ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΑΝΟΙΚΤΑ ΑΚΑΔΗΜΑΪΚΑ ΜΑΘΗΜΑΤΑ

Μαθηματικά και Φυσική με Υπολογιστές

Σημειώσεις Μαθήματος

Διδάσκων: Καθηγητής Ι. Ρίζος

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Περιγραφή Μαθήματος

- Εισαγωγή: Ιστορικά Στοιχεία, συμβολικοί υπολογισμοί και σχετικό λογισμικό. Ανοικτό λογισμικό- Το λογισμικό SAGE. Βασικές Έννοιες: Απλοί αλγεβρικοί και αριθμητικοί υπολογισμοί, συναρτήσεις, παράγωγοι, ολοκληρώματα, ρίζες εξισώσεων. Γραφικές αναπαραστάσεις: Γραφικές αναπαραστάσεις συναρτήσεων στις δύο και τρεις διαστάσεις, γραφικές αναπαραστάσεις δεδομένων, γραφική αναπαράσταση διανυσματικών πεδίων, κινούμενα γραφικά (animation). Σύνθετα προβλήματα: Γραμμική Άλγεβρα, Ιδιοτιμές, Ιδιοσυναρτήσεις, Σειρές, Διαφορικές εξισώσεις, Αριθμητικοί υπολογισμοί. Εφαρμογές στα Μαθηματικά και στη Φυσική.

Σκοπός Μαθήματος

- Σκοπός του μαθήματος είναι η εξοικείωση των φοιτητών με την χρήση των σύγχρονων εργαλείων για την εκτέλεση υπολογισμών με την βοήθεια των υπολογιστών. Οι υπολογισμοί αυτοί περιλαμβάνουν αριθμητικές και αναλυτικές πράξεις, γραφικές αναπαραστάσεις επίλυση αλγεβρικών και διαφορικών εξισώσεων. Έμφαση θα δοθεί στις μεθόδους κατάστρωσης και επίλυσης προβλημάτων μαθηματικών ή/και φυσικής με την βοήθεια του υπολογιστή με τη χρήση ανοικτού λογισμικού.

Μαθηματικά και Φυσική με Υπολογιστές

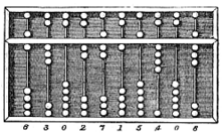
Ρίζος Ιωάννης
Καθηγητής Τμήματος Φυσικής
Πανεπιστημίου Ιωαννίνων
irizos@uoi.gr

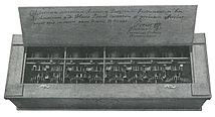
Ιωάννινα 11-5-2015

Εισαγωγή

Αναλυτικές πράξεις με υπολογιστή

- Η κύρια χρήση των υπολογιστικών συστημάτων αφορούσε την εκτέλεση αριθμητικών υπολογισμών
- Η ανάγκη για την εκτέλεση αναλυτικών υπολογισμών με τη βοήθεια υπολογιστή προήλθε από την έρευνα στην επιστήμη της φυσικής.



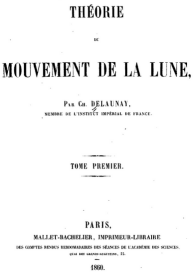


Αναλυτικές πράξεις με υπολογιστή

Στα μέσα του 19ου αιώνα ο Γάλλος αστρονόμος Charles-Eugène Delaunay ξεκίνησε ένα υπολογισμό για τη θέση της σελήνης τον οποίο δημοσίευσε με τον τίτλο «La Théorie du mouvement de la lune» σε δύο τόμους. Χρειάστηκε 20 χρόνια για να ολοκληρώσει και να ελέγξει τον υπολογισμό.

Το 1960 ο υπολογισμός ελέγχθηκε με τη βοήθεια ενός εξειδικευμένου προγράμματος (CAMAL) το οποίο ολοκλήρωσε τον υπολογισμό σε 20 ώρες και βρήκε ένα λάθος στον υπολογισμό του Delaunay.

Charles-Eugène Delaunay
(1816 – 1872)



THÉORIE

DU

MOUVEMENT DE LA LUNE,

PAR Ch. DELAUNAY,

MEMBRE DE L'INSTITUT IMPÉRIAL DE FRANCE.

TOME PREMIER.

PARIS,

MALLET-BACHELIER, IMPRIMEUR-LIBRAIRE

1860.

Ανάπτυξη λογισμικού αναλυτικών υπολογισμών

Η αρχή της ανάπτυξης λογισμικού αναλυτικών υπολογισμών (computer algebra) τοποθετείται γύρω στο 1960 και προέρχεται από δύο διαφορετικές κατευθύνσεις. Από τις ανάγκες των φυσικών, ιδιαίτερα στην περιοχή των στοιχειωδών σωματιδίων, και από την έρευνα στο πεδίο της τεχνητής νοημοσύνης.

Ένα από τα πρώτα λογισμικά είναι το Schoonschip το οποίο αναπτύχθηκε το 1963 από τον Martin Veltman (βραβείο Nobel Φυσικής 1999). Επίσης το MATHLAB (1964) αναπτύχθηκε σε LISP από τον Carl Engelman (MITRE). Ο Carl Engelman αργότερα αποχώρησε και άρχισε την ανάπτυξη του Macsyma στο MIT (1978). Επίσης πρέπει να αναφέρουμε τα Reduce(1968), muMATH(1980), Derive(1988).

Λογισμικό με δυνατότητες αναλυτικών υπολογισμών σήμερα

Εμπορικό Λογισμικό

- Maple, πρώτη έκδοση το 1984, πρόσφατη έκδοση 2014, Symbolic Computation Group, University of Waterloo, http://www.maplesoft.com/
- Mathematica, πρώτη έκδοση 1988, πρόσφατη έκδοση 2014, Wolfram Research, http://www.wolfram.com/mathematica/
- Magma, πρώτη έκδοση 1993 , πρόσφατη έκδοση 2014, University of Sydney, http://magma.maths.usyd.edu.au/magma/

Ελεύθερο Λογισμικό

- Axiom, πρώτη έκδοση 2002, πρόσφατη έκδοση 2014, Tim Daly, http://www.axiom-developer.org/
- Macsyma/Maxima, πρώτη έκδοση 1978, πρόσφατη έκδοση 2014, MIT Project MAC, διαθέσιμο και σε συσκευές Android, http://maxima.sourceforge.net/
- SAGE, πρώτη έκδοση το 2005, πρόσφατη έκδοση 2015 , William A. Stein, βασίζεται σε μια συλλογή λογισμικών διασυνδεδεμένων με τη γλώσσα Python, http://www.sagemath.org/
- Mathics, πρώτη έκδοση 2011, πρόσφατη έκδοση 2013, Jan Pöschko, διαθέτει γλώσσα προγραμματισμού παραπλήσια του Mathematica, http://www.mathics.org/
- SymPy, πρώτη έκδοση 2007, πρόσφατη έκδοση 2015, Βιβλιοθήκη της γλώσσας python με δυνατότητες συμβολικών υπολογισμών, http://www.sympy.org

Για έναν αναλυτικό κατάλογο δείτε: **http://en.wikipedia.org/wiki/List_of_computer_algebra_systems**

Το Λογισμικό SAGE

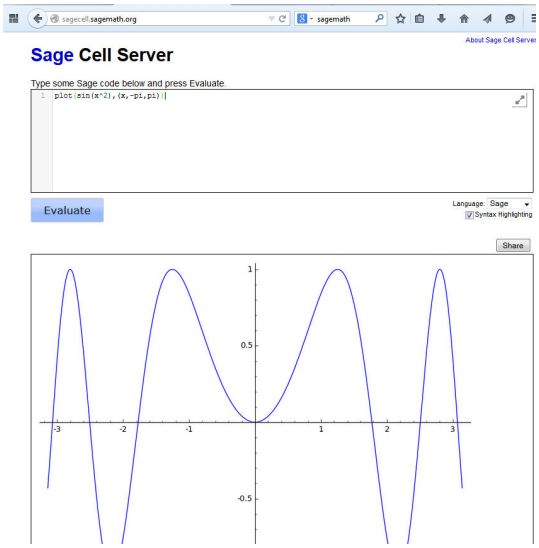
Το λογισμικό SAGE

Το Λογισμικό SAGE (System for Algebra and Geometry Experimentation). Η πρώτη δημόσια έκδοση παρουσιάστηκε το 2005 ως ελεύθερο λογισμικό (GNU General Public License) με σκοπό σύμφωνα με τον συγγραφέα τη δημιουργία ενός «..open source alternative to Magma, Maple, Mathematica, and MATLAB». Ο δημιουργός είναι ο William Stein, καθηγητής μαθηματικών στο University of Washington.

Είναι λογισμικό γενικής χρήσης, βασισμένο στη γλώσσα Python, με δυνατότητες αναλυτικών και αριθμητικών υπολογισμών καθώς και γραφικών.

Περισσότερα για το SAGE : http://www.sagemath.org/

Χρήση του SAGE online



Μπορούμε να χρησιμοποιήσουμε το SAGE online (χωρίς την εγκατάσταση λογισμικού στον υπολογιστή μας) μέσω της ιστοσελίδας: http://sagecell.sagemath.org/

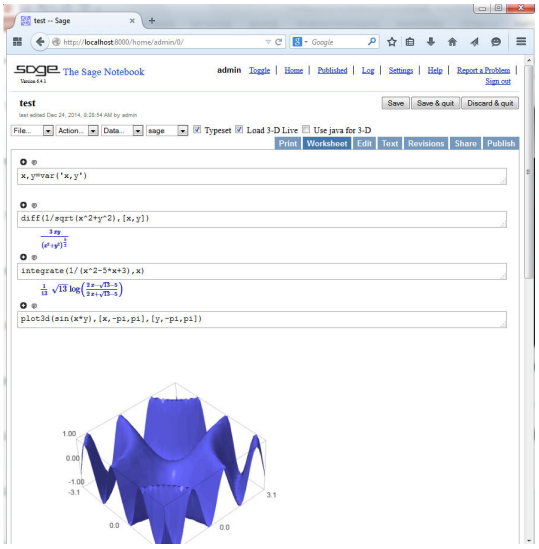
Η εγκατάσταση του SAGE

Ο ευκολότερος τρόπος να εγκαταστήσουμε το SAGE συνίσταται στην εγκατάσταση της αντίστοιχης εικονικής μηχανής (virtual machine).

Σε περιβάλλον Windows η εγκατάσταση μπορεί να γίνει ακολουθώντας τα παρακάτω βήματα:

1. Εγκατάσταση λογισμικού VirtualBox (ελεύθερο λογισμικό δημιουργίας εικονικών μηχανών) βλ https://www.virtualbox.org
2. Κατέβασμα εικονικής μηχανής SAGE από : http://www.sagemath.org
3. Εγκατάσταση και εκκίνηση εικονικής μηχανής SAGE με τη βοήθεια του VirtualBox
4. Άνοιγμα σε ένα πρόγραμμα πλοήγησης (FireFox, Chrome) της ιστοσελίδας: http://localhost:8000
5. Σύνδεση μέσω του λογαριασμού : Username: *admin*, Password: *sage*

Το περιβάλλον εργασίας του SAGE



Η επικοινωνία με το SAGE υλοποιείται κατά κύριο λόγο μέσω φύλλων εργασίας (notebooks) τα οποία είναι προσβάσιμα ως ιστοσελίδες.
Τα φύλλα εργασίας βρίσκονται και αποθηκεύονται στην εικονική μηχανή. Έχουμε όμως τη δυνατότητα να τα κατεβάσουμε στον υπολογιστή μας.

```
3+8*7-9*(3+2)
```

evaluate

Για να εκτελεστεί μια εντολή είτε κάνουμε κλικ στο «evaluate» είτε πατάμε SHIFT-ENTER

Το ενσωματωμένο σύστημα βοήθειας

Το όνομα μια εντολής ακολουθούμενο από (αγγλικό) ερωτηματικό δίνει συνοπτική περιγραφή και αρκετές φορές παραδείγματα χρήσης της εντολής.

```
ceil?
```

File: /home/sage/sage-6.4.1/local/lib/python2.7/site-packages/sage/functions/other.py

Type: <class 'sage.functions.other.Function_ceil'>

Definition: ceil(x, maximum_bits=20000)

Docstring:

The ceiling function.

The ceiling of x is computed in the following manner.

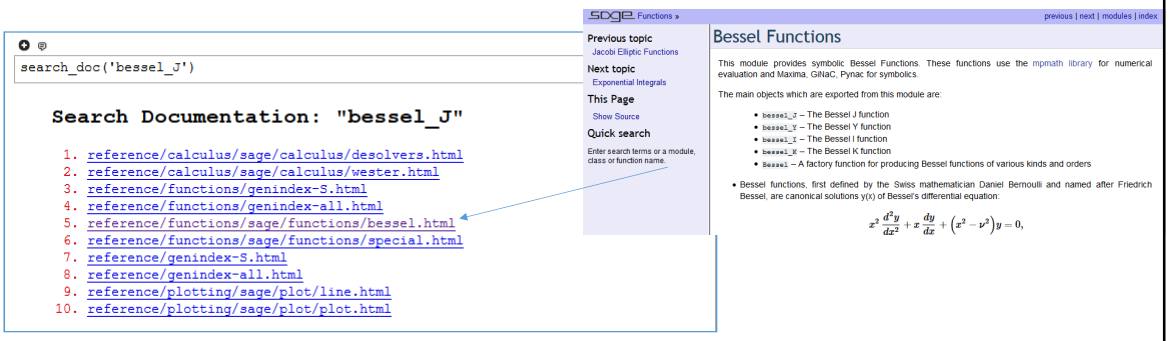
1. The `x.ceil()` method is called and returned if it is there. If it is not, then Sage checks if x is one of Python's native numeric data types. If so, then it calls and returns `Integer(int(math.ceil(x)))`.
2. Sage tries to convert x into a `RealIntervalField` with 53 bits of precision. Next, the ceilings of the endpoints are computed. If they are the same, then that value is returned. Otherwise, the precision of the `RealIntervalField` is increased until they do match up or it reaches `maximum_bits` of precision.
3. If none of the above work, Sage returns a `Expression` object.

EXAMPLES:

```
sage: a = ceil(2/5 + x)
sage: a
ceil(x + 2/5)
sage: a(x=4)
5
sage: a(x=4.0)
5
```

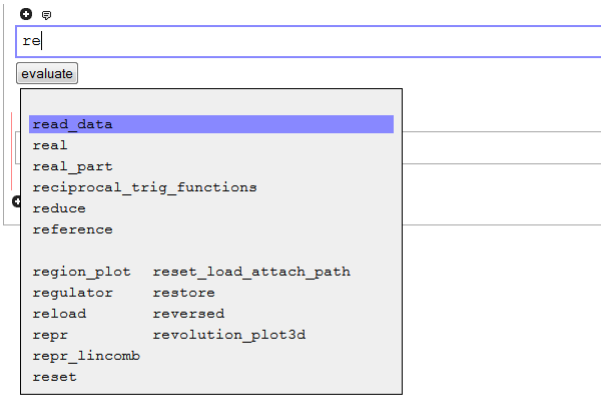
Αναζήτηση πληροφοριών

Η συνάρτηση search_doc() βοηθάει στην αναζήτηση τεκμηρίωσης για τις συναρτήσεις του SAGE. Η συνάρτηση search_def() επιστρέφει τους σχετικούς ορισμούς (σε PYTHON) .



