Institute for Astronomical and Physical Geodesy

ESPACE: Orbit Mechanics

Exercise 2:

«Numerical Integration of Satellite Orbits»

WS14/15

Prepared by Aleksei Petukhov

aleksei.petukhov@tum.de

Contents:

Given data

For numerical integration MATLAB provides the functions **ode23**, **ode45** and **ode113** (see **help**). As input parameters the integration time steps, the initial conditions (position, velocity) of the integration **y0** and the program **yprime**, which contains the differential equation are required.

Compute numerically the trajectory of the GOCE satellite for 3 revolutions. Use the following Kepler elements for the GOCE satellite:

Satellit	a [km]	e	i [°]	Ω [°]	ω [°]	τ[sec]
GOCE	6629	0.004	96.6	257.7	144.2	0

Disturbed equation of motion:

$$\ddot{\mathbf{r}} = -\frac{GM}{r^3}\mathbf{r}\left[1-\frac{3}{2}J_2\frac{a_e^2}{r^2}\begin{pmatrix}5\left(\frac{z}{r}\right)^2-1\\ 5\left(\frac{z}{r}\right)^2-1\\ 5\left(\frac{z}{r}\right)^2-3\end{pmatrix}\right] \qquad with:\ \mathbf{r}=\begin{pmatrix}x\\ y\\ z\end{pmatrix}$$

$$J_2 = -C_{20} = 0.00108263;\ GM = 398600.5\ km^3/s^2;\ a_e = 6378\ km$$

Brief overview

Our objective is to compare results of analytically derived satellite trajectory with numerically integrated. So, we can derive the difference in trajectory obtained through applying those two approaches. We consider two cases: un-disturbed and disturbed due to the flattening of Earth. Therefore we can observe to which extent perturbations change the satellite's orbit. Matlab environment is helpful for calculating orbital parameters and plotting the orbits.

The tutorial is carried out under the supervision of Prof. Dr. Urs Hugentobler and Ye Hao.

Problems 1, 2, 3

1) Analytically compute the undisturbed trajectory (positions & velocities) for 3 revolutions from the Kepler elements (program should be available from previous exercise in orbit mechanics).
2) Write a program yprime, providing the differential equation for the undisturbed Kepler problem. Compute the trajectory (positions & velocities) for 3 revolutions using two different MATLAB integrators and two different step sizes.
3) Compare the analytical and numerical derived results in terms of plots for each component.

From the previous exercise we obtain analytical solution for GOCE's orbit using functions **kep2orb** and **kep2cart**. Thus, we get positions and velocities values for 3 revolution of the satellite.

Then we use one set of parameters (3 components of position and 3 components of velocity) as an initial point for numerical integration method.

We use this set as an input parameters for **yprime** function, which computes new velocity components and accelerations (derivatives of velocity components).

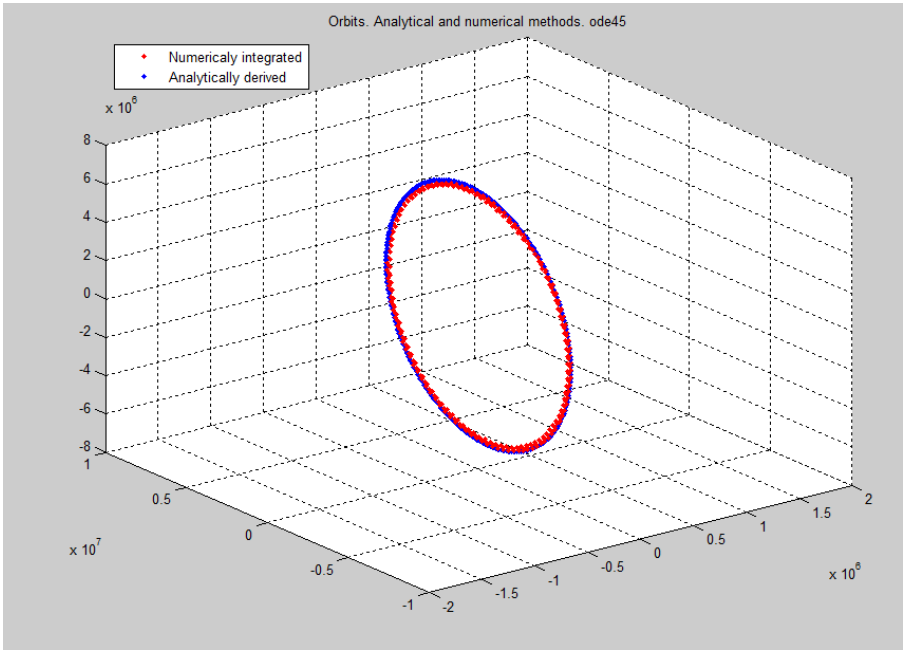
$$\ddot{r}_x = -\frac{GM}{R^3}x; \ddot{r}_y = -\frac{GM}{R^3}y; \ddot{r}_z = -\frac{GM}{R^3}z$$

Picture 1: Acceleration components

Than we use one of inner matlab functions (ode23, ode45, ode113) for solving differential equations. Given the accelerations (computed within the **yprime** function) we are capable of computing the position and velocity parameters for the next point and so on. After using ode functions we get set of positions and velocities.

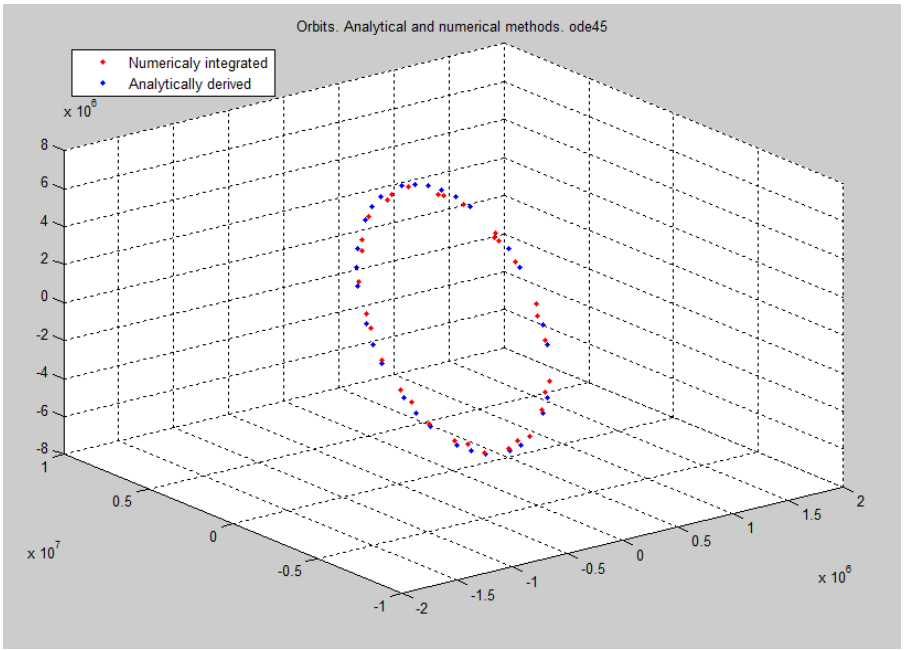
Thus, we obtain the orbit trajectory via numerical integration.

