlot παρόλο που ανοίξαμε το σωστό πακέτο :

```
ImplicitPlot[x^2 + 3 y^2 == 5, {x, -3, 3}]
```

```
ImplicitPlot[x^2 + 3 y^2 == 5, {x, -3, 3}]
```

Θα πρέπει να διαγράψουμε την **ImplicitPlot** απο το τρέχον αρχείο(εδώ το Global) γράφοντας:

```
Remove[ImplicitPlot]
```

Ας ρωτήσουμε τώρα ποιες συναρτήσεις και ποια σύμβολα βρίσκονται ακόμα μέσα στο Global

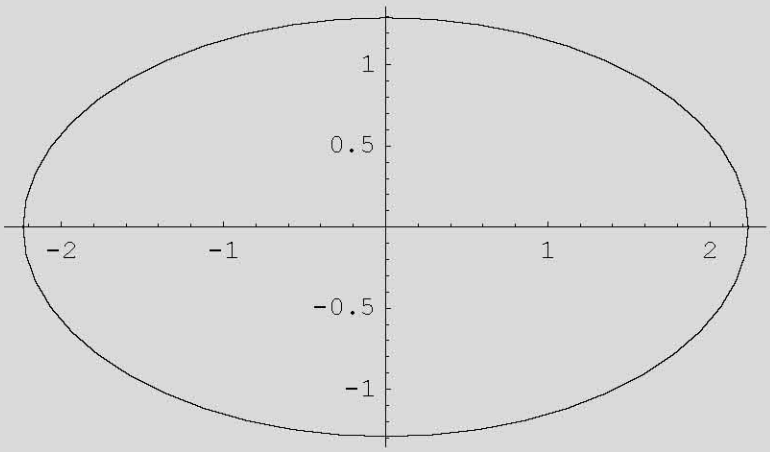
```
? Global`*
```

Global`

akoloythia kommatia n t x y

Όπως βλέπουμε **δεν υπάρχει** ποιά **η** ImplicitPlot *στο* Global *οπότε ας την καλέσουμε ξανά*

```
ImplicitPlot[x^2 + 3 y^2 == 5, {x, -3, 3}];
```



Βλέπουμε *λοιπόν ότι επειδή* ***δεν βρέθηκε η*** ImplicitPlot *στο* Global(ούτε φυσικά στο System), *το Mathematica* ***αναγκάστηκε να ψάξει*** (*με κάποια συγκεκριμμένη σειρά -αυτήν που υπάρχει μέσα στο* $ContextPath) *στα άλλα πακέτα που ήδη έχουν* ***ανοιχτεί*** *δηλ. το* System ***και το*** Graphics`ImplicitPlot` ***και*** *φυσικά βρήκε στο δεύτερο πακέτο την* (***σωστή) συνάρτηση*** *και την εφάρμοσε*!

2. *Για μερικές εντολές δεν χρειάζεται να καλέσουμε κάποιο ειδικό πακέτο για να εκτελεστούν διότι βρίσκονται στο* System και συνεπώς είναι ετοιμοπαράδοτες. *Εδώ μέσα βρίσκονται όλες οι γνωστές συναρτήσεις του* Kernel *του Mathematica. Γράφοντας* $Packages *μπορούμε να δούμε ποιά πακέτα έχουν φορτωθεί απο το Mathematica μέχρι τώρα. Με την εντολή* Context[f] *παίρνουμε το όνομα του πακέτου που περιέχει την συνάρτηση* f *ενώ με* $Context *βρίσκουμε το* Context *στο οποίο βρισκόμαστε αυτή την στιγμή. Με* $ContextPath *παίρνουμε την σειρά προτεραιότητος που θα πάει να ψάξει το Mathematica για να βρεί τους ορισμούς των συμβόλων και των συναρτήσεων. Αριστερότερα είναι εκείνο που ανοίχθηκε τελευταίο (απο εμάς με το* σύμβολο << *ή ίσως και απο το Mathematica χωρίς να το αντιληφθούμε) π.χ*

```
$Packages
```

```
{Utilities`FilterOptions`, Graphics`ImplicitPlot`, Global`, System`}
```

```
Context[Plot]
Context[Plus]
Context[ImplicitPlot]
$Context
$ContextPath

System`
```

```
System`
```

```
Graphics`ImplicitPlot`
```

```
Global`
```

```
{Graphics`ImplicitPlot`, Utilities`FilterOptions`, Global`, System`}
```

Άσκηση. Να βρείτε το είδος της καμπύλης που παράγει η παρακάτω εντολή.

```
ImplicitPlot[x^2 + 3 y^2 == 5, {y, -3, 3}]
```

4. Η *καμπύλη δίνεται ως γράφημα μιας συνάρτησης, που ορίζεται σε πολικές συντεταγμένες*. Τότε θα χρησιμοποιήσουμε την PolarPlot. Το *πακέτο* Graphics`Graphics` *πρέπει να ανοιχθεί κάθε φορά που χρειαζόμαστε την* PolarPlot.

```
<< Graphics`Graphics`
PolarPlot[Cos[2 t], {t, 0, 2 Pi}, Background → RGBColor[0, 0, .5],
  PlotStyle → {RGBColor[1, 0, 0], Thickness[.01]}];
```

Με PlotStyle *καθορίζουμε πως θέλουμε να σχεδιαστεί η γραφική παράσταση* ενώ με Background→-RGBColor[0,0,.5] *επιλέξαμε ένα βαθύ μπλέ για φόντο*. Το R(=0) *στη λέξη* RGBColor *είναι το κόκκινο το* G(=0) *το πράσινο και το* B(=.5) *είναι το μπλέ*. Ας *παρατηρήσουμε ότι αυτόματα επιλέχθηκε ένα κιτρινωπό χρώμα για τους άξονες για λόγους αντίθεσης*. Αν *δεν μας αρέσει το χρώμα αυτό*, τότε με AxesStyle->..... *μπορούμε να το αλλάξουμε*.

Σχόλιο: Αν *τώρα θέλουμε να δούμε τις διαδοχικές φάσεις σχεδιασμού της καμπύλης σε κίνηση θα γράψουμε*

```
pinakas = Table[PolarPlot[Cos[2 t], {t, 0, n Pi}], {n, 0.1, 2, 0.1}];
```

Αν *κάνουμε διπλό κλίκ πάνω σε μια απο τις παραπάνω γραφ. παραστά σεις θα δούμε τις φάσεις σχεδιασμού*! Επειδή *τα σχήματα δεν βγήκαν όπως ακριβώς περιμέναμε ας κάνουμε κάποιες αλλαγές στους αξονες*

```
Clear[pinakas]
pinakas = Table[PolarPlot[Cos[2 t], {t, 0, n Pi},
    PlotRange → {{-1, 1}, {-1, 1}}, Ticks → None], {n, 0.1, 2, 0.1}];
```

Το *ίδιο αποτέλεσμα με το παραπάνω θα πέρναμε με την* Do *χωρίς την χρήση της* Table. *Δοκιμάστε το παρακάτω*

```
Do[PolarPlot[Cos[2 t], {t, 0, n Pi},
  PlotRange → {{-1, 1}, {-1, 1}}], {n, 0.1, 2, 0.1}]
```

Αν μικράνουμε το βήμα 0.1 *στο* {n,0.1,2,0.1} *θα έχουμε περισσότερα καρέ και άρα πιο αργή κίνηση. πχ μπορείτε να αντικαταστήσετε* {n,0.1,2,0.01} *στο* Do *και να δοκιμάσετε ξανά*!

