

Numerical Analysis with Python

LECTURE 01

Introduction to Numerical Modelling with Python

Introduction to Numerical Analysis

- 'Numerical methods are techniques by which mathematical problems are formulated so that they can be solved with arithmetic operations".
- There are many kinds of numerical methods, they have one common characteristic: they invariably involve large numbers of tedious arithmetic calculations.
- The role of numerical methods in engineering modelling and problem solving has increased dramatically in recent years due to the development of fast and efficient **Digital Computers**

1. Analytical or Exact methods:

" provided excellent insight into the behavior of some systems. However, analytical solutions can be derived for only a limited class of problems. These include those that can be approximated with linear models and those that have simple geometry and low dimensionality."

2. Graphical Methods:

"These graphical solutions usually took the form of plots or nomographs. Although graphical techniques can often be used to solve complex problems, the results are not very precise. Furthermore, graphical solutions (without the aid of computers) are extremely tedious and awkward to implement.|"

Non-computer Approach to Numerical Analysis

3. Calculator and slides rules:

"Manual calculations are slow and tedious. Furthermore, consistent results are elusive because of simple blunders that arise when numerous manual tasks are performed."

Merits of Numerical Methods in Engineering

- 1. They are capable of handling large systems of equations, nonlinearities, and complicated geometries that are often impossible to solve **analytically**.
- 2. They can be commercially produced as prepackaged software that can be use by even non-expert to solve engineering problems.
- 3. Expert with underlining knowledge of system models can produce their own numerical software to solve particular problem.
- 4. Numerical methods are an efficient vehicle for learning to use computers.
- 5. Numerical methods provide a vehicle for you to reinforce your understanding of mathematics.

Mathematical background for NA



Mathematical Modeling & Problem solving

• What is a mathematical Model ?

"is broadly defined as a formulation or equation that expresses the essential features of a physical system or process in mathematical terms"

It can take the form;

 $Dependent \ variable = f \begin{pmatrix} independent , & parameters, & forcing \\ variable & functions \end{pmatrix}$

- *Dependent variable:* characteristic that usually reflects the behavior or state of the system
- *Independent variable*: are usually dimensions, such as time and space, along which the system's behavior is being determined
- Forcing Functions: are external influences acting upon the system
- **Parameters**: are reflective of the system's properties or composition

Mathematical Modeling & Problem solving

• Traffic mathematical Model as system of linear equations



Mathematical Modeling & Problem solving

• Electrical circuit mathematical Model as system of linear equations



$$I_1 - I_2 + I_3 = 0$$

$$7I_1 + 4I_2 - 15 = 0$$

$$4I_2 + 2I_3 - 20 = 0$$

Model of a falling Object

• Assume a free falling object near the earth surface

Newtons Second law, net force, on the object F, F = ma

$$a = \frac{F}{m} \tag{1.1}$$

Where F, net force in N M, is the mass of the body in Kg A, is acceleration in m/s^2

Substituting
$$a = \frac{dv}{dt}$$
 and net force $F = F_D + F_V$ in (1.1)

$$\frac{dv}{dt} = \frac{F_D + F_V}{m} \tag{1.2}$$



Model of a falling Object

- The Net force F acting on the body consist of two components
- 1. The downward force F_D due to the gravitational drag
- 2. The upward force F_V due to air resistance

 $F_D = mg$ where g is accelaration due to gravity $F_V = -cv$

where c is a constant called **drag coefficient** and v is the air velocity Or

$$F = F_D + F_V = \text{mg} - c\nu \tag{1.3}$$

Hence substituting (1.3) in (1.2) we have (1.4)

$$\frac{dv}{dt} = g - \frac{c}{m}v \tag{1.4}$$

- The model that relates the acceleration of a falling object to the forces acting on it.
- It is a differential equation because it is written in terms of the differential rate of change $(\frac{dv}{dt})$ of the variable that we are interested in predicting.
- Simple algebraic manipulation can not solve the equation (1.4), hence advanced techniques in calculus is required.