Step-size affects the accuracy of computation, *the larger the step-size the less resemblance* bear analytically derived trajectory and numerically integrated. When plotting the differences in components, the step-size affects smoothness of the curve. *The larger the step-size the less smooth the curve is.* This is proved with the following plots: *(not all plots are included into the report)*



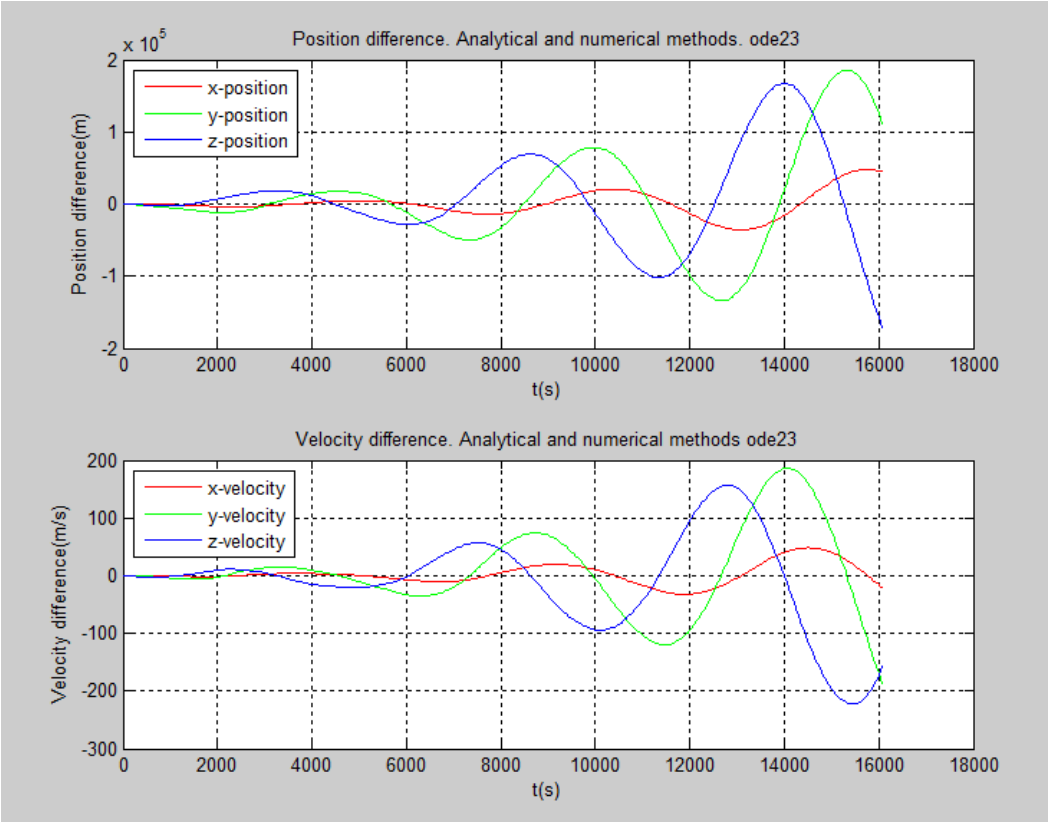
Picture 2: Numerically integrated and analytically derived orbits. **Undisturbed case.**

STEP-SIZE 50. ODE 45



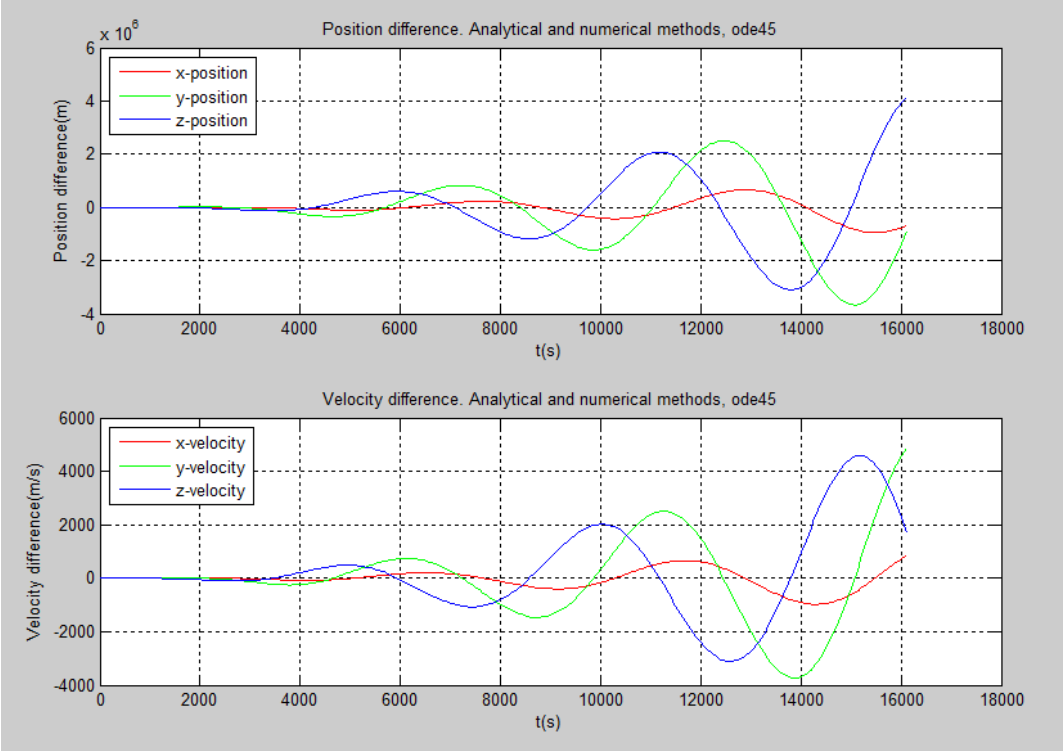
Picture 3: Numerically integrated and analytically derived orbits. **Undisturbed case.**

STEP-SIZE 500. ODE 45



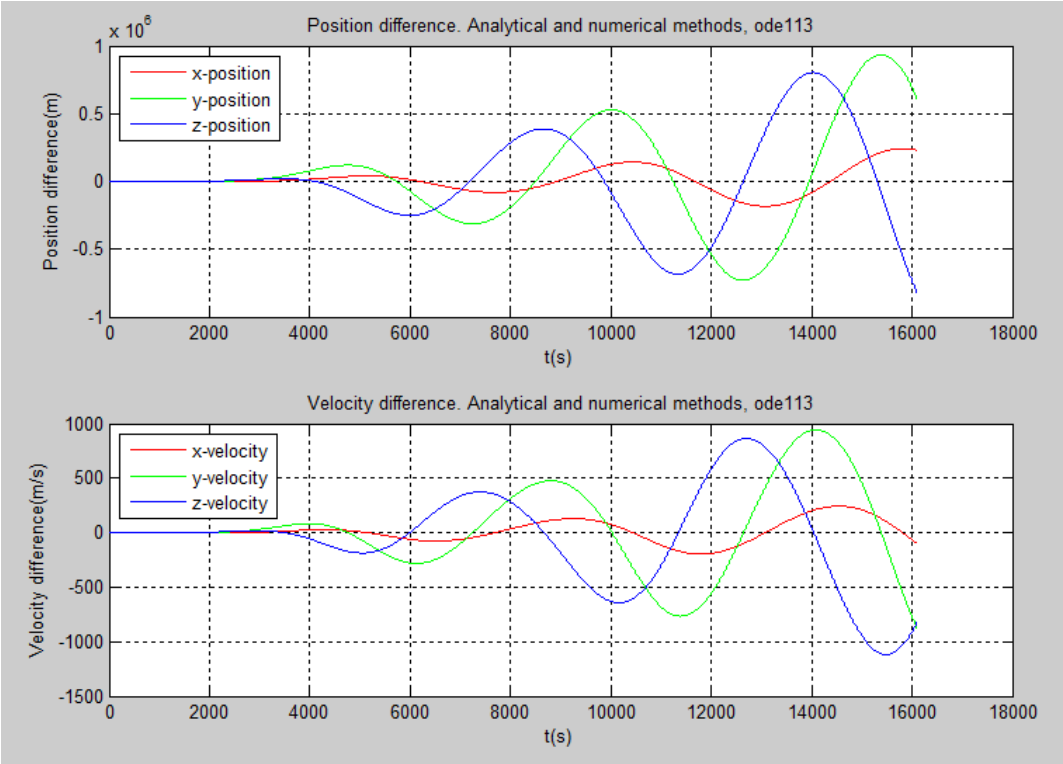
Picture 4: Analytical and numerical methods. Difference in position and velocity components.

