Λύση Απλών Διαφορικών Εξισώσεων με τις μεθόδους Euler και Euler-Cromer

Μέθοδος του Euler – Επίλυση διαφορικής εξίσωσης

Ξέρουμε ότι η ελεύθερη πτώση σωμάτων περιγράφεται από δύο διαφορικές εξισώσεις:

$$v = \frac{dy}{dt} \qquad \text{και} \qquad -g = \frac{dv}{dt}$$

Η αναλυτική λύση των εξισώσεων αυτών προκύπτει μετά από ολοκλήρωση, στη μορφή:

$$y(t) = y_0 + v_0 t - \frac{1}{2} g t^2 \quad \text{και} \quad v(t) = v_0 - gt$$

Οι προηγούμενοι ορισμοί εμπεριέχουν τον ορισμό της παραγώγου όπου εξετάζεται η μεταβολή του $\Delta y/\Delta t$ στο όριο που το χρονικό διάστημα είναι $\Delta t \to 0$

Ο ορισμός αυτός δεν μπορεί να χρησιμοποιηθεί σε Η/Υ. Ωστόσο μπορούμε να χρησιμοποιήσουμε διαφορετικές προσεγγίσεις.

Έστω ότι γνωρίζουμε την παράγωγο μιας συνάρτησης ως προς την ανεξάρτητη μεταβλητή (έστω για ευκολία αυτή είναι ο χρόνος t)

Ξέρουμε επομένως ότι: $\frac{dy}{dt} = f(y,t)$

Στην προηγούμενη εξίσωση, y είναι η λύση που ψάχνουμε να βρούμε και $f(y,t)$ είναι μια καλά συμπεριφερόμενη τυχαία συνάρτηση.

Μέθοδος του Euler – Επίλυση διαφορικής εξίσωσης

Το πρόβλημα που έχουμε να λύσουμε είναι να βρούμε την *y,* για οποιαδήποτε τιμή της ανεξάρτητης μεταβλητής *t* όταν ξέρουμε την αρχική τιμή της *y,* (έστω y_0) σε κάποια τιμή της ανεξάρτητης μεταβλητής (έστω t=0)

Θα θέλαμε επομένως να βρούμε την τιμή της *y* για τιμή της ανεξάρτητης μεταβλητής $t+\Delta t$ όταν ξέρουμε την τιμή της *y* για *t.*

Αλλά αυτή η τιμή μπορεί να προσεγγισθεί από το … ανάπτυγμα Taylor της συνάρτησης:

$$y(t+\Delta t) = y(t) + \frac{dy}{dt}\Delta t + \frac{1}{2!}\frac{d^2y}{d^2t}\Delta t^2 + \frac{1}{3!}\frac{d^3y}{d^3t}\Delta t^3 + \cdots$$

Το ευκολότερο που θα μπορούσαμε να κάνουμε στην προσέγγιση είναι να αγνοήσουμε όρους οι οποίοι είναι $2^{ης}$ και μεγαλύτερης τάξης σε Δt (αν το Δt έχει επιλεγεί αρκετά μικρό τότε οι όροι Δt^{η} θα είναι πολύ πολύ πιο μικροί).

Μπορούμε να γράψουμε επομένως: $y(t+\Delta t) = y(t) + \frac{dy}{dt}\Delta t + O(\Delta t^2)$

Ο τελευταίος όρος, αποτελεί την μέγιστη συνεισφορά στο σφάλμα της προσέγγισης που κάνουμε και είναι της τάξης του $\frac{1}{2!}\frac{d^2y}{d^2t}\Delta t^2$

Η παραπάνω προσέγγιση για την λύση μιας διαφορικής εξίσωσης ονομάζεται μέθοδος του Euler και είναι ακριβής σε τάξη $(\Delta t)^2$. Προφανώς για γραμμικές συναρτήσεις η 2^{η} παράγωγος είναι μηδέν και επομένως το σφάλμα μηδενίζεται

Μέθοδος του Euler – Επίλυση διαφορικής εξίσωσης

Με βάση τον προηγούμενο φορμαλισμό μπορούμε να γράψουμε τα βήματα της μεθόδου *του Euler*

$$y(t+\Delta t) = y(t) + \frac{dy}{dt}\Delta t = y(t) + f(y,t)\Delta t$$ όπου χρησιμοποιήσαμε: $$\frac{dy}{dt} = f(y,t)$$

Θα πρέπει να υπολογίσουμε την τιμή του $$\frac{dy}{dt} = f(y,t)$$

στην τιμή *t* της ανεξάρτητης μεταβλητής

Θεωρώντας ότι *y*(t=0) = 0 και ότι Δ*t*=*h* μπορούμε να γράψουμε την μέθοδο Euler ως:

