NUMERICAL METHODS FOR SOLVING THE SYSTEM OF DIFFERENTIAL EQUATIONS

Thesis submitted in partial fulfillment of the requirement for
The award of the degree of
Masters of Science
in

Mathematics and Computing

Submitted by:

Ripanjot Kaur

Roll.no. 301603022

Under the guidance of

Dr. Paramjeet Singh

Assistant Professor & Supervisor

SoM, TIET

Patiala

July 2018

School of Mathematics

Thapar Institute of Engineering and Technology

Patiala 147004

India

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled "**Numerical methods for solving the system of differential equations**" submitted for the MSc dissertation in the School of Mathematics, Thapar Institute of Engineering and Technology, Patiala, India, is a review work. This study is carried out by me under the supervision of **Dr. Paramjeet Singh**.

The matter presented in this thesis has neither published nor sent for publication anywhere.

Ripanjot Kaur

(Ripanjot Kaur)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Paramjeet Singh)
Assistant Professor & Supervisor
SoM, Thapar Institute of Engineering and Technology
Patiala

ACKNOWLEDGEMENT

First of all, I would like to thank the almighty for granting perseverance. I would like to pass my gratitude to **Dr. Paramjeet Singh**, Assistant Professor & Supervisor, SoM, Thapar Institute of Engineering and Technology, Patiala, for his constructive suggestions, patience guidance and support throughout this work. I am truely fortunate to have oppurtunity to work under him.

I would like to express my deep sense of gratitude to **Dr. Satish Kumar Sharma**, Head School of Mathematics, Thapar Institute of Engineering and Technology, Patiala, for providing the necessary facilities needed to carry out the results. I would like to thank the faculty members and administrative staff of School of Mathematics, Thapar Institute of Engineering and Technology, Patiala, who have contributed directly or indirectly to do this work.

I would like to thank my friends for providing their support in every way in Thapar Institute of Engineering and Technology, Patiala. Last but not the least I would like to thank my parents for their continuous support which inspired me and encouraged me, without them I would not have been able to complete this work.

ABSTRACT

The numerical methods are very useful in estimating the approximate solutions of differential equation's problems. Among such methods three methods namely **Euler's method**, **Modified Euler's method** and **Runge-Kutta method** are studied and are applied to the real life problems to have approximated solutions.

Chapter-wise account of this thesis is as follows :

Chapter 1 is preliminary in nature. This consists of the basic terminology regarding differential equations and it's system along with real life examples. Secondly it deals with the origin and applications of differential equations.

Chapter 2 deals with the study of numerical methods and their implementation in the problems of differential equations and system of differential equations. It consists of illustration of numerical methods along with examples.

Chapter 3 consists of real life problems where differential equations play an important role and for those problems the numerical methods studied in chapter 2 are applied. Two problems naming **Pricing policy of goods** and **Rössler system** which involves higher order differential equation and system of differential equations respectively are taken. Numerical methods are applied to have approximate solutions. It also consists of matlab implementation for these problems.

Contents

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

Numerical analysis is the branch of mathematics and computer science which deals with the creation, analyzation and implementation of algorithms for solving the problems numerically of continuous mathematics. Such problems emerges generally from real-world applications and involve the variables which vary continuously.
Numerical methods are chosen to approximate solutions of when exact solutions are difficult or can not be evaluated using algebraic methods. These methods carries successive approximations converging to exact solution of an equation or of system. Aim of this paper is to examine several numerical methods which are can be put forward to solve the system of differential equations. Differential equations have it's role in every field but all differential equations are not solvable so numerical methods are helpful in finding some approximations to it's solutions.
Higher ordered differential equations are convertible to the system of first order equations using several transformations and solving that system is equivalent of solving that higher order equation. Three methods namely **Euler's method**, **Euler's mid-point method** and **Runge-Kutta method** (second order and fourth order) are studied and used to solve such system of equations. Real life examples are taken where differential equations are involved and are converted into system of differential equations and their approximate solutions are evaluated. The concept of system of differential equations and these numerical methods can be viewed in the real life problem of predator-prey model and Van der Pol oscillator which are then solved numerically. Two problems are taken one from economics field naming **Pricing policy for production of goods** in which second order differential equation is involved which is further converted into the system of two first-order differential equations and is solved numerically and the results are compared with analytic solution. Second one is the **Rössler system** which is three dimensional autonomous system which is of form Lorenz equations. It is taken from fluid dynamics. Numerical study is applied on this system to know the behavior of the system.

1.2 ORIGIN AND APPLICATION OF DIFFERENTIAL EQUATIONS

It is interesting to know Where and How such equations actually originated. Differential equations happen to take place in connection with various problems that are encountered in the several branches of science and engineering like:

- The problem dealing with the motion of rockets, projectiles, satellites or planets.

$$\begin{aligned} \frac{dx}{dt} &= v_x, \\ \frac{dy}{dt} &= v_y. \end{aligned}$$

  Where v represents the velocity.

- The problem dealing with determination of charge or current in an electric circuit.

$$\frac{dQ}{dt} = \frac{V}{R} - \frac{Q}{RC}.$$

  Where Q is charge, V is potential difference, C is capacitance and R is resistance.

- The problem involving heat conduction in a rod or in a slab uses these equations.

$$\alpha \Delta T + \frac{q_v}{\rho c_p} = \frac{\partial T}{\partial t}.$$

  Where T is temperature, q is heat energy, ρ is mass density, c_p is specific heat.

- The problem of determining wire's vibrations or an oscillatory motion.

$$a = \frac{d^2x}{dt^2} = -\omega^2 x.$$

  Where x is displacement, a is acceleration and ω is constant.

- The study of decomposition rate of radioactive substances or growth rate of a population.

$$\frac{dP}{dt} = kP(1 - \frac{P}{M}).$$

  Where P is population, t is time, k, M are constants.

- The study involving reactions of various chemicals.

$$\frac{d[A]}{dt} = -k[A]^2.$$

  Where A is chemical and k is constant.

Differential equations follows from the mathematical formulation of such problems. In the situations under careful inspection in each of above cases the objects involved follows various scientific laws. These laws consists of some rates of change of one or more quantity with respect to another quantity which are represented mathematically as derivatives. In mathematical formulation, these rates of change are referred as derivatives and the scientific laws behaves as mathematical equations involving derivatives i.e. Differential equations.

1.3 PRELIMINARIES

1.3.1 Differential equation

An equation having derivatives of one or more dependent variables with respect to one or more independent variables is known as differential equation. For instance:

$$\begin{aligned} \frac{d^2z}{dz^2} + xz\left(\frac{dz}{dx}\right)^2 &= 0, \\ \frac{d^3x}{dt^3} + 5\frac{d^2x}{dt^2} + 3x &= \sin t. \end{aligned}$$

1.3.2 Order of differential equation

The highest ordered derivative present in a differential equation is called as order of differential equation. For example

$$\frac{d^2y}{dx^2} + xy\left(\frac{dy}{dx}\right)^2 = 0.$$

is of second order as highest derivative involved is second derivative.

1.3.3 Basic existence and uniqueness theorem

Considering the differential equation

$$\frac{dz}{dx} = f(x, z),$$

where f is a continuous function of x and z in some domain D of xz plane and the partial derivative $\dfrac{\partial f}{\partial z}$ is also a continuous function of x and z in D and $(x_{0,0})$ be a point in D then there exist a unique solution ϕ of the differential equation defined in some interval $|x - x_0| \leq h$, where h is sufficiently small, satisfying the condition $\phi(x_0) = z_0$.

1.3.4 System of differential equations

The differential equation's group or set forms a system of differential equation.
For example : System of two first-order differential equations represented in two unknown functions x and y is of form:

$$\begin{aligned} c_1(t)\frac{dx}{dt} + c_2(t)\frac{dy}{dt} + c_3(t)x + c_4(t)y &= F_1(t), \\ d_1(t)\frac{dx}{dt} + d_2(t)\frac{dy}{dt} + d_3(t)x + d_4(t)y &= F_2(t). \end{aligned}$$

A system of N first-order differential equations in N unknowns along with respective initial conditions in vector form is given as

$$z^{'} = f(t, z), \quad z(a) = z_0,$$

where $z = z(t) = (z_1(t), z_2(t), ..., z_N(t))$ is a vector of N unknown scaler functions,
$f(t, z) : \mathbb{R}^{N+1} \to \mathbb{R}^N$ is a vector function of the $N + 1$ variables t and $z = (z_1, z_2, ..., z_N)$ i.e.

$$f(t, z) = (f_1(t, z), f_2(t, z), ..., f_N(t, z)) .$$

While $z_0 = (z_{1,0}, z_{2,0}, ..., z_{N,0})$, is a vector in $\mathbb{R}^{\mathbb{N}}$ of initial values. The notation $z^{'}$ denotes the vector of the components of z with respect to t

$$z^{'} = z^{'}(t) = \left(z_1^{'}(t), z_2^{'}(t), ..., z_N^{'}(t)\right) .$$

1.3.5 Solution of differential equation

By the term solution of the differential equation, it means a vector-valued function $z(t)$ defined and continuously differentiable in an interval $a < t < b$, also satisfying the differential equation in its interval of definition.

1.3.6 Numerical method

A numerical method is a mathematical tool designed to solve numerical problems. The use of a numerical method with an appropriate convergence check in a programming language is called a numerical algorithm.

1.3.7 Convergence of numerical method

We say that a sequence converges if it has a limit.
Let p_n be a sequence that converges to p, where $p_n \neq p$. If constants $\lambda,\ \alpha > 0$ exist such that

$$\lim_{n \to \infty} \frac{\mid p_{n+1} - p \mid}{\mid p_n - p \mid^{\alpha}} = \lambda.$$

Then it is said that p_n converges to p of order α with a constant λ.
A numerical method for solving $z^{'} = f(t, z)$ with $z(0) = z_0$ where $0 \leq t \leq T$, is said to be convergent if

$$\max \mid z_{k+1} - z_k \mid \to 0$$

as $\Delta t \to 0$.

1.4 HIGHER ORDER DIFFERENTIAL EQUATIONS

Many practical real-life problems give rise to the systems of differential equations and sometimes higher order equations are involved in such problems.

1.4.1 System of first order equations

It is evident that the higher order differential equations can be reduced to system of first order equations as well. To illustrate this we take a second order equation :

$$z^{''} = t^2 + \sin(z + z_2),$$

given $z(0) = 1,\ z^{'}(0) = 0$.
Substituting the new function $z_2 = z^{'}$. Observe that $z_2^{'} = z^{''}$. Therefore the above differential equation is now re-written as

$$z_2^{'} = t^2 + \sin(z + z_2), \quad z(0) = 1, z_2 = 0.$$

On renaming z as $z_1 = z$, it is seen that the main equation is now rewritten as the system

$$\begin{aligned} z_1^{'} &= z_2, \quad z_1(0) = 1, \\ z_2^{'} &= t^2 + \sin(z_1 + z_2), \quad z_2 = 0. \end{aligned}$$

In other words,the differential equation of second order is convertible to the system of two first order equations.

In general a qth order equation can be converted to the system of q first order equations. Taking qth order differential equation

$$z^{(q)} = g(t, z, z^{'}, z^{''}, ..., z^{(q-1)}),$$

with initial conditions

$$z(a) = c_0,\ z^{'}(a) = c_1,\ \ldots,\ z^{q-2}(0) = c_{q-2},\ z^{q-1}(0) = c_{q-1}.$$

Which is equivalent to the system of $z_1, z_2, ..., z_q$.

$$\begin{aligned}
z_1^{'} &= z_2, \quad z_1(a) = c_0,\\
z_2^{'} &= z_3, \quad z_2(a) = c_1,\\
&\vdots\\
z_{q-1}^{'} &= z_q, \quad z_{q-1}(a) = c_{p-2},\\
z_q^{'} &= g(t, z_1, z_2, ..., z_{q-1}), \quad z_q(a) = c_{q-1}.
\end{aligned}$$

Where $z_1(t)$ the component solution of system agrees with solution $z(t)$ of differential equation.

1.4.2 System of higher order equations

A set of various differential equations of higher order form a higher ordered equation's system. As each higher order differential equation is written as system of first order equations, in that very way a system of differential equations can always be converted to a system of first order equations. Considering the system of differential equations given as

$$\begin{aligned}
r^{''} &= t + r^{'} + s^{'}, \quad r(0) = 1,\ r^{'}(0) = 2,\\
s^{'''} &= r^{'} s^{''} + r, \quad s(0) = -1,\ s'(0) = 1,\ s^{''}(0) = 2.
\end{aligned}$$

We introduce new functions $r_1 = r,\ r_2 = r^{'},\ s_1 = s,\ s_2 = s^{'}$ and $s_3 = s^{''}$. Then above system can be written as

$$\begin{aligned}
r_1^{'} &= r_2, \quad r_1(0) = 1,\\
r_2^{'} &= t + r_2 + s_2, \quad r_2(0) = 2,\\
s_1^{'} &= s_2, \quad s_1(0) = -1,\\
s_2^{'} &= s_3, \quad s_2(0) = 1,\\
s_3^{'} &= r_2 s_3 + r_1, \quad s_3(0) = 2.
\end{aligned}$$

This illustrates the conversion of a higher order equation's system into a system of first order equations.

Considering the following examples of system of differential equations:

1.4.3 Predator-prey model

There is a real life application of nonlinear system of differential equations in the field of mathematical biology/ ecology. In this there is a representation of relationship of predator-prey in our ecosystem.
A system of predator-prey is a simple ecological model which contains two species : the predators(first species) who feed on the prey(second species). This model is also referred as **Lotka-Volterra model** or **Competition model** (in which one category will depend on other category for their living). The equations of predator-prey are the pair of non-linear first-order differential equations depicting the biological system's dynamics for predator and prey interacting with one another.
The change in populations with time is given according to the pair of equations:

$$\begin{aligned} \frac{dr}{dt} &= Ar - Brf, \\ \frac{df}{dt} &= -Cf + Drf. \end{aligned}$$

In this

- r represents the number of prey (for example : rabbits).
- f represents the number of predators (for example : foxes).
- $\frac{dr}{dt}$ and $\frac{df}{dt}$ represent the rates of growth of two populations with time.
- t represent the time.
- A, B, C, D represents the positive real parameters which describes the interaction between two species:
  A - Growth rate of prey.
  B - Searching efficiency or attack rate.
  C - Predator mortality rate.
  D - Growth rate of predator.

Mathematical Formulation :
We initially begin- in the absence of prey(without food resources) what will happen to the predator's population? The number of predators is supposed to decline exponentially, given as:

$$\frac{df}{dt} = -Cf.$$

This equation carries the product of predator's number(f) and the predator mortality rate ($-C$) to show the rate of decline of the population of predators with respect to time (t).
Nextly in the presence of prey, this decrease is opposed by the birth-rate of predators Drf, which is evaluated by the product of predator's number (f), the prey's number (r) and the growth rate of predators (D). When the number of predator and prey (f and r respectively) increases, the chances of their encounters become more frequent. The equation representing the population dynamics of predators is as follows:

$$\frac{df}{dt} = -Cf + Drf.$$

Now going to the prey's population, initially supposing in the absence of predators, the prey's number would increase exponentially. The following equation shows the increase rate of prey's population with the time where A is the rate of growth of prey's population and r is the prey's number

$$\frac{dr}{dt} = Ar.$$

Further in the presence of predators, the prey's population stops increasing exponentially. The consumption rate (Brf) i.e. the product of attack rate (B), the number of predators (f) and the number of prey (r), is the prey's mortality. The equation representing the population dynamics of prey is as follows:

$$\frac{dr}{dt} = Ar - Brf.$$

The product Df represents the predator's numerical response or the per capita increase as a function of prey's number. The whole term Drf depicts the increase in the predator population is proportional to the product of predator and prey number. The product Bf is called predator's functional response or prey capture rate which is function of prey's number. Here the term Brf depicts that prey population declines due to predation and is proportional to the product of predator and prey numbers. Therefore equations

$$\begin{aligned} \frac{dr}{dt} &= Ar - Brf, \\ \frac{df}{dt} &= -Cf + Drf. \end{aligned}$$

This forms a system describing predator and prey population's dynamics in one another's presence and together these two equations forms the **Lotka Volterra predator-prey model**.

1.4.4 Van der Pol equation

In the branch of physics(dynamics), differential equations have it's role in the Van der Pol oscillator which is a non-conservative oscillator having non-linear damping. It is a model of a non-conservative system, in which the energy is given and drawn from the system in an autonomous way, which further yields to a periodic motion known as a limit cycle. A system producing a periodic signal is known as relaxation oscillator. Examples of relaxation oscillations are the heart beats and the

squeaking of fingernails on a blackboard. One of the relaxation oscillator that can be built with electronic components is the Van der Pol oscillator. It evolves in time according to the second-order differential equation:

$$\frac{d^2z}{dt^2} - \mu(z^2-1)\frac{dz}{dt} + z = 0.$$

It is a second order differential equation which is convertible to a system of first order equations as follows:
Using the following substitution

$$w = \frac{dz}{dt},$$

we get

$$\frac{dw}{dt} = \mu(z^2-1)w - z.$$

Therefore equivalent system of forced van der Pol equation is given by equations:

$$\begin{aligned} z^{'} &= w, \\ w^{'} &= \mu(z^2-1)w - z. \end{aligned}$$

This is an equivalent phase plane system.

