DEPARTMENT OF ………..

UNIVERSITY OF ……………..

Simulation of Rocket Motion

STUDENT: ……………………………………………

PROGRAM TUTOR: ……………………………..

ACADEMIC YEAR: 200…/0…

DATE: ………………………………………..

UNIT No: …………………………………….

Summary

In this report, motion of a rocket is simulated by creating Visual Basic macro in Microsoft Excel. The instantaneous velocity, altitude and acceleration at different points are calculated using both Euler and Runge-Kutta integration methods. The Runge-Kutta method is found to be more precise while the Euler method is faster.

Notation

$a\ (y, v, t)$	acceleration
h	timestep
$k_{1\text{-}4}$, $g_{1\text{-}4}$,	intermediate values
Q	fuel mass flow rate
v	velocity
y	altitude
t	time

1. Introduction

Spreadsheet can be used as a tool of modeling and simulation due to its excellent features of organizing data. It can be produced by Visual Basic macro to present complex systems. In this report, a spreadsheet is used to describe the state of rocket by producing the physical data in a series of points along the trajectory, in order to predict the process of rocket motion. Two numerical integration methods are used and their results are compared.

2. Theory

The task is to simulate the flight of a rocket. The basic model is a second-order differential equation, but with a number of variable parameters:

- the rocket mass reduces as the fuel is burnt ;
- the aerodynamic drag is a function of both velocity and altitude (i.e. the air density);
- the gravity is varying with the rocket altitude.

Therefore the only solution must be of numerical integration nature. In Euler integration method, linear approximation is used. Hence the altitude at any instant y_{i+1} and the velocity v_{i+1} are approximated on the basis of their values in the previous instant and a linear increase:

$$y_{i+1} \approx y_i + v_i h \tag{1}$$

$$v_{i+1} \approx v_i + ah \tag{2}$$

In Runge-Kutta method, four intermediate points are used, so for the $(n+1)^{th}$ timestep the integration is calculated as:

$$k_{1n} = v_n \tag{3a}$$

$$g_{1n} = a(y_n, v_{n,}, t_n) \tag{3b}$$

$$k_{2n} = v_n + \frac{1}{2} h g_{1n} \tag{4a}$$

$$g_{2n} = a(y_n + \frac{1}{2} h k_{1n}, k_{2n}, t + \frac{1}{2} h) \tag{4b}$$

$$k_{3n} = v_n + \frac{1}{2} h g_{2n} \tag{5a}$$

$$g_{3n} = a(y_n + \frac{1}{2} h k_{2n}, k_{3n}, t + \frac{1}{2} h) \tag{5b}$$

$$k_{4n} = v_n + h g_{3n} \tag{6a}$$

$$g_{4n} = a(y_n + h k_{3n}, k_{4n}, t + h) \tag{6b}$$

$$y_{n+1} = y_n + \frac{h}{6}(k_{1n} + 2k_{2n} + 2k_{3n} + k_{4n}) \tag{7}$$

$$v_{n+1} = v_n + \frac{h}{6}(g_{1n} + 2g_{2n} + 2g_{3n} + g_{4n}) \tag{8}$$

3. Results

Visual Basic macro, shown in the Appendix, is made to perform the calculations of rocket motion. Figure 1 shows the acceleration (a), velocity (b) and altitude (c) versus time generated by Euler method, using time step of 0.2 *s*, for the fuel flow rate of 20 *kg/s*.

Next, time step is changed between 0.07 *s* to 1 *s* and the maximum altitude calculated with Euler method is recorded, as shown in Figure 2a. A similar procedure is done with Runge-Kutta method, changing time step from 0.5 *s* to 16 *s*, as shown in Figure 2b. To achieve the calculated maximum altitude accuracy better than approximately 0.1% , a time step of 0.69 *s* or less is needed for Euler method and a much bigger time step of up to 14 *s* can be chosen for the Runge-Kutta method.

Next, fuel flow rate is changed and calculated maximum altitude is compared, using Runge-Kutta method at time step at 0.2 *s*. The results are plotted in Figure 3 and for some values, tabulated in Table 1, which shows that the approximate fuel flow rate of 34 *kg/s* gives the greatest altitude of about 1,346,271 meters.