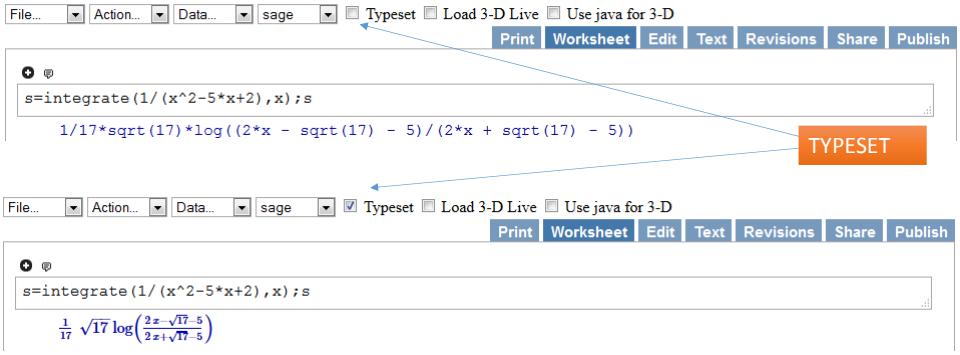
Αυτόματη συμπλήρωση εντολών

Γράφοντας τους πρώτους χαρακτήρες μιας «εντολής» και πιέζοντας το πλήκτρο **TAB** παίρνουμε ένα μενού με όλες τις «εντολές» που αρχίζουν από αυτούς τους χαρακτήρες.



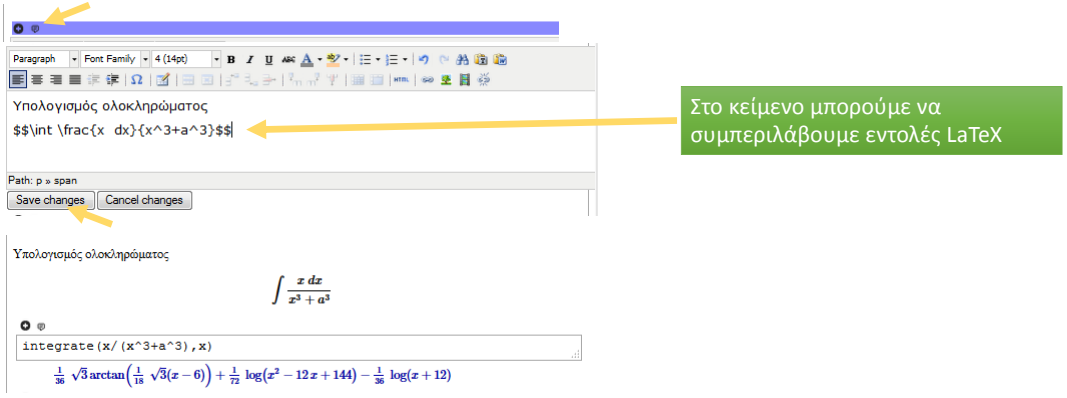
Μορφοποίηση αποτελεσμάτων

Ανάλογα με την επιλογή της δυνατότητας Typeset έχουμε την εμφάνιση των αποτελεσμάτων



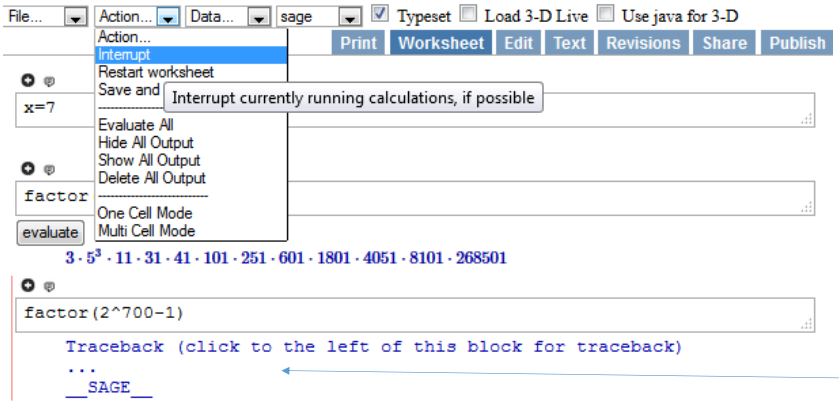
Επεξεργασία κειμένου - LaTeX

Μπορούμε να προσθέσουμε σχόλια στο φύλλο εργασίας χρησιμοποιώντας τον επεξεργαστή κειμένου



Διακοπή υπολογισμού

Αρκετές φορές είναι απαραίτητη η διακοπή ενός υπολογισμού ο οποίος παίρνει πολύ χρόνο. Σε αυτήν την περίπτωση χρησιμοποιούμε το **Interrupt** από το μενού **Action**



Όταν εργαζόμαστε σε κοινόχρηστο server (όπως αυτός που χρησιμοποιούμε στο μάθημα) είναι πολύ σημαντικό να διακόπτουμε τους μακροσκελείς υπολογισμούς για να μην προκαλούμε καθυστερήσεις στους άλλους χρήστες.

ΔΙΑΚΟΠΗ ΥΠΟΛΟΓΙΣΜΟΥ

Η γλώσσα προγραμματισμού του SAGE: PYTHON

Όλα τα σύγχρονα πακέτα αναλυτικών υπολογισμών (CAS) διαθέτουν και μια γλώσσα προγραμματισμού. Το SAGE αντί να εισάγει μια νέα δική του γλώσσα προγραμματισμού χρησιμοποιεί την δημοφιλή γλώσσα PYTHON.

Η PYTHON είναι μια μοντέρνα γλώσσα προγραμματισμού (δημιουργήθηκε το 1990) η οποία είναι διαθέσιμη ως λογισμικό ανοικτού κώδικα για μια πληθώρα λειτουργικών συστημάτων. Χρησιμοποιείται σήμερα ευρύτατα για επιστημονικούς υπολογισμούς.

Στα πλεονεκτήματά της συγκαταλέγονται : η καθαρότητα του κώδικα, η ευκολία εκμάθησης, η πληθώρα των διαθέσιμων βιβλιοθηκών. Στα μειονεκτήματα αναφέρεται το γεγονός ότι δεν διαθέτει μεταφραστή (compiler) αλλά μεταγλωττιστή (interpreter).

$$\{x | 1 \leq x < 200 \text{ and } x = 0 \mod 17\}$$

Μαθηματικός συμβολισμός

```
[x for x in range(1,200) if x % 17 == 0 ]
[17, 34, 51, 68, 85, 102, 119, 136, 153, 170, 187]
```

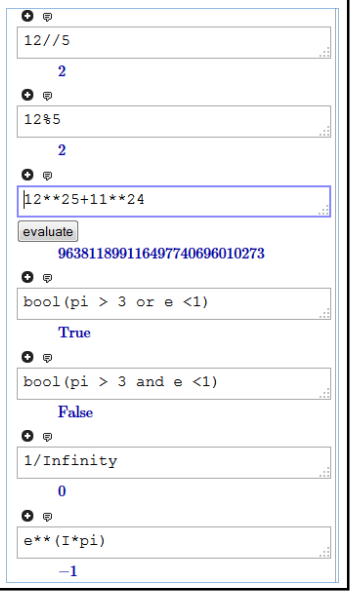
Συμβολισμός PYTHON

Βασικοί υπολογισμοί

Βασικές πράξεις και σταθερές

Πράξη	Συμβολισμός SAGE
Τέσσερεις πράξεις	`x+y, x-y, x*y, x/y`
Ύψωση σε δύναμη	`x^y` ή `x**y` (συμβατό με PYTHON)
Ακέραια διαίρεση	`a // b`
Υπόλοιπο ακέραιας διαίρεσης	`a % b`
OR	`or`
AND	`and`
NOT	`not`
TRUE, FALSE	`True, False`

Μαθηματικός συμβολισμός	Συμβολισμός SAGE
π	`pi`
∞(άπειρο)	`Infinity` ή `oo`
e (βάση φυσικών λογαρίθμων)	`e`
i (φανταστική μονάδα)	`I` ή `i`



Συνήθεις μαθηματικές συναρτήσεις

Συνάρτηση	Συμβολισμός SAGE
$\|x\|$ (απόλυτο)	`abs(x)`
n! (παραγοντικό)	`factorial(n)`
[n] (ακέραιο μέρος)	`floor(n)`
$\binom{n}{k}$	`binomial(n,k)`
e^x	`exp(x)`
$\sqrt{x}$	`sqrt(x)`
$\log_a x$	`log(x,a)`
$\ln x$	`log(x)` ή `ln(x)`
$\sin x, \cos x, \tan x, \cot x$	`sin(x),cos(x),tan(x),cot(x)`
$\sin^{-1} x, \cos^{-1} x, \tan^{-1} x, \cot^{-1} x$	`arcsin(x),arccos(x),arctan(x),arccot(x)`
$\sinh x, \cosh x, \tanh x, \coth x$	`sinh(x),cosh(x),tanh(x),coth(x)`
$\sinh^{-1} x, \cosh^{-1} x, \tanh^{-1} x, \coth^{-1} x$	`arcsinh(x),arccosh(x),arctanh(x),arccoth(x)`

Συνήθεις μαθηματικές συναρτήσεις (συνέχεια)

Συνάρτηση	Συμβολισμός SAGE
z^*	`conjugate(z)`
$\|z\|$ (μέτρο μιγαδικού)	`abs(z)`
Re[z] (πραγματικό μέρος)	`real(z)`
Im[z] (φανταστικό μέρος)	`imag(z)`
Έλεγχος συμβολικών ισοτήτων-ανισοτήτων	`bool(....)`
Μετατροπή σε ακέραιο	`int(...)`
Μετατροπή σε πραγματικό	`float(...)`
Μετατροπή σε χαρακτήρες	`str(...)`

Προσοχή στον πολλαπλασιασμό

Το σύμβολο * για τον πολλαπλασιασμό είναι απαραίτητο και δεν μπορεί να παραληφθεί.

```
2x
```

```
Traceback (click to the left of this block for traceback)
...
SyntaxError: invalid syntax
```

```
2*x
```

$2x$

Τύποι μεταβλητών - σταθερών

Η γλώσσα PYTHON είναι αντικειμενοστραφής γλώσσα προγραμματισμού και οι οντότητες που χρησιμοποιούμε σε ένα πρόγραμμα για να αναπαραστήσουμε συγκεκριμένες ποσότητες ή αφηρημένες έννοιες, ονομάζονται αντικείμενα (objects). Τα ομοειδή αντικείμενα θεωρούνται ως στιγμιότυπα (instances) μιας κλάσης (class).

Για παράδειγμα οι ακέραιοι αριθμοί 122 και 97 μπορούν να θεωρηθούν ως στιγμιότυπα της κλάσης των ακεραίων. Η συνάρτηση type μας πληροφορεί για την κλάση της κάθε ποσότητας.

```
type(5)
```

```
<type 'sage.rings.integer.Integer'>
```

```
type(2/3)
```

```
<type 'sage.rings.rational.Rational'>
```

```
type(1.3222)
```

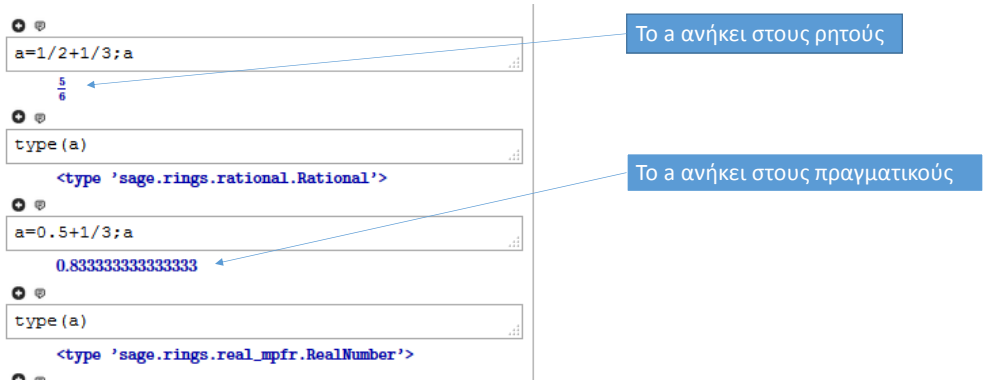
```
<type 'sage.rings.real_mpfr.RealLiteral'>
```

```
type(5/1)
```

```
<type 'sage.rings.rational.Rational'>
```

Τύποι μεταβλητών - σταθερών

Ο ορισμός της κλάσης/τύπου της κάθε ποσότητας είναι δυναμικός στην PYTHON (σε αντίθεση με γλώσσες όπως C/FORTRAN όπου είναι στατικός). Αυτό έχει συνέπειες στα αποτελέσματα των υπολογισμών



Δακτύλιοι και Πεδία

Όταν εργαζόμαστε με το SAGE είναι συνήθως χρήσιμο και μερικές φορές απαραίτητο να ορίσουμε το δακτύλιο στον οποίο ανήκουν τα μαθηματικά αντικείμενα με τα οποία εργαζόμαστε.

Ο **Δακτύλιος (ring)** είναι ένα σύνολο εφοδιασμένο με τις πράξεις της «πρόσθεσης» και του «πολλαπλασιασμού».

Τυπικό παράδειγμα δακτυλίου είναι το σύνολο των ακεραίων

$\mathbb{Z} = \{0, \pm 1, \pm 2, \cdots\}$ με τις συνήθεις πράξεις της πρόσθεσης και του πολλαπλασιασμού.

Το **Πεδίο (field)** είναι ένας μεταθετικός δακτύλιος ο οποίος διαθέτει το αντίστροφο, ως προς την πράξη του πολλαπλασιασμού, για όλα τα μη μηδενικά στοιχεία.

Το σύνολο των Ρητών είναι πεδίο (και δακτύλιος), το σύνολο των ακεραίων είναι δακτύλιος (δεν διαθέτει το αντίστροφο στοιχείων πχ του 2 που είναι το ½)

Βασικοί δακτύλιοι (συνέχεια)

Δακτύλιος	Συμβολισμός SAGE
Ακέραιοι αριθμοί	ZZ
Ρητοί αριθμοί	QQ
Πραγματικοί αριθμοί	RR
Μιγαδικοί αριθμοί	CC

```
w=Integers(3)(2);w
```
2
```
w^2
```
1

```
parent(1.33356)
```
evaluate
R
```
parent(2/3)
```
Q
```
parent(pi)
```
SR
```
1.14 in QQ
```
True
```
2/3 in ZZ
```
False

Επιπροσθέτως ορίζονται και οι παρακάτω ειδικοί δακτύλιοι

Δακτύλιος	Συμβολισμός SAGE
Συμβολικές μεταβλητές	SR
$\mathbb{Z}_n$ = Ακέραιοι αριθμοί mod n	`Integer(n)`
Πολυώνυμα με ρητούς συντελεστές	`PolynomialRing(QQ)`

Η συνάρτηση parent πχ `parent(x)` μας ενημερώνει για το δακτύλιο στον οποίο ανήκει το x

Μεταβλητές με δείκτες στην PYTHON - Λίστες

Οι μεταβλητές με δείκτες οι οποίες περιγράφουν συλλογές αντικειμένων στην PYTHON περιλαμβάνουν τρία είδη τα οποία ονομάζονται Λίστες, Πλειάδες, Λεξικά, Σύνολα (Lists, Tuples, Dictionaries, Sets) και συμβολίζονται με `[...,....,...]`, `(...,...,...)`, `{...,...,...}`, `([...,....,...])` αντίστοιχα.
Το SAGE ακολουθεί αυτό το συμβολισμό το οποίο θα χρησιμοποιήσουμε ευρύτατα, όμως για τον ορισμό ειδικών μαθηματικών αντικειμένων, όπως διανύσματα και πίνακες, το SAGE διαθέτει τις δικές του εξειδικευμένες εντολές. Ως βασική κατηγορία μεταβλητών με δείκτες μπορούμε να θεωρήσουμε τις λίστες

```
s=[17,11,2,3];
u=[9,8,7]
```
```
s[0]
```
17
```
s[1]
```
11
```
s[1:4]
```
[11, 2, 3]
```
s+u
```
[17, 11, 2, 3, 9, 8, 7]

Προσοχή χρειάζεται στην αναφορά των συνιστωσών: Η πρώτη συνιστώσα στην PYTHON αντιστοιχεί στο δείκτη με τιμή 0 (μηδέν).