Undisturbed case. STEP-SIZE 50, ODE 23



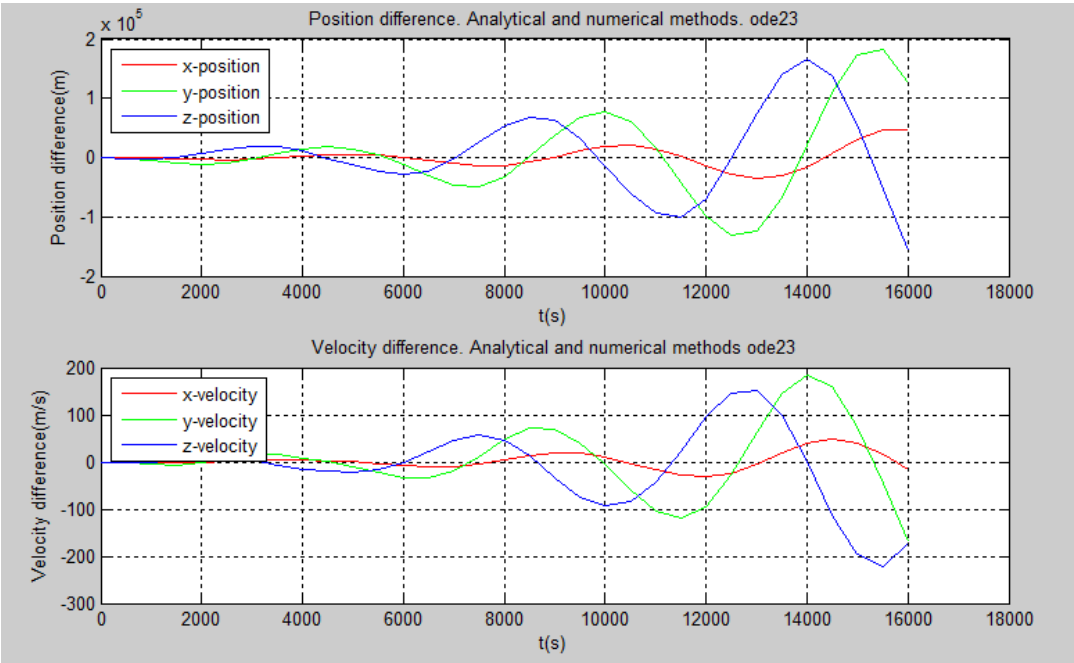
Picture 5: Analytical and numerical methods. Difference in position and velocity components.

Undisturbed case. STEP-SIZE 50, ODE 45



Picture 5: Analytical and numerical methods. Difference in position and velocity components.

Undisturbed case. STEP-SIZE 5, ODE 113



Picture 6: Analytical and numerical methods. Difference in position and velocity components.

Undisturbed case. STEP-SIZE 500, ODE 23

The error keeps aggregating with every revolution and thus the difference between orbits increases.

Elapsed time for different ode functions (undisturbed case)

ODE23: step 5 – *0,275640* sec, step 50 – *0,192969*

ODE45: step 5 – *0,186490* sec, step 50 – *0,136168*

ODE113: step 5 – *0,190267* sec, step 50 – *0,132603*

Problem 4

Take into account the Earth's flattening and update your program yprime accordingly. Apply the formulas given below. Plot the differences between the un-disturbed and the disturbed case.

Previously, we considered undisturbed orbit. Now we take into account the Earth's flattening. First we update **yprime function** adding to the differential equations perturbing acceleration:

$$\ddot{\mathbf{r}} = -\frac{GM}{r^3}\mathbf{r}\left[1-\frac{3}{2}J_2\frac{a_e^{\,2}}{r^2}\begin{bmatrix}5(\frac{z}{r})^2-1\\ 5(\frac{z}{r})^2-1\\ 5(\frac{z}{r})^2-3\end{bmatrix}\right]$$

Picture 7: Orbit perturbation due to oblateness (flattening). Pluging-in the perturbing acceleration component into the differential equation of motion

Flattening is a measure of the compression of a circle or a sphere along a diameter to form an ellipse or an ellipsoid of revolution (spheroid) respectively.

Oblateness of the earth only depends on the z-axis, therefore it's a function of z on all three components

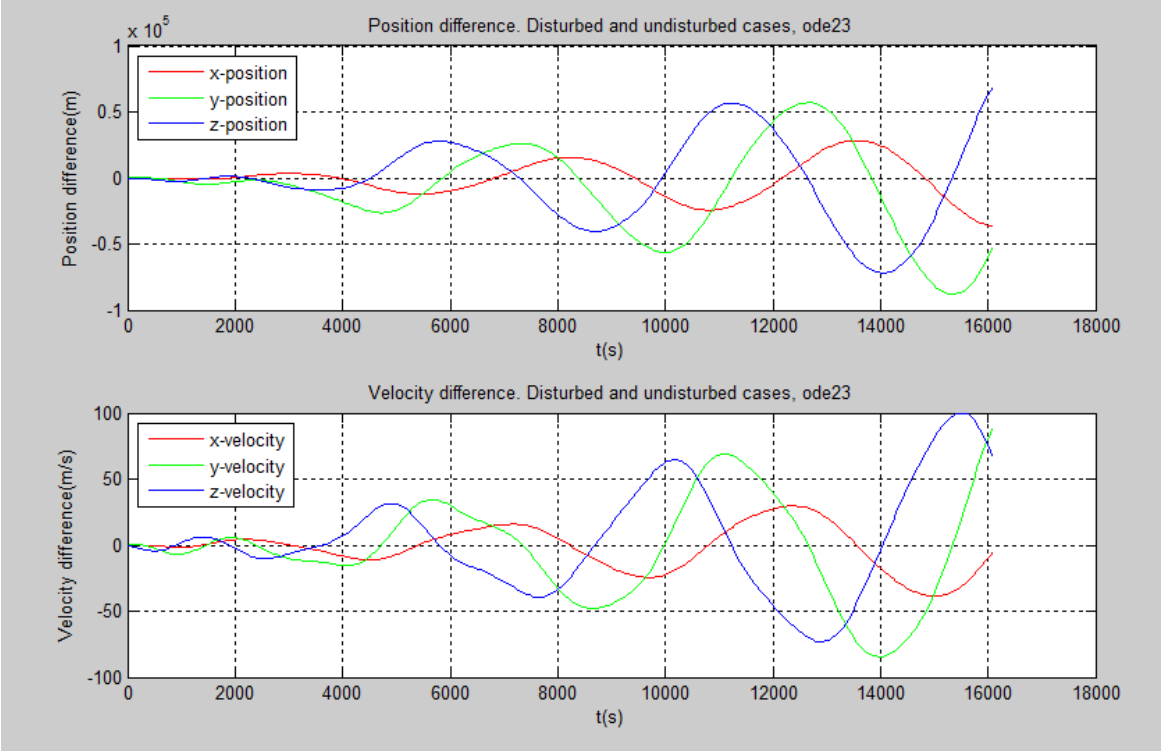
J_2 – special zonal term.