1.4.5 Newtonian equations

A mass m moving in a potential force field follows Newtonian equations which are a second order system of form :

$$m\frac{d^2\mathbf{a}}{dt^2} = -\nabla F(\mathbf{a}),$$

in which $\mathbf{a}(t) = (a(t), b(t), c(t))^T$ is mass' position and $F(\mathbf{a}) = F(a, b, c)$ is the potential function. In components,

$$\begin{aligned} m\frac{d^2a}{dt^2} &= -\frac{\partial F}{\partial a}, \\ m\frac{d^2b}{dt^2} &= -\frac{\partial F}{\partial b}, \\ m\frac{d^2c}{dt^2} &= -\frac{\partial F}{\partial c}. \end{aligned}$$

For instance a planet moving in the sun's gravitational field follows the Newtonian system of the gravitational potential

$$F(\mathbf{a}) = -\frac{\alpha}{\parallel \mathbf{a} \parallel} = -\frac{\alpha}{\sqrt{a^2+b^2+c^2}},$$

where α depends on the mass and universal gravitational constant. Thus the motion of mass in such gravitational force field follows the solution to the second order Newtonian system i.e.

$$m\frac{d^2\mathbf{a}}{dt^2} = -\nabla F(\mathbf{a}) = -\frac{\alpha \mathbf{a}}{\| \mathbf{a} \|^3} = -\frac{\alpha}{(a^2+b^2+c^2)^{\frac{3}{2}}}\begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$

The motion of a charged particle in a Coulomb electric force field obeys the same system of ordinary differential equations where sign of α is positive for attracting opposite charges and negative for repelling like charges.

Converting the Newton second order equations into a first order system,
set $\mathbf{b} = \dot{\mathbf{a}}$, to be the mass' velocity vector, with components

$$\begin{aligned} p &= \frac{da}{dt}, \\ q &= \frac{db}{dt}, \\ r &= \frac{dc}{dt}. \end{aligned}$$

So we get system of first order equations as :

$$\begin{aligned} \frac{da}{dt} &= p, \\ \frac{db}{dt} &= q, \\ \frac{dc}{dt} &= r, \\ \frac{dp}{dt} &= -\frac{1}{m}\frac{\partial F}{\partial a}(a,b,c), \\ \frac{dq}{dt} &= -\frac{1}{m}\frac{\partial F}{\partial b}(a,b,c), \\ \frac{dr}{dt} &= -\frac{1}{m}\frac{\partial F}{\partial c}(a,b,c). \end{aligned}$$

Chapter 2

NUMERICAL METHODS

2.1 NUMERICAL METHODS FOR SOLVING SYSTEM OF FIRST ORDER EQUATIONS

Numerical methods for ordinary differential equations and it's system are the techniques which are employed to have numerical approximations to the solutions. Analytical solutions are the exact solutions used to study the nature of system with varying characteristics. But there are very few practical systems leading to analytical solutions and these are of limited use. For this reason we carry out numerical approach to have close answer to the practical result. Numerical methods have the capability of handling large systems of equations having different degrees of those nonlinearities common in engineering practice. Any complicated physical geometries not easily solvable by analytic approach are handled by these numerical methods.
Following are the numerical methods used to solve the system of first order differential equations are discussed:

2.1.1 Euler's method

It is one of the simplest and easiest numerical technique named after Leonhard Euler, carried out to approximate the solution of system of first order equations.

$$z^{'} = f(t, z), \quad z(a) = z_0.$$

Beginning with initial time t_0 we represent successive time points(or sample times) $t_0 < t_1 < t_2 < t_3 < ...$, continuing until we are at desired final time $t_n = t^*$. These points should be chosen in such a way that these are fairly closely spaced so that the step size is

$$h = t_{k+1} - t_k > 0,$$

is independent of k and is relatively small. For a uniform step size, k^{th} time point is at $t_k = t_0 + kh$. Numerical algorithm will iteratively compute the approximations $z_k \approx z(t_k)$ for $k = 0, 1, 2, 3, ...$
The simplest approximation of $z(t)$ is given by it's tangent line or by Taylor polynomial of first order. Thus, near the time point t_k

$$z(t) \approx z(t_k) + (t - t_k)\frac{dz}{dt}(t_k) = z(t_k) + (t - t_k)f(t_k, z(t_k)).$$

In particular, the solution's approximate value at the subsequent time point is

$$z(t_{k+1}) \approx z(t_k) + (t_{k+1} - t_k)f(t_k, z(t_k)).$$

This is a basis of Euler's method.
Since z_k is the value of $z(t_k)$ at ongoing time point, therefore replacing $z(t_k)$ by its approximation z_k and putting value of h in preceding formula , we get

$$z_{k+1} = z_k + hf(t_k, z_k).$$

ALGORITHM (Euler's method).Begin with $z' = f(t, z)$ given together with the initial condition $z(a) = z_0$, taking solution interval $[a, b]$, and the number of steps involved are n. If the following algorithm is performed

$$\begin{aligned} \text{set } h &= (b - a)/n; \\ t_0 &= a; \\ z_{k+1} &= z_k + hf(t_k, z_k); \\ t_{k+1} &= a + (k + 1)h; \end{aligned}$$

for

$$k = 0, 1, ..., n - 1,$$

the value z_k will be an approximated value of solution $z(t_k)$ of the differential equation.
For example consider the system with initial conditions

$$\begin{aligned} u' &= uv + \cos w, \quad u(0) = u_0, \\ v' &= 2 - t^2 + w^2 v, \quad v(0) = v_0, \\ w' &= \sin t - u + v, \quad w(0) = w_0, \end{aligned}$$

where $\mathbf{u} = (u, v, w)$,
$\mathbf{u}_0 = (u_0, v_0, w_0)$,
and $\mathbf{f}(t, \mathbf{u}) = (f_1(t, \mathbf{u}), f_2(t, \mathbf{u}), f_3(t, \mathbf{u}))$.

Therefore

$$\begin{aligned}\mathbf{f}(t,\mathbf{u}) &= (f_1(t,u,v,w), f_2(t,u,v,w), f_3(t,u,v,w)) \\ &= \left(uv+\cos w, 2-t^2+w^2v, \sin t-u+v\right).\end{aligned}$$

Euler's method is applied as

$$\mathbf{u}_{k+1} = \mathbf{u}_k + h\mathbf{f}(t_k, \mathbf{u}_k), \quad k = 0, 1, 2, ..., n-1.$$

Writting three components explicitly, this becomes:

$$\begin{aligned}u^{k+1} &= u^k + hf_1(t_k, u^k, v^k, w^k) = u^k + h(u^kv^k + \cos w^k), \\ v^{k+1} &= v^k + hf_2(t_k, u^k, v^k, w^k) = v^k + h(2 - t^2 + (w^k)^2v^k), \\ w^{k+1} &= w^k + hf_3(t_k, u^k, v^k, w^k) = w^k + h(\sin t - u^k + v^k).\end{aligned}$$

For $k = 0, 1, ...n-1$, with initial values (a, u^0, v^0, w^0) given by initial condition i.e. for $k = 0$

$$\begin{aligned}u^1 &= u^0 + hf_1(a, u^0, v^0, w^0) = u^0 + h(u^0v^0 + \cos w^0), \\ v^1 &= v^0 + hf_2(a, u^0, v^0, w^0) = v^0 + h(2 - a^2 + (w^0)^2v^0), \\ w^1 &= w^0 + hf_3(a, u^0, v^0, w^0) = z^0 + h(\sin a - u^0 + v^0).\end{aligned}$$

For $k = 1$ use these values of u^1, v^1, w^1 and so on for k upto $n-1$.

2.1.2 Improved Euler's method

Many problems of the Euler method requires very small step size for the producing necessary accurate results, much efforts has been emphasized in creating more efficient methods.
The improved Euler method is also known as the **Heun formula** or **Euler mid-point method**. This method gives more accurate approximation than Euler rule and gives explicit formula for computing z_{n+1}. The basic idea is to correct some error of original Euler method.

One of the way of upgrading Euler's method is hence the estimating each line segment's slope in better way. This is carried out via a two-step technique which targets at the estimating the slope at the midpoint between the two solution points. In continuing from (t_k, z_k) to (t_{k+1}, z_{k+1}) we need to have tangent of solution curve at the midpoint $t_k + \frac{h}{2}$.
The solution is stepped forward from (t_k, z_k) to $(t_k + h, z_{k+1})$ in two steps: Firstly solution's approximation is computed at the midpoint $t_k + \frac{h}{2}$ by using Euler's method with step length $\frac{h}{2}$,

$$z_{k+1/2} = z_k + \frac{h}{2}f(t_k, z_k).$$

Further solution is stepped forward to t_{k+1} via straight line from (t_k, z_k) with slope given as $f(t_k + h/2, z_{k+1/2})$,

$$z_{k+1} = z_k + hf(t_k + h/2, z_{k+1/2}).$$

ALGORITHM (Euler's midpoint method). Begin with $z' = f(t, z)$ given with the initial condition $z(a) = z_0$, the solution interval taken as $[a, b]$ and the number of steps be n. Euler's midpoint method is given as

$$\begin{aligned} \text{set } h &= (b-a)/n; \\ t_0 &= a; \\ z_{k+1/2} &= z_k + hf(t_k, z_k)/2; \\ z_{k+1} &= z_k + hf(t_k + h/2, z_{k+1/2}); \\ t_{k+1} &= a + (k+1)h; \end{aligned}$$

for

$$k = 0, 1, ..., n-1.$$

After performing these steps, the value z_k is an approximated value of the solution $z(t_k)$ of the differential equation at t_k .
For example consider the system with initial conditions

$$\begin{aligned} u' &= uv + \cos w, \quad u(0) = u_0 \\ v' &= 2 - t^2 + w^2 v, \quad v(0) = v_0 \\ w' &= \sin t - u + v, \quad w(0) = w_0, \end{aligned}$$

where $\mathbf{u} = (u, v, w)$,
$\mathbf{u}_0 = (u_0, v_0, w_0)$,
and $\mathbf{f}(t, \mathbf{u}) = (f_1(t, \mathbf{u}), f_2(t, \mathbf{u}), f_3(t, \mathbf{u}))$.
Therefore

$$\begin{aligned} \mathbf{f}(t, \mathbf{u}) &= (f_1(t, u, v, w), f_2(t, u, v, w), f_3(t, u, v, w)) \\ &= \left(uv + \cos w, 2 - t^2 + w^2 v, \sin t - u + v\right). \end{aligned}$$

Method is applied as

$$\mathbf{u}_{k+1/2} = \mathbf{u}_k + \frac{h}{2}\mathbf{f}(t_k, \mathbf{u}_k), \quad k = 0, 1, 2, ..., n-1,$$

then

$$\mathbf{u}_{k+1} = \mathbf{u}_k + h\mathbf{f}(t_k + h/2, \mathbf{u}_{k+1/2}).$$

Writting three components explicitly, this becomes:

$$\begin{aligned} u^{k+1}_{k+1/2} &= u^k + \frac{h}{2} f_1(t_k, u^k, v^k, w^k) = u^k + \frac{h}{2}(u^k v^k + \cos w^k), \\ v^{k+1}_{k+1/2} &= v^k + \frac{h}{2} f_2(t_k, u^k, v^k, w^k) = v^k + \frac{h}{2}(2 - t^2 + (w^k)^2 v^k), \\ w^{k+1}_{k+1/2} &= w^k + \frac{h}{2} f_3(t_k, u^k, v^k, w^k) = w^k + \frac{h}{2}(\sin t - u^k + v^k). \end{aligned}$$

$$\begin{aligned} u^{k+1}_{k+1} &= u^k + h f_1(t_k + h/2, u^{k+1}_{k+1/2}) = u^k + h f_1(t_k + h/2, u^k + \frac{h}{2}(u^k v^k + \cos w^k)), \\ v^{k+1}_{k+1} &= v^k + h f_2(t_k + h/2, v^{k+1}_{k+1/2}) = v^k + h f_2(t_k + h/2, v^k + \frac{h}{2}(2 - t^2 + (w^k)^2 v^k)), \\ w^{k+1}_{k+1} &= w^k + h f_3(t_k + h/2, z^{k+1}_{k+1/2}) = w^k + h f_3(t_k + h/2, w^k + \frac{h}{2}(\sin t - u^k + v^k)). \end{aligned}$$

For $k = 0, 1, ...n-1$, with the starting values (a, u^0, v^0, w^0) given by initial condition i.e. for $k = 0$

$$\begin{aligned} u^1_{1/2} &= u^0 + \frac{h}{2} f_1(a, u^0, v^0, w^0) = u^0 + \frac{h}{2}(u^0 v^0 + \cos w^0), \\ v^1_{1/2} &= v^0 + \frac{h}{2} f_2(a, u^0, v^0, w^0) = v^0 + \frac{h}{2}(2 - a^2 + (w^0)^2 v^0), \\ w^1_{1/2} &= w^0 + \frac{h}{2} f_3(a, u^0, v^0, w^0) = w^0 + \frac{h}{2}(\sin a - u^0 + v^0). \end{aligned}$$

$$\begin{aligned} u^1_1 &= u^0 + h f_1(a + h/2, u^1_{1/2}) = u^0 + h f_1(a + h/2, u^0 + \frac{h}{2}(u^0 v^0 + \cos w^0)), \\ v^1_1 &= v^0 + h f_2(a + h/2, v^1_{1/2}) = v^0 + h f_2(a + h/2, v^0 + \frac{h}{2}(2 - a^2 + (w^0)^2 v^0)), \\ w^1_1 &= w^0 + h f_3(a + h/2, w^1_{1/2}) = w^0 + h f_3(a + h/2, w^0 + \frac{h}{2}(\sin a - u^0 + v^0)). \end{aligned}$$

For $k = 1$ use these values of u^1_1, v^1_1, w^1_1 and so on for k upto $n-1$.

2.1.3 Runge-Kutta method

These methods are generalisation of the midpoint Euler method. These were developed by the German mathematicians C. Runge and M. W. Kutta in around 1900. Methods make use of several calculations of f between each and every step in a different manner further yielding in great extent to accuracy. By so far these methods are said to be most popular and powerful numerical methods for integrating the ordinary differential equations.

Aim is to achieve higher extent of accuracy by sacrificing the Euler method's efficiency by re-evaluating $f(t, z)$ at the points intermediating between $(t_n, z(t_n))$ and $(t_{n+1}, z(t_{n+1}))$. In general

R-stage Runge-Kutta family is defined as

$$z_{n+1} = z_n + hf(t_n, z_n, h),$$

where

$$\begin{aligned} f(t,z,h) &= \sum_{r=1}^{R} c_r k_r, \\ k_1 &= f(t,z), \\ k_r &= f(t + ha_r, z + h\sum_{s=1}^{r-1} b_{rs}k_s), \\ a_r &= \sum_{s=1}^{r-1} b_{rs}, \\ r &= 2, ..., R. \end{aligned}$$

One-stage Runge-Kutta method. For R = 1.Resulting one stage Runge-Kutta method is simply Euler's explicit method:

$$z_{n+1} = z_n + hf(t_n, z_n).$$

Two-stage Runge-Kutta method. For R = 2,

$$z_{n+1} = z_n + h(c_1k_1 + c_2k_2),$$

where

$$\begin{aligned} k_1 &= f(t_n, z_n), \\ k_2 &= f(t_n + a_2h, z_n + b_{21}hk_1). \end{aligned}$$

Where c_1, c_2, a_2 and b_{21} parameters are to be determined. For $b_{21} = a_2$, $c_2 = 1/(2a_2)$ and $c_1 = 1 - 1/(2a_2)$, the method is second-order accurate. If $a_2 = 1$ formula for Runge-Kutta method of second order becomes:

$$\begin{aligned} z_{n+1} &= z_n + h(k_1 + k_2)/2, \\ k_1 &= f(t_n, z_n), \\ k_2 &= f(t_n + h, z_n + hk_1). \end{aligned}$$

Fourth order Runge-Kutta method

The formula carries a weighted average of values of $f(t, z)$ at different points in the interval $t_n \leq t \leq t_{n+1}$. It is given as

$$z_{n+1} = z_n + h\frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6},$$

where

$$\begin{aligned} k_1 &= f(t_n, z_n), \\ k_2 &= f(t_n + h/2, z_n + hk_1/2), \\ k_3 &= f(t_n + h/2, z_n + hk_2/2), \\ k_4 &= f(t_n + h, z_n + hk_3). \end{aligned}$$

Here the sum $(k_1 + 2k_2 + 2k_3 + k_4)/6$ is chosen to be average slope. In which k_1 denotes slope at the first end of the interval, k_2 slope at midpoint evaluated using the Euler formula in going from t_n to $t_n + h/2$, k_3 the second approximation to slope at midpoint, and finally k_4 slope at $t_n + h$ evaluated via Euler formula and the slope k_4 goes from t_n to $t_n + h$.

ALGORITHM (Second order Runge-Kutta method). Begin with $z^{'} = f(t, z)$ given along with the initial condition $z(a) = z_0$, solution interval as $[a, b]$ and the number of steps be n. Runge-Kutta of second order method is given by

$$\begin{aligned} \text{set } h &= (b - a)/n; \\ t_0 &= a; \\ k_1 &= f(t_n, z_n); \\ k_2 &= f(t_n + h, z_n + hk_1); \\ z_{n+1} &= z_n + h(k_1 + k_2)/2; \\ t_{k+1} &= a + (k + 1)h; \end{aligned}$$

$$k = 0, 1, ..., n - 1.$$

After these steps the value z_k is an approximated value of the solution $z(t_k)$ of the differential equation at t_k.