Άσκηση: Η *παραμετρική εξίσωση της προηγούμενης καμπύλης είναι*(σε καρτεσιανές συντεταγμένες) x[t]=-Cos[2 t] Cos[t] *και* y[t]=Cos[2 t] Sin[t]. *Προσπαθείστε να πάρετε τις διαδοχικές φάσεις χρησιμοποιώντας την* ParametricPlot *αντί της* PolarPlot *και τις παραμετρικές εξισώσεις αντι τις πολικές*.

5. Η *καμπύλη δίνεται ως ένα* ***πεπερασμένο σύνολο σημείων*** **data** *που έχουν προκύψει απο μετρήσεις π.χ θερμοκρασίας ή πίεσης κ.ο.κ π.χ*

```
data = Table[{x, 1/x}, {x, 1, 10}]
synarthsh = Interpolation[data]
Plot[synarthsh[x], {x, 1, 10},
  PlotRange → Automatic, AxesOrigin → {0, 0}];
```

$$\left\{\{1, 1\}, \left\{2, \frac{1}{2}\right\}, \left\{3, \frac{1}{3}\right\}, \left\{4, \frac{1}{4}\right\}, \left\{5, \frac{1}{5}\right\}, \left\{6, \frac{1}{6}\right\}, \left\{7, \frac{1}{7}\right\}, \left\{8, \frac{1}{8}\right\}, \left\{9, \frac{1}{9}\right\}, \left\{10, \frac{1}{10}\right\}\right\}$$

```
InterpolatingFunction[{{1, 10}}, <>]
```

Η *συνάρτηση* Interpolation *κάνει παρεμβολή στα σημεία των* data *και επιστρέφει μια ομαλή καμπύλη που περναει απο τα σημεία αυτά*.

Μπορούμε *να χρησιμοποιήσουμε και την* **ListPlot** *και να ενώσουμε τα σημεία των* data *με ευθύγραμμα τμήματα αλλά το αποτέλεσμα δεν είναι τόσο ικανοποιητικό*:

```
ListPlot[data, PlotJoined → True,
  PlotRange → Automatic, AxesOrigin → {0, 0}];
```

Η *επιλογή* PlotJoined→True *αναγκάζει την* ListPlot *να ενώνει με ευθύγραμμα τμήματα τα σημεία που σχεδιάζει*! *Είναι πολύ χρήσιμη στην περίπτωση που τα* data *μας δεν παριστάνουν σημεία μιας συνάρτησης*. *Π.χ με την* ListPlot *μπορούμε να ζωγραφίσουμε αστείες φατσούλες*:

```
koryfes =
  {{0, 0}, {0.5, 1}, {0.7, 0.5}, {0.5, 0.3}, {0.6, 0.2}, {.3, 0}};
fatsoyla = ListPlot[koryfes, PlotJoined → True, PlotRange → Automatic,
   AxesOrigin → {0, 0}, DisplayFunction → Identity];
mati = Graphics[{PointSize[0.03], RGBColor[1, 0, 0],
    Point[{0.54, 0.72}]}];
Show[fatsoyla, mati, DisplayFunction → $DisplayFunction]
```

Με *την* **Show** *μπορούμε να δούμε πολλά γραφήματα μαζί. Θα μιλήσουμε ξανα για αυτήν παρακάτω.* Με **DisplayFunction→Identity** *αποφύγαμε να σχεδιάσουμε το περίγραμμα του κεφαλιού ενώ με* Display-Function→$DisplayFunction *επαναφέραμε την δυνατότητα στο* Show *να μας εμφανίσει την φατσούλα μαζί με το κόκκινο μάτι.*

Σχόλιο: Με *την* Graphics *μπορούμε να σχεδιάσουμε σημεία, δίσκους,ευθύγραμμα τμήματα, πολύγωνα,να προσθέτουμε κείμενο και πολλά άλλα. Ένα κοίταγμα στο* Help *του* Graphics *θα σας πείσει...*
Άσκηση: *Χρησιμοποιώντας την συνάρτηση* Polygon *για τις* koryfes *να ζωγραφίσετε ξανά την φατσούλα αλλά με το εσωτερικό του κεφαλιού να είναι μπλέ (και το μάτι κόκκινο).*

6. Στην περίπτωση που η y ικανοποιεί μια εξίσωση με παραγώγους ή μερικές παραγώγους (δηλ. ***κάποια διαφορική εξίσωση***) τότε θα πρέπει πρώτα να επιλύσουμε την εξίσωση και μετά να κάνουμε την γραφική παράσταση της y . Π.χ

```
eqn = y''[x] + 5 Log[y[x]] == 0
soll = NDSolve[{eqn, y'[0] == 1, y[0] == 1}, y[x], {x, 0, 4}]
Plot[y[x] /. soll, {x, 0, 4}, PlotStyle → RGBColor[1, 0, 0]];
```

```
5 Log[y[x]] + y''[x] == 0
```

```
{{y[x] → InterpolatingFunction[{{0., 4.}}, <>][x]}}
```

Προσέξτε ότι η λύση y[x] δεν είναι ολόκληρη συνάρτηση! Γνωρίζουμε μόνο κάποια συγκεκριμμένα σημεία που ανήκουν στην λύση y και αυτά μαζεύονται σε κάποια λίστα που την συμβολίζουμε <>. Με βάση αυτή την λίστα και με την μέθοδο της παρεμβολής (στα αγγλικά interpolation δηλ. κάτι παρόμοιο όπως κάναμε και στην προηγούμενη περίπτωση 5) υπολογίζονται προσεγγιστικά οι τιμές της y στο διάστημα [0,4]. Με y[x]/.Soll αντικαθιστούμε την άγνωστη y[x] με τις προσεγγιστικές τιμές της.

Άσκηση: Αφού λυθεί με χρήση της NDSolve,το σύστημα των εξισώσεων {x'[t]==-y[t]-x[t]^2, y'[t]==2 x[t]-y[t], x[0]==y[0]==1} ως προς τις άγνωστες συναρτήσεις {x,y} που θεωρούνται συναρτήσεις του t με Πεδίο Ορισμού το διάστημα $t \in [1,10]$, να σχεδιαστεί η προκύπτουσα καμπύλη {x[t],y[t]} για $t \in [1,10]$ με χρήση της Parametric-Plot. Αν προκύψει πρόβλημα να χρησιμοποιήσετε την Evaluate. Για την χρήση της Evaluate μπορείτε να διαβάσετε την ενότητα που ακολουθει.