In reality the Earth is squashed and can't most accurately be treated as a point mass, as it is treated in the two-body assumption. This squashing shape is called *oblateness*.

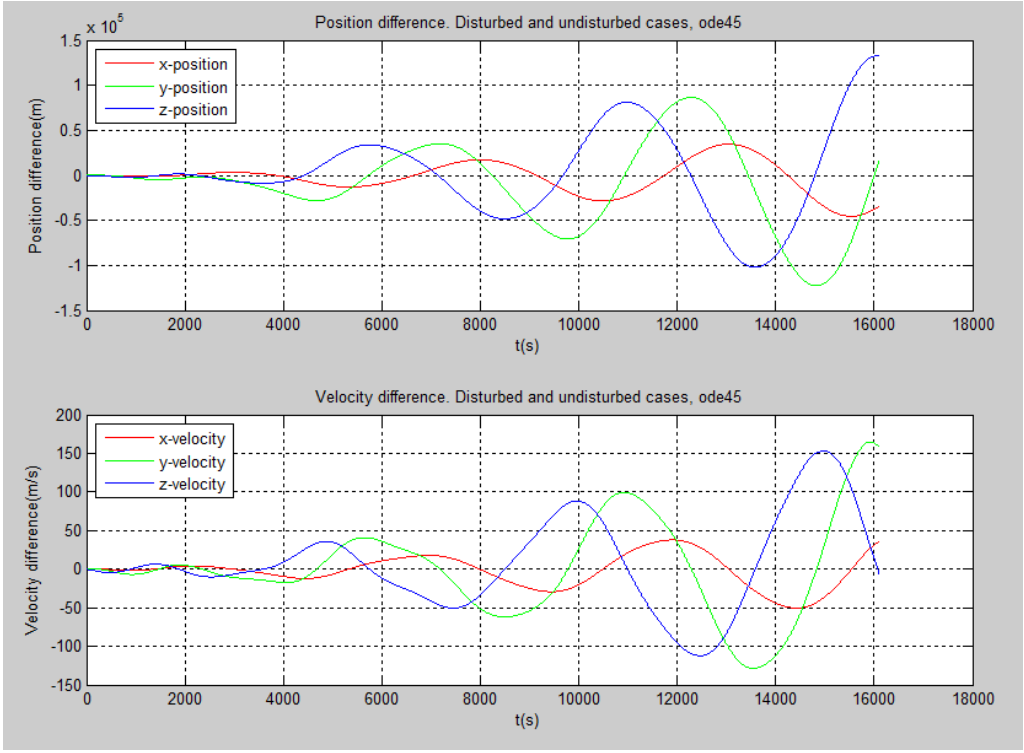
J2 is a constant describing *the size of the bulge* in the mathematical formulas used to model the oblate Earth. This term arises from the mathematical short-hand used to describe Earth's gravitational field. (Gravitational acceleration at any point on Earth is commonly expressed as a geopotential function expressed in terms of Legendre polynomials and dimensionless coefficients Jn). We consider J2 since it's the most dominating term.

Now every point on orbit is being calculated taking into account this perturbation and therefore the orbit changes. Now ode functions use as a solver updated **yprime** function.

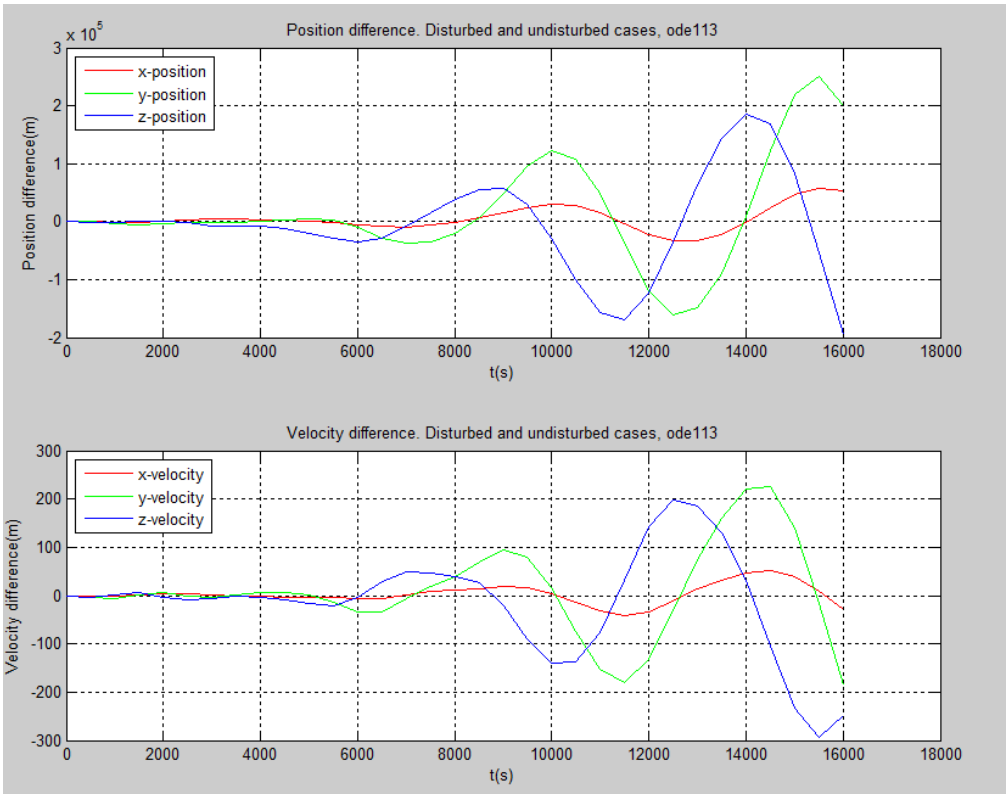
Let's see the difference in disturbed and undisturbed cases in these plots below: (not all plots are included in the report)