Έστω ότι: $t_n = \eta h$ και $y(t_n)=y_n$ για n=0, 1, 2, …

$$y_1 = y_0 + f(y_0,t)h$$
$$y_2 = y_1 + f(y_1,t)h$$

$$y_{n+1} = y_n + f(y_n,t)h$$

Για την περίπτωση της ελεύθερης πτώσης και για τον υπολογισμό της ταχύτητας έχουμε:

$$\frac{dv}{dt} = f(v,t) = -g \qquad \frac{dy}{dt} = f(y,t) = v$$

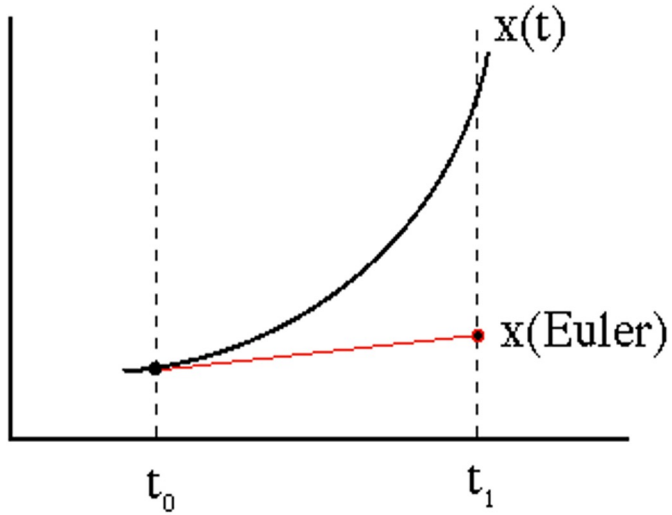
Ο αλγόριθμος του Euler

- Η εφαρμογή της μεθόδου του Euler στον υπολογιστή, εμπεριέχει τα ακόλουθα βήματα
  - Εισάγουμε την αρχική συνθήκη, το μέγεθος και αριθμό των βημάτων (Δt ή Δx) που ορίζονται μέσα στο διάστημα $[x_0,x_n]$ ή $[t_0,t_n]$
  - Υπολογίζουμε τη τιμή της συνάρτησης και της παραγώγου στην αρχή κάθε διαστήματος
  - Υπολογίζουμε τη τιμή της συνάρτησης στο τέλος του διαστήματος
  - Επαναλαμβάνουμε n φορές τα βήματα 2 και 3, όσα είναι τα βήματα που έχουμε καθορίσει βάσει του μεγέθους του βήματος και του ολικού διαστήματος

Για να καθορίσουμε την ευστάθεια του αλγόριθμου θα πρέπει να τρέξουμε το πρόγραμμά μας πολλές φορές για διαφορετικές τιμές για το μέγεθος του βήματος (Δx ή Δt) και να δούμε για ποιες τιμές τα αποτελέσματα του αλγορίθμου παραμένουν σταθερά.

Αυτό γιατί ο αλγόριθμος δίνει αποτελέσματα που συμφωνούν με την αναλυτική λύση για μικρές τιμές των διακριτοποιημένων διαστημάτων ενώ για μεγάλες τιμές αποκλίνει του σωστού αποτελέσματος

Γραφική Αναπαράσταση – Μέθοδος Euler



Ουσιαστικά αυτό που κάνουμε με τη μέθοδο του Euler είναι να ακολουθήσουμε την εφαπτομένη της καμπύλης της λύσης για το διάστημα (t_1-t_0)

Κατόπιν υπολογίζουμε και πάλι την κλίση της καμπύλης στο νέο σημείο και προχωρούμε με βάση τη κλίση αυτή στο επόμενο διάστημα

Αν θυμηθούμε, από το ανάπτυγμα Taylor έχουμε:

