ΦΥΣ 145 – Υπολογιστικές Μέθοδοι στη Φυσική

Πρόοδος **1 Μαρτίου 2021**

Γράψτε το ονοματεπώνυμο και αριθμό ταυτότητάς σας στο πάνω μέρος της αυτής της σελίδας.

Η εξέταση αποτελείται από δύο τμήματα. Στο 1ο τμήμα διάρκειας 45 λεπτών θα πρέπει να απαντήσετε γραπτώς στα ερωτήματα που σας δίνονται. Στο τέλος του 45-λεπτου θα πρέπει να επιστρέψετε το 1ο τμήμα της δοκιμίου που σας δόθηκε. Στο 2ο μέρος της εξέτασης θα πρέπει να γράψετε στον υπολογιστή τα προγράμματα για τα προβλήματα που σας δίνονται. Πρέπει να απαντήσετε σε όλα τα προβλήματα που σας δίνονται.

Ο χρόνος εξέτασης είναι 150 λεπτά και ο συνολικός αριθμός μονάδων είναι 100 (30 και 70 μονάδες αντίστοιχα για το 1ο και 2ο μέρος).

Από τη στιγμή αυτή δεν υπάρχει συνεργασία/συζήτηση ανταλλαγή αρχείων και e-mails με κανένα και φυσικά κουδούνισμα κινητού που πρέπει να κλείσουν. Σημειώσεις, χαρτάκια κλπ απαγορεύονται όπως και επισκέψεις σε ιστοσελίδες ή accounts που δεν αναφέρονται στην ιστοσελίδα του μαθήματος. **Απαγορεύεται επίσης η χρήση του e-mail σας καθ΄ όλη τη διάρκεια της εξέτασης**.

Καλή επιτυχία

Ασκήσεις στο χαρτί

Άσκηση 1 [6μ]

(Α) Τι θα τυπώσει το ακόλουθο τμήμα κώδικα;

```
alphabet = "abcdefghijklmnopqrstuvwxyz"
for i in range(2,10,5):
     print("Letter %d is %s"%(i,alphabet[i])
```

(Β) Τι θα τυπώσει το ακόλουθο τμήμα κώδικα;

```
class Y:
     def __init__(self, v0):
          self.v0 = v0

     def __str__(self):
          return 'v0*t - 0.5*g*t**2; v0=%g'%self.v0
y = Y(5)
print(y)
```

(Γ) Τι θα τυπωθεί στην οθόνη όταν τρέξει το ακόλουθο πρόγραμμα;

```
a = 4
b = a
a = 2
print("b=", b):
```

Άσκηση 2 [6μ]

(A) Σημειώστε στο χαρτί τι θα τυπώσει το ακόλουθο τμήμα κώδικα;

```
def myfunc(a,b):
     c = a.copy()
     for k in b:
          if k in c:
               c[k] +=b[k]
          else:
               c[k] = b[k]
     return c

print(myfunc({1:-1,3:1},{1:3,2:2})
```

(B) Να γράψετε μια συνάρτηση *polyadd(p,q)* που να επιστρέφει την αναπαράσταση σε μορφή dictionary του αθροίσματος δύο πολυωνύμων των οποίων η αναπαράσταση σε μορφή dictionary είναι *p* και *q* αντίστοιχα. Για παράδειγμα, *polyadd*({1:-1, 3:1}, {1:3, 2:2}) θα πρέπει να επιστρέφει σαν αποτέλεσμα {1:2, 2:2, 3:1} εφόσον τα πολυώνυμα είναι $p = -x + x^3$ και το $q = 3x+2x^2$ και το αποτέλεσμα της πρόσθεσης $p + q = -x + x^3 + 3x + 2x^2 = 2x + 2\,x^2 + x^3$.

Άσκηση 3 [6μ]

Να γράψετε μια συνάρτηση με όνομα *Position*(*t, v0*) η οποία θα επιστρέφει την τιμή της συνάρτησης: $y(t) = v_o t - \frac{1}{2} g t^2$ και της παραγώγου της $y'(t)$. Να γράψετε επίσης ένα *main* πρόγραμμα το οποίο καλεί τη συνάρτηση *Position* για να υπολογίσει την $y(0.1)$ και $y'(0.1)$ για $v_0 = 2$. Να γράψετε τις δύο τιμές που επιστρέφει η συνάρτηση *Position*. Η παράμετρος g είναι η επιτάχυνση της βαρύτητας, η οποία μπορεί να τεθεί ίση με 9.81.