ALGORITHM (Fourth order Runge-Kutta method). Begin with $z^{'} = f(t, z)$ given along with the initial condition $z(a) = z_0$, solution interval as $[a, b]$ and the number of steps be n. Runge-Kutta of fourth order method is given by

$$\begin{aligned} \text{set } h &= (b - a)/n; \\ t_0 &= a; \\ k_1 &= f(t_n, z_n); \\ k_2 &= f(t_n + h/2, z_n + hk_1/2); \\ k_3 &= f(t_n + h/2, z_n + hk_2/2); \\ k_4 &= f(t_n + h, z_n + hk_3); \\ z_{n+1} &= z_n + h(k_1 + 2k_2 + 2k_3 + k_4)/6; \\ t_{k+1} &= a + (k + 1)h; \end{aligned}$$

For

$$k = 0, 1, ..., n-1.$$

After these steps the value z_k is an approximated value of solution $z(t_k)$ of the differential equation at t_k.
For example consider the system with initial conditions

$$\begin{aligned} u' &= uv + \cos w, \quad u(0) = u_0, \\ v' &= 2 - t^2 + w^2 v, \quad v(0) = v_0, \\ w' &= \sin t - u + v, \quad w(0) = w_0. \end{aligned}$$

Where $\mathbf{u} = (u, v, w)$,
$\mathbf{u}_0 = (u_0, v_0, w_0)$,
and $\mathbf{f}(t, \mathbf{u}) = (f_1(t, \mathbf{u}), f_2(t, \mathbf{u}), f_3(t, \mathbf{u}))$.
Therefore

$$\begin{aligned} \mathbf{f}(t, \mathbf{u}) &= (f_1(t, u, v, w), f_2(t, u, v, w), f_3(t, u, v, w)) \\ &= \left(uv + \cos w, 2 - t^2 + w^2 y, \sin t - u + v\right). \end{aligned}$$

Second order Runge-Kutta method is applied as

$$\mathbf{u}_{n+1} = \mathbf{u}_n + h(\mathbf{k}_1 + \mathbf{k}_2)/2,$$

where

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{z}_n), \\ \mathbf{k}_2 &= \mathbf{f}(t_n + h, \mathbf{z}_n + h\mathbf{k}_1). \end{aligned}$$

Therefore

$$\begin{aligned} \mathbf{k}_1 &= (k_1(u), k_1(v), k_1(w)) = (f_1(t, u, v, w), f_2(t, u, v, w), f_3(t, u, v, w)) \\ &= \left(uv + \cos w, 2 - t^2 + w^2 v, \sin t - u + v\right), \\ \mathbf{k}_2 &= (k_2(u), k_2(v), k_2(w)) = \mathbf{f}(t_n + h, \mathbf{u}_n + h\mathbf{k}_1) \\ &= \mathbf{f}(t_n + h, \mathbf{u}_n + h(f_1(t, u, v, w), f_2(t, u, v, w), f_3(t, u, v, w))). \end{aligned}$$

For $n = 0$

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_0, \mathbf{u}_0) = \left(u_0 v_0 + \cos w_0, 2 - a^2 + w_0^2 v_0, \sin a - u_0 + v_0\right), \\ \mathbf{k}_2 &= \mathbf{f}(t_0 + h, \mathbf{u}_0 + h\mathbf{k}_1) = \mathbf{f}(a + h, \mathbf{u}_0 + h\left(u_0 v_0 + \cos w_0, 2 - a^2 + w_0^2 v_0, \sin a - u_0 + v_0\right). \end{aligned}$$

$$u_{n+1} = u_n + h(k_1(u) + k_2(u))/2,$$
$$v_{n+1} = v_n + h(k_1(v) + k_2(v))/2,$$
$$w_{n+1} = w_n + h(k_1(w) + k_2(w))/2.$$

For $n = 0$ we get

$$\begin{aligned}
u_1 &= u_0 + h(k_1(u_0) + k_2(u_0))/2,\\
v_1 &= v_0 + h(k_1(v_0) + k_2(v_0))/2,\\
w_1 &= w_0 + h(k_1(w_0) + k_2(w_0))/2,\\
u_1 &= u_0 + h[(u_0v_0 + \cos w_0) + (a + h, u_0 + h(u_0v_0 + \cos w_0))]/2,\\
v_1 &= v_0 + h[(2 - a^2 + w_0^2v_0) + (a + h, v_0 + h(2 - a^2 + w_0^2v_0))]/2,\\
w_1 &= w_0 + h[(\sin a - u_0 + v_0) + (a + h, w_0 + h(\sin a - u_0 + v_0))]/2.
\end{aligned}$$

$\mathbf{u}_1 = (u_1, v_1, w_1)$ this value will be used in next iteration for finding $\mathbf{u}_2$ and so on.

For similar problem **Fourth order Runge-Kutta method** is applied as:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + h(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)/6,$$

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{u}_n),\\
\mathbf{k}_2 &= \mathbf{f}(t_n + h/2, \mathbf{u}_n + hk_1/2),\\
\mathbf{k}_3 &= \mathbf{f}(t_n + h/2, \mathbf{u}_n + hk_2/2),\\
\mathbf{k}_4 &= \mathbf{f}(t_n + h, \mathbf{u}_n + hk_3).
\end{aligned}$$

For $n = 0$

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(t_0, \mathbf{u}_0) = \mathbf{f}(a, u_0, v_0, w_0),\\
\mathbf{k}_2 &= \mathbf{f}(t_0 + h/2, \mathbf{u}_0 + hk_1/2) = \mathbf{f}(a + h/2, (u_0, v_0, w_0) + hk_1/2),\\
\mathbf{k}_3 &= \mathbf{f}(t_0 + h/2, \mathbf{u}_0 + hk_2/2) = \mathbf{f}(a + h/2, (u_0, v_0, w_0) + hk_2/2),\\
\mathbf{k}_4 &= \mathbf{f}(t_0 + h, \mathbf{u}_0 + hk_3) = \mathbf{f}(a + h, (u_0, v_0, w_0) + hk_3).
\end{aligned}$$

Therefore

$$\begin{aligned}
u_{n+1} &= u_n + h[k_1(x) + 2k_2(x) + 2k_3(x) + k_4(x)]/6,\\
v_{n+1} &= v_n + h[k_1(y) + 2k_2(y) + 2k_3(y) + k_4(y)]/6,\\
w_{n+1} &= w_n + h[k_1(z) + 2k_2(z) + 2k_3(z) + k_4(z)]/6.
\end{aligned}$$

For $n = 0$ we get

$$\begin{aligned} u_1 &= u_0 + h[k_1(u_0) + 2k_2(u_0) + 2k_3(u_0) + k_4(u_0)]/6, \\ v_1 &= v_0 + h[k_1(v_0) + 2k_2(v_0) + 2k_3(v_0) + k_4(v_0)]/6, \\ w_1 &= w_0 + h[k_1(w_0) + 2k_2(w_0) + 2k_3(w_0) + k_4(w_0)]/6, \end{aligned}$$

where

$$\begin{aligned} k_1 &= (u_0v_0 + \cos w_0, 2 - a^2 + w_0^2v_0, \sin a - u_0 + v_0), \\ k_2 &= \mathbf{f}(a + h/2, \mathbf{u}_0 + h(u_0v_0 + \cos w_0, 2 - a^2 + w_0^2v_0, \sin a - u_0 + v_0)/2), \\ k_3 &= \mathbf{f}(a + h/2, \mathbf{u}_0 + h(\mathbf{f}(a + h/2, \mathbf{u}_0 + h(u_0v_0 + \cos w_0, 2 - a^2 + w_0^2y_0, \sin a - u_0 + v_0)/2))/2), \\ k_4 &= \mathbf{f}(a + h, \mathbf{u}_0 + h(\mathbf{f}(a + h/2, \mathbf{u}_0 + h(\mathbf{f}(a + h/2, \mathbf{u}_0 + h(u_0v_0 + \cos w_0, 2 - a^2 + w_0^2v_0, \sin a - u_0 + v_0)/2))/2))). \end{aligned}$$

$\mathbf{u}_1 = (u_1, v_1, w_1)$ this value will be used in next iteration for finding $\mathbf{u}_2$ and so on.

2.1.4 Solving Predator-prey model using numerical methods

System is given as:

$$\begin{aligned} r^{'} &= \frac{dr}{dt} = Ar - Brf, \\ f^{'} &= \frac{df}{dt} = -Cf + Drf. \end{aligned}$$

Here

$$\begin{aligned} \mathbf{r} &= (r, f), \\ r^{'} &= f(t, r), \\ f^{'} &= f(t, f), \\ \mathbf{f}(t, \mathbf{r}) &= (f_1(t, r, f), f_2(t, r, f) = (Ar - Brf, -Cf + Drf). \end{aligned}$$

The below given set of ordinary differential equations gives the dynamics of a predator-prey system.

$$\begin{aligned} \frac{dr}{dt} &= Ar - Brf, \quad r(0) = 3 \text{ Rabbits (prey)} \\ \frac{df}{dt} &= Drf - Cf, \quad f(0) = 2 \text{ Foxes (predator)}, \end{aligned}$$

where $A = 2,\ B = 2,\ C = 1,\ D = 1$.
We need the evaluation of $r(0.1)$ and $f(0.1)$ starting from the initial position (at $t = 0$).

Euler's method is applied as

$$\mathbf{z}_{k+1} = z_k + h\mathbf{f}(t_k, z_k), \quad k = 0, 1, 2, ..., n-1.$$

Here taking step size $h = 0.1$,
putting the A, B, C, D in the system we get

$$\begin{aligned} \frac{dr}{dt} &= 2r - 2rf, \\ \frac{df}{dt} &= rf - f. \end{aligned}$$

Starting with $r = 3$ and $f = 2$ and slopes evaluated at these starting points are

$$\begin{aligned} \frac{dr}{dt} &= 2(3) - 2(3)(2) = 6 - 12 = -6, \\ \frac{df}{dt} &= (3)(2) - (2) = 6 - 2 = 4. \end{aligned}$$

With $h = 0.1$

$$\begin{aligned} r(0.1) &= r_1 = r_0 + hf(t_0, r_0) = r_0 + h(r_0^{'}) \\ &= 3 + 0.1(-6) = 2.4, \\ f(0.1) &= f_1 = f_0 + hf(t_0, f_0) = f_0 + h(f_0^{'}) \\ &= 2 + 0.1(4) = 2.4 \end{aligned}$$

After ist iteration we get $\mathbf{r}_1 = (r_1, f_1) = (2.4, 2.4) \approx (2, 2)$.

Improved Euler's method

$$\begin{aligned} \frac{dr}{dt} &= Ar - Brf, \quad r(0) = 3 \text{ Rabbits (prey)}, \\ \frac{df}{dt} &= Drf - Cf, \quad f(0) = 2 \text{ Foxes (predator)}, \end{aligned}$$

where $A = 2,\ B = 2,\ C = 1,\ D = 1$.
We have calculate $r(0.1)$ and $f(0.1)$ starting from the initial position (at $t = 0$). Putting A, B, C, D in the system we get

$$\begin{aligned} \frac{dr}{dt} &= 2r - 2rf, \\ \frac{df}{dt} &= rf - f. \end{aligned}$$

Above method is applied as:

$$\begin{aligned} \mathbf{r}_{k+1/2} &= r_k + \frac{h}{2} f(t_k, r_k), \\ \mathbf{r}_{k+1} &= r_k + h f(t_k + h/2, r_{k+1/2}), \end{aligned}$$

Here $h = 0.1$

$$\begin{aligned} \frac{dr}{dt} &= -6, \\ \frac{df}{dt} &= 4, \\ r_{1/2} &= r_0 + \frac{h}{2}(r_0^{'}) = 3 + \frac{0.1}{2}(-6) = 2.7, \\ f_{1/2} &= f_0 + \frac{h}{2}(f_0^{'}) = 2 + \frac{0.1}{2}(4) = 2.2, \\ r_1 &= r_0 + hf(0.1/2, 2.7) = 3 + 0.1(-6.48) = 2.352, \\ f_1 &= f_0 + hf(0.1/2, 2.2) = 2 + 0.1(3.74) = 2.374 \end{aligned}$$

After ist iteration we get $\mathbf{r}_1 = (r_1, f_1) = (2.352, 2.374) \approx (2, 2)$.

Runge-Kutta method of second order

$$\begin{aligned} \frac{dr}{dt} &= Ar - Brf, \quad r(0) = 3 \text{ Rabbits (prey)}, \\ \frac{df}{dt} &= Drf - Cf, \quad f(0) = 2 \text{ Foxes (predator)}, \end{aligned}$$

in this $A = 2, B = 2, C = 1, D = 1$.
We have evaluate $r(0.1)$ and $f(0.1)$ starting from the initial position (at $t = 0$). Putting A, B, C, D in the system we get

$$\begin{aligned} \frac{dr}{dt} &= 2r - 2rf, \\ \frac{df}{dt} &= rf - f. \end{aligned}$$

Method is applied as:

$$\begin{aligned} z_{n+1} &= z_n + h(k_1 + k_2)/2, \\ k_1 &= f(t_n, z_n), \\ k_2 &= f(t_n + h, z_n + hk_1). \end{aligned}$$

Here $h = 0.1$

$$\begin{aligned}
\frac{dr}{dt} &= -6, \\
\frac{df}{dt} &= 4, \\
k_1(r) &= r^{'} = -6, \\
k_2(r) &= hf(0.1, r_0 + k_1(r)) = hf(0.1, 3 + (-6)) = hf(0.1, -3) = 0.1(30) = 3, \\
k_1(f) &= y^{'} = 4, \\
k_2(f) &= hf(0.1, f_0 + k_1(f)) = hf(0.1, 2 + 4) = hf(0.1, 6) = 0.1(-24) = -2.4,
\end{aligned}$$

therefore using these values we get

$$\begin{aligned}
r_1 &= r_0 + h(k_1(r) + k_2(r))/2 = 3 + 0.1(-6 + 3)/2 = 3 - 0.15 = 2.85, \\
f_1 &= f_0 + h(k_1(f) + k_2(f))/2 = 2 + 0.1(4 - 2.4)/2 = 2 + 0.08 = 2.08
\end{aligned}$$

After ist iteration we get $\mathbf{r}_1 = (r_1, f_1) = (2.85, 2.08) \approx (3, 2)$.

Runge-Kutta method of fourth order

$$\begin{aligned}
\frac{dr}{dt} &= Ar - Brf, \quad r(0) = 3 \text{ Rabbits (prey)}, \\
\frac{df}{dt} &= Drf - Cf, \quad f(0) = 2 \text{ Foxes (predator)},
\end{aligned}$$

given $A = 2,\ B = 2,\ C = 1,\ D = 1$.
We need to evaluate $r(0.1)$ and $f(0.1)$ starting from the initial position (at $t = 0$). Putting A, B, C, D in the system we get

$$\begin{aligned}
\frac{dr}{dt} &= 2r - 2rf, \\
\frac{df}{dt} &= rf - f.
\end{aligned}$$

Method is applied as:

$$z_{n+1} = z_n + h\frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6},$$

where

$$\begin{aligned}
k_1 &= f(t_n, z_n), \\
k_2 &= f(t_n + h/2, z_n + hk_1/2), \\
k_3 &= f(t_n + h/2, z_n + hk_2/2), \\
k_4 &= f(t_n + h, z_n + hk_3).
\end{aligned}$$

Here $h = 0.1$

$$\begin{aligned}
\frac{dr}{dt} &= -6, \\
\frac{df}{dt} &= 4, \\
k_1(r) &= r' = -6, \\
k_1(f) &= f' = 4, \\
k_2(r) &= hf(0.1/2, r_0 + k_1(r)/2) = hf(0.1/2, 3 + (-6)/2) = 0.1f(0.1/2, 0) = 0, \\
k_2(f) &= hf(0.1/2, y_0 + k_1(f)/2) = hf(0.1/2, 2 + 4/2) = hf(0.1/2, 4) = 0.1(-2) = -0.2, \\
k_3(r) &= hf(0.1/2, r_0 + k_2(r)/2) = hf(0.1/2, 3 + 0) = hf(0.1/2, 3) = 0.1(-5.4) = -0.54, \\
k_3(f) &= hf(0.1/2, f_0 + k_2(f)/2) = hf(0.1/2, 2 + (-0.2)/2) = hf(0.1/2, 1.9) = 0.1(3.8) = 0.38, \\
k_4(r) &= hf(0.1, r_0 + k_3(r)) = hf(0.1, 3 + (-0.54)) = 0.1f(0.1, 2.46) = -0.678, \\
k_4(f) &= hf(0.1, f_0 + k_3(f)) = hf(0.1, 2 + 0.38) = 0.1f(0.1, 2.38) = 0.1(3.475) = 0.3475,
\end{aligned}$$

therefore we get

$$\begin{aligned}
r_1 &= r_0 + \frac{h}{6}(k_1(r) + 2k_2(r) + 2k_3(r) + k_4(r)) \\
&= 3 + \frac{0.1}{6}(-6 + 2(0) + 2(-0.54) + -0.678) \\
&= 2.87, \\
f_1 &= f_0 + \frac{h}{6}(k_1(f) + 2k_2(f) + 2k_3(f) + k_4(f)) \\
&= 2 + \frac{0.1}{6}(4 + 2(-0.2) + 2(0.38) + 0.3475) \\
&= 2.078
\end{aligned}$$

After ist iteration we get $\mathbf{r}_1 = (r_1, f_1) = (2.87, 2.078) \approx (3, 2)$.

