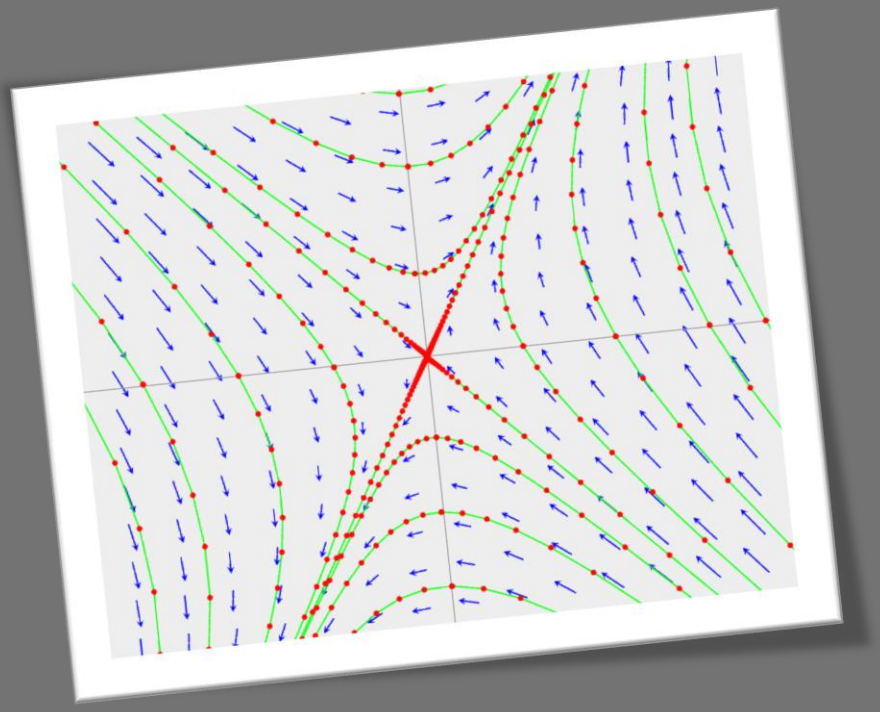
ΕΙΣΑΓΩΓΗ ΣΤΟ MAXIMA & ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΘΟΔΟΙ

ΕΦΑΡΜΟΓΕΣ ΑΠΟ ΤΗ ΔΥΝΑΜΙΚΗ ΣΥΣΤΗΜΑΤΩΝ



Φώτιος Κασόλης
Ελλάδα 2008

Εισαγωγή στο Maxima και αριθμητικές μεθόδοι

E-mail: fotios.kasolis@gmail.com

Περιεχόμενα

ΜΕΡΟΣ Α: Maxima

Α.1 ΕΙΣΑΓΩΓΗ

Το Maxima είναι στον πυρήνα του μία γραμμή εντολών και σαν τέτοια δεν είναι ικανό να παρουσιάσει τις μαθηματικές εκφράσεις παρά μόνο σε επίπεδο χαρακτήρων ascii. Ίσως κάτι τέτοιο να ακούγεται απογοητευτικό αλλά ας ελπίζουμε να σας εκπλήξουν σε τέτοιο βαθμό οι ικανότητες που δεν θα είναι παρατηρήσιμη μία αδυναμία αισθητικής φύσης. Με την εγκατάσταση του Maxima στα Windows έχουμε στη διάθεση μας δύο γραφικά περιβάλλοντα εργασίας (GUI). Το πρώτο (και πιο απλό) από αυτά ονομάζεται XMaxima και είναι αυτό που θα χρησιμοποιήσουμε.

Το παράθυρο του XMaxima χωρίζεται σε δύο μέρη. Το πρώτο μέρος αποτελεί τη γραμμή εντολών του Maxima – περιοχή εισαγωγής εντολών, ενώ το δεύτερο τμήμα παρέχει ένα σύντομο εγχειρίδιο.

Στη γραμμή εντολών, πέρα από τις πληροφορίες έκδοσης και άδειας, υπάρχει ο αριθμός εισόδου στη μορφή `(%i1)`. Αυτός είναι ο τρόπος να μας δείχνει το Maxima ότι είναι έτοιμο να δεχτεί είσοδο. Αναλογικά η έξοδος θα παρουσιάζεται αριθμημένη στην ίδια μορφή `(%o1)`. Όλα τα προγράμματα συμβολικής άλγεβρας έχουν κάποιους συντακτικούς κανόνες, μία δομημένη γλώσσα την οποία χρησιμοποιεί ο χρήστης για να επικοινωνίσει με το σύστημα του. Θεμελιώδους σημασίας στη γλώσσα του Maxima είναι οι τέσσερις βασικές πράξεις που τις δηλώνουμε με τα συνήθη σύμβολα `+`, `-`, `*`, `/`. Με μοναδική επιπλέον πληροφορία ότι κάθε στοιχείο εισόδου για να εκτελεστεί πρέπει να λήγει σε ελληνικό ερωτηματικό `;` και έπειτα να κτυπάμε το πλήκτρο `enter` (κάτι που θα δηλώνεται με το σύμβολο ↵) μπορούμε να εκτελέσουμε όλες τις γνωστές πράξεις. Ας υποθέσουμε οτι θέλουμε να εκτελέσουμε τον παρακάτω πολλαπλασιασμό:

```
(%i1) 2.279*11.126;↵
(%o1) 25.356154
(%i2)
```

Ενώ αν θέλουμε να εκτελεστεί ή «πράξη» αλλά να μην εκτυπωθεί το αποτέλεσμα της στην οθόνη αντικαθιστούμε το ερωτηματικό με το σύμβολο `$`, όπως στο ακόλουθο παράδειγμα:

```
(%i2) a+a$↵
(%i3)
```

Αρκετά όμως χρησιμοποιήσαμε το Maxima σαν υπολογιστή τσέπης, ας περάσουμε στην επόμενη παράγραφο όπου θα δούμε τις βασικές συναρτήσεις/εντολες (και γράψαμε «συναρτήσεις/εντολές» γιατί οι εντολές του Maxima δεν είναι παρά συναρτήσεις σαν αυτές που θα μάθουμε να ορίζουμε αργότερα κι εμείς). Τέλος δηλώνουμε ότι από εδώ και στο εξής δε θα σημειώνουμε το γεγονός οτι πατάμε το πλήκτρο `enter` για την εκτέλεση των εντολών, καθώς επίσης ούτε και την επόμενη είσοδο, είναι σαφές ότι το Maxima με τη λήξη της εκτέλεσης μίας εντολής είναι έτοιμο να δεχτεί την επόμενη.

Α.2 ΒΑΣΙΚΟΙ ΥΠΟΛΟΓΙΣΜΟΙ

Α.2.1 Απλοί αριθμητικοί υπολογισμοί

Δεν είναι μόνο οι βασικές πράξεις που έχουν το συνήθη συμβολισμό στο Maxima, ο συμβολισμός αρκετών συναρτήσεων, όπως για παράδειγμα των τριγωνομετρικών $\sin(x)$, $\cos(x)$, ... είναι επίσης ίδιος με αυτόν που έχουμε γνωρίσει σε μαθήματα μαθηματικών και φυσικής, έτσι έχουμε τις αντίστοιχες συναρτήσεις του Maxima `sin(x)`, `cos(x)`. Όσο για τις μαθηματικές σταθερές υπάρχει ο απλός κανόνας ότι αρχίζουν με το σύμβολο `%` και ακολουθεί η καθιερωμένη λατινική τους γραφή, παραδείγματος χάριν η μιγαδική μονάδα $i = \sqrt{-1}$ συμβολίζεται στο Maxima `%i`, ενώ το $\pi = 3.141592653589793\ldots$ γράφεται σαν `%pi`.

```
(%i1) %i*%i;
(%o1) -1
(%i2) %pi*%pi;
(%o2) %pi^2
```

Παρακάτω δίνουμε τις βασικές πράξεις, σταθερές και συναρτήσεις: `+, -, *, /, ^(ή **), %e, %pi, %i, exp(x), log(x), sin(x), cos(x), tan(x), sinh(x), cosh(x), tanh(x), asin(x), acos(x), atan(x), asinh(x), acosh(x), atanh(x), abs(x), sqrt(x), sign(x).`

Τώρα που πλέον γνωρίζουμε κάποιες συναρτήσεις μπορούμε να επιδοθούμε σε δοκιμές εκτελώντας υπολογισμούς.

```
(%i3) sin(0.4);
(%o3) 0.38941834230865
(%i4) log(1.5);
(%o4) 0.40546510810816
(%i5) cos(%pi/3);
(%o5) 1/2
(%i6) sin(2);
(%o6) sin(2)
```

όπου παρατηρούμε ότι όλα πήγαιναν καλά μέχρι τον υπολογισμό του `sin(2)` που το Maxima μας επέστρεψε την αντίστοιχη είσοδο. Το γιατί είναι πολύ απλό. Το Maxima επιστρέφει την είσοδο όταν δεν μπορεί να δώσει απάντηση ίδιας ακρίβειας με αυτήν της μεταβλητής της εισόδου. Αν αντί `sin(2)` γράψουμε `sin(2.0)` θα έχουμε:

```
(%i7) sin(2.0);
(%o7) 0.90929742682568
```

Ένας άλλος τρόπος να αναγκάσουμε το Maxima να μας δώσει την αριθμητική τιμή μίας οποιασδήποτε έκφρασης είναι οι εντολές `float(x)` και `bfloat(x)` των οποίων η χρήση γίνεται κατανοητή στα ακόλουθα παραδείγματα.

```
(%i8) float(sin(2));
(%o8) 0.90929742682568
(%i9) bfloat(sin(2));
(%o9) 9.092974268256817b-1
```

όπου η `bfloat` δίνει το αποτέλεσμα με την πλήρη ακρίβεια των 16 ψηφίων που χρησιμοποιεί η μηχανή. Αν ζητήσουμε την τιμή της μεταβλητής `fpprec` θα πάρουμε την απάντηση 16. Αυτή είναι η μεταβλητή που καθορίζει την ακρίβεια της `bfloat`, έτσι για παράδειγμα μπορούμε να ορίσουμε ακρίβεια 30 σημαντικών ψηφίων αλλάζοντας την τιμή της `fpprec` δίνοντας `fpprec:30`, όπου `:` (και όχι το ίσον `=`) είναι ο τελεστής ανάθεσης τιμής.

```
(%i10) fpprec:30$
(%i11) bfloat(%e);
(%o11) 2.71828182845904523536028747135b0
```

Θα ολοκληρώσουμε αυτή την παράγραφο σημειώνοντας ότι μπορούμε να αναφερθούμε στο αποτέλεσμα οποιασδήποτε εξόδου με χρήση του αύξοντα αριθμού της ως εξής:

```
(%i12) %o11;
(%o12) 2.71828182845904523536028747135b0
```

A.2.2 Όρια, παράγωγοι και ολοκληρώματα

Οι εντολές αυτής της παραγράφου προέρχονται απο το διαφορικό και ολοκληρωτικό λογισμό έχουν την απλή γενική σύνταξη:

`εντολή(έκφραση,μεταβλητή,τιμή,επιλογές)`

όπου το όρισμα της **τιμής** είναι μία ή δύο αριθμητικές τιμές χωρισμένες με κόμμα και σε κάποιες περιπτώσεις είναι προεραιτικό.

Όρια

Ο υπολογισμός του ορίου της συνάρτησης $f(x)$ όταν $x \to a$ γίνεται με την εντολή **limit(f(x),x,a)**, στην οποία μπορούμε να προσθέσουμε την επιλογή **plus** ή **minus** για τον υπολογισμό των πλευρικών ορίων.

```
(%i1) limit((3*x-2)/(9*x+7),x,inf);
(%o1) 1/3
(%i2) limit(1/(3+2^(1/x)),x,0);
(%o2) und
(%i3) limit(1/(3+2^(1/x)),x,0,plus);
(%o3) 0
(%i4) limit(1/(3+2^(1/x)),x,0,minus);
(%o4) 1/3
```

Στη δεύτερη είσοδο το Maxima απαντάει ότι το όριο δεν ορίζεται, γεγονός που εύκολα μπορούμε να ελέγξουμε υπολογίζοντας τα πλευριά όρια. Ένας τέτοιος υπολογισμός μας πείθει μιας και

γνωρίζουμε ότι όταν $\lim_{x\to a+} f(x) \neq \lim_{x\to a-} f(x)$ το όριο $\lim_{x\to a} f(x)$ δεν ορίζεται.

Παράγωγοι

Για να παραγωγίσουμε μία συνάρτηση $f(x)$ χρησιμοποιούμε την εντολή `diff(f(x),x,k)` όπου `k` η τάξη της παραγώγισης, ενώ όταν `k=1` μπορούμε να το παραλείψουμε.

```
(%i5) sin(x)*cos(x);
(%o5) cos(x)sin(x)
(%i6) diff(%,x);
(%o6) cos^2(x)-sin^2(x)
(%i7) diff(%o1,x,2);
(%o7) -4cos(x)sin(x)
```

Επίσης με την ίδια εντολή μπορούμε να υπολογίσουμε τη μικτή παράγωγο $\frac{\partial^n}{\partial x^n}\frac{\partial^k}{\partial y^k}$ όπως στο ακόλουθο παράδειγμα:

```
(%i8) diff(log(x^y),x,2,y,1);
(%o8) -1/x^2
(%i9) diff(diff(log(x^y),y),x,2);
(%o9) -1/x^2
```

όπου εκτελείται η παραγώγιση $\frac{\partial^2}{\partial x^2}\frac{\partial}{\partial y}\ln(x^y) = \frac{\partial^2}{\partial x^2}\left(\frac{\ln(x)\cdot e^{y\cdot ln(x)}}{e^{y\cdot ln(x)}}\right) = \frac{\partial}{\partial x}\left(\frac{1}{x}\right) = -\frac{1}{x^2}$.

Ολοκληρώματα

Τέλος ο υπολογισμός του αόριστου και του ορισμένου ολοκληρώματος της συνάρτησης $f(x)$ γίνεται αντίστοιχα με τις εντολές `integrate(f(x),x)` και `integrate(f(x),x,x0,xf)` όπως δείχνει το παρακάτω παραδείγμα.

```
(%i10) integrate(m^2*exp(-a*m),m);
(%o10) -((a^2*m^2+2*a*m+2)%e^(-a*m))/a^3
```

Υπάρχουν σίγουρα και ολοκληρώματα των οποίων ο υπολογισμός γίνεται πιο πολύπλοκος όπως για παράδειγμα αυτός του $\int_0^1 \sin(x^2)dx$:

```
(%i11) float(integrate(sin(x^2),x,0,1));
(%o11) 0.22155673136319((1.414213562373095%i+
       1.414213562373095)erf(0.5(1.414213562373095%i+
       1.414213562373095))+(1.414213562373095%i-
       1.414213562373095)erf(0.5(1.414213562373095%i-
       1.414213562373095)))
```

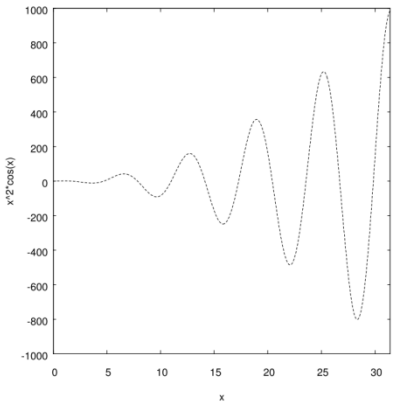
που όπως βλέπουμε το αποτέλεσμα δίνεται σα συνάρτηση της συνάρτησης σφάλματος `erf(x)`. Αυτό που είναι γνωστό για τη συνάρτηση αυτή είναι ότι η παράγωγος της είναι `2*exp(-x^2)/sqrt(%pi)`.

A.2.3 Εισαγωγή στα γραφικά

Και ερχόμαστε τώρα στις εντολές σχεδιασμού γραφικών παραστάσεων. Πριν όμως ας σημειώσουμε ότι το Maxima σχεδιάζει με χρήση του gnuplot ή του openmath και αυτό το κάνει να εμπεριέχει όλα τα πλεονεκτήματα και μειονεκτήματα αυτών των σχεδιστικών πακέτων.

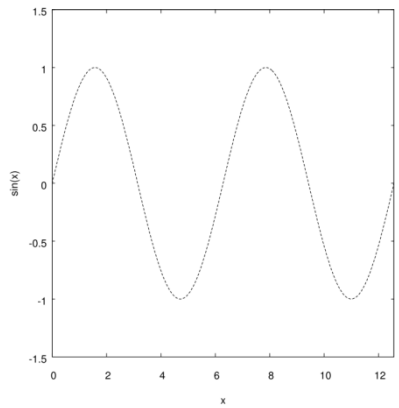
Η εντολή για την παραγωγή 2D γραφικών είναι η `plot2d` και στην πιο απλή της εκδοχή `plot2d(f(x),[x,x0,xf])`, η οποία δίνει τη γραφική παράσταση της συνάρτησης $f(x)$ στο διάστημα $[x0, xf]$.

```
(%i1) plot2d(x^2*cos(x),[x,0,10*%pi]);
```

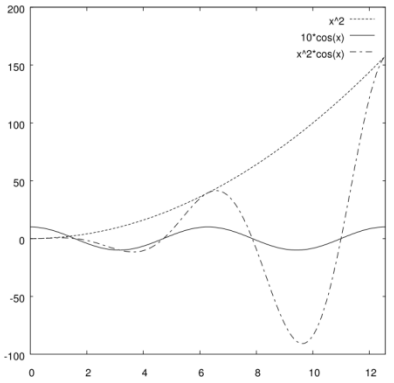


που όπως βλέπουμε στο σχήμα οι άξονες του γραφήματος ονομάζονται αυτόματα. Όμως η εντολή `plot2d` όπως και πολλές άλλες εντολές επιδέχονται μία σειρά απο προσθήκες στη βασική τους σύνταξη. Έτσι για παράδειγμα μπορούμε να ορίσουμε το κατακόρυφο διάστημα που θα παραχθεί στην έξοδο και αυτό επιτυγχάνεται με την προσθήκη `[y,a,b]` στην εντολή μας.

```
(%i2) plot2d(sin(x),[x,0,4*%pi],[y,-1.5,1.5]);
```

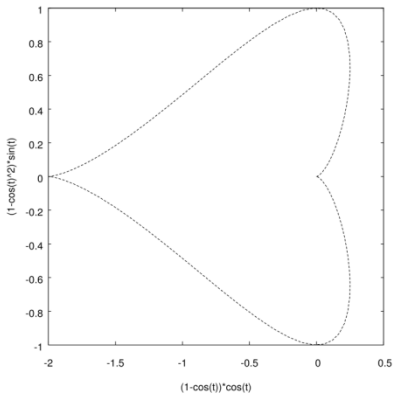


Μία άλλη δυνατότητα που θα θέλαμε να έχουμε είναι αυτή του σχεδιασμού δύο ή και παραπάνω γραφικών παραστάσεων στο ίδιο σύστημα αξόνων. Αυτό επιτυγχάνεται στο Maxima ομαδοποιώντας με αγκύλες τις συναρτήσεις που θέλουμε να σχεδιάσουμε (δημιουργώντας δηλαδή μία λίστα).




```
(%i3) plot2d([x^2,10*cos(x),x^2*cos(x)],[x,0,4*%pi]);
```

Τέλος μπορούμε να σχεδιάσουμε καμπύλες που δίνονται σε παραμετρική μορφή $x = f(t),\ y = g(t)$ με την εντολή `plot2d([parametric, f(t),g(t),[t,a,b]])`.




```
(%i4) plot2d([parametric,cos(t)*(1-cos(t)),sin(t)*(1-
      cos(t)^2),[t,0,2*%pi],[nticks,80]]);
```

όπου επιπλέον χρησιμοποιήσαμε το όρισμα `[nticks,80]` που καθορίζει τον αριθμό των σημείων που θα χρησιμοποιηθούν κατά το σχεδιασμό της γραφικής παράστασης.

Α.2.4 Ορισμός συναρτήσεων

Όπως έχουμε είδη αναφέρει ο τελεστής ανάθεσης τιμών είναι `:`, έτσι μπορούμε να δώσουμε ένα όνομα σε μία σταθερά ή σε ολόκληρες εκφράσεις που μπορεί να περιέχουν και μεταβλητές. Με αυτό τον τρόπο μπορούμε να καλούμε αυτές τις εκφράσεις χωρίς να είμαστε υποχρεωμένοι να τις ξαναγράφουμε. Για τον ορισμό μίας σταθεράς αρκεί αριστερά απο τον τελεστή `:` να δώσουμε το όνομα της αρεσκείας μας και δεξιά την ποσότητα που επιθυμούμε.

```
(%i1) x:9$
(%i2) sqrt(x);
(%o2) 3
```

Στην περίπτωση που θέλουμε να αποδεσμεύσουμε το όνομα απο την ποσότητα που του έχουμε αναθέσει δεν έχουμε παρά να πληκτρολογήσουμε:

```
(%i3) remvalue(x)$
(%i4) x;
(%o4) x
```

Επίσης μπορούμε να καλέσουμε την εντολή **remvalue** με όρισμα **all** ώστε να αποδεσμεύσουμε όλα τα ονόματα απο τις τιμές ή τις εκφράσεις που τους έχουμε αποδώσει.

```
(%i5) x:x+1$
(%i6) y:2$
(%i7) z:x+1$
(%i8) remvalue(all)$
(%i9) x;y;z;
(%o9) x, y, z
```

Συναρτήσεις

Ο ορισμός συναρτήσεων γίνεται με τον τελεστή **:=** και χρησιμοποιώντας παρενθέσεις για τις μεταβλητές:

$$f(x) = \acute{\epsilon}\kappa\varphi\rho\alpha\sigma\eta \rightarrow \texttt{f(x):=έκφραση}$$

Έχοντας ορίσει μία συνάρτηση μπορούμε να υπολογίσουμε την τιμή της σε οποιοδήποτε σημείο του πεδίου ορισμού της, να υπολογίσουμε τη σύνθεση με τον εαυτό της ή και με άλλες συναρτήσεις, να την παραγωγίσουμε κ.ο.κ. Θα φανεί αργότερα από το κείμενο η σημασία των συναρτήσεων, αλλά τώρα μπορούμε να αναφέρουμε τη σημαντικότητα τους στο προγραμματιστικό περιβάλλον του Maxima. Ακολουθούν μερικά απλά παραδείγματα.

```
(%i10) f(x):=x^2-4*x+3$
(%i11) f(5);
(%o11) 8
(%i12) f(a-b);
(%o12) (a-b)^2-4*(a-b)+3
(%i13) f(f(x));
(%o13) (x^2-4*x+3)^2-4*(x^2-4*x+3)+3
(%i14) diff(f(x),x);
(%o14) 2*x-4
```

Ο καθαρισμός της μνήμης από τις συναρτήσεις γίνεται με την εντολή **remfunction** η οποία, όπως και η **remvalue**, μπορεί να καλεστεί με όρισμα την επιλογή **all**.

```
(%i15) remfunction(f)$
(%i16) remfunction(all)$
```

Πρέπει να είμαστε προσεκτικοί όταν ζητάμε την τιμή μίας συνάρτησης σε κάποιο x. Στο ακόλουθο παράδειγμα δείχνουμε πως μπορεί κάποιος να απορεί για το αποτέλεσμα του.

```
(%i17) x:3$
⋮
(%i25) f(x):=x^2;
(%o25) f(x):=x^2
(%i26) f(x);
(%o26) 9
```

όπου παρατηρούμε ότι η συνάρτηση ορίζεται σωστά παρα το ότι έχουμε δώσει την τιμή 3 στο x, αλλά όταν ζητήσουμε την τιμή της στο x είναι σα να ζητάμε την $f(3) = 9$.

Κλείνουμε αυτή την παράγραφο με την εντολή `block` της οποίας το όρισμα είναι μία σειρά από εντολές χωρισμένες με κόμμα. Η `block` επιστρέφει την τιμή της τελευταίας έκφρασης αν δεν ζητηθεί κάτι άλλο ενώ αργότερα θα δούμε ότι συναρτήσεις της μορφής `f(x,y,z,…):=block([a,b,…],…,…,…)` όπου το πρώτο όρισμα της `block` είναι μία λίστα των τοπικών μεταβλητών, αποτελούν τον πυρήνα των προγραμμάτων που θα κατασκευάσουμε.

A.2.5 Αλγεβρικές εξισώσεις

Σε αυτή την παράγραφο θα ασχοληθούμε με την αναλυτική και αριθμητική λύση αλγεβρικών εξισώσεων, πριν από αυτό όμως είναι απαραίτητο να αναφερθούμε σε μερικές βασικές εντολές για την παραγοντοποίηση ή ανάπτυξη αλγεβρικών και τριγωνομετρικών εκφράσεων.

- `factor(f(x))`: Παραγοντοποίηση πολυωνύμων

```
(%i1) f(x):=x^2+x-6$
(%i2) factor(f(x));
(%o2) (x-2)(x+3)
```

- `expand(f(x))`: Ανάπτυξη πολυωνύμων

```
(%i3) f(x):=(x+3)^3$
(%i4) expand(f(x));
(%o4) x^3+9*x^2+27*x+27
```

- `ratsimp(f(x))`: Απλοποίηση ρητών εκφράσεων

```
(%i5) f(x):=(x^2-1)/(x+1)$
(%i6) ratsimp(f(x));
(%o6) x-1
```

- `trigepand(f(x))`: Ανάπτυξη τριγωνομετρικών εκφράσεων

```
(%i7) f(x):=sin(2*x)+cos(2*x)$
(%i8) trigexpand(f(x));
(%o8) -sin(x)^2+2*cos(x)sin(x)+cos(x)^2
```

- `trigsimp(f(x))`:Απλοποίηση τριγωνομετρικών εκφράσεων

```
(%i9)  f(x):=2*cos(x)^2+sin(x)^2$
(%i10) trigsimp(f(x));
(%o10) cos(x)^2+1
```

Η λύση αλγεβρικών εξισώσεων επιτυγχάνεται με τις εντολές `solve(f(x))`, `allroots(f(x))` και `find_root(f(x))`. Η πρώτη απο αυτές αναζητά ακριβείς λύσεις, η δεύτερη υπολογίζει αριθμητικά τις ρίζες πολυωνυμικών εξισώσεων και η τελευταία υπολογίζει επίσης αριθμητικά μία ρίζα οποιασδήποτε εξίσωσης. Οι εξισώσεις μπορούν να γραφούν είτε σαν $f(x) = g(x)$ είτε απλά σαν $f(x) - g(x)$ όπου το Maxima θα υποθέσει ότι η ζητούμενη εξίσωση είναι η $f(x) - g(x) = 0$.

```
(%i11) solve(x^3-3*x^2-x+3,x);
(%o11) [x=1,x=-1,x=3]
(%i12) allroots(x^3+3*x+1);
(%o12) [x=-0.32218535462609,
        x=1.754380959783722%i+0.16109267731304,
        x=0.16109267731304-1.754380959783722%i]
(%i13) find_root(cos(x)=x,x,0,1);
(%o13) 0.73908513321516
```

Αν μία μη πολυωνυμική εξίσωση $f(x) = 0$ έχει περισσότερες της μία ρίζες, αυτές πρέπει να αναζητηθούν μία προς μία με την `find_root` επιλέγοντας κατάλληλο διάστημα αναζήτησης. Μπορούμε να σχεδιάσουμε την $f(x)$ ώστε να προσδιορίσουμε γραφικά αυτά τα διαστήματα. Τέλος μπορούμε να χρησιμοποιήσουμε την `find_root` για την εύρεση των ακρότατων μίας συνάρτησης $f(x)$ ως ακολούθως `find_root(diff(f(x),x),x,a,b)`.