9.1.2 Σχεδιάζοντας μια λίστα απο καμπύλες

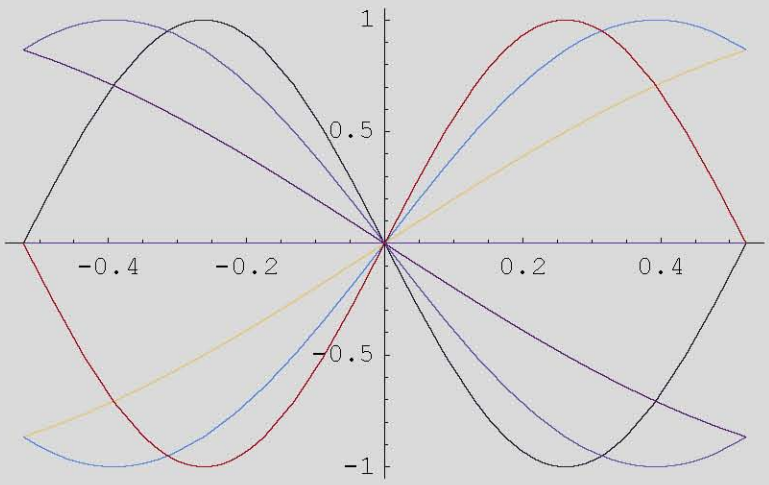
Με την εντολή Show μπορούμε να συνδυάσουμε γραφικές παραστάσεις που έχουν γίνει με διαφορετικά πακέτα(ή καί με τα ίδια αλλά με διαφορετικές εντολές) σε ένα και μοναδικό γράφημα. Στο παράδειγμα που ακολουθεί σχεδιάσαμε δυο γραφήματα g1, g2 και βάλαμε κόκκινα βελλάκια στους άξονες.

```
trigX[x_] := {{x, -0.03}, {x, 0.03}, {x + 0.03, 0}}
trigY[y_] := {{-0.03, y}, {0.03, y}, {0, y + 0.03}}
g1 = Graphics[{RGBColor[1, 0, 0], Polygon[trigX[1.5]],
    Polygon[trigY[1]]}, DisplayFunction → Identity];
g2 = Plot[Sin[3 x], {x, -Pi / 6, Pi / 6}, DisplayFunction → Identity];
g3 = PolarPlot[Cos[2 t], {t, 0, 2 Pi}, DisplayFunction → Identity];
Show[g1, g2, g3, Axes → True, PlotRange → {{-1, 1.53}, {-1, 1.03}},
  DisplayFunction → $DisplayFunction];
```

Με την βοήθεια της Polygon της Graphics σχεδιάζουμε δυο κόκκινα τριγωνάκια ένα στο σημείο x του άξονα Ox και ένα στο σημείο y του άξονα Oy αντίστοιχα.Οι συντεταγμένες τους είναι trigX[x] και trigY[y] αντίστοιχα. Η Show συνδυάζει όλες αυτές μαζί τις γραφικές παραστάσεις.

Αν βέβαια είχαμε γραφικές παραστάσεις που σχεδιάζονται με την ίδια εντολή τότε φυσικά δεν είναι ανάγκη να χρησιμοποιήσουμε την Show.
Ας κάνουμε για παράδειγμα τις γραφικές παραστάσεις των Sin[n x] για {n,-6,6,2} δηλ. για n=-6,-4,-2,0,2,4,6 με την Show και χωρίς αυτήν

```
grafika = Table[Plot[Sin[n x],
    {x, -Pi / 6, Pi / 6}, DisplayFunction → Identity, PlotStyle →
     RGBColor[Random[ ], Random[ ], Random[]]], {n, -6, 6, 2}];
Show[grafika, DisplayFunction → $DisplayFunction];
```



Δώσαμε ένα τυχαίο χρώμα σε κάθε γραφική παράσταση για να πάρουμε ένα αποτέλεσμα πιο ελκυστικό. Το Random[ ] να υπενθυμήσουμε επιστέφει ένα τυχαίο πραγματικό μεταξύ 0 και 1 οπότε με RGBColor[Random[ ],-Random[ ],Random[]] δώσαμε ένα εντελώς τυχαίο χρώμα στην καμπύλη μας Sin[n x].
Χωρίς την Show:

Πρώτα θα μαζεψουμε τα ημίτονα σε ένα πίνακα και μετά θα κάνουμε την γραφική παράσταση του πίνακα αυτού με την Plot:

```
Remove[pinakas]
pinakas = Table[Sin[n x], {n, -6, 6, 2}]
Head[pinakas]
style =
  Table[{RGBColor[Random[], Random[], Random[]]}, {n, -6, 6, 2}];
Plot[pinakas, {x, -Pi / 6, Pi / 6}, PlotStyle → style];

{-Sin[6 x], -Sin[4 x], -Sin[2 x], 0, Sin[2 x], Sin[4 x], Sin[6 x]}
```

```
List
```

```
– Plot::plnr : pinakas is not a machine-size real number at x = -0.523599.

– Plot::plnr : pinakas is not a machine-size real number at x = -0.481117.

– Plot::plnr : pinakas is not a machine-size real number at x = -0.434787.

– General::stop :
   Further output of Plot::plnr will be suppressed during this calculation.
```

Βλέπουμε αυτό δεν απέδωσε! Ο λόγος είναι ότι η Plot **δεν μπορεί να σχεδιάσει μια λίστα** ! (η λέξη Table που βρίσκεται μεσα στο pinakas μπερδεύει την Plot διότι δεν την θεωρεί ως μια συνάρτηση) Άρα θα πρέπει με κάποιο τρόπο να αναγκαστεί να θεωρεί τον pinakas ως λίστα **συναρτήσεων**. Αυτό ακριβώς κάνει η **Evaluate**. Όπως έχουμε δεί και σε προηγούμενα μαθήματα η Evaluate αλλάζει την σειρά των εντολών(των ενεργειών) και δίνεται προτεραιότητα στις εντολές που βρίσκονται μέσα στην Evaluate. Παρακάτω καλέσαμε επιπλέον για διδακτικούς λόγους και το πακετο Graphics`Legend` γιατί θέλουμε να εισάγουμε και ένα πίνακα με **επεξηγηματικές ετικέτες(δηλ. ενα Legend)**:

```
<< Graphics`Legend`
etiketes = Table[ToString[Sin[n x]], {n, -6, 6, 2}];
Plot[Evaluate[pinakas], {x, -Pi / 6, Pi / 6}, PlotStyle → style,
  PlotRange → {{-0.6, 0.6}, {-1, 1}}, PlotLegend → etiketes,
  LegendShadow → {0, 0}, LegendSize → {.6, .5},
  LegendPosition → {0.8, -.6}, LegendBackground → GrayLevel[0.8]];
```

Σχόλια:

α)Προσέξτε μέσα στο style θα πρέπει να υπάρχουν 7 ακριβώς περιγραφές όσες και οι συναρτήσεις που υπάρχουν στο pinakas. Αν υπήρχε μόνο μία π.χ PlotStyle→RGBColor[1,0,0] τότε θα βγαίνανε όλες κόκκινες! Αν υπήρχαν δύο περιγραφές π.χ PlotStyle→{RGBColor[1,0,0],RGBColor[0,1,0]} τότε θα βγάλει τις "μισές" με πράσινο και τις άλλες με κόκκινο!! Γενικά μπορείτε να φτιάξετε το style όπως εσείς θέλετε. Π.χ με PlotStyle→{{RGBColor[1,0,0], Dashing[{.05,.02}]}, {GrayLevel[0.5]},{Hue[.03]},....} θα πάρουμε την πρώτη κόκκινη και διακεκκομένη την δεύτερη συνάρτηση μια απόχρωση του γκρίζου, την τρίτη μια απόχρωση του Hue κ.ο.κ