Implementation in MATLAB

Results are obtained by performing five iterations

Euler's method

predator.m

```
h=0.1;
iter=5;
fox(1)=3; rab(1)=2; t(1)=0;
a=2; b=2; c=1; d=1;
for i=2:iter
    dfox = a*fox(i-1) - b*fox(i-1)*rab(i-1);
    drab = -c*rab(i-1) + d*rab(i-1)*fox(i-1);
    fox(i) = fox(i-1) + h*dfox;
    rab(i) = rab(i-1) + h*drab;
    t(i) = t(i-1) + h;

end
```

Output

fox

1×5 double

	1	2	3	4	5
1	3	2.4000	1.7280	1.1280	0.6914

fox rab

1×5 double

	1	2	3	4	5
1	2	2.4000	2.7360	2.9352	2.9728

Modified Euler's method

imp.m *

```
h=0.1;
iter=5;
fx(1)=3; rb(1)=2; t(1)=0;
a=2; b=2; c=1; d=1;
for i=2:iter
    dfx = a*fx(i-1) - b*fx(i-1)*rb(i-1);
    drb = -c*rb(i-1) + d*rb(i-1)*fx(i-1);
    X(i) = fx(i-1) + (0.5*h)*dfx;
    Y(i)  = rb(i-1) + (0.5*h)*drb;
    dfx(i) = a*X(i)-b*X(i)*Y(i);
    drb(i) = -c*Y(i) + d*Y(i)*X(i);
    fx(i) = fx(i-1) + h*dfx(i);
    rb(i) = rb(i-1) + h*drb(i);
    t(i) = t(i-1) + h;
end
```

Output

rb fx

1×5 double

	1	2	3	4	5
1	3	2.3520	1.7294	1.2286	0.8664

rb

1×5 double

	1	2	3	4	5
1	2	2.3740	2.6348	2.7567	2.7603

Second order Runge-Kutta method

ru2.m

```
h=0.1;
iter=5;
fo(1)=3; ra(1)=2; t(1)=0;
a=2; b=2; c=1; d=1;
for i=2:iter
    dfo = a*fo(i-1) - b*fo(i-1)*ra(i-1);
    dra = -c*ra(i-1) + d*ra(i-1)*fo(i-1);
    k1fo(i) = h*dfo ; k1ra(i) = h*dra;
    k2fo(i) = h*(a*(fo(i-1) + k1fo(i)) - b*(fo(i-1)+k1fo(i))*(ra(i-1)+k1ra(i)));
    k2ra(i) = h*(-c*(ra(i-1)+k1ra(i)) + d*(fo(i-1)+k1fo(i))*(ra(i-1)+k1ra(i)));
     fo(i) = fo(i-1) + (k1fo(i) + k2ra(i))*(1/2);
     ra(i) = ra(i-1) + (k1ra(i) + k2ra(i))*(1/2);
end
```

Output

fo ×

1×5 double

	1	2	3	4	5
1	3	2.8680	2.6279	2.2838	1.8674

fo × ra ×

1×5 double

	1	2	3	4	5
1	2	2.3680	2.7414	3.0781	3.3338

Fourth order Runge-Kutta method

predator.m × imp.m × ru2.m × ru4.m × +

```
h=0.1;
iter=5;
p(1)=3; r(1)=2; t(1)=0;
a=2; b=2; c=1; d=1;
for i=2:iter
    dp = a*p(i-1) - b*p(i-1)*r(i-1);
    dr = -c*r(i-1) + d*r(i-1)*p(i-1);
    k1p(i) = h*dp ; k1r(i) = h*dr;
    k2p(i) = h*(a*(p(i-1) +  0.5*k1p(i)) - b*(p(i-1)+0.5*k1p(i))*(r(i-1)+0.5*k1r(i)));
    k2r(i) = h*(-c*(r(i-1)+ 0.5*k1r(i)) + d*(r(i-1) + 0.5*k1r(i))*(p(i-1)+0.5* k1p(i)));
   k3p(i) = h*(a*(p(i-1) +  0.5*k2p(i)) - b*(p(i-1)+0.5*k2p(i))*(r(i-1)+0.5*k2r(i)));
    k3r(i) = h*(-c*(r(i-1)+ 0.5*k2r(i)) + d*(r(i-1) + 0.5*k2r(i))*(p(i-1)+0.5* k2p(i)));
    k4p(i) = h*(a*(p(i-1) + k3p(i)) - b*(p(i-1)+ k3p(i))*(r(i-1)+ k3r(i)));
    k4r(i) = h*(-c*(r(i-1)+ k3r(i)) + d*(r(i-1) + k3r(i))*(p(i-1)+ k3p(i)));
    p(i) = p(i-1) + (k1p(i) + 2*(k2p(i)+k3p(i)) + k4p(i))*(1/6);
    r(i) = r(i-1) + (k1r(i) + 2*(k2r(i)+k3r(i)) + k4r(i))*(1/6);
    t(i) =t(i-1) + h;
end
```

Output

k1p × k1r × k2p × k2r × k3p × k3r × k4p × k4r × p ×

1×5 double

	1	2	3	4	5
1	3	2.3645	1.7485	1.2432	0.8717

k1p × k1r × k2p × k2r × k3p × k3r × k4p × k4r × p × r ×

1×5 double

	1	2	3	4	5
1	2	2.3673	2.6295	2.7601	2.7730

2.1.5 Solving Van der Pol equation using numerical methods

The second order differential equation of forced Van der Pol oscillator is given as:

$$\frac{d^2z}{dt^2} - \mu(z^2-1)\frac{dz}{dt} + z = f(t),$$

given with initial conditions as $z(0) = z^{'}(0) = 1$. Starting with $t = 0$ and step size $h = 0.1$, the above equation is convertible to a system of first order equations as follows:

$$\begin{aligned} z^{'} &= w, \\ w^{'} &= \mu(z^2-1)w - z. \end{aligned}$$

For $\mu = 1$ system is:

$$\begin{aligned} z^{'} &= w, \\ w^{'} &= (z^2-1)w - z, \end{aligned}$$

where

$$\begin{aligned} f_1(t,z,w) &= w, \\ f_2(t,z,w) &= (z^2-1)w - z, \\ F(t,z,w) &= (f_1(t,z,w), f_2(t,z,w) = (w, (z^2-1)w - z). \end{aligned}$$

Applying **Euler's method** :

$$z_{k+1} = z_k + hF(t_k, z_k),$$

$$(z_{k+1}, w_{k+1}) = (z_k, w_k) + h(f_1(t,z,w), f_2(t,z,w),$$

here $z = (z, w)$. Therefore

$$\begin{aligned} z_1 &= z_0 + hf_1(t,z,w) \\ &= 1 + 0.1(w_0) = 1 + 0.1(1) = 1 + 0.1 = 1.1, \\ w_1 &= w_0 + hf_2(t,z,w) \\ &= 1 + 0.1((z_0^2-1)w_0 - z_0) = 1 + 0.1((1-1)1 - 1) = 0.9 \end{aligned}$$

After ist iteration we get $z_1 = (z_1, w_1) = (1.1, 0.9)$.
Applying **Improved Euler's method** :

$$\begin{aligned} z_{k+1/2} &= z_k + \frac{h}{2}F(t_k, z_k), \\ z_{k+1} &= z_k + hF(t_k + h/2, z_{k+1/2}), \end{aligned}$$

i.e.

$$\begin{aligned}(z_{k+1/2}, w_{k+1/2}) &= (z_k, w_k) + h(f_1(t, z, w), f_2(t, z, w),\\ (z_{k+1}, w_{k+1}) &= (z_k, w_k) + hF(t_k + h/2, z_{k+1/2}),\end{aligned}$$

here $z = (z, w)$. Therefore

$$\begin{aligned}z_{1/2} &= z_0 + hf_1(t, z, w) = 1 + 0.1(w_0) = 1 + 0.1(1) = 1.1,\\ w_{1/2} &= w_0 + hf_2(t, z, w) = 1 + 0.1((z_0^2 - 1)w_0 - z_0) = 1 + 0.1(-1) = 0.9,\\ z_1 &= z_0 + hf_1(h/2, z_{1/2}) = z_0 + hf_1(0.1/2, 1.1) = 1 + 0.1(1) = 1.1,\\ w_1 &= w_0 + hf_2(h/2, w_{1/2}) = w_0 + hf_2(0.1/2, 0.9) = 1 + 0.1(-1) = 0.9\end{aligned}$$

After ist iteration we get $z_1 = (z_1, w_1) = (1.1, 0.9)$.

Applying **Runge-Kutta method of second order** :

$$\begin{aligned}z_{n+1} &= z_n + h(k_1 + k_2)/2,\\ k_1 &= f(t_n, z_n),\\ k_2 &= f(t_n + h, z_n + hk_1),\end{aligned}$$

here $z = (z, w)$.

$$(z_{n+1}, w_{n+1}) = (z_n, w_n) + h(k_1 + k_2)/2.$$

$$\begin{aligned}k_1 &= (k_1(z), k_1(w)),\\ k_2 &= (k_2(z), k_2(w),\\ z_{n+1} &= z_n + h(k_1(z_n) + k_2(z_n))/2,\\ w_{n+1} &= w_n + h(k_1(w_n) + k_2(w_n))/2.\end{aligned}$$

Therefore

$$\begin{aligned}k_1(z_0) &= f_1(t, z, w) = w = w_0 = 1,\\ k_2(z_0) &= hf_1(0.1, z_0 + k_1(z_0)) = hf_1(0.1, 1 + 1) = hf_1(0.1, 2) = 0.1(1) = 0.1,\\ k_1(w_0) &= f_2(t, z, w) = (z_0^2 - 1)w_0 - z_0 = -1,\\ k_2(w_0) &= hf_2(0.1, w_0 + k_1(w_0)) = hf_2(0.1, 0) = 0.1((u_0^2 - 1)v_0 - u_0) = 0.1(-1) = -0.1,\end{aligned}$$

using these values we get

$$\begin{aligned} z_1 &= z_0 + h(k_1(z_0) + k_2(z_0))/2 = 1 + 0.1(1 + 0.1)/2 = 1 + 0.1(1.1/2) = 1.055, \\ w_1 &= w_0 + h(k_1(w_0) + k_2(w_0))/2 = 1 + 0.1(-1 - 0.1)/2 = 1 + 0.1(-1.1/2) = 0.945 \end{aligned}$$

After ist iteration we get $z_1 = (z_1, w_1) = (1.055, 0.945)$.

Runge-Kutta method of fourth order is applied as:

$$z_{n+1} = z_n + h\frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6},$$

where

$$\begin{aligned} k_1 &= f(t_n, z_n, \\ k_2 &= f(t_n + h/2, z_n + hk_1/2), \\ k_3 &= f(t_n + h/2, z_n + hk_2/2), \\ k_4 &= f(t_n + h, z_n + hk_3), \end{aligned}$$

here $z = (z, w)$.

$$\begin{aligned} (z_{n+1}, w_{n+1}) &= (z_n, w_n) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ z_{n+1} &= z_n + \frac{h}{6}(k_1(z_n) + 2k_2(z_n) + 2k_3(z_n) + k_4(z_n)), \\ w_{n+1} &= w_n + \frac{h}{6}(k_1(w_n) + 2k_2(w_n) + 2k_3(w_n) + k_4(w_n)), \end{aligned}$$

where

$$\begin{aligned} k_1(z_0) &= f_1(t, z, w) = w = w_0 = 1, \\ k_2(z_0) &= hf_1(h/2, z_0 + k_1(z_0)/2) = 0.1f_1(0.1/2, 1 + 1/2) = 0.1f_1(0.1/2, 1.5) = 0.1(1) = 0.1, \\ k_3(z_0) &= hf_1(h/2, z_0 + k_2(z_0)/2) = 0.1f_1(0.1/2, 1 + 0.1/2) = 0.1(1) = 0.1, \\ k_4(z_0) &= hf_1(h, z_0 + k_3(z_0)) = 0.1f_1(0.1, 1.1) = 0.1(1) = 1.1, \\ k_1(w_0) &= f_2(t, z, w) = (z^2 - 1)w - z = (z_0^2 - 1)w_0 - z_0 = -1, \\ k_2(w_0) &= hf_2(h/2, w_0 + k_1(w_0)/2) = hf_2(0.1/2, 1 - 1/2) = hf_2(0.1/2, 0.5) = 0.1(-1) = -0.1, \\ k_3(w_0) &= hf_2(h/2, w_0 + k_2(w_0)/2) = hf_2(0.1/2, 1 - 0.1/2) = hf_2(0.1/2, 1 - 0.05) = 0.1(-1) = -0.1, \\ k_4(w_0) &= hf_2(h, w_0 + k_3(w_0)) = hf_2(0.1, 1 - 0.1) = hf_2(0.1, 0.9) = 0.1(-1) = -0.1, \end{aligned}$$

therefore using these values we get

$$\begin{aligned} z_{n+1} &= z_n + \frac{h}{6}(k_1(z_0) + 2k_2(z_0) + 2k_3(z_0) + k_4(z_0)), \\ z_1 &= z_0 + \frac{0.1}{6}(k_1(z_0) + 2k_2(z_0) + 2k_3(z_0) + k_4(z_0)) \\ &= 1 + \frac{0.1}{6}(1 + 2(0.1) + 2(0.1) + 0.1) \\ &= 1 + \frac{0.1}{6}(1.5) = 1.025, \end{aligned}$$

$$\begin{aligned} w_{n+1} &= w_n + \frac{h}{6}(k_1(w_0) + 2k_2(w_0) + 2k_3(w_0) + k_4(w_0)), \\ w_1 &= w_0 + \frac{0.1}{6}(k_1(w_0) + 2k_2(w_0) + 2k_3(w_0) + k_4(w_0)) \\ &= 1 + \frac{0.1}{6}(-1 + 2(-0.1) + 2(-0.1) - 0.1) \\ &= 1 + \frac{0.1}{6}(-1.5) = 0.975 \end{aligned}$$

After ist iteartion we get $z_1 = (z_1, w_1) = (1.025, 0.975)$.

Implementation in MATLAB

Results are obtained by performing five iterations

Euler's method

vaneu.m × predator.m × +

```
h=0.1;
iter =5;
u(1) = 1; v(1)=1; t(1)=0;
mu=1;
for i=2:iter
    du = v(i-1);
    dv = mu*(u(i-1)^2 -1)*v(i-1) -u(i-1);
    u(i) = u(i-1) + h*du;
    v(i) = v(i-1) + h*dv;
    t(i) = t(i-1) + h;
end
```

Output

v × u ×

1×5 double

	1	2	3	4	5
1	1	1.1000	1.1900	1.2709	1.3432

v × u ×

1×5 double

	1	2	3	4	5
1	1	0.9000	0.8089	0.7236	0.6410

Modified Euler's method

vanimp.m

```
h=0.1;
iter =5;
uo(1) = 1; vo(1)=1; t(1)=0;
mu=1;
for i=2:iter
    duo = vo(i-1);
    dvo = mu*(uo(i-1)^2 -1)*vo(i-1) - uo(i-1);
    U(i) = uo(i-1) + (0.5*h)*duo;
    V(i) = vo(i-1) + (0.5*h)*dvo;
    duo(i) = U(i);
    dvo(i) = mu*(U(i)^2 -1)*V(i) - U(i);
    uo(i)= uo(i-1) + h*duo(i);
    vo(i) = vo(i-1) + h*dvo(i);
    t(i) = t(i-1) + h;
end
```

Output

uo

1×5 double

	1	2	3	4	5
1	1	1.1050	1.2200	1.3461	1.4844

vo

1×5 double

	1	2	3	4	5
1	1	0.9047	0.8175	0.7372	0.6628

Second order Runge-Kutta method

vanimp.m vanru2.m ru2.m

```
h=0.1;
iter =5;
ur(1) = 1; vr(1)=1; t(1)=0;
mu=1;
for i=2:iter
     dur = vr(i-1);
    dvr = mu*(ur(i-1)^2 -1)*vr(i-1) - ur(i-1);
    k1ur(i) = h*dur ; k1vr(i) = h*dvr;
    k2ur(i) = h*( vr(i-1)  + k1vr(i));
    k2vr(i) = h*(mu*(ur(i-1)^2 + k1ur(i) -1)*(vr(i-1)  + k1vr(i)) - (ur(i-1) + k1ur(i)));
    ur(i) =  ur(i-1) + (k1ur(i) + k2ur(i))*(1/2);
    vr(i) =  vr(i-1) + (k1vr(i) + k2vr(i))*(1/2);
    t(i) = t(i-1) + h;
end
```

Output

ur

1×5 double

	1	2	3	4	5
1	1	1.0950	1.1804	1.2567	1.3242

vr

1×5 double

	1	2	3	4	5
1	1	0.8995	0.8061	0.7170	0.6291

Fourth order Runge-Kutta method

vanru4.m * | vanru2.m | ru4.m

```
h=0.1;
iter =5;
uk(1) = 1; vk(1)=1; t(1)=0;
mu=1;
for i=2:iter
    duk = vk(i-1);
    dvk = mu*(uk(i-1)^2 -1)*vk(i-1) - uk(i-1);
    k1uk(i) = h*duk; k1vk(i) = h*dvk;
    k2uk(i) = h*(vk(i-1) + 0.5*k1vk(i));
    k2vk(i) = h*(mu*(uk(i-1)^2 -1 + 0.5*k1uk(i))*(vk(i-1) +0.5*k1vk(i)) -(uk(i-1)+0.5*k1uk(i)));
    k3uk(i) = h*(vk(i-1) + 0.5*k2vk(i));
    k3vk(i) = h*(mu*(uk(i-1)^2 -1 + 0.5*k2uk(i))*(vk(i-1) +0.5*k2vk(i)) -(uk(i-1)+0.5*k2uk(i)));
    k4uk(i) = h*(vk(i-1) + k3vk(i));
    k4vk(i) = h*(mu*(uk(i-1)^2 -1 + k3uk(i))*(vk(i-1) +k3vk(i)) -(uk(i-1)+k3uk(i)));
    uk(i) = uk(i-1) + (k1uk(i) + 2*(k2uk(i)+k3uk(i)) + k4uk(i))*(1/6);
    vk(i) = vk(i-1) + (k1vk(i) + 2*(k2vk(i)+k3vk(i)) + k4vk(i))*(1/6);
    t(i) =t(i-1) + h;
end
```

Output

uk × vk ×

1×5 double

	1	2	3	4	5
1	1	1.0950	1.1803	1.2566	1.3240

uk × vk ×

1×5 double

	1	2	3	4	5
1	1	0.8997	0.8065	0.7174	0.6296

Chapter 3

IMPLEMENTING NUMERICAL METHODS ON REAL LIFE PROBLEMS

3.1 PROBLEM 1 : PRICING POLICY FOR THE PRODUCTION OF GOODS

Differential equations have it's role in every field and economics is one of them. Pricing policy for the production of goods is one of the applications of differential equations in economics. This section deals with how the numerical methods can be helpful in solving the problems of economics field. Numerical methods can be used as a tool by businessmen to decide various pricing strategies for their companies.