• For example, if the parachutist is initially at rest (v=0 at t=0), calculus can be used to solve Eq. (1.4) for **V(t)**.

$$V(t) = \frac{gm}{c} \left(1 - e^{-\left(\frac{c}{m}\right)t}\right)$$
(1.5)
Exercise: prove (1.5)

Problem statement 1:

- A parachutist of mass 68.1 kg jumps out of a stationary hot air balloon. Using analytical solution of Eq. (1.5) to compute velocity prior to opening the chute. The drag coefficient is equal to 12.5 kg/s.
- Solution: Substituting the parameters in the equation below

$$V(t) = \frac{gm}{c} \left(1 - e^{-\left(\frac{c}{m}\right)t}\right)$$
$$V(t) = \frac{9.81(68.1)}{12.5} \left(1 - e^{-\left(\frac{12.5}{68.1}\right)t}\right)$$
$$V(t) = 53.44 \left(1 - e^{-0.18355t}\right)$$



• Computing velocities at different times yields the following results

 $V(t) = 53.44 (1 - e^{-0.18355t})$

Time(s)	Velocity (m/s)
0	0
2	16.41995476
4	27.79472025
6	35.67447948
8	41.13310679
10	44.91451827
12	47.53405465
14	49.34871325
16	50.60580051
18	51.47663561
20	52.07989823
22	52.4978026
24	52.78730183
26	52.98784963
28	53.12677718
30	53.22301791
00	53.44





• To solve the problem modelled by Eq. (1.4) numerically, we can use finite difference method (Euler's Method) to approximate $\frac{dv}{dt}$ in Eq. (1.4)

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}$$
(1.6)

• Now using (1.6) in Eq. 1.4 we have,

$$\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} = g - \frac{c}{m}v$$

• On rearranging the equation above

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c}{m}v(t_i)\right](t_{i+1} - t_i)$$
(1.7)

- $v(t_{i+1})$ is new velocity
- v(t_i) is old velocity,
- $(t_{i+1} t_i) = step size or time difference$

Problem statement 2

• A parachutist of mass 68.1 kg jumps out of a stationary hot air balloon. Use numerical solution of Eq. (1.7) to compute velocity prior to opening the chute. The drag coefficient is equal to 12.5 kg/s. Use $t_i = 0$ and $t_{i+1} = 2s$.

Solution: Substituting the parameters in the equation below

Step 1:
$$t_i = 0, v(t_i) = 0$$

stepsize
$$(t_{i+1} - t_i) = 2 - 0 = 2s$$

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c}{m}v(t_i)\right](t_{i+1} - t_i)$$



$$v(t_{i+1}) = \mathbf{0} + \left[9.81 - \frac{12.5}{68.1} * \mathbf{0}\right] * 2$$
$$v(t_{i+1}) = 19.62 \ m/s$$

Step 2: now at , $t_i = 2 s$, $v(t_i) = 19.62m/s$

$$v(t_{i+1}) = 19.62 + \left[9.81 - \frac{12.5}{68.1} * 19.62\right] * 2$$

 $v(t_{i+1}) = 32.04 m/s$

Step 3: : now at , $t_i = 4 s$, $v(t_i) = 32.04 m/s$

$$v(t_{i+1}) = 32.04 + \left[9.81 - \frac{12.5}{68.1} * 32.04\right] * 2$$

 $v(t_{i+1}) = 39.90 \ m/s$

• Repeating at subsequent intervals will yield the results below



Time(s)	Velocity (m/s)
0	0
2	19.62
4	32.03735683
6	39.89621262
8	44.8700259
10	48.01791654
12	50.01019387
14	51.27109186
16	52.06910513
18	52.57416198
20	52.89380883
22	53.09611102
	53 22414662
26	53 30517943
28	53 356/6/52
20	52 20002240
30	JJ.JOÖYZZ4Ö





- Given the two approaches (analytical and numerical), its obvious that the numerical approach can find solution similar to the exact solution.
- Tedious computation is needed to get more accurate results, but fortunately this can easily be done using computer.
- The numerical solution to falling object problem can be implemented using a software package like **PYTHON** and **MATLAB** to easily fine solution.
- Throughout the course PYTHON will be used for numerical analysis

- Python IDE and Installation: There many Integrated development environment (IDE) use to create python program.
- In this course Python SPYDER IDE distributed by Anaconda is recommended
- For instructions on how to download and install python SPYDER on different operating systems visit the link below:
- 1. <u>https://www.anaconda.com/distribution/</u>

• Anaconda is one of several Python distributors. Python distributions provide the Python interpreter, together with a list of Python packages and sometimes other related tools, such as editors.