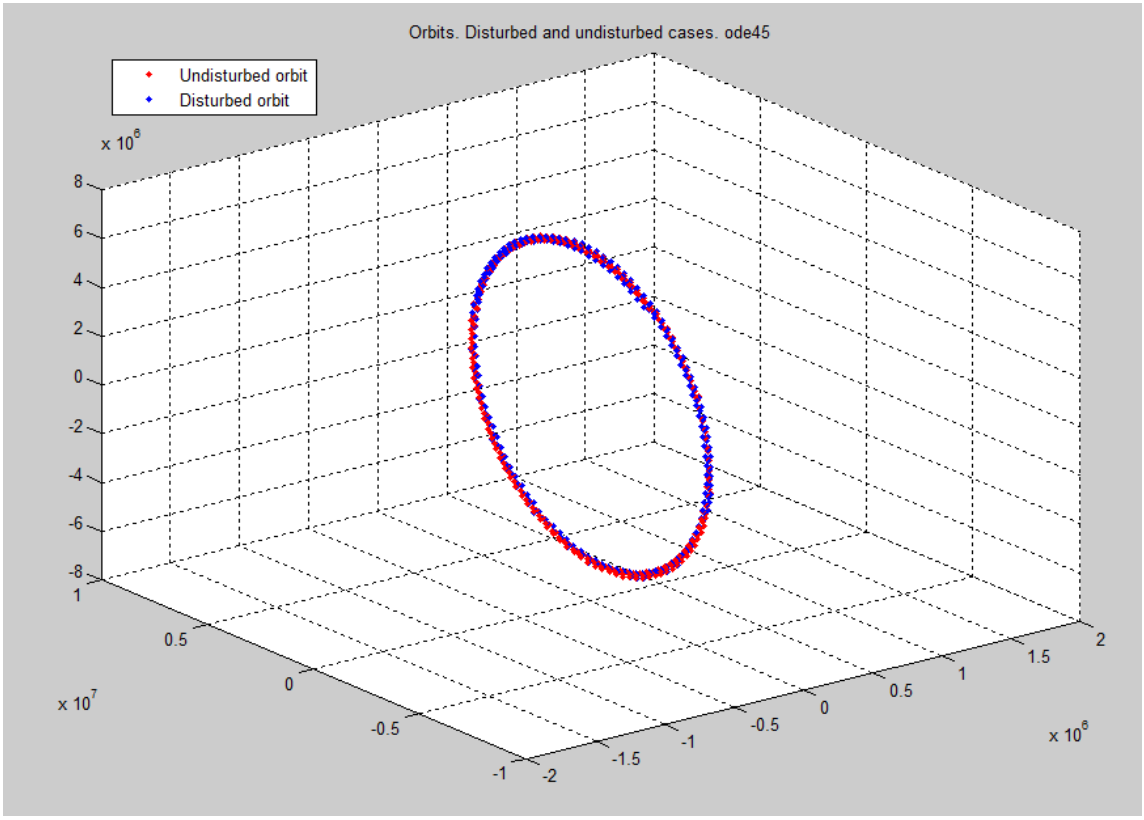
Picture 8: The difference between disturbed and undisturbed cases. **ODE 23 STEP-SIZE 50**



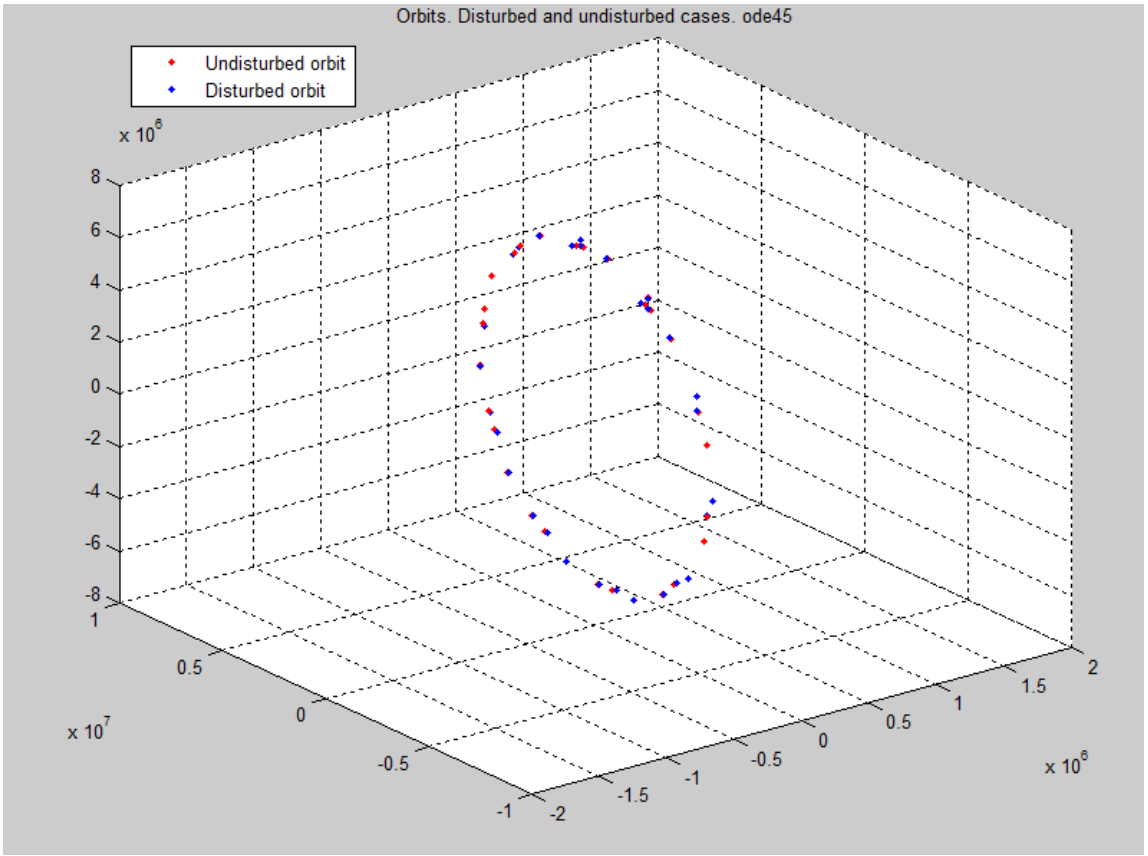
Picture 9: The difference between disturbed and undisturbed cases. **ODE 45 STEP-SIZE 5**



Picture 10: The difference between disturbed and undisturbed cases. **ODE 113 STEP-SIZE 500**



Picture 11: Orbits. Disturbed and undisturbed cases. **ODE 45 STEP-SIZE 50**



Picture 12: Orbits. Disturbed and undisturbed cases. **ODE 45 STEP-SIZE 500**

Obviously, we can observe that the difference grows with every revolution. Difference in positions and velocities is the least in x-component.

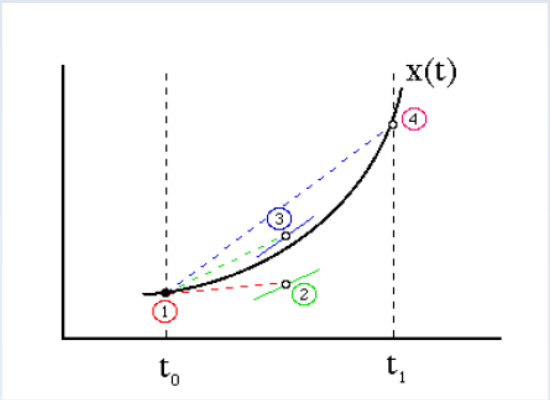
Problem 5

Develop your own simple integrator for the undisturbed case and investigate the impact of the step size on your result. Compare results with the analytical solution computed above.

Previously we used inner matlab functions for solving differential equations and whereby those functions carried out the numerical integration process.

Now we going to create simple integrator based on Runge-Kutta method. This method is used in intemporal discretization for the approximation of solutions of ordinary differential equations.

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + \mathrm{k}_4) + O(\mathrm{h}^5)$$
$$\mathrm{k}_1 = hf(\mathrm{t}_i, \mathrm{y}_i)$$
$$\mathrm{k}_2 = hf(\mathrm{t}_i + \frac{h}{2}, \mathrm{y}_i + \frac{k_1}{2})$$
$$\mathrm{k}_3 = hf(\mathrm{t}_i + \frac{h}{2}, \mathrm{y}_i + \frac{k_2}{2})$$
$$\mathrm{k}_4 = hf(\mathrm{t}_i + h, \mathrm{y}_i + k_3)$$



Picture 13: Visualisation of Runge-Kutta method of 4th order (RK4)

k_1 is the increment based on the slope at the beginning of the interval, using $\dot{y}$,
k_2 is the increment based on the slope at the midpoint of the interval, using $\dot{y} + \frac{h}{2}k_1$;
k_3 is again the increment based on the slope at the midpoint, but now using $\dot{y} + \frac{h}{2}k_2$;
k_4 is the increment based on the slope at the end of the interval, using $\dot{y} + hk_3$.