3.1.1 What is pricing policy?

The policy/strategy which company plans to determine the wholesale and retail prices for it's products or services. Activities involved in finding a product's optimum price considering each and every objectives of marketing, consumer's demands, product properties, pricing of competitors and market and economic trends all together forms a pricing policy of a product.

3.1.2 Mathematical formulation of pricing policy

Mathematical techniques have an immense role in planning the management's decision-making, administration of business and in economics which is being the longest quantified of the social sciences. The source of motivation for this work is the model of Burghes and wood (1980) on pricing for optimum inventory level.

Pricing Policy and Inventory Management

Terms to be used

• **Forecasting sales**- It is process of estimating future sales.

• **Inventory level**- It is the current amount of product that firm has in it's stock.

• **optimum level**- It is the sufficient amount of product required in firm's stock.

Inventory management becomes necessary in business management because many producers carry a some basic stock in their product's inventory so as to meet an all of sudden demand by customers. There are various factors that may affect the quantity of the goods to be carried. Some of these factors include:

(1) The availability of storage facilities.

(2) Storage capacity.

(3) Safety requirements.

Model :

We assume that a Brick Company fixes an optimum level H_0 for storing it's unsold bricks. We also assume that if the inventory level is above H_0, the price of bricks decline and this lead to increase in sales and a fall in inventory. If inventory is below H_0, the price goes up, in return discouraging buyers and raise the inventory. Let the inventory level be H and $P(t)$ be price of brick at any time t, which is calculated in proportion to the difference of the inventory level at time t, $H(t)$ and optimum level H_0. This is mathematically described as:

$$\frac{dP}{dt} = -k(H(t) - H_0), \tag{3.1}$$

where k is positive constant of proportionality.

For forecasting sales S (this is given as units of a thousand bricks) the company uses the formula:

$$S = 600 - 62P - 12\frac{dP}{dt}.$$

From this we observe that S increases when $\frac{dP}{dt} < 0$ and while evaluating production level Q (also in thousands), company used

$$Q = 250 - 12P.$$

Since without any doubt this policy maintains the inventory about it's optimum level, concern in the company is that it requires the check on frequent large change in price, causing fear and confusion that discourage customers in the long term. A more precise examination of the long-run nature of $P(t)$ with respect to it's stability is therefore a requirement. This is always correct that the difference between the quantities of bricks produced and sold is what we call change in inventory i.e.

$$\frac{dH}{dt} = Q(t) - S(t).$$

Differentiating (3.1) we get

$$\begin{aligned}\frac{d^2P}{dt^2} &= -k(\frac{dH}{dt}) = -k(Q(t) - S(t)) \\ &= -k(250 - 12P - (600 - 62P - 12\frac{dP}{dt})) \\ &= -k(-350 + 50P + 12\frac{dP}{dt}).\end{aligned}$$

Therefore we get second-order differential equation as

$$\frac{d^2P}{dt^2} + \ 12\ k(\frac{dP}{dt}) + \ 50\ kP = \ 350\ k.$$

3.1.3 Solving numerically second-order differential equation of pricing policy

From the pricing policy model we get the second-order differential equation as

$$\frac{d^2P}{dt^2} + \ 12\ k(\frac{dP}{dt}) + \ 50\ kP = \ 350\ k.$$

Here k is positive constant of proportionality, let $k = 1$. Therefore equation becomes

$$\frac{d^2P}{dt^2} + \ 12(\frac{dP}{dt}) + \ 50P = \ 350.$$

CONVERSION OF EQUATION TO SYSTEM OF EQUATIONS

Since the differential equation of second order is convertible into 2 first-order differential equations' system. This is done by following substitution i.e.

$$p = \frac{dP}{dt},$$

and we get

$$\frac{d^2P}{dt^2} = \frac{d}{dt}(\frac{dP}{dt}) = p^{'}.$$

And the differential equation becomes

$$p^{'} + \ 12p + 50\ P = \ 350,$$

which is equivalent to the system

$$\begin{aligned}P^{'} &= \frac{dP}{dt} = p, \\ p^{'} &= 350 - 50P - 12p.\end{aligned}$$

3.1.4 Numerical methods

Now solving the system using numerical methods.

$$\begin{aligned} P^{'} &= \frac{dP}{dt} = p, \\ p^{'} &= 350 - 50P - 12p. \end{aligned}$$

Assuming $P(0) = 2$ and $p(0) = 2$.

Euler's method

Applying Euler's method

$$\mathbf{z}_{k+1} = \mathbf{z}_k + h\mathbf{f}(t_k, \mathbf{z}_k), \quad k = 0, 1, 2, ..., n-1,$$

here $\mathbf{z} = (P, p)$ and $\mathbf{f}(t_k, \mathbf{z}_k) = (f_1(t_k, \mathbf{z}_k), f_2(t_k, \mathbf{z}_k))$.
Here $h = 0.1$ and starting from $t = 0$.
For $P^{'} = p = f_1(t, P, p)$ and $p^{'} = 350 - 50P - 12p = f_2(t, P, p)$, we start with
Iteration 1

$$\begin{aligned} P_1 &= P_0 + hf_1(t_0, P_0, p_0) \\ &= 2 + 0.1(2) = 2.2, \\ p_1 &= p_0 + hf_2(t_0, P_0, p_0) \\ &= 2 + 0.1(350 - 50P_0 - 12p_0) = 2 + 0.1(226) = 24.6 \end{aligned}$$

Therefore after ist iteration $\mathbf{z}_1 = (P_1, p_1) = (2.2, 24.6)$.

Iteration 2

$$\begin{aligned} P_2 &= P_1 + hf_1(t_1, P_1, p_1) \\ &= 2.2 + 0.1(24.6) = 4.66, \\ p_2 &= p_1 + hf_2(t_1, P_1, p_1) \\ &= 24.6 + 0.1(350 - 50P_1 - 12p_1) = 24.6 + 0.1(-55.2) = 19.08 \end{aligned}$$

Therefore after 2nd iteration $\mathbf{z}_2 = (P_2, p_2) = (4.66, 19.08)$.
Continuing in this way we get
$\mathbf{z}_3 = (P_3, p_3) = (6.56, 7.89)$.
$\mathbf{z}_4 = (P_4, p_4) = (7.35, 0.62)$.
$\mathbf{z}_5 = (P_5, p_5) = (7.4, -1.87)$.
$\mathbf{z}_6 = (P_6, p_6) = (7.22, -1.676)$.

$\mathbf{z}_7 = (P_7, p_7) = (7.052, -0.766)$.
$\mathbf{z}_8 = (P_8, p_8) = (6.97, -0.107)$.
$\mathbf{z}_9 = (P_9, p_9) = (6.96, 0.171)$.
$\mathbf{z}_{10} = (P_{10}, p_{10}) = (6.98, 0.1658)$.
$\mathbf{z}_{11} = (P_{11}, p_{11}) = (6.99, 0.067)$.
Therefore we get the approximate solution as $P(t) \approx 7$.

Improved Euler's method

Applying Euler's midpoint method

$$\begin{aligned} \mathbf{z}_{k+1/2} &= \mathbf{z}_k + \frac{h}{2}\mathbf{f}(t_k, \mathbf{z}_k), \quad k = 0, 1, 2, ..., n-1 \\ \mathbf{z}_{k+1} &= \mathbf{z}_k + h\mathbf{f}(t_k + h/2, \mathbf{z}_{k+1/2}), \end{aligned}$$

here $\mathbf{z} = (P, p)$ and $\mathbf{f}(t_k, \mathbf{z}_k) = (f_1(t_k, \mathbf{z}_k), f_2(t_k, \mathbf{z}_k))$.
Here $h = 0.1$ and starting from $t = 0$.
For $P' = p = f_1(t, P, p)$ and $p' = 350 - 50P - 12p = f_2(t, P, p)$, we start with

Iteration 1

$$\begin{aligned} P_{1/2} &= P_0 + \frac{h}{2} f_1(t_0, P_0, p_0) \\ &= 2 + \frac{0.1}{2}(p_0) = 2 + \frac{0.1}{2}(2) = 2.1, \\ p_{1/2} &= p_0 + \frac{h}{2} f_2(t_0, P_0, p_0) \\ &= 2 + \frac{0.1}{2}[350 - 50(2) - 12(2)] \\ &= 2 + \frac{0.1}{2}[226] = 13.3, \end{aligned}$$

$$\begin{aligned} P_1 &= P_0 + h f_1(t_0 + h/2, P_{1/2}, p_{1/2}) \\ &= 2 + 0.1 f_1(0.1/2, 2.1, 13.3) = 2 + 0.1(13.3) \\ &= 2 + 1.33 = 3.33, \\ p_1 &= p_0 + h f_2(t_0 + h/2, P_{1/2}, p_{1/2}) \\ &= p_0 + 0.1 f_2(0.1/2, 2.1, 13.3) = 2 + 0.1[350 - 50(2.1) - 12(13.3)] \\ &= 2 + 0.1[85.4] = 2 + 8.54 = 10.54 \end{aligned}$$

Therefore after ist iteration $\mathbf{z}_1 = (P_1, p_1) = (3.33, 10.54)$.

Iteration 2

$$\begin{aligned} P_{1+1/2} &= P_1 + \frac{h}{2} f_1(t_1, P_1, p_1) \\ &= 3.33 + \frac{0.1}{2}(p_1) = 3.33 + \frac{0.1}{2}(10.54) \\ &= 3.33 + 0.527 = 3.86, \\ p_{1+1/2} &= p_1 + \frac{h}{2} f_2(t_1, P_1, p_1) \\ &= 10.54 + \frac{h}{2}[350 - 50(3.33) - 12(10.54)] = 10.54 + \frac{0.1}{2}[57.02] \\ &= 10.54 + 2.851 = 13.39, \end{aligned}$$

$$\begin{aligned} P_2 &= P_1 + h f_1(t_1 + h/2, P_{1+1/2}, p_{1+1/2}) \\ &= 3.33 + 0.1 f_1(t_1 + 0.1/2, 3.86, 13.39) \\ &= 3.33 + 0.1(13.39) = 4.67, \\ p_2 &= p_1 + h f_2(t_1 + h/2, P_{1+1/2}, p_{1+1/2}) \\ &= 10.54 + 0.1 f_2(t_1 + 0.1/2, 3.86, 13.39) = 10.54 + 0.1[350 - 50(3.86) - 12(13.39)] \\ &= 10.54 + 0.1[-3.68] = 10.17 \end{aligned}$$

Therefore after 2nd iteration $\mathbf{z}_2 = (P_2, p_2) = (4.67, 10.17)$.
Continuing in this way we get
$\mathbf{z}_3 = (P_3, p_3) = (5.66, 7.398)$.
$\mathbf{z}_4 = (P_4, p_4) = (6.29, 4.68)$.
$\mathbf{z}_5 = (P_5, p_5) = (6.65, 2.68)$.
$\mathbf{z}_6 = (P_6, p_6) = (6.84, 1.42)$.
$\mathbf{z}_7 = (P_7, p_7) = (6.94, 0.71)$.
$\mathbf{z}_8 = (P_8, p_8) = (6.98, 0.34)$.
$\mathbf{z}_9 = (P_9, p_9) = (6.99, 0.17)$.
Therefore we get the approximate solution as $P(t) \approx 7$.

Runge-Kutta method

Applying **Runge-Kutta method of second-order**

$$\begin{aligned} \mathbf{z}_{k+1} &= \mathbf{z}_k + (k_1 + k_2)/2, \\ k_1 &= h\mathbf{f}(t_k, \mathbf{z}_k), \\ k_2 &= h\mathbf{f}(t_k + h, \mathbf{z}_k + k_1), \end{aligned}$$

here $\mathbf{z} = (P, p)$, $\mathbf{f}(t_k, \mathbf{z}_k) = (f_1(t_k, \mathbf{z}_k), f_2(t_k, \mathbf{z}_k))$.
$k_1 = (k_1(P), k_1(p))$ and $k_2 = (k_2(P), k_2(p))$.

Here $h = 0.1$ and starting from $t = 0$.
For $P^{'} = p = f_1(t, P, p)$ and $p^{'} = 350 - 50P - 12p = f_2(t, P, p)$, we start with
Iteration 1

$$\begin{aligned} P_1 &= P_0 + \frac{k_1(P) + k_2(P)}{2}, \\ p_1 &= p_0 + \frac{k_1(p) + k_2(p)}{2}, \end{aligned}$$

here

$$\begin{aligned} k_1(P) &= hf_1(t_0, P_0, p_0) = h(p_0) = 0.1(2) = 0.2, \\ k_1(p) &= hf_2(t_0, P_0, p_0) = h[350 - 50(2) - 12(2)] = 0.1[226] = 22.6, \\ k_2(P) &= hf_1(t_0 + h, P_0 + k_1(P), p_0 + k_1(p)) = h(2 + 22.6) = 0.1(24.6) = 2.46, \\ k_2(p) &= hf_2(t_0 + h, P_0 + k_1(P), p_0 + k_1(p)) = h[350 - 50(2 + 0.2) - 12(2 + 22.6)] \\ &= 0.1[350 - 110 - 295.2] = 0.1[-55.2] = -5.52, \end{aligned}$$

therefore we get

$$\begin{aligned} P_1 &= 2 + \frac{0.2 + 2.46}{2} \\ &= 2 + \frac{2.66}{2} = 2 + 1.33 = 3.33, \end{aligned}$$

and

$$\begin{aligned} p_1 &= 2 + \frac{22.6 + (-5.52)}{2} \\ &= 2 + \frac{17.08}{2} = 2 + 8.54 = 10.54 \end{aligned}$$

Therefore after ist iteration $\mathbf{z}_1 = (P_1, p_1) = (3.33, 10.54)$.
Iteration 2

$$\begin{aligned} P_2 &= P_1 + \frac{k_1(P) + k_2(P)}{2}, \\ p_2 &= p_1 + \frac{k_1(p) + k_2(p)}{2}, \end{aligned}$$

$$\begin{aligned} k_1(P) &= hf_1(t_1, P_1, p_1) = h(p_1) = 0.1(10.54) = 1.054, \\ k_1(p) &= hf_2(t_1, P_1, p_1) = h[350 - 50(3.33) - 12(10.54)] = 0.1[57.02] = 5.702, \\ k_2(P) &= hf_1(t_1 + h, P_1 + k_1(P), p_1 + k_1(p)) = h(10.54 + 5.702) = 0.1(16.24) = 1.624, \\ k_2(p) &= hf_2(t_1 + h, P_1 + k_1(P), p_1 + k_1(p)) = h[350 - 50(3.33 + 1.054) - 12(10.54 + 5.702)] \\ &= 0.1[350 - 219.2 - 194.88] = 0.1[-64.08] = -6.408, \end{aligned}$$

therefore we get

$$\begin{aligned} P_2 &= 3.33 + \frac{1.054 + 1.624}{2} \\ &= 3.33 + \frac{2.678}{2} = 3.33 + 1.339 = 4.67, \end{aligned}$$

and

$$\begin{aligned} p_2 &= 10.54 + \frac{5.702 + (-6.408)}{2} \\ &= 10.54 + \frac{-0.706}{2} = 10.54 - 0.353 = 10.18 \end{aligned}$$

Therefore after 2nd iteration $\mathbf{z}_2 = (P_2, p_2) = (4.67, 10.18)$.
Continuing in this way we get
$\mathbf{z}_3 = (P_3, p_3) = (5.66, 7.4)$.
$\mathbf{z}_4 = (P_4, p_4) = (6.29, 4.68)$.
$\mathbf{z}_5 = (P_5, p_5) = (6.65, 2.68)$.
$\mathbf{z}_6 = (P_6, p_6) = (6.85, 1.42)$.
$\mathbf{z}_7 = (P_7, p_7) = (6.94, 0.68)$.
$\mathbf{z}_8 = (P_8, p_8) = (6.98, 0.27)$.
$\mathbf{z}_9 = (P_9, p_9) = (6.99, 0.11)$.
Therefore we get the approximate solution as $P(t) \approx 7$.
Applying **Runge-Kutta method of fourth-order**

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6},$$

where

$$\begin{aligned} k_1 &= h\mathbf{f}(t_k, \mathbf{z}_k), \\ k_2 &= h\mathbf{f}(t_k + h/2, \mathbf{z}_k + k_1/2), \\ k_3 &= h\mathbf{f}(t_k + h/2, \mathbf{z}_k + k_2/2), \\ k_4 &= h\mathbf{f}(t_k + h, \mathbf{z}_k + k_3). \end{aligned}$$

