Lecture 6

Numerical Techniques: Differential Equations - Initial Value Problems

Projectile with Air Resistance, Planetary Motion

IKC-MH.55 *Scientific Computing with Python* at November 17, 2023

Dr. Cem Özdoğan
Engineering Sciences Department
İzmir Kâtip Çelebi University

Contents

Differential Equations I

- Most **problems in the real world** are modeled with **differential equations** because it is easier to see the relationship in terms of a derivative.
- e.g. Newton's Law: F=Ma, $d^2s/dt^2 = a = F/M$ (constant acceleration). **2^{nd} order ordinary differential equation.**
  - It is **ordinary** since it does not involve partial differentials.
  - **Second order** since the order of the derivative is two.
  - The solution to this equation is a function, $s(t) = (1/2)at^2 + v_0 t + s_0$.
  - Two arbitrary constants, v_0 and s_0, the initial values for the velocity and position.
  - The equation for s(t) allows the computation of a numerical value for s, the position of the object, at any value for time, the independent variable, t.
- e.g. Harmonic oscillator problem in mechanics,
- e.g. One-dimensional Schrödinger equation in quantum mechanics,
- e.g. One-dimensional Laplace equation in electromagnetic theory, etc.

Differential Equations II

- Analytical solutions of these equations are often non-existent or very complicated.
- Numerical solutions are the remedy. In terms of solution technique, we can divide differential equations into three groups:

1 Initial Value Problems:

In time-dependent problems, the initial state at time t=0 is given and a solution is searched for later t values. For example, in the following quadratic equation

$$\frac{d^2y}{dt^2} = f(y, y', t)$$

two initial conditions must be given at t=0, namely $y(0)$ and $y'(0)$ values. (Nth order DE $\rightarrow$ N initial conditions).

2 Boundary Value Problems.

3 Eigenvalue (characteristic-value) Problems.

Projectile Motion with Air Resistance I

- In addition to a vertical gravitational force on a 2D projectile motion, there is also a friction force to a certain extent due to air resistance.
- This frictional force is usually in the opposite direction to velocity and is proportional to the square of the velocity: $\vec{F}_r = -kv\vec{v}$ (Drag force, $F_D = -(1/2)c\rho A v^2 \vec{v}/|\vec{v}|$ here, c is the drag coefficient, ρ the air density, and A the projectile's cross-sectional area).

If we write Newton's 2^{nd} law as a vector in two dimensions,

$$
\begin{aligned}
m\vec{a} &= \vec{F}_{net} \\
m\frac{d^2\vec{r}}{dt^2} &= m\vec{g} - kv\vec{v}
\end{aligned}
$$

- and component wise (where $k/m = \gamma$):

$$
\frac{d^2x}{dt^2} = -\gamma\left(\sqrt{v_x^2 + v_y^2}\right)v_x \quad \& \quad \frac{dx}{dt} = v_x
$$

$$
\frac{d^2y}{dt^2} = -g - \gamma\left(\sqrt{v_x^2 + v_y^2}\right)v_y \quad \& \quad \frac{dy}{dt} = v_y
$$

- **Now, we have a set of equations.**

Planetary Motion I

- In the previous projectile motion example, we used the gravitational force with the expression $F = mg$ and gravitational acceleration as being constant near the Earth's surface.
- However, the gravitational force between masses is most generally given by Newton's law of universal gravitation:

$$F = G\frac{m_1 m_2}{r^2}$$

  Here, $G = 6.6743 \times 10^{-11}\ m^3 kg^{-1} s^{-2}$ is called the universal gravitational constant. The force is attractive and along the direction connecting the two masses.
- This expression should be used when studying the motion of planets and moons.
- Let's study the motion of a planet (mass m)moving under the gravitational force of the Sun (mass M). If we take the sun at the origin, the vector expression of the force acting on the planet would be:

$$\vec{F} = -G\frac{Mm}{r^3}\vec{r}$$

Planetary Motion II

- Since the orbit of the planet will be at a plane (2D), the position vector $\vec{r}$ and accordingly the acceleration vector $\vec{a}$ would have two components as:

$$\vec{r} = x\hat{i} + y\hat{j}$$
$$\vec{a} = \frac{d^2x}{dt^2}\hat{i} + \frac{d^2y}{dt^2}\hat{j}$$