$$x(t) = x(t_0) + x'(t)\delta t + \frac{1}{2}x''(t)\delta t^2 + \cdots$$

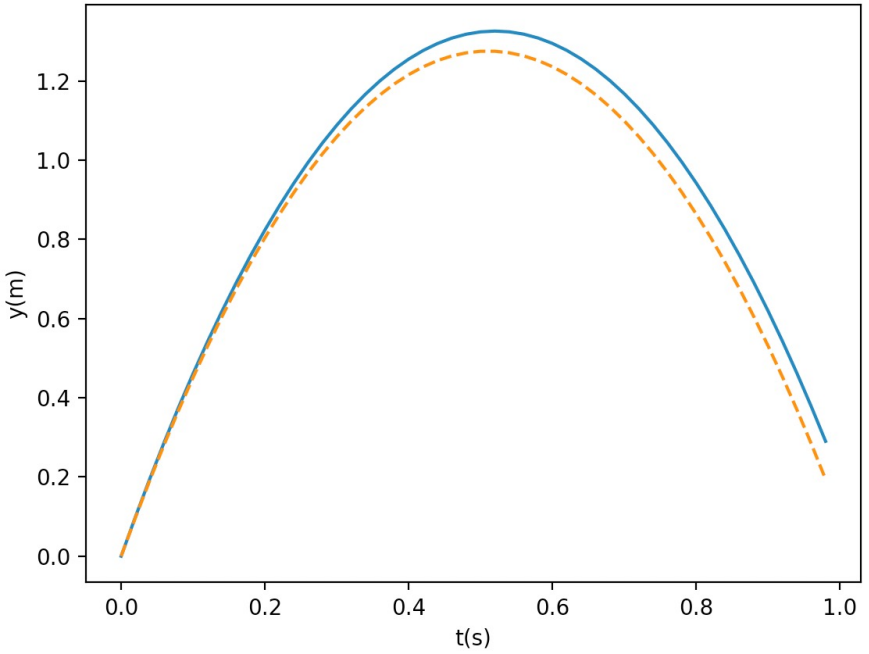
Χρησιμοποιώντας την κλίση σε κάθε διάστημα ουσιαστικά χρησιμοποιούμε τη 1η παράγωγο και αγνοούμε τους ανώτερους όρους τάξης $O(\Delta t^2)$

Το σφάλμα επομένως σε κάθε βήμα είναι της τάξης $O(\Delta t^2)$ και για N βήματα σε ένα συγκεκριμένο διάστημα αυξάνει σχεδόν γραμμικά.

Επειδή ο αριθμός των βημάτων είναι ανάλογος του διαστήματος $1/\Delta t$ ουσιαστικά το σφάλμα είναι γραμμικό με Δt

Ελεύθερη πτώση σωμάτων – Μέθοδος Euler

```
#/usr/bin/python3

import numpy as np
import matplotlib.pyplot as plt

g,dt = 9.8, 0.02    # g and time step

y,v0 = 0.0, 5.0    # arxiki thesi toy swmatos kai taxytita

ta, ya = [],[]

t, yb = 0.0, []    # yb is the theory curve

v = v0             #
while t < 1.0:     # ekseliksi tou sustimatos gia 1 sec
    ta.append(t)
    ya.append(y)
    yb.append(v0*t - g * t * t/2.0)    # theory

    y = y + v*dt      # Euler vima gia ti thesi
    v = v - g*dt      # Euler vima gia tin taxytita
    t = t + dt

plt.figure()
plt.plot(ta,ya,ta,yb,'--')

plt.xlabel('t(s)')
plt.ylabel('y(m)')
plt.show()
```



Ανάγκη βελτίωσης της μεθόδου του Euler

- Η μέθοδος κάνει γραμμική εξέλιξη ως προς την ανεξάρτητη μεταβλητή και επομένως το σφάλμα αυξάνει ως h^2

  Οι εξισώσεις αντιστοιχούν στο πρώτο όρο του αναπτύγματος γύρω από το σημείο t_0 ή x_0 και επομένως είναι τάξης h^2, $O(h^2)$

  → Η μέθοδος Euler δεν είναι ακριβής.

- Ο φορμαλισμός της μεθόδου του Euler είναι ασύμμετρος.

  Προχωρά τη λύση μέσω του διαστήματος h αλλά χρησιμοποιεί την πληροφορία σχετικά με την παράγωγο μόνο στην αρχή του διαστήματος

- Αν έχουμε τις εξισώσεις κίνησης ενός σώματος γράφουμε: $\frac{d\vec{v}}{dt} = \vec{a}(\vec{r},\vec{v})$; $\frac{d\vec{r}}{dt} = \vec{v}$

- Με τη μέθοδο του Euler θα γράψουμε: $\vec{v}_{n+1} = \vec{v}_n + h\vec{a}_n \quad \vec{r}_{n+1} = \vec{r}_n + h\vec{v}_n$