Here $\mathbf{z} = (P, p)$, $\mathbf{f}(t_k, \mathbf{z}_k) = (f_1(t_k, \mathbf{z}_k), f_2(t_k, \mathbf{z}_k))$.
$k_1 = (k_1(P), k_1(p))$, $k_2 = (k_2(P), k_2(p))$, $k_3 = (k_3(P), k_3(p))$, $k_4 = (k_4(P), k_4(p))$.
Here $h = 0.1$ and starting from $t = 0$.
For $P^{'} = p = f_1(t, P, p)$ and $p^{'} = 350 - 50P - 12p = f_2(t, P, p)$, we start with
Iteration 1

$$\begin{aligned} P_1 &= P_0 + \frac{k_1(P) + 2k_2(P) + 2k_(P) + k_4(P)}{6}, \\ p_1 &= p_0 + \frac{k_1(p) + 2k_2(p) + 2k_(p) + k_4(p)}{6}, \end{aligned}$$

here

$$\begin{aligned}
k_1(P) &= hf_1(t_0, P_0, p_0) = h(p_0) = 0.1(2) = 0.2,\\
k_1(p) &= hf_2(t_0, P_0, p_0) = h[350 - 50(2) - 12(2)] = 0.1[226] = 22.6,\\
k_2(P) &= hf_1(t_0 + h/2, P_0 + k_1(P)/2, p_0 + k_1(p)/2) = h(p_0 + k_1(p)/2)\\
&= 0.1(2 + 22.6/2) = 0.1(13.3) = 1.33,\\
k_2(p) &= hf_2(t_0 + h/2, P_0 + k_1(P)/2, p_0 + k_1(p)/2) = h[350 - 50(2 + 0.2/2) - 12(2 + 22.6/2)]\\
&= 0.1[350 - 105 - 159.6] = 0.1[85.4] = 8.54,\\
k_3(P) &= hf_1(t_0 + h/2, P_0 + k_2(P)/2, p_0 + k_2(p)/2) = h(p_0 + k_2(p)/2)\\
&= 0.1(2 + 8.54/2) = 0.1(6.27) = 0.627,\\
k_3(p) &= hf_2(t_0 + h/2, P_0 + k_2(P)/2, p_0 + k_2(p)/2) = h[350 - 50(2 + 1.33/2) - 12(2 + 8.54/2)]\\
&= 0.1[350 - 133.25 - 75.24] = 0.1[141.51] = 14.51,\\
k_4(P) &= hf_1(t_0 + h, P_0 + k_3(P), p_0 + k_3(p)) = h(p_0 + k_3(p))\\
&= 0.1(2 + 14.51) = 0.1(16.51) = 1.651,\\
k_4(p) &= hf_2(t_0 + h, P_0 + k_3(P), p_0 + k_3(p)) = h[350 - 50(2 + 0.627) - 12(2 + 1.651)]\\
&= 0.1[350 - 131.35 - 43.8] = 0.1[174.85] = 17.85,
\end{aligned}$$

therefore we get

$$\begin{aligned}
P_1 &= 2 + \frac{0.2 + 2(1.33) + 2(0.627) + 1.651}{6}\\
&= 2 + \frac{5.765}{6} = 2 + 0.96 = 2.96,
\end{aligned}$$

and

$$\begin{aligned}
p_1 &= 2 + \frac{22.6 + 2(8.54) + 2(14.51) + 17.85}{6}\\
&= 2 + \frac{86.55}{6} = 2 + 14.425 = 16.425
\end{aligned}$$

Therefore after ist iteration $\mathbf{z}_1 = (P_1, p_1) = (2.96, 16.425)$.

Iteration 2

$$\begin{aligned}
P_2 &= P_1 + \frac{k_1(P) + 2k_2(P) + 2k_(P) + k_4(P)}{6},\\
p_2 &= p_1 + \frac{k_1(p) + 2k_2(p) + 2k_(p) + k_4(p)}{6},
\end{aligned}$$

here

$$\begin{aligned}
k_1(P) &= hf_1(t_1, P_1, p_1) = h(p_1) = 0.1(16.425) = 1.6425,\\
k_1(p) &= hf_2(t_1, P_1, p_1) = h[350 - 50(2.96) - 12(16.425)] = 0.1[4.9] = 0.49,\\
k_2(P) &= hf_1(t_1 + h/2, P_1 + k_1(P)/2, p_1 + k_1(p)/2) = h(p_1 + k_1(p)/2)\\
&= 0.1(16.425 + 0.49/2) = 0.1(16.67) = 1.667,\\
k_2(p) &= hf_2(t_1 + h/2, P_1 + k_1(P)/2, p_1 + k_1(p)/2) = h[350 - 50(2.96 + 1.6425/2) - 12(16.425 + 0.49/2)]\\
&= 0.1[350 - 189 - 200.4] = 0.1[-39.04] = -3.9,\\
k_3(P) &= hf_1(t_1 + h/2, P_1 + k_2(P)/2, p_1 + k_2(p)/2) = h(p_1 + k_2(p)/2)\\
&= 0.1(16.425 - 3.9/2) = 0.1(14.475) = 1.447,\\
k_3(p) &= hf_2(t_1 + h/2, P_1 + k_2(P)/2, p_1 + k_2(p)/2) = h[350 - 50(2.96 + 1.667/2) - 12(16.425 - 3.9/2)]\\
&= 0.1[350 - 189.5 - 173.7] = 0.1[-13.2] = -1.32,\\
k_4(P) &= hf_1(t_1 + h, P_1 + k_3(P), p_1 + k_3(p)) = h(p_1 + k_3(p))\\
&= 0.1(16.425 - 1.32) = 0.1(15.105) = 1.51,\\
k_4(p) &= hf_2(t_1 + h, P_1 + k_3(P), p_1 + k_3(p)) = h[350 - 50(2.96 + 1.447) - 12(16.425 - 1.32)]\\
&= 0.1[350 - 220.35 - 181.26] = 0.1[-51.61] = -5.161,
\end{aligned}$$

therefore we get

$$\begin{aligned}
P_2 &= 2.96 + \frac{1.6425 + 2(1.667) + 2(1.447) + 1.51}{6}\\
&= 2.96 + \frac{9.3805}{6} = 2.96 + 1.5634 = 4.52,
\end{aligned}$$

and

$$\begin{aligned}
p_2 &= 16.425 + \frac{0.49 + 2(-3.9) + 2(-1.32) - 5.161}{6}\\
&= 16.425 + \frac{-15.111}{6} = 16.425 - 2.5185 = 13.9
\end{aligned}$$

Therefore after 2nd iteration $\mathbf{z}_2 = (P_2, p_2) = (4.52, 13.9)$.
Continuing in this way we get,
$\mathbf{z}_3 = (P_3, p_3) = (5.68, 9.26)$.
$\mathbf{z}_4 = (P_4, p_4) = (6.4, 5.31)$.
$\mathbf{z}_5 = (P_5, p_5) = (6.786, 2.613)$.
$\mathbf{z}_6 = (P_6, p_6) = (7.01, 0.84)$.
Therefore we get the approximate solution as $P(t) \approx 7$.

From all the methods the solution of differential equation of pricing policy model i.e price of brick $P(t) \longrightarrow 7$ as $t \longrightarrow \infty$.

Implementation in MATLAB

Results are obtained for 12 iterations

Euler's method

poleu.m

```
h=0.1;
iter = 12;
pk(1) = 2; p_k(1)=2; t(1) =0;
A =350; B=50; C=12;
for i= 2:iter
    dpk = p_k(i-1);
    dp_k = A - B*pk(i-1) -12*p_k(i-1);
    pk(i) = pk(i-1) + h*dpk;
    p_k(i) = p_k(i-1) + h*dp_k;
    t(i) =t(i-1) + h;
end
```

Output

p_k × pk ×

1×12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	2	2.2000	4.6600	6.5680	7.3564	7.4147	7.2249	7.0555	6.9769	6.9649	6.9788	6.9936

p_k × pk ×

1×12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	2	24.6000	19.0800	7.8840	0.5832	-1.8986	-1.6939	-0.7855	-0.1202	0.1395	0.1476	0.0763

Improved Euler's method

poleu.m polimp.m

```
h=0.1;
iter = 12;
P(1) = 2; p_(1)=2; t(1) =0;
A =350; B=50; C=12;
for i= 2:iter
    dP = p_(i-1);
    dp_ = A - B*P(i-1) -C*p_(i-1);
    H(i) = P(i-1) + (0.5*h)*dP;
    J(i) = p_(i-1) + (0.5*h)*dp_;
    dP(i) = J(i);
    dp_(i) = A - B*H(i) -C*J(i);
    P(i) = P(i-1) + h*dP(i);
    p_(i) = p_(i-1) + h*dp_(i);
    t(i) =t(i-1) +h;
end
```

Output

p_ × P ×

1×12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	2	3.3300	4.6691	5.6593	6.2909	6.6555	6.8489	6.9432	6.9848	7.0005	7.0048	7.0048

Pri × pri × p_ ×

1×12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	2	10.5400	10.1858	7.4120	4.6827	2.6825	1.4133	0.6837	0.2982	0.1110	0.0290	-0.0018

Second order Runge-Kutta method

poleu.m | polimp.m | polru2.m

```
h=0.1;
iter =12;
Pr(1)=2; pr(1)=2; t(1) =0;
A =350; B=50; C=12;
for i=2:iter
    dPr = pr(i-1);
    dpr = A - B*Pr(i-1) -C*pr(i-1);
    k1Pr(i) = h*dPr ; k1pr(i) = h*dpr;
    k2Pr(i) = h*( pr(i-1)  + k1pr(i));
    k2pr(i) = h*(A -B*(Pr(i-1)+ k1Pr(i)) - C*(pr(i-1)  + k1pr(i)));
    Pr(i) = Pr(i-1) + (k1Pr(i) + k2Pr(i))*(1/2);
    pr(i) = pr(i-1) + (k1pr(i) + k2pr(i))*(1/2);
    t(i) = t(i-1) +h;
end
```

Output

Pr | pr

1×12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	2	3.3300	4.6691	5.6593	6.2909	6.6555	6.8489	6.9432	6.9848	7.0005	7.0048	7.0048

Pr | pr

1×12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	2	10.5400	10.1858	7.4120	4.6827	2.6825	1.4133	0.6837	0.2982	0.1110	0.0290	-0.0018

Fourth order Runge-Kutta method

poleu.m | polimp.m | polru2.m | polru4.m

```
Pri(1) =2; pri(1)=2; t(1) = 0;
A=350; B=50; C=12;
for i =2:iter
    dPri = pri(i-1);
    dpri = A -B*Pri(i-1) -C*pri(i-1);
    k1Pri(i) = h*dPri; k1pri(i) = h*dpri;
    k2Pri(i) = h*(pri(i-1)+ 0.5*k1pri(i));
    k2pri(i) = h*(A - B*(Pri(i-1) + 0.5*k1Pri(i)) - C*(pri(i-1)+ 0.5*k1pri(i)));
    k3Pri(i) = h*(pri(i-1)+ 0.5*k2pri(i));
    k3pri(i) = h*(A - B*(Pri(i-1) + 0.5*k2Pri(i)) - C*(pri(i-1)+ 0.5*k2pri(i)));
    k4Pri(i) = h*(pri(i-1)+ k3pri(i));
    k4pri(i) = h*(A - B*(Pri(i-1) + k3Pri(i)) - C*(pri(i-1)+ k3pri(i)));
    Pri(i) = Pri(i-1) + (k1Pri(i) + 2*(k2Pri(i)+k3Pri(i)) + k4Pri(i))*(1/6);
    pri(i) = pri(i-1) + (k1pri(i) + 2*(k2pri(i)+k3pri(i)) + k4pri(i))*(1/6);
    t(i) = t(i-1) +h;
end
```

Output

Pri | pri

1×12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	2	2.9548	4.3757	5.5377	6.2961	6.7209	6.9265	7.0087	7.0309	7.0289	7.0202	7.0119

Pri | pri

1×12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	2	13.7443	13.4092	9.5475	5.7120	2.9602	1.3050	0.4429	0.0602	-0.0713	-0.0908	-0.0711

3.1.5 Comparison of analytic and numerical solution

Differential equation of pricing policy is given as

$$\frac{d^2P}{dt^2} + 12k(\frac{dP}{dt}) + 50kP = 350k,$$

where k is constant of proportionality which is positive in nature.
We find the conditions under which the solution of above second-order differential equation tend to a limit as t becomes larger. Through inspection $P = 7$, is particular solution of above differential equation . To have general solution, we consider characteristic polynomial

$$\lambda^2 + 12k\lambda + 50k,$$

which has roots $\lambda = -6 \pm \sqrt{36k^2 - 50k}$. Since $k > 0$. When $18k > 25$ roots are real, distinct and negative since $\sqrt{36k^2 - 50k} < 6k$. Therefore general solution is

$$P(t) = Ae^{\lambda_1 t} + Be^{\lambda_2(t)} + 7,$$

where $\lambda_1, \lambda_2 < 0$. Thus the solution $P(t)$ declines exponentially to fixed level $P(t) = 7$ and making the pricing policy is stable.
From the numerical methods we get approximately the same result i.e $P(t) \to 7$ as $t \to \infty$ which again makes the pricing policy stable. Results obtained from analytic and numerical method are almost same. Therefore numerical approach for pricing policy is as helpful as analytic approach as both yields almost same result.

3.2 PROBLEM 2 : THE RÖSSLER SYSTEM

The Rössler attractor is a attarctor for Rössler system which itself is non-linear ordinary differential equation's system containg three equations. This system was studied by Otto Rössler. This system arose from work in chemical kinetics. The system exhibits continuous-time chaos. In dynamical system's terminology, an **attractor** is a group of numerical values around which the system seems to evolve with variety of initial conditions. An attractor is a point or it can be a finite set of points or it can be a curve, or manifold, or even a complicated set along with a fractal structure is referred as a strange attractor. **Chaos theory** is a mathematical branch which studying the dynamical system's nature which are highly sensitive to initial conditions. An interdisciplinary theory **Chaos** says that within the apparent randomness of chaotic complex systems, there is presence of underlying patterns, constant feedback loops, repetition, self-similarity, fractals, self-organization and reliance on programming at the initial point known as sensitive dependence on initial conditions.
Edward N. Lorenz led to an autonomous three-dimensional system which is nonlinear in nature that is described as

$$\begin{aligned} \frac{dp}{dt} &= \sigma(-p+q), \\ \frac{dq}{dt} &= rp - q - ps, \\ \frac{ds}{dt} &= -bs + pq. \end{aligned}$$

this is commonly known as **Lorenz equations**. The variable p in above system is depicts the fluid motion's intensity, variables q and s shows temperature variations in the horizontal and vertical directions. These equations involve three parameters σ, r and b all real and positive. The parameters σ and b are dependent of material and geometrical properties of the fluid layer.
The Rössler system defined as

$$\begin{aligned} \frac{dp}{dt} &= -q - r, \\ \frac{dq}{dt} &= p + aq, \\ \frac{dr}{dt} &= b + r(p - c), \end{aligned}$$

is a form of **Lorenz equations** with $a,\ b$ and c as three positive parameters. It is comparatively easier system that involves two linear equations and third one as quadratic nonlinear equation.
We have to perform some numerical process on this system, aiming at system's period-doubling property(i.e a small change in any of parameter value in the system's equations resulting the system to switch to a new behavior with two times the period of the original system.)

3.2.1 Solving numerically the Rössler system

Considering the Rössler system i.e

$$\begin{aligned} \frac{dp}{dt} &= -q - r, \\ \frac{dq}{dt} &= p + aq, \\ \frac{dr}{dt} &= b + r(p - c), \end{aligned}$$

with the parameteres $a = 0.25,\ b = 0.5$ and $c = 1$ i.e system is

$$\begin{aligned} \frac{dp}{dt} &= -q - r, \\ \frac{dq}{dt} &= p + 0.25q, \\ \frac{dr}{dt} &= 0.5 + r(p - 1), \end{aligned}$$

with initial points as $(1, 1, 1)$ and step size as $h = 0.1$.
Applying the numerical methods on above system to get it's solution's approximation.

3.2.2 Solving Rössler system with numerical methods

Euler's method

Applying Euler's method

$$\mathbf{z}_{k+1} = \mathbf{z}_k + h\mathbf{f}(t_k, \mathbf{z}_k), \quad k = 0, 1, 2, ..., n - 1,$$

here $\mathbf{z} = (p, q, r)$ and $\mathbf{f} = (f_1, f_2, f_3) = (\frac{dp}{dt}, \frac{dq}{dt}, \frac{dr}{dt}) = (-q - r, p + 0.25q, 0.5 + r(p - 1))$.
From $t = 0$,we start with
Iteration 1
For $k = 0$

$$\mathbf{z}_1 = \mathbf{z}_0 + h\mathbf{f}(t_0, \mathbf{z}_0),$$

$$\begin{aligned} p_1 &= p_0 + hf_1(t_0, \mathbf{z}_0) = 1 + 0.1(-2) = 1 - 0.2 = 0.8, \\ q_1 &= q_0 + hf_2(t_0, \mathbf{z}_0) = 1 + 0.1(1 + 0.25) = 1 + 0.125 = 1.125, \\ r_1 &= r_0 + hf_3(t_0, \mathbf{z}_0) = 1 + 0.1(0.5 + 0) = 1 + 0.05 = 1.05 \end{aligned}$$

Therefore after ist iteration $\mathbf{z}_1 = (p_1, q_1, r_1) = (0.8, 1.125, 1.05)$.