- Newton's 2^{nd} law as $\vec{a} = \vec{F}/m$ and also velocity expressions for the x- and y-components:

$$\frac{d^2x}{dt^2} = -G\frac{M}{r^3}x \quad \& \quad \frac{dx}{dt} = v_x$$
$$\frac{d^2y}{dt^2} = -G\frac{M}{r^3}y \quad \& \quad \frac{dy}{dt} = v_y$$

- **Now, we have a set of equations.**

Euler Method I

- In an **initial-value problem**, the numerical solution **begins at the initial point and marches from there** to increasing values for the independent variable.
- **The Euler method.** Describes a method that is easy to use but is not very precise unless the step size, the intervals for the projection of the solution, is very small.
- Consider the following first-order differential equation:

$$\frac{dy}{dx} = y'(x) = f(x, y) \quad \& \quad y(x_0) = y_0 \qquad (1)$$

- Here x is the variable, y(x) and f(x,y) are real functions, and the initial condition y_0 is a real number.
- From the solution of this equation, we get $y_1, y_2, \ldots, y_n$ values for the function at the points $x_1, x_2, \ldots, x_n$ with equal step lengths h.
- **Equations of higher order are solved by converting them to a system of linear equations.**

Euler Method II

- The expression given by Equation 1 is written as the forward-difference approximation at a point x_i by Euler's method.

$$\frac{y_{i+1} - y_i}{h} + O(h) = f(x_i, y_i)$$

- If we solve this expression for y_{i+1}, we get the Euler method formula:

$$\boxed{y_{i+1} = y_i + hf(x_i, y_i) + O(h^2)}$$

- This expression shows that the error in one step of Euler method is $O(h^2)$. But, this error is just the local error. Over many steps, the global error becomes $O(h)$ (as $NO(h^2) \approx O(h)$ for N steps).
- *The method is easy to program when we know the formula for $y'(x)(\equiv f(x_i, y_i))$ and a starting value, $y_0 = y(x_0)$.*

Euler Method III

- Let's see the application of this method on an example. Given differential equation,

$$\frac{dy}{dx} = x + y$$

- The analytical solution of this equation is given as $y(x) = 2e^x - x - 1$. Initial condition: $y(x = 0) = 1$

Step x	Euler y	Exact y	Euler-Exact Error	SciPy y
0.00	1.0000000000000000	1.0000000000000000	0.0000000000000000	1.0000000000000000
0.10	1.1000000000000001	1.1103418361512953	0.0103418361512952	1.1103418365038888
0.20	1.2200000000000002	1.2428055163203395	0.0228055163203393	1.2428055171581294
0.30	1.3620000000000001	1.3997176151520065	0.0377176151520064	1.3997176170100418
0.40	1.5282000000000000	1.5836493952825408	0.0554493952825408	1.5836493990278593
0.50	1.7210200000000000	1.7974425414002564	0.0764225414002564	1.7974425476900568
0.60	1.9431220000000000	2.0442376007810177	0.1011156007810177	2.0442376098866673
0.70	2.1974342000000000	2.3275054149409531	0.1300712149409531	2.3275054266863835
0.80	2.4871776200000002	2.6510818569849350	0.1639042369849348	2.6510818708124289
0.90	2.8158953820000003	3.0192062223138993	0.2033108403138990	3.0192062374603896
1.00	3.1874849202000002	3.4365636569180902	0.2490787367180900	3.4365636726612259

Table: Solution of the differential equation $dy/dx = x + y$ in the interval [0, 1] by Euler method.

(**Example py-file:** myeuler.py)

Euler Method IV

As can be seen from the table, the margin of error is large in the Euler method.