Α.2.6 Διαφορικές εξισώσεις και αριθμητική ολοκλήρωση

Η ακριβης επίλυση μίας διαφορικής εξίσωσης 1ης ή και 2ης τάξης γίνεται με τις `ode2, ic1, ic2`. Η πρώτη λύνει τη διαφορική εξίσωση και βρίσκει τη γενική λύση ενώ οι άλλες δύο λύνουν το πρόβλημα αρχικών τιμών ανάλογα με την τάξη της διαφορικής εξίσωσης (1ης `ic1`, 2ης `ic2`) και αφού πρώτα έχουμε βρει τη γενική λύση με χρήση της `ode2`. Για τη λύση γραμμικών συστημάτων διαφορικών εξισώσεων με χρήση μετασχηματισμών Laplace υπάρχει η εντολή `desolve`.

```
(%i1) x^2*'diff(y,x)+3*x*y=sin(x)/x;
(%i2) ode2(%,y,x)$
(%o2) y=(%c-cos(x))/x^3
(%i3) ic1(%,x=%pi,y=0);
(%o3) y=-(1+cos(x))/x^3
```

όπου ο τελεστής ' αποτρέπει τον υπολογισμό της έκφρασης που ακολουθεί, σε αυτή την περίπτωση της παραγώγου. Ακολουθεί ένα παράδειγμα της `desolve`.

```
(%i4) eq1:'diff(f(x),x)='diff(g(x),x)+sin(x)$
(%i5) eq2:'diff(g(x),x,2)='diff(f(x),x)-cos(x)$
(%i6) atvalue('diff(g(x),x),x=0,a)$
(%i7) atvalue(f(x),x=0,1)$
(%i8) desolve([eq1,eq2],[f(x),g(x)]);
(%o8) [f(x)=a%e^x-a+1,g(x)=cos(x)+a%e^x-a+g(0)-1]
```

όπου παρατηρούμε ότι η μεταβλητή πρέπει να δηλώνεται στις συναρτήσεις και στις παραγώγους αυτών. Η εντολή `atvalue(f(x),x=x0,a)` αναθέτει την τιμή `a` στη συνάρτηση `f(x)` όταν `x=x0`, δηλαδή `f(x0)=a`.

Όσο αφορά την αριθμητική λύση διαφορικών εξισώσεων το Maxima παρέχει μία Runge-Kutta 4ης τάξης η οποία είναι μέρος του πακέτου της δυναμικής και θα μελετηθεί αργότερα. Εδώ θα συνεχίσουμε με την αριθμητική ολοκλήρωση συναρτήσεων. Η μέθοδος που μας παρέχει το Maxima είναι μία Gauss-Kronrod της οποίας την τάξη επιλέγουμε εμείς (1ης εως 6ης). Η υλοποίηση της μεθόδου γίνεται με την εντολή `quad_qag(f(x),x,x0,xf,τάξη)` της οποίας η χρήση φαίνεται στο παρακάτω παράδειγμα.

```
(%i9) quad_qag(sqrt(x)*log(1/x),x,0,1,3);
(%o9) [0.44444444444921,3.1700968509523805E-9,961,0]
```

όπου το Maxima μας επέστρεψε μία λίστα με τα εξής στοιχεία:

- την αριθμητική τιμή του ολοκληρώματος,
- το απόλυτο σφάλμα,
- τον αριθμό των υπολογισθέντων συναρτήσεων και
- έναν κωδικό σφάλματος (0:δεν υπήρξαν προβλήματα, 1:μεγάλο πλήθος διαστημάτων, 2:μεγάλο σφάλμα στρογγυλοποίησης, 3:η συνάρτηση προς ολοκλήρωση παρουσιάζει «κακή» συμπεριφορά , 6: σφάλμα στην είσοδο).

Α.2.7 Λίστες στοιχείων

Σε αυτή την παράγραφο θα ασχοληθούμε με την παραγωγή στοιχείων που ονομάζονται λίστες. Κάθε λίστα αποτελείται από έναν αριθμό στοιχείων εγκλεισμένα σε αγκύλες που αριθμούνται με δείκτη

$$[s_1, s_2, \dots, s_n] \rightarrow \texttt{[s[1],s[2],…,s[3]]}$$

Αν γυρίσετε πίσω στο κείμενο θα διαπυστώσετε ότι σε πολλές περιπτώσεις οι έξοδοι δόθηκαν από το Maxima σαν λίστες στοιχείων. Μπορούμε να ορίσουμε λίστες ως εξής:

```
(%i1) s:[a,b,c,d];
(%o1) [a,b,c,d]
```

καθώς και να καλέσουμε οποιοδήποτε στοιχείο της λίστας γράφοντας τον αύξοντα αριθμό του μέσα σε αγκύλες όπως παρακάτω:

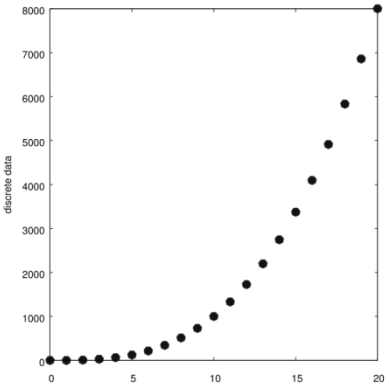
```
(%i2) s[3];
(%o2) c
```

Πέρα όμως απο αυτές τις λίστες, των οποίων τα στοιχεία έχουν οριστεί ένα προς ένα απο το χρήστη μπορούμε να παράγουμε λίστες των οποίων τα στοιχεία υπακούουν σε κάποιον κανόνα με την εντολή `makelist(f(n),n,n0,nf)`. Ο κανόνας ορίζεται απο τη συνάρτηση $f(n)$ και το αποτέλεσμα της εντολής είναι η λίστα $[f(n0), f(n0+1), \dots, f(nf)]$.

```
(%i3) s:makelist(n!,n,1,10);
(%o3) [1,2,6,24,120,720,5040,40320,362880,3628800]
```

όπου το `n` πηγαίνει απο το ένα μέχρι το δέκα με βήμα ένα. Φυσικά μπορούμε να σχεδιάσουμε τα στοιχεία μίας λίστας σαν συνάρτηση του αύξοντα αριθμού των στοιχείων της.

```
(%i4) x:makelist(n,n,0,20)$
(%i5) y:makelist(n^3,n,0,20)$
(%i6) plot2d([discrete,x,y],[style,points]);
```



Είναι σχεδόν αυτονόητο ότι μπορούμε να παράγουμε λίστες με στοιχεία άλλες λίστες, παραδείγματος χάριν με την εντολή `s:makelist([f(n),g(n)],n,n0,nf)`. Σε αυτή την περίπτωση η `s[i][j]` επιστρέφει το `j` στοιχείο του `i` στοιχείου λίστας. Επίσης η `length(λίστα)` επιστρέφει τον αριθμό των στοιχείων μιας λίστας, ενώ η `append(s1,s2,…,sn)` επιστρέφει μία λίστα με τα στοιχεία της λίστας `s1` ακολουθούμενα απο τα στοιχεία της `s2` κ.ο.κ. Προσέξτε ότι για την προσθήκη ενός μόνο στοιχείου `q` σε μία λίστα `s` πρέπει να γράψουμε `append(s,[q])`.

A.2.8 Ορισμός πινάκων – Βασικές πράξεις

Η άλγεβρα πινάκων είναι από τα πιο σημαντικά μέρη στην εκπαίδευση ενός εφαρμοσμένου επιστήμονα (φυσικού,μηχανικού κ.ο.κ.) και σαν τέτοια δε θα μπορούσε να λείπει απο ένα πακέτο συμβολικών μαθηματικών.

Το Maxima παρέχει την εντολή matrix για τον ορισμό πινάκων, της οποίας η σύνταξη είναι `matrix([γραμμή1],[γραμμή2],…,[γραμμήn])`, όπου όπως παρατηρούμε κάθε γραμμή είναι μία λίστα στοιχείων.

```
(%i1) A:matrix([1,2],[4,9]);
(%o1)
                              [1  2]
                              [    ]
                              [4  9]
```

Οι πράξεις **+, -, *, /** και ^ εκτελούνται στοιχείο προς στοιχείο, ο συνήθης μη μεταθετικός πολλαπλασιασμός πινάκων συμβολίζεται με μία τελεία **.** μεταξύ των πινάκων, ενώ τέλος η μη μεταθετική δύναμη συμβολίζεται με ^^.

```
(%i2) B:matrix([1,3],[2,2])$
(%i3) A+B;
(%o3)
                          [2   5]
                          [     ]
                          [6  11]
(%i4) A-B;
(%o4)
                          [0  -1]
                          [     ]
                          [2   7]
(%i5) A*B;
(%o5)
                          [1   6]
                          [     ]
                          [8  18]
(%i6) A.B;
(%o6)
                         [5    7]
                         [      ]
                         [22  30]
(%i7) A/B;
(%o7)
                         [1  2/3]
                         [      ]
                         [2  9/2]
(%i8) A^B;
(%o8)
                                [1  3]
                                [    ]
                                [2  2]
                          [1  2]
                          [    ]
                          [4  9]
(%i9) B^2;
(%o9)
                          [1  9]
                          [    ]
                          [4  4]
```

```
(%i10) A^^-1;
(%o10)
                          [9   -2]
                          [      ]
                          [-4   1]
```

όπου στην (%o10) έχουμε τον αντίστροφο του πίνακα A. Τέλος να σημειώσουμε ότι όταν έχουμε βαθμωτή βάση υψωμένη σε δύναμη πίνακα οι τελεστές ^ και ^^ δίνουν το ίδιο αποτέλεσμα δράσης στοιχείο προς στοιχείο.

A.2.9 Δημιουργία και χειρισμός πινάκων

Οι εντολές που θα δούμε σε αυτή την παράγραφο είναι οι: entermatrix, zeromatrix, ident, diagmatrix, row, col, addrow, addcol, invert, transpose, determinant, charpoly, eigenvalues και eigenvectors. Η «πράξη» που εκτελεί η κάθε εντολή είναι σχεδόν προφανής, αυτό που θα δούμε με χρήση παραδειγμάτων είναι η σύνταξη τους.

Η entermatrix είναι η διαδραστική ισοδύναμη της matrix, συνεπώς σκοπό έχει τη δημιουργία πινάκων. Δέχεται σαν όρισμα τη διάσταση του πίνακα που θέλουμε να δημιουργίσουμε και έπειτα αλληλεπιδρά με το χρήστη ώστε να του επιστρέψει τον πίνακα.

```
(%i1) A:entermatrix(2,2);
Is the matrix 1.Diagonal 2.Symmetric 3.Antisymmetric
4.General
Answer 1, 2, 3 or 4 :
1;
Row 1 Column 1:
2;
Row 2 Column 2:
2;
Matrix entered.
(%o1)
                          [ 2  0 ]
                          [      ]
                          [ 0  2 ]
```

Η εντολή **zeromatrix** δέχεται σαν όρισμα τον αριθμό γραμμών και στηλών ώστε να επιστρέψει έναν πίνακα αυτών των διαστάσεων με όλα τα στοιχεία του μηδενικά.

```
(%i2) zeromatrix(2,4);
(%o2)
                        [0 0 0 0]
                        [       ]
                        [0 0 0 0]
```

Η **ident** επιστρέφει το μοναδιαίο $n \times n$ πίνακα, συνεπώς:

```
(%i3) ident(2);
(%o3)
                         [ 1  0 ]
                         [      ]
                         [ 0  1 ]
```

Κλείνουμε τις εντολές κατασκευής πινάκων με την **diagmatrix(n,x)**, η οποία επιστρέφει τον $n \times n$ διαγώνιο πίνακα του οποίου τα στοιχεία είναι όλα **x**. Έχουμε δηλαδή ότι **ident(n)=diagmatrix(n,1)**.

```
(%i4) diagmatrix(2,x^3);
(%o4)
                         [x^3  0]
                         [      ]
                         [0  x^3]
```

Η εξαγωγή γραμμών ή στηλών ενός πίνακα M γίνεται με τις εντολές **row(M,n)** και **col(M,n)** των οποίων τα στοιχεία επιστροφής είναι πίνακες.

```
(%i5) M:matrix([a,b],[c,d])$
(%i6) row(M,2);
(%o6)
                           [c  d]
(%i7) col(M,1);
(%o7)
                           [ a ]
                           [   ]
                           [ c ]
```

Η προσθήκη γραμμών ή στηλών γίνεται με τις εντολές `addrow(M,λίστα1,λίστα2,...)` και `addcol(M,λίστα1,λίστα2,...)` αντίστοιχα.

```
(%i8) addrow(M,[e,f]);
(%o8)
                        [ a  b ]
                        [      ]
                        [ c  d ]
                        [      ]
                        [ e  f ]
(%i9) addcol(M,[e,f]);
(%o9)
                       [ a  b  e ]
                       [         ]
                       [ c  d  f ]
```

Οι εντολές `invert`, `transpose` και `determinant` δέχονται σαν όρισμα έναν πίνακα M και επιστρέφουν τον αντίστροφο, τον πίνακα που προκύπτει αν κάνουμε τις γραμμές στήλες και την ορίζουσα του πίνακα M αντίστοιχα.

```
(%i10) invert(M);
(%o10)
               [      d              b     ]
               [  ---------   - --------- ]
               [  a d - b c     a d - b c ]
               [                          ]
               [       c           a      ]
               [ - ---------   ---------  ]
               [   a d - b c   a d - b c  ]
(%i11) transpose(M);
(%o11)
                         [a   c]
                         [     ]
                         [b   d]
(%i12) determinant(M);
(%o12) ad-bc
```

Η εντολή `charpoly` επιστρέφει άλυτο το χαρακτηριστικό πολυώνυμο ενός πίνακα A. Φυσικά το χαρακτηριστικό πολυώνυμο

μπορεί έπειτα να λυθεί με μία απο τις εντολές επίλυσης αλγεβρικών εξισώσεων.

```
(%i13) A:matrix([3,1],[2,4])$
(%i14) expand(charpoly(A,lambda));
(%o14) lambda^2-7lambda + 10
(%i15) sol:solve(%);
(%o15) [lambda=5,lambda=2]
(%i16) sol[1];sol[2];
(%o16) lambda=5
(%o17) lambda=2
```

Οι εντολές **eigenvalues** και **eigenvectors** δέχονται σαν όρισμα έναν πίνακα του οποίου επιστρέφουν τις ιδιοτιμές και τα ιδιοδιανύσματα όπως στο ακόλουθο παράδειγμα:

```
(%i18) eigenvalues(A);
(%o18) [[5,2],[1,1]]
(%i19) eigenvectors(A);
(%o19) [[[5,2],[1, 1]],[1, 2],[1,-1]]
```

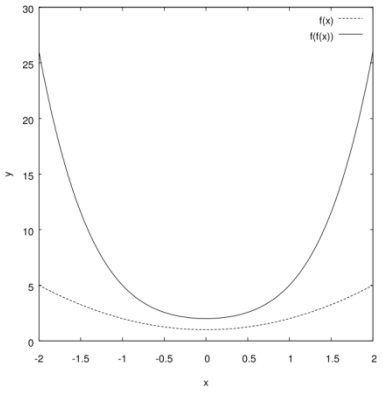
Όπως βλέπουμε και οι δύο εντολές επιστρέφουν λίστες με στοιχεία λίστες. Στην έξοδο **(%o18)** το πρώτο στοιχείο λίστα περιέχει τις ιδιοτιμές ενώ το δεύτερο τις πολλαπλότητες αυτών αντίστοιχα.

Α.3 ΓΡΑΦΙΚΑ ΔΥΟ ΔΙΑΣΤΑΣΕΩΝ

Επανερχόμαστε τώρα στο θέμα των γραφικών. Όπως έχουμε είδη αναφέρει η βασική εντολή σχεδιασμού γραφικών παραστάσεων δύο διαστάσεων είναι η **plot2d**. Τώρα θα δούμε μερικές απο τις επιλογές με τις οποίες μπορούμε να τροφοδοτήσουμε την **plot2d** (ή και την **plot3d** όπως θα δούμε αργότερα).

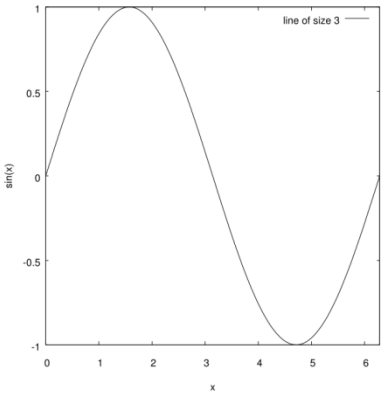
Έχουμε είδη δει ότι ο ορισμός του κατακόρυφου διαστήματος γίνεται με την **[y,a,b]**. Ένα άλλο σημαντικό θέμα είναι η ονομασία των αξόνων που επιτυγχάνεται με τις εντολές **[xlabel,"όνομαx"]** και **[ylabel,"όνομαy"]**. Επίσης μπορούμε να ορίσουμε τα ονόματα που θέλουμε στην επεξήγηση/επιγραφή χρησιμοποιώντας την εντολή **[legend,"όνομα1","όνομα2",…]**.

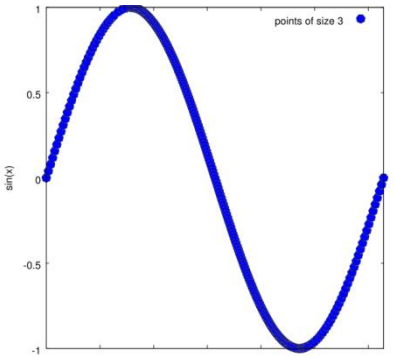
```
(%i1) plot2d([x^2+1,(x^2+1)^2+1],[x,-2,2],[xlabel,"x"],
      [ylabel,"y"],[legend,"f(x)","f(f(x))"]);
```



Αν θέλουμε να ορίσουμε το είδος και το πάχος της γραμμής ή των σημείων που θα χρησιμοποιηθούν κατά τη σχεδίαση πρέπει να προσθέσουμε την **[style,…]** όπως παρακάτω.

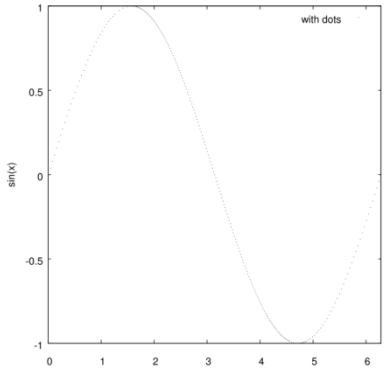
```
(%i2) plot2d(sin(x),[x,0,2*%pi],[style,[lines,3]],
      [legend,"line of size 3"]);
(%i3) plot2d([x^2+1,(x^2+1)^2+1],
      [x,-2,2],[style,[points,3]],
      [legend,"points of size 3"]);
(%i4) plot2d([x^2+1,(x^2+1)^2+1],[x,-2,2],[style,dots],
      [legend,"with dots"]);
```





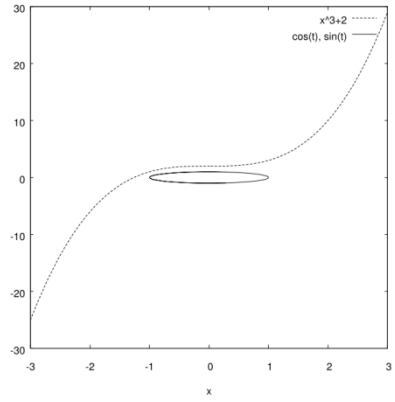
Σημαντικό είναι το γεγονός ότι μπορούμε να σχεδιάσουμε ταυτόχρονα μία καμπύλη που δίνεται σε παραμετρική μορφή με κάποια συνάρτηση με την εντολή:

```
(%i5) plot2d([x^3+2,[parametric,cos(t),sin(t),[t,-5,5],
      [nticks,80]]],[x,-3, 3]);
```



Τέλος, οι προσθήκες **[logx]** και **[logy]** ορίζουν τους άξονες σε κλίμακα λογάριθμου. Αλλά αρκετά με την **plot2d**, θα περάσουμε τώρα στην εντολή **plotdf** η οποία σχεδιάζει το διανυσματικό πεδίο σε δύο διαστάσεις ώστε να παράγουμε ένα σχήμα σαν αυτό του εξώφυλλου. Στην πιο γενική της μορφή η **plotdf** συντάσεται ως εξής:

plotdf([dxdt,dydt],[x,y],επιλογές)

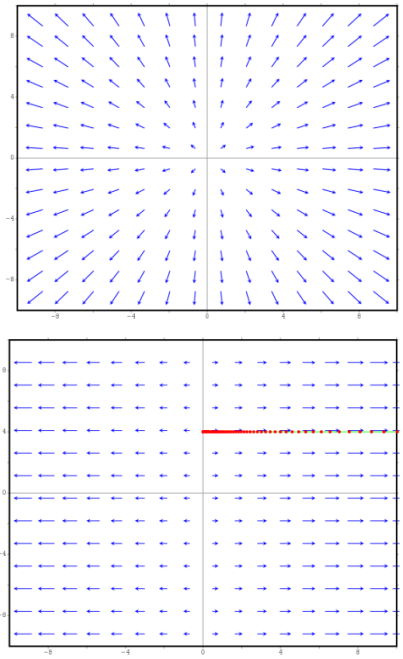


όπου οι «ταχύτητες» **dxdt** και **dydt** είναι συναρτήσεις των **x** και **y**. Επιπρόσθετα μπορεί να υπάρχει και εξάρτηση απο πλήθος παραμέτρων με αριθμητικές τιμές που δίνονται με την επιλογή **parameters**. Η **plotdf** πριν σχεδιάσει λύνει αριθμητικά (με τη μέθοδο των Adams Moulton ή με Runge Kutta $4^{ης}$ τάξης) το σύστημα διαφορικών εξισώσεων

$$\frac{dx}{dt} = f(x,y), \frac{dy}{dt} = g(x,y)$$

όπου φυσικά μπορούμε να ορίσουμε τον αριθμό των βημάτων με χρήση της επιλογής **nsteps** ενώ το χρονικό διάστημα ολοκλήρωσης με την **tstep**. Έτσι εκτός από το διανυσματικό πεδίο μπορούμε να σχεδιάσουμε και τις φασικές τροχιές για αρχικές συνθήκες που

ορίζονται απο την επιλογή `trajectory_at`. Επίσης το διάγραμμα επιτρέπει στο χρήστη τη διαδραστική αλληλεπίδραση όπου με το πάτημα του πλήκτρου του ποντικιού σχεδιάζεται η αντίστοιχη φασική τροχιά. Τέλος να σημειώσουμε ότι πριν απο τη χρήση της `plotdf` πρέπει να φορτώσουμε το πακέτο που την περιέχει που δεν είναι άλλο από το `plotdf` όπως βλέπουμε στα παρακάτω παραδείγματα.



```
(%i6) load(plotdf)$
(%i7) plotdf([x,y]);
(%i8) plotdf([x,-k*y],[x,y],[parameters,"k=0"],
      [sliders,"k=-1:1"],[trajectory_at,4,4]);
```

όπου στην είσοδο `(%i8)` η επιλογή `sliders` προσθέτει στο διάγραμμα διαδραστική ρύθμιση της (ή των) παραμέτρου.

Τέλος να αναφέρουμε ότι η `plotdf` κάνει χρήση του πακέτου openmath, ένα σχεδιαστικό πακέτο το οποίο μπορούμε να χρησιμοποιούμε αντί του gnuplot με την επιλογή `[plot_format,openmath]` στην `plot2d`.

Α.4 ΣΤΟΙΧΕΙΩΔΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Σε αυτή την παράγραφο θα εισαχθούμε στο προγραμματιστικό περιβάλλον του Maxima. Για να επιτύχουμε αυτό το άλμα θα πρέπει να αναφέρουμε τα προγραμματιστικά στοιχεία του Maxima. Να τονίσουμε ότι δε θα χρησιμοποιούμε αριθμούς εισόδου – εξόδου όπως κάναμε μέχρι τώρα αλλά θα γράφουμε με το παραδοσιακό πλέον μπλε τις γραμμές κώδικα. Ας αρχίσουμε με τη δήλωση λογικής συνθήκης `if` με το παρακάτω παράδειγμα.

```
f1(x):=if x<1 then 1 else x*f1(x-1)$
f1(5);
120
```

όπου εκτελέστηκε ο υπολογισμός `f(5)=5f(4)=5(4f(3))=20(3f(2))=60(2f(1))=120`. Το ίδιο παράδειγμα μπορεί να γραφτεί με χρήση της `while` ως εξής:

```
f2(x):=block([t],t:1,while x>1 do (t:x*t,x:x-1),t)$
f2(5);
120
```

όπου το πρώτο μέρος της `block` περιέχει μία λίστα με τις τοπικές μεταβλητές. Αν μας ρωτούσαν «είναι το Α μεγαλύτερο απο το Β;» θα μπορούσαμε να απαντήσουμε με «ναι» ή «όχι». Βέβαια στην περίπτωση που τα δεδομένα που μας δίνουν για τα Α και Β δεν είναι αρκετά για να πάρουμε μία απόφαση θα μπορούσαμε να απαντήσουμε «δεν είμαι σε θέση να γνωρίζω». Ας δούμε ένα παράδειγμα που θα εκτελί αυτόν τον έλεγχο

```
test(x,y):=block([],
     if x>y then print(x,"is greater than",y)
     elseif x<y then print(x,"is smaller than",y)
     elseif x=y then print(x,"is equal to",y)
     else print("who knows!"),
     return(alldone))$
```

η εκτέλεση της οποίας δίνει τα εξής:

```
test(1,2);
1 is smaller than 2 alldone
test(3,2);
3 is greater than 2 alldone
test(%i,2);
who knows! alldone
test(x**2+3,2);
x**2+3 is greater than 2 alldone
test(x**2+2,2);
alldone
```

όπου στο τελευταίο παράδειγμα το Maxima δεν ειναι σε θέση να γνωρίζει την απάντηση. Γιατί όμως δε δίνει στην έξοδο `who knows!`; Αυτό συμβαίνει γιατί το Maxima δε γνωρίζει ποιά σχέση υπάρχει μεταξύ των στοιχείων που δώσαμε και απο τη στιγμή που η μεταβλητή `prederror;` επιστρέφει `false` η εκτέλεση συνεχίζει στην επόμενη

δήλωση. Για να λύσουμε το πρόβλημα μας θα μπορούσαμε να θεωρήσουμε ότι το $x > 0$ με την παρακάτω εντολή

```
assume(x>0)$
test(x**2+2,2);
x**2+2 is greater than 2 alldone
```

ενώ αν θέλουμε να ξεχαστεί η υπόθεση μας πληκτρολογούμε `forget(x>0)`. Ένα άλλο σημαντικό στοιχείο είναι η κατασκευή συναρτήσεων με τυχαίο αριθμό μεταβλητών εισόδου. Η επίτευξη είναι σχετικά απλή αν σκεφτούμε ότι το όρισμα μπορεί να είναι μία λίστα στοιχείων.

```
prog([list]):=block([],
     print("your list is",list,"and its length is"),
     return(length(list)))$
prog(1,2,3,4,5,6,a);
your list is [1,2,3,4,5,6,a] and its length is 7
```

Στοιχείο κάθε σύγχρονης γλώσσας προγραμματισμού είναι οι πίνακες (της προγραμματιστικής ορολογίας, σαν απόδοση δηλαδή του όρου array). Ο ορισμός ενός τέτοιου πίνακα στο Maxima είναι απλός και μπορεί να έχει μία από τις παρακάτω μορφές:

```
a[3]:2*x$
a[x]:mystery$
a[x]:=cos(x)$
a[x+1];
cos(x+1)
```

ενώ μετά από όλα αυτά

```
a;
a
```

Μπορούμε να πάρουμε πληροφορίες για τον πίνακα με χρήση της εντολής `arrayinfo(a); [hashed,1,[3],[x],[x+1]]`. Ένα άλλο σημαντικό στοιχείο στη δομή των γλωσσών προγραμματισμού είναι οι επαναληπτικοί βρόγχοι.

```
f(n):=block([temp],temp:1
```

```
     for i:1 step 1 thru n do(temp:temp*i,print(temp))
     return(temp))$
f(4);
1
2
6
24
24
```

Όπου παρατηρούμε ότι μπορούμε να ομαδοποιούμε μία σειρά από εντολές με τη χρήση παρενθέσεων. Τέλος σημειώνουμε τις εντολές αποθήκευσης και κλήσης αρχείων και συναρτήσεων.

