5η Εργαστηριακή Άσκηση

Όνομα: Βασίλειος Βλασσόπουλος

AM: 1083780

Explicit Euler :

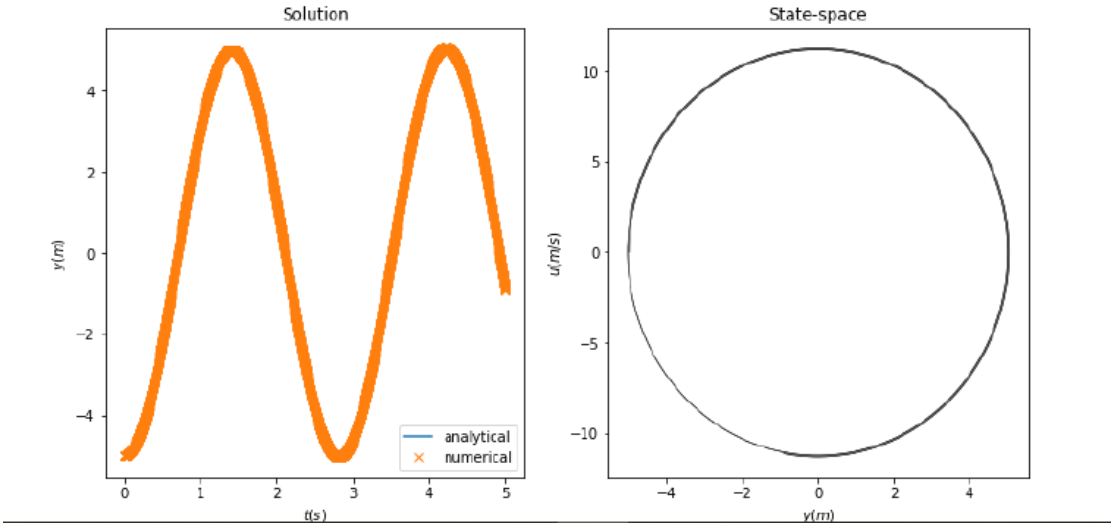
1) Ο συμπληρωμένος κώδικας είναι ο εξής :

```
# integration parameters
end_time = 5
dt = 0.01

# solution vectors
t = [0.0]
y = [y0]
u = [u0]

# numerical integration
while t[-1] < end_time:
    t0 = t[-1]
    y_t0 = y[-1]
    u_t0 = u[-1]

    #TASK calculate a, u and y
    a_t0 = -(k/m)*y_t0-(d/m)*u_t0
    u_t0_dt = u_t0+a_t0*dt
    y_t0_dt =y_t0 + u_t0*dt

    u.append(u_t0_dt)
    y.append(y_t0_dt)
    t.append(t0 + dt)
# visualization
visualize(t, y, u, calc_analytical_solution())
```

2) Παρατηρούμε ότι καθώς αλλάζουμε τις διάφορες παραμέτρους του προγράμματος αλλάζει η ευστάθεια του συστήματος. Συγκεκριμένα καθώς αυξάνουμε το dt αυξάνεται και η ευστάθεια (ισχύει και το αντίστροφο).Επίσης καθώς αυξάνεται το end time ,μεγαλώνει το μήκος της συνάρτησης που υπολογίζουμε με αποτέλεσμα να μεταβάλλεται η μορφή των αποτελεσμάτων.

In [10]:

```
# integration parameters
end_time = 5
dt = 0.01

# solution vectors
t = [0.0]
y = [y0]
u = [u0]

# numerical integration
while t[-1] < end_time:
    t0 = t[-1]
    y_t0 = y[-1]
    u_t0 = u[-1]

    #TASK calculate a, u and y
    a_t0 = -(k/m)*y_t0-(d/m)*u_t0
    u_t0_dt = u_t0+a_t0*dt
    y_t0_dt =y_t0 + u_t0*dt

    u.append(u_t0_dt)
    y.append(y_t0_dt)
    t.append(t0 + dt)
# visualization
visualize(t, y, u, calc_analytical_solution())
```



3)Το δεξιό διάγραμμα απεικονίζει την ταχύτητα συναρτήσει της παραμόρφωσης ελατηρίου Έτσι με βάση τις παραμέτρους που χαρακτηρίζουν το πρόβλημα το γράφημα να παίρνει ανάλογη μορφή. Άμα για παράδειγμα μειώσουμε το dt τότε το γράφημα τείνει να γίνει τέλειος κύκλος.

4) Η αριθμητική λύση διαφέρει σε σχέση με την πραγματική γιατί στο ανάπτυγμα Ταυλορ κρατάμε τους πρώτους όρους έτσι ώστε

να δημιουργείται σφάλμα O(h^2). Ένας άλλος παράγοντας είναι το σφάλμα στρογγύλευσης του υπολογιστή.

5) Για να προσεγγίσουμε καλύτερα την πραγματική λύση αυτό που μπορούμε να κάνουμε είναι να μικρύνουμε την μεταβλητή dt όσο το δυνατό περισσότερο έτσι ώστε να προσεγγίζει το μαθηματικό dt->0.Έτσι αυξάνεται η ακρίβεια της λύσης.

Semi-implicit

Συμπληρωμένος κώδικας:

In [15]:

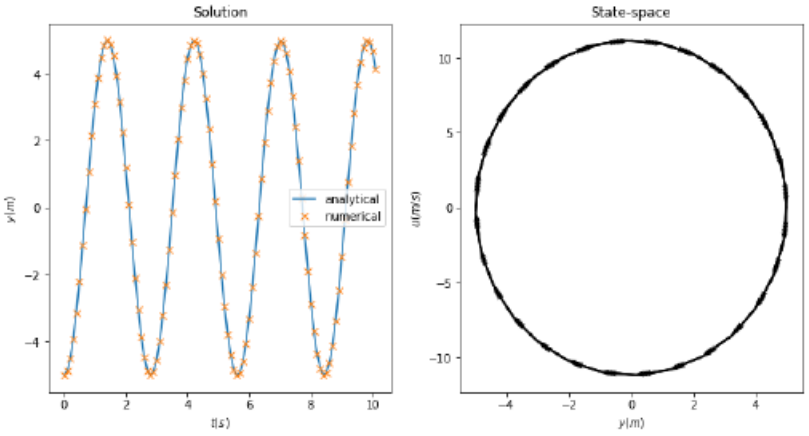
```
# integration parameters
end_time = 10
dt = 0.01

# solution vectors
t = [0]
y = [y0]
u = [u0]

# numerical integration
while t[-1] < end_time:
    t0 = t[-1]
    y_t0 = y[-1]
    u_t0 = u[-1]

    #TASK calculate a, u and y
    a_t0 = -(k/m)*y_t0-(d/m)*u_t0
    u_t0_dt = u_t0+a_t0*dt
    y_t0_dt = y_t0 + u_t0_dt*dt

    y.append(y_t0_dt)
    u.append(u_t0_dt)
    t.append(t0 + dt)

# visualization
visualize(t, y, u, calc_analytical_solution())
```

1) Η μέθοδος παρουσιάζει ευστάθεια όταν το δεξιό διάγραμμα σχηματίζει σπείρα που τείνει προς το κέντρο του κύκλου. Δηλαδή το πλάτος του ελατηρίου δεν πάει ποτέ στο άπειρο.
2) Το σφάλμα της μεθόδου αυτής είναι της τάξης του O(h^3).

The midpoint method and the Runge-Kutta method of Order 4

In [17]:

```
# integration parameters
end_time = 10
dt = 0.1

# solution vectors
t = [0]
x = [[y0, u0]]

# first-order derivative function
def f(x, t):
    return np.array([x[1], -k / m * x[0] - d / m * x[1]])

# numerical integration
while t[-1] < end_time:
    t0 = t[-1]
    x_t0 = np.array(x[-1])

    #Calculate k1, k2, k3, k4 and finally x_t0_dt
    k1 = f(x_t0,t0)*dt
    k2 = f(x_t0+k1/2,t0+dt/2)*dt
    k3 = f(x_t0+k2/2,t0+dt/2)*dt
    k4 = f(x_t0+k3,t0+dt)*dt
    x_t0_dt = x_t0+(1/6)*(k1+2*k2+2*k3+k4)


    x.append(x_t0_dt)
    t.append(t0 + dt)

# visualization
x = np.array(x)
visualize(t, x[:, 0], x[:, 1], calc_analytical_solution())
```



1)Το σφάλμα της συγκεκριμένης μεθόδου είναι μικρότερο από εκείνων των προηγούμενων μεθόδων και είναι της τάξης των O(h^4).

2) Η μέθοδος προσεγγίζει αρκετά κάλα την πραγματική λύση παρόλα αυτά αλλάζοντας της παραμέτρους (π.χ. αυξηση του dt) παρατηρούμε αλλαγές στο αποτέλεσμα (αποκλίνει περισσότερο σε σχέση με πριν από την πραγματική λύση).

Adaptive Time Stepping

1) Καθώς αυξάνουμε (μειώνουμε) την πυκνότητα των σημείων η λύση γίνεται πιο ακριβής (λιγότερο ακριβής), επειδή έχουμε περισσότερα (λιγότερα) σημεία .

2) Με την αύξηση των σημείων η προσσέγγιση είναι καλύτερη αλλά δεν είναι ίδια με την πραγματική.

3)Η odeint υποστηρίζει LSODA αλγόριθμους.