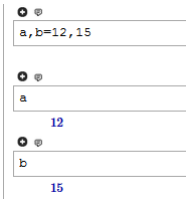
Η PYTHON διαθέτει εξειδικευμένες μεθόδους για το κάθε είδος μεταβλητών με δείκτες.

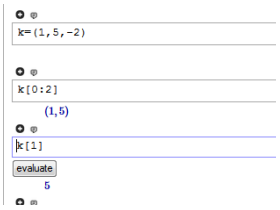
```
s[2]=0;s
```
[17, 11, 0, 3]

```
max(s)
```
17
```
len(s)
```
4
```
s.append(1821);s
```
[17, 11, 0, 3, 1821]
```
s.reverse();s
```
[1821, 3, 0, 11, 17]

Πλειάδες (tuples)

Για την αναπαράσταση των πλειάδων (tuples) χρησιμοποιούμε είτε παρενθέσεις. Μερικές φορές μπορούμε να τις παραλείψουμε





Οι πλειάδες διαφέρουν από τις λίστες οι οποίες συμβολίζονται με αγκύλες [...]. Η κυριότερη διαφορά είναι ότι οι πλειάδες (σε αντίθεση με τις λίστες) δεν μπορούν να τροποποιηθούν μετά τη δημιουργία τους.

Προσοχή στην «άθροιση» των πλειάδων (αντιστοιχεί σε συνένωση) . Το ίδιο ισχύει και για τις λίστες.

Ορισμός συμβολικών μεταβλητών

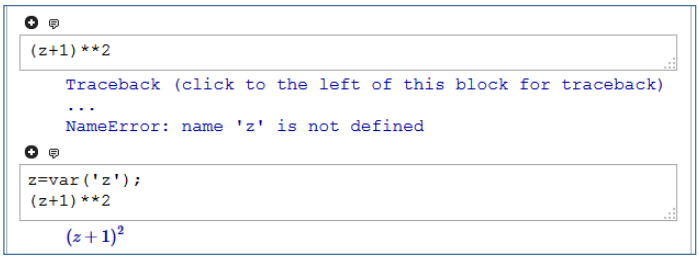
Οι συμβολικές μεταβλητές πρέπει να οριστούν κατάλληλα στο SAGE (με εξαίρεση το x το οποίο έχει προκαθοριστεί ως συμβολική μεταβλητή). Ο τρόπος είναι ο εξής:

```
z=SR.var('z')
```

ή ισοδύναμα

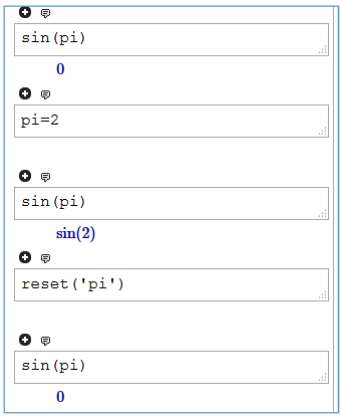
```
z=var('z')
```

Ο ορισμός είναι απαραίτητος για όλες τις μεταβλητές στις συμβολικές εκφράσεις.



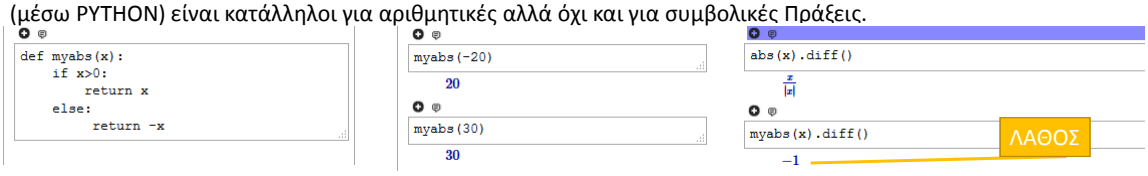
Αποκατάσταση προκαθορισμένων σταθερών

Το SAGE δεν απαγορεύει στο χρήστη να δώσει τιμή σε μια προκαθορισμένη σταθερά. Σε αυτή την περίπτωση η προκαθορισμένη τιμή δεν ισχύει πλέον.
Αν θέλουμε να την αποκαταστήσουμε πρέπει να χρησιμοποιήσουμε τη συνάρτηση `reset` όπως στο διπλανό παράδειγμα.
Συνήθως καταστρέφουμε τη σταθερά `i` (φανταστική μονάδα) με τη χρήση σε εντολές ανακύκλωσης.



Συμβολικές συναρτήσεις και συναρτήσεις PYTHON

Στο SAGE μπορούμε να ορίσουμε συναρτήσεις όπως και στην γλώσσα PYTHON. Οι ορισμοί (μέσω PYTHON) είναι κατάλληλοι για αριθμητικές αλλά όχι και για συμβολικές Πράξεις.



Ο συμβολικές ορισμός συναρτήσεων SAGE είναι απαραίτητος για συμβολικές πράξεις

numerical_approx()

Αριθμητικά αποτελέσματα

Το SAGE προσπαθεί να υπολογίσει αναλυτικά εκφράσεις που αφορούν σε συμβολικές σταθερές ή μεταβλητές. Αν επιθυμούμε αριθμητικά αποτελέσματα θα πρέπει να τα ζητήσουμε με τη βοήθεια της συνάρτησης `numerical_approx(...)` ή αλλιώς `N(...)` η οποία μας επιτρέπει να καθορίσουμε και τον αριθμό των σημαντικών ψηφίων.

```
sin(pi/3)
```
$\frac{1}{2}\sqrt{3}$

```
sin(pi/3.0)
```
$\sin(0.333333333333333\,\pi)$

```
sin(pi/8)
```
$\sin\left(\frac{1}{8}\,\pi\right)$

```
N(sin(pi/8))
```
0.382683432365090

```
N(sin(pi/8),digits=50)
```
0.38268343236508977172845998403039886676134456248563

Παραγοντοποίηση-Ανάπτυγμα αλγεβρικών παραστάσεων

Για την παραγοντοποίηση αλγεβρικών παραστάσεων χρησιμοποιούμε τη συνάρτηση ή τη μέθοδο `factor(...)`

```
factor(x^14 - 7*x^12 + 21*x^10 - 35*x^8 + 35*x^6 - 21*x^4 + 7*x^2 - 1)
```
```
(x + 1)^7*(x - 1)^7
```

```
factor(x^12-1)
```
```
(x^4 - x^2 + 1)*(x^2 + x + 1)*(x^2 - x + 1)*(x^2 + 1)*(x + 1)*(x - 1)
```

Και για το ανάπτυγμα τη συνάρτηση `expand(...)`

```
expand((x-1)^20)
```
```
x^20 - 20*x^19 + 190*x^18 - 1140*x^17 + 4845*x^16 - 15504*x^15 +
38760*x^14 - 77520*x^13 + 125970*x^12 - 167960*x^11 + 184756*x^10 -
167960*x^9 + 125970*x^8 - 77520*x^7 + 38760*x^6 - 15504*x^5 + 4845*x^4 -
1140*x^3 + 190*x^2 - 20*x + 1
```

Χειρισμός παραστάσεων

Μπορούμε να ομαδοποιήσουμε όρους που αφορούν σε μια μεταβλητή με τη μέθοδο `collect(...)`

```
u=3*x^2*y^5 - x*y^6 - 3*x^4*y^2 - 2*x^3*y^3 + x^2*y^4 + x*y^5 - 2*y^6 + 3*x^5 - x^4*y - x^3*y^2 + x^2*y^3 + 2*x*y^4 - 5*y^5 + x^4
- 2*x^3*y + 5*x^2*y^2 + 5*x*y^3 - 5*x^3;
w=4*x^3 - 4*x^2*y + x^2 - 3*x*y + 2*y^2 - 5*x + 5*y;
```

```
u.collect(y)
```

$$-(x+2)y^5+(x-5)y^4+x^4+(x^3+3x^2+2x)y^3-5x^3-(x^3-4x^2-5x)y^2-(x^4+2x^3)y$$

```
w.collect(x)
```

$$4x^3-x^2(4y-1)-x(3y+5)+2y^2+5y$$

Επίσης μπορούμε να απλοποιήσουμε παραγοντοποιώντας ή να αναπτύξουμε σε απλά κλάσματα

```
r=(u/w).factor();r
```

$$\frac{(xy-x+2y+5)(y^2-x)(x+y)}{4x^2+x-2y-5}$$

```
r.partial_fraction()
```

$$\frac{1}{4}y^3-\frac{1}{4}x(y-1)-\frac{1}{2}y^2-\frac{3}{16}y+\frac{40y^4+84y^3+(16y^4+12y^3+48y^2-89y+41)x-46y^2-57y-105}{16(4x^2+x-2y-5)}-\frac{21}{16}$$

Ανάπτυξη-απλοποίηση τριγωνομετρικών παραστάσεων

Ειδικά για τις τριγωνομετρικές συναρτήσεις χρησιμοποιούμε τις `expand_trig()`, `reduce_trig()`, `simplify_trig()`

```
f=(1/32*cos(6*x) + 3/16*cos(4*x) + 15/32*cos(2*x) + 5/16)
f.simplify_trig()
```

$$\cos(x)^6$$

```
(cos(2*x)+sin(3*x)).expand_trig()
```

$$3\cos(x)^2\sin(x)-\sin(x)^3+\cos(x)^2-\sin(x)^2$$

```
f=8*cos(x)^7*sin(x) - 56*cos(x)^5*sin(x)^3 + 56*cos(x)^3*sin(x)^5 - 8*cos(x)*sin(x)^7;
f.reduce_trig()
```

$$\sin(8x)$$

```
sin(10*x).trig_expand()
```

$$10\cos(x)^9\sin(x)-120\cos(x)^7\sin(x)^3+252\cos(x)^5\sin(x)^5-120\cos(x)^3\sin(x)^7+10\cos(x)\sin(x)^9$$

Απλοποίηση παραστάσεων υπό συνθήκες

Για την απλοποίηση αλγεβρικών παραστάσεων χρησιμοποιούμε τη συνάρτηση `simplify(...)`

```
simplify(arctan(sqrt(3)))
```

$\frac{1}{3}\,\pi$

Πιθανές συνθήκες για τις μεταβλητές μπορούν να προστεθούν μέσω της `assume(...)` και να αφαιρεθούν με τη βοήθεια της `forget(...)`

```
assume(x>10);
```

```
simplify(sqrt((x-10)^2))
```

$x - 10$

```
k=var('k');assume(k,'even');
simplify(cos(k*pi))
```

1

```
assumptions()
```

$[x > 10, \mathrm{k\ is\ even}]$

```
forget();assumptions()
```

$[]$

Αναλυτική Επίλυση Εξισώσεων - Συστημάτων

Για την επίλυση εξισώσεων η απλούστερη επιλογή είναι η συνάρτηση `solve(...)`. Πρέπει να προσέξουμε ότι για να εισάγουμε την εξίσωση χρησιμοποιούμε τη διπλή ισότητα ==

```
solve(x^4 - 5*x^3 + x^2 + 25*x - 30 == 0,x)
```

$[x = 3, x = 2, x = -\sqrt{5}, x = \sqrt{5}]$

Για την επίλυση συστημάτων ακολουθούμε την ίδια μέθοδο εισάγοντας τις εξισώσεις σε λίστα

```
y,z=var('y,z');
solve([x+2*x+z==3,7*x-9*y-2*z==1,x+y+7*z==3],x,y,z)
```

$\left[\left[x = \left(\frac{155}{167}\right), y = \left(\frac{94}{167}\right), z = \left(\frac{36}{167}\right)\right]\right]$

Η `solve(...)` επιστρέφει υπό συνθήκες και μερικές λύσεις μη πολυωνυμικών εξισώσεων

```
solve(tan(x+pi/3)==1/2,x)
```

$\left[x = -\frac{1}{3}\,\pi + \arctan\left(\frac{1}{2}\right)\right]$

Αναλυτική Επίλυση Εξισώσεων - Συστημάτων

Στο παρακάτω παράδειγμα λύνουμε την εξίσωση $3x^3 - ax^2 + 2 = 0$ ως προς x και παρουσιάζουμε τις λύσεις με τη σειρά

```
a=var('a');
s=solve(3*x^3-a*x^2+2==0,x);
s[0]
```

$$x = -\frac{a^2\left(-i\sqrt{3}+1\right)}{162\left(\frac{1}{729}a^3+\frac{1}{27}\sqrt{-\frac{2}{3}a^3+81}-\frac{1}{3}\right)^{\frac{1}{3}}} - \frac{1}{2}\left(\frac{1}{729}a^3+\frac{1}{27}\sqrt{-\frac{2}{3}a^3+81}-\frac{1}{3}\right)^{\frac{1}{3}}\left(i\sqrt{3}+1\right)+\frac{1}{9}a$$

```
s[1]
```

$$x = -\frac{a^2\left(i\sqrt{3}+1\right)}{162\left(\frac{1}{729}a^3+\frac{1}{27}\sqrt{-\frac{2}{3}a^3+81}-\frac{1}{3}\right)^{\frac{1}{3}}} - \frac{1}{2}\left(\frac{1}{729}a^3+\frac{1}{27}\sqrt{-\frac{2}{3}a^3+81}-\frac{1}{3}\right)^{\frac{1}{3}}\left(-i\sqrt{3}+1\right)+\frac{1}{9}a$$

```
s[2]
```

$$x = \frac{a^2}{81\left(\frac{1}{729}a^3+\frac{1}{27}\sqrt{-\frac{2}{3}a^3+81}-\frac{1}{3}\right)^{\frac{1}{3}}} + \frac{1}{9}a + \left(\frac{1}{729}a^3+\frac{1}{27}\sqrt{-\frac{2}{3}a^3+81}-\frac{1}{3}\right)^{\frac{1}{3}}$$

Αναλυτική επίλυση πολυωνυμικών εξισώσεων και ανισώσεων

Ειδικά για τις πολυωνυμικές εξισώσεις το SAGE διαθέτει και τη μέθοδο `roots(...)`.
Προσοχή εδώ δίνουμε μόνο το δεξί μέλος μιας εξίσωσης της οποίας το αριστερό ισούται με μηδέν.

```
(x^4 - 3*a*x^3 + a^2*x^2 + 3*a^3*x  -2*a^4).roots(x)
```

```
[(-a, 1), (2*a, 1), (a, 2)]
```

Η `roots(...)` επιστρέφει τις ρίζες και την πολλαπλότητα της κάθε μιας.