- `save(filename,all)`: αποθήκευση ολόκληρου του χώρου εργασίας.
- `save(filename,funcs)`: αποθήκευση συναρτήσεων.
- `load(filename)`: κλήση αποθηκευμένου αρχείου.

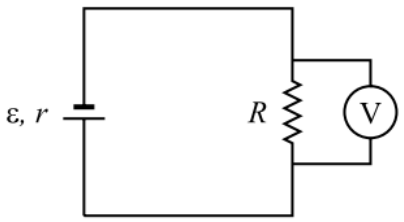
ΜΕΡΟΣ Β: Εφαρμογές

Β.1 ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΦΥΣΙΚΗΣ ΜΕ ΤΟ MAXIMA

Όπως έχουμε αναφέρει το Maxima είναι ένα πακέτο λογισμικού από την κατηγορία των CAS, δηλαδή, είναι ένα πρόγραμμα που μπορεί να χρησιμοποιηθεί οχι μόνο για αριθμητικούς υπολογισμούς, αλλά και για το χειρισμό αφηρημένων μεταβλητών. Υπάρχουν διάφορα τέτοια πακετα, έχουμε αποφασίσει να χρησιμοποιήσουμε το Maxima γιατί είναι Ελεύθερο Λογισμικό. Αυτό σημαίνει ότι μπορεί να εγκατασταθεί και να χρησιμοποιηθεί από τους σπουδαστές χωρίς να χρειαστεί άδεια χρήσης.

Το Maxima μπορεί να χρησιμοποιηθεί για να λυθούν προβλήματα φυσικής και μαθηματικών καθώς και να γραφτούν προγράμματα όπως γίνεται με τις παραδοσιακές γλώσσες προγραμματισμού. Τα ακόλουθα παραδείγματα βαδίζουν στην κατεύθυνση των εφαρμογών φυσικής. Τα παραδείγματα αυτά προέρχονται από τον τομέα της δυναμικής υλικών σημείων καθώς και απλών ηλεκτρικών κυκλωμάτων.

Παραδειγμα 1.1



Μια μπαταρία είναι συνδεδεμένη με μια εξωτερική αντίσταση R, και η τάση στα άκρα της μετράται με ένα ιδανικό βολτόμετρο V. Για να βρούμε την ΗΕΔ καθώς και την εσωτερική αντισταση r της μπαταρίαα, χρησιμοποιήθηκαν δύο εξωτερικές αντιστάσεις των $1.13k\Omega$ και $17.4k\Omega$. Η πτώση τάσης στις δύο περιπτώσεις ήταν $6{,}26V$ και $6{,}28\,V$. Βρείτε την ένταση του ρεύματος και στις δύο περιπτώσεις. Να βρεθεί η ΗΕΔ και η εσωτερική αντίσταση της πηγής καθώς και να σχεδιαστεί η ισχύς σα συνάρτηση της αντίστασης R στο διάστημα $[0{,}5r]$.

Λύση. Το ρεύμα που περνάει από την αντίσταση R βρίσκεται από το νόμο του Ohm:

$$I = \frac{V_R}{R}$$

```
(%i1) 6.26/1.13e3;
(%o1) .005539823008849558
```

Η έκφραση `1.13e3` είναι η μορφή που χρησιμοποιείται για τον αριθμό 1.13×10^3. Το ρεύμα στη δεύτερη περίπτωση είναι:

```
(%i2) 6.28/17.4e3;
(%o2) 3.609195402298851E-4
```

Έτσι το ρεύμα στην αντίσταση των $1.13\ k\Omega$ ειναι $5.54\ mA$, και στην αντίσταση των $17.4\ k\Omega$ είναι $0{,}361\ mA$. Για να υπολογίσουμε την ΗΕΔ και την εσωτερική αντίσταση της γεννήτριας έχουμε:

$$V_R = \varepsilon - rI$$

αντικαθιστώντας τις δύο τιμές για τα ΔV και R, έχουμε ένα σύστημα δύο εξισώσεων με δύο αγνώστους. Θα αποθηκεύσουμε αυτές τις δύο εξισώσεις στο Maxima ως `eq1` και `eq2`.

```
(%i3) eq1:6.26=emf-r*%o1$
(%i4) eq2:6.28=emf-r*%o2$
```

Οι δύο τελευταίες εξισώσεις αποτελούν ένα γραμμικό σύστημα δύο εξισώσεων με δύο μεταβλητές και μπορούμε να το λύσουμε με χρήση της συνάρτησης `solve`.

```
(%i5) bfloat(solve([eq1,eq2]));
(%o5) [[r=3.861821352953423b0, emf=6.281393806787158b0]]
```

Επομένως, η ΗΕΔ είναι περίπου $6{,}2814V$ και εσωτερική αντίσταση $3{,}8618\Omega$. Η ηλεκτρική ισχύς που διαχέεται στην αντίσταση R είναι:

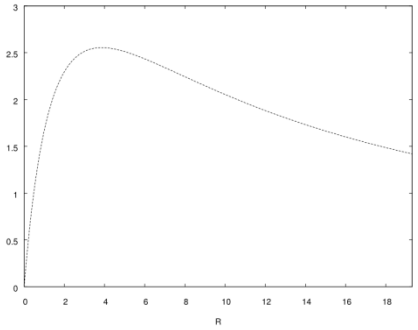
$$P_R = RI^2$$

ενώ το ρεύμα δίνεται πλεον από:

$$I = \frac{\varepsilon}{r + R}$$

άρα η ισχύς γράφεται:

$$P_R = R\frac{\varepsilon}{r+R}$$




```
(%i6) r:3.861821352953423b0$
(%i7) plot2d(R*(6.2814/(R+r))^2,[R,0,5*r]);
```

Παραδειγμα 1.2

Το διάνυσμα θέσης ενός σωματιδίου σα συνάρτηση του χρόνο t δίνεται από την εξίσωση:

$$\vec{r} = \left(5 - t^2 e^{-t/5}\right)\hat{e}_x + \left(3 - e^{-t/12}\right)\hat{e}_y$$

Να βρεθούν τα διανύσματα της θέσης, της ταχύτητας και της επιτάχυνσης για $t = 0$, $t = 15s$ και $t \to \infty$. Να σχεδιαστεί η τροχιά του σωματιδίου κατά τη διάρκεια των πρώτων 60 δευτερολέπτων της κίνησης του.

Λυση. Ξεκινάμε ορίζοντας το διάνυσμα $\vec{r}$ σα μία λίστα δύο στοιχείων, το πρώτο στοιχείο θα είναι η συντεταγμένη x και το δεύτερο η y.

```
(%i8) r:[5-t^2*exp(-t/5),3-exp(-t/12)]$
```

η διανυσματική ταχύτητα ισούται με την παράγωγο του διανύσματος θέσης ενώ η επιτάχυνση από την παράγωγο του διανύσματος της ταχύτητας. Οι αντίστοιχες εντολές του Maxima είναι:

```
 (%i9) v:diff(r,t)$
(%i10) a:diff(v,t)$
```

Για να βρούμε τη θέση, την ταχύτητα και την επιτάχυνση σε $t = 0$, χρησιμοποιουμε τις εξης εντολες:

```
(%i11) r,t=0,numer;
(%o11) [5,2]
(%i12) v,t=0,numer;
(%o12) [0,.08333333333333333]
(%i13) a,t=0,numer;
(%o13) [-2,-.006944444444444444]
```

Για $t = 15s$ τα αποτελέσματα θα παραχθούν με παρόμοιο τρόπο:

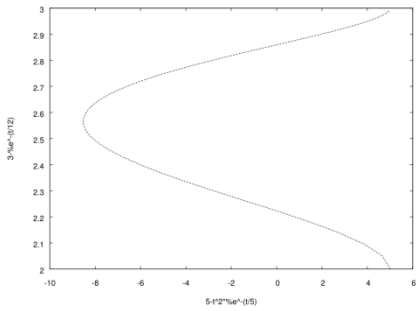
```
(%i14) r,t=15,numer;
(%o14) [-6.202090382769388,2.71349520313981]
(%i15) v,t=15,numer;
(%o15) [.7468060255179592,.02387539973834917]
(%i16) a,t=15,numer;
(%o16) [0.0497870683678639,-.001989616644862431]
```

Τις οριακές τιμές μπορούμε να τις υπολογίσουμε με χρήση ορίων πως παρακάτω:

```
(%i17) limit(r,t,inf);
(%o17) [5,3]
(%i18) limit(v,t,inf);
(%o18) [0,0]
(%i19) limit(a,t,inf);
(%o19) [0,0]
```

Έτσι, το σωματίδιο θα προσεγγίζει το σημείο [5,3], όπου και θα παραμείνει σε ηρεμία. Για να σχεδιάσουμε την τροχιά θα χρησιμοποιήσουμε τη συνάρτηση `plot2d`.

```
(%i20) plot2d([parametric,r[1],r[2],
                                [t,0,60],[nticks,100]]);
```



Β.2 ΔΥΝΑΜΙΚΑ ΣΥΣΤΗΜΑΤΑ ΣΕ ΔΙΑΚΡΙΤΟ ΧΡΟΝΟ - ΑΠΕΙΚΟΝΙΣΕΙΣ

Β.2.1 Εξέλιξη

Ονομάζουμε δικριτά δυναμικά συστήματα εκείνα τα οποία εξελίσσονται σε διακριτό χρόνο $\{t_0, t_1, t_2, \dots\}$. Στο διάστημα μεταξύ δύο χρονικών στιγμών καμία αλλαγή δε συμβαίνει στο σύστημα. Θα μελετήσουμε μονοδιάστατα τέτοια συστήματα τα οποία θεωρούμε ότι περιγράφονται από απεικονίσεις της μορφής:

$$x_{n+1} = f(x_n)$$

όπου n ο χρόνος. Η παραπάνω σχέση ορίζει κάθε επόμενη κατάσταση του συστήματος σα συνάρτηση της αμέσως προηγούμενες. Η πολλαπλή επανάληψη αυτής της σχέσης δίνει μία ακολουθία

καταστάσεων $\{x_0, x_1, x_2, \ldots\}$ η οποία ονομάζεται τροχιά. Για παράδειγμα, αν η απεικόνιση δίνεται από τη σχέση:

$$x_{n+1} = cosx_n$$

και γνωρίζουμε ότι $x_0 = 2$, οι διαδοχικές καταστάσεις του συστήματος θα είναι:

$$\{2, cos2, cos(cos2), cos\big(cos(cos2)\big), \ldots\}$$

Στη γενική της μορφή μία τροχιά γράφεται σαν:

$$\{x_0, f(x_0), f\big(f(x_0)\big), f(f(f(x_0))) \ldots\}$$

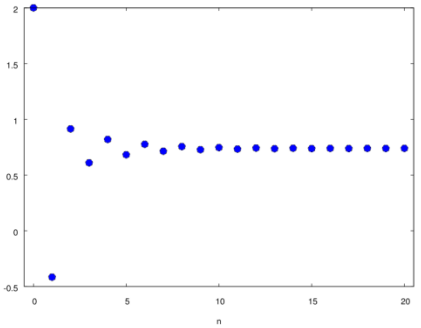
ή στην πιο συμπαγή μορφή:

$$\{x_0, f(x_0), f^2(x_0), f^3(x_0), \ldots\}$$

Β.2.2 Γραφική ανάλυση

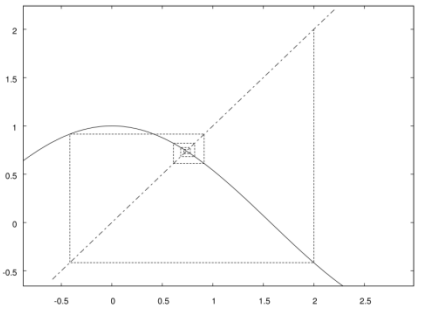
Μία γραφική μέθοδος για να παραστήσουμε την εξέλιξη ενός συστήματος συνίσταται στη σχεδίαση ενός σημείου για κάθε βήμα – χρονική στιγμή. Στο Maxima, η συνάρτηση `evolution` του πακέτου της δυναμικής (`dynamics`) σχεδιάζει ένα τέτοιο γράφημα. Τρια ορίσματα θα πρεπει να δοθούν σε αυτή τη συνάρτηση: μια έκφραση που εξαρτάται μόνο από τη μεταβλητη $x(\rightarrow x_n)$, η αρχική συνθήκη x_0 και τέλος ο αριθμός των επαναλήψεων. Για παράδειγμα θεωρούμε την απεικόνιση $x_{n+1} = cosx_n$ την οποία θα επαναλάβουμε 20 φορές ξεκινώντας από το $x_0 = 2$.

```
(%i1) load("dynamics")$
(%i2) evolution(cos(x),2,20)$
```



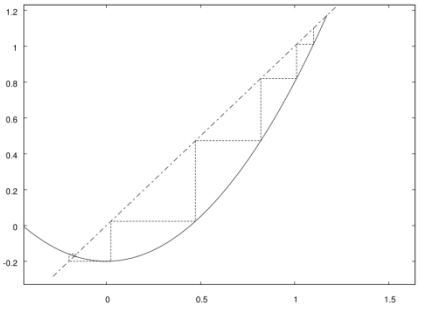
Ένα άλλο είδος διαγράμματος το οποίο είναι πολύ χρήσιμο για την ανάλυση διακριτών δυναμικών συστημάτων είναι το δίκτυο επαναλήψεων, το οποίο συνίσταται στη σχεδίαση των συναρτήσεων: $y = f(x), y = x$, καθώς και μία εναλλασσόμενη ακολουθία οριζόντιων και καθέτων τμηματων που ενώνουν τα σημεία $(x_0, x_0), (x_0, x_1), (x_1, x_1), (x_1, x_2)$, κ.ο.κ. Για παραδειγμα, το δίκτυο

επαναλήψεων για την απεικόνιση $x_{n+1} = cosx_n$ ξεκινώντας από το $x_0 = 2$ δίνεται από την εντολή:



```
(%i3) staircase(cos(x),2,8)$
```

Το δίκτυο επαναλήψεων επιτρέπει να κατανοήσουμε άμεσα, πότε μία ακολουθία συγκλίνει ή αποκλίνει. Ένα ακόμη παράδειγμα δίνεται από την απεικόνιση $x_{n+1} = x_n^2 - 0.2$ με σημείο έναρξης $x_0 = 1.1$.



```
(%i4) staircase(x^2-0.2,1.1,8)$
```

B.2.3 Σημεία ισορροπίας

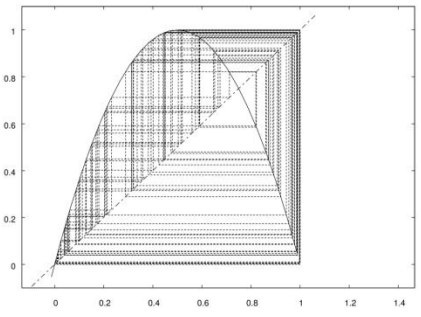
Ένα σημείο ισορροπίας είναι ένα σημείο x^* για το οποίο η κατάσταση του συστήματος θα παραμείνει σταθερή. Για να συμβεί κάτι τέτοιο, η αναγκαία και ικανή συνθήκη είναι:

$$f(x^*) = x^*$$

Από γραφικής άποψης, τα σημεία ισορροπίας είναι όλα εκείνα τα σημεία όπου η καμπύλη της $y = f(x)$ τέμνει τη γραμμή $y = x$ στο δίκτυο επαναλήψεων. Για παράδειγμα, στην περίπτωση της λογιστικής απεικόνισης $x_{n+1} = rx_n(1 - x_n)$ για $r = 2$ και $r = 4$ υπάρχουν δύο σημεία ισορροπίας το ένα εκ των οποίων είναι $x^* = 0$. Μπορούμε να χρησιμοποιήσουμε την εντολή `solve` για να βρούμε τα σημεία ισορροπίας. Για την περίπτωση που $r = 4$ έχουμε:

```
(%i1) logmap:4*x*(1-x)$
(%i2) fixed:solve(logmap-x);
(%o2) [x=3/4,x=0]
```

Έτσι τα σημεία ισορροπίας είναι $x_1^* = 0$ και $x_2^* = 0.75$. Επίσης το δίκτυο επαναλήψεων για $r = 4$ δίνεται από:



```
(%i3) load("dynamics")$
(%i4) staircase(4*x*(1-x),0.1,100)$
```

Θεωρούμε ένα σημείο ισορροπίας x^*, όπου η καμπύλη $C_{f(x)}$ τέμνει τη γραμμή $y = x$ και η παράγωγος σε αυτό το σημείο είναι

$f'(x^*) > 1$. Σε αυτή την περίπτωση η πορεία μίας τροχιάς που ξεκινάει κοντά στο x^* στο δίκτυο επαναλήψεων απομακρύνεται από το σημείο ισορροπίας. Τέτοια σημεία ισορροπίας ονομάζονται ασταθή. Το ίδιο ισχύει και στην περίπτωση που $f'(x^*) < -1$. Στην περίπτωση που $|f'(x^*)| < 1$ το σημείο ισορροπίας είναι ελκυστής υπο την έννοια ότι για αρχικές συνθήκες κοντα σε αυτό η τροχιά σταθεροποιείται επάνω του, τέτοια σημεία ισορροπίας ονομάζονται ασταθή.

Ας δούμε το είδος (της ισορροπίας) των σημείων ισορροπίας που υπολογίσαμε προηγουμένως για την λογιστική απεικόνιση όταν $r = 4$.

```
(%i5) dflogmap:diff(logmap,x);
(%o5) 4(1-x)-4x
(%i6) dflogmap,fixed[1];
(%o6) -2
(%i7) dflogistic,fixed[2];
(%o7) 4
```

Έτσι στην περίπτωση που $r = 4$ και τα δύο σημεία ισορροπίας είναι ασταθή.

Β.2.4 Περιοδικά σημεία

Αν η ακολουθια $\{x_0, x_1, x_2, \dots\}$ αποτελεί μία λύση του συστήματος $x_{n+1} = f(x_n)$ κάθε στοιχείο της ακολουθίας μπορεί να ληφθεί εφαρμόζοντας τη σύνθετη συνάρτηση f^n

$$x_n = f^n(x_0) = \underbrace{f(f(\dots f(x_0)))}_{n\ φορ\acute{ε}ς}$$

Μια λυση ονομάζεται περιοδική περιόδου 2, εαν ειναι μια ακολουθία από μόνο δυο εναλλασσομενες τιμες $\{x_0, x_1, x_0, x_1, \dots\}$. Τα δυο σημεια x_0, x_1 είναι περιοδικά με περιοδο 2 δεδομένου ότι $x_2 = f^2(x_0) = x_0$ και $x_3 = f^2(x_1) = x_1$. Ο κύκλος περιόδου 2 θα είναι ευσταθής ή ασταθής ανάλογα με την τιμή της παραγώγου f^2 σε κάθε σημείο του.

$$(f^2(x_0))' = (f(f(x_0)))' = f'(f(x_0))f'(x_0) = f'(x_0)f'(x_1)$$

Εαν η απόλυτη τιμή του γινομένου των παραγώγων είναι μεγαλύτερη του 1 ο περιοδικός κύκλος είναι ασταθής, ενώ αν είναι μικρότερη ο κύκλος είναι ευσταθής. Για τη λογιστική απεικόνιση $(r = 3.1)$ έχουμε:

```
(%i1) f(x):=3.1*x*(1-x)$
(%i2) f2:f(f(x));
(%o2) 9.610000000000001(1-x)x(1-3.1(1-x)x)
(%i3) cycle:solve(f2-x),numer;
(%o3) [x=0.55801412541898,x=0.76456651974231,
       x=0.67741935483871,x=0]
```

Τα δυο τελευταία σημεία είναι σημεία ισορροπίας και δεν ανήκουν στον περιοδικό κύκλο. Τα άλλα δύο σημεία, αποτελούν ένα κύκλο περιόδου 2. Για να μάθουμε εαν ο κύκλος είναι ευσταθής ή ασταθής, θα υπολογίσουμε τις παραγώγους σε κάθε ένα από τα σημεία του κύκλου:

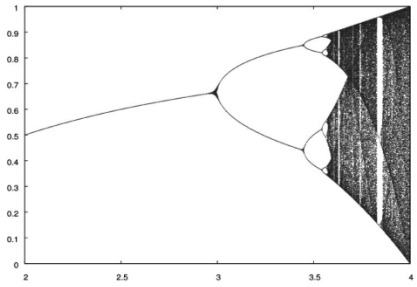
```
(%i4) dfdx:diff(f(x),x);
(%o4) 3.1(1-x)-3.1x
(%i5) dfdx,cycle[1];
(%o5) -0.35968757759769
(%i6) dfdx,cycle[2];
(%o6) -1.640312422402312
```

Η απόλυτη τιμή του γινομένου των παραγώγων είναι < 1, συνεπώς ο περιοδικός κύκλος είναι ευσταθής.

Β.2.5 Διάγραμμα διακλαδώσεων

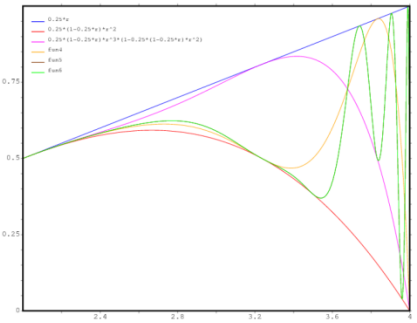
Τέλος θα αναφερθούμε στο διάγραμμα διακλαδώσεων της λογιστικής απεικόνισης που μας δείχνει την εξάρτηση του συστήματος από την παράμετρο r. Σε αυτό το διάγραμμα επαναλαμβάνουμε την απεικόνιση μεγάλο πλήθος φορών για κάθε τιμή της παραμέτρου και σημειώνουμε τις τιμές της ακολουθίας (αγνοούμε όμως τις αρχικές επαναλήψεις ώστε αν υπάρχουν σημεία ισορροπίας ή κύκλοι να σημειωθούν μόνο αυτά και οχι οι καταστάσεις του συστήματος πριν καταλήξει σε αυτά.). Το Maxima παρέχει έτοιμη τη συνάρτηση για τη σχεδίαση του διαγράμματος αυτού τη σύνταξη της οποίας βλέπουμε στο παρακάτω πάραδειγμα:

```
(%i1) load("dynamics")$
(%i2) orbits(r*x*(1-x),0.5,100,400,[r,2,4,0.0001],
      [nticks,1000],[style, dots]);
```



Οι σκοτεινές γραμμές πού διατρέχουν τα πυκνά μέρη του διαγράμματος διακλαδώσεων μπορούν να κατανοηθούν. Ορίζουμε το κρίσημου σημείο $x_c = 0.5$ στο οποίο η πρώτη παράγωγος μηδενίζεται. Το γεγονός αυτό σημαίνει οτι αν μία επανάληψη πλησιάσει το x_c η επόμενη επανάληψη θα πλησιάσει το $f(x_c)$ και η επόμενη το $f(f(x_c))$ κ.ο.κ. Έτσι η πυκνότητα σημείων είναι αυξημένη σε αυτές τις καμπύλες.

```
(%i3) f(x):=r*x*(1-x)$
(%i4) f1:f(0.5)$
      f2:f(f(0.5))$
      f3:f(f(f(0.5)))$
      f4:f(f(f(f(0.5))))$
      f5:f(f(f(f(f(0.5)))))$
      f6:f(f(f(f(f(f(0.5))))))$
(%i5) plot2d([f1,f2,f3,f4,f5,f6],[r,2,4],
      [plot_format,openmath])
```



B.3 ΑΡΙΘΜΗΤΙΚΗ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΕΞΙΣΩΣΕΩΝ ΚΙΝΗΣΗΣ

B.3.1 Ο αλγόριθμος της «μακριάς γαϊδάρας»

Η πιο απλή μέθοδος για την επίλυση διαφορικών εξισώσεων είναι αυτή του Euler, στην οποία απλά προσεγγίζουμε την παράγωγο με μία πεπερασμένη διαφορά:

$$x'(t) \approx \frac{\Delta x}{\Delta t}$$

έτσι για παράδειγμα η διαφορική εξίσωση $\frac{dx}{dt} = f(x)$ θα προσεγγιστεί από την επαναληπτική σχέση:

$$x_{n+1} = x_n + \Delta t f(x)$$

Μπορούμε να χρησιμοποιήσουμε την ίδια περίπου ιδέα σε μια απλή και κομψή μέθοδο για τις εξισώσεις της κίνησης του νόμου του Νεύτωνα, οι οποίες έχουν το πλεονέκτημα ότι η ταχύτητα δεν εξαρτάται από τη θέση και η επιτάχυνση από την ταχύτητα. Για αδιάστατο σωματίδιο που κινείται στην πραγματική γραμμή έχουμε:

$$\begin{cases} \dfrac{dx}{dt} = u \\ \dfrac{du}{dt} = F(x) = -\dfrac{dV}{dx} \end{cases}$$

όπου $F(x)$ η δύναμη που δρα στο σωματίδιο όταν είναι στη θέση x και $V(x)$ η δυναμική ενέργεια (για απλότητα θέτουμε $m = 1$). Μια καλύτερη προσέγγιση από αυτή της μεθόδου του Euler είναι η αντικατάσταση της ταχύτητας με την τιμή της στο μέσο του μελετούμενου διαστήματος,

$$x_1 = x_0 + hu_{1/2}$$

όπου υποθέστε ότι μπορούμε να πάρουμε την τιμή $u_{1/2}$ με κάποιον τρόπο. Κατόπιν μπορούμε να εφαρμόσουμε έναν παρόμοιο κανόνα μεσαίου σημείου στη δεύτερη εξίσωση:

$$u_{3/2} = u_{1/2} + hF(x_1)$$

δεδομένου ότι ξέρουμε το x_1. Έπειτα μπορούμε βήμα προς βήμα να εκτελέσουμε την αριθμιτική ολοκλήρωση με τη μέθοδο της «μακριάς γαϊδούρας», της οποίας το γενικό σχήμα είναι:

$$x_{n+1} = x_n + hu_{n+1/2}, \;\; u_{n+3/2} = u_{n+1/2} + hF(x_{n+1})$$

Το σφάλμα της μεθόδου για ένα υποδιάστημα είναι $\sim h^3$, ενώ για το συνολικό διάστημα έχουμε $\sim h^3 \times (1/h) = h^2$, που σημαίνει ότι η μέθοδος είναι δεύτερης τάξης.