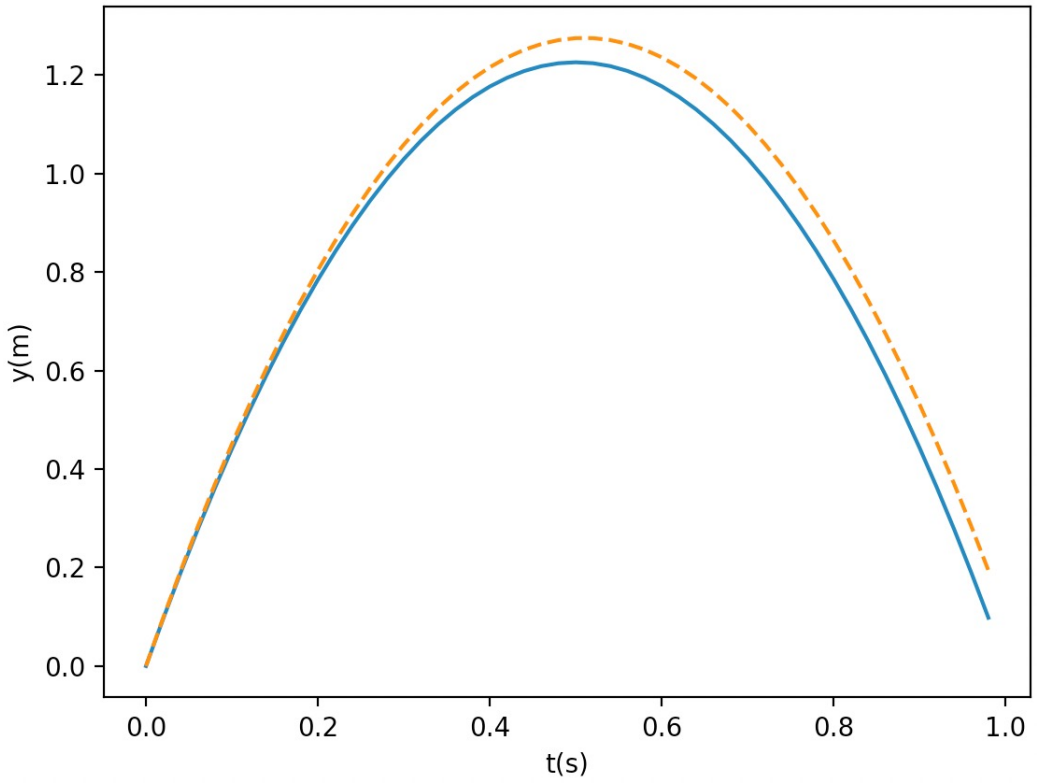
- Μια απλή παραλλαγή της μεθόδου είναι η χρησιμοποίηση των εξισώσεων:

  $\vec{v}_{n+1} = \vec{v}_n + h\vec{a}_n \quad \vec{r}_{n+1} = \vec{r}_n + h\vec{v}_{n+1}$ ← Euler-Cromer (χρήση της νέας ταχύτητας)

  - Δεν κερδίζουμε όμως τίποτα σε ακρίβεια. Απλά η μέθοδος αυτή δίνει σωστότερα αποτελέσματα για μια κατηγορία προβλημάτων

Ελεύθερη πτώση σωμάτων – Μέθοδος Euler - Cromer

```
#/usr/bin/python3

import numpy as np
import matplotlib.pyplot as plt

g,dt = 9.8, 0.02    # g and time step

y,v0 = 0.0, 5.0     # arxiki thesi toy swmatos
                    # kai taxytita

ta, ya = [],[]

t, yb = 0.0, []     # yb is the theory curve

v = v0              #
while t < 1.0:      # ekseliksi tou
                    # sustimatos gia 1 sec
    ta.append(t)
    ya.append(y)
    yb.append(v0*t - g * t * t/2.0)    # theory

                    # Sti methodo toy Euler-Crommer
                     # prwta upologizetai i taxutita
                     # (i paragwgos tis thesis) sti
                     # xroniki stigmi t kai katopin
                     # xrisimopoieitai i paragwgos
                     # ayti gia ton upologismo tis
                     # thesis tou swmatos,
    v = v - g*dt     # Euler vima gia tin taxytita
    y = y + v*dt     # Euler-Crommer vima gia ti thesi
    t = t + dt

plt.figure()
plt.plot(ta,ya,ta,yb,'--')

plt.xlabel('t(s)')
plt.ylabel('y(m)')
plt.show()
```



Κίνηση σώματος σε 2 διαστάσεις με αντίσταση ρευστού

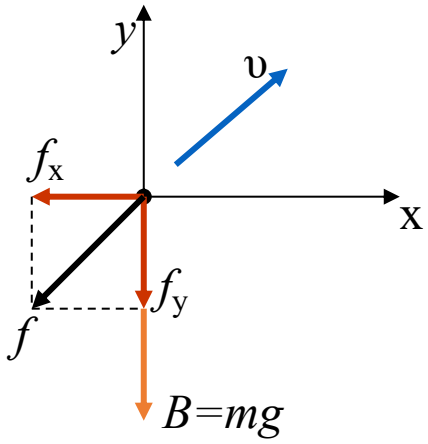
Θεωρούμε ότι έχουμε ένα βλήμα το οποίο κινείται με αρχική ταχύτητα υ σε ένα μέσο