β)Αν δεν θέλετε τον πίνακα με τις ετικέτες *διαγράψτε ότι χαρακτηριστικό περιέχει την λέξη* Legend. *Αν όμως σας ενδιαφέρει τι ακριβώς κάνει τότε πειραματιστείτε με τα διάφορα χαρακτηριστικά της* Legend *και πάρτε βοήθεια απο το* Help! *Η εντολή* ToString[Sin[n x]] *επιστρέφει την λέξη* "Sin[n x]" . *Όμως επειδή π.χ* Sin[-6 x] *είναι ίσο με* -Sin[6 x] *για αυτό επιστρέφει αυτά που βλέπετε στη πινακίδα. Δοκιμάστε* ToString[Sin[ToString[n x]] *αν δεν σας αρέσει το παραπάνω αποτέλεσμα.*

Άσκηση: *Δίνεται η έλλειψη* x^2+5 y^2==9, *και η καμπύλη που έχει παραμετρική εξίσωση* x[t_]:=Sin[t] *και* y[t_]:=Sin[2 t] *για* t *στο διάστημα* {t,-4,4}. *Να σχεδιάσετε την έλλειψη σε κόκκινο χρώμα με την* ImplicitPlot, *την παραμετρική εξίσωση με την* ParametricPlot *και με διακεκομμένη γραμμή (π.χ θέστε* PlotStyle->Dushing[{.1,.05}]) *και με την* Show *ζωγραφίστε αυτές απο κοινού. Θέστε επίσης στην* Show Background->GrayLevel[.4]

Άσκηση: *Προσπαθείστε να σχεδιάσετε την παράγωγο* (D[x Cos[x],x]) *της* x Cos[x] *για* x *απο* 0 *μέχρι* 10. *Προσέξτε να χρησιμοποιήσετε πρώτα την* Evaluate στην παραγώγιση! *Αλλιώς θα έχετε πρόβλημα.* Η Plot *δεν ξέρει πως να ζωγραφίζει την παράγωγο μιας συνάρτησης*!

Άσκηση: Δείξτε με ένα παράδειγμα ότι η Show δεν έχει πρόβλημα να δουλέψει με ένα πίνακα απο γραφικές παραστάσεις δηλ. δεν χρειάζεται να μπει κάποιου είδους Evaluate για να εμφανιστούν όλες μαζί. Φτιάξτε για παράδειγμα τον πίνακα pinakas=Table[g[n],{n,1,6}] όπου το g[n] ορίζεται να είναι η γραφική παράσταση της Sin[n x] για x απο -Pi/6 εως Pi/6.

Άσκηση: Ελένξτε αν η Plot[{-Sin[6 x],-Sin[4 x],-Sin[2 x],0,Sin[2 x],Sin[4 x],Sin[6 x]},{x,-Pi/6,Pi/6}] έχει πρόβλημα. Επίσης ελένξτε αν η Plot έχει πρόβλημα με μια συνάρτηση που ξεκινάει με If π.χ If[x>0, Sin[x], -x^2]. Τί συμπέρασμα βγάζετε;

9.1.3 Οι επιλογές των διδιάστατων γραφικών.

Με Options[ ] *μπορούμε να δούμε τις επιλογές μιας οποιασδήποτε συναρτήσης*. Π.χ

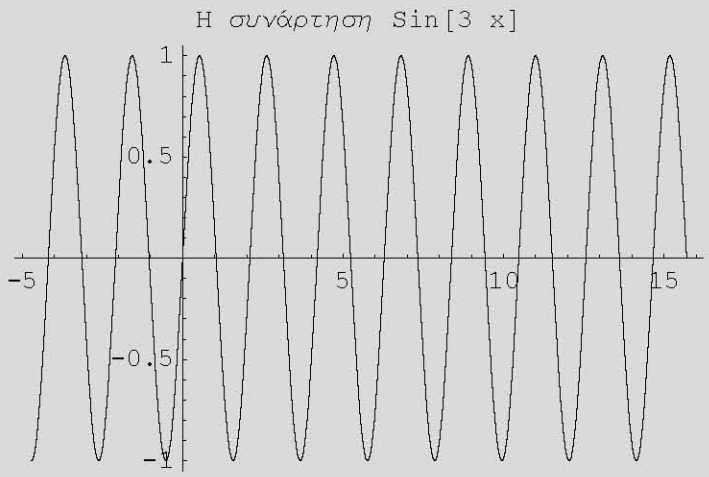
```
Options[Plot]
```

$$\left\{\text{AspectRatio} \to \frac{1}{\text{GoldenRatio}}, \text{Axes} \to \text{Automatic},\right.$$
AxesLabel → None, AxesOrigin → Automatic, AxesStyle → Automatic,
Background → Automatic, ColorOutput → Automatic, Compiled → True,
DefaultColor → Automatic, Epilog → {}, Frame → False,
FrameLabel → None, FrameStyle → Automatic, FrameTicks → Automatic,
GridLines → None, ImageSize → Automatic, MaxBend → 10.,
PlotDivision → 30., PlotLabel → None, PlotPoints → 25,
PlotRange → Automatic, PlotRegion → Automatic, PlotStyle → Automatic,
Prolog → {}, RotateLabel → True, Ticks → Automatic,
DefaultFont :→ $DefaultFont, DisplayFunction :→ $DisplayFunction,
FormatType :→ $FormatType, TextStyle :→ $TextStyle$\left.\right\}$

Εδώ *εμφανίζονται οι προεπιλεγμένες τιμές στις επιλογές*. Π.χ *η προεπ. τιμή της* AspectRatio *είναι* $\text{AspectRatio} \to \frac{1}{\text{GoldenRatio}}$, *των αξόνων είναι* Axes→Automatic *κ.ο.κ Χρειαζόμαστε ένα βιβλίο για να αναλύσουμε διεξοδικά κάθε μια απο τις παραπάνω επιλογές. Θα αναφερθούμε με συντομία σε κάποιες απ'αυτές που δεν έχουμε δεί έως τώρα.*

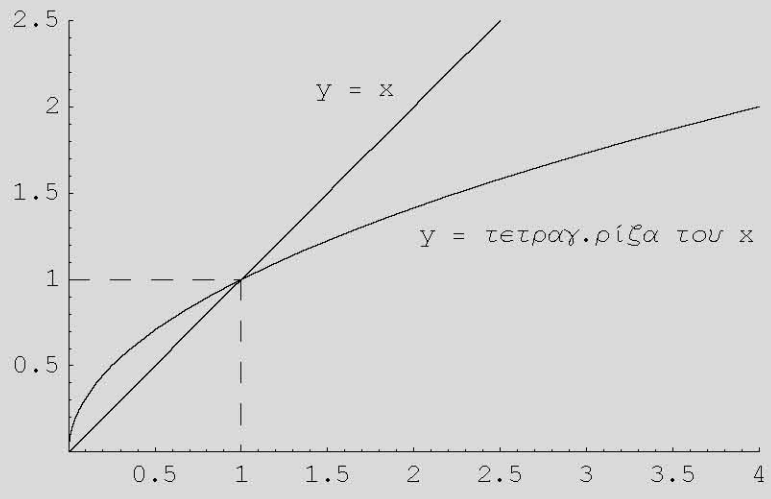
α) Επιγραφές(Labels): *Μπορούμε να προσθέσουμε κάποια επιγραφή με την* **PlotLabel**

```
synhmitono = Plot[Sin[3 x],
   {x, -3 Pi / 2, 5 Pi}, PlotLabel -> "Η συνάρτηση Sin[3 x]"];
```



β) Epilog. *Μπορούμε να χρησιμοποιήσουμε και την* FrameLabel ή *την* AxesLabel(*λίγο παρακάτω θα τις δούμε περισσότερο αναλυτικά*) ή *την* **Epilog** *για τον ίδιο σκοπό*. Η Epilog *είναι ο "επίλογος" στο γράφημά μας δηλ. κάποια στοιχεία* (*π.χ κείμενο,ετικέτα,επιπλέον σημεία*) *που θέλουμε να μπουν αφότου ολοκληρωθεί η γραφική παράσταση. Στο επόμενο παράδειγμα θέσαμε τα κείμενα* "y=x","y=*τετρ.ρίζα του* x", *καθώς και τις διακεκομμένες προς τους αξονες απο το σημείο* (1,1). Για περισσότερες πληροφορίες ανατρέξτε στο Help.