Iteration 2

For $k = 1$

$$\mathbf{z}_2 = \mathbf{z}_1 + h\mathbf{f}(t_1, \mathbf{z}_1),$$

$$\begin{aligned} p_2 &= p_1 + hf_1(t_1, \mathbf{z}_1) = 0.8 + 0.1(-1.125 - 1.05) = 0.5825, \\ q_2 &= q_1 + hf_2(t_1, \mathbf{z}_1) = 1.125 + 0.1(0.8 + 0.25(1.125)) = 1.233, \\ r_2 &= r_1 + hf_3(t_1, \mathbf{z}_1) = 1.05 + 0.1(0.5 + 1.05(0.8 - 1)) = 1.079 \end{aligned}$$

Therefore after 2nd iteration $\mathbf{z}_2 = (p_2, q_2, r_2) = (0.5825, 1.233, 1.079)$.
Continuing in same manner we get
$\mathbf{z}_3 = (p_3, q_3, r_3) = (0.3513, 1.322, 1.084)$.
$\mathbf{z}_4 = (p_4, q_4, r_4) = (0.1107, 1.39, 1.0636)$.
$\mathbf{z}_5 = (p_5, q_5, r_5) = (-0.134, 1.435, 1.019)$ and so on.

Modified Euler's method

Applying Euler's midpoint method

$$\begin{aligned} \mathbf{z}_{k+1/2} &= \mathbf{z}_k + \frac{h}{2}\mathbf{f}(t_k, \mathbf{z}_k), \quad k = 0, 1, 2, ..., n-1, \\ \mathbf{z}_{k+1} &= \mathbf{z}_k + h\mathbf{f}(t_k + h/2, \mathbf{z}_{k+1/2}). \end{aligned}$$

Here $\mathbf{z} = (p, q, r)$ and $\mathbf{f} = (f_1, f_2, f_3) = (\frac{dp}{dt}, \frac{dq}{dt}, \frac{dr}{dt}) = (-q - r, p + 0.25q, 0.5 + r(p - 1))$.
From $t = 0$, we start with
Iteration 1

$$\begin{aligned} \mathbf{z}_{1/2} &= \mathbf{z}_0 + \frac{h}{2}\mathbf{f}(t_0, \mathbf{z}_0), \\ (p_{1/2}, q_{1/2}, r_{1/2}) &= (p_0, q_0, r_0) + \frac{h}{2}(f_1(t_0, p_0, q_0, r_0), f_2(t_0, p_0, q_0, r_0), f_3(t_0, p_0, q_0, r_0)). \end{aligned}$$

$$\begin{aligned} p_{1/2} &= p_0 + \frac{0.1}{2}(-q_0 - r_0) \\ &= 1 + 0.05(-1 - 1) = 1 - 0.1 = 0.9, \\ q_{1/2} &= q_0 + \frac{0.1}{2}(p_0 + 0.25q_0) \\ &= 1 + 0.05(1 + 0.25) = 1 + 0.0625 = 1.0625, \\ r_{1/2} &= r_0 + \frac{0.1}{2}(0.5 + r_0(p_0 - 1)) \\ &= 1 + 0.05(0.5 + 1(1 - 1)) = 1 + 0.025 = 1.025, \end{aligned}$$

therefore

$$\mathbf{z}_1 = \mathbf{z}_0 + h\mathbf{f}(t_0 + h/2, \mathbf{z}_{1/2}).$$

$$\begin{aligned} p_1 &= p_0 + 0.1(-q_{1/2} - r_{1/2}) \\ &= 1 + 0.1(-1.0625 - 1.025) = 0.7912, \\ q_1 &= q_0 + 0.1(p_{1/2} + 0.25(q_{1/2})) \\ &= 1 + 0.1(0.9 + 0.25(1.0625)) = 1.117, \\ r_1 &= r_0 + 0.1(0.5 + r_{1/2}(p_{1/2} - 1)) \\ &= 1 + 0.1(0.5 + 1.025(0.9 - 1)) = 1.039 \end{aligned}$$

Therefore after ist iteration $\mathbf{z}_1 = (p_1, q_1, r_1) = (0.7912, 1.117, 1.039)$.

Iteration 2

$$\begin{aligned} \mathbf{z}_{1+1/2} &= \mathbf{z}_1 + \frac{h}{2}\mathbf{f}(t_1, \mathbf{z}_1), \\ (p_{1+1/2}, q_{1+1/2}, r_{1+1/2}) &= (p_1, q_1, r_1) + \frac{h}{2}(f_1(t_1, p_1, q_1, r_1), f_2(t_1, p_1, q_1, r_1), f_3(t_1, p_1, q_1, r_1)). \end{aligned}$$

$$\begin{aligned} p_{1+1/2} &= p_1 + \frac{0.1}{2}(-q_1 - r_1) \\ &= 0.7912 + 0.05(-1.116 - 1.039) = 0.6834, \\ q_{1+1/2} &= q_1 + \frac{0.1}{2}(p_1 + 0.25(q_1)) \\ &= 1.116 + 0.05(0.6834 + 0.25(1.116)) = 1.1641, \\ r_{1+1/2} &= r_1 + \frac{0.1}{2}(0.5 + r_1(p_1 - 1)) \\ &= 1.039 + 0.05(0.5 + 1.039(0.7912 - 1)) = 1.053, \end{aligned}$$

therefore

$$\mathbf{z}_2 = \mathbf{z}_1 + h\mathbf{f}(t_1 + h/2, \mathbf{z}_{1+1/2}).$$

$$\begin{aligned} p_2 &= p_1 + 0.1(-q_{1+1/2} - r_{1+1/2}) \\ &= 0.7912 + 0.1(-1.1641 - 1.053) = 0.569; \\ q_2 &= q_1 + 0.1(p_{1+1/2} + 0.25(q_{1+1/2})) \\ &= 1.116 + 0.1(0.6834 + 0.25(1.1641)) = 1.213, \\ r_2 &= r_1 + 0.1(0.5 + r_{1+1/2}(p_{1+1/2} - 1)) \\ &= 1.039 + 0.1(0.5 + 1.053(0.6834 - 1)) = 1.056 \end{aligned}$$

Therefore after 2nd iteration $\mathbf{z}_2 = (p_2, q_2, r_2) = (0.569, 1.213, 1.056)$.
Continuing in same manner we get
$\mathbf{z}_3 = (p_3, q_3, r_3) = (0.337, 1.278, 1.048)$.
$\mathbf{z}_4 = (p_4, q_4, r_4) = (0.1022, 1.332, 1.0171)$.
$\mathbf{z}_5 = (p_5, q_5, r_5) = (-0.1327, 1.364, 0.966)$ and so on.

Runge-Kutta method

Applying **Runge-Kutta method of second-order**

$$\begin{aligned} \mathbf{z}_{k+1} &= \mathbf{z}_k + (k_1 + k_2)/2, \\ k_1 &= h\mathbf{f}(t_k, \mathbf{z}_k), \\ k_2 &= h\mathbf{f}(t_k + h, \mathbf{z}_k + k_1). \end{aligned}$$

Here $\mathbf{z} = (p, q, r)$ and $\mathbf{f} = (f_1, f_2, f_3) = (\frac{dp}{dt}, \frac{dq}{dt}, \frac{dr}{dt}) = (-q - r, p + 0.25q, 0.5 + r(p - 1))$.
From $t = 0$, we start with
Iteration 1

$$\begin{aligned} \mathbf{z}_1 &= \mathbf{z}_0 + (k_1 + k_2)/2, \\ k_1 &= h\mathbf{f}(t_0, \mathbf{z}_0,) \\ k_2 &= h\mathbf{f}(t_0 + h, \mathbf{z}_0 + k_1). \\ (p_1, q_1, r_1) &= (p_0, q_0, r_0) + (k_1 + k_2)/2, \end{aligned}$$

where $k_1 = (k_1(p), k_1(q), k_1(r))$ and $k_2 = (k_2(p), k_2(q), k_2(r))$. Therefore

$$\begin{aligned} k_1(p) &= hf_1(t_0, p_0, q_0, r_0) = 0.1(-1 - 1) = -0.2, \\ k_1(q) &= hf_2(t_0, p_0, q_0, r_0) = 0.1(1 + 0.25(1)) = 0.125, \\ k_1(r) &= hf_3(t_0, p_0, q_0, r_0) = 0.1(0.5 + 1(1 - 1)) = 0.005, \\ k_2(p) &= hf_1(t_0 + h, p_0 + k_1(p), q_0 + k_1(q), r_0 + k_1(r)) = 0.1(-(1 + 0.125) - (1 + 0.05)) = -0.2175, \\ k_2(q) &= hf_2(t_0 + h, p_0 + k_1(p), q_0 + k_1(q), r_0 + k_1(r)) = 0.1((1 - 0.2) + 0.25(1 + 0.125)) = 0.108, \\ k_2(r) &= hf_3(t_0 + h, p_0 + k_1(p), q_0 + k_1(q), r_0 + k_1(r)) = 0.1(0.5 + (1 + 0.05)(1 - 0.2 - 1) = 0.029, \end{aligned}$$

$$\begin{aligned} p_1 &= p_0 + \frac{k_1(p) + k_2(p)}{2} \\ &= 1 + \frac{-0.2 - 0.2175}{2} = 0.7912, \end{aligned}$$

$$\begin{aligned} q_1 &= q_0 + \frac{k_1(q) + k_2(q)}{2} \\ &= 1 + \frac{0.125 + 0.108}{2} = 1.116, \\ r_1 &= r_0 + \frac{k_1(r) + k_2(r)}{2} \\ &= 1 + \frac{0.05 + 0.029}{2} = 1.039 \end{aligned}$$

Therefore after ist iteration $\mathbf{z}_1 = (p_1, q_1, r_1) = (0.7912, 1.116, 1.039)$.
Continuing in same way we get
$\mathbf{z}_2 = (p_2, q_2, r_2) = (0.568, 1.213, 1.056)$.
$\mathbf{z}_3 = (p_3, q_3, r_3) = (0.337, 1.278, 1.048)$.
$\mathbf{z}_4 = (p_4, q_4, r_4) = (0.1022, 1.332, 1.0171)$.
$\mathbf{z}_5 = (p_5, q_5, r_5) = (-0.1327, 1.364, 0.966)$ and so on.

Applying above method

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6},$$

where

$$\begin{aligned} k_1 &= h\mathbf{f}(t_k, \mathbf{z}_k), \\ k_2 &= h\mathbf{f}(t_k + h/2, \mathbf{z}_k + k_1/2), \\ k_3 &= h\mathbf{f}(t_k + h/2, \mathbf{z}_k + k_2/2), \\ k_4 &= h\mathbf{f}(t_k + h, \mathbf{z}_k + k_3). \end{aligned}$$

Here $\mathbf{z} = (p, q, r)$ and $\mathbf{f} = (f_1, f_2, f_3) = (\frac{dp}{dt}, \frac{dq}{dt}, \frac{dr}{dt}) = (-q - r, p + 0.25q, 0.5 + r(p - 1))$.
From $t = 0$, we start with
Iteration 1

$$\mathbf{z}_1 = \mathbf{z}_0 + \frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6},$$

where $k_1 = (k_1(p), k_1(q), k_1(r))$, $k_2 = (k_2(p), k_2(q), k_2(r))$, $k_3 = (k_3(p), k_3(q), k_3(r))$ *and* $k_4 = (k_4(p), k_4(q), k_4(r))$.
Therefore

$$\begin{aligned} k_1(p) &= hf_1(t_0, p_0, q_0, r_0) = 0.1(-1 - 1) = -0.2, \\ k_1(q) &= hf_2(t_0, p_0, q_0, r_0) = 0.1(1 + 0.25(1)) = 0.125, \\ k_1(r) &= hf_3(t_0, p_0, q_0, r_0) = 0.1(0.5 + 1(1 - 1)) = 0.05, \end{aligned}$$

$$
\begin{aligned}
k_2(p) &= hf_1(t_0+h/2, p_0+k_1(p)/2, q_0+k_1(q)/2, r_0+k_1(r)/2) = 0.1(-(1+0.125/2)-(1+0.05/2)) \\
&= -0.20875, \\
k_2(q) &= hf_2(t_0+h/2, p_0+k_1(p)/2, q_0+k_1(q)/2, r_0+k_1(r)/2) = 0.1((1-0.2/2)+0.25(1+0.125/2)) \\
&= 0.1166, \\
k_2(r) &= hf_3(t_0+h/2, p_0+k_1(p)/2, q_0+k_1(q)/2, r_0+k_1(r)/2) = 0.1[0.5+(1+0.05/2)((1-0.2/2)-1)] \\
&= 0.03975, \\
k_3(p) &= hf_1(t_0+h/2, p_0+k_2(p)/2, q_0+k_2(q)/2, r_0+k_2(r)/2) = 0.1(-(1+0.1166/2)-(1+0.03975/2)) \\
&= -0.2078, \\
k_3(q) &= hf_2(t_0+h/2, p_0+k_2(p)/2, q_0+k_2(q)/2, r_0+k_2(r)/2) = 0.1((1-0.20875/2)+0.25(1+0.1166/2)) \\
&= 0.116, \\
k_3(r) &= hf_3(t_0+h/2, p_0+k_2(p)/2, q_0+k_2(q)/2, r_0+k_2(r)/2) = 0.1[0.5+(1+0.03975/2)(1-0.20875/2-1)] \\
&= 0.03935, \\
k_4(p) &= hf_1(t_0+h, p_0+k_3(p), q_0+k_3(q), r_0+k_3(r)) = 0.1(-(1+0.116)-(1+0.03935)) \\
&= -0.2155, \\
k_4(q) &= hf_2(t_0+h, p_0+k_3(p), q_0+k_3(q), r_0+k_3(r)) = 0.1((1-0.2078)+0.25(1+0.116)) \\
&= 0.107, \\
k_4(r) &= hf_3(t_0+h, p_0+k_3(p), q_0+k_3(q), r_0+k_3(r)) = 0.1[0.5+(1+0.03935)(1-0.2078-1)] \\
&= 0.0284,
\end{aligned}
$$

$$
\begin{aligned}
p_1 &= p_0 + \frac{k_1(p)+2k_2(p)+2k_3(p)+k_4(p)}{6} \\
&= 1 + \frac{-0.2+2(-0.20875)+2(-0.2078)-0.2155}{6} = 0.7919, \\
q_1 &= q_0 + \frac{k_1(q)+2k_2(q)+2k_3(q)+k_4(q)}{6} \\
&= 1 + \frac{0.125+2(0.1166)+2(0.116)+0.107}{6} = 1.116, \\
r_1 &= r_0 + \frac{k_1(r)+2k_2(r)+2k_3(r)+k_4(r)}{6} \\
&= 1 + \frac{0.05+2(0.03975)+2(0.03935)+0.0284}{6} = 1.039
\end{aligned}
$$

Therefore after ist iteration $\mathbf{z}_1 = (p_1, q_1, r_1) = (0.7919, 1.116, 1.039)$.
Continuing in same way we get
$\mathbf{z}_2 = (p_2, q_2, r_2) = (0.57, 1.213, 1.055)$.
$\mathbf{z}_3 = (p_3, q_3, r_3) = (0.33, 1, 29, 1.04)$ and so on.