Β.3.2 Ο αλγόριθμος ταχύτητας Verlet

Για να μπορέσουμε να χρησιμοποιήσουμε την παραπάνω μέθοδο, δύο ερωτήματα πρέπει να εξεταστούν. Το πρώτο είναι η αρχικοποίηση της μεθόδου, εφόσον δεν είναι γνωστή η ταχύτητα $u_{1/2}$. Η απλούστερη προσέγγιση είναι η εκτέλεση ενός μισού βήματος:

$$u_{1/2} = u_0 + \frac{1}{2}hF(x_0)$$

Αν και αυτή η προσέγγιση δεν είναι μεσαίου σημείου, και το σφάλμα είναι h^2, η μοναδική εκτέλεση ενός τέοιου βήματος δεν επιρεάζει τη γενική απόδοση της μεθόδου. Το δεύτερο ερώτημα είναι το πώς μπορούμε να πάρουμε την ταχύτητα στον ίδιο χρόνο με τη θέση (κατάσταση που είναι βολική π.χ. για το σχεδισμό του φασικού χώρου). Η απλούστερη προσέγγιση είναι η διάσπαση της $u_{n+3/2} = u_{n+1/2} + hF(x_{n+1})$ σε δύο μισά βήματα, τα οποία επιτυχώς συσχετίζουν την u_{n+1} με τις $u_{n+1/2}, u_{n+3/2}$. Ο αλγόριθμος που πλέον ονομάζεται μέθοδος ταχυτήτων Verlet είναι:

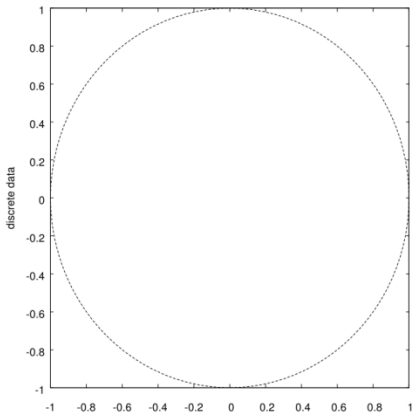
$$\begin{aligned} u_{n+1/2} &= u_n + \frac{1}{2}hF(x_n), \\ x_{n+1} &= x_n + hu_{n+1/2}, \\ u_{n+1} &= u_{n+1/2} + \frac{1}{2}hF(x_{n+1}) \end{aligned}$$

ενώ με όμοιο τρόπο βρίσκουμε τη μέθοδο θέσεων Verlet:

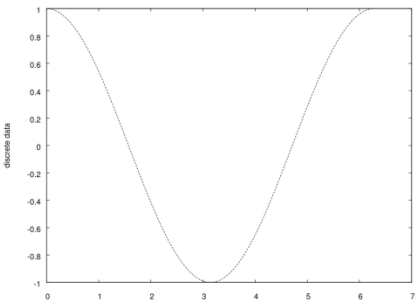
$$\begin{aligned} x_{n+1/2} &= x_n + \frac{1}{2}hu_n, \\ u_{n+1} &= u_n + hF(x_{n+1/2}), \\ x_{n+1} &= x_{n+1/2} + \frac{1}{2}hu_{n+1} \end{aligned}$$

Η υλοποίηση της μεθόδου στο Maxima γίνεται με το παρακάτω πρόγραμμα/συνάρτηση την οποία και θα χρησιμοποιήσουμε για τη λύση του απλού αρμονικού ταλαντωτή.

```
leapfrog(f,initial,tfinal,steps):=block(
   [h:(tfinal-initial[1])/steps,
    t:initial[1],
    x:initial[2],
    u:initial[3],
    tnew,umid,xnew,unew],
   time:[t],position:[x],velocity:[u],
   for  i:1  thru  steps  do  block(
    tnew:float(t+h),
    umid:float(u+0.5*h*f(x)),
    xnew:float(x+h*umid),
    unew:float(umid+0.5*h*f(xnew)),
    t:tnew,x:xnew,u:unew,
    time:append(time,[t]),
    position:append(position,[x]),
    velocity:append(velocity,[u])),
plot2d([discrete,position,velocity]))$
save(leapfrog,leapfrog)$
f(x):=-x$
leapfrog(f,[0,1,0],2*%pi,100);
```



Η εκτέλεση του προγράμματος επιστρέφει γραφικά το φασικό χώρο, αλλά εφόσον έχουμε αποθηκεύσει τις τιμές χρόνου, θέσης και ταχύτητας (**time, position, velocity**) μπορούμε να σχεδιάσουμε όποιο άλλο μέγεθος επιθυμούμε σα συνάρτηση του χρόνου:




```
plot2d([discrete,time,position])$
```

ΜΕΡΟΣ Γ: Αριθμητικές μέθοδοι

Γ.1 ΕΙΣΑΓΩΓΗ

Οι αριθμητικές μέθοδοι στη φυσική είναι απλές. Πάρτε τον τύπο σας και αντικαταστήστε κάθε απειροστή διαφορά dx με την **πεπερασμένη διαφορά** Δx. Η αντικατάσταση αυτή επιβάλλει την επιπλέον αντικατάσταση των ολοκληρωματων από αθροίσματα, δηλαδή,

$$\int_a^b f(x)dx \approx \sum_{n=0}^{N} f(a+n\Delta x)\Delta x$$

με $N\Delta x = b-a$. Τα άσχημα νέα είναι ότι με τις παραπάνω προσεγγίσεις εισαγάγουμε ένα υπολογιστικό σφάλμα. Όσο μικρότερο είναι το **βήμα** Δx τόσο μικρότερο θα είναι το **σφάλμα αποκοπής** (σφάλμα που οφείλεται στις προσεγγίσεις μας), αλλά ο απαραίτητος υπολογιστικός χρόνος αυξάνεται δεδομένου ότι έχουμε να κάνουμε με μεγαλύτερο αριθμό συναρτήσεων. Για να γίνει κατανοητό το σφάλμα αποκοπής απλά θεωρούμε το ανάπτυγμα Taylor:

$$f(x+\Delta x) = \sum_{n=0}^{\infty} \frac{\Delta x^n}{n!} f^{(n)}(x)$$

Αυτά είναι όλα που πρέπει να ξέρετε. Σχεδόν. Για να κάνετε αποτελεσματικά αριθμητικούς υπολογισμούς πρέπει να αποκτήσετε εμπειρία και να μάθετε πολλά τεχνάσματα. Είναι ιδιαιτερα σημαντικό να γίνουν κατανοητά τα τεχνάσματα εκείνα που θα συναντήσουμε όταν μας απασχολήσουν ψευδοτυχαίοι αριθμοί καθώς και αυτές οι μέθοδοι που αφορούν την επίλυση διαφορικών εξισώσεων.

Γ.2 ΣΦΑΛΜΑΤΑ, ΠΑΡΕΜΒΟΛΗ, ΠΑΡΕΚΤΑΣΗ

Γ.2.1 Σφάλματα

Οι αριθμητικοί υπολογισμοί γίνονται με ακέραιους αριθμούς ή με αριθμούς κινητής υποδιατολής. Η αριθμητική με ακέραιους είναι ακριβής ενώ όταν οι αριθμοί είναι κινητής υποδιαστολής ειπεισέχονται σφάλματα στρογγυλοποίησης. Η αναπαράσταση των δεύτερων γίνεται ως εξής:

$$s \cdot M \cdot 2^p \begin{cases} s: \pi\rho\acute{o}\sigma\eta\mu o \\ M: \delta\epsilon\kappa\alpha\delta\iota\kappa\acute{o}\ \mu\acute{\epsilon}\rho o\varsigma \\ p: \alpha\kappa\acute{\epsilon}\rho\alpha\iota o\varsigma\ \epsilon\kappa\theta\acute{\epsilon}\tau\eta\varsigma \end{cases}$$

Οποιοσδήποτε πραγματικός αριθμός μπορεί να παρασταθεί με αυτόν τον τρόπο, και η αναπαράσταση αυτή γίνεται μοναδική εάν απαιτήσουμε π.χ. $\frac{1}{2} \leq M < 1$ για κάθε μη μηδενικό αριθμό. Στον υπολογιστή, κάθε αριθμός κινητής υποδιαστολής γίνεται μια σειρά από bits, με κάθε bit να είναι 0 ή 1. Υποθέστε ότι ο αριθμός bits ανά αριθμό (το «μήκος λέξης») είναι 32, με 1 bit για το πρόσημο, 8 bits για το p και 23 bits για το M. Το p και το M μπορούν έπειτα (παραδείγματος χάριν) να παρασταθούν σαν:

$$p = x_7 2^7 + x_6 2^6 + \cdots + x_0 2^0 - 2^7, \qquad x_i \in \{0,1\}$$
$$M = y_1 2^{-1} + y_2 2^{-2} + \cdots + y_{23} 2^{-23}, \qquad y_i \in \{0,1\}$$

Τα σφάλματα στρογγυλοποίησης προκύπτουν επειδή ο αριθμός των y_i είναι πεπερασμένος. Πόσο μεγάλα είναι τα σφάλματα αυτά; Ο αριθμός 1 μπορεί να αντιπροσωπευθεί ακριβώς ($x_7 = x_0 = y_1 = 1$ και όλα τα άλλα bits μηδέν). Ο μικρότερος αριθμός > 1 που μπορεί να παραχθεί είναι $1 + 2^{-22}$($x_7 = x_0 = y_1 = y_{23} = 1$ και όλα τα άλλα bits μηδέν). Αυτό δίνει μια εκτίμηση της χαρακτηριστικής σχετικής ακρίβειας ε των τριανταδυάμπιτων κινητών δεκαδικών, $\varepsilon \sim 2^{-22} \sim 10^{-7}$. Τα σφάλματα στρογγυλοποίησης είναι ιδιαίτερα ενοχλητικά κατά την αφαίρεση δύο αριθμών με μικρή σχετική διαφορά. Ένα παράδειγμα είναι μια από τις λύσεις της εξίσωσης $ax^2 + bx + c = 0$:

$$x = -\frac{b - \sqrt{b^2 - 4ac}}{2a}$$

για $\sqrt{ac} \ll b$. Εδώ το τέχνασμα είναι απλό. Πολλαπλασιάζουμε με $b + \sqrt{b^2 - 4ac}$ αριθμητή και παρονομαστή:

$$x = -\frac{b - \sqrt{b^2 - 4ac}}{2a} \times \frac{b + \sqrt{b^2 - 4ac}}{b + \sqrt{b^2 - 4ac}} = -\frac{2c}{b + \sqrt{b^2 - 4ac}}$$

Εκτός από τα σφάλματα στρογγυλοποίησης, οι περισσότεροι αριθμητικοί υπολογισμοί περιέχουν σφάλματα λόγω προσεγγίσεων, όπως η διακρητοποίηση ή η αποκοπή μιας άπειρης σειράς. Τέτοια σφάλματα ονομάζονται σφάλματα αποκοπής. Τα σφάλματα αποκοπής θα ενέμεναν ακόμα και σε έναν υποθετικό υπολογιστή με άπειρη ακρίβεια.

Παράδειγμα: Αριθμητική παραγώγιση

Υποθέστε ότι η παράγωγος $f'(x)$ υπολογίζεται με χρηση του αναπτύγματος Taylor:

$$f(x + h) = f(x) + hf^{'(x)} + \frac{h^2}{2}f''(x) + \cdots$$
$$f'(x) \approx \frac{f(x + h) - f(x)}{h} + O(h)$$

- Το σφάλμα αποκοπής είναι $\varepsilon_t \sim h|f''(x)|$.
- Τα σφάλματα στρογγυλοποίησης στις $f(x + h)$ και $f(x)$ είναι $\gtrsim \varepsilon|f(x)|$, όπου ε η σχετική δεκαδική ακρίβεια. Επομένως, ένα κατώτατο όριο στο σφάλμα στρογγυλοποίησης ε_r είναι $\varepsilon_r \gtrsim \varepsilon|f(x)|/h$.

Για το συνολικό σχετικό σφάλμα, παίρνουμε την εκτίμηση:

$$\frac{\varepsilon_r + \varepsilon_t}{|f'|} \gtrsim \frac{h|f''| + \varepsilon|f|/h}{|f'|}$$
$$= \frac{1}{|f'|}\left[\left(\sqrt{h|f''|} - \sqrt{\varepsilon|f|/h}\right)^2 + 2\sqrt{\varepsilon|f''f|}\right]$$

Με μία βέλτιστη επιλογή του h (έτσι ώστε το τετράγωνο εξαφανίζεται), αυτό γίνεται:

$$\frac{\varepsilon_r + \varepsilon_t}{|f'|} \gtrsim \sqrt{\varepsilon}\sqrt{\frac{|f''f|}{f'^2}}$$

Έτσι, το σφάλμα είναι $O(\varepsilon^{1/2})$ αντί για $O(\varepsilon)$, το οποίο κάνει μια διαφορά $\varepsilon = 10^{-7} \Rightarrow \varepsilon^{1/2} \approx 3 \cdot 10^{-4}$. Πώς μπορούμε να βελτιωσούμε το αποτέλεσμα αυτό;

- Ένας τρόπος είναι να παραγάγουμε μία καλύτερη προσέγγιση της παραγώγου από το ανάπτυγμα Taylor:

$$\left.\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + O(h^3) \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2}f''(x) + O(h^3) \end{aligned}\right\} \overset{(-)}{\Longrightarrow}$$
$$f(x+h) - f(x-h) = 2hf'(x) + O(h^3) \Rightarrow$$
$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

  Το σφάλμα αποκοπής για την κεντρική αυτή διαφορά είναι $O(h^2)$ είναι βελτιωμένο κατά μία τάξη.
- Μια άλλη δυνατότητα είναι να χρησιμοποιηθεί η παρέκταση Richardson.

Γ.2.2 Παρέκταση Richardson

Υποθέστε ότι θέλουμε να καθορίσουμε μια ποσότητα a_0 που ικανοποιεί τη σχέση $a_0 = \lim_{h\to 0} a(h)$, όπου το h μπορεί να θεωρηθεί ως παράμετρος βήματος. Περαιτέρω αυτού υποθέτουμε ότι

- η συναρτησιακή μορφή $a(h)$ είναι γνωστό ότι είναι $a(h) = a_0 + a_1 h^2 + a_2 h^4 + \cdots$,
- η τιμή του $a(h)$ είναι γνωστή για $h = h_0, 2h_0, 2^2 h_0, \ldots$

(a_0 θα μπορούσε να είναι η παράγωγος και $a(h)$ η προσέγγιση κεντρικής διαφοράς).

Μπορούμε έπειτα να λάβουμε μία βελτιωμένη εκτίμηση $\tilde{a}(h)$ του a_0 με τον ακόλουθο τρόπο:

$$\left.\begin{aligned} 4a(h) &= 4a_0 + 4a_1 h^2 + 4a_2 h^4 + O(h^6) \\ a(2h) &= a_0 + 4a_1 h^2 + 16a_2 h^4 + O(h^6) \end{aligned}\right\} \overset{(-)}{\Longrightarrow}$$
$$4a(h) - a(2h) = 3a_0 - 12a_2 h^4 + O(h^6) \Rightarrow$$
$$\tilde{a}(h) = \frac{4a(h) - a(2h)}{3} = a_0 - 4a_2 h^4 + O(h^6)$$

$\tilde{a}(h)$ είναι βελτιωμένη εκτίμησή του a_0 επειδή το σφάλμα είναι $O(h^4)$ αντί $O(h^2)$. Μπορούμε τώρα να συνεχίσουμε για να αποβληθεί ο όρος $\propto h^4$ με το σχηματισμό της ποσότητας:

$$\tilde{\tilde{a}}(h) = \frac{16\tilde{a}(h) - \tilde{a}(2h)}{15} = a_0 + O(h^6)$$

Παραδείγματα μεθόδων που χρησιμοποιούν αυτή την τεχνική είναι η μέθοδος ολοκλήρωσης του Romberg και η μέθοδος Burlirsh - Stoer για τις συνήθεις διαφορικές εξισώσεις.

Γ.2.3 Παρεμβολή και παρέκταση

Υποθέστε ότι μας δίνεται ένα σύνολο σημείων $(x_i, y_i), i = 1, \ldots, N$ και θέλουμε να βρούμε μια προσέγγιση ή μία συνάρτηση παρεμβολής $f(x; \{c_j\})$, όπου τα c_j είναι παράμετροι. Για μία δεδομένη συναρτησιακή μορφή της f, το πρόβλημα μας μετατοπίζεται στην εύρεση των βέλτιστων τιμών των παραμέτρων c_j.

Υποθέτουμε ότι ο τύπος της συνάρτησης f είναι ένα πολυώνυμο βαθμού $M - 1$,

$$f(x; \{c_j\}) = c_1 + c_2 x + \cdots + c_M x^{M-1}$$

Εάν είναι ή όχι δυνατό να βρεθούν c_j τέτοια ώστε:

$$f(x_i;\{c_j\}) = y_i, i = 1,2,\dots,N$$
$$\Leftrightarrow \begin{cases} c_1 + c_2 x_1 + \cdots + c_M x_1^{M-1} = y_1 \\ \vdots \\ c_1 + c_2 x_N + \cdots + c_M x_N^{M-1} = y_N \end{cases}$$

εξαρτάται από τα N και M, έχουμε ένα γραμμικό σύστημα N εξισώσεων για τις M άγνωστες παράμετρους c_j.

- $\boldsymbol{M > N}$**:** Αυτή η περίπτωση είναι λιγότερο ενδιαφέρουσα, υπάρχουν πάρα πολλές παράμετροι.
- $\boldsymbol{M = N}$**:** Σε αυτή την περίπτωση, υπάρχει μια μοναδική λύση που δίνεται από τον τύπο παρεμβολής του Lagrange:

  $$f(x) = \frac{(x-x_2)(x-x_3)\dots(x-x_N)}{(x_1-x_2)(x_1-x_3)\dots(x_1-x_N)} y_1 + \cdots + \frac{(x-x_1)\dots(x-x_{N-1})}{(x_N-x_1)\dots(x_N-x_{N-1})} y_N$$

  με $x_i \neq x_j$ για $i \neq j$ (γεωμετρικά: μέσω οποιωνδήποτε δύο σημείων υπάρχει μοναδική ευθεία ή πρώτου βαθμού πολυώνυμο, μέσω οποιωνδήποτε τριών σημείων υπάρχει μοναδικό δευτέρου βαθμού πολυώνυμο κ.ο.κ.).
- $\boldsymbol{M < N}$**:** Γενικά, δεν υπάρχει καμία λύση σε αυτήν την περίπτωση, έτσι πρέπει να προσεγγίσουμε αντί να παρεμβάλουμε. Αυτό γίνεται συχνά με χρήση της μεθόδου των ελαχίστων τετραγώνων, τα c_j καθορίζονται με ελαχιστοποίηση της ποσότητας:

  $$\sum_{i=1}^{N} [f(x_i;\{c_j\}) - y_i]^2$$

Το ότι έχουμε πάρει την $f(x;\{c_j\})$ να είναι ένα πολυώνυμο είναι κατάλληλη επιλογή, παραδείγματος χάριν, όταν ολοκληρώνουμε

αριθμητικά. Δύο εναλλακτικές των πολυωνύμων λύσεις είναι οι ρητές συναρτήσεις και τις κυβικές αυλακώσεις (splines).

Ρητές συναρτήσεις

Παρεμβολή. Υποθέστε ότι υπάρχουν τρία σημεία (x_i, y_i) και ότι $f\left(x, \{c_j\}\right) = (x - c_1)/(c_2 x + c_3)$. Η παρεμβολή ανέρχεται έπειτα σε επίλυση ενός γραμμικού συστήματος $f\left(x_i, \{c_j\}\right) = y_i \Leftrightarrow c_1 + c_2 x_i y_i + c_3 y_i = x_i, i = 1,2,3$. Γενικά μπορούμε να χρησιμοποιήσουμε οποιοδήποτε ρητό πολυώνυμο:

$$R_{\mu\nu} = \frac{P_\mu(x)}{Q_\nu(x)} = \frac{1 + \sum_1^\mu p_i x^i}{\sum_0^\nu q_j x^j}$$

με την επιλογή $\mu + \nu + 1 = N$, ώστε καταλήγουμε:

$$1 + \sum\nolimits_1^\mu p_i x_k^i - y_k \sum\nolimits_0^\nu q_j x_k^j = 0$$

Προσέγγιση Pade. Υποθέστε ότι θέλουμε να προσεγγίσουμε μία συνάρτηση $g(x)$ και γνωρίζουμε την $g(x)$ καθώς επίσης και τις παραγώγους $g', g'', \ldots$ για $x = 0$. Στη μέθοδο Pade, η συνάρτηση προσέγγισης $f(x; \{c_j\})$ είναι ρητή, και οι παράμετροι c_j καθορίζονται έτσι ώστε:

$$f\left(x = 0; \{c_j\}\right) = g(0), \frac{d^k}{dx^k} f\left(x = 0; \{c_j\}\right) = \frac{d^k}{dx^k} g(0)$$

μέχρι την υψηλότερη πιθανή τάξη k.

Παρεμβολή κυβικής αυλάκωσης (spline)

Ας είναι $a = x_1 < x_2 < \cdots < x_N = b$. Η συνάρτηση κυβικής αυλάκωσης λαμβάνεται με τη χρηση ενός κυβικού πολυωνύμου για κάθε διάστημα $[x_i, x_{i+1}]$. Αυτά τα πολυώνυμα τίθενται κατά τέτοιο τρόπο ώστε η προκύπτουσα συνάρτηση καθώς επίσης και οι δύο πρώτες παράγωγοι να είναι συνεχείς σε ολόκληρο το διάστημα $a \le x \le b$. Αυτό θα μας δώσει $N - 1$ συναρτήσεις με τέσσερις παραμέτρους η κάθε μία, δηλ. $4(N - 1)$ άγνωστος. Έχουμε $2(N - 1)$ εξισώσεις που αναγκάζουν κάθε συνάρτηση να περνάει από τα δύο

συνδεώμενα σημεία και $2(N-2)$ εξισώσεις για τη συνέχεια των πρώτων και δεύτερων παραγώγων. Αυτό σημαίνει ότι θα πρέπει να καθορίσουμε δύο επιπλέον συνθήκες, π.χ. τις τρίτες παραγώγους στα άκρα. Σημειώστε ότι τα αυλακώματα μπορούν να παραποιήσουν τις συναρτήσεις που περιέχουν συστροφές, ή εάν τα σημεία παρουσιάζουν στατιστικές διακυμάνσεις γύρω από τις τιμές της συνάρτησης.

Γ.3 ΑΡΙΘΜΗΤΙΚΗ ΟΛΟΚΛΗΡΩΣΗ (τετραγωνισμός)

Θα συζητήσουμε διάφορες μεθόδους για τον αριθμητικό υπολογισμό ολοκληρωμάτων. Για ένα ολοκλήρωμα:

$$I = \int_a^b f(x)dx$$

όλες αυτές οι μέθοδοι μπορούν να γραφτούν στη μορφή:

$$I \approx \sum_i A_i f(x_i) \begin{cases} A_i & \beta\acute{\alpha}\rho\eta \\ x_i & \sigma\eta\mu\epsilon\acute{\iota}\alpha \end{cases}$$

Όπως αναφέρεται στην εισαγωγή, όσο περισσότερα σημεία χρησιμοποιούμε, τόσο καλύτερη η ακρίβεια και τόσο μεγαλύτερος ο χρόνος υπολογισμού. Έτσι προσπαθούμε να επιλέξουμε τα σημεία, x_i, όσο το δυνατόν αποτελεσματικότερα καθώς και να δούμε εάν μπορούμε να βρούμε μερικά τεχνάσματα για να μειώσουμε τον αριθμό των σημείων χωρίς μείωση της ακρίβειας. Εδώ θα ερευνήσουμε τις παρακάτω δυνατότητες:

- Ισαπέχοντα x_i.
- Γκαουσσιανός τετραγωνισμός, όπου τα x_i είναι ρίζες πολυωνύμων.
- Monte Carlo, όπου τα x_i επιλέγονται τυχαία.

Μια άλλη επιλογή είναι να χρησιμοποιηθούν μέθοδοι για τις συνήθεις διαφορικές εξισώσεις.

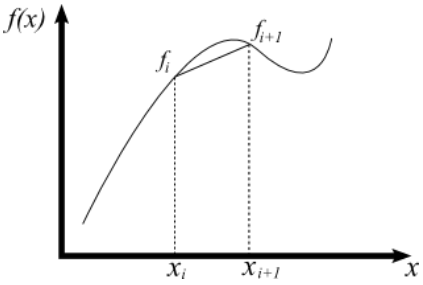
Γ.3.1 Ισαπέχοντα x_i

Θεωρούμε

$$I = \int_{x_1}^{x_N} f(x)dx, \;\; f_i = f(x_i), \;\; x_{i+1} = x_i + h, \;\; i = 1, \dots, N-1$$

Ο τραπεζοειδής κανόνας

Χρησιμοποιούμεστε γραμμική προσέγγιση της $f(x)$ σε κάθε διάστημα $[x_i, x_{i+1}]$ και συνεπώς το ολοκλήρωμα σε κάθε τέτοιο διάστημα δίνεται από το εμβαδό του παραλληλόγραμμου χωρίου:



$$\int_{x_i}^{x_{i+1}} f(x)dx \approx hf_i + \frac{1}{2}h(f_{i+1} - f_i) = \frac{1}{2}h(f_i + f_{i+1})$$

Από το ανάπτυγμα $f(x) = f_i + (x - x_i)f'_i + O(h^2) = f_i + (x - x_i)\frac{f_{i+1} - f_i}{h} + O(h^2)$ ακολουθεί ότι το σφάλμα αποκοπής στην $f(x)$ είναι $O(h^2)$. Αυτό στη συνέχεια σημαίνει ότι το σφάλμα στον υπολογισμό του ολοκληρώματος είναι $O(h^3)$ (το μήκος του διαστήματος είναι h). Πρόσθεση των επιμέρους διαστημάτων δίνει $O(1/h)$ ώστε τελικά το συνολικό σφάλμα να είναι $O(h^2)$.

Ο κανόνας Simpson

Αυτή η μέθοδος είναι παρόμοια, αλλά χρησιμοποιεί δευτέρου βαθμού πολυώνυμα αντί πρώτου βαθμού. Αυτό σημαίνει ότι απαιτούνται τρία σημεία για την παρεμβολή, μιας και τα μελετούμενα διαστήματα θα είναι της μορφής $[x_i, x_{i+2}]$. Η παρεμβολή Lagrange και η ολοκλήρωση δίνουν:

$$\int_{x_i}^{x_{i+2}} f(x)dx \approx \frac{1}{3}h(f_i + 4f_{i+1} + f_{i+2})$$

Το ίδιο είδος εκτίμησης σφάλματος όπως ανωτέρω δείχνει ότι το λάθος αποκοπής είναι $O(h^4)$ σε αυτήν την περίπτωση. Εντούτοις, ότι

το πραγματικό λάθος είναι μικρότερο από αυτό, $O(h^5)$, χάρι στην ακύρωση.

Μέχρι τώρα, εξετάσαμε υποδιαστήματα. Το πλήρες ολοκλήρωμα λαμβάνεται με την πρόσθεση των αποτελεσμάτων για τα υποδιαστήματα. Εξετάζουμε, για απλότητα, τον τραπεζοειδή κανόνα.