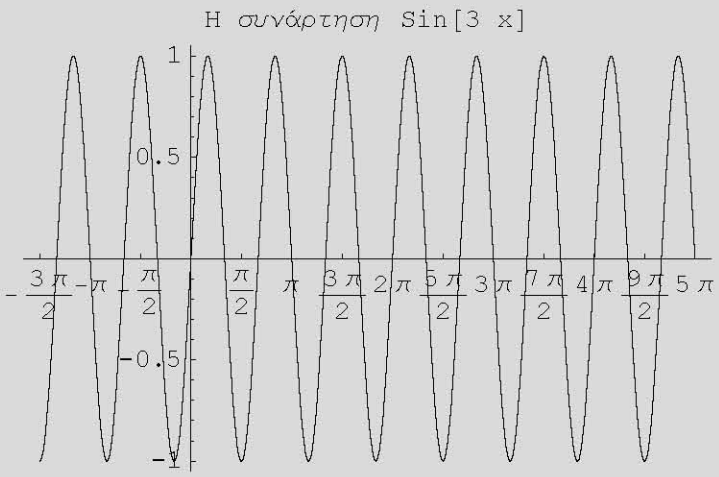
```
Plot[ { x, Sqrt[x] }, {x, 0, 4},
AspectRatio -> Automatic,
PlotRange -> {{0, 4}, {0, 2.5}},
Epilog -> { Text[ "y = x", {2, 2}, {1.3, -1} ],
          Text[
      "y = τετραγ.ρίζα του x", {2, Sqrt[2]}, {-1, 1.4} ],
          Dashing[ {0.03} ], Line[ {{0, 1}, {1, 1}} ],
          Dashing[ {0.03} ], Line[ {{1, 1}, {1, 0}} ] }];
```



γ)Ticks. *Μπορούμε να βάλουμε* **Ticks** (σημαδάκια κάθετα με κάποια τιμή)*στα σημεία* του άξονα *που εμείς θέλουμε*. *Παρακάτω βάζουμε στον άξονα* Ox Ticks *στα σημεία απο* -3 Pi/2 *έως* 5 Pi *με βήμα* Pi/2. Το

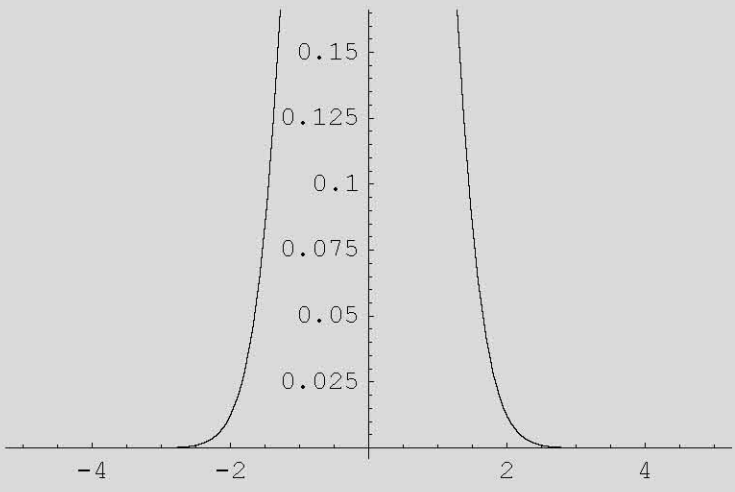
Ticks→{units,Automatic} *σημαίνει ότι στον* Oy *θα ακολουθηθεί η προεπιλεγμένη ακολουθία(η* Automatic *του Mathematica) των* Ticks *ενώ στον* Ox *κατα την δικιά μας* units. Με Ticks->None δεν μπάινουν καθόλου Ticks.

```
units = Range[-3 Pi / 2, 5 Pi, Pi / 2];
Show[synhmitono, Ticks → {units, Automatic}];
```



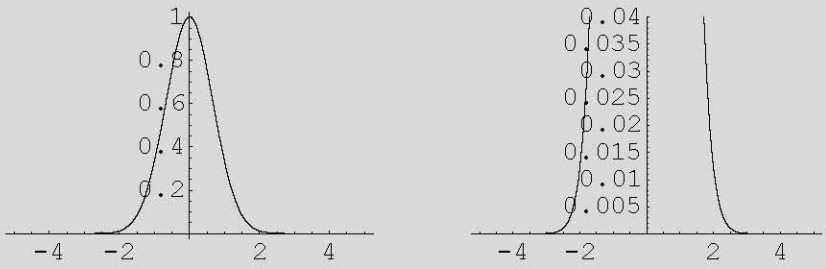
δ)**PlotRange**. Με PlotRange->Automatic καθορίζεται αυτόματα απο το *Mathematica* ποιές τιμές y της καμπύλης θα εμφανιστούν στο τελικό γράφημα. Σε μερικές περιπτώσεις αυτό δεν πετυχαίνει με αποτέλεσμα να χάνουμε σημαντικές πληροφορίες. Π.χ

```
Plot[3^-x^2, {x, -5, 5}];
```



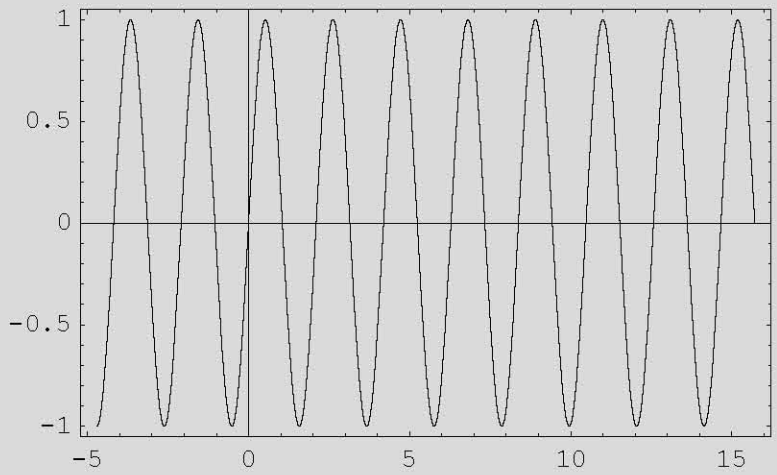
Το σχήμα έχει αποκοπεί στην γειτονιά του 0. Σε τέτοιες περιπτώσεις καλό είναι να βάζουμε **PlotRange->All** έτσι ώστε να εμφανίζονται όλες οι τιμές της καμπύλης. Αν πάλι δεν έχουμε το επιθυμητό αποτέλεσμα ή θέλουμε να εμφανίζονται κάποιες συγκεκριμμένες τιμές του y, θα μπορούσαμε να βάλουμε ένα συγκεκριμμένο διάστημα **PlotRange**->{ymin,ymax}π.χ

```
gr1 = Plot[3^-x^2, {x, -5, 5},
   PlotRange → All, DisplayFunction → Identity];
gr2 = Plot[3^-x^2, {x, -5, 5}, DisplayFunction → Identity,
   PlotRange → {0, 0.04}];
Show[GraphicsArray[{gr1, gr2}]];
```