4. Discussion

The process of the rocket motion with an initial fuel burn rate of 20 *kg/s* and time step of 0.2 *s* is discussed here as an example. Figure 4a was zoomed in from figure 1a, showing the acceleration of the rocket within the first 200 *s* of flight. From the launch until about 23 *s*, the atmospheric drag increases sharply to a peak value as Figure 4b shows. This leads to a small decrease in acceleration. Then, due to decrease of atmospheric drag as the rocket clears the atmosphere, coupled with a reduction of the rocket mass, the acceleration increases by about 20 m/s^2. After 70 *s*, due to the absence of air, the drag becomes almost negligible. Trend of acceleration is similar to the thrust (Figure 4c) until 150 *s*, when the fuel was used out. When the acceleration became zero, the velocity attains its maximum positive value.

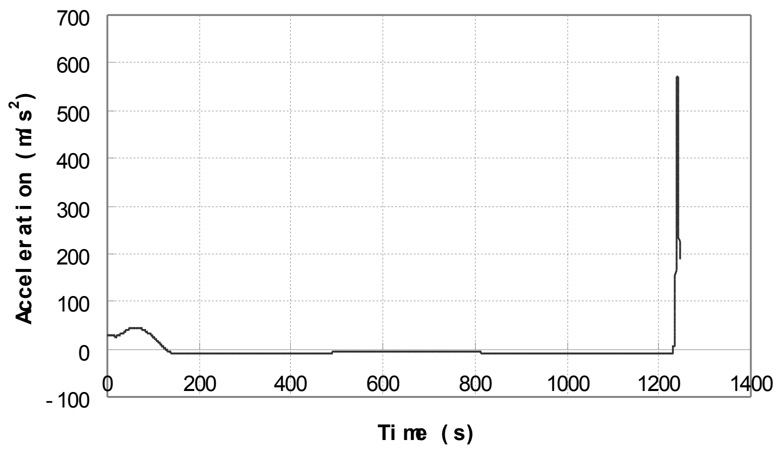
After 150 *s*, the acceleration of the rocket can be considered as the gravitational acceleration, which is a negative value. As the rocket climbs up its velocity decreased gradually, until it falls to zero. At that moment in time, the rocket reaches its maximum altitude of about 1276400 *m*. Then the rocket moves downwards and the absolute value of both acceleration and velocity increase accordingly. When the rocket drops to a certain altitude, the air density begins to increase. In addition, the rocket has a high velocity at that time, so the atmospheric drag increases sharply. The acceleration hence decreases, but the rocket falls to the Earth at a high speed.

The algorithm is implemented as a loop, in which the calculation of variables in each step is based on the previous step, as shown in the Appendix. A difference between the two methods is that Runge-Kutta method calls the acceleration function three times within one loop rather than only once as in the Euler method. Hence the Euler method is much faster than the Runge-Kutta method of a same time step. On the other hand, the Runge-Kutta method has a greater precision and a larger time step can be used, compensation the slower execution of the more complex algorithm.

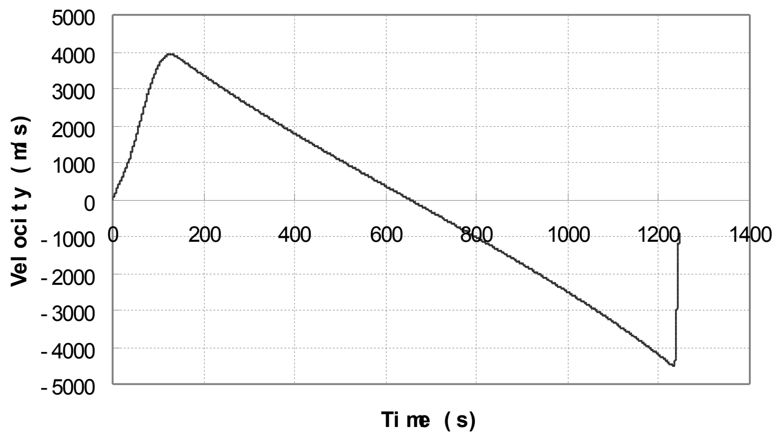
5. Conclusion

It can be concluded from the simulations that errors exist in both numerical integration methods and that reduction of timestep leads to more accurate results, but requires more time for the process. Selecting the same timestep, Runge-Kutta method is more precise than Euler method, whereas it is less efficient. The influence of different parameters (like timestep and fuel flow rate) on the motion of rocket was investigated, using both methods.

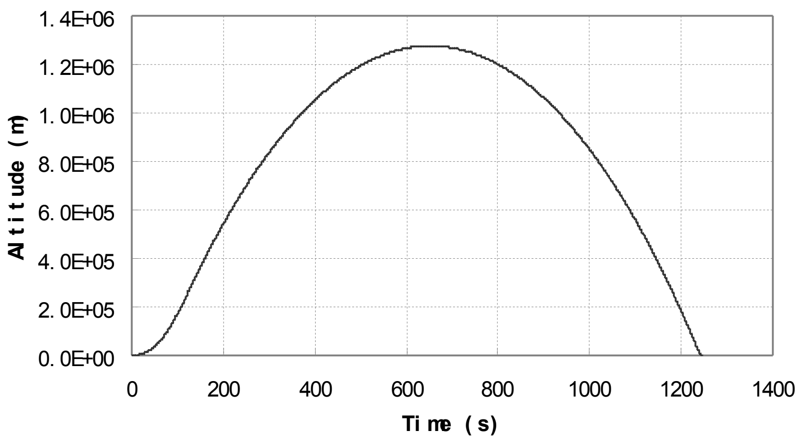
Figures and tables



(a) acceleration



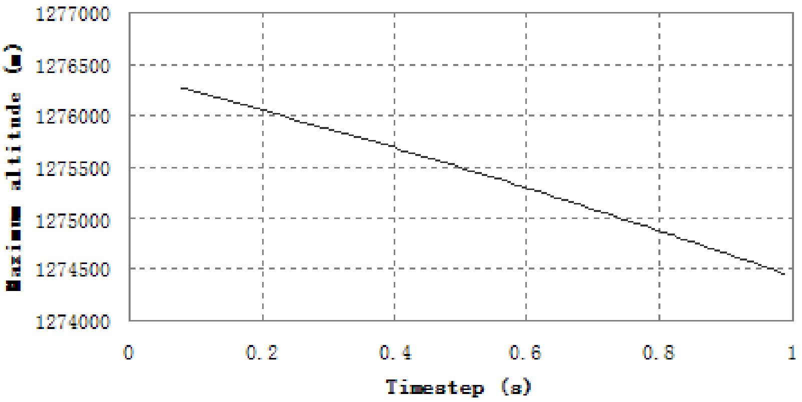
(b) velocity



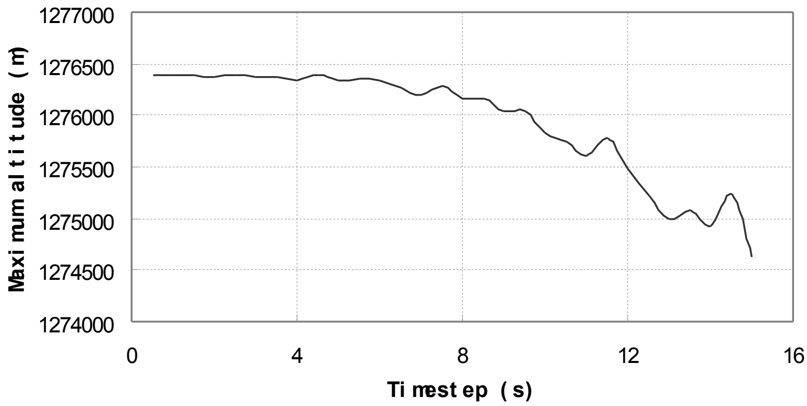
(c) altitude

Figure 1 *Simulated acceleration, velocity and altitude traces produced by Euler method*

(Q_0=20 kg/s, timestep=0.2 s)



(a) maximum altitude vs. timestep for Euler method



(b) maximum altitude vs. timestep for Runge-Kutta method

Figure 2: *The maximum altitude the rocket reaches, calculated by the two methods*

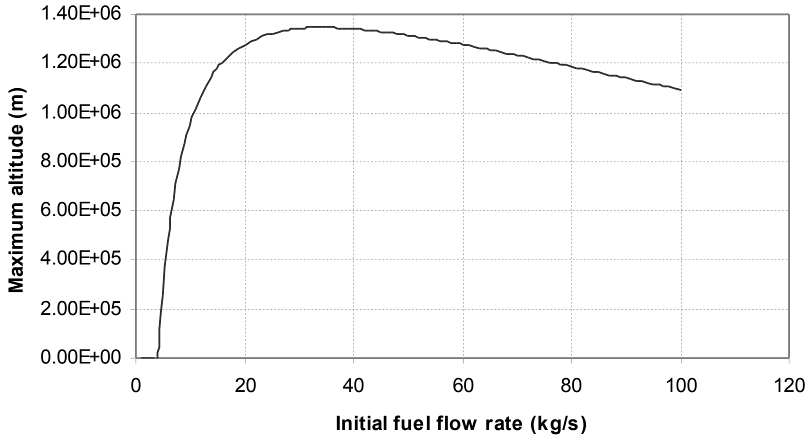
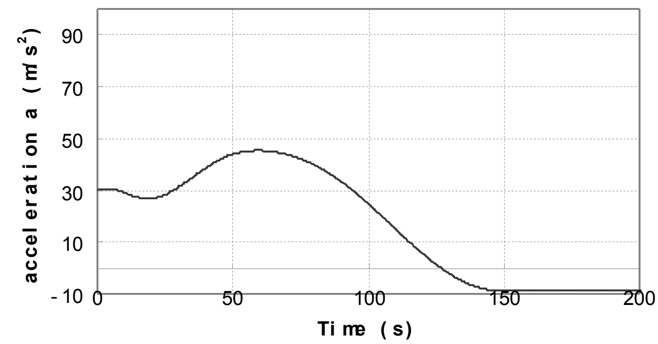
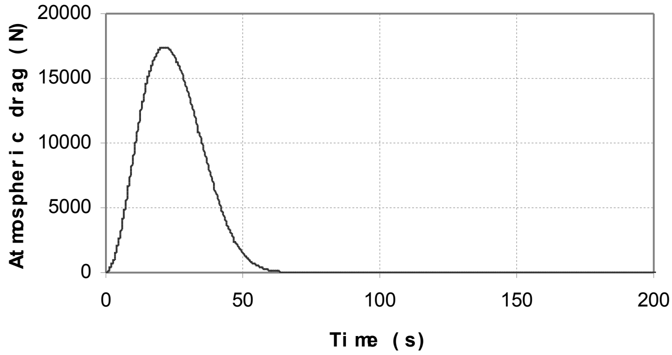


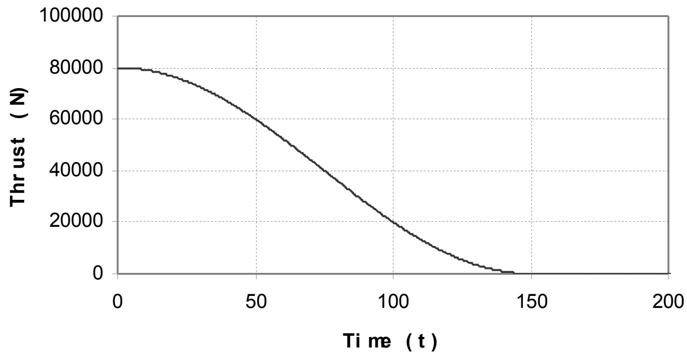
Figure 3: *The maximum altitude for different fuelburn rates Q_0 , Runge-Kutta method*



(a) acceleration



(b) atmospheric drag



(c) thrust

Figure 4: *Variations of acceleration, drag and thrust in time period from 0 to 200 seconds*

Table 1: *The maximum altitude for different fuelburn rates Q_0 ,Runge-Kutta method*

Q0 (kg/s)	20	21	22	23	24	25
Maximum altitude (m)	1276431	1288796	1299411	1308500	1316253	1322829
Q0 (kg/s)	26	27	28	29	30	31
Maximum altitude (m)	1328365	1332976	1336763	1339812	1342199	1343990
Q0 (kg/s)	32	33	34	35	36	37
Maximum altitude (m)	1345242	1346006	1346328	1346248	1345800	1345016

Appendix – programming code for the Macros

Euler Method:

```
Function accel(y, v, t, q0)
Pi = 3.14
tau = 3000 / q0
If t < tau Then
  q = 0.5 * q0 * (1 + Cos(Pi * t / tau))
  m = 2000 - 0.5 * q0 * (t + tau * Sin(Pi * t / tau) / Pi)
Else
  q = 0
  m = 500
End If
thrust = 4000 * q
rou = 1.2 * Exp(-(y / 7000))
drag = 0.1 * rou * v * Abs(v)
g = 6370000 ^ 2 * 10 / (6370000 + y) ^ 2
accel = (thrust - m * g - drag) / m
End Function

Sub ra()
y = 0
v = 0
t = 0
q0 = 20
i = 1
h = 0.2
While y >= 0 And i < 10000
    a = accel(y, v, t, q0)
    Cells(i, 1).Value = t
    Cells(i, 2).Value = v
    Cells(i, 3).Value = y
    Cells(i, 4).Value = a
    y = y + v * h
    v = v + a * h
    t = t + h
    i = i + 1
Wend
End Sub

Sub rb()
h = 0.07
j = 1
While h <= 1
  ye = 0
    Cells(i, 3).Value = y
    Cells(i, 4).Value = a
    y = y + h * (k1 + 2 * k2 + 2 * k3 + k4) / 6
    v = v + h * (g1 + 2 * g2 + 2 * g3 + g4) / 6

    t = t + h
    i = i + 1
Wend
End Sub

Sub rd()
h = 0.5
j = 2
While h <= 15
  yrk = 0
  y = 0
  v = 0
  t = 0
  q0 = 20
  i = 1
```

```
  While v >= 0 And i < 10000
    k1 = v
    g1 = accel(y, v, t, q0)
    k2 = v + 0.5 * h * g1
    g2 = accel(y + 0.5 * h * k1, k2, t + 0.5 * h, q0)
    k3 = v + 0.5 * h * g2
    g3 = accel(y + 0.5 * h * k2, k3, t + 0.5 * h, q0)
    k4 = v + h * g3
    g4 = accel(y + h * k3, k4, t + h, q0)
    v = v + h * (g1 + 2 * g2 + 2 * g3 + g4) / 6
    y = y + h * (k1 + 2 * k2 + 2 * k3 + k4) / 6
    yrk = y
    t = t + h
    i = i + 1
   Wend
   Cells(j, 1).Value = h
   Cells(j, 2).Value = yrk
   j = j + 1
   h = h + 0.5
Wend
End Sub
```

Runge-Kutta Method

```
 y = 0
 v = 0
 t = 0
 q0 = 20
 i = 1
  While y >= 0 And i < 10000
    a = accel(y, v, t, q0)
    y = y + v * h
    v = v + a * h
    If y > ye Then
      ye = y
    End If
    t = t + h
    i = i + 1
   Wend
   Cells(j, 1).Value = h
   Cells(j, 2).Value = ye
   j = j + 1
   h = h + 0.01
Wend
End Sub

Sub rc()
y = 0
v = 0
t = 0
q0 = 20
tau = 3000 / q0
i = 1
h = 0.2
While y >= 0 And i < 10000
k1 = v
g1 = accel(y, v, t, q0)
k2 = v + 0.5 * h * g1
g2 = accel(y + 0.5 * h * k1, k2, t + 0.5 * h, q0)
k3 = v + 0.5 * h * g2
g3 = accel(y + 0.5 * h * k2, k3, t + 0.5 * h, q0)
k4 = v + h * g3
g4 = accel(y + h * k3, k4, t + h, q0)
   a = accel(y, v, t, q0)
   Cells(i, 1).Value = t
   Cells(i, 2).Value = v
Sub re()
q0 = 0.5
j = 1
While q0 <= 100
  y = 0
  ymax = 0
  v = 0

  t = 0
  i = 1
  h = 0.2
  While y >= 0 And i < 10000
    k1 = v
    g1 = accel(y, v, t, q0)
    k2 = v + 0.5 * h * g1
    g2 = accel(y + 0.5 * h * k1, k2, t + 0.5 * h, q0)
    k3 = v + 0.5 * h * g2
    g3 = accel(y + 0.5 * h * k2, k3, t + 0.5 * h, q0)
    k4 = v + h * g3
    g4 = accel(y + h * k3, k4, t + h, q0)
    a = accel(y, v, t, q0)
    v = v + h * (g1 + 2 * g2 + 2 * g3 + g4) / 6
```

```
  y = y + h * (k1 + 2 * k2 + 2 * k3 + k4) / 6
  If y > ymax Then
    ymax = y
  End If
  t = t + h
  i = i + 1
Wend
    Cells(j, 1).Value = q0
    Cells(j, 2).Value = ymax
    q0 = q0 + 1
    j = j + 1
Wend
End Sub
```