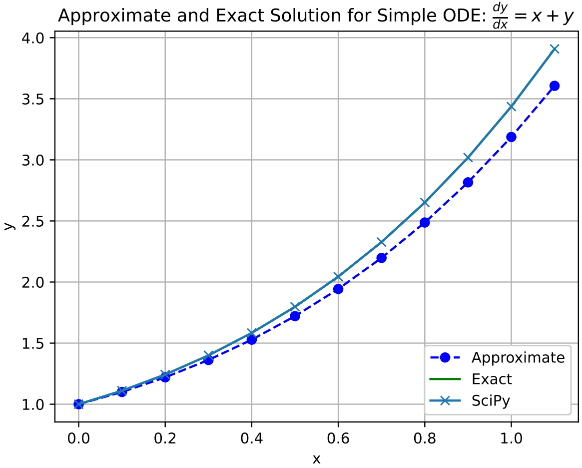
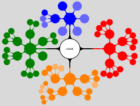


Figure: Solution of the differential equation $dy/dx = x + y$ in the interval [0, 1] by Euler method.

Runge-Kutta Method I

- Simple Euler method comes from using just one term from the Taylor series for y(x) expanded about $x = x_0$.
- What if we use more terms of the Taylor series? Runge and Kutta, developed algorithms from using more than two terms of the series.
- In the Euler method, the increment is directly from x_i to x_{i+1}.
- Second-order Runge-Kutta methods are obtained by using a weighted average of two increments to $y(x_0)$, k_1 and k_2.
- Let's take a "trial step" in the middle and then increment to x_{i+1} by using these middle x- and y-values. Two quantities are defined here as k_1 and k_2,

$$\begin{aligned} k_1 &= hf(x_i, y_i) \\ k_2 &= hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1) \end{aligned}$$



Runge-Kutta Method II

- The parameter;
  - k_1 is for the calculation at x_i, y_i,
  - k_2 is for a half-step away $(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1)$ calculation.
- Accordingly, the 2^{nd} order Runge-Kutta formula becomes:

$$\boxed{y_{i+1} = y_i + k_2 + O(h^3)}$$

- In the Runge-Kutta method, the margin of error in one step is $O(h^3)$ and is $O(h^2)$ in the entire interval.
- It works better than the Euler method, but it comes at a **cost**: f(x, y) will be **calculated twice** at each step.
- This "trial step" technique can be taken even further. Fourth-order Runge-Kutta (RK4) methods are most widely used and are derived in similar fashion.

Runge-Kutta Method III

$$
\begin{aligned}
k_1 &= f(x_i, y_i) \\
k_2 &= f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1) \\
k_3 &= f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_2) \\
k_4 &= f(x_i + h, y_i + hk_3) \\
y_{i+1} &= y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(h^5)
\end{aligned}
$$

- The local error term for the fourth-order Runge-Kutta method is $O(h^5)$; the global error would be $O(h^4)$.
- It is computationally more efficient than the (modified) Euler method because the steps can be manyfold larger for the same accuracy.
- **However, four evaluations of the function are required per step rather than two.**

Runge-Kutta Method IV

- Let's apply the RK4 method on the previous example. Given differential equation,

$$\frac{dy}{dx} = x + y$$

- The analytical solution of this equation is given as $y(x) = 2e^x - x - 1$. Initial condition: $y(x = 0) = 1$

Step x	RK4 y	Exact y	RK4-Exact Error	SciPy y
0.00	1.0000000000000000	1.0000000000000000	0.0000000000000000	1.0000000000000000
0.10	1.1103416666666668	1.1103418361512953	0.0000001694846286	1.1103418365038888
0.20	1.2428051417013890	1.2428055163203395	0.0000003746189505	1.2428055171581294
0.30	1.3997169941250756	1.3997176151520065	0.0000006210269310	1.3997176170100418
0.40	1.5836484801613715	1.5836493952825408	0.0000009151211693	1.5836493990278593
0.50	1.7974412771936765	1.7974425414002564	0.0000012642065799	1.7974425476900568
0.60	2.0442359241838663	2.0442376007810177	0.0000016765971513	2.0442376098866673
0.70	2.3275032531935538	2.3275054149409531	0.0000021617473993	2.3275054266863835
0.80	2.6510791265846310	2.6510818569849350	0.0000027304003041	2.6510818708124289
0.90	3.0192028275601421	3.0192062223138993	0.0000033947537572	3.0192062374603896
1.00	3.4365594882703321	3.4365636569180902	0.0000041686477581	3.4365636726612259

Table: Solution of the differential equation $dy/dx = x + y$ in the interval [0, 1] by 4th order Runge-Kutta method.