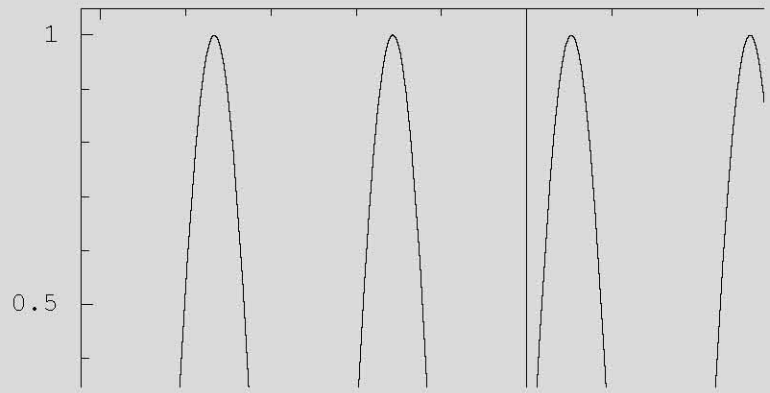
ε).Με **PlotRegion** *μπορούμε να θέσουμε περιθώριο γύρω απο ένα γράφημα* έτσι ώστε να μετατοπιστεί η γραφική παράσταση σε κάποια θέση του πλαισίου. Αν πειραματιστούμε με ***κατάλληλες αρνητικές τιμές στο*** PlotRegion τότε ***μπορούμε να ζουμάρουμε*** σε κάποιο σημείο της γρ. παράστασης *π.χ* συγκρίνετε τα παρακάτω

```
Plot[Sin[3 x], {x, -3 Pi / 2, 5 Pi}, Frame → True]
```



- Graphics -

```
Plot[Sin[3 x], {x, -3 Pi / 2, 5 Pi},
 Frame → True, PlotRegion → {{-0.5, 3}, {-1.7, 0.8}}]
```

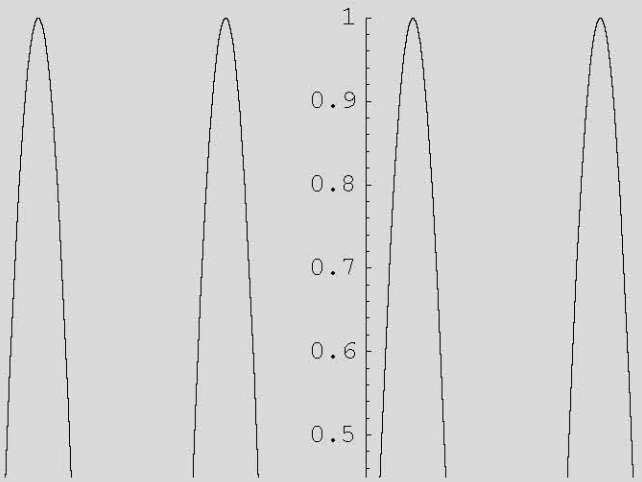


- Graphics -

Άσκηση: Αλλάξτε *τις τιμές του* PlotRegion (*προτιμήσετε τιμές μεταξύ* 0 *και* 1 *π.χ* PlotRegion→{{0.5,1},{0,0.5}})*Δείτε το αποτέλεσμα*. *Αφαιρέστε το* όλο *το* PlotRegion *και δείτε το αποτέλεσμα*! Το {0.5,1}*στο* PlotRegion→{{0.5,1},{0,0.5}}*σημαίνει ότι το γράφημα θα καταλάβει το διάστημα* 50% *έως* 100% *στη οριζόντια κατεύθυνση του πλαισίου*(*δηλ. το ήμισυ δεξιό*) *ενώ το* {0,0.5} *σημαίνει ότι το γράφημα θα καταλάβει το διάστημα* 0% *έως* 50% *στη κάθετη κατεύθυνση του πλαισίου*(*δηλ. το κάτω ήμισυ*). Γενικά χρειάζεται αρκετός πειραματισμός με το PlotRegion.

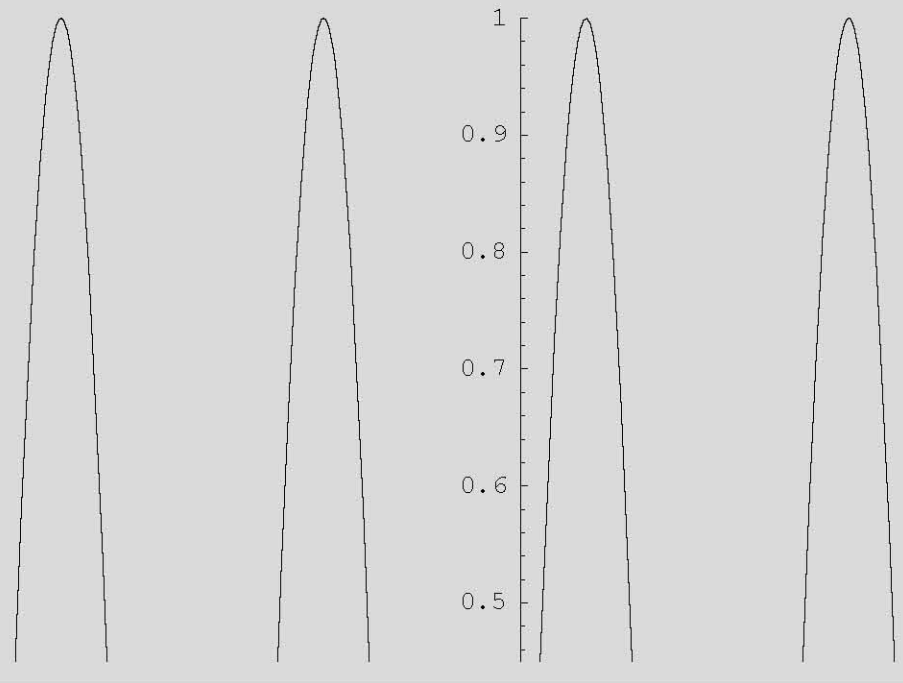
Σχόλιο: *Μπορούμε* επίσης *και να ζουμάρουμε θέτοντας πιο μικρό* PlotRange. *Το* PlotRange *έχει σχέση να κάνει με τα σημεία* (x,y) *της καμπύλης, που επιθυμούμε να εμφανίζονται στο γράφημα* που πρόκειται να κατασκευαστεί. Το PlotRegion *έχει σχέση με την τελική θέση και εμφάνιση του γραφήματος* που έχει ήδη κατασκευαστεί *μέσα στο πλαίσιο*(*που εμφανίζεται στην οθόνη μας όταν κάνουμε κλίκ στο γράφημα*). Δείτε επίσης τα παραδείγματα παρακάτω με την χρήση της PlotRange.

```
Plot[Sin[3 x], {x, -3 Pi / 2, 5 Pi},
  PlotRange → {{-4.3, 4}, {0.45, 1}}, Axes -> {False, True}];
```