Για τις ανισώσεις μπορούμε επίσης να κάνουμε χρήση της `solve(...)`.

```
solve([x-2*x^2+1>0,x-1<2],x)
```

```
[[(-1/2) < x, x < 1]]
```

```
solve([x-2*y>3, x-5*y<2],x,y)
```

```
[[2*y + 3 < x, x < 5*y + 2, (1/3) < y]]
```

Αριθμητική επίλυση εξισώσεων

Ο πιο εύκολος τρόπος αριθμητικής εύρεσης των ριζών πολυωνυμικών εξισώσεων είναι η μέθοδος `roots(...)` όπου καθορίζουμε τον δακτύλιο των λύσεων

```
(x^6-2*x^2-3*x+4).roots(ring=ZZ)
```

```
[(0, 1)]
```

Ακέραιες ρίζες

```
(x^6-2*x^2-3*x+4).roots(ring=RR)
```

```
[(0.000000000000000, 1), (1.42360584855233, 1)]
```

Πραγματικές ρίζες

```
(x^6-2*x^2-3*x+4).roots(ring=CC)
```

```
[(0.000000000000000, 1),
 (1.42360584855233, 1),
 (-0.958532181186730 - 0.498427779031846*I, 1),
 (-0.958532181186730 + 0.498427779031846*I, 1),
 (0.246729256910564 - 1.32081634745025*I, 1),
 (0.246729256910564 + 1.32081634745025*I, 1)]
```

Μιγαδικές ρίζες

Αντίστοιχα αποτελέσματα μπορούμε να επιτύχουμε με κατάλληλη χρήση της `solve(...)`

```
sols=solve(x^4-2*x==0,x);sols
```

$$\left[x = \tfrac{1}{2}\,i\sqrt{3}\,2^{\frac{1}{3}} - \tfrac{1}{2}\cdot 2^{\frac{1}{3}}, x = -\tfrac{1}{2}\,i\sqrt{3}\,2^{\frac{1}{3}} - \tfrac{1}{2}\cdot 2^{\frac{1}{3}}, x = 2^{\frac{1}{3}}, x = 0\right]$$

```
[N(y.rhs()) for y in sols]
```

$$[-0.629960524947437 + 1.09112363597172i, -0.629960524947437 - 1.09112363597172i, 1.25992104989487, 0.000000000000000]$$

Επιλογή του δεξιού μέρους της λύσης (η `solve(...)` επιστρέφει τις λύσεις ως εξισώσεις)

Αριθμητική εύρεση ρίζας μη πολυωνυμικής εξίσωσης

Για μη πολυωνυμικές εξισώσεις δεν υπάρχουν αριθμητικές μέθοδοι οι οποίες να εγγυώνται τον προσδιορισμό όλων των λύσεων. Σε αυτές τις περιπτώσεις είναι χρήσιμη η συνάρτηση `find_root(...)` η οποία μπορεί να Προσδιορίσει μια ρίζα σε δεδομένη περιοχή. Ως παράδειγμα εξετάζουμε τις ρίζες της συνάρτησης

$$\sin(x^2) - \cos(e^x) = 0$$

```
f=sin(x^2)-cos(e^x)
```

```
find_root(f,0,1)
```

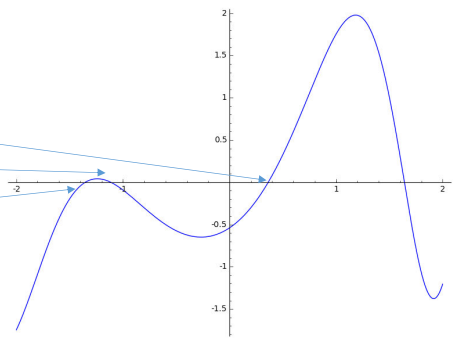
```
0.363642787145
```

```
find_root(f,-1.3,0)
```

```
−1.11481915097
```

```
find_root(f,-2,-1.3)
```

```
−1.35254489297
```



Αντικατάσταση σε αλγεβρική παράσταση

Σε μερικές περιπτώσεις χρειαζόμαστε να αντικαταστήσουμε μια μεταβλητή σε μια αλγεβρική παράσταση. Σε αυτές τις περιπτώσεις χρησιμοποιούμε την μέθοδο `substitute (...)` όπως παρακάτω

```
t=var('t')
```

```
o=x^2-2*x*sin(t)
```

```
o.substitute(t=pi/2)
```

```
    x^2 - 2*x
```

```
o.substitute(x=1,t=pi/3)
```

```
    -sqrt(3) + 1
```

```
eq=x^4-x^2-x-1==0
```

Αντικατάσταση ρίζας εξίσωσης σε αλγεβρική παράσταση.

```
p=x^4-2*x^2-5;
sols=solve(x^2-5*x+2==0,x);sols
```

$$\left[x = -\frac{1}{2}\sqrt{17} + \frac{5}{2}, x = \frac{1}{2}\sqrt{17} + \frac{5}{2}\right]$$

```
p.substitute(sols[0]).expand()
```

$$-\frac{95}{2}\sqrt{17} + \frac{381}{2}$$

Απλός Προγραμματισμός (βρόχος επανάληψης, έλεγχος ροής)

Στις βασικές διαδικασίες του προγραμματισμού, συμπεριλαμβάνονται ο βρόχος επανάληψη και ο έλεγχος ροής την απλοποιημένη χρήση των οποίων δείχνουμε στα παρακάτω παραδείγματα

```
p=x^4-2*x^3-3*x-1;
s=p.roots(ring=CC);s
```

```
    [(-0.310319971846122, 1),
     (2.53030083756844, 1),
     (-0.109990432861157 - 1.12314660756467*I, 1),
     (-0.109990432861157 + 1.12314660756467*I, 1)]
```

```
for xx in s:
    print xx[0]
```

```
    -0.310319971846122
    2.53030083756844
    -0.109990432861157 - 1.12314660756467*I
    -0.109990432861157 + 1.12314660756467*I
```

```
for i in range(0,len(s)):
    print u'Ρίζα :',i+1,s[i][0]
```

```
    Ρίζα : 1 -0.310319971846122
    Ρίζα : 2 2.53030083756844
    Ρίζα : 3 -0.109990432861157 - 1.12314660756467*I
    Ρίζα : 4 -0.109990432861157 + 1.12314660756467*I
```

```
for i in range(0,len(s)):
    root,mult=s[i]
    if imag_part(root)==0:
        print u'Ρίζα',i+1,'(Πραγματική):',root,', Πολλ:',mult
    else:
        print u'Ρίζα',i+1,'(Μιγαδική)  :',root,', Πολλ:',mult
```

```
    Ρίζα 1 (Πραγματική): -0.310319971846122 , Πολλ: 1
    Ρίζα 2 (Πραγματική): 2.53030083756844 , Πολλ: 1
    Ρίζα 3 (Μιγαδική)  : -0.109990432861157 - 1.12314660756467*I , Πολλ: 1
    Ρίζα 4 (Μιγαδική)  : -0.109990432861157 + 1.12314660756467*I , Πολλ: 1
```

Η συνάρτηση `range(...)`

```
a = range(5)          # a = [0,1,2,3,4]
b = range(1,8)        # b = [1,2,3,4,5,6,7]
c = range(0,14,3)     # c = [0,3,6,9,12]
d = range(8,1,-1)     # d = [8,7,6,5,4,3,2]
```

Η συνάρτηση επιστρέφει ακεραίους . Η γενικότερη συνάρτηση μπορεί να χρησιμοποιηθεί για δεκαδικούς.

```
[m for m in srange(1,8,0.82)]
```

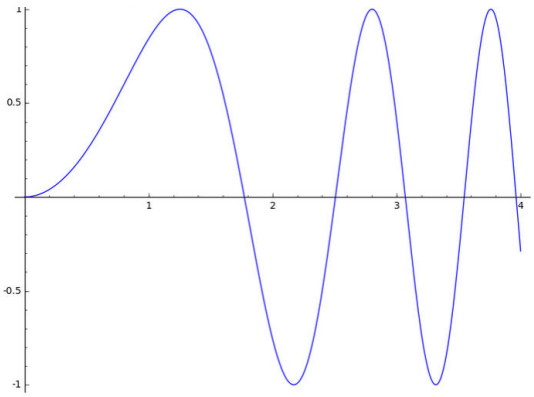
```
[1.00000000000000,
 1.82000000000000,
 2.64000000000000,
 3.46000000000000,
 4.28000000000000,
 5.10000000000000,
 5.92000000000000,
 6.74000000000000,
 7.56000000000000]
```

Γραφικά

Γραφική αναπαράσταση συναρτήσεων

Η βασική εντολή γραφικής αναπαράστασης συναρτήσεων μιας μεταβλητής είναι η `plot(...)`

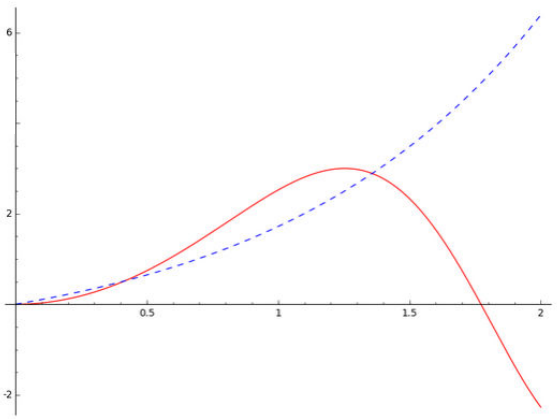
```
plot(sin(x^2),(x,0,4))
```



Γραφική αναπαράσταση συναρτήσεων

Τα γραφικά μπορούν να μορφοποιηθούν με την πληθώρα παραμέτρων που διαθέτει η `plot(...)` ή να συνδυαστούν όπως φαίνεται στο παράδειγμα

```
g1=plot(3*sin(x^2),(x,0,2),color="red");
g2=plot(e^x-1,(x,0,2),linestyle="--");
g1+g2
```



Για αναλυτικές πληροφορίες σχετικά με τις παραμέτρους της `plot(...)` δείτε στη βοήθεια

```
plot?
```

File: /usr/lib/sagemath/local/lib/python2.7/site-packag

Type: <type 'function'>

Definition: plot(funcs, exclude=None, fillalpha=0.5, fi
aspect_ratio='automatic', alpha=1, legend_label=Non

Docstring:

Use plot by writing

Γραφική αναπαράσταση καμπυλών σε παραμετρική μορφή

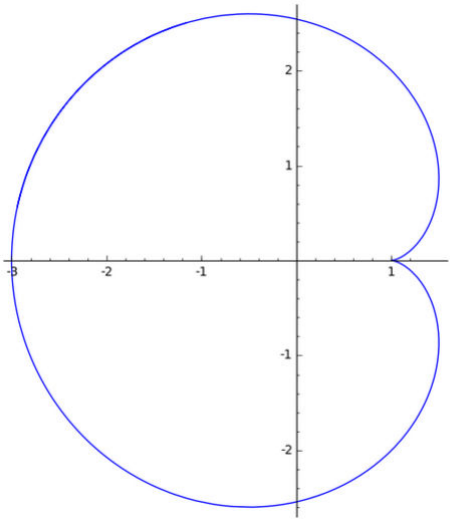
Η γραφική αναπαράσταση καμπυλών σε παραμετρική μορφή γίνεται με την `parametric_plot(...)` όπως φαίνεται στο παράδειγμα

Γραφική αναπαράσταση της καμπύλης

$$x(t) = 2\cos(t) - \cos(2t)$$
$$y(t) = 2\sin(t) - \sin(2t)$$

```
t=var('t')
```

```
parametric_plot((2*cos(t)-cos(2*t),2*sin(t)-sin(2*t)),(t,-4,3))
```



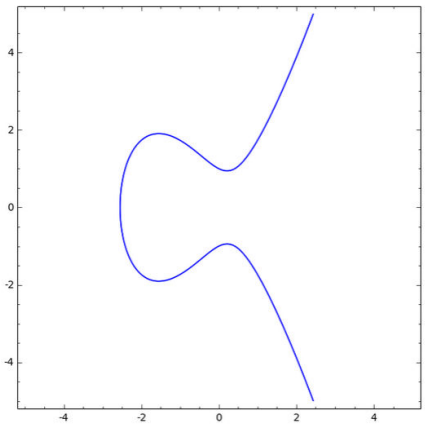
Καμπύλες σε κλειστή μορφή

Για καμπύλες σε κλειστή μορφή χρησιμοποιούμε την `implicitic_plot(...)` όπως φαίνεται στο παράδειγμα

Γραφική αναπαράσταση της καμπύλης

$$y^2 - x^3 - 2x^2 + x - 1 = 0$$

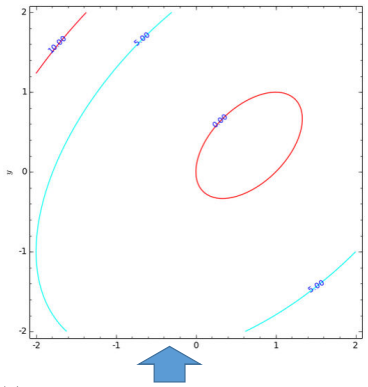
```
implicit_plot(y^2-x^3-2*x^2+x-1,(x,-5,5),(y,-5,5))
```

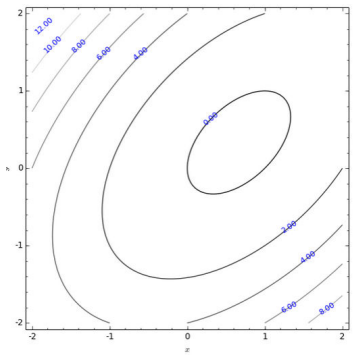


Γραφικά τύπου contour

Τα γραφικά συναρτήσεων δύο μεταβλητών τύπου contour υλοποιούνται ως κάτωθι

```
y=var('y');
contour_plot(x^2+y^2-x*(y+1),(x,-2,2),(y,-2,2),fill=False,labels=True)
```



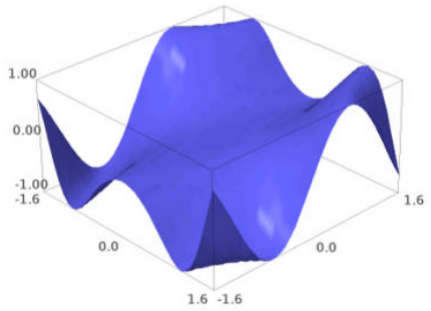


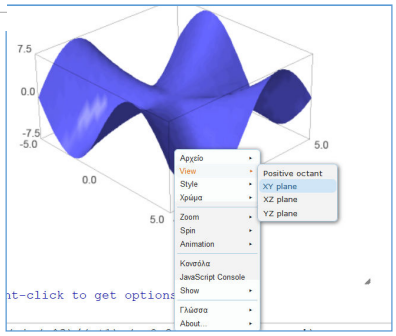

```
contour_plot(x^2+y^2-x*(y+1),(x,-2,2),(y,-2,2),fill=False,labels=True,axes_labels=['$x$','$y$'],contours=[0,5,10],cmap='hsv')
```

Γραφικά στις τρεις διαστάσεις

Για τη γραφική αναπαράσταση συναρτήσεων δύο μεταβλητών χρησιμοποιούμε συνήθως τρισδιάστατη Αναπαράσταση με τη βοήθεια της `plot3d(...)` όπως παρακάτω

```
x,y=var('x y');
plot3d(x*y*(x^2 - y^2) / (x^2 + y^2),(x,-5,5),(y,-5,5))
```

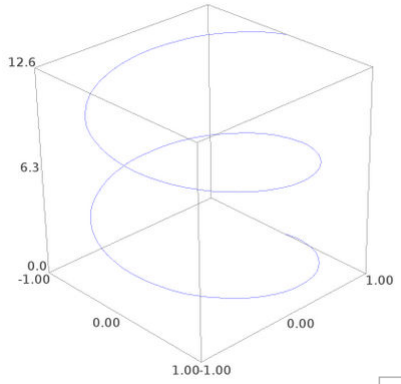


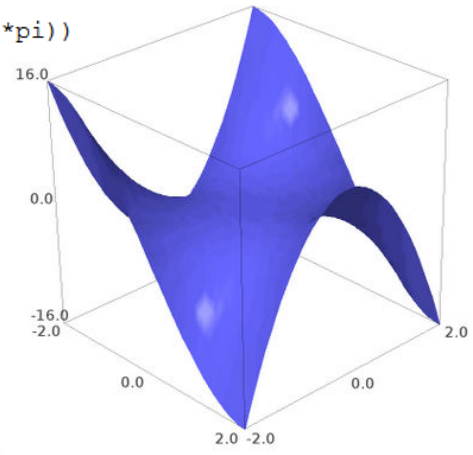


Μπορούμε να μορφοποιήσουμε το γραφικό με το δεξί κλικ και το ενσωματωμένο μενού .