(**Example py-file:** myrungekutta.py)

Runge-Kutta Method V

As can be seen from the Table, much more sensitive results are obtained compared to the Euler method.

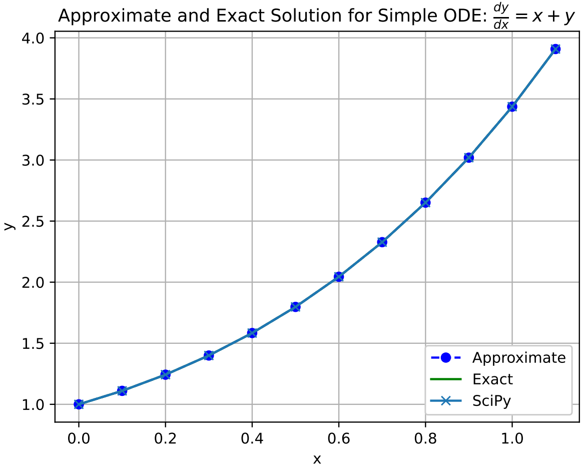


Figure: Solution of the differential equation $dy/dx = x + y$ in the interval [0, 1] by Euler method.

2^{nd} Degree Equations & Linear Systems I

- **Any second-order or higher-order differential equation can be converted into a system of first-order (linear) equations.** For example,

$$\frac{d^2y}{dx^2} + A(x)\frac{dy}{dx} + B(x)y(x) = 0$$

- Let's define two new functions for the equation, $y_1(x)$ and $y_2(x)$:

$$y_1(x) = y(x) \quad \& \quad y_2(x) = \frac{dy}{dx}$$

- With this transformation, instead of one 2^{nd} order equation, two 1^{st} order equations are formed:

$$\begin{aligned} (1)\ \frac{dy_1}{dx} &= y_2(x) \\ (2)\ \frac{dy_2}{dx} &= -A(x)y_2 - B(x)y_1 \end{aligned}$$

2^{nd} Degree Equations & Linear Systems II

- All we need to do to solve higher-order equations, even a **system** of higher-order initial-value problems, is to reduce them to a system of first-order equations.
- Such as: **One M-order equation → a system with M first-order equations.**
- Let's take the most general system of differential equations with M unknowns:

$$
\begin{aligned}
\frac{dy_1}{dx} &= f_1(x, y_1, \ldots, y_M) \quad \& \quad y_1(0) = y_{10} \\
&\vdots \qquad\qquad\qquad\qquad\qquad \vdots \\
\frac{dy_M}{dx} &= f_M(x, y_1, \ldots, y_M) \quad \& \quad y_M(0) = y_{M0}
\end{aligned}
\tag{2}
$$

- The next step for solving is to apply the methods (such as; Euler, Runge-Kutta) for the 1^{st}t order differential equation to these linear system.

Projectile Motion with Air Resistance II

We had a set of equations. Two second degree and two first degree differential equations with two unknowns.

$$(3)\quad \frac{d^2x}{dt^2} = -\gamma\left(\sqrt{v_x^2+v_y^2}\right)v_x \quad \& \quad (1)\ \frac{dx}{dt} = v_x$$

$$(4)\quad \frac{d^2y}{dt^2} = -g-\gamma\left(\sqrt{v_x^2+v_y^2}\right)v_y \quad \& \quad (2)\ \frac{dy}{dt} = v_y$$

- To solve these two 2^{nd} degree equations (plus two 1^{st} degree equations) given above, we first convert them to a system of 4 1^{st} degree (linear) equations.
- To this end, let's define the four unknowns as follows:

- $x \to y_1$
- $y \to y_2$
- $v_x \to y_3$
- $v_y \to y_4$

Projectile Motion with Air Resistance III

- Accordingly, the above 2^{nd} degree system is written as:

$$
\begin{aligned}
(1)\ \frac{dy_1}{dt} &= y_3 \\
(2)\ \frac{dy_2}{dt} &= y_4 \\
(3)\ \frac{dy_3}{dt} &= -\gamma\left(\sqrt{y_3^2+y_4^2}\right)y_3 \\
(4)\ \frac{dy_4}{dt} &= -g-\gamma\left(\sqrt{y_3^2+y_4^2}\right)y_4
\end{aligned}
$$

- When $\gamma = 0$ in this system of equations, we obtain our usual parabolic curve $y = (v_{0y}/v_{0x})x - (g/2v_{0x}^2)x^2$.