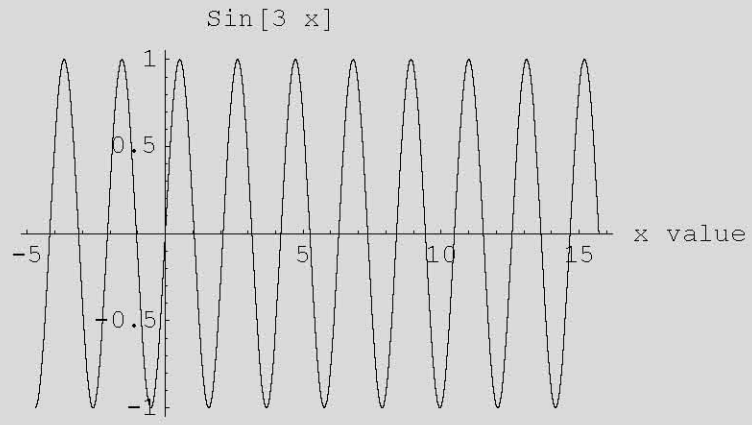
Για να ζουμάρουμε ακόμα περισσότερο μπορούμε να τραβήξουμε προς τα έξω τις λαβές ή *να θέσουμε μια μεγάλη τιμή στο* **ImageSize** *π.χ* ImageSize→400. Μπορείτε να πειραματιστείτε με διαφορετικές τιμές του **ImageSize.**

```
Plot[Sin[3 x], {x, -3 Pi / 2, 5 Pi}, PlotRange → {{-4.3, 4}, {0.45, 1}},
  Axes -> {False, True}, ImageSize → 400];
```



στ) *Άξονες*. Με Axes->{False,True} *αναγκάσαμε να μην σχεδιαστεί ο άξονας* Ox. *Μπορούμε να βάλουμε ετικέτες στους άξονες*

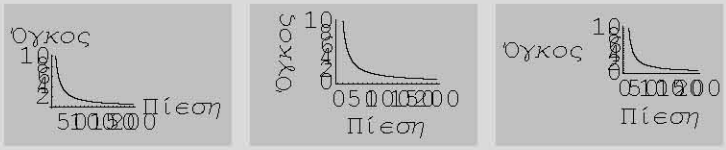
```
Plot[Sin[3 x], {x, -3 Pi / 2, 5 Pi},
 AxesLabel → {"x value", "              Sin[3 x]"}]
```




```
- Graphics -
```

ζ) **Frames** *και* **GridLines**. *Μπορούμε να βάλουμε πλαίσια γύρω απο την καμπύλη.* Η *βασική εντολή είναι* Frame->True. Με FrameStyle, FrameLabel, FrameTicks RotatedLabel *και* GridLines *μπορούμε να προσθέσουμε κάποια ειδιαίτερα χαρακτηριστικά στο τελικό πλαίσιο.* Π.χ

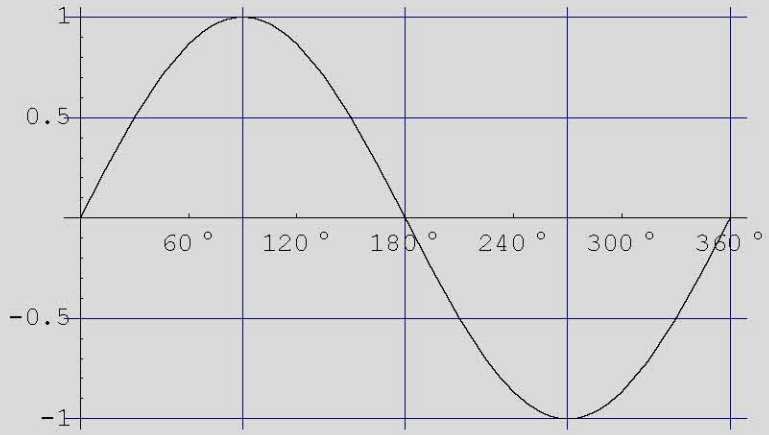
```
gr1 = Plot[ 100 / P, {P, 10, 200}, AxesLabel -> {"Πίεση", "Όγκος"},
   Background -> GrayLevel[3 / 4], DisplayFunction -> Identity ];
gr2 = Show[ gr1, Frame -> {True, True, False, False},
   FrameLabel -> {"Πίεση", "Όγκος"} ];
gr3 = Show[ gr2, RotateLabel -> False];
Show[GraphicsArray[{ gr1, gr2, gr3 }]];
```




```
Null[4]
```

(Τραβήξτε *προς τα έξω από τις λαβές αν χρειαστεί να μεγενθύνετε*) Με Frame->{True,True,False,False} *βάλαμε πλαίσιο μόνο αριστερά και κάτω.* Με RotateLabel->False *στρύψαμε την ετικέτα οριζόντια με αποτέλεσμα να έχουμε χειρότερη εμφάνιση στο γράφημα* gr3 *απ' ότι στο* gr2 *στο οποίο η προεπιλεγμένη τιμή του* RotateLabel *είναι* RotateLabel->True. Με GridLines *μπορούμε να προσθέσουμε πλέγμα στα σημεία που θέλουμε (με προεπιλεγμένη τιμή* GridLines→Automatic, *το πλέγμα τοποθετείται αυτομάτα απο το Mathematica με κάποιο εσωτερικό αλγόριθμο) π.χ*

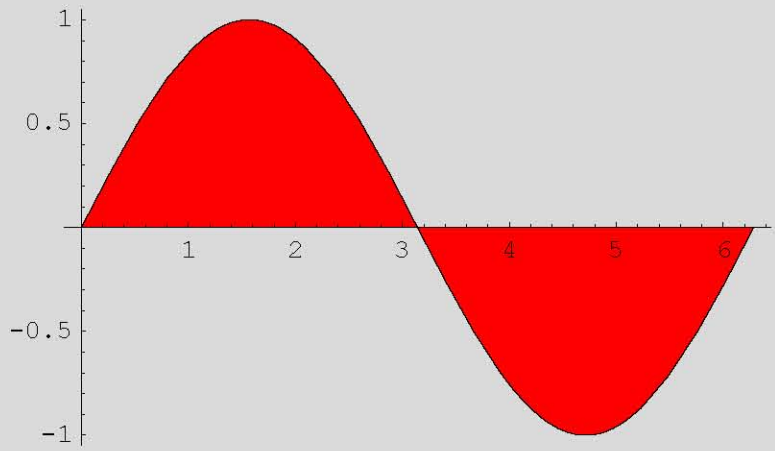
```
units = Range[ 0 Degree, 360 Degree,  60 Degree];
Plot[Sin[x], {x, 0, 2 Pi}, Ticks → {units, Automatic},
  GridLines → {{Pi/2, Pi, 3 Pi/2, 2 Pi, 5 Pi/2}, Automatic}];
```