Παραμετρικά γραφικά στις τρεις διαστάσεις

```
u=var('u');
parametric_plot3d([sin(u),cos(u),u], (u,0,4*pi))
```



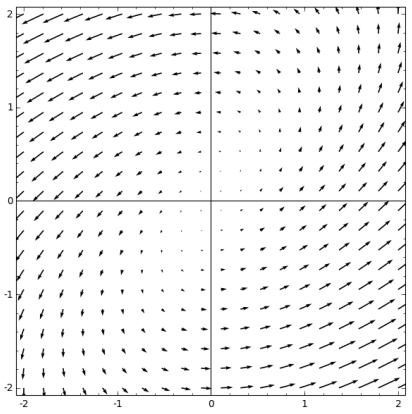



```
v,u=var('v u')
parametric_plot3d((u, v,u^3-3*u*v^2),(u,-2,2),(v,-2,2))
```

Γραφική αναπαράσταση διανυσματικών πεδίων

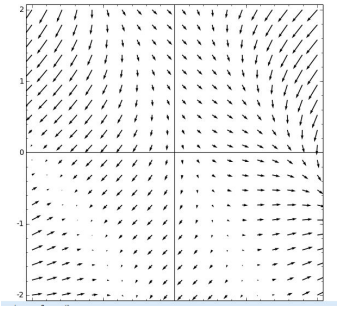
Χρήσιμη στη Φυσική είναι και η γραφική αναπαράσταση διανυσματικών πεδίων η οποία πραγματοποιείται με τη βοήθεια της `plot_vector_field(...)` όπως παρακάτω

```
y=var('y');plot_vector_field((x-y,x),(x,-2,2),(y,-2,2),aspect_ratio=1)
```



```
y=var('y');f=x*sin(x+y)-y;
plot_vector_field(f.gradient(),(x,-2,2),(y,-2,2),aspect_ratio=1)
```

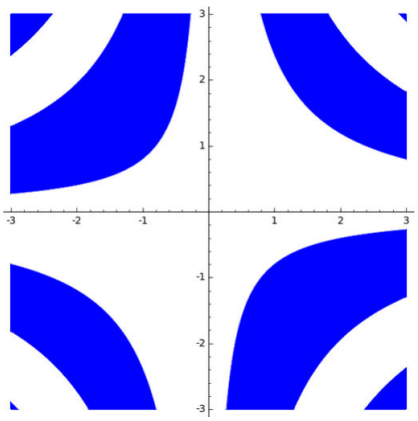
Η μέθοδος `gradient()` υπολογίζει αυτόματα την απόκλιση μιας βαθμωτής συνάρτησης.



Γραφική αναπαράσταση περιοχών

Η συνάρτηση `region_plot(...)` είναι χρήσιμη για τη γραφική αναπαράσταση περιοχών όπως παρακάτω

```
region_plot(sin(x*y)+cos(x*y) <= 0, (x, -3, 3), (y, -3, 3),aspect_ratio=1)
```

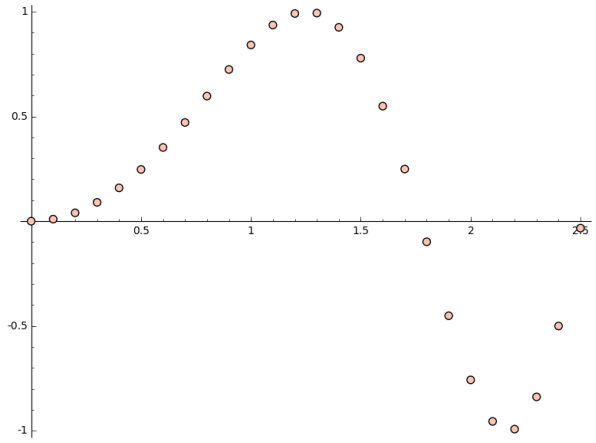


Γραφική αναπαράσταση δεδομένων

Η `scatter_plot(...)` είναι χρήσιμη για τη γραφική αναπαράσταση δεδομένων, τα οποία εισάγουμε σε ζευγάρια με μορφή πίνακα, όπως στο παράδειγμα

```
data=[[0.0, 0.0],[0.1, 0.01],
 [0.2, 0.04],[0.3, 0.09],
 [0.4, 0.159],[0.5, 0.247],
 [0.6, 0.352],[0.7, 0.471],
 [0.8, 0.597],[0.9, 0.724],
 [1.0, 0.841],[1.1, 0.936],
 [1.2, 0.991],[1.3, 0.993],
 [1.4, 0.925],[1.5, 0.778],
 [1.6, 0.549],[1.7, 0.249],
 [1.8, -0.098],[1.9, -0.451],
 [2.0, -0.757],[2.1, -0.955],
 [2.2, -0.992],[2.3, -0.838],
 [2.4, -0.5],[2.5, -0.033]]
```

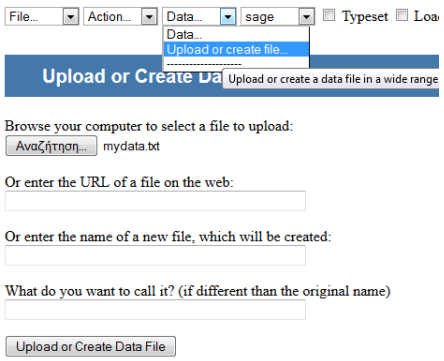
```
g1=scatter_plot(data);g1
```



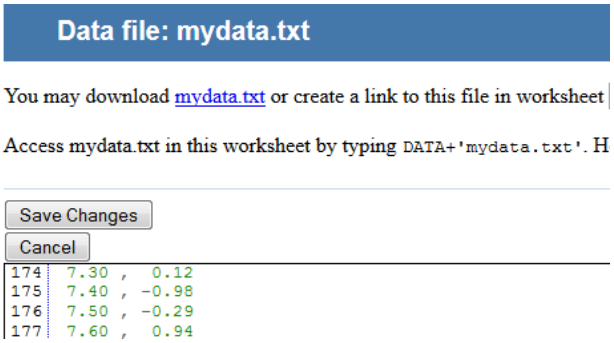
Εισαγωγή δεδομένων από αρχείο

Η εισαγωγή δεδομένων από αρχείο μπορεί να πραγματοποιηθεί με τρία απλά βήματα όπως παρακάτω:

1. Αποστολή αρχείου στον server του SAGE



2. Αποθήκευση αρχείου στον server του SAGE

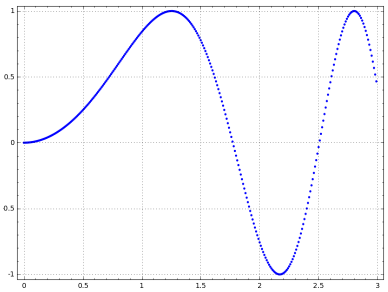


Εισαγωγή δεδομένων από αρχείο (συνέχεια)

3. Ανάγνωση αρχείου και αποθήκευση σε πίνακα

Η χρήση της float είναι απαραίτητη για τη μετατροπή των δεδομένων σε πραγματικούς αριθμούς

```
dfile=open(DATA+'mydata.txt','r')
data=[];
for line in dfile:
    a,b=line.split(',')
    data.append([float(a),float(b)])
```



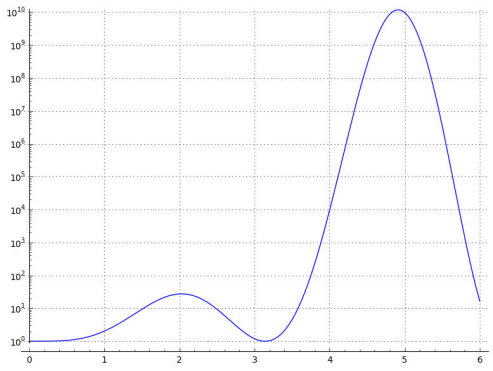
Κατόπιν μπορούμε να επεξεργαστούμε τα δεδομένα όπως Επιθυμούμε. Για παράδειγμα μπορούμε να δημιουργήσουμε μια γραφική αναπαράσταση ως εξής:

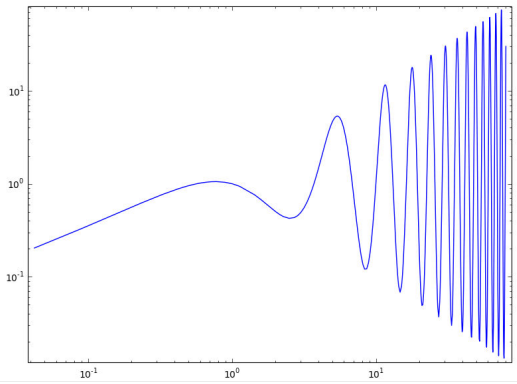
```
points(data,frame=True,axes=False,gridlines=True)
```

Λογαριθμικοί άξονες

Με κατάλληλη επιλογή των παραμέτρων της `plot(...)` μπορούμε να έχουμε λογαριθμικούς άξονες

```
plot(e^(x^2*sin(x)^2),(x,0,6),scale="semilogy",gridlines=True)
```



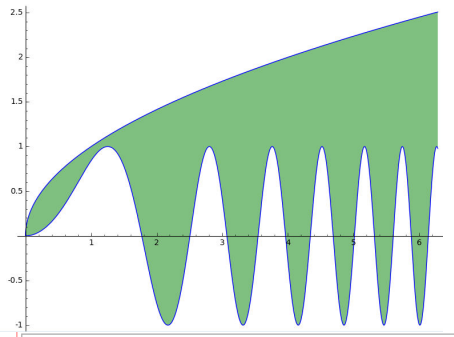


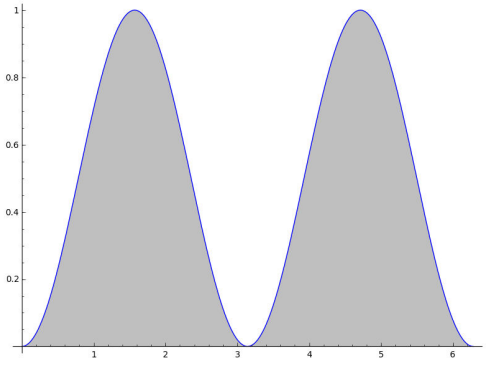
```
plot(x^(cos(x+1)),(x,-2,80),scale="loglog",frame=True)
```

Μορφοποίηση γραφικών

Με κατάλληλη επιλογή των παραμέτρων της `plot(...)` μπορούμε επίσης να σημειώσουμε περιοχές μεταξύ καμπυλών, όπως στα παραδείγματα

```
plot(sin(x)^2,(x,0,2*pi),fill='axis')
```

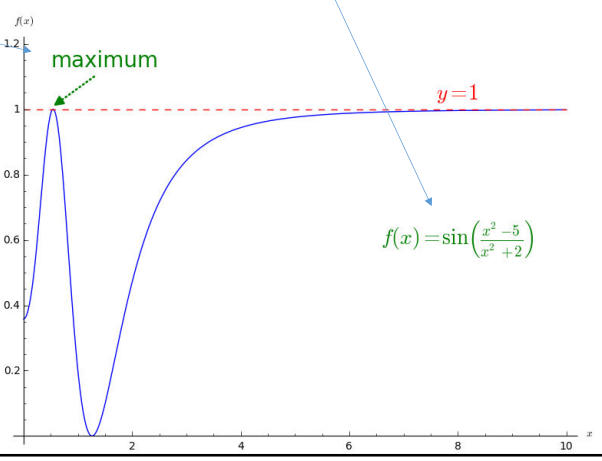




```
plot([sqrt(x),sin(x^2)],(x,0,2*pi),fill={0:[1]},fillcolor='green')
```

Μορφοποίηση γραφικών: Χρήση LaTeX

```
p=plot(sin((pi*x^2-5)/(2*x^2+2))^2,(x,0,10));
p1=plot(1,(x,0,10),color='red',linestyle='--');
pt=text(r'$f(x)=\sin\left(\frac{x^2-5}{x^2+2}\right)$',(8,0.6),fontsize=20,color='green', \
        ymax=1.2,axes_labels=['$x$','$f(x)$']);
pta=text('$y=1$',(8,1.05),fontsize=20,color='red');
par=arrow2d((1.3,1.1),(0.55,1.01),color='green',linestyle=':');
pax=text('maximum',(1.5,1.15),fontsize=20,color='green');
p+p1+pt+pta+par+pax
```



Παραδείγματα συμβολισμού σε LaTeX

$g_{\mu\nu}X^\mu X^\nu$ `g_{\mu\nu}X^\mu X^\nu`

$\frac{x}{\sqrt{x+1}}$ `\frac{x}{\sqrt{x+1}}`

$\int_{-\infty}^{\infty} dx e^{-x^2} = \sqrt{\pi}$ `\int_{-\infty}{\infty}dx e^{-x^2} =\sqrt{\pi}`

Αποθήκευση γραφικών σε αρχείο

Με τη βοήθεια της μεθόδου `save(...)` μπορούμε να αποθηκεύουμε τα γραφικά σε αρχείο στην επιθυμητή μορφή (η οποία συνάγεται από την κατάληξη του ονόματος)

```
g=plot(log(abs(sin(x)+2)),(x,-2*pi,2*pi));
g.save('plot.pdf')
```

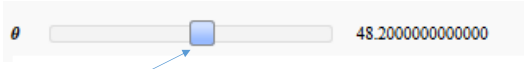
plot.pdf

```
g.save('plot.eps')
```

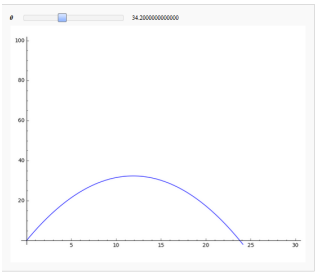
plot.eps

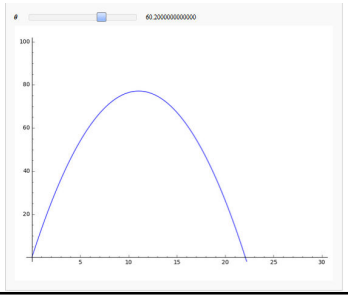
Αλληλεπιδραστικά γραφικά

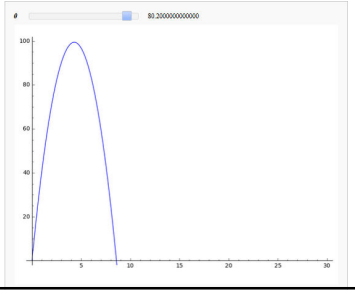
Μπορούμε να δημιουργήσουμε αλληλεπιδραστικά γραφικά με τη βοήθεια μιας συνάρτησης και του decorator @interact όπως στο παράδειγμα




```
v0=8;g=10;
@interact
def voli(u=slider(0.2,89,2,30,r'$\theta$')):
    th=u*pi/180;
    gg=plot(v0*tan(th)*x-g*x^2/(v0^2*cos(th)^2),(x,0,30),ymin=0,ymax=80);
    show(gg)
```





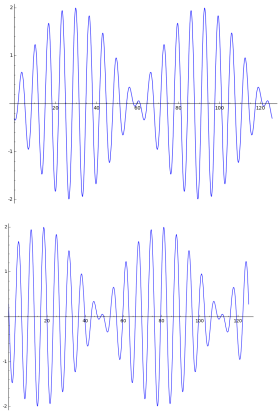


Κινούμενα Γραφικά (Animation)