The RK4 method is a fourth-order method, meaning that the local truncation error is of the order $O(h^5)$, while the total accumulated error is of order $O(h^4)$.

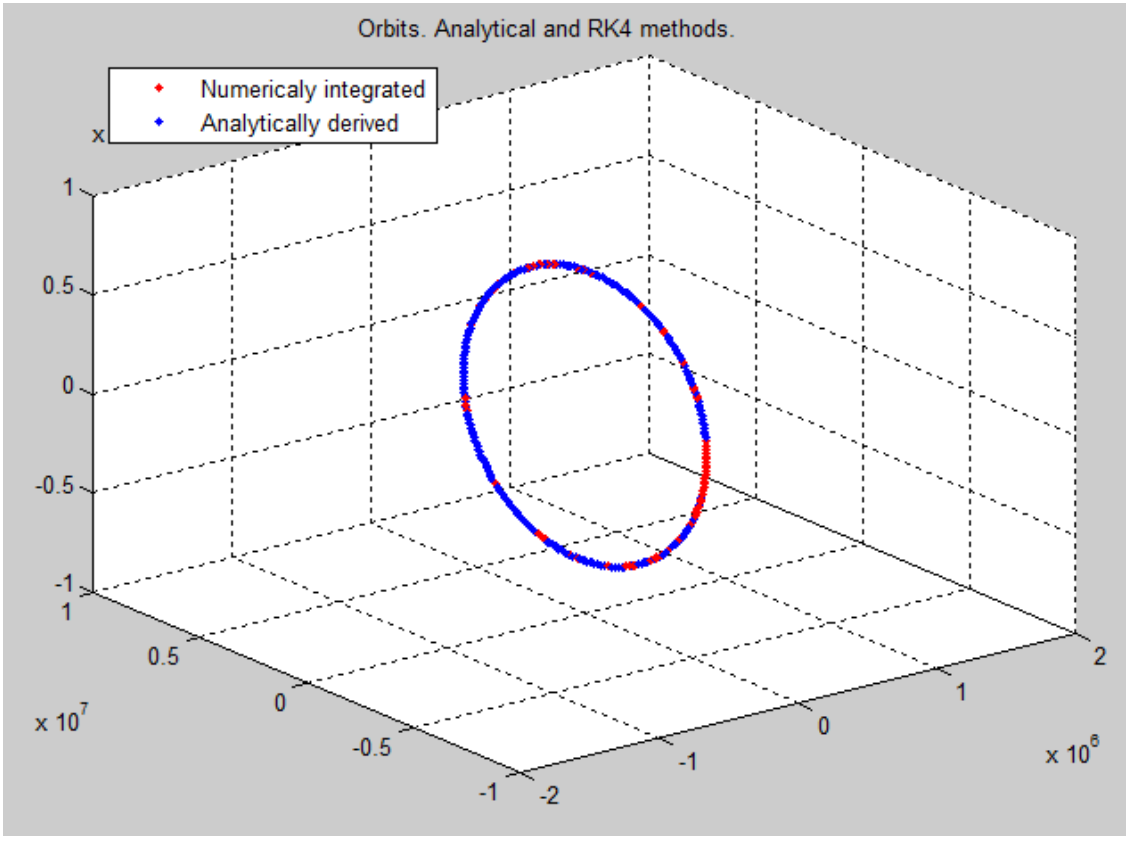
```
h=result;
y=zeros(length(y0),length(t));
y(:,1)=y0;
for i=2:length(t)
   k1=h*yprime(t,y0);
   k2=h*yprime(t+h/2,y0+k1/2);
   k3=h*yprime(t+h/2,y0+k2/2);
   k4=h*yprime(t+h,y0+k3);
   y(:,i)=y0 +(1/6)*(k1+2*k2+2*k3+k4);
   y0=y(:,i); %prevoius y(i) becomes y0 for the next point
end
```

Picture 14: implementation of RK4 method in MATLAB. h is the step size (input prompt parameter)

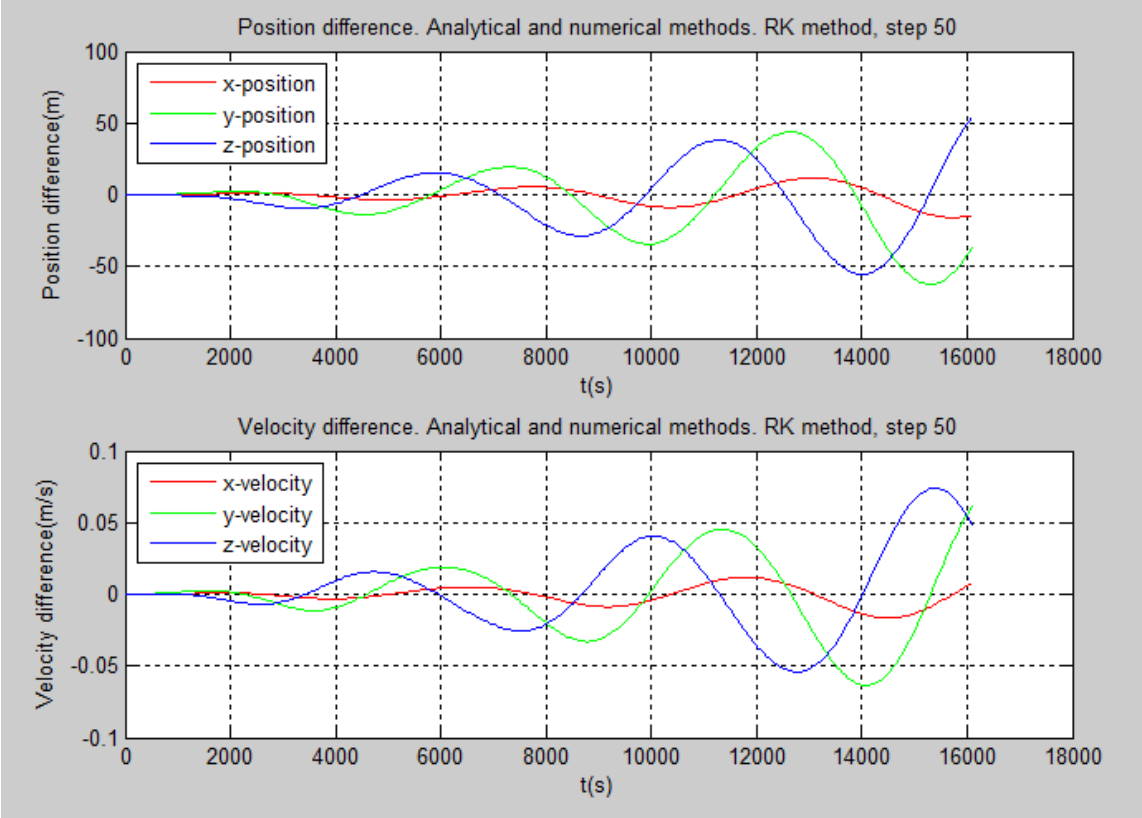
yprime (undisturbed case) is the function that is being approximated.

Implementing this method, we obtain the following pictures:

Step size 50

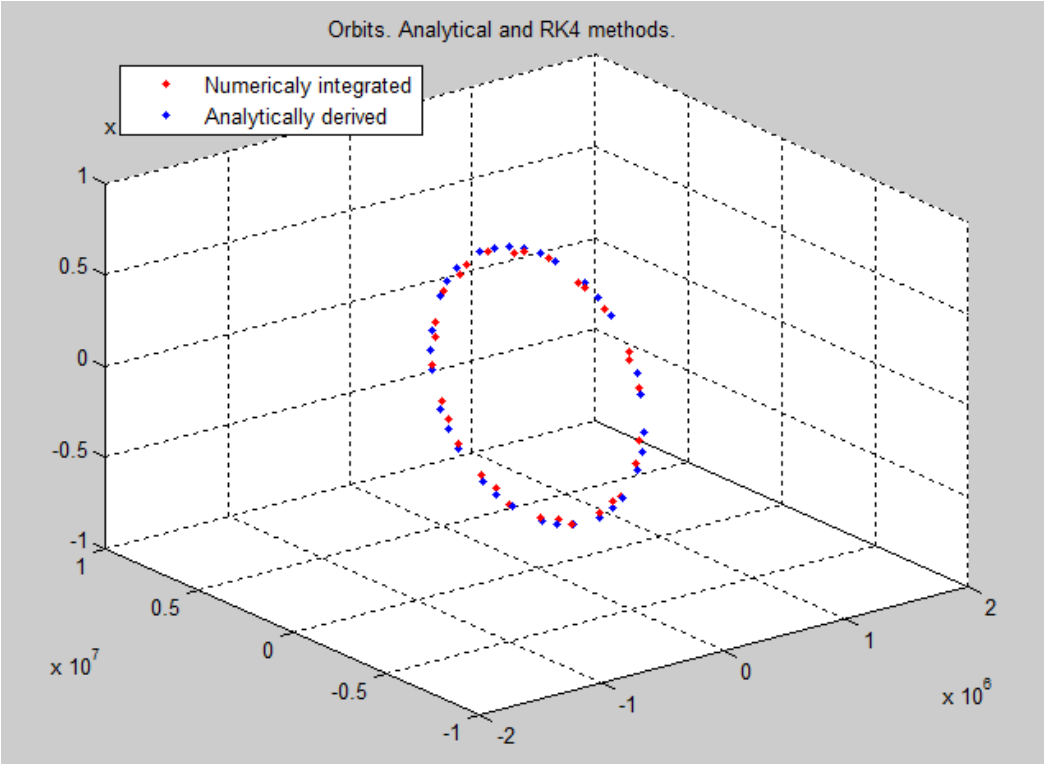


Picture 15: Orbits. Implementation of RK4 method. **Step-size 50**

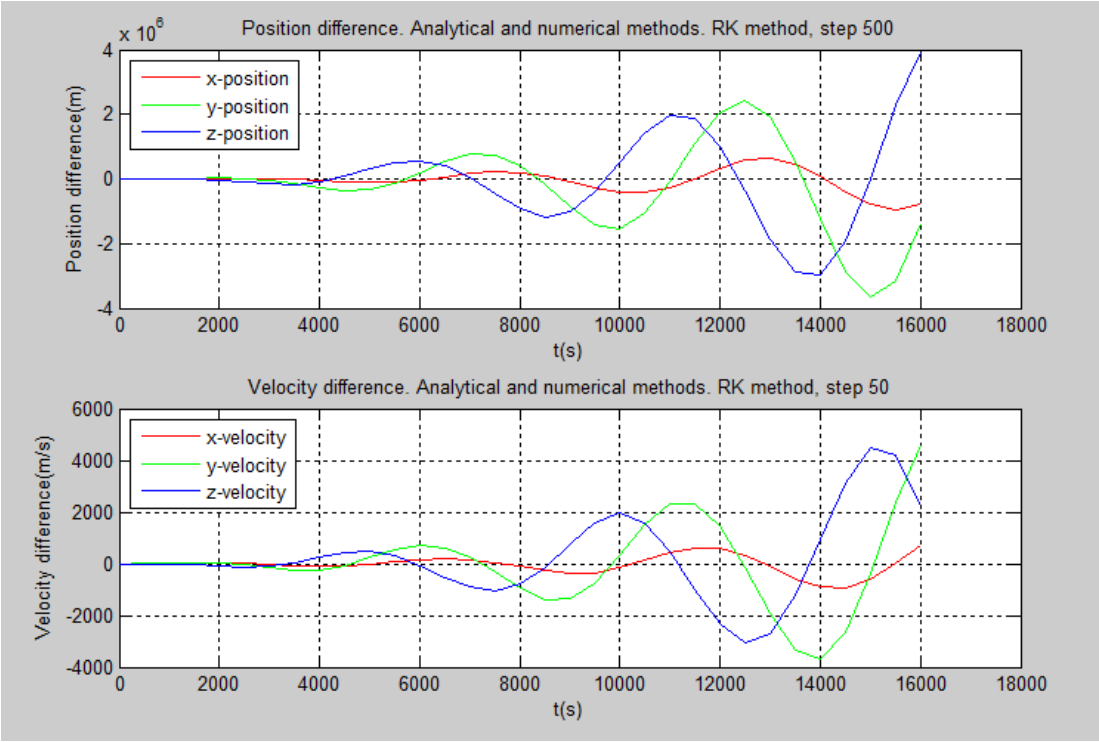


Picture 16: Difference between RK4 and analytical methods. **Step-size 50**

Step size 500

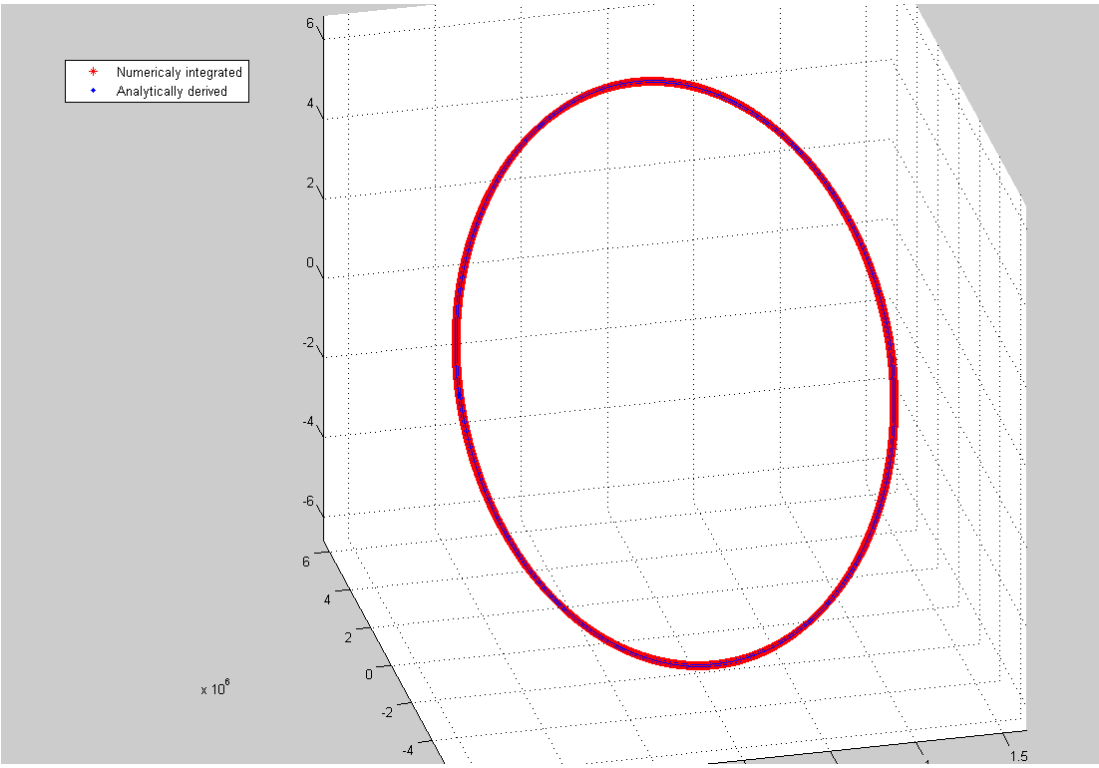


Picture 17: Orbits. Implementation of RK4 method. **Step-size 500**

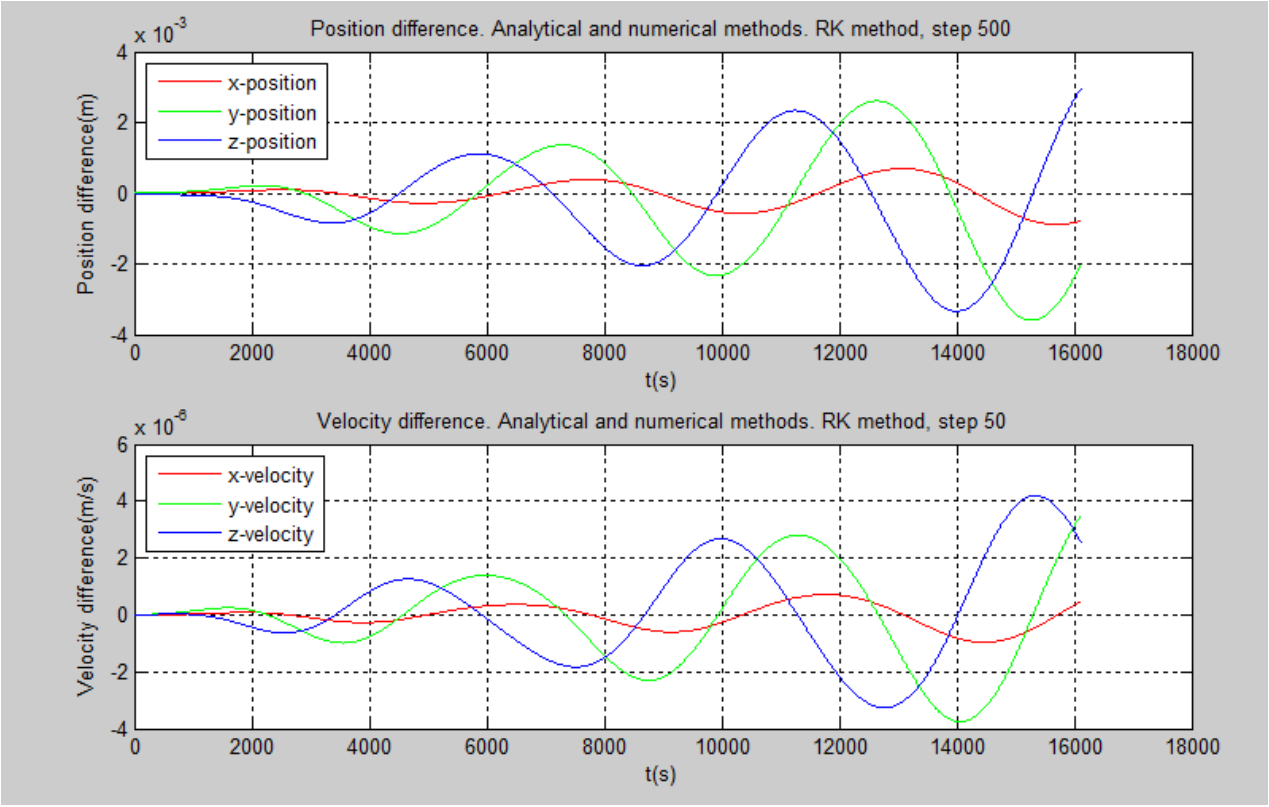


Picture 18: Difference between RK4 and analytical methods. **Step-size 500**

Step size 5



Picture 19: Orbits. Implementation of RK4 method. **Step-size 5.** Almost complete match, difference is very small



Picture 20: Difference between RK4 and analytical methods. **Step-size 5**

With the small step-size numerical solution comes very close to analytical solution. With a step size of 5 the difference is of 10^{-6} order in velocity and 10^{-3} in position. Whereas with the step size of 50 the velocity difference is of 10^{-2} order and position difference is of 10^{2} order.

With the growth of step size, the growth of velocity difference is bigger than the growth of position difference.

The results bear much resemblance with the ode-function results, since those ode-functions (apart from ode113, which uses Adams method) also use Runge-Kutta method but more explicit.

Code:

yprime

```
function yp =yprime(t,y0)
%the first derivative of y
%y0 is initial parameters of y
GM=398.6005*10^12;
yp=zeros(6,1);
yp(1)=y0(4);
yp(2)=y0(5);
yp(3)=y0(6);
R=sqrt(y0(1)^2+y0(2)^2+y0(3)^2);
yp(4)=-GM/R^3*y0(1);
yp(5)=-GM/R^3*y0(2);
yp(6)=-GM/R^3*y0(3);
end
```

yprime_disturbed

```
function yp_disturbed = yprime_disturbed(t,y0)
%the first derivative of y
