Simulation of a satellite trajectory

Lars Schiller

May 13, 2015

Contents

1 Introduction and task description **1**

2 Mathematical model of the task **1**

3 Description and evaluation of the results **2**

4 Interpretation of the results **3**

1 Introduction and task description

Ziel des Wettbewerbs ist die numerische Simulation der ebenen Flugbahn eines Satelliten um die Erde. Es sollen hierbei die Gravitationskräfte zwischen dem Satelliten, der Erde und der Sonne beachtet werden, wobei die Gravitationskonstante $\gamma = 1$ beträgt. Alle weiteren Effekte dürfen vernachlässigt werden. The initial conditions are given with

	r_{1x}	r_{1y}	r_{2x}	r_{2y}	r_{3x}	r_{3y}	$\dot{r}_{1x}$	$\dot{r}_{1y}$	$\dot{r}_{2x}$	$\dot{r}_{2y}$	$\dot{r}_{3x}$	$\dot{r}_{3y}$	m_1	m_2	m_3	γ
value:	0	0	20	0	20	-2	-.02	-.22	0	2	2.1	2.1	100	10	1	1

.

2 Mathematical model of the task

In the following the sun will be subscripted by 1, the earth by 2 and the satellite by 3. Furthermore vectors will be written as bold symbols, like $\boldsymbol{r}_1$ as the position vector of the sun.

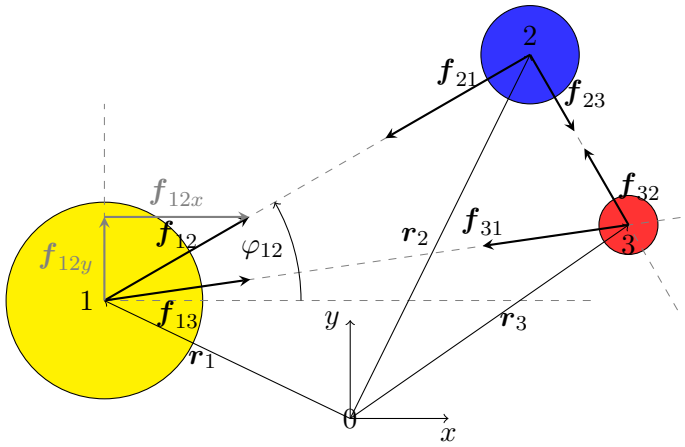


Figure 1: Resultant Forces on the objects

From *Newtons* Gravity Law it is known[1] that mass afflicted bodies generate forces to each other with the following rule:

$$F = \gamma \frac{m_1 m_2}{|\boldsymbol{r}|^2}. \tag{1}$$

The problem we have two solve consist of three mass points with two degrees of freedom respectively. The resultant forces between these points can be described with respect to equation 1 by:

$$\boldsymbol{f}_{ij} = \underbrace{\gamma \frac{m_i m_j}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2}}_{\text{magnitude}} \underbrace{\frac{\boldsymbol{r_j} - \boldsymbol{r}_i}{|\boldsymbol{r_j} - \boldsymbol{r}_i|}}_{\text{direction}} = \gamma \frac{m_i m_j}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^3} \left(\boldsymbol{r_j} - \boldsymbol{r}_i\right), \tag{2}$$

where i, j are the indices of the mass points. The force $\boldsymbol{f}_{ij}$ acts on the point i and is generated by the presence of point j. To obtain the x-part of the force one can scalar multiply $\boldsymbol{f}_{ij}$ with the unit vector in x-direction, and accordingly with y:

$$f_{ijx} = \boldsymbol{f}_{ij}^T \boldsymbol{e}_x, \qquad f_{ijy} = \boldsymbol{f}_{ij}^T \boldsymbol{e}_y \tag{3}$$

Figure 1 illustrates these forces. In aim to obtain a differential equation to simulate the given task one can use *Newtons* Second Law:

$$m\ddot{r} = \sum_n f_n. \tag{4}$$

Since this is a differential equation of order 2, we can use the well known trick and reformulate equation 4 in:

$$\begin{bmatrix} 1 & 0 \\ 0 & \boldsymbol{M} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{r}} \\ \ddot{\boldsymbol{r}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \boldsymbol{f}(\boldsymbol{r}) & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{r} \\ \dot{\boldsymbol{r}} \end{bmatrix}, \tag{5}$$

where $\boldsymbol{M}$ is the mass matrix and $\boldsymbol{f}(\boldsymbol{r})$ denotes a function which depends on the actual position of the objects. By defining the state vector $\boldsymbol{x}$ as:

$$\boldsymbol{x} = [r_{1x} \;\; r_{1y} \;\; r_{2x} \;\; r_{2y} \;\; r_{3x} \;\; r_{3y} \;\; \dot{r}_{1x} \;\; \dot{r}_{1y} \;\; \dot{r}_{2x} \;\; \dot{r}_{2y} \;\; \dot{r}_{3x} \;\; \dot{r}_{3y}]^T$$

equation 5 can be written as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}(7:12) \\ \frac{1}{m_1}\left(f_{12x} + f_{13x}\right) \\ \frac{1}{m_1}\left(f_{12y} + f_{13y}\right) \\ \frac{1}{m_2}\left(f_{21x} + f_{23x}\right) \\ \frac{1}{m_2}\left(f_{21y} + f_{23y}\right) \\ \frac{1}{m_3}\left(f_{31x} + f_{32x}\right) \\ \frac{1}{m_3}\left(f_{31y} + f_{32y}\right) \end{bmatrix} \tag{6}$$