Η δημιουργία κινούμενων γραφικών, χρήσιμων σε προσομοιώσεις στη Φυσική, μπορεί να γίνει εύκολα σε δύο βήματα: (α) δημιουργούμε μια σειρά από στιγμιότυπα και τα αποθηκεύουμε σε ένα πίνακα (β) Χρησιμοποιούμε τη συνάρτηση `animate(...)` για την παραγωγή του animation. Μετά με τη βοήθεια της `show(...)` μπορούμε να εμφανίσουμε τα γραφικά στην οθόνη. Παρακάτω εξετάζουμε την επαλληλία δύο κυμάτων με παραπλήσιες συχνότητες που οδηγεί στην παραγωγή διακροτημάτων

```
w1=1;k1=1;w2=0.9;k2=0.9;
tmin=0;tmax=5*pi;ns=30.;tstep=(tmax-tmin)/ns;
data = [plot(sin(k1*x-w1*t)+sin(k2*x-w2*t),(x,0,40*pi),ymin=-2,ymax=2) \
       for t in sxrange(tmin,tmax+tstep,tstep)];
a=animate(data);
```

```
a.show(delay=10,iterations=10)
```



Μπορούμε να αποθηκεύσουμε τα αποτελέσματα σε αρχείο βίντεο τύπου .avi και μετά να τα κατεβάσουμε στον υπολογιστή μας, όπως παρακάτω

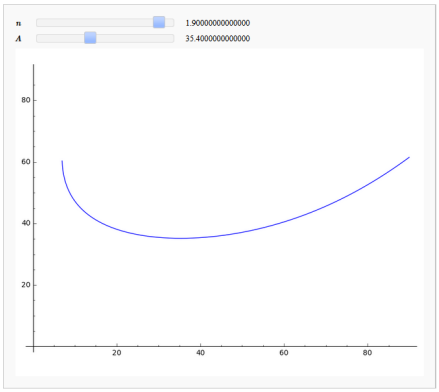
```
a.save(filename='anim.avi')
```

anim.avi

Παράδειγμα: Γωνία απόκλισης πρίσματος

Να δημιουργηθεί ένα αλληλεπιδραστικό γραφικό στο οποίο να απεικονίζεται η γωνία απόκλισης πρίσματος, δείκτη διάθλασης n και γωνίας A, συναρτήσει της γωνίας πρόσπτωσης.

```
@interact
def delta(nn=slider(1.1,2,0.1,1.3,'$n$'),a=slider(1.,90.,0.2,30.,'$A$')):
    data=[]
    for ii in sxrange(0.,90.,0.05):
        aa=float(a*pi/180.);
        rii=float(ii*pi/180.)
        r=arcsin(sin(rii)/nn)
        rp=aa-r
        w=float(nn*sin(rp))
        if w <=1.:
            ip=arcsin(w)
            mip=ip*180./pi
            d=ii+mip-a
            data.append([ii,d])
    a=line(data,xmin=0,xmax=90,ymin=0,ymax=90)
    show(a)
```



Προχωρημένοι Υπολογισμοί

Ανάπτυγμα σε σειρά

Το ανάπτυγμα συνάρτησης σε σειρά μπορεί να πραγματοποιηθεί με τη βοήθεια των συναρτήσεων/μεθόδων `series(...)` και `taylor(...)` όπως παρακάτω

```
f=log(1+x);
f.taylor(x,0,10)
```

$$-\frac{1}{10}x^{10}+\frac{1}{9}x^9-\frac{1}{8}x^8+\frac{1}{7}x^7-\frac{1}{6}x^6+\frac{1}{5}x^5-\frac{1}{4}x^4+\frac{1}{3}x^3-\frac{1}{2}x^2+x$$

```
f1=arctan(x).series(x==0,10);f1
```

$$1x+(-\frac{1}{3})x^3+\frac{1}{5}x^5+(-\frac{1}{7})x^7+\frac{1}{9}x^9+\mathcal{O}(x^{10})$$

```
f2=sin(x).series(x==pi,4);f2
```

$$(-1)(-\pi+x)+\frac{1}{6}(-\pi+x)^3+\mathcal{O}\left((\pi-x)^4\right)$$

```
f2.truncate()
```

$$\pi-\frac{1}{6}(\pi-x)^3-x$$

Διανύσματα

Για τον ορισμό των διανυσμάτων το SAGE χρησιμοποιεί την ειδική συνάρτηση `vector(...)` όπως στο παράδειγμα

```
a=vector(QQ,[1,2/3,-1]);
b=vector(QQ,[-3,1/3,2]);
```

όπου το πρώτο όρισμα είναι ο δακτύλιος (ή πεδίο) ορισμού και το δεύτερο η λίστα με τις συνιστώσες. Μετά τον ορισμό μπορούμε να εκτελούμε τις συνήθεις πράξεις όπως παρακάτω:

Πρόσθεση

```
a+b
```

```
(-2, 1, 1)
```

Εσωτερικό γινόμενο

```
a.dot_product(b)
```

```
-43/9
```

Εξωτερικό γινόμενο

```
a.cross_product(b)
```

```
(5/3, 1, 7/3)
```

Για τις συνιστώσες του διανύσματος χρησιμοποιούμε συμβολισμό παρόμοιο με τις λίστες (προσοχή το πρώτο στοιχείο είναι αυτό με δείκτη 0)

```
a[0]
```

```
1
```

```
b[1]
```

```
1/3
```

Επίσης το μέτρο υπολογίζεται με τη βοήθεια της `norm(...)`

```
norm(a)
```

```
1/3*sqrt(22)
```

Κλίση, απόκλιση, περιστροφή

Για τον ορισμό της κλίσης μιας βαθμωτής συνάρτησης χρησιμοποιούμε τη μέθοδο `gradient(...)` όπως στο παράδειγμα

```
x,y,z=var('x y z');
f=(x-z)^2+y^2*x+z*x;
f.gradient([x,y,z])
```

```
(y^2 + 2*x - z, 2*x*y, -x + 2*z)
```

Η παράγωγος διανυσματικής συνάρτησης υπολογίζεται με τη βοήθεια της μεθόδου `derivative(...)` όπως παρακάτω

```
vector([-x^2*z, x,z*x]).derivative(x)
```

```
(-2*x*z, 1, z)
```

Για την απόκλιση και περιστροφή το SAGE διαθέτει τις μεθόδους `div(...)`, `curl(...)` αντίστοιχα, όμως αυτές δεν είναι λειτουργικές τουλάχιστον στην έκδοση 6.5 (στην οποία βασίζονται αυτές οι σημειώσεις), και είναι απαραίτητος ο ορισμός τους κατά περίπτωση όπως παρακάτω

```
f=vector([-x^2*z, x,z*x])
divf=diff(f[0],x)+diff(f[1],y)+diff(f[2],z)
divf
```

```
-2*x*z + x
```

```
curl=vector([diff(f[2],y)-diff(f[1],z),diff(f[0],z)-diff(f[2],x), \
             diff(f[1],x)-diff(f[0],y)]);
curl
```

```
(0, -x^2 - z, 1)
```

Πίνακες

Για τον ορισμό πινάκων χρησιμοποιούμε την ειδική συνάρτηση `matrix(...)` όπως στο παράδειγμα

```
m=matrix(SR,3,3,[3,2,4,2,0,2,4,2,4]);
m
```

$$\begin{pmatrix} 3 & 2 & 4 \\ 2 & 0 & 2 \\ 4 & 2 & 4 \end{pmatrix}$$

```
a=matrix(SR,3,3,[1,0,1,1,0,1,2,2,2]);a
```

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 2 & 2 & 2 \end{pmatrix}$$

Μπορούμε ακολούθως να εκτελέσουμε πράξεις με τους πίνακες ή να υπολογίσουμε τα ιδιοανύσματα και ιδιοτιμές όπως παρακάτω

```
a*m
```

$$\begin{pmatrix} 7 & 4 & 8 \\ 7 & 4 & 8 \\ 18 & 8 & 20 \end{pmatrix}$$

```
m^(-1)
```

$$\begin{pmatrix} -1 & 0 & 1 \\ 0 & -1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 \end{pmatrix}$$

```
det(m)
```

$$4$$

```
b=vector(SR,[1,2,-1]);m*b
```

$$(3, 0, 4)$$

```
matrix.identity(3)
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
m.eigenvalues()
```

$$[-2\sqrt{5}+4, 2\sqrt{5}+4, -1]$$

```
m.characteristic_polynomial()
```

$$x^3 - 7x^2 - 12x - 4$$

```
m.eigenvalues()
```

$$[-2\sqrt{5}+4, 2\sqrt{5}+4, -1]$$

```
e=m.eigenvectors_right();e
```

$$\left[\left(-2\sqrt{5}+4, \left[\left(1, \frac{1}{2}, -\frac{1}{2}\sqrt{5}\right)\right], 1\right), \left(2\sqrt{5}+4, \left[\left(1, \frac{1}{2}, \frac{1}{2}\sqrt{5}\right)\right], 1\right), (-1, [(1, -2, 0)], 1)\right]$$

Ολοκλήρωση (αναλυτικά)

Για την αναλυτική ολοκλήρωση χρησιμοποιούμε τη συνάρτηση `integrate(...)` όπως παρακάτω

```
integrate(sin(x)^2,x)
```

$$\frac{1}{2}x - \frac{1}{4}\sin(2x)$$

```
integrate(sin(x^2),x,0,Infinity)
```

$$\frac{1}{4}\sqrt{2}\sqrt{\pi}$$

Μπορεί να χρειαστεί να προσδιορίσουμε περιοχές τιμών για κάποιες παραμέτρους, όπως

```
a=var('a');
integrate(exp(-a*x^2),x,-Infinity,Infinity)
```

```
Traceback (click to the left of this block for traceback)
...
Is a positive, negative or zero?
```

```
assume(a>0);
integrate(exp(-a*x^2),x,-Infinity,Infinity)
```

$$\frac{\sqrt{\pi}}{\sqrt{a}}$$

Ολοκλήρωση (αριθμητικά)

Για την αριθμητική ολοκλήρωση χρησιμοποιούμε τη συνάρτηση numerical_integral(...) η οποία επιστρέφει δύο αριθμούς, το αποτέλεσμα και μια εκτίμηση για το λάθος.

```
numerical_integral(exp(-x^2)*sin(x),0,infinity)
```

$$\left(0.424436383502, 5.85997145025 \times 10^{-08}\right)$$

Μπορούμε να εξάγουμε την (μόνο) τιμή του ολοκληρώματος όπως παρακάτω

```
numerical_integral(exp(-x^2)*sin(x),0,1)[0]
```

$$0.294698182249$$

Η συνάρτηση numerical_integral(...) περιλαμβάνει μια σειρά από παραμέτρους οι οποίες μας επιτρέπουν να επιλέξουμε τη μέθοδο αριθμητικής ολοκλήρωσης, τον αριθμό των σημείων κλπ

Διαφορικές εξισώσεις (αναλυτική επίλυση)

Για την αναλυτική επίλυση διαφορικών εξισώσεων χρησιμοποιούμε τη συνάρτηση desolve(...) , αφού βέβαια ορίσουμε κατάλληλα την εξαρτημένη και την ανεξάρτητη μεταβλητή. Στο παρακάτω παράδειγμα επιλύουμε τη διαφορική εξίσωση

$$y'(x) + y(x) = \sin(x)$$

```
x = var('x'); y = function('y', x)
```

```
z=desolve(diff(y,x) + y == sin(x), y);z
```

$$-\frac{1}{2}\left((\cos(x) - \sin(x))e^{x} - 2\,C\right)e^{(-x)}$$

```
expand(z)
```

$$Ce^{(-x)} - \frac{1}{2}\cos(x) + \frac{1}{2}\sin(x)$$

Μερικές φορές το SAGE επιστρέφει τη λύση σε μορφή εξίσωσης η οποία χρειάζεται περαιτέρω επεξεργασία, όπως παρακάτω

```
yy=desolve(diff(y,x) + y+y^2 == 0, y);
yy
```

$$\log(y(x) + 1) - \log(y(x)) = C + x$$

```
solve(yy.simplify_log(),y(x))
```

$$\left[y(x) = \frac{1}{e^{(C+x)} - 1}\right]$$

Διαφορικές εξισώσεις (αναλυτική επίλυση, συνέχεια)

Οι αρχικές/συνοριακές συνθήκες δίνονται σε ειδική μορφή με την παράμετρο `ics= [...]` ανάλογα με την τάξη της εξίσωσης. Στο παρακάτω παράδειγμα, διαφορική εξίσωση πρώτης τάξης δίνουμε το x0 και το y(x0)

```
g(x)=desolve(diff(y,x) + y == sinh(x), y,ics=[0,1]).expand();
g(x)
```

$$-\frac{1}{2}\,x e^{(-x)} + \frac{3}{4}\,e^{(-x)} + \frac{1}{4}\,e^{x}$$

```
g(0)
```

$$1$$

Για διαφορικές εξισώσεις 2ης τάξης υπάρχουν δύο τρόποι, ανάλογα με το είδος των συνθηκών

```
desolve(diff(y,x,2) + 4*diff(y,x)+3*y == sin(x), y,ics=[0,1,0])
```

y(0)=1, y'(0)=0

$$-\frac{1}{5}\cos(x) + \frac{7}{4}\,e^{(-x)} - \frac{11}{20}\,e^{(-3x)} + \frac{1}{10}\sin(x)$$

```
desolve(diff(y,x,2) + 4*diff(y,x)+3*y == sin(x), y,ics=[0,1,1,0])
```

y(0)=1, y(1)=0

$$\frac{(2\cos(1)e^3 - e^3\sin(1) - 12)e^{(-x)}}{10\,(e^2-1)} - \frac{(2\cos(1)e^3 - e^3\sin(1) - 12\,e^2)e^{(-3x)}}{10\,(e^2-1)} - \frac{1}{5}\cos(x) + \frac{1}{10}\sin(x)$$

Διαφορικές Εξισώσεις (αριθμητική επίλυση)

Ως υπόδειγμα για την αριθμητική επίλυση **συστημάτων διαφορικών εξισώσεων πρώτης τάξης** χρησιμοποιούμε την `desolve_odeint(..)` όπως στο παράδειγμα:

$$\frac{dx(t)}{dt} = x(t)^2 - y(t)$$
$$\frac{dy(t)}{dt} = x(t)y(t) + x(t)$$
$$x(0) = 1, y(0) = 2$$

Δημιουργία μια λίστας σημείων όπου επιθυμούμε να πάρουμε τη λύση. Εδώ 0<t<30 με βήμα 0.05

```
tt=srange(0,30.05,0.05);
sol=desolve_odeint([x^2-y,x*y+x],[1,2],tt,[x,y],ivar=t)
```

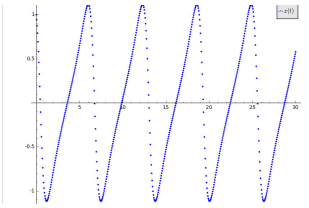
Δεξί μέρος διαφορικών εξισώσεων

Αρχικές συνθήκες x(0), y(0)

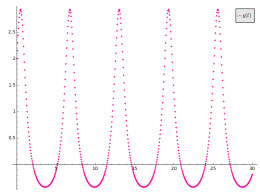
Διαφορικές Εξισώσεις (συνέχεια)

Μπορούμε να αναπαραστήσουμε γραφικά τα αποτελέσματα ως κάτωθι

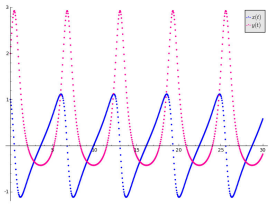
```
g1=points(zip(tt,sol[:,0]),legend_label='$x(t)$');g1
```



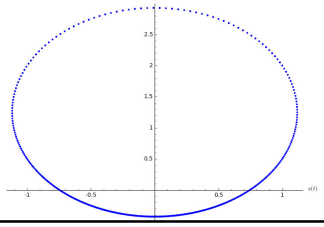
```
g2=points(zip(tt,sol[:,1]),legend_label='$y(t)$',hue=0.9);g2
```



```
g1+g2
```



```
points(sol,axes_labels=['$x(t)$','$y(t)$'])
```



Διαφορικές Εξισώσεις (συνέχεια)