Κλειστός τύπος

Εφαρμόζουμε το ανωτέρο αποτέλεσμα σε κάθε ένα από τα υποδιαστήματα, το οποίο σημαίνει ότι τα άκρα x_1 και x_N χρησιμοποιούνται. Αυτό δίνει $\int_{x_1}^{x_N} f(x)dx \approx h\left(\frac{1}{2}f_1 + f_2 + \cdots + f_{N-1} + \frac{1}{2}f_N\right)$. Το σφάλμα αποκοπής δεν μποτρί να είναι χειρότερο από $\sim Nh^3 \sim h^2$ (το συνολικό μήκος $x_N - x_1 = (N-1)h$ είναι σταθερό).

Ανοικτός τύπος

Εδώ, για να αποφύγουμε τα άκρα, τα δύο διαστήματα των άκρων θα αντιμετωπιστούν διαφορετικά. Αυτό είναι απαραίτητο εάν η f έχει μια ιδιομορφία στα άκρα (π.χ. $\int_0^1 \frac{dx}{\sqrt{x}}$), ή έχει οριακή τίμή σε κάποιο άκρο (π.χ. $\sin(x)/x$ στο 0). Για τα διαστήματα των άκρων, οι ακόλουθες εκτιμήσεις μπορούν να χρησιμοποιηθούν: $\int_{x_1}^{x_2} f(x)dx \approx hf_2 + O(h^2)$, $\int_{x_{N-1}}^{x_N} f(x)dx \approx hf_{N-1} + O(h^2)$.

Μέθοδος Romberg (τραπεζοειδείς κανόνας + παρέκταση Richardson)

Η μέθοδος του Romberg είναι βασισμένη στον τύπο αθροίσματος Euler-Maclaurin, τον οποίο αναφέρουμε χωρίς απόδειξη. Ας είναι:

$$T(x) = h\left(\frac{1}{2}f_1 + f_2 + \cdots + f_{N-1} + \frac{1}{2}f_N\right)$$

ο κλειστός τραπεζοειδής κανόνας. Ο τύπος αθροίσματος Euler-Maclaurin δηλώνει ότι:

$$T(h) - \int_{x_1}^{x_N} f(x)dx = c_1h^2 + c_2h^4 + \cdots$$

όπου

$$\begin{cases} c_k = \dfrac{B_{2k}}{(2k)!}\left(f^{(2k-1)}(x_N) - f^{(2k-1)}(x_1)\right) \\ \quad B_n \text{ είναι οι αριθμοί Bernoulli,} \\ \quad \delta\eta\lambda.\ B_n = \dfrac{d^n}{dt^n}\left(\dfrac{t}{e^t - 1}\right)\Big|_{t=0} \end{cases}$$

Αυτό το αποτέλεσμα δείχνει ότι το σφάλμα αποκοπής του τραπεζοειδούς κανόνα περιέχει μόνο άρτιες δυνάμεις του h. Εάν χρησιμοποιησούμε την παρέκταση Richardson, θα έχουμε κέρδος δύο δυνάμεις του h σε κάθε βήμα. Η κατάσταση είναι ακριβώς η ίδια όπως για την προσέγγιση κεντρικής διαφοράς της παραγώγου, έτσι αμέσως ακολουθεί:

$$\tilde{T}(h) = \frac{4T(h) - T(2h)}{3} = \int_{x_1}^{x_N} f(x)dx + O(h^4)$$

Τραπεζοειδής κανόνας + 1 βήμα Richardson = κανόνας Simpson.

Για να το δούμε αυτό, θεωρούμε αυθαίρετο διάστημα$[x_i, x_{i+2}]$:

$$\begin{cases} T(2h) = h(f_i + f_{i+2}) \\ T(h) = h(\frac{1}{2}f_i + f_{i+1} + \frac{1}{2}f_{i+2}) \end{cases} \Rightarrow \tilde{T}(h) = \underbrace{\frac{h}{3}(f_i + 4f_{i+1} + f_{i+2})}_{Τύπος\ κανόνα\ Simpson}$$

Γ.3.2 Γκαουσιανός τετραγωνισμός

Ας είναι f ένα πολυώνυμο βαθμού $< N$. Ο τύπος παρεμβολής του Lagrange μας πληροφορεί

$$f(x) = \sum_{i=1}^{N} L_i\,(x) f(x_i),\ \ L_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

για όλες τις πιθανές επιλογές των $x_1, \dots, x_N$ με $x_i \neq x_j$ αν $i \neq j$. Αυτό σημαίνει ότι, για όλες αυτές τις f ο τύπος ολοκλήρωσης:

$$\int_a^b f(x)w(x) \approx \sum_{i=1}^{N} f(x_i) \underbrace{\int_a^b L_i(x)w(x)dx}_{\equiv A_i}$$

είναι ακριβής. Όπου το $w(x) \geq 0$ είναι μια αυθαίρετη συνάρτηση «βάρους» και μπορεί, φυσικά να τεθεί ίση με τη μονάδα, αλλά μπορεί επίσης να τεθεί κάποια άλλη συνάρτηση έτσι ώστε να απαλοίφεται κάποια ιδιομορφία, όπως για παράδειγμα για μία συνάρτηση που συμπεριφέρεται σαν $1/\sqrt{x}$ καθώς $x \to 0$, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση βάρους $w(x) = 1/\sqrt{x}$ και να αντικαταστήσουμε την $f(x)$ με την $\sqrt{x}f(x)$ που μπορεί να προσεγγιστεί καλύτερα από ένα πολυώνυμο.

Στο Γκαουσιανό τετραγωνισμό, αυτός ο τύπος γίνεται ακριβής για όλα τα πολυώνυμα βαθμού $< 2N$, με μια προσεκτική επιλογή των x_i. Για να δούμε πώς λειτουργεί η μέθοδος, πρέπει να ορίσουμε πότε δύο συναρτήσεις θα ονομάζονται ορθογώνιες. Ορίζουμε το βαθμωτό προϊόν δύο αυθαίρετων συναρτήσεων f και g

$$\langle f|g\rangle = \int_a^b f(x)g(x)w(x)dx$$

και λέμε ότι οι συναρτήσεις f και g είναι ορθογώνιες αν $\langle f|g\rangle = 0$. Για οποιοδήποτε δεδομένο βαθμωτό γινόμενο, είναι δυνατόν να κατασκευάστει μια ακολουθία ορθογώνιων πολυωνύμων $\varphi_0, \varphi_1, \dots$ βαθμών $0,1, \dots$ αντίστοιχα,με τη χρήση της μεθόδου Gram-Schmidt:

- $\varphi_0(x) = 1$
- $\varphi_1(x) = x + a\varphi_0(x),\ \langle\varphi_1|\varphi_0\rangle = 0 \Rightarrow a = -\langle\varphi_0|x\rangle/\langle\varphi_0|\varphi_0\rangle$
- $\varphi_2(x) = x^2 + b\varphi_1(x) + c\varphi_0(x),\ \langle\varphi_2|\varphi_1\rangle = \langle\varphi_2|\varphi_0\rangle = 0 \Rightarrow \{b = -\frac{\langle\varphi_1|x^2\rangle}{\langle\varphi_1|\varphi_1\rangle}\ \kappa\alpha\iota\ c = -\frac{\langle\varphi_0|x^2\rangle}{\langle\varphi_0|\varphi_0\rangle}\}$

- ...

Υποθέτουμε ότι φ_N είναι το N-οστού βαθμού πολυώνυμο που λάβαμε με τη μέθοδο Gram-Schmidt. Ο Γκαουσιανός τετραγωνισμός λέει ότι αν τα $x_1, \dots, x_N$ επιλεχθούν σαν ρίζες των πολυωνύμων φ_N ο τύπος ολοκλήρωσης:

$$\int_a^b f(x)w(x) \approx \sum_{i=1}^{N} A_i f(x_i)$$

είναι ακριβής για όλα τα πολυώνυμα βαθμού $< 2N$.

Απόδειξη

Ας είναι f ένα αυθαίρετο πολυώνυμο βαθμού $< 2N$. Μπορούμε έπειτα να γράψουμε $f = q\varphi_N + r$ για κάποια πολυώνυμα q, r βαθμού $< N$. Ακολουθεί ότι:

$$\int fw = \int q\varphi_N w + \int rw$$

όπου $\int q\varphi_N w = 0$ επειδή το q είναι γραμμικός συνδυασμός των$\varphi_1, \dots, \varphi_{N-1}$, τα οποία είναι ορθογώνα στο φ_N. Ο δεύτερος όρος στο δεξιό μέλος μπορεί να γραφτεί:

$$\int rw = \sum_{i=1}^{N} A_i r(x_i)$$

επειδή ξέρουμε ότι αυτός ο τύπος είναι ακριβής για τα πολυώνυμα βαθμού $< N$. Αλλά:

$$\sum_{i=1}^{N} A_i r(x_i) = \sum_{i=1}^{N} A_i f(x_i) - \sum_{i=1}^{N} A_i q(x_i) \underbrace{\varphi_N(x_i)}_{0}$$

Όπου $\varphi_N(x_i) = 0$ λόγω της επιλογής των x_i. Αυτό ολοκληρώνει την απόδειξη.

Παράδειγμα

Καθορίστε τα x_1, x_2, A_1, A_2 έτσι ώστε ο τύπος

$$\int_{-1}^{1} f(x)dx \approx A_1 f(x_1) + A_2 f(x_2)$$

να είναι ακριβής για όλα τα πολυώνυμα βαθμού < 4:

Λύση

Χρήσιμοποιούμε Γκαουσιανό τετραγωνισμό με $N = 2$. Το σχετικό βαθμωτό γινόμενο είναι:

$$\langle f|g\rangle = \int_{-1}^{1} f(x)g(x)dx\,,\ \ w(x) = 1$$

Τα ορθογώνια πολυώνυμα είναι γνωστά σε αυτήν την περίπτωση. Ονομάζονται πολυώνυμα Legendre και τα πρώτα τρία είναι:

$$P_0(x) = 1; P_1(x) = x; P_2(x) = \tfrac{1}{2}(2x^2 - 1)$$

Τα επιθυμητά x_1, x_2 είναι οι ρίζες του $P_2(x)$,

$$x_{1,2} = \pm\frac{1}{\sqrt{3}}$$

Τα βάρη A_1, A_2 μπορούν να καθοριστούν με απλή εκτέλεση των ολοκληρωμάτων στον ανωτέρω ορισμό. Ένας άλλος, ελαφρώς απλούστερος, τρόπος είναι να χρησιμοποιήσουμε το γεγονός ότι ο τύπος ολοκλήρωσης πρέπει να είναι ακριβής για τις συναρτήσεις $f(x) = 1, f(x) = x$, το οποίο δίνει:

$$\left.\begin{aligned} A_1 + A_2 &= \int_{-1}^{1} dx = 2 \\ -\frac{1}{\sqrt{3}}A_1 + \frac{1}{\sqrt{3}}A_2 &= \int_{-1}^{1} x dx = 0 \end{aligned}\right\} \Rightarrow A_1 = A_2 = 1$$

Με αυτά τα x_1, x_2, A_1, A_2, ο τύπος ολοκλήρωσης γίνεται ακριβής για όλα τα πολυώνυμα βαθμού < 4.

Γ.4 ΣΥΝΗΘΕΙΣ ΔΙΑΦΟΡΙΚΕΣ ΕΞΙΣΩΣΕΙΣ

Γ.4.1 Εισαγωγή

Σε αυτό το τμήμα εξετάζουμε την αριθμητική ολοκλήρωση ΣΔΕ. Θα συζητήσουμε τις μεθόδους για επίλυση συστημάτων πρωτοτάξιων ΣΔΕ:

$$\frac{dx_1}{dt} = f_1(t, x_1, \dots, x_n)$$
$$\vdots$$
$$\frac{dx_n}{dt} = f_n(t, x_1, \dots, x_n)$$

για δεδομένες αρχικές συνθήκες. Σε διανυσματική μορφή, αυτό το πρόβλημα αρχικών τιμών μπορεί να γραφτεί:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}), \;\; \mathbf{x}(t_0) = \mathbf{x_0}$$

Η μελέτη μόνο πρωτοτάξιων ΣΔΕ δεν είναι ισχυρός περιορισμός, επειδή οποιαδήποτε n-τάξια ΣΔΕ μπορεί να γραφτεί ως σύστημα πρωτοτάξιων ΣΔΕ. Για ευκολία, θα υποθέτουμε συχνά ότι $n = 1$. Αυτό δεν σημαίνει ότι αυτές οι μέθοδοι λειτουργούν μόνο για $n = 1$, αντίθετα, η γενίκευση για $n > 1$ είναι χαρακτηριστικά εύκολη.

Παράδειγμα

Εξετάστε τη μονοδιάστατη εξίσωση του Νεύτωνα $m\frac{d^2x}{dt^2} = -V'(x)$. Με την αντικατάσταση $x_1 = x, x_2 = dx/dt$, λαμβάνουμε τις πρωτοτάξιες ΣΔΕ:

$$\begin{cases} \dfrac{dx_1}{dt} = x_2 \\ \dfrac{dx_2}{dt} = -V'(x_1)/m \end{cases}$$

οι όποιες είναι οι αντίστοιχες εξισώσεις Hamilton.

Γ.4.2 Μέθοδος Euler

Θεωρούμε την πρωτοτάξια ΣΔΕ:

$$\frac{dx}{dt} = f(t, x)$$

Για να λύσουμε αριθμητικά αυτή την εξίσωση διακριτοποιούμε το t με χρήση ενός βήματος h,

$$t_n = t_0 + nh, \;\; n = 0,1,2, \ldots$$
$$x_n = x(t_n)$$

προσεγγίζοντας την παράγωγο με την πεπερασμένη διαφορά:

$$\frac{dx}{dt} = \frac{x_{n+1} - x_n}{h} + O(h)$$

λαμβάνουμε την επαναληπτική (ή ανδρομική) σχέση:

$$x_{n+1} = x_n + hf(t_n, x_n)$$

Αυτή είναι η μέθοδος του Euler. Το τοπικό σφάλμα αποκοπής είναι $\sim h^2$ (ένα βήμα), το οποίο κάνει το καθολικό σφάλμα αποκοπής $\sim h$. Μία μέθοδος της οποίας το καθολικό σφάλμα είναι $\sim h^n$, ονομάζονται μέθοδοι n-οστής τάξης, έτσι η μέθοδος του Euler είναι $1^{ης}$ τάξης. Οι μέθοδοι χαμηλής τάξης απαιτούν μικρά βήματα, τα οποία κάνουν τον απαραίτητο αριθμό βημάτων μεγάλο. Επομένως, το υπολογιστικό κόστος τείνει να είναι υψηλό.

Γ.4.3 Ανάπτυγμα Taylor

Ένας απλός τρόπος να μειωθεί το σφάλμα αποκοπής είναι να χρησιμοποιηθεί το ανάπτυγμα Taylor:

$$x(t_n + h) = x_n + h\frac{dx}{dt}\Big|_{t=t_n} + \frac{h^2}{2}\frac{d^2x}{dt^2}\Big|_{t=t_n} + \cdots$$

όπου $\frac{dx}{dt} = f$ και:

$$\frac{d^2x}{dt^2} = \frac{df}{dt} = \partial_t f + \partial_x f \cdot f$$

Κρατώντας τους πρώτους τρεις όρους λαμβάνουμε:

$$\begin{aligned} x(t_n + h) = x_n &+ hf(t_n, x_n) \\ &+ \frac{h^2}{2}[\partial_t f(t_n, x_n) + \partial_x f(t_n, x_n) \cdot f(t_n, x_n)] \\ &+ O(h^3) \end{aligned}$$

η όποια είναι μια μέθοδος δεύτερης τάξης. Σαφώς, η ίδια διαδικασία μπορεί να χρησιμοποιηθεί για να λάβουμε μεθόδους υψηλότερης τάξης. Αυτή η προσέγγιση έχει, εντούτοις, το μειονέκτημα ότι όλο και υψηλότερης τάξης παράγωγοι απαιτούνται.

Γ.4.4 Η μέθοδος Runge-Kutta

Η μέθοδος Runge-Kutta αποφεύγει αυτό το πρόβλημα – καμία παράγωγος δεν απαιτείται . Αντ' αυτού, η εκτίμηση x_{n+1} κατασκευάζεται χρησιμοποιώντας τις τιμές της f σε προσεκτικά επιλεγμένα σημεία μεταξύ των x_n και x_{n+1}. Με αύξηση του αριθμού των σημείων, η τάξη της μεθόδου μπορεί να αυξηθεί. Υπάρχουν διάφορες μορφές της μεθόδου.

Ένα p-σχήμα της μεθόδου για το πρόβλημα αρχικών τιμών που ορίσαμε παραπάνω είναι:

$$\begin{aligned} t_{n+1} &= t_n + h \\ x_{n+1} &= x_n + \omega_1 k_1 + \omega_2 k_2 + \cdots + \omega_p k_p \end{aligned}$$

όπου τα ω_i ονομάζονται σταθερές βάρους και:

$$k_1 = hf(t_n, x_n)$$
$$k_2 = hf(t_n + c_2 h, x_n + a_{21} k_1)$$
$$k_3 = hf(t_n + c_3 h, x_n + a_{31} k_1 + a_{32} k_2)$$
$$\vdots$$
$$k_p = hf(t_n + c_p h, x_n + a_{p1} k_1 + a_{p2} k_2 + \cdots + a_{p,p-1} k_{p-1})$$

Στην παραπάνω εξίσωση τα $c_2, \dots, c_p, a_{21}, \dots, a_{p,p-1}$ είναι γνωστές σταθερές. Η αναπαράσταση των μεθόδων Runge-Kutta σε πινακική μορφή (πίνακας Butcher) είναι:

$$\left\{ \begin{array}{c|c} \vec{c} & \mathbf{A} \\ \hline & \vec{\omega} \end{array} \right\}$$

όπου ο πίνακας $\mathbf{A} = \{a_{ij}\}$, $\forall i > j$ και 0 διαφορετικά (κάτω τριγωνικός), $\vec{c}$ το διάνυσμα στήλη $c_i, i = 2, \dots, p$ και $\vec{\omega}$ το διάνυσμα γραμμή $\omega_j, j = 1, \dots, p$. Πιο ειδικά ας δούμε το παράδειγμα μιας $2^{ης}$ τάξης Runge-Kutta (τοπικό σφάλμα $\sim h^3$). Το 2-σχήμα είναι:

$$k_1 = hf(t_n, x_n)$$
$$k_2 = hf(t_n + mh, x_n + mk_1)$$
$$x_{n+1} = x_n + \omega_1 k_1 + \omega_2 k_2$$

Θέλουμε να προσδιορίσουμε όλες τις σταθερές ώστε να ελαχιστοποιήσουμε το σφάλμα, για αυτό παίρνουμε το ανάπτυγμα Taylor της x_{n+1} γύρω από το t_n:

$$k_2 = hf(t_n, x_n) + mh^2 \partial_t f(t_n, x_n) + mh^2 f(t_n, x_n) \partial_x f(t_n, x_n) + O(h^3) \Rightarrow$$
$$x_{n+1} = x_n + (\omega_1 + \omega_2) hf(t_n, x_n) + \omega_2 mh^2 (\partial_t f(t_n, x_n) + f(t_n, x_n) \partial_x f(t_n, x_n)) + O(h^3)$$

Με σύγκριση αυτής με το ανάπτυγμα Taylor της ακριβούς λύσης $x(t_n + h)$ παρατηρούμε ότι το τοπικό σφάλμα είναι $O(h^3)$ αν:

$$\omega_1 + \omega_2 = 1 \; και \; m\omega_2 = 1/2$$

Για παράδειγμα η μέθοδος Heun απαιτεί $\omega_1 = \omega_2 = 1/2 \; και \; m = 1$.

Γ.4.5 Προσαρμογή μεγέθους βημάτων

Μέχρι τώρα, έχουμε υποθέσει ότι το μέγεθος h των βημάτων είναι σταθερό κατά την ολοκλήρωση. Εντούτοις, η συνάρτηση $f(t,x)$ μπορεί να είναι πολύ διαφορετική στα διαφορετικά μέρη του διαστήματος. Κατά συνέπεια, μπορεί να υπάρχουν «εύκολες» περιοχές όπου ένα μεγάλο βήμα μπορεί να χρησιμοποιηθεί καθώς και «δύσκολες» όπου ένα πολύ μικρότερο μέγεθος βημάτων απαιτείται. Επομένως, είναι συχνά πολύ χρήσιμο να μεταβάλουμε το μέγεθος των βημάτων. Μια φυσική επιλογή είναι να μεταβάλουμε το μέγεθος των βημάτων κατά τέτοιο τρόπο ώστε το τοπικό σφάλμα να κρατιέται σε σταθερό επίπεδο. Για αυτό, υπάρχει η ανάγκη να έχουμε μια εκτίμηση του τοπικού σφάλματος. Ένας κατάλληλος τρόπος να υπολογιστεί το τοπικό σφάλμα είναι με διπλασιασμό του μεγέθους των βημάτων. Αυτό σημαίνει ότι επαναλαμβάνουμε όλους τους υπολογισμούς χρησιμοποιώντας διπλάσιο μέγεθος βημάτων. Με σύγκριση των δύο υπολογισμών, παίρνουμε μια εκτίμηση του τοπικού σφάλματος. Για να δούμε πώς γίνεται αυτό, θεωρούμε την $4^{ης}$ τάξης Runge-Kutta, για την οποία το τοπικό σφάλμα κλιμακώνεται ως h^5. Ας είναι $x(t+h;h/2)$ και $x(t+h;h)$ οι οι εκτιμήσεις της ακριβούς λύσης $x(t+h)$ που λαμβάνεται με τη χρήση δύο βημάτων μεγέθους $h/2$ και ενός βήματος μέγεθους h, αντίστοιχα. Για μικρά h, αυτές οι εκτιμήσεις πρέπει να συμπεριφερθούν:

$$x\left(t+h;\frac{h}{2}\right) - x(t+h) \sim \left(\frac{h}{2}\right)^5 c(t) + O(h^6)$$
$$x(t+h;h) - x(t+h) \sim h^5 c(t) + O(h^6)$$

όπου το $c(t)$ είναι κάποια άγνωστη συνάρτηση. Η ποσότητα $\Delta = |x\left(t+h;\frac{h}{2}\right) - x(t+h;h)|$ παρέχει μια κατά προσέγγιση εκτίμηση του σφάλματος στην $x(t+h;h)$. Για να ελέγξουμε το τοπικό σφάλμα, μπορούμε να απαιτήσουμε $\Delta < \Delta_{tol}$, όπου Δ_{tol} είναι το προκαθορισμένο όριο ανοχής. Εάν $\Delta > \Delta_{tol}$, ξανακάνουμε τον

υπολογισμό χρησιμοποιώντας μικρότερο μέγεθος h'. Το πόσο μικρό πρέπει το νέο βήμα υπολογίζεται με χρήση του ότι $\Delta \sim h^5$. Αυτό δίνει:

$$h' \approx S\left(\frac{\Delta_{tol}}{\Delta}\right)^{1/5} h$$

όπου h είναι το βήμα που μας έδωσε Δ. Το S είναι ένας παράγοντας «ασφάλειας» που υποθέτουμε ότι είναι μικρότερος της μονάδας. Εάν $\Delta \ll \Delta_{tol}$, το μέγεθος των βημάτων πρέπει να αυξηθεί. Το πόσο πρέπει να αυξηθεί υπολογίζεται πάλι απο το ότι $\Delta \sim h^5$. Το κέρδος από τη χρήση μεθόδων με προσαρμογή βήματος μπορεί να είναι τεράστιο σε σχέση με το κόστος των επιπρόσθετων υπολογισμών που απαιτούνται για να παρακολουθούμε το τοπικό σφάλμα.

Γ.4.6 Η τροποποιημένη μέθοδος μεσαίου σημείου

Θεωρούμε την ίδια πρωτοτάξια ΣΔΕ, όπως και στα προηγούμενα,

$$\dot{x} = f(t, x)$$

Η προσέγγιση κεντρικής διαφοράς της παραγώγου,

$$\left.\frac{dx}{dt}\right|_{t=t_n} = \frac{x_{n+1} - x_{n-1}}{2h} + \mathcal{O}(h^2)$$

μας δίνει την αποκαλούμενη μέθοδο μεσαίου σημείου,

$$x_{n+1} = x_{n-1} + 2hf(t_n, x_n)$$

με τοπικό σφάλμα $\mathcal{O}(h^3)$. Αυτή είναι μία μέθοδος «δύο-σημείων», στην οποία απαιτούνται τα x_n και x_{n-1} προκειμένου να ληφθεί η τιμή x_{n+1}. Η τροποποιημένη μέθοδος μεσαίου σημείου έχει τρία συστατικά:

- Ένα αρχικό βήμα Euler για να πάρουμε τη δεύτερη αρχική τιμή:

$$x_1 = x_0 + hf(t_0, x_0)$$

- $N-1$ βήματα με τη μέθοδο μεσαίου σημείου:

$$x_{n+1} = x_{n-1} + 2hf(t_n, x_n), \;\; n = 1, \dots, N-1$$

- Μια «διόρθωση» της τελευταίας τιμής x_N:

$$x(t_0 + H; h) = \frac{1}{2}(x_N + x_{N-1} + hf(t_N, x_N))$$

όπου $H = Nh$. $x(t_0 + H; h)$ είναι η τελική εκτίμηση.

Για σταθερά t_0 και H καθώς και άρτιο N, έχει δειχθεί (από τον Gragg) ότι:

$$x(t_0 + H; h) - x(t_0 + H) = c_1 h^2 + c_2 h^4 + \cdots$$

Με άλλα λόγια, το σφάλμα αποκοπής περιέχει μόνο άρτιες δυνάμεις του h^2, αυτή η ιδιότητα κάνει τη μέθοδος κατάλληλη για εφαρμογή της παρέκτασης Richardson. Η αποκαλούμενη μέθοδος Bulirch-Stoer είναι βασισμένη στην τροποποιημένα μέθοδο μεσαίου σημείου και μία παρέκταση παρόμοια με αυτή του Richardson (αλλά με ρητές συναρτήσεις αντί πολυωνύμων). Αυτή η μέθοδος είναι μια καλή επιλογή εάν απαιτείται υψηλή ακρίβεια.

Γ.4.7 Μέθοδοι πρόβλεψης-διόρθωσης

Όλες οι μέθοδοι που συζητήθηκαν μέχρι τώρα είναι αναλυτικές δεδομένου ότι το x_{n+1} μπορούσε να υπολογιστεί με έναν άμεσο τρόπο. Υπάρχουν επίσης υποκρυπτόμενες μέθοδοι, στις οποίες μια εξίσωση πρέπει να λυθεί προκειμένου να ληφθεί η τιμή x_{n+1}. Οι υποκρυπτόμενες μέθοδοι έχουν τα πλεονεκτήματά τους, όπως θα δειξει η κατωτέρω συζήτηση γύρω απο την ευστάθεια. Ένα απλό παράδειγμα μίας υποκρυπτόμενης μεθόδου λαμβάνεται με την εφαρμογή του τραπεζοειδούς κανόνα ολοκλήρωσης στο πρότυπο παράδειγμα μας.

$$x_{n+1} - x_n = \int_{t_n}^{t_{n+1}} f(t,x)dx \approx \frac{h}{2}[f(t_n, x_n) + f(t_{n+1}, x_{n+1})]$$

Αυτή η εξίσωση μπορεί να λυθεί ως προς x_{n+1} με χρήση της μεθόδου του Νεύτωνα για την εύρεση ρίζών. Μια άλλη δυνατότητα είναι να χρησιμοποιηθεί η συναρτησιακή επανάληψη. Ένα απλό σχήμα μεθόδου πρόβλεψης-διόρθωσης είναι:

- Πρόβλεψη: $x^{(0)} = x_n + hf(t_n, x_n)$
- Διόρθωση: $x_{n+1} = x^{(1)} = x_n + \frac{h}{2}[f(t_n, x_n) + f(t_{n+1}, x_{n+1})]$

Μια ευρέως χρησιμοποιούμενη μέθοδος πρόβλεψης-διόρθωσης είναι αυτή των Adams-Bashforth-Moulton.