Projectile Motion with Air Resistance IV

To calculate the effect of air friction, let's take the initial conditions ($t = 0$) and constants (g & γ):

$$x_0 = y_1(t=0) = 0 \quad \& \quad y_0 = y_2(t=0) = 0$$
$$v_{0x} = y_3(t=0) = 6.0 \quad \& \quad v_{0y} = y_4(t=0) = 8.0$$
$$g = 10.0 \quad \& \quad \gamma = 0.01$$

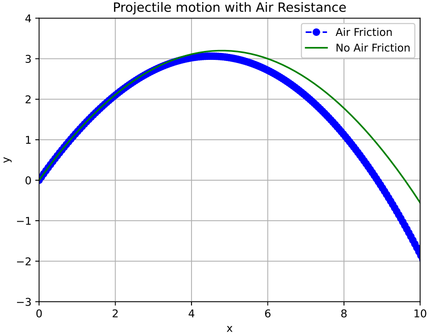


Figure: Numerical solution of projectile motion with and without air friction. (**Example py-file:** airfriction.py)

Planetary Motion III

We had a set of equations. Two second degree and two first degree differential equations with two unknowns.

$$(3)\ \frac{d^2x}{dt^2} = -G\frac{M}{r^3}x \quad \& \quad (1)\ \frac{dx}{dt} = v_x$$

$$(4)\ \frac{d^2y}{dt^2} = -G\frac{M}{r^3}y \quad \& \quad (2)\ \frac{dy}{dt} = v_y$$

- To solve these two 2^{nd} degree equations (plus two 1^{st} degree equations) given above, we first convert them to a system of 4 1^{st} degree (linear) equations.
- To this end, let's define the four unknowns as follows:

- $x \rightarrow y_1$
- $y \rightarrow y_2$
- $v_x \rightarrow y_3$
- $v_y \rightarrow y_4$

Planetary Motion IV

- Accordingly, the above 2^{nd} degree system is written as:

$$
\begin{aligned}
(1)\ \frac{dy_1}{dt} &= y_3 \\
(2)\ \frac{dy_2}{dt} &= y_4 \\
(3)\ \frac{dy_3}{dt} &= -\frac{GM}{[y_1^2 + y_2^2]^{3/2}} y_1 \\
(4)\ \frac{dy_4}{dt} &= -\frac{GM}{[y_1^2 + y_2^2]^{3/2}} y_2
\end{aligned}
\qquad (3)
$$

- For the motion of the planets, we use the astronomical unit system. The Earth-Sun average distance would be in units of astronomical length: 1 $au \approx 1.5 \times 10^{11}$ m. The time taken for the Earth to go around the Sun once is 1 year (y) as the unit of time.
- Calculated in these units, the product of GM,

$$GM \approx 40(au)^3/y^2$$

Planetary Motion V

- To calculate the planetary motion, let's take the initial conditions at time t=0 in terms of four unknowns:

$$x_0 = y_1(t=0) = 1.0\ au \quad \& \quad y_0 = y_2(t=0) = 0$$
$$v_{0x} = y_3(t=0) = 0.0 \quad \& \quad v_{0y} = y_4(t=0) = 6.0\ au/y$$

- Then, also take $v_{0y} = y_4(t=0) = 8.0\ au/y$.

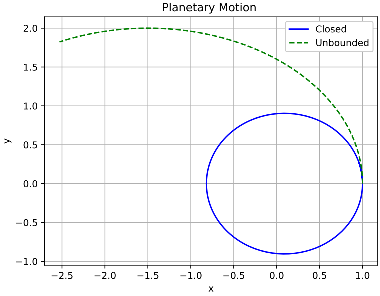


Figure: Numerical solution of planetary motion. There can be closed orbits (ellipse), or solutions going to infinity (unbounded, hyperbola) for different velocities. (**Example py-file:** planetarymotion.py)