Για την επίλυση διαφορικών εξισώσεων ανώτερης τάξης χρησιμοποιούμε και πάλι την `desolve_odeint(..)` αφού μετατρέψουμε τις διαφορικές εξισώσεις σε σύστημα πρώτης τάξης, όπως στο παράδειγμα

$$\frac{d^2x(t)}{dt^2} = -a\,x(t)^3 - b\,x(t) - \cos(\omega\,t) \quad \text{με} \quad x(0) = 1, x'(0) = 0$$

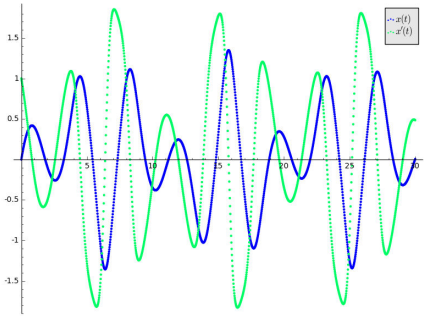
Το οποίο γράφεται ισοδύναμα ως

$$\frac{dx(t)}{dt} = y(t)$$
$$\frac{dy(t)}{dt} = -a\,x(t)^3 - b\,x(t) - \cos(\omega\,t) \quad \text{με} \quad x(0) = 1, y(0) = 0$$

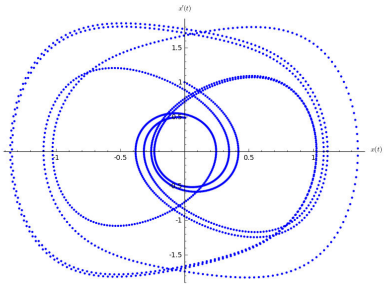
Διαφορικές Εξισώσεις (συνέχεια)

```
t=var('t');x=var('x');y=var('y');
a=2;b=1;w=1;
tt=srange(0,30.02,0.02);
sol=desolve_odeint([y,-a*x^3-b*x-cos(w*t)],[0,1],tt,[x,y],ivar=t)
```

```
g1=points(zip(tt,sol[:,0]),legend_label='$x(t)$');
g2=points(zip(tt,sol[:,1]),hue=0.4,legend_label='$x\'(t)$');
g1+g2
```




```
points(sol,axes_labels=['$x(t)$','$x\'(t)$'])
```



Εφαρμογές στα Μαθηματικά

Τυχαίοι αριθμοί – Αριθμητικά πειράματα

Πολλές φορές μπορούμε να χρησιμοποιήσουμε αριθμητικά πειράματα αφενός ως μέσο επίλυσης προβλημάτων στα Μαθηματικά και τη Φυσική και αφετέρου ως εύκολο μέσο επίδειξης χαρακτηριστικών ιδιοτήτων φυσικών συστημάτων.
Για το σκοπό αυτό χρειαζόμαστε μια γεννήτρια τυχαίων (ή για την ακρίβεια ψευδοτυχαίων) αριθμών.
Το SAGE διαθέτει δύο συναρτήσεις που χρησιμοποιούνται ως κάτωθι:

Τυχαίος αριθμός μεταξύ 0,1

```
random()
```

```
0.3675276790228389
```

Πίνακας με 10 τυχαίους αριθμούς

```
[random() for i in range(10)]
```

```
[0.11050328624094419,
 0.3565093135705997,
 0.015316127115346467,
 0.04269228263803637,
 0.3816442062718678,
 0.28519852876845597,
 0.5199427440819391,
 0.30993137310732854,
 0.034924334590274886,
 0.007696019131140153]
```

Τυχαίος ακέραιος αριθμός μεταξύ 50 και 80

```
randint(50,80)
```

```
74
```

Πίνακας με 10 τυχαίους ακεραίους αριθμούς μεταξύ 10 και 20

```
[randint(10,20) for i in range(10)]
```

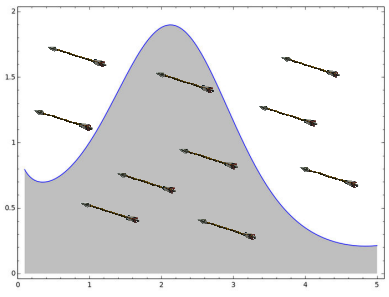
```
[16, 12, 15, 10, 16, 13, 17, 14, 20, 14]
```

Αριθμητικός υπολογισμός ολοκληρώματος με τη χρήση τυχαίων αριθμών

Ένα ολοκλήρωμα μπορεί να υπολογιστεί με τη χρήση τυχαίων αριθμών χωρίς καμμιά γνώση μεθόδων ολοκλήρωσης.
Η απλούστερη αυτού του τύπου μέθοδος βασίζεται στο γεγονός ότι το ολοκλήρωμα είναι ένα εμβαδόν το οποίο μπορούμε να εκτιμήσουμε αν το σχεδιάσουμε για παράδειγμα σε ένα κάδρο και ρίξουμε τυχαία σε αυτό βελάκια.
Το ολοκλήρωμα είναι ανάλογο του αριθμού από τα βελάκια που βρίσκονται στη σκιασμένη περιοχή προς το σύνολο αυτών που χρησιμοποιήσαμε. Ας δούμε μια εφαρμογή στο παρακάτω παράδειγμα

Παράδειγμα 1: Να υπολογιστεί το ολοκλήρωμα της συνάρτησης $f(x) = x^{\sin x}$ στην $0.1 < x < 5$

```
f(x)=x^sin(x)
plot(f,x,0.1,5,frame=True,axes=False)
```



```
pin=0
nop=25000
x1=0.1;x2=5.;y1=0.;y2=2.;
for i in range(nop):
    xx=x1+(x2-x1)*random()
    yy=y1+(y2-y1)*random()
    if yy<f(xx):
        pin+=1
print (x2-x1)*(y2-y1)*float(pin/nop)
```

```
4.76593600000000
```

Υπολογισμός χρησιμοποιώντας την αριθμητική ολοκλήρωση του SAGE

```
numerical_integral(x^sin(x),0.1,5.)
```

```
(4.773813346559158, 3.0854447345393065e-13)
```

Αριθμητικός υπολογισμός ολοκληρώματος με τη χρήση τυχαίων αριθμών

Ένας άλλος τρόπος υπολογισμού ολοκληρώματος με τη χρήση τυχαίων αριθμών βασίζεται στα παρακάτω.
Η μέση τιμή μιας συνάρτησης $f(x)$ σε ένα διάστημα $a < x < b$ ορίζεται ως

$$\bar{f} = \frac{1}{b-a}\int_a^b dx f(x) \Rightarrow \int_a^b dx f(x) = (b-a)\,\bar{f}$$

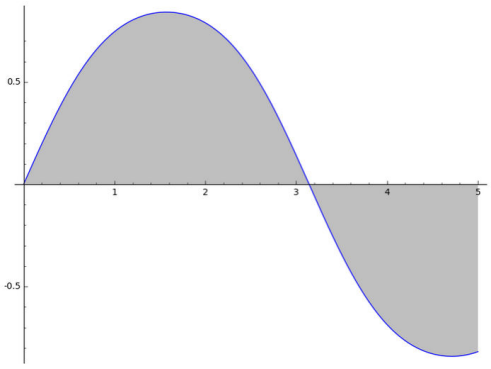
Για την εκτίμηση της μέσης τιμής παίρνουμε N τιμές της συνάρτησης σε τυχαία σημεία $x_1, x_2, \ldots, x_N$

$$\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Αριθμητικός υπολογισμός ...συνέχεια

Παράδειγμα 2: Να υπολογιστεί το ολοκλήρωμα της συνάρτησης $\mathrm{f(x)} = \sin(\sin x)$ **στην περιοχή** $\mathbf{0 < x < 5}$

```
plot(sin(sin(x)),x,0,5,fill=True)
```




```
f(x)=sin(sin(x))
nop=200000
sf=0.
x1=0.;x2=5.;
for i in range(nop):
    xx=x1+(x2-x1)*random()
    sf+=f(xx)
fm=sf/float(nop)
print fm*(x2-x1)
```

```
0.646797054094543
```

Υπολογισμός χρησιμοποιώντας την αριθμητική ολοκλήρωση του SAGE

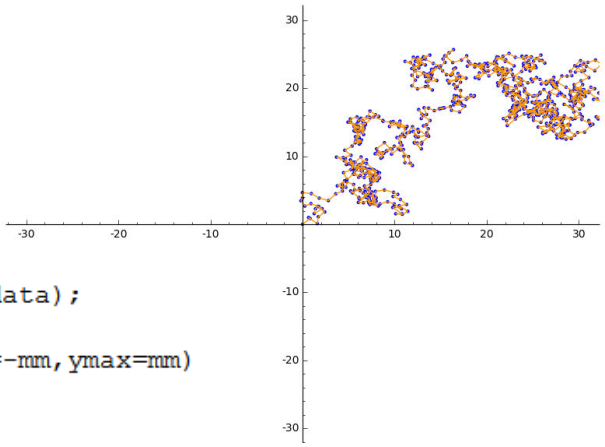
```
numerical_integral(f(x),x1,x2)
```

$$\left(0.653401711337, 3.23916970455 \times 10^{-14}\right)$$

Τυχαίος περίπατος

Ένας περιπατητής κινείται στο επίπεδο ως εξής: Ξεκινάει από τη θέση (0,0) και σε κάθε χρονική στιγμή κάνει ένα βήμα μήκους ℓ σε τυχαία κατεύθυνση. Σχεδιάστε την τροχιά του.

```
x=0;y=0
nop=300;l=1
data=[[x,y]]
for i in range(nop):
    th=n(2.0*pi*random())
    x,y=x+l*cos(th),y+l*sin(th)
    data.append([x,y])
```




```
g=line(data,hue=0.1)+points(data);
mm=int(sqrt(nop))+1
g.show(xmin=-mm,xmax=mm,ymin=-mm,ymax=mm)
```

Σειρές Fourier

Μια περιοδική συνάρτηση ορισμένη στην περιοχή $[-\mathrm{L}, \mathrm{L}]$, αναπτύσσεται σε σειρά Fourier ως εξής

$$f(x) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty}\left[a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)\right]$$

Παρακάτω αναπτύσσουμε τη συνάρτηση

$$f(x) = \begin{cases} +1 & -\pi < x < 0 \\ -1 & 0 < x < \pi \end{cases}$$

L

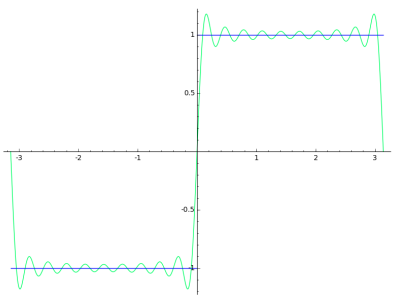
```
x=var('x');
f1(x)=-1.;f2(x)=1.
f =  Piecewise([[(-pi,0),f1],[(0,pi),f2]])
ft=f.fourier_series_partial_sum(20,pi);ft
```

```
0.2105263157894737*sin(19*x)/pi + 0.2352941176470588*sin(17*x)/pi +
0.2666666666666667*sin(15*x)/pi + 0.3076923076923077*sin(13*x)/pi +
0.3636363636363636*sin(11*x)/pi + 0.4444444444444444*sin(9*x)/pi +
0.5714285714285714*sin(7*x)/pi + 0.8*sin(5*x)/pi +
1.333333333333333*sin(3*x)/pi + 4.0*sin(x)/pi
```

μέγιστο n

Προσοχή στον τρόπο γραφικής αναπαράστασης της σειράς

```
plot(ft,x,-pi,pi,hue=0.4)+f.plot()
```



Αριθμητική εύρεση ρίζας εξίσωσης

Το SAGE διαθέτει μεθόδους οι οποίες υπολογίζουν αριθμητικά τις ρίζες συναρτήσεων. Αξίζει όμως να παρουσιάσουμε εδώ, ως εφαρμογή, την απλούστερη μέθοδο αριθμητικού υπολογισμού ριζών η οποία ονομάζεται μέθοδος Newton-Raphson. Η μέθοδος συνίσταται στην εύρεση μιας ρίζας της εξίσωσης $f(x) = 0$ ξεκινώντας από ένα αρχικό σημείο x_0 και προχωρώντας με διαδοχικά βήματα

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

έως ότου $|f(x)| < \epsilon$ όπου ϵ η επιθυμητή ακρίβεια

Παρακάτω εφαρμόζουμε για την εύρεση ρίζας της συνάρτησης $f(x) = e^{\cos x} + 2x^2$

```
f(x)=-exp(cos(x))+2*x^2
df=diff(f,x)
acc=10^-9
x0=-50.
x=x0
mstep=100
istep=0
for istep in range(mstep):
    x= x-f(x)/df(x)
    print istep+1,x , f(x)
    if abs(f(x))<acc:
        break
```

```
1 -24.9267895484845 1240.02823577110
2 -12.4218277977656 305.913529175461
3 -6.21666402618705 74.5815468477795
4 -3.19550045175173 20.0540320360537
5 -1.62413274127288 4.32752933339756
6 -1.04273093093250 0.519472648979054
7 -0.949977469622775 0.0158345741002841
8 -0.946964316999087 0.000017522924030455 3
9 -0.946960975172041 2.15887308030460e-11
```

Πρώτοι αριθμοί

Ένας φυσικός αριθμός, μεγαλύτερος της μονάδας, ονομάζεται πρώτος αν διαιρείται μόνο με τον εαυτό του και τη μονάδα.
Μπορούμε να ελέγξουμε αν ένας φυσικός αριθμός είναι πρώτος χρησιμοποιώντας τη συνάρτηση `is_prime(...)`

```
for p in range(0,100):
    if is_prime(p):
        print p
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```

Το SAGE όμως διαθέτει και εξειδικευμένες συναρτήσεις όπως η `prime_range(...)` ή την `prime_first_n(...)` η χρήση των οποίων φαίνεται στο παράδειγμα

```
prime_range(0,100)
```

$[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]$

```
prime_range(1821,1900)
```

$[1823, 1831, 1847, 1861, 1867, 1871, 1873, 1877, 1879, 1889]$

```
primes_first_n(20)
```

$[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71]$

Πρώτοι αριθμοί … συνέχεια

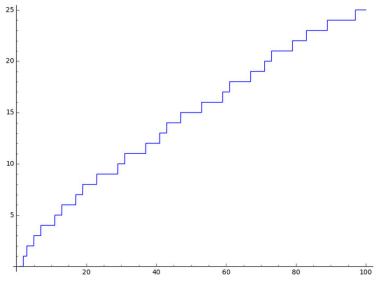
Η συνάρτηση `prime_pi(...)` αντιστοιχεί στη μαθηματική συνάρτηση $\pi(x)$ η τιμή της οποίας ισούται με το πλήθος των πρώτων που είναι ίσο ή μικρότερο του x.
Η συνάρτηση $\pi(x)$ δεν είναι γνωστή αναλυτικά αλλά για μεγάλα x προσεγγίζεται από το λογαριθμικό ολοκλήρωμα

$$\lim_{x\to+\infty} \pi(x) = \mathrm{li}(x) = \int_0^x \frac{dt}{\ln t}$$

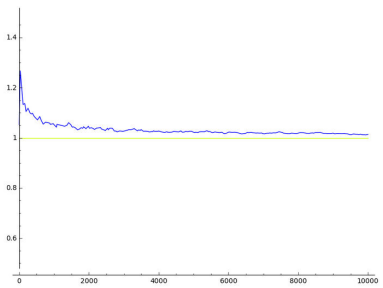
```
prime_pi(100)
```
25