Γ.4.8 Ευστάθεια

Εάν $n > 1$, μπορεί να συμβεί οι διαφορετικές συνιστώσες x_i να εξελίσσονται με πολύ διαφορετικές ταχύτητες. Το πρόβλημα τότε καλείται δύσκαμπτο (stiff). Τα δύσκαμπτα προβλήματα είναι γενικά δύσκολο να λυθούν.

Παράδειγμα

Θεωρούμε το γραμμικό σύστημα διαφορικών εξισώσεων:

$$\frac{d\mathbf{x}}{dt} = -\mathbf{A}\mathbf{x}$$

μαζί με τις αρχικές συνθήκες $\mathbf{x}(0) = \mathbf{x_0}$, όπου $\mathbf{x}$ ένα διάνυσμα δύο συνιστωσών και $\mathbf{A}$ ένας 2×2 πίνακας. Υποθέτουμε επιπλέον ότι:

$$\mathbf{A}\mathbf{v_i} = \lambda_i \mathbf{v_i}, \;\; i = 1,2, \;\; \lambda_2 \gg \lambda_1 > 0$$

και $\mathbf{x_0} = c_1\mathbf{v_1} + c_2\mathbf{v_2}$ για κάποια c_1, c_2. Η αναλυτική λύση δίνεται έπειτα από:

$$\mathbf{x}(t) = c_1 e^{-\lambda_1 t}\mathbf{v_1} + c_2 e^{-\lambda_2 t}\mathbf{v_2}$$

όπως μπορεί να ελεγχθεί εύκολα. Εδώ, ο δεύτερος όρος σβήνει πολύ γρηγορότερα από τον πρώτο, επειδή $\lambda_2 \gg \lambda_1$. Κατά συνέπεια, ο δεύτερος όρος δεν είναι πολύ συναφής εάν ενδιαφερόμαστε για χρονικές κλίμακες $t \geq 1/\lambda_1$. Υποθέστε ότι θέλουμε να λύσουμε αυτό το πρόβλημα αριθμητικά με τη μέθοδο του Euler. Αυτή μας δίνει την αναδρομική σχέση:

$$\mathbf{x_{n+1}} = \mathbf{x_n} + h(-\mathbf{A}\mathbf{x_n}) = (\mathbf{1} - h\mathbf{A})\mathbf{x_n}$$

η οποία μπορεί εύκολα να λυθεί:

$$\mathbf{x_n} = (\mathbf{1} - h\mathbf{A})^n \mathbf{x_0} \Rightarrow$$
$$\mathbf{x_n} = c_1(\mathbf{1} - h\mathbf{A})^n \mathbf{v_1} + c_2(\mathbf{1} - h\mathbf{A})^n \mathbf{v_2} \Rightarrow$$
$$\mathbf{x_n} = c_1(1 - h\lambda_1)^n \mathbf{v_1} + c_2(1 - h\lambda_2)^n \mathbf{v_2}$$

Εάν πάρνουμε το όριο $h \to 0$ για σταθερό $t = nh$, πρέπει να ανακτήσουμε την ακριβή λύση, και αυτή είναι πράγματι, επειδή:

$$(1 - h\lambda_i)^n = (1 - \lambda_i t/n)^n \xrightarrow{n\to\infty} e^{-\lambda_i t}$$

Η ευστάθεια απαιτεί $|1 - h\lambda_i| \leq 1$ για $i = 1,2$ (ειδάλλως δύο λύσεις που αντιστοιχούν σε ελαφρώς διαφορετικές αρχικές συνθήκες θα αποκλίνουν εκθετικά), και για να αληθεύει αυτό είναι απαραίτητο να πάρουμε το h πολύ μικρό εάν η ιδιοτιμή λ_2 είναι μεγάλη,

$$1 - h\lambda_2 \geq -1 \Rightarrow h \leq 2/\lambda_2$$

Υποθέστε τώρα ότι θέλουμε να μελετήσουμε τη συμπεριφορά σε μεγάλες χρονοκλίμακες, $t \geq 1/\lambda_1$. Ο αριθμός των απαιτούμενων βημάτων θα είναι πολύ μεγάλος, $t/h \geq \lambda_2/2\lambda_1$. Σημειώστε ότι είναι η ταχέως αποσβένουσα και «χωρίς ενδιαφέρον» συνιστώσα που υπαγορεύει ποιο το μέγεθος των βημάτων.

Η πεπλεγμένη μέθοδος Euler

Εάν αντί μιας ευθείας διαφοράς χρησιμοποιούμε μια οπίσθια διαφορά για να προσεγγίσουμε την παράγωγο,

$$\left.\frac{d\mathbf{x}}{dt}\right|_{t=t_n} \approx \frac{\mathbf{x_n} - \mathbf{x_{n-1}}}{h}$$

λαμβάνουμε την αποκαλούμενη πεπλεγμένη μέθοδο Euler. Αυτή μπορεί να μοιάζει δευτερεύουσα τροποποίηση, αλλά οι ιδιότητες ευστάθειας αλλάζουν δραστικά. Ο αναδρομικός τύπος για την πεπλεγμένη μέθοδο Euler είναι:

$$\mathbf{x_n} = \mathbf{x_{n-1}} + h(-\mathbf{A}\mathbf{x_n}) \Rightarrow \mathbf{x_n} = (\mathbf{1} + h\mathbf{A})^{-1}\mathbf{x_{n-1}}$$

η οποία έχει τη λύση:

$$\mathbf{x_n} = c_1(1 + h\lambda_1)^{-n}\mathbf{v_1} + c_2(\mathbf{1} + h\lambda_2)^{-n}\mathbf{v_2}$$

Είναι εύκολο να ελεγχθεί ότι η ακριβής λύση ανακτάται όταν $h \to 0$ για σταθερό $t = nh$. Το σχετικό σφάλμα στο δεύτερο όρο θα είναι μεγάλο εκτός αν το h είναι πολύ μικρό, αλλά αυτή τη φορά το σφάλμα αυτό είναι μικρό σε απόλυτους αριθμούς $-\left|\frac{1}{1+h\lambda_i}\right| \leq 1,\ \forall h > 0$, έτσι τα προβλήματα ευστάθειας έχουν εξαφανιστεί. Αυτό είναι ένα σημαντικό πλεονέκτημα έναντι της προηγούμενης μεθόδου. Το μειονέκτημα είναι ότι πρέπει να λύσουμε μία εξίσωση ως προς $\mathbf{x_n}$ για κάθε n, το οποίο μπορεί να είναι απαγορευτικά χρονοβόρο.

Γ.5 ΔΙΑΦΟΡΙΚΕΣ ΕΞΙΣΩΣΕΙΣ ΜΕΡΙΚΩΝ ΠΑΡΑΓΩΓΩΝ

Γ.5.1 Εισαγωγή

Η συζήτηση μας γύρω από τις ΜΔΕ θα εστιάσει σε τρεις απλές αλλά σημαντικές εξισώσεις:

- εξίσωση θερμότητας (ή διάχυση),
- εξίσωση Poisson και
- εξίσωση κυμάτος.

Η εξίσωση Poisson διαφέρει από τις άλλες δύο δεδομένου ότι δεν υπάρχει χρονική εξάρτηση. Αυτή η εξίσωση λύνεται για δεδομένες

συνοριακές συνθήκες. Η διάχυση και οι εξισώσεις κυμάτων είναι χρονοεξαρτώμενες και μπορούν να λυθούν με εξέλιξη γνωστών αρχικών καταστάσεων δεδομένου συνόρου.

Οι μέθοδοι που συζητούνται είναι μέθοδοι πεπερασμένων διαφορών, στις οποίες το σύστημα είναι τεθειμένο σε χωροχρονικό πλέγμα και οι παράγωγοι προσεγγίζονται από πεπερασμένες διαφορές. Αυτή είναι μια απλή και χρήσιμη προσέγγιση. Μια άλλη προσέγγιση που χρησιμοποιείται ευρέως, είναι η μέθοδος πεπερασμένων στοιχείων. Υπάρχουν επίσης μέθοδοι μεταβολών. Πολλά φυσικά προβλήματα ΜΔΕ μπορούν να επανακατασκευαστούν σαν προβλήματα μεταβολών, όπου η ζητούμενη συνάρτηση $u(\mathbf{x}, t)$ είναι ακρότατο κάποιου ολοκληρώματος I.

Γ.5.2 Η εξίσωση διάχυσης

Θεωρούμε ένα σύστημα σωματιδίων που υποβάλλονται στις τυχαίες συγκρούσεις σε d διαστάσεις. Ας είναι $u(\mathbf{x}, t)$ η πυκνότητα σωματιδίων σε χρόνο $t, (\mathbf{x} \in R^d)$. Η τυχαία κίνηση των σωματιδίων γεννά μια ροή, ή ρεύμα, που δίνεται από:

$$\mathbf{j}(\mathbf{x}, t) = -D(\mathbf{x})\nabla u(\mathbf{x}, t)$$

όπου D είναι ο αποκαλούμενος συντελεστής διάχυσης. Σε ισορροπία, η καθαρή ροή σωματιδίων είναι μηδέν, $\mathbf{j} = 0$, και η κατανομή ομοιόμορφη. Από τη διατήρηση του αριθμού των σωματιδίων ακολουθεί (μέσω του θεωρήματος του Gauss):

$$\frac{\partial u}{\partial t} + \nabla \mathbf{j} = 0 \Rightarrow \frac{\partial u}{\partial t} = \nabla \cdot (D\nabla u)$$

δηλαδή το ρεύμα υπακούει την εξίσωση συνέχειας. Εάν υπάρχει επίσης μία εξωτερική δύναμη $\mathbf{F}(\mathbf{x}) = -\nabla V(\mathbf{x})$ που ενεργεί σωματίδια, όπου $V(\mathbf{x})$ το δυναμικό, αυτά θα έχουν μία ταχύτητα ολίσθησης $\mathbf{v}_d = m\mathbf{F}$, όπου m η κινητικότητα. Η ολίσθηση προκαλεί μία επιπλέον ροή, που περιγράφεται από το ρεύμα $\mathbf{j}_d = u\mathbf{v}_d$ και τελικά το συνολικό ρεύμα θα είναι $\mathbf{j}_t = -D(\mathbf{x})\nabla u(\mathbf{x}, t) + \mathbf{j}_d$. Σε ισορροπία, το ρεύμα ολικό ρεύμα μηδενίζεται,

$$\mathbf{j}_t = -D(\mathbf{x})\nabla u(\mathbf{x},t) + \mathbf{j}_d = 0 \Rightarrow \frac{mu}{D}\mathbf{F} = \nabla u$$

Αλλά για τη u γνωρίζουμε ότι $u \propto e^{-V/kT}$ στην κατάσταση ισορροπίας. Συνδυασμός αυτών των δύο εξισώσεις ισορροπίας δίνει $D = mkT$, η οποία ονομάζεται σχέση Einstein. Ακολουθεί ότι η χρονική εξέλιξη της u παρουσία μίας εξωτερικής δύναμης $\mathbf{F}$ δίνεται από:

$$\frac{\partial u}{\partial t} = \nabla \cdot \left[D\left(\nabla u + \frac{1}{kT} u\mathbf{F} \right) \right]$$

Στη συζήτησή μας, θα κάνουμε την υπόθεση ότι δεν υπάρχει εξωτερική δύναμη, $\mathbf{F} = 0$, καθώς επίσης και ότι ο συντελεστής διάχυσης D είναι σταθερός. Η εξίσωση μπορεί έπειτα να γραφτεί

$$\frac{\partial u}{\partial t} = D\Delta u$$

όπου $\Delta u = \nabla\nabla u = \sum_{i=1}^{d} \partial^2 u / \partial x_i^2$ ο τελεστής Laplace. Μπορούμε επιπλέον να θέσουμε $D = 1$ με κατάλληλη επιλογή των μονάδων.

Γ.5.2.1 Παράδειγμα ακριβούς λύσης

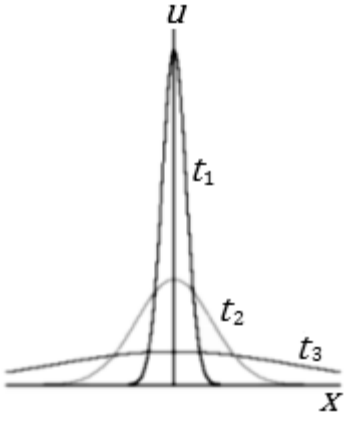
Μπορεί να ελεγχθεί εύκολα ότι η:

$$u(x,t) = \frac{1}{\sqrt{4\pi t}} e^{-x^2/4t}$$

λύνει το μονοδιάστατο πρόβλημα διάχυσης:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Αυτό σημαίνει πώς μία κατανομή που εντοπίζεται αρχικά στο $x = 0$ εξελίσσεται με δυναμική διάχυσης. Η κατανομή είναι Γκαουσσιανή με μέση τιμή $\langle x \rangle = 0, \forall t > 0$ ενώ η διασπορά αυξάνει γραμμικά με το χρόνο, $\langle x^2 \rangle = 2t$:



$$u(x,t) \overset{t\searrow 0}{\longrightarrow} \delta(x)$$

Γ.5.2.2 Ένα απλό σχήμα πεπερασμένων διαφορών

Σαν πρώτο παράδειγμα μίας μεθόδου πεπερασμένων διαφορών για τη μονοδιάστατη εξίσωση διάχυσης, θωρούμε την ακόλουθη προσέγγιση:

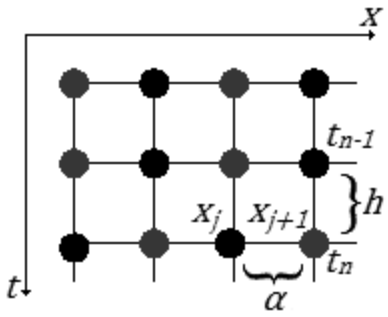
$$u(x_j, t_n) = u_j^n$$
$$\frac{u_j^{n+1} - u_j^n}{h} \approx \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \approx \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{a^2}$$

όπου η πεπερασμένη διαφορά της δεύτερης παραγώγου προκύπτει από τα αναπτύγματα Taylor ως εξής:

$$u(x+a) = u(x) + au'(x) + \frac{a^2}{2}u''(x)$$
$$u(x-a) = u(x) - au'(x) + \frac{a^2}{2}u''(x)$$

που με πρόσθεση κατά μέλη δίνουν:

$$u(x+a) + u(x-a) = 2u(x) + \alpha^2 u''(x) \Rightarrow$$
$$u''(x) = \frac{u(x+a) - 2u(x) + u(x-a)}{\alpha^2}$$



Ας επανέλθουμε στη διακριοποιημένη έκδοση της εξίσωσης μας για να επιλέξουμε $h/a^2 = 1/2$, επιλογή που κάνει την εξίσωση μας ιδιαίτερα απλή,

$$u_j^{n+1} = \frac{1}{2}(u_{j+1}^n + u_{j-1}^n)$$

Υποθέστε ότι αρχίζουμε από μία κατάσταση με $u_j^n = 1$ και όλα τα άλλα μηδέν. Με επανάληψη της διακριτοποιημένης εξίσωσής μας λαμβάνουμε τη λύση:

$$
\begin{array}{cccccccccc}
 & & & & & x \rightarrow & & & & \\
 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
t & 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 \\
\downarrow & 0 & 0 & 1/4 & 0 & 2/4 & 0 & 1/4 & 0 & 0 \\
 & 0 & 1/8 & 0 & 0 & 0 & 3/8 & 0 & 1/8 & 0
\end{array}
$$

η οποία παρουσιάζει παρόμοια συμπεριφορά με την ακριβή λύση ανωτέρω.

Γ.5.2.3 Μία συστηματικότερη προσέγγιση

Λύστε τώρα το ίδιο μονοδιάστατο πρόβλημα κατά τρόπο πιο μεθοδικό, έτσι αρχικά διακριτοποιούμε τη χωρική συντεταγμένη x. Θέτουμε:

$$
u_j(t) = u(x_j, t), \;\; x_j = ja
$$

$$
\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_j} \approx \frac{u_{j+1} - 2u_j + u_{j-1}}{a^2}
$$

Η εξίσωση διάχυσης γίνεται έπειτα ένα σύστημα πρωτοτάξιων ΣΔΕ:

$$
\frac{d\mathbf{u}}{dt} = \mathbf{A}\mathbf{u}
$$

όπου $\mathbf{u}(t)$ ένα διάνυσμα με συνιστώσες $u_j(t)$ και ο πίνακας $\mathbf{A}$ δίνεται απο:

$$
\mathbf{A} = \frac{1}{\alpha^2}\begin{pmatrix}
\ddots & \ddots & \ddots & & & 0 \\
 & 1 & -2 & 1 & & \\
 & & 1 & -2 & 1 & \\
0 & & & \ddots & \ddots & \ddots
\end{pmatrix}
$$

Έχουμε αντιμετωπίσει αυτό τον τύπο προβλημάτων πριν. Τρεις είναι οι πιθανές μέθοδοι για να λύσουμε το παραπάνω σύστημα των εξισώσεων,

- **Η αναλυτική μέθοδος Euler.**

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{h}=\mathbf{A}\mathbf{u}^{n}\Rightarrow\mathbf{u}^{n+1}=(\mathbf{1}+h\mathbf{A})\mathbf{u}^{n}$$

- **Η πεπλεγμένη μέθοδος Euler.**

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{h}=\mathbf{A}\mathbf{u}^{n+1}\Rightarrow\mathbf{u}^{n+1}=(\mathbf{1}-h\mathbf{A})^{-1}\mathbf{u}^{n}$$

- **Ο τραπεζοειδής κανόνας.**

$$\mathbf{u}^{n+1}-\mathbf{u}^{n}=\frac{h}{2}(\mathbf{A}\mathbf{u}^{n}+\mathbf{A}\mathbf{u}^{n+1})\Rightarrow\mathbf{u}^{n+1}$$
$$=\left(\mathbf{1}-\frac{h}{2}\mathbf{A}\right)^{-1}\left(\mathbf{1}+\frac{h}{2}\mathbf{A}\right)\mathbf{u}^{n}$$

αυτή η μέθοδος ονομάζεται και Crank-Nicolson.

Η αναλυτική και η πεπλεγμένη μέθοδος Euler είναι πρωτοτάξιες στο χρόνο ενώ η Crank-Nicolson είναι δεύτερης τάξης. Ξέρουμε, εντούτοις, ότι η τάξη της μεθόδου δεν είναι το μόνο ζήτημα σε ένα πρόβλημα όπως αυτό, πρέπει επίσης να εξετάσουμε την ευστάθεια.

Γ.5.2.4 Ανάλυση ευστάθειας Neumann

Για τη συνεχή εξίσωση διάχυσης $\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}$ είναι δυνατόν να βρούμε λυσεις της μορφής $u(x,t)=T(t)X(x)$. Αυτές οι λύσεις είναι $u=e^{-\omega(k)t}e^{ikx}$, όπου $\omega(k)=k^2$. Για τη διακριτοποιημένη έδοση της εξίσωσης, μπορούμε να ξεκινήσουμε με παρόμοιο τρόπο,

$$\mathbf{u}^{n}=\xi(k)^{n}\tilde{\mathbf{u}}^{(k)}$$

όπου $\xi(k)$ είναι ένας αριθμός που αντιστοιχεί στο χρονικό μέρος $e^{-\omega(k)h}$ και $\tilde{\mathbf{u}}^{(k)}$ ένα διάνυσμα με συνιστώσες $\tilde{u}_j^{(k)}=e^{ikx_j}$ και αντιστοιχεί στο χωρικό μέρος $X(x)=e^{ikx}$. Είναι λογικό να

αναμένουμε το $\tilde{\mathbf{u}}^{(k)}$ να είναι ιδιοδιάνυσμα του πίνακα $\mathbf{A}$, επειδή η $X(x) = e^{ikx}$ είναι ιδιοσυνάρτηση του τελεστή $\partial^2/\partial x^2$ με ιδιοτιμή $-k^2$ και ο $\mathbf{A}$ δεν είναι παρα η διακριτοποιημένη έκδοση αυτού του τελεστή. Ο άμεσος υπολογισμός δείχνει ότι αυτή είναι πράγματι η κατάσταση:

$$(\mathbf{A}\tilde{\mathbf{u}}^{(k)})_j = \frac{1}{a^2}\left(e^{ikx_{j-1}} - 2e^{ikx_j} + e^{ikx_{j+1}}\right) = \underbrace{-\frac{1}{a^2}(2 - e^{-ika} - e^{ika})}_{\lambda_k}\tilde{u}_j^{(k)}$$

Έτσι έχουμε:

$$\mathbf{A}\tilde{\mathbf{u}}^{(k)} = \lambda_k\tilde{\mathbf{u}}^{(k)}, \quad \lambda_k = -\frac{4}{\alpha^2}sin^2(\frac{ka}{2})$$

από όπου παρατηρούμε ότι $\lambda_k \approx -k^2$ αν $ka \ll 1$. Υποθέτοντας ότι η αρχική κατάσταση μπορεί να γραφτεί σαν σύνθεση των λύσεων της μορφής $\mathbf{u}^n = \xi(k)^n\tilde{\mathbf{u}}^{(k)}$, μπορούμε να έχουμε συμπεράσματα για την ευστάθειαγια τις τρεις μεθόδους που προαναφέραμε.

- **Η αναλυτική μέθοδος Euler.**

  $$\xi(k)^{n+1}\tilde{\boldsymbol{u}}^{(k)} = \xi(k)^n(\mathbf{1} + h\mathbf{A})\tilde{\boldsymbol{u}}^{(k)} \Rightarrow \xi(k) = 1 + h\lambda_k$$

  Η ευστάθεια απαιτεί $|\xi(k)| \leq 1 \forall k$, συνεπώς πρέπει $(\lambda_k \leq 0)$:

  $$1 + h\lambda_k \geq -1 \Leftrightarrow = \frac{2h}{\alpha^2}sin^2(\frac{ka}{2}) \leq 1$$

  αυτό αληθεύει για όλα τα k αν $h \leq a^2/2$.

- **Η πεπλεγμένη μέθοδος Euler.**

  $$\xi(k)^{n+1}\tilde{\boldsymbol{u}}^{(k)} = \xi(k)^n(\mathbf{1} - h\mathbf{A})^{-1}\tilde{\boldsymbol{u}}^{(k)} \Rightarrow \xi(k) = (1 - h\lambda_k)^{-1}$$

Εφόσον $\lambda_k \leq 0$ άμεσα καταλήγουμε $0 \leq \xi(k) \leq 1$ για όλα τα k και $h > 0$. Συνεπώς η μέθοδος είναι ευσταθής για $h > 0$.

- **Crank-Nicolson.**

$$\xi(k)^{n+1}\tilde{\mathbf{u}}^{(k)} = \xi(k)^n \left(\mathbf{1} - \frac{h}{2}\mathbf{A}\right)^{-1} \left(\mathbf{1} + \frac{h}{2}\mathbf{A}\right)\tilde{\mathbf{u}}^{(k)} \Rightarrow \xi(k) = \frac{1 + \frac{h}{2}\lambda_k}{1 - \frac{h}{2}\lambda_k}$$

Αυτό σημαίνει $|\xi(k)| \leq 1$ για όλα τα k και $h > 0$, εφόσον $\lambda_k \leq 0$ και $|(1-x)/(1+x)| \leq 1$ για όλα τα $x \geq 0$. Συνεπώς και αυτή η μέθοδος είναι ευσταθής για $h > 0$.

Γ.5.3 Η εξίσωση Poisson

Θεωρούμε την εξίσωση Poisson:

$$\Delta u(\mathbf{x}) = \nabla \cdot \nabla u(\mathbf{x}) = -\rho(\mathbf{x}), \quad \mathbf{x} \in \Omega$$

με τη συνοριακή συνθήκη $u(\mathbf{x}) = f(\mathbf{x}), \mathbf{x} \in \partial\boldsymbol{\Omega}$ (με $\partial\boldsymbol{\Omega}$ συμβολίζουμε το σύνορο). Μία τέτοια συνθήκη που καθορίζει την u στο σύνορο (f είναι γνωστή συνάρτηση) ονομάζεται συνοριακή συνθήκη Dirichlet. Αυτό το πρόβλημα εμφανίζεται σε διάφορα μέρη στη φυσική. Ένα παράδειγμα είναι η ηλεκτροστατική. Το u είναι έπειτα το ηλεκτροστατικό δυναμικό και ρ η πυκνότητα φορτίων.

Υποθέστε ότι θέλουμε να λύσουμε αυτό το πρόβλημα με πεπερασμένες διαφορές. Το πρόβλημα έπειτα μετασχηματίζεται σε ένα πρόβλημα γραμμικής άλγεβρας της μορφής $\mathbf{Au} = \mathbf{b}$, όπου $\mathbf{A}$ ο πίνακας που αντιστοιχεί στον τελεστή Δ και $\mathbf{b}$ ένα διάνυσμα που καθορίζεται από το ρ. Η επίλυση αυτού του συστήματος είναι, σε γενικές γραμμές, απλή. Το πρόβλημα είναι το μέγεθος του συστήματος. Εάν, παραδείγματος χάριν, εργαζόμαστε με δύο διαστάσεις και εργαζόμαστε με ένα πλέγμα 1000×1000, υπάρχουν 10^6 συνιστώσες $\mathbf{u}$ και ο πίνακας $\mathbf{A}$ έχει 10^{12}.