9.1.4 Άλλες δυνατότητες των διδιάστατων γραφικών

1) Μπορούμε με **FilledPlot** του πακέτου Graphics`FilledPlot` να γεμίσουμε με ένα χρώμα το χώρο που βρίσκεται μεταξύ δυο ή παραπάνω καμπυλών καμπυλών ή μεταξύ μιας καμπύλης και ένα άξονα. Έτσι για παράδειγμα η FilledPlot[f[x],g[x],{x,xin,xmax}] γεμίζει με χρώμα τον χώρο μεταξύ της f[x], και g[x] για {x,xin,xmax}, ενώ με σκετο FilledPlot[f[x],{x,xin,xmax}] ή με FilledPlot[f[x],0,{x,xin,xmax}] γεμίζεται ο χώρος μεταξύ της f[x] και του Ox. Π.χ

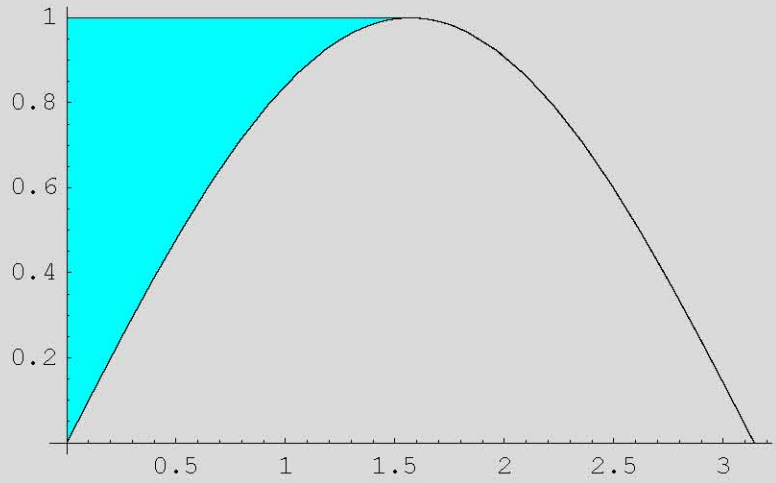
```
<< Graphics`FilledPlot`
FilledPlot[Sin[x], {x, 0, 2 Pi}, Fills → {RGBColor[1, 0, 0]}]
```



- Graphics -

Στο επόμενο παράδειγμα βάλαμε χρώμα μεταξύ της f[x]=Sin[x] , με 0<= x<=Pi , *και της ευθείας* g[x]=1*αλλά μόνο στο διάστημα* [0.Pi/2]. *Αναγκαστήκαμε να ορίσουμε την* g[x]=1 *για το διάστημα* [0,Pi/2] *και* f[x] *στο υπόλοιπο διάστημα έτσι ώστε να βγεί το επιθυμητό αποτέλεσμα*

```
FilledPlot[ { Sin[x], If[x<=Pi/2&&x>=0,1,Sin[x]]},
        {x, 0, Pi},  Curves -> Front];
```



2) Με **PlotVectorField** *του πακέτου* Graphics`Plotfield` *παίρνουμε την γραφικη παράσταση ενός διανυσματικού πεδίου ενώ με* CartesianMap *και* PolarMap *του πακέτου* Graphics`ComplexMap` , *σχεδιάζουμε μιγαδικές συναρτήσεις*.

3)Με **PlotGradientField**[f[x,y],{x,xmin,xmax},{y,ymin,ymax}] *του πακέτου* Graphics`PlotField` *παίρνουμε την κατεύθυνση κατα την οποία η συνάρτηση μας* f[x,y] *αυξάνεται με τον ταχύτερο ρυθμό απο το σημείο* {x,y}.

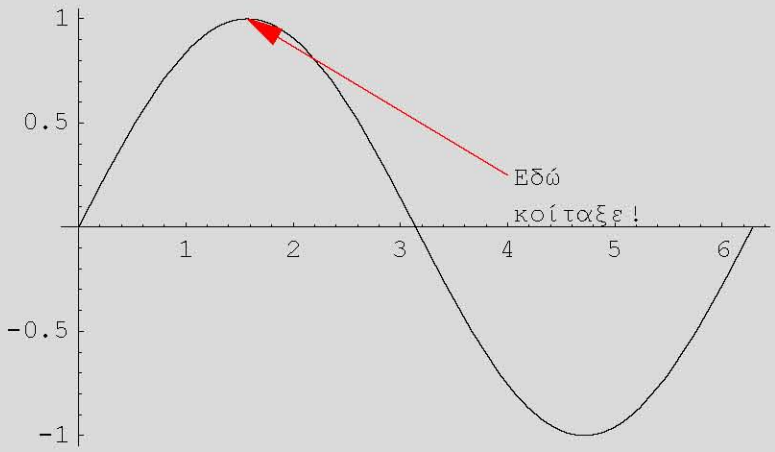
4)*Δεν πρέπει να ξεχάσουμε και την* **Animate** *του πακέτου* Graphics`Animation` *που μας δίνει την δυνατότητα της κινούμενης εικόνας από λίστα σημείων ή λίστα συναρτήσεων. Παρακάτω βλέπουμε την κατασκευή μιας ακολουθίας όρων χρησιμοποιώντας την* ListPlot.

```
<< Graphics`Animation`
<< Graphics`Graphics`
Animate[ListPlot[Table[1/n + Sin[n π/2] + Cos[n π/2], {n, m}],
  PlotRange → {{0, 20}, {-1.5, 1.5}}, AxesOrigin → {0, 0},
  PlotStyle → {PointSize[0.02], Hue[.6]}], {m, 1, 20, 1}]
```

Πατώντας διπλό *κλίκ σε ένα απο τα παραπάνω γραφήματα έχουμε την ζητούμενη* Animation. *Προσέξτε καλέσαμε δύο διαφορετικά πακέτα για δύο διαφορετικούς σκοπούς. Καλέσαμε το* Graphics`Graphics` *για την* ListPlot *και το* <<Graphics`Animation` *για το* Animate. *Απο την κατασκευή βλέπουμε ότι η ακολουθία μας* $\frac{1}{n} + \text{Sin}[\frac{n\pi}{2}] + \text{Cos}[\frac{n\pi}{2}]$ *έχει δυο συγκλίνουσες υπακολουθίες*.

5)Τελος με **Arrow[{x0,y0},{x1,y1}]** μπορούμε να σχεδιάσουμε ένα βέλος απο το σημείο A_0 (x0, y0) στο A_1 (x1, y1). Είναι χρήσιμο όταν θέλουμε να σχεδιάσουμε βέλη ή προσανατολισμένους άξονες ή όταν θέλουμε να τονίσουμε ένα τμήμα του γραφήματος. Η Arrow απαιτει το πακέτο Graphics`Arrow` π.χ.

```
<< Graphics`Arrow`
Plot[Sin[x], {x, 0, 2 Pi},
  Epilog → {{RGBColor[1, 0, 0], Arrow[{4, .25}, {Pi / 2, 1}]},
    Text["Εδώ \nκοίταξε!", {4, .15}, {-1, 0}]}];
```



Με το \n μπορούμε να αλλάξουμε γραμμή στο κείμενό μας! Για την Epilog είχαμε μιλήσει στην προηγούμενη ενότητα.