```
prime_pi(10^6)
```
78498

```
plot(prime_pi(x),x,1,100)
```



```
plot(log_integral(x)/prime_pi(x),x,1,10000,\
    ymin=0.5,ymax=1.5)+ plot(1,x,0,10000,hue=0.2)
```



Φίλιοι Αριθμοί (Amical Numbers)

Δύο ακέραιοι n και m ονομάζονται φίλιοι (amical) όταν το άθροισμα των διαιρετών του ενός (εκτός του εαυτού του) ισούται με τον άλλο και αντιστρόφως. Το μικρότερο ζευγάρι φίλιων αριθμών είναι οι (220, 284). Οι διαιρέτες του 220 είναι οι 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 και 110, με άθροισμα 284 και οι διαιρέτες του 284 είναι οι 1, 2, 4, 71 και 142, με άθροισμα 220 . Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει του φίλιους αριθμούς μικρότερους των 6000.

Η παρακάτω συνάρτηση επιστρέφει μια λίστα με τους διαιρέτες ενός ακεραίου

```
def divs(k):
    e=[]
    for i in range(1,k):
        if k % i ==0:
            e.append(i)
    return e
```

```
m=6000;
sdivs=[0];
for i in range(1,m):
    sdivs.append(sum(divs(i)))
for i in range(2,m):
    for j in range(i+1,m):
        if(sdivs[i]==j and sdivs[j]==i):
            print i,j
```

Υπολογίζουμε μια φορά το άθροισμα των διαιρετών όλων των ακεραίων 1<m<6000

```
220 284
1184 1210
2620 2924
5020 5564
```

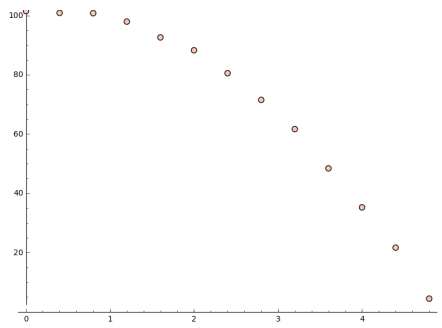
Εφαρμογές στη Φυσική

Προσαρμογή καμπύλης σε πειραματικά δεδομένα

Τα παρακάτω δεδομένα περιγράφουν τη θέση σώματος το οποίο εκτοξεύεται με ταχύτητα v_0 από ύψος h. Η επιτάχυνση της βαρύτητας είναι g. Με προσαρμογή στην εξίσωση κίνησης να υπολογιστούν τα v_0,h,g

```
data=[[0.000, 101.],
 [0.400, 101.],
 [0.800, 101.],
 [1.20, 97.9],
 [1.60, 92.6],
 [2.00, 88.3],
 [2.40, 80.5],
 [2.80, 71.5],
 [3.20, 61.7],
 [3.60, 48.4],
 [4.00, 35.2],
 [4.40, 21.7],
 [4.80, 4.44]]
```

```
scatter_plot(data)
```



Προσαρμογή καμπύλης … συνέχεια

```
x0,v0,g,c=var('x0 v0 g c');
model(t)=x0+v0*t+g*t^2/2+c*t^3
```

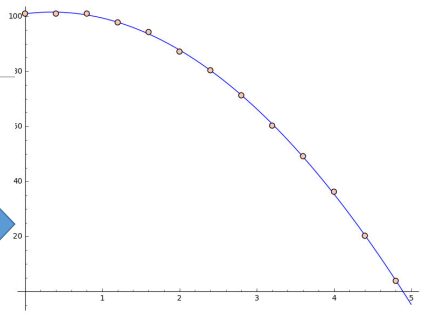
```
s=find_fit(data,model,parameters = [x0,v0,g,c], variables = [t],solution_dict=True);s
```

```
{x0: 101.12411230892106,
 g: -9.90407581618355,
 c: 0.024904485264741125,
 v0: 3.077526595421961}
```

```
f=model(t).subs(s);f
```

```
0.04880532940024988*t^3 - 5.190184605932377*t^2 + 3.620462553563161*t +
100.91054949271113
```

```
plot(f,t,0,5)+scatter_plot(data)
```



Προσαρμογή καμπύλης σε δεδομένα με τη χρήση βιβλιοθηκών της python

Η python προσφέρει τις δικές της ρουτίνες για την προσαρμογή καμπύλης σε πειραματικά δεδομένα. Μια από αυτές είναι η `curve_fit(..)` η οποία προσφέρει μερικά πλεονεκτήματα ως προς την αντίστοιχη συνάρτηση του SAGE όπως η εκτίμηση του τυπικού σφάλματος των παραμέτρων προσαρμογής και η δυνατότητα χειρισμού δεδομένων με σφάλματα.
Το παράδειγμα που δίνουμε παρακάτω μπορεί να χρησιμοποιηθεί και ως πρότυπο χρήσης βιβλιοθηκών python στο SAGE.

Πρόβλημα

Η ένταση σε φάσμα περίθλασης από απλή σχισμή πάχους a όταν φωτίζεται με φως μήκους κύματος λ δίνεται συναρτήσει της γωνίας θ από

$$I = I_0 \cos^2\left(\pi\, a\, \frac{\sin(\theta)}{\lambda}\right)$$

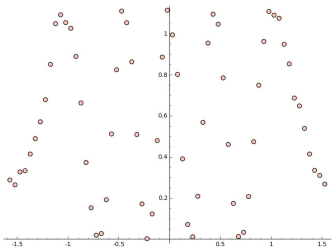
Για προσπίπτουσα ακτινοβολία μήκους κύματος $\lambda = 1500nm$ δίνονται δεδομένα μετρήσεων της έντασης συναρτήσει της γωνίας θ και ζητείται με προσαρμογή στην ανωτέρω φόρμουλα να προσδιοριστούν οι η ένταση I_0 και το πάχος της σχισμής a

Προσαρμογή καμπύλης…

Για χάριν του παραδείγματος παράγουμε ενδεικτικά δεδομένα για το $\mathrm{I}(\theta)$ με $\mathrm{I}_0 = 1.00, \alpha$=3500nm εισάγοντας αποκλίσεις με τη χρήση τυχαίων αριθμών

```
a=3500;la=1500;i0=1.;
data=[];xx=[];yy=[];
for t in sxrange(-pi/2,pi/2,0.05):
    d=pi*a*sin(t)/la
    data.append([n(t,digits=3),n(i0*cos(d)^2*(1+0.2*random()),digits=3)])
```

```
scatter_plot(data)
```



Εισαγωγή βιβλιοθήκης python και ορισμός δεδομένων

```
import numpy as np
from scipy.optimize import curve_fit

la=1500.;
xx=[data[i][0] for i in range(len(data))];
yy=[data[i][1] for i in range(len(data))];
xxx=np.array(xx);yyy=np.array(yy)
```

Προσαρμογή καμπύλης…

Ορισμός συνάρτησης προσαρμογής

```
def ffun(t,ii0,aa):
    return ii0*np.cos(np.pi*aa*np.sin(t)/la)^2

popt, pcov = curve_fit(ffun, xxx, yyy,p0=[2,4000])
iii0,aaa=popt
popt
```

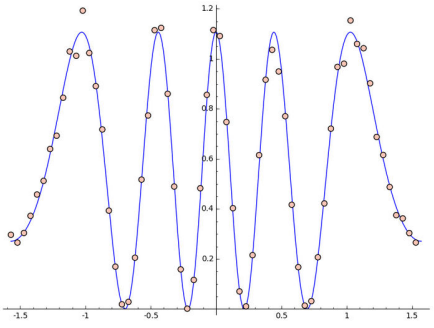
```
array([  1.10624392e+00,   3.50285325e+03])
```

Υπολογισμός τυπικού σφάλματος

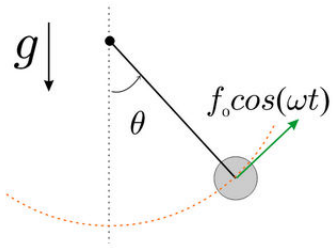
```
perr = np.sqrt(np.diag(pcov));perr
```

```
array([ 0.00743057,  3.50257886])
```

```
plot(ffun(x,iii0,aaa),x,-pi/2,pi/2)+scatter_plot(data)
```



Εξαναγκασμένο εκκρεμές με τριβή –Χαοτική συμπεριφορά



Θεωρούμε ένα εκρεμμές το οποίο αποτελείται από άκαμπτη αβαρή μπάρα μήκους ℓ στην άκρη της οποία έχει προσαρμοστεί σφαίρα μάζας m. Θεωρούμε επίσης ότι υπάρχει τριβή ανάλογη της γωνιακής ταχύτητας και ότι στη μάζα ενεργεί αρμονική δύναμη $f_0 \cos(\omega t)$ (κατά τη διεύθυνση της ταχύτητας). Η εξίσωση κίνησης του συστήματος συναρτήσει της γωνίας $\theta(t)$ δίνεται από

$$m\ell \frac{d^2\theta}{dt^2} + \gamma \frac{d\theta}{dt} = -mg \sin(x) + f_0 \cos(\omega t)$$

όπου g η επιτάχυνση της βαρύτητας και γ ο συντελεστής που σχετίζεται με την τριβή. Χρησιμοποιώντας διαστατική ανάλυση μπορούμε να θέσουμε $m = g = \ell = 1$ και το σύστημα χαρακτηρίζεται από τρεις παραμέτρους τις f_0, ω, γ. Θεωρούμε αρχικές συνθήκες $\theta(0) = \theta_0, \dot{\theta}(0) = \dot{\theta}_0$

Βλέπε επίσης: http://www.thphys.uni-heidelberg.de/~gasenzer/index.php?n1=teaching&n2=chaos

Εξαναγκασμένο εκκρεμές με τριβή…

```
gamma,f0,w=var('gamma f0 w');
t,x,th,dth=var('t x th dth');
gamma=0.2;f0=0   ;w=1;
ts=n(2.*pi/w/100.)
tt=srange(0.,900.,ts);
```

Ορίζουμε το βήμα ολοκλήρωσης ως το 1/100 της περιόδου της εξωτερικής δύναμης

```
def redu(u):
    o=u % n(2.*pi)
    if o > pi :
        return o-n(2.*pi)
    else:
        return o
```

Χρησιμοποιείται για να επαναφέρει τη γωνία του εκκρεμούς στο διάστημα $-\pi < \theta(t) < \pi$

$$\frac{d\theta}{dt} = \dot{\theta}$$

$$\frac{d\dot{\theta}}{dt} = -\gamma\dot{\theta} - \sin(\theta) + f_0 \cos(\omega t)$$

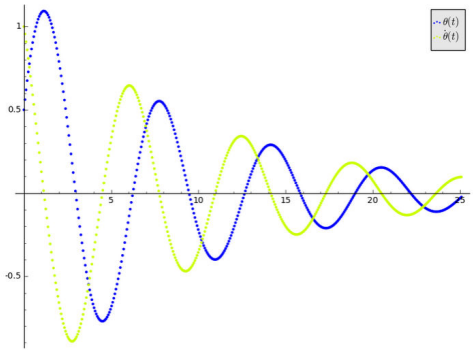
```
s=desolve_odeint([dth,-gamma*dth-sin(th)+f0*cos(w*t)],\
[0.5,1.],tt,[th,dth],ivar=t)
```

$\theta(0) = 0.5, \dot{\theta}(0) = 1.0,$

Εξαναγκασμένο εκκρεμές με τριβή…

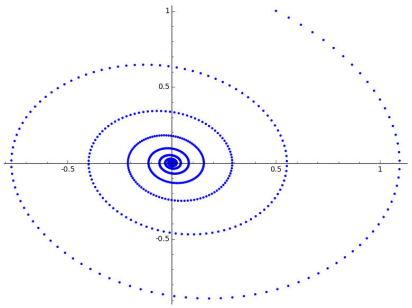
$\gamma = 1, \omega = 1, f_0 = 0$

```
g=points(zip(tt,map(redu,s[0:400,0])),legend_label=r'$\theta(t)$')+\
points(zip(tt,s[0:400,1]),hue=0.2,legend_label=r'$\dot\theta(t)$')
g.show()
```



Γραφική αναπαράσταση $\dot{\theta}(t), \theta(t)$

```
points(zip(map(redu,s[:,0]),s[:,1]))
```



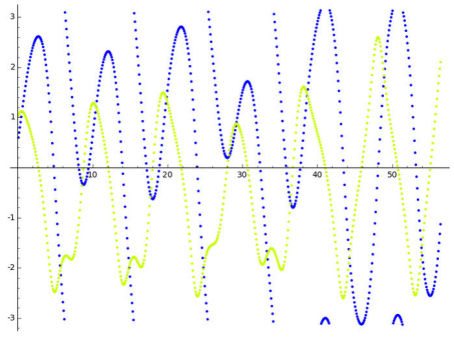
Χώρος φάσεων $\dot{\theta} \; - \theta$

Εξαναγκασμένο εκκρεμές με τριβή…

$\gamma = 0.5, \omega = 0.667, f_0 = 1.5$

```
gamma=0.5;f0=1.5  ;w=0.6667;
ts=n(2.*pi/w/100.)
tt=srange(0.,900.,ts);
s=desolve_odeint([dth,-gamma*dth-sin(th)+f0*cos(w*t)],[0.5,1.],tt,[th,dth],ivar=t)
```

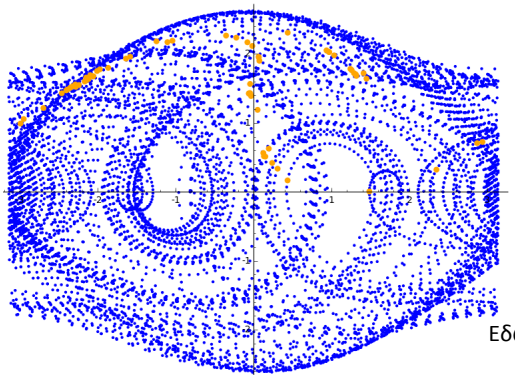
```
mm=zip(tt,map(redu,s[:,0]));mmd=zip(tt,s[:,1]);
points(mm[1:600])+points(mmd[1:600],hue=0.2)
```



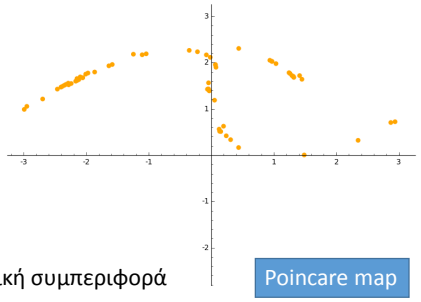
Εξαναγκασμένο εκκρεμές με τριβή…

```
poi=[[redu(s[i,0]),s[i,1]] for i in range(0,len(tt),100)];
pmap=points(poi[30:len(poi)],size=40,color='orange')
```

```
points(zip(map(redu,s[:,0]),s[:,1]))+pmap
```



```
pmap.show(xmin=-pi,xmax=pi,ymin=-pi,ymax=pi)
```



Εδώ έχουμε χαοτική συμπεριφορά

Poincare map

Βιβλιογραφία

Βιβλιογραφία

- Michael O'Sullivan, David Monarres, SSDU SAGE http://www-rohan.sdsu.edu/~mosulliv/Teaching/sdsu-sage-tutorial/index.html
- Ted Kosan, SAGE For Newbies, February, 2008, https://www.uam.es/personal_pdi/ciencias/pangulo/laboratorio/sage_for_newbies_v1.23.pdf
- Gregory V. Bard, Sage for Undergraduates, American Mathematical Society, 2014.
- David Joyner and Marshall Hampton, Introduction to Differential Equations Using Sage, Johns Hopkins University Press, 2012.
- A. Casamayou, N. Cohen, G. Connan, T. Dumont, L. Fousse, F. Maltey, M. Meulien, M. Mezzarobba, C. Pernet, N. M. Thiéry, P. Zimmermann , Calcul mathématique avec Sage, http://sagebook.gforge.inria.fr/

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Ιωαννίνων**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Έχουν προηγηθεί οι κάτωθι εκδόσεις:

- Έκδοση 1.0 διαθέσιμη εδώ. http://ecourse.uoi.gr/course/view.php?id=1240 .

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης