04.23

finiteDiff.py

1. '''
2. Find the first and second derivatives at x = 0, 0.2, and 0.4 of the following dataset
3. x = 0 0.1 0.2 0.3 0.4
4. f(x) = 0.0000 0.0819 0.1341 0.1646 0.1797
5. '''
7. x = [0,0.1,0.2,0.3,0.4]
8. f = [0.0,0.0819,0.1341,0.1646,0.1797]
10. # Finite difference in the abscissa h
11. h = x[1]-x[0]
13. nc = 2
14. # Forward difference calculation
15. f1f = (f[nc+1] - f[nc]) / h
16. f2f = (f[nc] - 2\*f[nc+1] + f[nc+2]) / h\*\*2
18. # Central difference calculation
19. f1c = (f[nc+1] - f[nc-1]) / (2\*h)
20. f2c = (f[nc+1] - 2\*f[nc] + f[nc-1]) / h\*\*2
22. # Backward difference calculation
23. f1b = (-f[nc-1] + f[nc]) / h
24. f2b = (f[nc-2] - 2\*f[nc-1] + f[nc]) / h\*\*2
26. r = lambda x:round(x,8)
28. print("Forward vs Central vs Backward f1 at x = 0.2 ",r(f1f),r(f1c),r(f1b))
29. print("Forward vs Central vs Backward f2 at x = 0.2 ",r(f2f),r(f2c),r(f2b))
31. # Forward difference at x = 0
32. nc = 0
33. f1f = (f[nc+1] - f[nc]) / h
34. f2f = (f[nc] - 2\*f[nc+1] + f[nc+2]) / h\*\*2
36. print('')
37. print('Forward f1 and f2 at x = 0 ',r(f1f),r(f2f))

 $ python finiteDiff.py

Forward vs Central vs Backward f1 at x = 0.2 0.305 0.4135 0.522

Forward vs Central vs Backward f2 at x = 0.2 -1.54 -2.17 -2.97

Forward f1 and f2 at x = 0 0.819 -2.97

richardson.py

1. ## f(x) = exp(-x), find f''(x = 1)
2. ## solution = 0.36787944117144233
3. ## Consider h from 1 to 0.001
5. import numpy as np
7. mapli = lambda f,x:list(map(f,x))
8. h = [1,10,50,100,500,1000]
10. h = mapli(lambda x:1/x,h)
12. for i in h:
13. x = -1
14. npr = 12
15. f1 = (round(np.exp(x + i),npr) - round(np.exp(x - i),npr)) / (2\*i)
16. print('Derivatives, h ',f1, i)

 $ python richardson.py

Derivatives, h 0.43233235838149997 1.0

Derivatives, h 0.36849288021500004 0.1

Derivatives, h 0.3679039669500006 0.02

Derivatives, h 0.3678855725000013 0.01

Derivatives, h 0.3678796862500028 0.002

Derivatives, h 0.36787950250000145 0.001

h가 줄어들수록 Derivatives 값이 수렴하는 것으로 보인다.

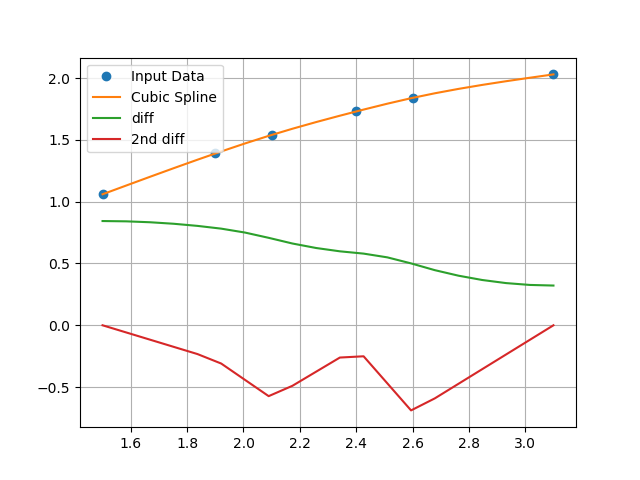
04.30

interpolation.py

1. import numpy as np
2. import matplotlib.pyplot as plt
4. x = np.array([1.5,1.9,2.1,2.4,2.6,3.1])
5. y = np.array([1.06,1.39,1.54,1.73,1.84,2.03])
7. xar = np.linspace(1.5,3.1,20)
9. import os
10. import sys
11. sys.path.append(os.path.dirname(os.path.abspath(os.path.dirname(\_\_file\_\_))))
13. from cubicSpline import \*
15. xData = x; yData = y
16. kvals = curvatures(xData,yData)
18. def SplineDiff(xData,yData,k,x):
19. def findSegment(xData,x):
20. iLeft = 0
21. iRight = len(xData) - 1
22. while 1:
23. if (iRight-iLeft) <= 1:
24. return iLeft
25. i = round((iLeft + iRight)/2)
26. # print('i ',i, 'x ',x,' xData[i]', xData[i])
27. if x < xData[i] :
28. iRight = i
29. else:
30. iLeft = i
32. i = findSegment(xData,x)
33. h = xData[i] - xData[i+1]
34. y = (3\*(x - xData[i+1])\*\*2/h - h)\*k[i]/6.0 \
35. - (3\*(x - xData[i])\*\*2/h - h)\*k[i+1]/6.0 \
36. + (yData[i] - yData[i+1])/h
37. return y
39. def SplineDiff2(xData,yData,k,x):
40. def findSegment(xData,x):
41. iLeft = 0
42. iRight = len(xData) - 1
43. while 1:
44. if (iRight-iLeft) <= 1:
45. return iLeft
46. i = round((iLeft + iRight)/2)
47. # print('i ',i, 'x ',x,' xData[i]', xData[i])
48. if x < xData[i] :
49. iRight = i
50. else:
51. iLeft = i
53. i = findSegment(xData,x)
54. h = xData[i] - xData[i+1]
55. y = (x - xData[i+1])/h\*k[i] - (x - xData[i])/h\*k[i+1]
56. return y
58. yar = []; dy = []; d2y = []
59. for i in range(len(xar)):
60. ydata = evalSpline(xData,yData,kvals,xar[i])
61. dydata = SplineDiff(xData,yData,kvals,xar[i])
62. d2ydata = SplineDiff2(xData,yData,kvals,xar[i])
63. yar.append(ydata)
64. dy.append(dydata)
65. d2y.append(d2ydata)
67. plt.plot(xData,yData,'o',xar,yar,'-',xar,dy,'-',xar,d2y,'-')
68. plt.legend(['Input Data','Cubic Spline','diff','2nd diff'])
69. plt.grid()
70. plt.show()
72. from polyFit import \*
73. m = 2
74. coeffs = polyFit(kvals,yData,m)
75. print( 'Coefficients ', coeffs)

 $ python interpolation.py

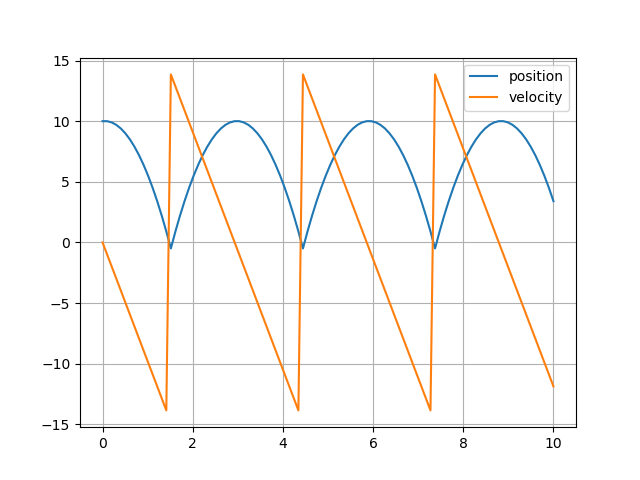
Coefficients [1.58253871 0.62313381 1.24690869]



freeFallMotion.py

1. import numpy as np
2. import matplotlib.pyplot as plt
4. # We wnt to find the height versus time
6. time = np.linspace(0,10,100)
7. y0 = 10 # Ten meters high
8. a = -9.8 # Meters/sec^2
9. v0 = 0

12. var = [0.0]
13. yar = [10.0]
14. for i in range(len(time)-1) :
16. dt = time[i+1] - time[i]
17. var.append( var[i] + a\*dt )
18. yar.append( yar[i] + var[i]\*dt )
20. if yar[i+1] < 0 : var[i+1] = -var[i]
22. plt.plot(time,yar,'-',time,var,'-')
23. plt.legend(['position','veloity'])
24. plt.grid()
25. plt.show()



I want to show moving visualized object so made physics engine using c++.

This is that program’s url.

<https://github.com/ingBE/comphy/blob/master/HW7/freefallMotion.exe>

and maby you need this .dll files.

<https://github.com/ingBE/comphy/blob/master/HW7/bin/>

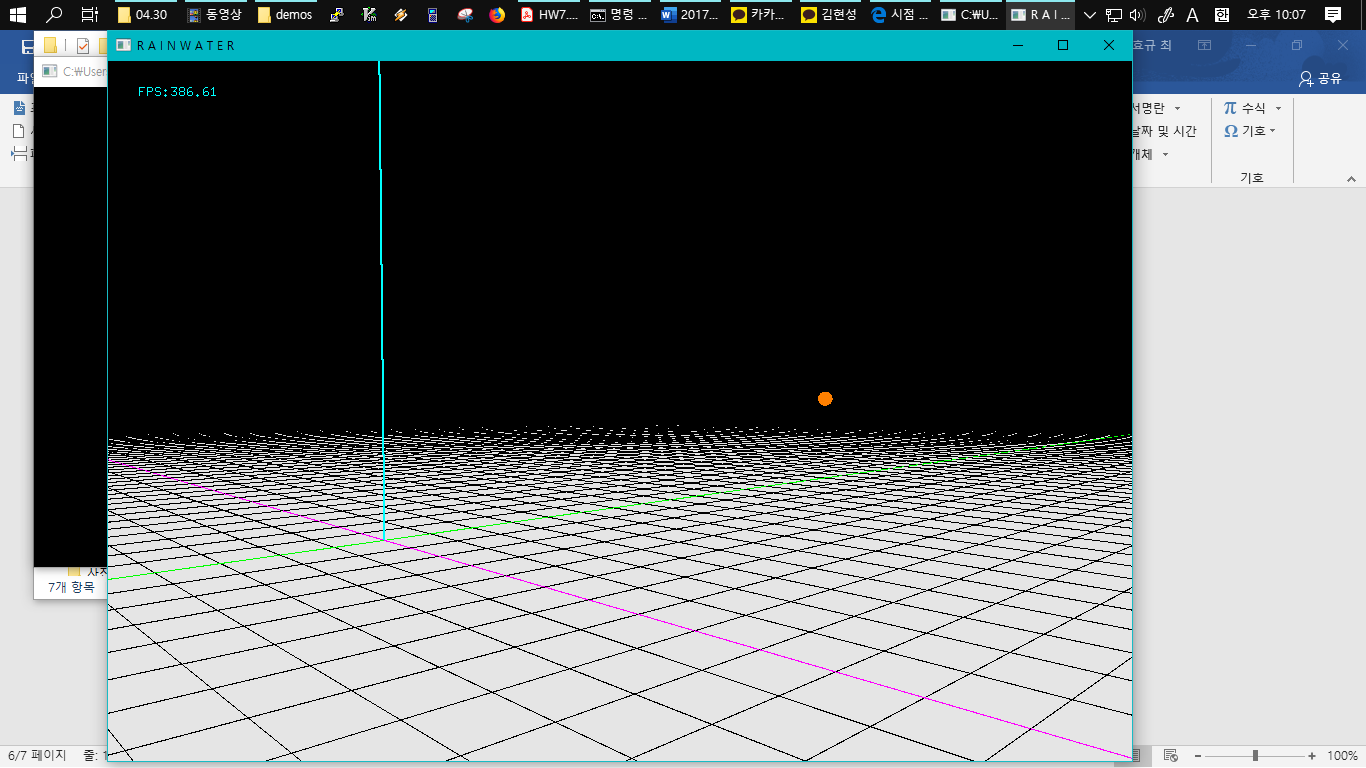
This is codes of physics engine and freeFallMotion program.

<https://github.com/ingBE/RWengine/blob/master/demos/freefallMotion.cpp>

Move in program to press w, a, s, d, space, c.

Control viewpoint to drag mouse.

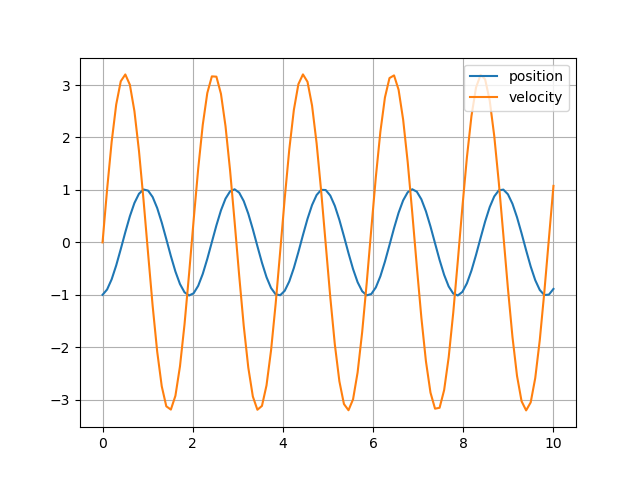
Press enter button for initializing position and velocity of object.



2. a)

HW7\_2a.py

1. import numpy as np
3. time = np.linspace(0,10,100)
4. x = [-1.0] # x0 = -1.0 meter
5. v = [ 0.0] # v0 = 0 meter/sec
6. m = 0.5 # kg
7. k = 5 # N/m
9. for i in range(len(time)-1):
11. dt = time[i+1] - time[i]
12. a = - k/m \* x[i] # calculate the accelerate
13. v.append( v[i] + a\*dt ) # calculate the velocity
14. x.append( x[i] + v[i+1]\*dt ) # calculate the new position
16. import matplotlib.pyplot as plt
17. plt.plot(time,x,'-',time,v,'-')
18. plt.legend(['position','velocity'])
19. plt.grid()
20. plt.savefig('HW7\_2a')

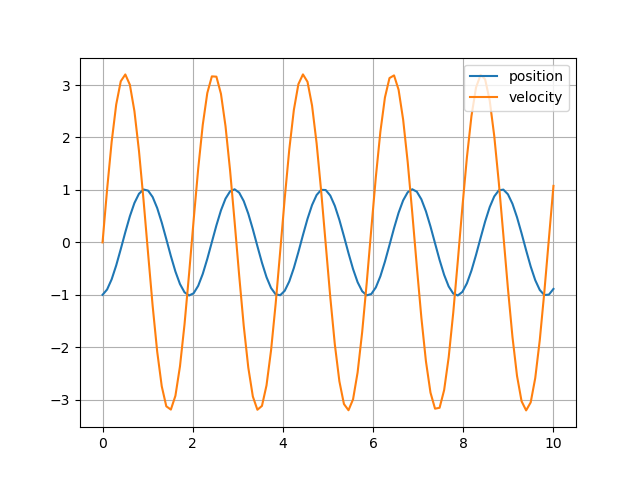
HW7\_2a.png

Position graph is same with solution of newton’s equation .

2. b)

1. import numpy as np
3. time = np.linspace(0,10,100)
4. x = [-1.0] # x0 = -1.0 meter
5. v = [ 0.0] # v0 = 0 meter/sec
6. m = 0.5 # kg
7. k = 5 # N/m
9. # calculate to use leapfrog method
10. for i in range(len(time)-1):
12. dt = time[i+1] - time[i]
13. a = - k/m \* x[i] # calculate the accelerate
14. v\_ = v[i] + a\*dt/2 # calculate v[i+1/2]
15. v.append( v\_ + a\*dt/2 ) # calculate the velocity
16. x.append( x[i] + v[i+1]\*dt ) # calculate the new position
18. import matplotlib.pyplot as plt
19. plt.plot(time,x,'-',time,v,'-')
20. plt.legend(['position','velocity'])
21. plt.grid()
22. plt.savefig('HW7\_2b')

HW7\_2b.png

It is almost same with HW7\_2a.png.