Implementation in MATLAB

Results are obtained for 20 iterations

Euler's method

ros.m

```
h=0.1;
iter=20;
x(1)=1; y(1)=1; z(1)=1; t(1)=0;
a=0.25; b=0.5; c=1;
for i=2:iter
    dx = -y(i-1) - z(i-1);
    dy = x(i-1) + a*y(i-1);
    dz = b + z(i-1)*(x(i-1) - 1);
    x(i) = x(i-1) + h*dx;
    y(i) = y(i-1) + h*dy;
    z(i) = z(i-1) + h*dz;
    t(i) = t(i-1) + h;

end

```

Output

MATLAB Drive

iter × x ×

1×20 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	0.8000	0.5825	0.3513	0.1107	-0.1347	-0.3803	-0.6215	-0.8544	-1.0755	-1.2823	-1.4725	-1.6446	-1.7975	-1.9302	-2.0420	-2.1324	-2.2006	-2.2460	-2.2682

MATLAB Drive

iter × x × y ×

1×20 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1.1250	1.2331	1.3222	1.3904	1.4362	1.4586	1.4571	1.4314	1.3817	1.3087	1.2132	1.0963	0.9592	0.8035	0.6305	0.4421	0.2399	0.0258	-0.1981

MATLAB Drive

iter × x × y × z ×

1×20 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1.0500	1.0790	1.0840	1.0636	1.0190	0.9534	0.8718	0.7805	0.6857	0.5934	0.5080	0.4324	0.3680	0.3151	0.2728	0.2398	0.2147	0.1960	0.1824

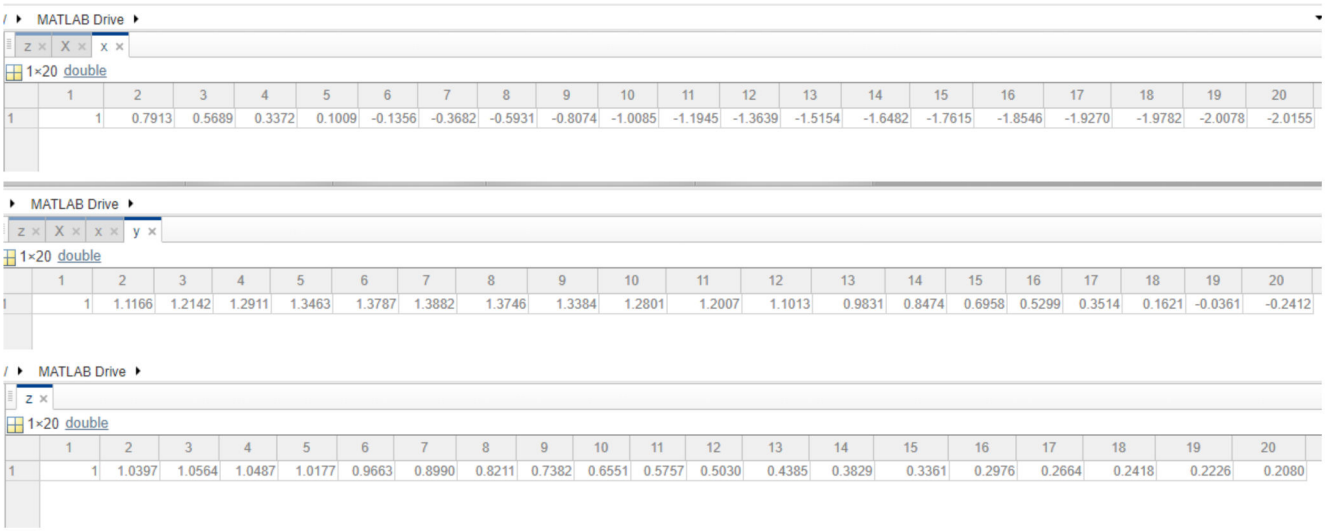
Improved Euler's method

ros.m mod.m

```
h=0.1;
iter=20;
x(1)=1; y(1)=1; z(1)=1; t(1)=0;
a=0.25; b=0.5; c=1;
for i=2:iter
    dx = -y(i-1) - z(i-1);
    dy = x(i-1) + a*y(i-1);
    dz = b + z(i-1)*(x(i-1) - c);
    X(i) = x(i-1) + (0.5*h)*dx;
    Y(i)  = y(i-1) + (0.5*h)*dy;
    Z(i) = z(i-1) + (0.5*h)*dz;
    dx(i) = -Y(i) - Z(i);
    dy(i) = X(i) + a*Y(i);
    dz(i) = b + Z(i)*(X(i )- c);
    x(i) = x(i-1) + h*dx(i);
    y(i) = y(i-1) + h*dy(i);
    z(i) = z(i-1) + h*dz(i);
    t(i) = t(i-1) + h;

end
```

Output

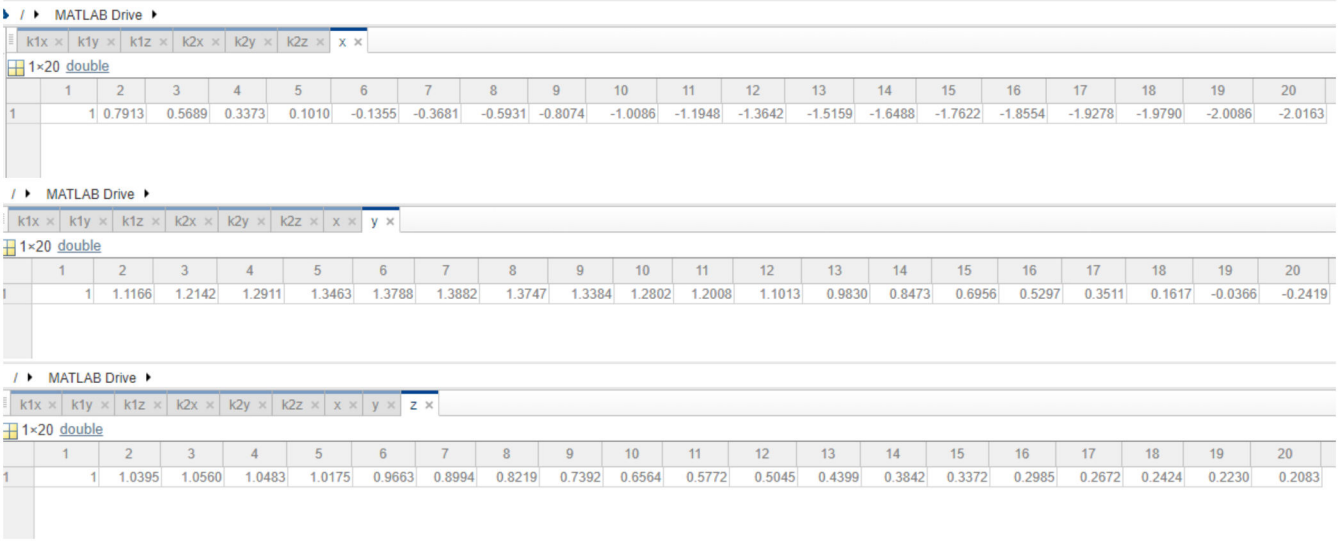


Second order Runge-Kutta method

ros.m × mod.m × ru2.m ×

```
h=0.1;
iter=5;
x(1)=3; y(1)=2; t(1)=0;
a=2; b=2; c=1; d=1;
for i=2:iter
    dx = a*x(i-1) - b*x(i-1)*y(i-1);
    dy = -c*y(i-1) + d*y(i-1)*x(i-1);
    k1x(i) = h*dx ; k1y(i) = h*dy;
    k2x(i) = h*(a*(x(i-1) + k1x(i)) - b*(x(i-1)+k1x(i))*(y(i-1)+k1y(i)));
    k2y(i) = h*(-c*(y(i-1)+k1y(i)) + d*(x(i-1)+k1x(i))*(y(i-1)+k1y(i)));
     x(i) = x(i-1) + (k1x(i) + k2x(i))*(1/2);
     y(i) = y(i-1) + (k1y(i) + k2y(i))*(1/2);
end
```

Output



Fourth order Runge-Kutta method

ros.m | mod.m | ru2.m | ru4.m | rung.m

```
h=0.1;
iter=20;
x(1)=1; y(1)=1; z(1)=1; t(1)=0;
a=0.25; b=0.5; c=1;
for i=2:iter
    dx = -y(i-1) - z(i-1);
    dy = x(i-1) + a*y(i-1);
    dz = b + z(i-1)*(x(i-1) - c);
    k1x(i) = h*dx ; k1y(i) = h*dy; k1z(i) =h*dz;
    k2x(i) =h*(-(y(i-1) + (0.5*k1y(i))) -(z(i-1) + (0.5*k1z(i))));
    k2y(i) = h*(x(i-1)+ (0.5*k1x(i)) + a*(y(i-1)+ (0.5*k1y(i))));
    k2z(i) = h*(b + (z(i-1)+ (0.5*k1z(i)))*(x(i-1) + (0.5*k1x(i)) -c));
    k3x(i) =h*(-(y(i-1) + (0.5*k2y(i))) -(z(i-1) + (0.5*k2z(i))));
    k3y(i) = h*(x(i-1)+ (0.5*k2x(i)) + a*(y(i-1)+ (0.5*k2y(i))));
    k3z(i) = h*(b + (z(i-1)+ (0.5*k2z(i)))*(x(i-1) + (0.5*k2x(i)) -c));
    k4x(i) =h*(-(y(i-1) + k3y(i)) -(z(i-1) + k3z(i)));
    k4y(i) = h*(x(i-1)+ (k3x(i)) + a*(y(i-1)+ (k3y(i))));
    k4z(i) = h*(b + (z(i-1)+ (k3z(i)))*(x(i-1) + (k3x(i)) -c));
    x(i) = x(i-1) + (k1x(i) + 2*(k2x(i)+k3x(i)) + k4x(i))*(1/6);
    y(i) = y(i-1) + (k1y(i) + 2*(k2y(i)+k3y(i)) + k4y(i))*(1/6);
    z(i) = z(i-1) + (k1z(i) + 2*(k2z(i)+k3z(i)) + k4z(i))*(1/6);

end
```

Output

MATLAB Drive

k1 | k1x | k1y | k1z | k2x | k2y | k2z | k3x | k3y | k3z | k4x | k4y | k4z | x

1×20 double

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0.7919	0.5703	0.3395	0.1039	-0.1319	-0.3639	-0.5885	-0.8027	-1.0038	-1.1900	-1.3597	-1.5116	-1.6448	-1.7585	-1.8521	-1.9251	-1.9769	-2.0071	-2.0156

MATLAB Drive

k1 | k1x | k1y | k1z | k2x | k2y | k2z | k3x | k3y | k3z | k4x | k4y | k4z | x | y

1×20 double

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1.1162	1.2136	1.2905	1.3457	1.3783	1.3881	1.3750	1.3393	1.2817	1.2030	1.1042	0.9867	0.8517	0.7008	0.5356	0.3578	0.1691	-0.0285	-0.2331

MATLAB Drive

k1 | k1x | k1y | k1z | k2x | k2y | k2z | k3x | k3y | k3z | k4x | k4y | k4z | x | y | z

1×20 double

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1.0394	1.0560	1.0486	1.0181	0.9674	0.9008	0.8235	0.7409	0.6579	0.5784	0.5053	0.4403	0.3842	0.3370	0.2981	0.2666	0.2417	0.2224	0.2077

For more accurate results more number of iterations would be required. Somehow numerical methods help in estimating the points for Rössler system. These methods can help in having an idea about Rössler system where the period-doubling property(i.e small variation in value of any of parameter in the system's equations resulting the system to switch to a new nature with double the period of the original system.) can be achieved. For Rössler system advanced non-linear methods such as Poincaré maps and bifurcation diagrams are used. Numerical methods with large number of iterations and high accuracy level give almost same results as those advanced methods.

3.3 Errors in Numerical approximations

The application of numerical approach in problems of initial value raises many questions that are mandatory to be answered before evaluating the approximate solution that is satisfactory to

accept. Among these questions one is of **convergence** i.e. with step size h tending to zero, values of the numerical approximation $z_1, z_2, ..., z_n$ does approach the respective values of exact solution? Further question arises- How small must be step size? Answer to it is step size chosen is supposed to be small enough to give the accuracy to its best, but should not be too small. An unnecessarily small step size reduces the speed of calculations and in some cases there is loss of accuracy. The main causes of errors in numerical approach are

(a) Programming blunder,

(b) Truncation error, i.e. inexact evaluation of mathematical operators,

(c) Roundoff errors, i.e. inexact arithmetic calculations.

In dealing with vectors, matrices and functions, the problem of measuring their exactness to a certain given quantity is usually done by the concept of norms.
With the assumption that computers are able to run all the computations exactly; i.e. at each step retaining infinitely many digits. Difference Er_n between the initial value problem's solution $z = \phi(t)$ and the numerical approximation z_n at the point $t = t_n$ is given as

$$Er_n = \phi(t_n) - z_n.$$

This error Er_n is **global truncation error** and error at each n^{th} step is **local truncation error** denoted as e_n. In actual we must compute approximations to the finite precision arithmetic i.e. at each step keeping only finite number of digits. Further it results in a round-off error R_n defined as

$$R_n = z_n - Z_n,$$

where Z_n is value obtained in numerical method. Total absolute value of error in computing $\phi(t_n)$ is given as

$$|\phi(t_n) - Z_n| = |\phi(t_n) - z_n + z_n - Z_n|.$$

By virtue of triangle's inequality i.e $|a + b| \leq |a| + |b|$, we get

$$\begin{aligned} |\phi(t_n) - Z_n| &\leq |\phi(t_n) - z_n| + |z_n - Z_n| \\ &\leq |Er_n| + |R_n|, \end{aligned}$$

this is absolute values' summation of the global truncation and round-off errors.

Error analysis of Euler's method
Suppose the $z = \phi(t)$ be solution of corresponding i.v.p with continuous second order derivative in the interval of choice. Assuming $f, f_t \; and \; f_z$ to be continuous.

$$\phi^{'}(t) = f[t, \phi(t)],$$

using chain rule,

$$\phi^{''}(t) = f_t[t, \phi(t)] + f_z[t, \phi(t)]\phi^{'}(t)$$
$$= f_t[t, \phi(t)] + f_z[t, \phi(t)]f[t, \phi(t)].$$

By Taylor's polynomial along with remainder expanding ϕ about t_n, we get

$$\phi(t_n + h) = \phi(t_n) + \phi^{'}(t_n)h + \frac{1}{2}\phi^{''}(\tau_n)h^2,$$

where τ_n is any point in $t_n < \tau_n < t_n + h$. Therefore

$$\phi(t_{n+1}) = \phi(t_n) + hf[t_n, \phi(t_n)] + \frac{1}{2}\phi^{''}(\tau_n)h^2.$$

Euler formula for calculating approximate value to $\phi(t_{n+1})$ under the supposition that exact value for z_n at t_n is known, $z_n = \phi(t_n)$. The result is

$$z_{n+1} = \phi(t_n) + hf[t_n, \phi(t_n)].$$

The local truncation error i.e. difference of $\phi(t_{n+1})$ *and* z_{n+1} for the $(n + 1)$ step in the Euler method, denoted as e_{n+1} and given as

$$e_{n+1} = \phi(t_{n+1}) - z_{n+1} = \frac{1}{2}\phi^{''}(\tau_n)h^2.$$

Therefore for Euler method the local truncation error is directly proportional to the square of it's step size h. A uniform bound is well founded on solution's interval given as

$$|e_n| \leq Mh^2/2,$$

here M is the maximum value of $|\phi^{''}(t)|$ on the solution interval . Step size (h) should be chosen in such a way that local truncation error must be not greater than ϵ (tolerance level). Global truncation error has more importance in comparison with local truncation error. Estimating Er_n is much more difficult than evaluating e_n. It is shown that the global truncation error on a finite interval is not greater than a constant times h. Thus

$$|Er_n| \leq Kh,$$

for some constant K. The Euler method is of **first order** due to it's global truncation error which is directly proportional to the step size's first power.

Error analysis of Modified Euler's method

Considering the initial value problem

$$z^{'} = f(t, z), \quad z(t_0) = z_0,$$

where $z = \phi(t)$ denote its solution. After making some alterations in Euler's method the Improved

Euler's method is formed as

$$z_{n+1} = z_n + \frac{f(t_n, z_n) + f[t_n + h, z_n + hf(t_n, z_n)]}{2}h,$$

$$= z_n + \frac{f_n + f(t_n + h, z_n + hf_n)}{2}h.$$

Improved Euler formula is a two-stage method i.e. firstly $z_n + hf_n$ is evaluated using Euler formula and secondly this same result is carried forward to find z_{n+1}. Since the improved Euler formula improvisation of Euler formula with the local truncation error directly proportional to h^3 while in Euler method it is proportional to h^2. In improved Euler method the global truncation error is bounded by a constant times h^2, so this method is of **second order**. It is observed that at the expense of more computational work,greater accuracy is achieved as it is now required to evaluate $f(t, z)$ two times in going from t_n to t_{n+1}.

Error analysis of Runge-Kutta method

Considering the initial value problem

$$z^{'} = f(t, z), \quad z(t_0) = z_0,$$

where $z = \phi(t)$ denote its solution. The Runge-Kutta method of second order behaves same as the modified Euler's method, therefore it's trunaction error and convergence speed is same as that of modified Euler's method.

Classical Runge-Kutta method is given as

$$z_{n+1} = z_n + h\frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6},$$

where

$$\begin{aligned} k_1 &= f(x_n, z_n), \\ k_2 &= f(x_n + h/2, z_n + hk_1/2), \\ k_3 &= f(x_n + h/2, z_n + hk_2/2), \\ k_4 &= f(x_n + h, z_n + hk_3). \end{aligned}$$

The intermediate slopes k_1, k_2, k_3, k_4 are evaluated with use of Euler's and midpoint Euler's method. Therefore the local truncation error is proportional to the h^5 . And from this it follows that global truncation error is proportional to constant times h^4 so this method is of **fourth order** which reflects that there are four intermediate stages in calculation. Which in return is more computationaly expensive as more number of computations are required but accuracy is achieved to highest.

References

[1] William E. Boyce and Richard C. DiPrima, Elementary Differential Equations and Boundary Value Problems, Tenth Edition, Wiley, 2012.

[2] Shepley L. Ross, Differential Equations, Third Edition, Wiley, 2012.

[3] J.C. Butcher, Numerical Methods for Ordinary Differential Equations, Second Edition, Wiley, 2008.

[4] Brian D. Storey, Numerical Methods for Ordinary Differential Equations, course notes, Franklin W. Olin College of Engineering, 2003.

[5] Endre Süli, Numerical Solution of Ordinary Differential Equations, lecture notes, University of Oxford, 2014.

[6] Peter J. Olver, Numerical Solution of Ordinary Differential Equations, Numerical Analysis Lecture Notes, University of Minnesota, 2008.

[7] Jim Lambers, Higher-Order Equations and Systems of Differential Equations, Lecture 7 Notes, University of Southern Mississippi, 2009.

[8] K.B. Devaki and S.Swathi, The system of Differential equations in Prey Predator model, International Journal of Multidisciplinary Research and Modern Education (IJMRME) Volume II, Issue I, 2016.

[9] Anne Kværnø, Numerical solution of Ordinary Differential equations, lecture notes, Norwegian University of Science and Technology, 2009.

[10] Dr. Ir. Bob Foster, Determining Dynamic Market Equilibrium Price Function Using Second Order Linear Differential Equations, International Journal of Humanities and Social Science Vol. 6, No. 11, 2016.

[11] R.B. Ogunrinde and J. Sunday, On some models based on second order differential equations, American Journal of Scientific and Industrial Research, 2012.

[12] Ibijola E.A, Obayomi A. A., Non-standard method for the solution of Ordinary Differential Equations arising from Pricing Policy for Optimal Inventory Level, Canadian Journal on Computing in Mathematics, Natural Sciences, Engineering and Medicine Vol. 3 No. 7, 2012.