- **Python packages:** For scientific computing and computational modelling, we need additional libraries (so called packages) that are not part of the Python standard library.
- The packages we generally need are:
- NumPy: (Numeric Python): For matrices and linear algebra
- **Pandas**: Python data science tools (Series and Dataframes)
- SciPy: (Scientific Python): many numerical routines
- **matplotlib**: (Plotting Library) creating plots of data
- **Sympy:** (Symbolic Python): symbolic computation
- **Pytest**: (Python Testing): a code testing framework

• Python Spyder IDE

Editor

Major Components

ē ×

H E

@ Spyder (Python 2.7 - 0 × Ele Edit Search Source Bun Debug Consoles Jools View Help 🕨 🕪 🚴 🏂 🔰 🔉 😤 🔅 🔝 🔳 🌆 🎲 🏹 🎸 + 🔶 C:Wisers Kenambermudez/Documents/GitHub/Simulators - 4-1 - 4 4 Editor - C: \Users\keriambermudez\Documents\GitHub\Simulations\reading_from_jacop.py 8 × Object inspector Source Console Object - D E 1 # -*- coding: utf-8 -*-3 Created on Tue Jun 23 18:50:27 2815 5 Bauthor: keriambermudez Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console. Object Variable utomatically after writing a left parenthese an activate this behavior in Preferences > 8 import pandas as pd 9 import os Inspector Explorer ...File 11 path = 'F:\\David Fenyos\\Image_Analysis\\General_measurements\\Testing_Ja 12 tables = [] 13 for i in os.listdir(path): 14 print(1) Explorer table = pd.read_table(path+i,sep='\n',header=None,names=[i[:-8]],skipr print(table) 17 table = table.transpose() 18 tables.append(table) Object inspector Variable explorer Pile explorer Profiler 20 all_table = pd.concat(tables) Console 1/A 22 #string 2 to # 23 #Pearson's Coefficient:, Overlap Coefficient, Overlap Coefficient:thresho Python 2.7.10 [Anaconda 2.3.0 (64-bit)] (default, May 28 2015, 16:44:52) [MSC v.1500 64 bit (AMD64)] 24 string_2_8 = [3,5,11] Type "copyright", "credits" or "license" for more information. 25 #string 3_8 + [10, 11, 20, 21, 24, 25, 28, 29] 26 0 12 10 IPython 3.2.0 -- An enhanced Interactive Python. 27 string_3_8 = [7,8,13,14,16,17,19,28] Anaconda is brought to you by Continuum Analytics. Please check out: http://continuum.io/thanks and https://anaconda.org 29 for i in string_2_8: -> Introduction and overview of IPython's features. test = all_table.ix[:,i].str[2:8] %quickref -> Quick reference. test = test.str.replace('(',' help -> Python's own help system. test = test.astype(float) object? -> Details about 'object', use 'object??' for extra details. all_table.ix[:,i]= test Xguiref -> A brief reference about the graphical user interface. 14 all_table.ix[1,i] In [1]: 36 for i in string_3_8: test = all_table.ix[:,i].str[3:8] 37 Console History log IPython console

IPython Console

- **Packages Installation:** all python packages can be installed via the python installation manager found in python version 3.4 and above.
- In python command window or Ipython (Spyder):

>> pip install numpy

- Will download and install NumPy packages
- In the program script:

import numpy as np

• Will import all libraries from NumPy packages and create an object np of those libraries.