Γ.5.3.1 Μέθοδοι χαλάρωσης

Μία ευρέως χρησιμοποιούμενη στρατηγική για το πρόβλημα συνοριακών τιμών είναι να θεωρήσουμε ότι το σύστημα εξελίσσεται σε έναν φανταστικό χρόνο τ με δυναμική διάχυσης. Έπειτα θεωρούμε:

$$\begin{cases} \dfrac{\partial u}{\partial \tau} = \Delta u + \rho, \ \ \mathbf{x} \in \boldsymbol{\Omega}, \ \ \tau > 0 \\ \quad u = f, \ \ \mathbf{x} \in \partial\boldsymbol{\Omega}, \ \ \tau > 0 \\ \quad \alpha\rho\chi\iota\kappa\acute{\varepsilon}\varsigma \ \sigma\upsilon\nu\theta\acute{\eta}\kappa\varepsilon\varsigma \ \tau = 0 \end{cases}$$

Γενικά, το u θα πλησιάσει μία στάσιμη κατάσταση καθώς $\tau \to \infty$ που σημαίνει ότι $\partial u/\partial \tau \to 0$. Αυτή η στάσιμη κατάσταση είναι η ζητούμενη λύση. Υποθέστε ότι η u εξελίσσεται στο χρόνο τ με χρήση μεθόδου πεπερασμένων διαφορών της μορφής:

$$\mathbf{u}^{n+1} = \mathbf{T}\mathbf{u}^{n} + \sigma\tau\alpha\theta\varepsilon\rho\acute{o} \ \delta\iota\acute{\alpha}\nu\upsilon\sigma\mu\alpha$$

Ο πίνακας $\mathbf{T}$ ονομάζεται πίνακας επαναλήψεων. Ο ρυθμός σύγκλισης κυβερνάται από τη φασματική ακτίνα ρ_T αυτού του πίνακα, που ορίζεται ως:

$$\rho_T = \max|\lambda|, \ \ \lambda \ \iota\delta\iota\sigma\tau\iota\mu\acute{\varepsilon}\varsigma \ \tau\sigma\upsilon \ \mathbf{T}$$

Για να το δούμε αυτό σημειώνουμε ότι η επιθυμητή λύση αντιστοιχεί σε σημείο ισορροπίας της αναδρομικής σχέσης. Σημειώνουμε με $\mathbf{u}^*$ το σημείο ισορροπίας, απόκλιση από το σημείο ισορροπίας ικανοποιεί τη σχέση:

$$\mathbf{u}^{n+1} - \mathbf{u}^* = \mathbf{T}(\mathbf{u}^n - \mathbf{u}^*) \Rightarrow |\mathbf{u}^n - \mathbf{u}^*| \sim \rho_T^n, \ \ n \to \infty$$

Αυτό καταρχήν δείχνει ότι η μέθοδος θα συγκλίνει εάν $\rho_T < 1$. Βλέπουμε επίσης ότι για να είναι γρήγορη η σύγκλιση η ρ_T πρέπει να είναι όσο το δυνατόν μικρότερη.

Γ.5.3.2 Η μέθοδος Jacobi

Μια απλή μέθοδος χαλάρωσης είναι η μέθοδος Jacobi, η οποία λαμβάνεται με χρήση βημάτων Euler στο φανταστικό χρόνο τ. Η μέθοδος δίνεται από:

$$\boldsymbol{u}^{n+1} = \mathbf{T}\boldsymbol{u}^{n} + \boldsymbol{\rho}$$

όπου ο πίνακας επαναλήψεων $\mathbf{T} = (\mathbf{1} + h\mathbf{A})$ και το διάνυσμα $\boldsymbol{\rho}$ αντιπροσωπεύει την πυκνότητα φορτίων (για απλότητα, υποθέτουμε $f = 0$). Όπως συνήθως ο πίνακας $\mathbf{A}$ είναι η προσέγγιση του τελεστή Δ. Υποθέστε ότι είμαστε σε δύο διαστάσεις και ότι η περιοχή $\boldsymbol{\Omega}$ είναι ένα τετράγωνο,

$$\boldsymbol{\Omega} = \{(x, y) | 0 \leq x, y \leq Ja\}$$

Έπειτα είναι εύκολο να βρεθούν οι ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα $\mathbf{A}$, τα οποία δίνονται από:

$$\mathbf{A}\mathbf{v}^{(k_x,k_y)} = \lambda_{k_x,k_y}\mathbf{v}^{(k_x,k_y)}, \;\; k_x, k_y = 1, \dots, J-1$$

$$\lambda_{k_x,k_y} = -\frac{4}{a^2}\left(sin^2\frac{k_x\pi}{2J} + sin^2\frac{k_y\pi}{2J}\right)$$

$$\mathbf{v}_{jl}^{(k_x,k_y)} = sin\frac{jk_x\pi}{J}sin\frac{lk_x\pi}{J}$$

Οι ιδιοτιμές του πίνακα επανάληψης $\mathbf{T}$ δίνονται από

$$\lambda_{k_x,k_y}^{T} = 1 + h\lambda_{k_x,k_y}$$

από όπου είναι σαφές ότι $\lambda_{k_x,k_y}^{T} \leq 1$εφόσον $\lambda_{k_x,k_y} \leq 0$. Η μικρότερη ιδιοτιμή του $\mathbf{T}$ είναι:

$$\lambda_{J-1,J-1}^{T} = 1 + h\lambda_{J-1,J-1} \approx 1 - 2\frac{4h}{a^2}$$

Αυτό δείχνει ότι προκειμένου να υπάρξει $\rho_T < 1$, πρέπει να πάρουμε

το $h \leq a^2/4$. Στη μέθοδο Jacobi επιλέγουμε $h = a^2/4$. Με αυτήν την επιλογή, η χρονική εξέλιξη δίνεται από:

$$u_{jl}^{n+1} = \frac{1}{4}\left(u_{j+1,l}^{n} + u_{j-1,l}^{n} + u_{j,l+1}^{n} + u_{j,l-1}^{n}\right) + \frac{a^2}{4}\rho_{jl}$$

Για να βρούμε το ρ_T και με αυτόν τον τρόπο το ρυθμό σύγκλισης, σημειώνουμε ότι:

$$max\lambda_{k_x,k_y}^{T} = 1 + h\lambda_{1,1} = 1 - 2sin^2\frac{\pi}{2J} \approx 1 - \frac{\pi^2}{2J^2}$$

$$min\lambda_{k_x,k_y}^{T} = 1 + h\lambda_{J-1,J-1} = 1 - 2sin^2\frac{(J-1)\pi}{2J}$$

$$= cos\frac{(J-1)\pi}{J} \approx -1 + \frac{\pi^2}{2J^2}$$

Έτσι, $\rho_T \approx 1 - \frac{\pi^2}{2J^2}$. Ο αριθμός των επαναλήψεων, r, που απαιτείται για να μειώθει το σφάλμα κατά έναν παράγοντας 10^{-p} ικανοποιεί τη σχέση:

$$\rho_T^r \sim 10^{-p} \Rightarrow r \approx \frac{-p \cdot ln10}{ln\rho_T} \approx \frac{2 \cdot p \cdot ln10}{\pi^2}J^2$$

Ως εκ τούτου, η τετραγωνική εξάρτηση απο το J κάνει τη μέθοδο πολύ αργή ώστε να είναι πρακτικού ενδιαφέροντος.

Γ.5.3.3 Η μέθοδος Gauss-Seidel

Αυτή η μέθοδος είναι παρόμοια με τη μέθοδο Jacobi. Η μόνη διαφορά είναι ότι νέες τιμές της u χρησιμοποιούνται μόλις διατίθενται. Αυτό δίνει ένα ελαφρώς καλύτερο ρυθμό σύγκλισης, αλλά ο απαραίτητος αριθμός επαναλήψεων παρουσιάζει πάλι τετραγωνική εξάρτηση. Η μέθοδος Gauss-Seidel μπορεί να γραφτεί:

$$u_{jl}^{n+1} = \frac{1}{4}\left(u_{j+1,l}^{n} + \boxed{u_{j-1,l}^{n+1}} + u_{j,l+1}^{n} + \boxed{u_{j,l-1}^{n+1}}\right) + \frac{a^2}{4}\rho_{jl}$$

Οι μέθοδοι των Jacobi και Gauss-Seidel είναι αργές αλλά σημαντικές αφετηρίες για πιό προηγμένες μέθοδοι (όπως η SOR, βλ. NR).

Γ.5.3.4 Η μέθοδος μετασχηματισμών Fourier

Θεωρούμε το πρόβλημα συνοριακών τιμών:

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = -\rho \\ u(0,y) = u(X,y), u(x,0) = (x,Y) \quad \textit{περιοδικές συνοριακές συνθήκες} \\ u(0,0) = u_0 \end{cases}$$

Μια πολύ αποδοτική μέθοδος για αυτό το πρόβλημα μπορεί να ληφθεί με χρήση ανάλυσης Fourier:

$$u(x,y) = \sum_{k_x=-\infty}^{\infty} \sum_{k_y=-\infty}^{\infty} \hat{u}(k_x,k_y) e^{2\pi i\left(\frac{k_x x}{X} + \frac{k_y y}{Y}\right)}$$

$$\hat{u}(k_x,k_y) = \frac{1}{XY} \int_0^X dx \int_0^Y dy u(x,y) e^{2\pi i\left(\frac{k_x x}{X} + \frac{k_y y}{Y}\right)}$$

Κάνουμε το ίδιο για το $\rho(x,y)$. Η εισαγωγή στην εξίσωση Poisson δίνει:

$$-4\pi^2 \left[\left(\frac{k_x}{X}\right)^2 + \left(\frac{k_y}{Y}\right)^2\right] \hat{u}(k_x,k_y) = -\hat{\rho}(k_x,k_y)$$

η όποια είναι μια απλή αλγεβρική εξίσωση για τη συνάρτηση $\hat{u}$, αντί της αρχικής διαφορικής εξίσωσης. Αυτό μας δίνει έναν αλγόριθμο τριών βημάτων για την επίλυση του δεδομένου προβλήματος,

- Μετασχηματισμός $\rho \to \hat{\rho}$.
- Υπολογίσμός της $\hat{u}(k_x,k_y)$ με χρήση της ανωτέρω αλγεβρικής εξίσωσης για όλα τα ζεύγη $(k_x,k_y) \neq (0,0)$. Για $k_x = k_y = 0$ η $\hat{u}$ καθορίζεται από την πρόσθετη συνθήκη $u(0,0) = u_0$.

- Μετασχηματισμός $\hat{u} \to u$.

Αριθμητικά, αυτοί οι υπολογισμοί γίνονται χρησιμοποιώντας διακριτοποιημένους μετασχηματισμούς Fourier,

$$f_n = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} \hat{f}_k e^{\frac{2\pi i k n}{N}}, \quad \hat{f}_k = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} f_n e^{-\frac{2\pi i k n}{N}}$$

Με τη μέθοδο του γρήγορου μετασχηματισμού Fourier (FFT), το κόστος μιας μετατροπής $f_n \mapsto \hat{f}_k$ (ή αντίστροφα) γίνεται $\sim N \cdot lnN$ αντί N^2. Αυτό καθιστά τη μέθοδο πολύ γρήγορη. Εντούτοις, απαιτεί ότι η εξεταζόμενη περιοχή να είναι ορθογώνια, που είναι ισχυρός περιορισμός (αν και οι συνοριακές συνθήκες δεν είναι απαραίτητα περιοδικές).

Γ.5.4 Διακριτοποίηση κύματων

Η μονοδιάστατη κυματική εξίσωση είναι:

$$\frac{\partial^2 u}{\partial t^2} = \upsilon \frac{\partial^2 u}{\partial x^2}$$

Αντί αυτής της εξίσωσης θα μελετήσουμε την απλουστευμένη έκδοση:

$$\frac{\partial u}{\partial t} + \upsilon \frac{\partial u}{\partial x} = 0$$

που έχει λύσεις κινούμενες στα δεξιά μόνο. Ας δούμε πως θα λύσουμε αυτή την εξίσωση με πεπερασμένες διαφορές. Όπως συνήθως $u_j^n = u(x_j, t_n)$, όπου $x_j = ja$ και $t_n = nh$. Για τη χωρική παράγωγο χρησιμοποιούμε την προσέγγιση κεντρικής διαφοράς:

$$\left.\frac{\partial u}{\partial x}\right|_{t=t_n} = \frac{u_{j+1}^n - u_{j-1}^n}{2a} + \mathcal{O}(a^2)$$

Η μέθοδος Euler

Με χρήση της μεθόδου Euler, έχουμε:

$$\frac{u_j^{n+1} - u_j^n}{h} = -\upsilon \frac{u_{j+1}^n - u_{j-1}^n}{2a} \Rightarrow u_j^{n+1} = u_j^n - \frac{h\upsilon}{2a}(u_{j+1}^n - u_{j-1}^n)$$

Για να ελέγξουμε την ευστάθεια της μεθόδου, κάνουμε μία Neumann αντικατάσταση $u_j^n = \xi(k)^n e^{ikx_j}$. Αυτό δίνει:

$$\xi(k) = 1 - \frac{h\upsilon}{2a}\left(e^{ika} - e^{-ika}\right) = 1 - i\frac{h\upsilon}{a} sinka \Rightarrow$$
$$|\xi(k)|^2 = 1 + \left(\frac{h\upsilon}{a}\right)^2 sin^2 ka > 1$$

δείχνοντας ότι $\left|u_j^n\right| = |\xi(k)|^n$ αυξάνει εκθετικά με το $n, \forall h > 0$. Έτσι, αυτή καταστροφή - είναι πάντα ασταθής.

Η μέθοδος της «μακριάς γαϊδούρας»

Η μέθοδος της «μακριάς γαϊδούρας» είναι δεύτερης τάξης ως προς το χρόνο και το διστημα και προκύπτει με προσέγγιση κεντρικής διαφοράς για τη χρονική παράγωγο,

$$u_j^{n+1} = u_j^{n-1} - \frac{\upsilon h}{a}(u_{j+1}^n - u_{j-1}^n)$$

Για αυτήν τη μέθοδο, η αντικατάσταση $u_j^n = \xi(k)^n e^{ikx_j}$ δίνει:

$$\xi(k) = -i\frac{\upsilon h}{a} sinka \pm \sqrt{1 - \left(\frac{\upsilon h}{a}\right)^2 sin^2 ka}$$

Μπορεί εύκολα να ελεγχθεί ότι αυτή η μέθοδος, είναι ευσταθής για $\frac{\upsilon h}{a} \leq 1$. Η μέθοδος χωρίζει το πλέγμα σε δύο πλέγματα που δεν επηρεάζουν το ένα το άλλο. Αυτό μπορεί να προκαλέσει αριθμητικά προβλήματα.

Γ.5.5 Η εξίσωση Schrödinger

Θεωρούμε τη μονοδιάστατη εξίσωση Schrödinger:

$$i\frac{\partial\psi}{\partial t} = -\frac{\partial^2\psi}{\partial x^2} + V(x)\psi = H\psi$$

(σε μονάδες τέτοιες ώστε $\hbar = 1, m = 1/2$). Για $V = 0$, η εξίσωση περιγράφει διάχυση σε φανταστικός χρόνο, για $\tau = it, V = 0$ λαμβάνουμε $\frac{\partial\psi}{\partial\tau} = -\frac{\partial^2\psi}{\partial x^2}$. Μετά από διακριτοποίηση του x η εξίσωση του Schrödinger γίνεται:

$$i\frac{d\boldsymbol{\psi}}{dt} = \mathbf{H}_D\boldsymbol{\psi}$$

όπου το $\boldsymbol{\psi}$ είναι ένα διάνυσμα με συνιστώσες $\psi_j(t) = \psi(x_j, t)$ και $\mathbf{H}_D$ είναι η διακριτοποιημένη έκδοση του H που υποθέτουμε ότι είναι Ερμιτιανός, $\mathbf{H}_D^\dagger = \mathbf{H}_D$. Μία θεμελιώδης ιδιότητα της ακριβούς λύσης είναι ότι η κανονικοποίηση διατηρείται,

$$\int_{-\infty}^{+\infty} |\psi(x,t)|^2 dx = 1$$

Πρέπει να ελεγχθεί αν η διακριτοποίηση διατηρεί την κανονικοποίηση για τις τρεις μεθόδους που σχολιάστηκαν και παραπάνω.

$$\frac{d}{dt}\left(\boldsymbol{\psi}^\dagger\boldsymbol{\psi}\right) = \left(\tfrac{1}{i}\mathbf{H}_D\boldsymbol{\psi}\right)^\dagger\boldsymbol{\psi} + \tfrac{1}{i}\boldsymbol{\psi}^\dagger\mathbf{H}_D\boldsymbol{\psi} = 0$$

Οι ιδιοτιμές E_I του $\mathbf{H}_D$ είναι πραγματικές, δεδομένου ότι ο $\mathbf{H}_D$ είναι Ερμιτιανός. Συμβολίζοντας τα ιδιοδιανύσματα με $\boldsymbol{\psi}_I$,

$$\mathbf{H}_D\boldsymbol{\psi}_I = E_I\boldsymbol{\psi}_I$$

Εξετάζουμε τώρα τρείς τρόπους για τη διακριτοποίσηση του χρόνου,

Η αναλυτική μέθοδος Euler

$\boldsymbol{\psi}^{n+1} = \boldsymbol{\psi}^n - ih\mathbf{H}_D\boldsymbol{\psi}^n = \mathbf{T}\boldsymbol{\psi}^n$, όπου ο πίνακας επαναλήψεων $\mathbf{T} = \mathbf{1} - ih\mathbf{H}_D$. Οι ιδιοτιμές του πίνακα $\mathbf{T}$ είναι $\lambda_I = 1 - ihE_I$ και ικανοποιούν την ανισότητα $|\lambda_I| \geq 1,\ \forall I$. Υποθέτουμε τώρα ότι $\boldsymbol{\psi}^0 = \sum_I c_I\boldsymbol{\psi}_I \Rightarrow \boldsymbol{\psi}^n = \sum_I c_I\lambda_I^n\boldsymbol{\psi}_I$ και από το γεγονός ότι τα ιδιοδιανύσματα είναι ορθογώνια ακολουθεί $|\boldsymbol{\psi}^n|^2 = \sum_I |c_I|^2\lambda_I^{2n}$, που σημαίνει ότι η $|\boldsymbol{\psi}^n|^2$ αυξάνει εκθετικά με το n (για μεγάλα n).

Η πεπλεγμένη μέθοδος Euler

$\boldsymbol{\psi}^{n+1} = \boldsymbol{\psi}^n - ih\mathbf{H}_D\boldsymbol{\psi}^{n+1} \Rightarrow \boldsymbol{\psi}^{n+1} = \mathbf{T}\boldsymbol{\psi}^n$, όπου ο πίνακας επαναλήψεων $\mathbf{T} = (\mathbf{1} + ih\mathbf{H}_D)^{-\mathbf{1}}$**.** Οι ιδιοτιμές του πίνακα $\mathbf{T}$ είναι $\lambda_I = 1/(1 + ihE_I)$ και ικανοποιούν την ανισότητα $|\lambda_I| \leq 1,\ \forall I$. Συνεπώς η $|\boldsymbol{\psi}^n|^2$ αυξάνει σταθερά με το n.

Η μέθοδος Crank-Nicholson (ή τραπεζοειδής κανόνας).

$\boldsymbol{\psi}^{n+1} = \boldsymbol{\psi}^n + \frac{h}{2}(-i\mathbf{H}_D\boldsymbol{\psi}^{n+1} - i\mathbf{H}_D\boldsymbol{\psi}^n) \Rightarrow \boldsymbol{\psi}^{n+1} = \mathbf{T}\boldsymbol{\psi}^n$ όπου ο πίνακας επαναλήψεων δίνεται από:

$$\mathbf{T} = \left(\mathbf{1} + i\tfrac{h}{2}\mathbf{H}_D\right)^{-\mathbf{1}}\left(\mathbf{1} - i\tfrac{h}{2}\mathbf{H}_D\right)$$

και ο ιδιοτιμές του:

$$\lambda_I = \frac{1 - i\frac{h}{2}E_I}{1 + i\frac{h}{2}E_I}$$

Εφόσον οι λ_I δίνονται σαν ο λόγος δύο συζυγών μιγαδικών αριθμών έχουμε $|\lambda_I| = 1$, επομένως η $|\boldsymbol{\psi}^n|^2$ είναι σταθερή. Έτσι, η μέθοδος Crank-Nicolson έχει το πλεονέκτημα ότι διατηρεί την κανονικοποίηση.

Γ.6 ΤΥΧΑΙΟΙ ΑΡΙΘΜΟΙ ΚΑΙ MONTE CARLO

Οι υπολογισμοί Monte Carlo είναι ένας ευρέως χρησιμοποιούμενος όρος που μπορεί να σημάνει διαφορετικά πράγματα. Το κοινό αυτών των υπολογισμών είναι ότι εμπλέκονται τυχαίοι αριθμοί.

- Οι υπολογισμοί Monte Carlo μπορεί να σημάνουν την προσομοίωση μιας διαδικασίας που είναι πράγματικά στοχαστική (τυχαία) στη φύση.
- Αλλά μπορεί επίσης να είναι ένας υπολογισμός ενός ολοκληρώματος ή ενός αθροίσματος, στα οποία οι τυχαίοι αριθμοί χρησιμοποιούνται ως υπολογιστικό εργαλείο.

Η δομή αυτού του τμήματος είναι η ακόλουθη. Αρχίζουμε με τα βασικά στοιχεία τυχαίων μεταβλητών και θεωρίας πιθανοτήτων. Συζητάμε έπειτα τις μεθόδους παραγωγής τυχαίων αριθμών με χρήση υπολογιστή. Τέλος, συζητάμε την Monte Carlo ολοκλήρωση και άθροιση.

Γ.6.1 Τυχαίες μεταβλητές

Γ.6.1.1 Χρήσιμοι ορισμοί

Μία τυχαία μεταβλητή X καθορίζεται από την κατανομή πιθανότητάς της (ή συνάρτηση συχνότητας) $p(x) \geq 0$ που έχει τις ακόλουθες ιδιότητες:

$$p(x)dx = P\{x < X < x + dx\} = \pi\iota\theta\alpha\nu\acute{o}\tau\eta\tau\alpha\ \nu\alpha\ \beta\rho o\acute{\upsilon}\mu\varepsilon\ x < X < x + dx$$

$$\int_a^b p(x)dx = P\{a < X < b\}$$

$$\int_{-\infty}^{+\infty} p(x)dx = 1,\ \ \kappa\alpha\nu o\nu\iota\kappa o\pi o\acute{\iota}\eta\sigma\eta$$

Ο μέσος όρος $\langle f(X)\rangle$ μίας τυχαίας συνάρτησης $f(X)$ ορίζεται από:

$$\langle f(X)\rangle = \int_{-\infty}^{+\infty} f(x)p(x)dx$$

Μερικές ποσότητες για το χαρακτηρισμό της κατανομής μίας τυχαίας μεταβλητής X είναι:

- **Η μέση τιμή.**

$$\langle X\rangle = \int_{-\infty}^{+\infty} xp(x)dx$$

η όποα δεν είναι γενικά ίδια με την πιθανότερη τιμή.

- **Η διασπορά-διακύμανση.**

$$\sigma_X^2 = \langle (X - \langle X\rangle)^2\rangle = \langle X^2\rangle - 2\langle X\rangle^2 + \langle X\rangle^2 = \langle X^2\rangle - \langle X\rangle^2$$

η όποιο αποτελεί μέτρο των διακυμάνσεων της X. Η τετραγωνική ρίζα της διακύμανσης, σ_X, ονομάζεται τυπική απόκλιση.

- **Ροπές υψηλότερης τάξης.** $\langle X^n\rangle,\ \ n = 1,2,3,\ldots$

Η κανονική κατανομή με μέση τιμή $\langle X\rangle = \mu$ και διακύμανση $\sigma_X^2 = \sigma^2$ δίνεται από:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Η ταυτόχρονη κατανομή $p(x,y)$ δύο τυχαίων μεταβλητών X και Y ονομάζεται αρθρωτή κατανομή και ορίζεται από:

$$p(x,y)dxdy = P\{x < X < x + dx \ \ και \ \ y < Y < y + dy\}$$

Μερικές χρήσιμες μονοδιάστατες κατανομές που μπορούν να ληφθούν από την αρθρωτή κατανομή είναι:

- **Οι περιθωριακές κατανομές.**

$$p(x) = \int_{-\infty}^{+\infty} p(x,y)dy\,,\;\; p(y) = \int_{-\infty}^{+\infty} p(x,y)dx$$

- **Οι υπο συνθήκη πιθανότητες $p(x|y)$ και $p(y|x)$**. $p(x|y)$ είναι η πιθανότητα του x δεδομένου y. Για κάθε y, $p(x|y)$ είναι μία κανονικοποιημένη στο x κατανομή. Η συνάρτηση $x \mapsto p(x,y)$ είναι ανάλογη προς την $p(x|y)$ αλλά δεν είναι κανονικοποιημένη. Η ακριβής σχέση μεταξύ αυτών των δύο συναρτήσεων δίνεται από τον κανόνα Bayes, ο οποίος δηλώνει ότι:

$$p(x,y) = p(x|y)p(y) = p(y|x)p(x)$$

Δύο τυχαίες μεταβλητές X, Y είναι ανεξάρτητες εάν:

$$p(x,y) = p(x)p(y)$$

Σε αυτή την περίπτωση $p(x|y) = \frac{p(x,y)}{p(y)} = p(x), \forall y$. Δύο τυχαίες μεταβλητές X, Y είναι ασύνδετες εάν:

$$\langle XY \rangle = \langle X \rangle \langle Y \rangle$$

Η ανεξαρτησία είναι ισχυρότερη από την ασυνδετότητα, εάν οι X, Y είναι ανεξάρτητες, αμέσως ακολουθεί ότι είναι και ασύνδετες. Το αντίστροφο δεν ισχύει, όπως παρουσιάζει το ακόλουθο παράδειγμα.

Παράδειγμα (ασύνδετων αλλά εξαρτώμενων τυχαίων μεταβλητών).

Ας είναι X, Y ανεξάρτητοι δυαδικοί τυχαίοι αριθμοί με πιθανές τιμές 0 και 1. Υποθέτουμε ότι $p_X(0) = p_Y(0) = p_X(1) = p_Y(1) = 1/2$ καθώς επίσης θέτουμε $U = X + Y$ και $V = |X - Y|$. Υπάρχουν τέσσερεις ισοπίθανες καταστάσεις του συστήματος:

X	Y	*Πιθανότητα*	U	V
0	0	1/4	0	0
0	1	1/4	1	1
1	0	1/4	1	1
1	1	1/4	2	0

Οι περιθωριακές κατανομές των U, V είναι:

$$\begin{pmatrix} p_U(0) \\ p_U(1) \\ p_U(2) \end{pmatrix} = \begin{pmatrix} 1/4 \\ 1/2 \\ 1/4 \end{pmatrix}, \quad \begin{pmatrix} p_V(0) \\ p_V(1) \end{pmatrix} = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}$$

τα U, V είναι εξαρτώμενα επειδή:

$$p(U = 0, V = 1) = 0 \neq p_U(0)p_V(1)$$

Τέλος τα U, V είναι ασύνδετα επειδή:

$$\left.\begin{aligned} \langle UV \rangle &= \tfrac{1}{4}0 + \tfrac{1}{4}1 + \tfrac{1}{4}1 + \tfrac{1}{4}0 = \tfrac{1}{2} \\ \langle U \rangle &= \tfrac{1}{4}0 + \tfrac{1}{2}1 + \tfrac{1}{4}2 = 1 \\ \langle V \rangle &= \tfrac{1}{2}0 + \tfrac{1}{2}1 = \tfrac{1}{2} \end{aligned}\right\} \Rightarrow \langle UV \rangle = \langle U \rangle \langle V \rangle$$

Γ.6.1.2 Αθροίσματα ανεξάρτητων και όμοια κατανεμημένων τυχαίων μεταβλητών

Υποθέτουμε ότι $X_1, \dots, X_N$ είναι ανεξάρτητες και όμοια κατανεμημένες τυχαίες μεταβλητές με μέσες τιμές $\langle X \rangle = \mu$ και διακύμανση σ^2. Ποια είναι έπειτα η κατανομή αθροίσματος τους; Αυτή η ερώτηση έχει μια εκπληκτικά απλή απάντηση στο όριο $N \to \infty$. Αυτό το αποτέλεσμα, θεώρημα κεντρικού ορίου, λέει ότι η τυχαία μεταβλητή:

$$\tilde{S}_N = \frac{\sum_{i=1}^{N} X_i - \mu N}{\sigma\sqrt{N}}$$

είναι κανονικά κατανεμημένη με $\langle \tilde{S}_N \rangle = 0$ και μοναδιαία διακύμανση στο όριο $N \to \infty$.