Άσκηση 4 [6μ]

Θεωρήστε τη μαθηματική συνάρτηση $f(t; p.q, r) = sin(\pi t) + psin(q\pi t) + rsin(4q\pi t)$. Θα πρέπει να την αναπαραστήσετε σε μια class με το όνομα *SineFunc*. Η class θα πρέπει να γραφεί με τέτοιο τρόπο ώστε το παρακάτω πρόγραμμα να δουλεύει όταν η class είναι αποθηκευμένη στο αρχείο SineFunc.py:

```
from SineFunc import SineFunc
f=SineFunc(p=0.1, q=4, r=0.05)
for t in [0.2, 0.4, 1, 2]:
     f_value = f(t)
     print('(%g)=%g'%(t,f value)
```

Άσκηση 5 [6μ]

Στο πρόγραμμα αυτό θα πρέπει να γράψετε μια class για την αναπαράσταση συνάρτησης παραβολής της μορφής: $f(x; a, b, c) = ax^2 + bx + c$. Το όνομα της class θα είναι *Parabola* και στην class αποθηκεύονται οι παράμετροι *a*, *b* και *c* ως *attributes* (χαρακτηριστικά δεδομένα) των αντικειμένων της class. Η class παρέχει τρεις μεθόδους:

__init__ για την αποθήκευση των σταθερών της παραβολής: *a*, *b* και *c*.

__call__ για τον υπολογισμό της τιμής της παραβολής f σε ένα τυχαίο σημείο *x*.

Roots για τον υπολογισμό των δύο ριζών της παραβολής:

$$x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a} \quad \text{και} \quad x_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}$$

Μπορείτε να υποθέσετε ότι μόνο real μη μιγαδικές λύσεις ενδιαφέρουν. Να γράψετε ένα σύντομο test πρόγραμμα το οποίο να υπολογίζει τις τιμές της παραβολής $-1 + x^2$ και επίσης να υπολογίζει τις ρίζες της.

ΦΥΣ 145 – Υπολογιστικές Μέθοδοι στη Φυσική

Πρόοδος **1 Μαρτίου 2021**

2ο Μέρος της Εξέτασης

Γράψτε το ονοματεπώνυμο και αριθμό ταυτότητάς σας στο πάνω μέρος της αυτής της σελίδας.

Στο 2ο μέρος της εξέτασης θα πρέπει να γράψετε στον υπολογιστή τα προγράμματα για τα τρία προβλήματα που σας δίνονται. Πρέπει να απαντήσετε σε όλα τα προβλήματα που σας δίνονται.

Ασκήσεις για τον Υπολογιστή

Δημιουργήστε ένα subdirectory midterm στον οποίο θα δουλέψετε τις παρακάτω ασκήσεις. Θα πρέπει στο τέλος της εξέτασης να δημιουργήσετε ένα tar file με όλα τα files τα οποία δημιουργήσατε. Το tar file θα πρέπει να βρίσκεται στο subdirectory midterm και να έχει όνομα με τη μορφή *<username>_midterm.tgz* όπου username ο e-mail account σας στο πανεπιστήμιο. Το file αυτό θα το στείλετε με e-mail στο fotis@ucy.ac.cy.

Άσκηση 1 [**20μ**]

(**α**) Να γράψετε ένα πρόγραμμα σε Python το οποίο υπολογίζει την παράγωγο της ακόλουθης συνάρτησης $f(x) = x^2 e^{-x}$ στο σημείο $x_0 = 3$, χρησιμοποιώντας την προσέγγιση τόσο της κεντρικής τιμής:

$$f_p^{(1)} = \frac{f(x_0 + \delta x) - f(x_0 - \delta x)}{2\delta x}$$

όσο και την προσέγγιση της μίας πλευράς:

$$f_p^{(2)} = \frac{f(x_0 + \delta x) - f(x_0)}{\delta x}$$

Θα πρέπει να υπολογίσετε και να συγκρίνετε το σφάλμα των δύο παραπάνω προσεγγιστικών μεθόδων με την αναλυτική τιμή της παραγώγου της συνάρτησης $f(x)$ συναρτήσει του *δx*. Συγκεκριμένα θα πρέπει να κάνετε το γράφημα της $e^{(1)} = \left|f_p^{(1)} - f'(x_0)\right|$ και $e^{(2)} = \left|f_p^{(2)} - f'(x_0)\right|$ ως προς *δx* σε ένα *log-log* γράφημα. Για *δx* θα πρέπει να χρησιμοποιήσετε τιμές $\delta x = 2^{-n}$, με *n* στο διάστημα [1,50].

(**β**) Στο ερώτημα αυτό θα διερευνήσετε πως μπορεί να εξαχθεί η παράγωγος με το να χρησιμοποιήσουμε αριθμητικά αποτελέσματα για να εκτιμήσουμε την τιμή της παραγώγου όταν *δx* = 0. Συγκεκριμένα, υπολογίστε $f_p^{(1)}$ για *δx* = 0.1, 0.05 και 0.025 και αποθηκεύστε τα αποτελέσματα σε ένα *numpy array* με όνομα *deriv*. Χρησιμοποιήστε την συνάρτηση *polyfit* του *module numpy* για να κάνετε προσαρμογή (*fit*) τον *array deriv* με πολυώνυμο δευτέρου βαθμού ως προς *δx*, το οποίο γράφεται όπως έχουμε δει ως ένας *array*, με τη μορφή: $e^{(1)} = p_0 \delta x^2 + p_1 \delta x + p_2$. Ο *array* με τους συντελεστές *p* επιστρέφεται μετά από κλίση της συνάρτησης *polyfit* ως εξής: ***p = polyfit(dx, deriv, 2)*** και *dx* είναι το *δx*.