% yois initial parameters of y
J=0.00108263;
ae=6378000; %m
GM=398.6005*10^12;
yp_disturbed=zeros(6,1);
yp_disturbed(1)=y0(4);
yp_disturbed(2)=y0(5);
yp_disturbed(3)=y0(6);
R=sqrt(y0(1)^2+y0(2)^2+y0(3)^2);
flattening_x = (1-(((3/2)*J*(ae^2)/R^2)*(5*((y0(3)/R)^2)-1)));
flattening_y = (1-(((3/2)*J*(ae^2)/R^2)*(5*((y0(3)/R)^2)-1)));
flattening_z = (1-(((3/2)*J*(ae^2)/R^2)*(5*((y0(3)/R)^2)-3)));
yp_disturbed(4)=-GM/R^3*y0(1)*flattening_x;
yp_disturbed(5)=-GM/R^3*y0(2)*flattening_y;
yp_disturbed(6)=-GM/R^3*y0(3)*flattening_z;
end
```

Runge-Kutta

```
h=result;  % prompt input
y=zeros(length(y0),length(t));
y(:,1)=y0;
for i=2:length(t)
```

```
    k1=h*yprime(t,y0);
    k2=h*yprime(t+h/2,y0+k1/2);
    k3=h*yprime(t+h/2,y0+k2/2);
    k4=h*yprime(t+h,y0+k3);
    y(:,i)=y0 +(1/6)*(k1+2*k2+2*k3+k4);
    y0=y(:,i); %prevoius y(i) becomes y0 for the next point
end
```

Ode functions and difference matrices:

```
%% Undisturbed case
options = odeset('RelTol',1e-10,'AbsTol',1e-10);
[tour y23] = ode23('yprime',t,y0);
[tour y45] = ode45('yprime',t,y0);
[tour y113] = ode113('yprime',t,y0);

%% Disturbed case
[tour y23_d] = ode23('yprime_disturbed',t,y0);
[tour y45_d] = ode45('yprime_disturbed',t,y0);
[tour y113_d] = ode113('yprime_disturbed',t,y0);

%% Exercise 1 2 3
analytical_orbit_matrix =
[position(1,:);position(2,:);position(3,:);velocity(1,:);velocity(2,:);velocity(3,:)]';
difference_matrix_23=analytical_orbit_matrix-y23;
difference_matrix_45=analytical_orbit_matrix-y45;
difference_matrix_113=analytical_orbit_matrix-y113;

%% Exercise 4
difference_dist_undist_matrix_23 = y23_d - y23;
difference_dist_undist_matrix_45 =  y45_d - y45;
difference_dist_undist_matrix_113 = y113_d - y113;
```

Code of plotting is unnecessary