• Flowchart and Pseudocodes



• Pseudocodes for solving falling Object Using Euler's Numerical Approach

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c}{m}v(t_i)\right](t_{i+1} - t_i)$$

- 1. Input g, c, m, stepsize
- 2. Initialize $v(t_i) = 0$, and $t_i = 0$
- 3. Compute $v(t_{i+1}) = v(t_i) + \left[g \frac{c}{m}v(t_i)\right] * stepsize$
- 4. Replace $v(t_i)$ with new velocity computed in step 3
- 5. If stoppage criteria is not met GOTO step 3 Else GOTO 6
 6. Return v(t_{i+1})
- 7. Stop

• Python function for solving falling Object Using Euler's Numerical Approach

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c}{m}v(t_i)\right](t_{i+1} - t_i)$$

Project explorer 🛛 🗗 🗙	Editor - C:\Users\OZEREN\Desktop\TIU\Courses\SPRING\Numerical_Analysis\NumericalAnalysisPython\Chapter1\FallingObject.py	♂ × Variable explorer	5 ×
🗸 📂 NumericalAnal	🗅 temp.py 🗵 FallingObject.py 🗵	🌣 🚣 🖹 🍡 🍠	0.
✓ ► Chapter1	1 # -*- coding: utf-8 -*-	Name Type Size Value	
🖹 FallingC	2 """ 3 Created on Fri Jan 10 21:02:01 2020	Time list 51 [0, 2, 4, 6, 8, 10, 12, 14,]	, 16, 18,
	4 5 @author: ASHIR	Velocity list 51 [0, 19.62, 32.0373568281938	336, 39.89
	6	cc tunle 2 (30 20)	
	7 8 #import numpy as np		
	9 #import sympy as sp		
	10 import matplotlib.pyplot as mpl		
	12 # Equation of Falling object under gravity using Eulers Numerical Method		
	$13^{****} v(t_{-}(i+1)) = v(t_{-}i) + [g-c/m v(t_{-}i)](t_{-}(i+1)-t_{-}i) ****$		
	14		
	16		
	17 def Euler_Method(g,c,m,stepsize,*argv):	Variable explorer File explorer Help Profiler	
	19 """ create list to hold the values of time and corresponding velocity """	IPython console	₽×
	20 time=[] 21 velocity=[]	Console 1/A 🔀	
	22 """ initialize time and velocities to 0"""	"data" keyword argument. IT such a	data ^
	23 t=0 24 V t=0	following arguments are replaced by	
	25 time.append(t)	**data[<arg>]**:</arg>	
	26 velocity.append(V_t)	* All arguments with the following na	mes:
	<pre>2/ 28 for iter in range(argv[0]):</pre>	'colors', 'x', 'ymax', 'ymin'.	
	29 V_t_1=V_t+(g-((c/m)*V_t))*stepsize	Objects passed as **data** must suppo	rt item
	30 31 """ update time and velocity and move to next step"""	access (``data[<arg>]``) and</arg>	C ICCI
	32 t=t+stepsize	<pre>membership test (``<arg> in data``).</arg></pre>	
	33 V_t=V_t_1 34	waitforbuttonpress(*args, **kwargs)	
	35 """ add new time and velocity in their list container """	Blocking call to interact with the figure	
	36 time.append(t) 37 velocity.append(V t)	This will return True is a key was presse	d, False
	38	if a mouse	
	39 40 return time velocity	button was pressed and None if "timeout" reached without	Nas
	41	either being pressed.	
	42 43 Time, Volasitur Evlag, Mathed/0, 81, 12, 5, 68, 1, 2, 50)	If *timeout* is negative, does not timeou	t.
	44		
	45 """ plot the results for 50 iterations """	winter() Set the colorman to "winter"	
	40 47 mpl.plot(Time, Velocity)	See the countries in whiter i	
	48	This changes the default colormap as well	as the
	50	image if there is one. See ``help(colorma	ps)``
	51	for more information.	
	52 53	xcorr(x, y, normed=True, detrend= <function< td=""><td>~</td></function<>	~
< >	54	V IPython console History log	

 Python function for solving falling Object Using Euler's Numerical Approach

$$\boldsymbol{v}(t_{i+1}) = \boldsymbol{v}(t_i) + \left[\boldsymbol{g} - \frac{\boldsymbol{c}}{\boldsymbol{m}}\boldsymbol{v}(t_i)\right](t_{i+1} - t_i)$$