Τυπώστε τους συντελεστές p_i και προσδιορίστε με τον τρόπο αυτό την εκτιμώμενη τιμή της παραγώγου για *δx* = 0 και την διαφορά της από την πραγματική τιμή. Πώς συγκρίνεται το σφάλμα αυτό με το σφάλμα που βρήκατε όταν χρησιμοποιήσατε την τιμή *δx* = 0.025;

(**γ**) Επαναλάβετε τα βήματα του υπο-ερωτήματος (**β**) χρησιμοποιώντας πολυώνυμο τρίτου βαθμού για την προσαρμογή του *array deriv*. Στην περίπτωση αυτή να χρησιμοποιήσετε τιμές που λαμβάνετε δίνοντας τιμές στο *δx* = 0.1, 0.05, 0.025 και 0.0125.

Άσκηση 2 [**25μ**]

Σε πολλές εφαρμογές της φυσικής χρειάζεται να υπολογιστεί το ολοκλήρωμα της κανονικής κατανομής πιθανότητας:

$$\int_{-x}^{x} e^{-t^2/2} dt.$$

Το ολοκλήρωμα αυτό δεν μπορεί να υπολογιστεί με την βοήθεια γνωστών συναρτήσεων και για την επίλυσή του χρησιμοποιείται μια νέα ιδιαίτερη συνάρτηση που ονομάζεται συνάρτηση σφάλματος και δίνεται από τη σχέση:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-t^2/2} dt$$

Για μικρές τιμές του *x* ένας καλός και αποδοτικός τρόπος υπολογισμού της *erf*(*x*) είναι με τη βοήθεια της σειράς:

$$\mathrm{erf}(x) = \frac{2x}{\sqrt{\pi}} \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)k!} x^{2k} \approx \frac{2x}{\sqrt{\pi}} \sum_{k=0}^{N-1} \frac{(-1)^k}{(2k+1)k!} x^{2k}$$

Να υπολογιστεί η τιμή της συνάρτησης *erf*(*x*) όταν το *x* παίρνει τιμές στο διάστημα [0.5-2.0] και αλλάζει με βήμα 0.25. Πόσους όρους θα πρέπει να χρησιμοποιήσετε για τον υπολογισμό της συνάρτησης όταν η ακρίβεια που ζητάτε είναι στο 5ο δεκαδικό ψηφίο; Το πρόγραμμά σας θα πρέπει να τυπώνει την τιμή του *x*, της συνάρτησης *erf*(*x*) που υπολογίζετε και το πλήθος των όρων που αθροίσατε για να φθάσετε στην επιθυμητή ακρίβεια. Τα αποτελέσματά σας θα πρέπει να τα γράψετε σε ένα file *askisi2.dat* το οποίο θα στείλετε μαζί με το πρόγραμμά σας. Για να τα αποθηκεύσετε στο αρχείο θα πρέπει να χρησιμοποιήσετε την *f-format* string.

Σημείωση: Παρατηρήστε ότι για τον υπολογισμό των όρων της σειράς δεν χρειάζεται να υπολογίσετε το παραγοντικό του αριθμού αλλά μπορείτε να πάρετε τον *N+1* όρο από τον *N* όρο.

Άσκηση 3 [**25μ**]

Σας δίνεται το αρχείο http://www2.ucy.ac.cy/~fotis/phy145/Exams/data.txt που περιέχει διάφορους αριθμούς του δεκαδικού και δεκα-εξαδικού συστήματος. Να γράψετε ένα πρόγραμμα σε Python που να υπολογίζει την κατανομή του πλήθους των ψηφίων στους ακέραιους του δεκαδικού συστήματος. Δηλαδή πόσοι αριθμοί αποτελούνται από 1 ψηφίο, πόσοι από δύο, τρία, … 9 ψηφία. Για παράδειγμα, αν υπήρχαν οι αριθμοί 23, 45, 104, 222, 103230, τότε το πρόγραμμά μας θα τυπώσει:

2-ψηφίων αριθμοί: 2
3-ψηφίων αριθμοί: 2
5-ψηφίων αριθμοί: 1

Περιπτώσεις στις οποίες δεν έχουμε αριθμούς με συγκεκριμένο πλήθος ψηφίων δεν τυπώνονται. Το πρόγραμμά σας θα πρέπει να περιέχει τη δομή ενός dictionary για τους υπολογισμούς σας. Το πρόγραμμά σας θα πρέπει να θεωρεί ότι οι ακέραιοι έχουν το πολύ 9 ψηφία. Θα πρέπει να τυπώνει σε φθίνουσα σειρά το πλήθος των αριθμών με *n*-ψηφία σύμφωνα με το παραπάνω παράδειγμα (δε θα πρέπει να τυπώνονται οι περιπτώσεις που δεν έχουν βρεθεί ακέραιοι με κάποιο αριθμό ψηφίων). Θα πρέπει να κάνετε επίσης τη γραφική παράσταση της κατανομής των ψηφίων και να την αποθηκεύσετε στο αρχείο *askisi03.pdf* το οποίο θα πρέπει να επιστρέψετε.