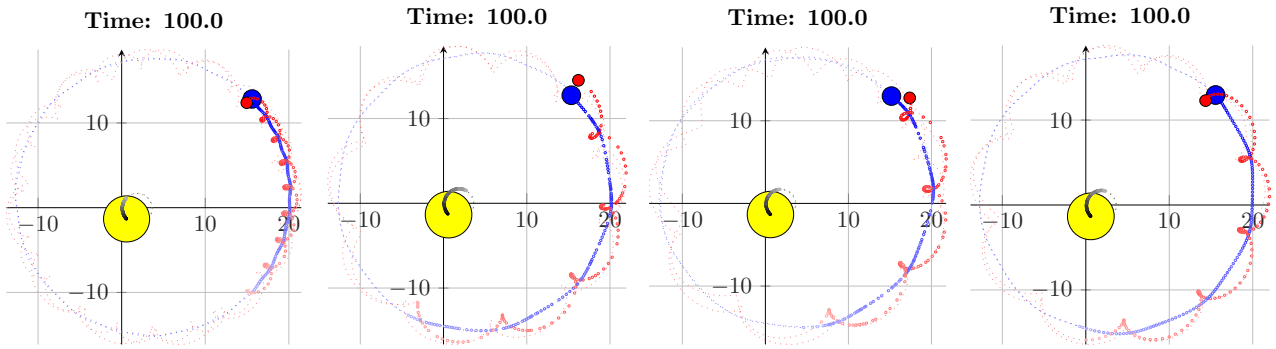
This is a non-linear ordinary system of differential equations, which can be solved with the known methods, like the advanced *Euler*-Method.

3 Description and evaluation of the results

The results shown in figure 2 are obtained by the `Matlab` implemented solvers `ode45`,`ode113`, `ode15s` and the self-written solver `euler.m` with a relative tolerance of `1e-3`. The results shown in figure 3 are obtained by the same solvers, but with a relative tolerance of `1e-6`, except the *Euler*-method, which has a relative tolerance of `1e-4`, due to the massive computational effort.

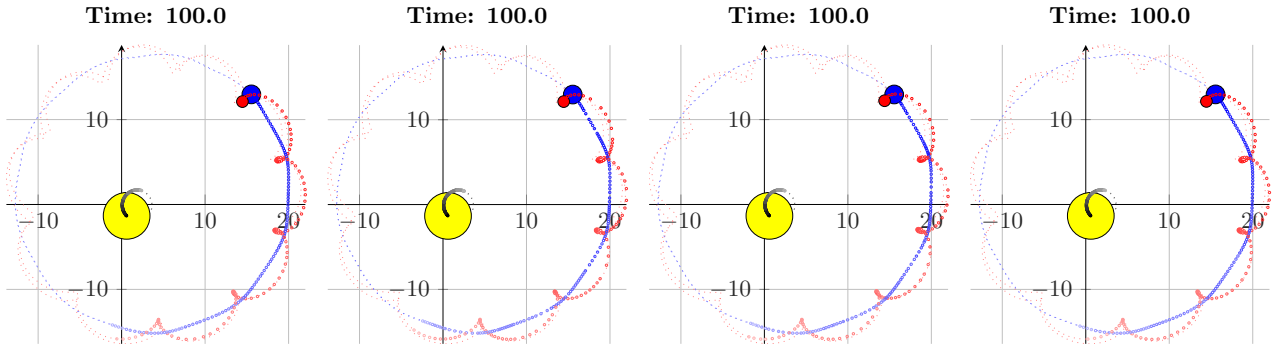
It's easy to see, that the results shown in figure 2 are inconsistent. For example in figure 2(a) the satellite is going to crash with the earth. This is reasoned by the fact that for the given problem a high accuracy is necessary. By increasing the relative tolerance the results of every method become similar. In table 1 the number of needed supporting points of every solving method are listed. For the needed high accuracy `ode113` get along with least supporting points and therefore with least computational effort.

[1] `http://en.wikipedia.org/wiki/Newton's_law_of_universal_gravitation`



(a) `ode45` (b) `ode113` (c) `ode15s` (d) *Euler*

Figure 2: Results of the Simulation obtained by several numerical solvers with `RelTol = 1e-3`



(a) `ode45` (b) `ode113` (c) `ode15s` (d) *Euler*

Figure 3: Results of the Simulation obtained by several numerical solvers with `RelTol = 1e-6`

4 Interpretation of the results

Since the the results of every solving method are nearly the same for a tight relative tolerance, its reasonable to trust in these. Therefore with the assumptions we have done and with the given initial conditions, the trajectory of the satellite will stable. But with a little phase shift of the satellite per rotation around the sun by the earth. By accepting this, a later control will not be necessary.

	`ode45`	`ode113`	`ode15s`	Advanced *Euler*
supporting points (`RelTol = 1e-3`)	1309	688	866	6410
supporting points (`RelTol = 1e-6`)	2881	1249	2180	-
supporting points (`RelTol = 1e-4`)	-	-	-	20292

Table 1: Evaluation of the results