Απόδειξη

Ας σκιαγραφήσουμε πώς θα γίνει η αποδείξει. Για να γίνει αυτό είναι απαραίτητο να εισαγάγουμε την αποκαλούμενη χαρακτηριστική συνάρτηση $\Phi(k)$ για την τυχαία μεταβλητή:

$$S_N = \frac{1}{N}\sum_{i=1}^{N} X_i - \mu$$

ορισμένη από:

$$\Phi(k) = \langle e^{ikS_N} \rangle$$

$\Phi(k)$ είναι ο μετασχηματισμός Fourier της κατανομής πιθανότητας $p_N(s)$ της S_N, το οποίο σημαίνει ότι η $p_N(s)$ μπορεί να ληφθεί ως ο αντίστροφος μετασχηματισμός Fourier,

$$\Phi(k) = \int_{-\infty}^{\infty} e^{iks} p_N(s) ds \Rightarrow p_N(s) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-iks}\Phi(k) dk$$

Για μεγάλα N, το $ln\Phi(k)$ μπορεί να υπολογιστεί με τον ακόλουθο τρόπο:

$$ln\Phi(k) = ln\langle e^{\frac{ik}{N}(X_1-\mu)} \cdots e^{\frac{ik}{N}(X_N-\mu)} \rangle = Nln\langle e^{\frac{ik}{N}(X-\mu)} \rangle =$$
$$Nln\langle 1 + \frac{ik}{N}(X-\mu) + \frac{1}{2}\left(\frac{ik}{N}\right)^2 (X-\mu)^2 + \mathcal{O}(N^{-3}) \rangle =$$
$$Nln\left[1 - \frac{k^2}{2N^2}\sigma^2 + \mathcal{O}(N^{-3})\right] \sim -\frac{k^2\sigma^2}{2N}, \;\; N \to \infty$$

Έτσι, $\Phi(k) \sim e^{-\frac{k^2\sigma^2}{2N}}$ καθώς $N \to \infty$, έτσι προκύπτει:

$$p_N(s) \sim \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-iks} e^{-\frac{k^2\sigma^2}{2N}} dk = \left\{u = k\frac{\sigma}{\sqrt{2N}}\right\} =$$

$$\frac{1}{2\pi}\frac{\sqrt{2N}}{\sigma}\int_{-\infty}^{\infty} e^{-(u^2+iu\sqrt{2N}s/\sigma)}du$$

$$= \frac{1}{2\pi}\frac{\sqrt{2N}}{\sigma}\underbrace{\int_{-\infty}^{\infty} e^{-(u+i\frac{s}{\sigma}\sqrt{\frac{N}{2}})^2}du}_{\sqrt{\pi}} e^{-\frac{s^2N}{2\sigma^2}} \Rightarrow$$

$$p_N(s) \sim \frac{1}{\sqrt{2\pi\sigma^2/N}} e^{-\frac{s^2}{2\sigma^2/N}},\ \ N \to \infty$$

Αυτό σημαίνει ότι η μεταβλητή $\tilde{S}_N$ πράγματι κατανέμεται κανονικά με με μέση τιμή $\langle \tilde{S}_N \rangle = 0$ και διακύμανση σ^2. Είναι πολύ σημαντικό να σημειώσουμε ότι σε αυτήν την παραγωγή δεν εξετάσαμε ποτέ την ακριβή μορφή της κατανομής των X_i. Αυτό κάνει το κεντρικό θεώρημα κεντρικού ορίου εξαιρετικά χρήσιμο.

Γ.6.1.3 Ροπές και σωρευτές

Ας είναι $\Phi(k) = \langle e^{ikX} \rangle$ η χαρακτηριστική συνάρτηση χαρακτηριστική λειτουργία για μία τυχαία μεταβλητή X. Το ανάπτυγμα Taylor του εκθετικού είναι:

$$\Phi(k) = \sum_{n=0}^{\infty} \frac{(ik)^n}{n!} \langle X^n \rangle$$

όπου $\langle X^n \rangle$ είναι η $n-$οστή ροπή του X. Στενά συνδεδεμένοι με τις ροπές είναι οι αποκαλούμενα σωρευτές, οι οποίοι σημειώνωνται με τα οποία με $\langle X^n \rangle_c$ και δίνονται από το ανάπτυγμα Taylor του φυσικού λογαρίθμου της χαρακτηριστικής συνάρτησης,

$$ln\Phi(k) = \sum_{n=0}^{\infty} \frac{(ik)^n}{n!} \langle X^n \rangle_c$$

Με ανάπτυξη του αριστερού μέλους αυτής της εξίσωσης σε όρους ροπών, είναι δυνατό να βρεθεί η σύνδεση μεταξύ σωρευτών και ροπών. Προκύπτει ότι ο $n-$οστός σωρευτής $\langle X^n \rangle_c$ μπορεί να εκφραστεί με χρήση των n πρώτων ροπών. Για τους δύο πρώτους σωρευτές έχουμε:

$$\langle X \rangle_c = \langle X \rangle,$$

$$\langle X^2 \rangle_c = \langle X^2 \rangle - \langle X \rangle^2$$

Μπορεί να φανεί περιττή η εισαγωγή αυτών των ποσοτήτων που μπορούν να εκφραστούν σαν συναρτήσεις των ροπών. Εντούτοις, οι σωρευτές έχουν δύο σημαντικές ιδιότητες:

- Οι σωρευτές παρέχουν ένα απλό μέτρο για το πόσο κοντά στην κανονική κατανομή είναι μία δεδομένη κατανομή, επειδή όλοι οι σωρευτές τρίτης και υψηλότερης τάξης είναι μηδενικοί για την κανονική κατανομή.
- Οι σωρευτές ανεξάρτητων μεταβλητών είναι πρόσθετικές, εάν οι X, Y είναι ανεξάρτητες και $Z = X + Y$, έπειτα $\langle Z^n \rangle_c = \langle X^n \rangle_c + \langle Y^n \rangle_c$ για όλα τα n.

Γ.6.2 Μετασχηματίζοντας τυχαίους αριθμούς

Στον υπολογιστή, έχουμε συνήθως πρόσβαση σε κάποια γεννήτρια τυχαίων αριθμών, αυτή παράγει τους (ψευδο-) τυχαίους αριθμούς R που κατανέμονται ομοιόμορφα μεταξύ 0 και 1,

$$p(r) = \begin{cases} 1, 0 < r < 1 \\ 0, \text{εδάλλως} \end{cases}$$

Σε αυτήν την υποενότητα, εξετάζουμε πώς να λάβουμε τυχαίους αριθμούς με άλλες κατανομές, υποθέτοντας ότι έχουμε στη διάθεση μας τους παραπάνω ομοιόμορφα τυχαίους αριθμούς.

Ας είναι X μία αυθαίρετη τυχαία μεταβλητή με κατανομή $p_X(x)$ και υποθέστε ότι $Y = f(X)$, όπου η f είναι μια μονότονη συνάρτηση. Ποια είναι έπειτα η κατανομή $p_Y(y)$ του Y; Για να απαντήσουμε ορίζουμε τις αθροιστικές κατανομές $P_X(x)$ και $P_Y(Y)$ των X, Y, αντίστοιχα:

$$P_X(x) = P\{X \leq x\} = \int_{-\infty}^{x} p_X(t)\, dt$$
$$P_Y(y) = P\{Y \leq y\} = \int_{-\infty}^{y} p_Y(t)\, dt$$

Αν η $y = f(x)$ είναι αύξουσα συνάρτηση του x,

$$P_X(x) = P_Y(y) \Rightarrow p_X(x) = \frac{dP_X}{dx} = \frac{dP_Y}{dy}\frac{dy}{dx} = p_Y(y)\frac{dy}{dx}$$

ενώ αν η $y = f(x)$ είναι φθίνουσα συνάρτηση του x,

$$P_X(x) = 1 - P_Y(y) \Rightarrow p_X(x) = -p_Y(y)\frac{dy}{dx}$$

Αυτό δείχνει ότι η σχέση μεταξύ των $p_X(x)$ και $p_Y(y)$ μπορεί να γραφτεί,

$$p_X(x) = p_Y(y)\left|\frac{dy}{dx}\right|$$

για όλες τις μονότονες συναρτήσεις (και η έκφραση παραμένει ισχύουσα για υψηλότερες διαστάσεις άν θεωρήσουμε ότι η $|dy/dx|$ είναι η Ιακωβιανή).
Υποθέστε τώρα ότι η $X = R$ κατανέμεται ομοιόμορφα μεταξύ 0 και 1. Το ερώτημα που θα απαντήσουμε είναι: Ποια θα πρέπει να είναι η συνάρτηση f του μετασχηματισμού ώστε η $Y = f(R)$ να έχει την καθορισμένη κατανομή $p_Y(y)$; Υποθέτουμε άυξουσα συνάρτηση f, οι αθροιστικές κατανομές πρέπει να ικανοποιούν:

$$P_Y(y) = P_R(r) = \int_{-\infty}^{r} p_R(t)\,dt = r \Rightarrow y = P_Y^{-1}(r),\ \ 0 \le r \le 1$$

που σημαίνει ότι $f = P_Y^{-1}$. Δεδομένου ότι οι $1 - R$ και R έχουν την ίδια κατανομή, μπορούμε ισοδύναμα να πάρουμε $Y = P_Y^{-1}(1 - R)$. Αυτό κάνει την Y φθίνουσα συνάρτηση του R. Αυτή η μέθοδος παραγωγής τυχαίων αριθμών με διαφορετικές κατανομές ονομάζεται μέθοδος μετασχηματισμού.

Παράδειγμα (η εκθετική κατανομή)

Υποθέστε ότι έχουμε πρόσβαση στους τυχαίους αριθμούς R που κατανέμονται ομοιόμορφα μεταξύ 0 και 1 και θέλουμε τυχαίους αριθμούς με την κατανομή:

$$p(y) = \begin{cases} \lambda e^{-\lambda y}, y \geq 0 \\ 0, y < 0 \end{cases}, \lambda > 0$$

Χρησιμοποιούμε τη μέθοδο μετασχηματισμού. Το πρώτο βήμα είναι να υπολογιστεί η αθροιστική κατανομή:

$$P_Y(y) = \int_{-\infty}^{y} p_Y(t)\, dt = \left[-e^{-\lambda t}\right]_0^y = 1 - e^{-\lambda y}$$

Ο μετασχηματισμός $r \to y$ λαμβάνεται έπειτα με επίλυση της:

$$P_Y(y) = P_R(r) = r \Rightarrow 1 - e^{-\lambda y} = r \Rightarrow y = -\frac{1}{\lambda} ln(1 - r)$$

Έτσι, η $Y = -\frac{1}{\lambda} ln(1 - R)$ έχει την επιθυμητή κατανομή.

Παράδειγμα (η μέθοδος Box-Muller)

Η μέθοδος μετασχηματισμού δεν μπορεί να εφαρμοστεί άμεσα στην κατανομή:

$$p(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}$$

επειδή δεν μπορούμε να πάρουμε κλειστή έκφραση για την $P(y)$. Εντούτοις, το πρόβλημα μπορεί να παρακαμφθεί με τη θεώρηση δύο ανεξάρτητων μεταβλητών Y_1, Y_2 με την ίδια κατανομή $p(y)$,

$$p(y_1, y_2) = \frac{1}{2\pi} e^{-\frac{y_1^2 + y_2^2}{2}}$$

Σε πολικές συντεταγμένες (ρ, θ), αυτή η κατανομή γίνεται:

$$p(\rho, \theta) = p(y_1, y_2) \underbrace{\left|\frac{\partial(y_1, y_2)}{\partial(\rho, \theta)}\right|}_{\rho} = \underbrace{\rho e^{-\frac{\rho^2}{2}}}_{p(\rho)} \underbrace{\frac{1}{2\pi}}_{p(\theta)}$$

Το γεγονός ότι αυτή η κατανομή παραγοντοποιείται, $p(\rho, \theta) = p(\rho)p(\theta)$, σημαίνει ότι οι ρ, θ μπορούν να παρχθούν ανεξάρτητα.

- Η κατανομή της θ είναι ομοιόμορφη. $\theta = 2\pi R_1$, όπου R_1 κατανέμεται ομοιόμορφα μεταξύ 0 και 1.
- Εφαρμόζουμε τη μέθοδο μετασχηματισμού για την ρ.

$$P(\rho) = \int_0^{\rho} te^{-\frac{t^2}{2}}dt = 1 - e^{-\frac{\rho^2}{2}} = r \Rightarrow \rho = \sqrt{-2ln(1-r)}$$

Έτσι, η ρ μπορεί να ληφθεί από $\rho = \sqrt{-2ln(1-R_2)}$, όπου R_2 κατανέμεται ομοιόμορφα μεταξύ 0 και 1. Ο μετασχηματισμός πίσω στις καρτεσιανές συντεταγμένες μας δίνει δύο ανεξάρτητους τυχαίους αριθμούς με την επιθυμητή κατανομή:

$$Y_1 = \sqrt{-2ln(1-R_2)}cos2\pi R_1$$
$$Y_2 = \sqrt{-2ln(1-R_2)}sin2\pi R_1$$

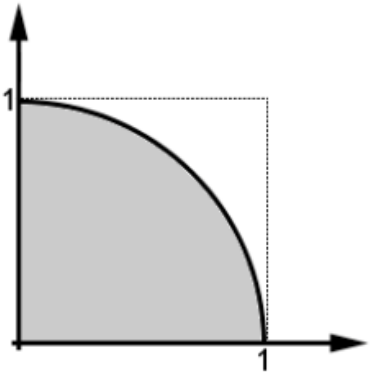
Γ.6.3 Ολοκλήρωση και άθροιση Monte Carlo

Παράδειγμα (υπολογισμός Monte Carlo του π)

Θεωρούμε το πρώτο τεταρτημόριο του μοναδιαίου κύκλου. Το εμβαδό του ($= \pi/4$) μπορεί να γραφτεί

$$I = \int_{πρώτο\ τεταρτημόριο} dxdy$$

Ας δούμε πως μπορούμε να υπολογίσουμε αυτό τ ολοκλήρωμα με χρήση τυχαίων αριθμών.



Υποθέστε ότι έχουμε N τυχαία σημεία από την ομοιόμορφη κατανομή στο μοναδιαίο τετράγωνο. Για κάθε σημείο i, εισάγουμε μία δυαδική τυχαία μεταβλητή χ_i, τέτοιοι ώστε $\chi_i = 1$ εάν το σημείο i είναι στο πρώτο τεταρτημόριο του μοναδιαίου κύκλου, ειδάλλως $\chi_i = 1$. Ο μέσος όρος κάθε μίας χ_i είναι I, το εμβαδό του πρώτου τεταρτημόριου (δεδομένου ότι το εμβαδό του τετραγώνου είναι 1). Ακολουθεί:

$$\frac{1}{N}\sum_{i=1}^{N}\chi_i = \frac{1}{N}\{\alpha\rho\iota\theta\mu\acute{o}\varsigma\ \sigma\eta\mu\epsilon\acute{\iota}\omega\nu\ \sigma\tau o\ \pi\rho\acute{\omega}\tau o\ \tau\epsilon\tau\alpha\rho\tau\eta\mu\acute{o}\rho\iota o\} \xrightarrow{N\to\infty} I$$

Έτσι έχουμε μία μέθοδο για τον υπολογισμό του I και συνεπώς του π. Ας γενικεύσουμε το παραπάνω παράδειγμα θεωρώντας το ολοκλήρωμα:

$$I = \int f(x)p(x)dx$$

όπου $f(x)$ μία αυθαίρετη συνάρτηση και $p(x)$ κάποια κατανομή πιθανότητας. Θεωρούμε ότι $X_1, \dots X_N$ είναι ανεξάρτητες τυχαίες μεταβλητές, όλες με κατανομή $p(x)$. Οι $f(X_1), \dots, f(X_N)$ είναι έπειτα ανεξάρτητες και όμοιομορφα κατανεμημένες τυχαίες μεταβλητές. Αυτό σημαίνει, σύμφωνα με το θεώρημα κεντρικού ορίου ότι η μεταβλητή:

$$I_N = \frac{1}{N}\sum_{i=1}^{N} f(X_i)$$

είναι κατα προσέγγιση κανονικά κατανεμημένη για $N \to \infty$, με:

- μέση τιμή $\langle I_N \rangle = \int f(x)p(x)dx = I$ και
- διακύμανση $\sigma_N^2 = \frac{\langle f(X)^2\rangle - \langle f(X)\rangle^2}{N} \xrightarrow{N\to\infty} 0$.

Ως εκ τούτου, το I_N μπορεί να χρησιμοποιηθεί ως εκτιμηση του I για μεγάλα N. Η μέθοδος μπορεί να γενικευτεί άμεσα σε μεγαλύτερες διαστάσεις. Τα αθροίσματα μπορούν να εξεταστούν με παρόμοιο τρόπο. Θεωρούμε:

$$S = \sum_i f(i)p(i)$$

όπου $f(i)$ μία αυθαίρετη συνάρτηση και $p(i)$ κάποια διακριτή κατανομή πιθανότητας. Αν $I_1, \dots, I_N$ ανεξάρτητοι τυχαίοι αριθμοί με κατανομή $p(i)$ μπορούμε να εκτίμησουμε την S με χρήση της

$$S \approx S_N = \frac{1}{N}\sum_{k=1}^{N} f(I_k)$$

Γ.6.3.1 Ρυθμός σύγκλισης

Τι γίνεται με την απόδοση της ολοκλήρωσης Monte Carlo; Ας συγκρίνουμε την απόδοση της, σε μία διάσταση $D = 1$, με αυτή του κανόνα Simpson. Θεωρούμε ένα ολοκλήρωμα σε ένα μήκους L, και συμβολίζουμε με T_ε το χρόνο υπολογισμού που δίνει ακρίβεια $\mathcal{O}(\varepsilon)$.

- **Κανόνας Simpson.**

$$\begin{cases} \varepsilon \sim h^4 \ (h\ \tau o\ \mu \acute{\varepsilon} \gamma \varepsilon \theta o \varsigma\ \tau o \upsilon\ \beta \acute{\eta} \mu \alpha \tau o \varsigma) \\ T_\varepsilon \propto \alpha \rho \iota \theta \mu o \acute{\upsilon}\ \tau \iota \mu \acute{\omega} \nu\ \sigma \upsilon \nu \alpha \rho \tau \acute{\eta} \sigma \varepsilon \omega \nu \sim L/h \end{cases} \Rightarrow T_\varepsilon \sim \varepsilon^{-1/4}$$

- **Μόντε Κάρλο.**

$$\begin{cases} \varepsilon \sim N^{-1/2} \\ T_\varepsilon \propto N \end{cases} \Rightarrow T_\varepsilon \sim \varepsilon^{-2}$$

Αυτή η σύγκριση δείχνει ότι η σύγκλιση της μεθόδου Monte Carlo είναι πολύ πιό αργή. Η δύναμη της μεθόδου Monte Carlo είναι η γενικότητά της. Εάν, παραδείγματος χάριν, το D είναι μεγάλο ή το σύνορο ολοκλήρωσης σύνθετο, λίγες εναλλακτικές λύσεις υπάρχουν.

Γ.6.3.2 Σημαντικότητα δειγματοληψίας (importance sampling)

Ένας υπολογισμός Monte Carlo ενός ολοκληρώματος:

$$I = \int f(x)dx$$

μπορεί να χρησιμοποιεί τυχαίους αριθμούς που προέρχονται από οποιαδήποτε κατανομή πιθανότητας $p(x) > 0$. Αν οι $X_1, \dots X_N$ προέρχονται από οποιαδήποτε κατανομή πιθανότητας $p(x) > 0$, μπορούμε να υπολογίσουμε το I ως εξής:

$$I = \int \frac{f(x)}{p(x)} p(x) dx \approx I_N^p = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}$$

Ο μέσος όρος της εκτίμησης I_N^p είναι, φυσικά, ανεξάρτητος του $p(x)$, $\langle I_N^p \rangle = I$, ενώ η διασπορά $\langle (I_N^p)^2 \rangle - \langle I_N^p \rangle^2$ εξαρτάται απο την p. Προκειμένου η απόδοση της μεθόδου να είναι λογική, είναι κρίσιμο να γίνει προσεκτική επιλογή της κατανομής πιθανότητας $p(x)$. Σε γενικές γραμμές, είναι γνωστό τι η βέλτιστη επιλογή της $p(x)$ είναι, $p(x) \propto |f(x)|$ (δείτε NR 7.8). Στην πράξη, αυτό βοηθεί λίγο επειδή το να βρεθεί η σταθερά αναλογίας είναι τόσο δύσκολο όσο το να βρεθεί το ολοκλήρωμα που θέλουμε να υπολογίσουμε.

Γ.6.4 Ο αλγόριθμος Metropolis

Θεωρούμε ένα σύστημα με καταστατικό χώρο T, και υποθέτουμε ότι θέλουμε κάποια κατανομή $\tilde{p}(\sigma)$, $\sigma \in T$. Για απλότητα, υποθέτουμε ότι ο καταστατικός χώρος είναι διακριτός. Ο αλγόριθμος Metropolis μπορεί να θεωρηθεί τυχαίο βάδισμα στον καταστατικό χώρο T,

$$\sigma_1 \to \sigma_2 \to \sigma_3 \to \cdots$$

Εδώ, τα σ_n συμβολίζουν τις καταστάσεις του συστήματος στον διακριτό χρόνο n. Το βάδισμα είναι τέτοιο ώστε

$$\lim_{n \to \infty} p_n(\sigma) = \tilde{p}(\sigma)$$

ανεξάρτητα από την p_1. Μια θεμελιώδης ιδιότητα του αλγορίθμου Metropolis είναι ότι η p_{n+1} καθορίζεται εξ' ολοκλήρου από την p_n. Μια στοχαστική διαδικασία με αυτή την ιδιότητα καλείται αλυσίδα Markov. Για μία αλυσίδα Markov η χρονική εξέλιξη μπορεί να περιγραφεί σε όρους του πίνακα μετάβασης $W(\sigma, \sigma')$, που είναι η υπό

συνθήκη πιθανότητα της εύρεσης του συστήματος στην κατάσταση σ τη χρονική στιγμή $n+1$, δεδομένου ότι ήταν στην κατάσταση σ' τη χρονική στιγμή n. Μία ακόμη σημαντική ιδιότητα του αλγορίθμου Metropolis είναι ότι ο πίνακας μετάβασης $W(\sigma,\sigma')$ παραμένει σταθερός με το χρόνο. Αυτές οι δύο ιδιότητες συνεπάγουν για τη χρονική εξέλιξη του p_n

$$p_{n+1}(\sigma) = \sum_{\sigma'} W(\sigma,\sigma')p_n(\sigma')$$

Η ερώτηση κλειδί είναι τώρα πώς να εξασφαλίσουμε ότι $p_n \xrightarrow{n\to\infty} \tilde{p}$. Χρήσιμες πληροφορίες μπορούν να ληφθούν από τη θεωρία για τις αλυσίδες Markov με σταθερούς πίνακες μεταβάσεων («στάσιμες» αλυσίδες Markov). Αναφορικά έχουμε:

- **Η κατανομή $\tilde{p}$ είναι στάσιμη.** Αυτό σημαίνει ότι $p_n = \tilde{p} \Rightarrow p_{n+1} = \tilde{p}$. Ένας άλλος τρόπος να δηλωθεί αυτό είναι ότι η $\tilde{p}$ πρέπει να είναι ιδιοδιάνυσμα του πίνακα W με ιδιοτιμή 1 δηλαδή:

$$\tilde{p} = \sum_{\sigma'} W(\sigma,\sigma')p(\sigma'), \quad \forall\sigma$$

- **Η διαδικασία είναι εργοδική**. Χωρίς να είμαστε αυστηροί, αυτό σημαίνει ότι κάθε κατάσταση μπορεί να επιτευχθεί από κάθε άλλη κατάσταση.

Δεν θα αποδείξουμε ότι αυτές οι δύο απαιτήσεις είναι επαρκείς για να εξασφαλίσουν τη συνθήκη $p_n \xrightarrow{n\to\infty} \tilde{p}$, αλλά για να δώσουμε μια ιδέα για το πώς λειτουργεί, θα αποδείξουμε μία λιγότερο ισχυρή δήλωση. Για αυτόν το λόγο, πρέπει να ορίσουμε την απόσταση μεταξύ δύο αυθαίρετων κατανομών p_a, p_b,

$$\|p_a - p_b\| = \sum_{\sigma} |p_a(\sigma) - p_b(\sigma)|$$

Δήλωση. Αν η $\tilde{\mathbf{p}}$ είναι στάσιμη η απόσταση $\|\mathbf{p_n} - \tilde{\mathbf{p}}\|$ είναι μη αύξουσα συνάρτηση του $\mathbf{n}$.

Με χρήση της $\lim_{n\to\infty} p_n(\sigma) = \tilde{p}(\sigma)$ και του γεγονότος ότι η κατανομή $\tilde{p}$ είναι στάσιμη έχουμε:

$$\|p_n - \tilde{p}\| = \sum_{\sigma} |p_{n+1}(\sigma) - \tilde{p}(\sigma)| =$$

$$\sum_{\sigma} \left| \sum_{\sigma'} W(\sigma, \sigma')(p_n(\sigma') - \tilde{p}(\sigma')) \right|$$

$$\leq \sum_{\sigma} \sum_{\sigma'} W(\sigma, \sigma') |p_n(\sigma') - \tilde{p}(\sigma')| =$$

$$\|p_n - \tilde{p}\|, \ \left\{ \sum_{\sigma'} W(\sigma, \sigma') = 1 \right\}$$

Ο αλγόριθμος Metropolis παρέχει έναν απλό και γενικό τρόπο για να εξασφαλιστεί τη στασιμότητα της $\tilde{p}$. Αυτό επιτυγχάνεται με το σχεδιασμό της βασικής αναπροσαρμογής του συστήματος με τέτοιο τρόπο ώστε να ικανοποιείται η αναλυτική ισορροπία, μία ισχυρότερη συνθήκη. Ο αλγόριθμος Metropolis μπορεί να εφαρμοστεί εύκολα και σε συνεχή συστήματα.

Βιβλιογραφία

1. *Maxima reference manual*, Version 5.13.0.
2. *A 10 minute tutorial for solving math problems with Maxima*, Antonio Cangiano.
3. *Introduction to Maxima*, Richard Rand.
4. *The CAS Maxima*, Boris Gaertner.
5. *Minimal Maxima*, Robert Dodier.
6. *A morsel of Maxima*, Alasdair McAndrew.
7. *Introduction to Dynamical Systems*, Jaime E. Villate
8. *MAHEMATICA και εφαρμογές*, Στέφανος Τραχανάς.
9. *http://maxima.sourceforge.net/*, η ιστοσελίδα του Maxima.
10. *Nonlinear dynamics and chaos*, Steven Strogatz.
11. *Chaos and integrability in nonlinear dynamics*, Michael Tabor.
12. *Lab tasks - System Theory (Lund, Sweden)*, Bo Söderberg.
13. *Introduction to Dynamical Systems*, Jaime E. Villate
14. *http://www.astro.auth.gr/~kokkotas/lesson/na_book/*
15. *Numerical methods for physicists – Lecture notes*, L. Lönnblad
16. *Numerical methods for ODE*, J. C. Butcher
17. *Computational physics*, M. Hjorth-Jensen