Έστω ότι η δύναμη της αντίστασης του μέσου είναι: $f = -D\,\upsilon^2 \Rightarrow \vec{f} = -D\upsilon^2\hat{\upsilon}$

$$\Rightarrow \vec{f} = -D\upsilon^2\frac{\upsilon_x\hat{\imath} + \upsilon_y\hat{\jmath}}{\upsilon} \Rightarrow \vec{f} = -D\upsilon\left(\upsilon_x\hat{\imath} + \upsilon_y\hat{\jmath}\right) \Rightarrow f_x\hat{\imath} + f_y\hat{\jmath} = -D\upsilon\left(\upsilon_x\hat{\imath} + \upsilon_y\hat{\jmath}\right)$$

Η διεύθυνση της *F* είναι αντίθετη αυτής της υ και το μέτρο της είναι:

$$f_x = -D\upsilon\upsilon_x \quad \text{και} \quad f_y = -D\upsilon\upsilon_y \quad \text{ενώ} \quad f = \sqrt{f_x^2 + f_y^2} \quad \text{και} \quad \upsilon = \sqrt{\upsilon_x^2 + \upsilon_y^2}$$



Επομένως μπορούμε να γράψουμε:

$$\sum F_x = f_x = -D\upsilon\upsilon_x = ma_x \qquad \sum F_y = f_y = -(mg + D\upsilon\upsilon_y) = ma_y$$

$$a_x = -(D/m)\upsilon\upsilon_x \qquad a_y = -g - (D/m)\upsilon\upsilon_y$$

Η σταθερά D εξαρτάται από την πυκνότητα του αέρα, ρ, το σχήμα Α του σώματος (επιφάνεια όπως φαίνεται από εμπρός) και από μια αδιάσταση σταθερά C που ονομάζεται σταθερά αντίστασης

$$D = \frac{\rho C A}{2}$$

Κίνηση σώματος σε 2 διαστάσεις με αντίσταση ρευστού

- Για ένα αρκετά μικρό χρονικό διάστημα Δt μπορούμε να τη θεωρήσουμε σταθερή

- Στο χρονικό διάστημα Δt η μέση x(y)-συνιστώσα της επιτάχυνσης είναι:

$$a_x = \frac{\Delta \upsilon_x}{\Delta t} \quad \text{και} \quad a_y = \frac{\Delta \upsilon_y}{\Delta t}$$

ενώ η ταχύτητα αλλάζει κατά: $\Delta \upsilon_x = a_x \Delta t$ και $\Delta \upsilon_y = a_y \Delta t$

- Επομένως στο τέλος του διαστήματος Δt η ταχύτητα έχει τιμή

$$\upsilon_x + \Delta \upsilon_x = \upsilon_x + a_x \Delta t \qquad \upsilon_y + \Delta \upsilon_y = \upsilon_y + a_y \Delta t$$

- Ανάλογα αλλάζουν οι συντεταγμένες του σώματος

$$\Delta x = (\upsilon_x + \Delta \upsilon_x)\Delta t = \upsilon_x \Delta t + \Delta \upsilon_x \Delta t = \upsilon_x \Delta t + \alpha_x \Delta t^2$$

$$x + \Delta x = x + \upsilon_x \Delta t + \alpha_x \Delta t^2$$

Κίνηση σώματος σε 2 διαστάσεις με αντίσταση ρευστού

```
import numpy as np
import matplotlib.pyplot as plt
dt=0.25
v0 = 700
theta = 55*np.pi/180
A_m = 4E-5
x,y = [],[]
x.append(0)
y.append(0)
xpos = 0.0
ypos = 0.0
vx = v0 * np.cos(theta)
vy = v0 * np.sin(theta)
istep = 0
doit = True
while doit:
    f = A_m * np.sqrt(vx**2 + vy**2) # Force
    vy = vy - 9.8 * dt - f * vy * dt
    vx = vx - f * vx * dt
    xpos = xpos + vx*dt # Euler-Cromer
    ypos = ypos + vy*dt
    x.append(xpos)
    y.append(ypos)
    istep = istep + 1
    if y[istep] <=  0.: # ktipise sto edafos
        doit = False
```

Γραμμική παρεμβολή για εύρεση βεληνεκούς

```
nstep = istep
slope = (y[nstep] – y[nstep-1])/(x[nstep] – x[nstep-1])
x[nstep] = x[nstep-1] – y[nstep-1]/slope
y[nstep] = 0

plt.plot(x,y)
plt.show()
```

Παράδειγμα: Αρμονικός ταλαντωτής

- Οριζόντιο ελατήριο με μάζα m στο ένα άκρο του. Η εξίσωση της κίνησης είναι:

$$m\frac{d^2x}{dt^2} = -kx$$ όπου x η απομάκρυνση από τη θέση ισορροπίας

- Θέτουμε τις αρχικές συνθήκες x(0) και υ(0) και μετατρέπουμε την αρχική εξίσωση σε σύστημα δύο εξισώσεων πρώτου βαθμού

$$\upsilon = \frac{dx}{dt} \qquad \frac{d\upsilon}{dt} = -\frac{k}{m}x$$

- Η πρακτική ερώτηση που παρουσιάζεται τώρα είναι πως μπορούμε να ελέγξουμε τις διάφορες μεθόδους αν δίνουν τα σωστά αποτελέσματα. Μια πολύ ισχυρή μέθοδος είναι αυτή των νόμων διατήρησης. Στη μηχανική όταν δεν έχουμε τριβές η ολική ενέργεια διατηρείται και μπορούμε να θέσουμε το ερώτημα κατά πόσο οι προσεγγιστικές μέθοδοι ικανοποιούν αυτή τη διατήρηση.
- Οι παραπάνω εξισώσεις κίνησης της μάζας εξαρτώμενης από ελατήριο δίνουν ενέργεια $E = m\upsilon^2/2 + kx^2/2$. Από τη μέθοδο του Euler οι εξισώσεις είναι

$$\left.\begin{aligned} x_{n+1} &= x_n + \upsilon_n h \\ \upsilon_{n+1} &= \upsilon_n + \left[(-k/m)x_n\right]h \end{aligned}\right\} \Rightarrow E_{n+1} = \frac{m\upsilon_{n+1}^2}{2} + \frac{kx_{n+1}^2}{2} = (1+\delta)E_n \qquad \delta = h^2\frac{k}{m}$$

Άρα $$E_{n+1} = (1+\delta)^n E_0 = (1 + n\delta + O(n^2\delta^2))E_0$$

Η μέθοδος Euler δεν διατηρεί τη ενέργεια

Σφάλμα – μη διατήρηση ενέργειας

- Μπορεί όμως να υποθέσουμε ότι η διαφορά για κάθε βήμα είναι μικρή αφού είναι της ίδιας τάξης μεγέθους με το σφάλμα της μεθόδου $O(h^2)$
- Το σφάλμα όμως είναι προσθετικό και μετά από n βήματα γίνεται nδ δηλαδή πρώτης τάξης στο δ. Αν το nδ γίνει συγκρίσιμο με τη μονάδα, η προσεγγιστική μέθοδος δεν έχει έννοια.
- Το σφάλμα, όντας προσθετικό, θέτει όρια για τον αριθμό των βημάτων που μπορούμε να χρησιμοποιήσουμε στη μέθοδο του Euler.

Για $n_{max}\delta \approx 1$ $\quad n_{max} = \frac{1}{\delta} = \frac{m}{kh^2}$

Βλέπουμε ότι ο μέγιστος αριθμός των βημάτων είναι ανάλογος της μάζας και αντιστρόφως ανάλογος της σταθεράς του ελατηρίου και του βήματος. Βλέπουμε επίσης ότι η ποσότητα $n_{max}h$ έχει διαστάσεις χρόνου και επειδή η περίοδος είναι ανάλογη με το $\sqrt{m/k}$ μας ενδιαφέρει $n_{max}h$ να είναι τουλάχιστον όσο η περίοδος.

Τυχαίοι Αριθμοί

και Monte Carlo

Monte Carlo και τυχαίοι αριθμοί

Τα προσδιορισμένα συστήματα (deterministic systems) περιγράφονται εν γένει από κάποιο μαθηματικό κανόνα

Κάποια συστήματα ωστόσο δεν είναι προσδιορισμένα Τυχαία ή στοχαστικά

Οποιαδήποτε διεργασία ή αλγόριθμος χρησιμοποιεί τυχαίους αριθμούς και αντιτίθεται σε προσδιορισμένους αλγόριθμους ονομάζεται Monte Carlo

Η μέθοδος Monte Carlo χρησιμοποιείται ευρέως στις επιστήμες:

Φυσική: προσομοίωση φυσικών διεργασιών

Μαθηματικά: αριθμητική ανάλυση

Βιολογία: προσομοίωση κυττάρων

Οικονομικά: εκτίμηση της διακύμανσης του χρηματιστηρίου αξιών

Μηχανική: προσομοίωση πειραματικών διατάξεων

Σημασία των μεθόδων Monte Carlo

Οι μέθοδοι Monte Carlo αποτελούν ένα από τα σημαντικότερα εργαλεία στη Φυσική

- Ανάλυση δεδομένων
- Προσομοίωση φυσικών γεγονότων που στηρίζονται σε τυχαίες διεργασίες - πιθανότητες
- Σχεδιασμό ανιχνευτών, βελτιστοποίηση και προσομοίωση

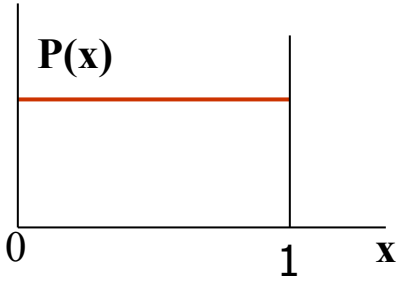
Επομένως ας μάθουμε μερικές από τις βασικές αρχές

- Σκοπός των γεννητόρων/προγραμμάτων Monte Carlo
- Γεννήτορες τυχαίων αριθμών
- Ολοκλήρωση
- Μερικά ιδιαίτερα δημοφιλή Monte Carlo προγράμματα

Τυχαίοι αριθμοί

Τυχαίος αριθμός είναι ένας αριθμός επιλεγμένος σαν να ήταν καθαρά τυχαία από μια συγκεκριμένη κατανομή

Σε μια ομοιόμορφη κατανομή τυχαίων αριθμών στο διάστημα [0,1), κάθε αριθμός έχει την ίδια τύχη να επιλεχθεί



Για παράδειγμα: Όταν ρίχνετε ένα ζάρι οι αριθμοί που μπορείτε να πάρετε είναι ομοιόμορφα κατανεμημένοι μεταξύ 1 και 6.

Κάθε αριθμός έχει την ίδια πιθανότητα να ‘‘βγεί’’

Γεννήτορες τυχαίων αριθμών

Ο καλύτερος τρόπος για να πάρουμε τυχαίους αριθμούς είναι να χρησιμοποιήσουμε μια διεργασία που συμβαίνει στη φύση.

- Ρίξιμο ενός ζαριού ή ενός νομίσματος
- Λόττο
- Τα αποτελέσματα του ποδοσφαίρου
- Η ραδιενεργός διάσπαση των πυρήνων

Φυσικά ο τρόπος αυτός για να διαλέξουμε τυχαίους αριθμούς δεν είναι ιδιαίτερα αποδοτικός

- Υπολογιστικές μέθοδοι αναπτύχθηκαν που κάνουν την ίδια διαδικασία

Πως μπορούμε όμως να κάνουμε κάποιο πρόγραμμα να υπολογίζει κάτι τυχαία;

- Με ένα γεννήτορα τυχαίων αριθμών

Γεννήτορες τυχαίων αριθμών

Γεννήτορας τυχαίων αριθμών είναι μία συνάρτηση η οποία δημιουργεί μια ακολουθία τυχαίων αριθμών

Όλοι οι υπολογιστές σήμερα περιέχουν στην βιβλιοθήκη τους ένα μηχανισμό για την δημιουργία ακολουθίας τυχαίων αριθμών οι οποίοι είναι ομοιόμορφα κατανεμημένοι στο διάστημα [0,1)

Η ακολουθία των αριθμών αυτών μπορεί να θεωρηθεί σαν ψευδο-τυχαία ακολουθία αφού για κάθε εκτέλεση του προγράμματος, θα πάρουμε και πάλι την ίδια ακολουθία τυχαίων αριθμών για την ίδια αρχική τιμή του ‘‘σπόρου’’ (seed) της ακολουθίας

Στην πραγματικότητα οι συναρτήσεις που καλούμε στον υπολογιστή χρησιμοποιούν μια μέθοδο (Lehmer 1948) στηριγμένη σε 32-bit ακεραίους και επομένως έχουν περίοδο το πολύ 2^{31}~10^9.

Αυτό είναι το πλήθος των τυχαίων αριθμών που μπορούν να δημιουργηθούν μέσα σε λίγα δευτερόλεπτα σε ένα μοντέρνο υπολογιστή

Η μέθοδος που ακολουθείται χρησιμοποιεί μια εξίσωση της μορφής:

$$x_{n+1} = mod[(ax_n + b), m]$$

όπου mod είναι το modulo. Οι σταθερές α,b και m διαλέγονται προσεκτικά ώστε η ακολουθία των αριθμών να γίνεται χαοτική και ομοιόμορφα κατανεμημένη

Γεννήτορες τυχαίων αριθμών $x_{n+1} = mod[(ax_n + b), m]$

Κανόνες

- Η πρώτη αρχική τιμή, x_0, (seed) επιλέγεται
- Η τιμή του m> x_0 και $a, b \geq 0$
- Το εύρος των τιμών είναι μεταξύ 0 και m
  (διαιρώντας με m μετατρέπεται μεταξύ 0 και 1)
- Η περίοδος του γεννήτορα αυτού είναι m-1

  Το m πρέπει να είναι αρκετά μεγάλο αφού η περίοδος δεν μπορεί ποτέ να γίνει μεγαλύτερη από m.

Για παράδειγμα:

Ας διαλέξουμε α=b=x_0=7 και m = 10 και από την σχέση: x_i =mod(a*x_{i-1}+b,m)

η ακολουθία ψευδο-τυχαίων αριθμών που θα πάρουμε θα είναι:

7, 6, 9, 0 7, 6, 9, 0 7, 6, 9, 0

Και διαιρώντας με 10 θα έχουμε την ακολουθία

0.7, 0.6, 0.9, 0.0 0.7, 0.6, 0.9, 0.0 0.7, 0.6, 0.9, 0.0 Πολύ κακή ακολουθία

Γεννήτορες τυχαίων αριθμών

Από τους πλέον δημοφιλής γεννήτορες είναι ο **RANDU** ο οποίος αναπτύχθηκε από την IBM το 1960 με τον ακόλουθο αλγόριθμο:

$$x_{n+1} = mod(65069x_n, 2^{31} - 1)$$

και αργότερα οι Park και Miller πρότειναν πως η εξίσωση $x_{n+1} = mod(16807x_n, 2^{31} - 1)$ δίνει την ελάχιστη συνθήκη για ένα ικανοποιητικό γεννήτορα

Τυχαίοι αριθμοί στην Python

Στην PYTHON μπορούμε να πάρουμε τυχαίους αριθμούς από την βιβλιοθήκη **random**

```
from random import randrange, seed, random
```

```
seed(42)
for i in range(4):
    print(randrange(10))
```

Τυπώνει ένα τυχαίο ακέραιο αριθμό από το 0, 9

```
    random()
```

Τυπώνει έναν τυχαίο δεκαδικό τυχαίο αριθμό sto [0,1)

```
    randrange(n)
```

Τυπώνει ένα τυχαίο ακέραιο αριθμό από το 0, n-1

```
    randrange(m,n)
```

Τυπώνει ένα τυχαίο ακέραιο αριθμό από το m, n-1

```
    randrange(m,n,k)
```

Τυπώνει ένα τυχαίο ακέραιο αριθμό από το m, n-1 με βηματα k-1

Οι συναρτήσεις αυτές δίνουν έναν νέο τυχαίο αριθμό κάθε φορά που καλούνται

MONTE CARLO ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ

ΕΥΡΕΣΗ ΑΚΡΟΤΑΤΩΝ ΣΥΝΑΡΤΗΣΗΣ

Monte Carlo βελτιστοποίηση

Μπορούμε να χρησιμοποιήσουμε τυχαίους αριθμούς για να βρούμε τη μέγιστη ή ελάχιστη τιμή μιας συνάρτησης πολλών μεταβλητών

Τυχαία αναζήτηση

Η μέθοδος αυτή υπολογίζει την συνάρτηση πολλές φορές σε τυχαία επιλεγμένες τιμές των ανεξάρτητων μεταβλητών.

Αν συγκεντρώσουμε ένα ικανοποιητικό αριθμό δειγμάτων τότε προφανώς θα έχουμε εντοπίσει και το ακρότατο.

Παράδειγμα: Χρησιμοποιήστε τη μέθοδο Monte Carlo για να υπολογίσετε το ελάχιστο της συνάρτησης $f(x) = x^2 - 6x + 5$ στο διάστημα x [1,5]

Η ακριβής λύση είναι f_{min}=-4.0 για x=3.0

Αλγόριθμος για ελάχιστα:

- Προσδιορισμός του πλήθους των πειραμάτων (N)
- Προσδιορισμός του διαστήματος [A,B]
- Αρχική τιμή για το ελάχιστο fmin=9E9 (πολύ μεγάλη τιμή)
- Επανάληψη της ακόλουθης διεργασίας N φορές)
  - Δημιουργία ενός τυχαίου αριθμού x στο [A,B]
  - Έλεγχος αν F(x) < fmin

    Αν ναι Βρήκαμε νέο ελάχιστο και κρατάμε τη τιμή του x