



Numerical Methods in Physics

Numerische Methoden in der Physik, 515.421.

Instructor: Ass. Prof. Dr. Lilia Boeri
Room: PH 03 090
Tel: +43-316-873 8191
Email Address: l.boeri@tugraz.at

Room: TDK Seminarraum

Time: 8:30-10 a.m.

Exercises: Computer Room, PH EG 004 F

http://itp.tugraz.at/LV/boeri/NUM_METH/index.html
(Lecture slides, Script, Exercises, etc).

Important notice (exam):

Exams for the lecture in numerical methods in Physics will take place in my office PH3.108, starting 20th of January 2013.

For the **winter semester**, I will offer dates in the weeks:

- 20th-26th of January
- 27th – 31st of January
- 3rd-9th of February.

I will post a list with available dates and slots **outside my office door** next week. You can register writing your name on the list, **up to 14th of January**.

There will be additional dates in May-June and October 2014.



TOPICS (this year):

- **Chapter 1: Introduction.**
- **Chapter 2: Numerical methods for the solution of linear inhomogeneous systems.**
- Chapter 3: Interpolation of point sets.
- **Chapter 4: Least-Squares Approximation.**
- **Chapter 5: Numerical solution of transcendental equations.**
- Chapter 6: Numerical Integration.
- **Chapter 7: Eigenvalues and Eigenvectors of real matrices.**
- **Chapter 8: Numerical Methods for the solution of ordinary differential equations: initial value problems.**
- Chapter 9: Numerical Methods for the solution of ordinary differential equations: marginal value problems.

Last week (3/12/2013)

Ordinary Differential Equations: Initial Value problems

- Initial Value problems: Definitions.
- Reduction of an n^{th} -order differential equation to a system of N first-order equations.
- Solution of a first-order differential system of equations: Runge-Kutta methods.
- Simple, modified and improved Euler's methods.
- Fourth-order Runge-Kutta formulas: classical Runge Kutta formula, meaning and examples.

Numerical Methods for the solution of ordinary differential equations: initial value problems.

Definition: A differential equation is a mathematical equation for an unknown function of one or several variables that relates the values of the function itself and its derivatives of various orders. We consider here only the case of **explicit** differential equations, i.e. those which can be solved for the highest possible derivative n :

$$y^{(n)} = F(x; y, y', \dots, y^{(n-1)})$$

An n -th order differential can always be recast into a first-order differential system of the form:

$$\mathbf{y}' = f(x, \mathbf{y})$$

Setting:

$$y'_1 = y_2 \equiv f_1(x)$$

...

$$y'_{t-1} = y_t = f_{t-1}(x)$$

$$y'_n = F(x; y_1, y_2, \dots, y_n) = f_n(x)$$

Runge-Kutta Methods:

Runge-Kutta methods are among the most popular methods for **initial value problems**. They are used alone, or as “pre-condition” methods for more refined algorithms (predictor-corrector methods). The biggest **disadvantage** is that it is difficult to obtain a reliable error estimate.

Properties:

- **Runge-Kutta** methods derive from a Taylor series expansion of the solution, truncated at p order. p is the **order of the R-K method**.
- Runge-Kutta methods are **one-step** methods: in order to determine the value of the solution in x_0+h , it is enough to know its value in the previous point x_0 .
- In order to calculate the approximate value of $y(x)$, it's enough to know $f(x_0, y_0)$, but not its derivatives!

Runge-Kutta Methods of Arbitrary Order:

Runge-Kutta *ansatz* for arbitrary p order:

$$\hat{y}_i(x_0 + h) = y_i(x_0) + h \sum_{j=1}^p c_j g_j$$

$$g_1 = f(x_0, y_0)$$

$$g_j = f(x_0 + a_j h; y_0 + h \sum_{l=1}^{j-1} b_{j,l} g_l)$$

The first g 's are for example:

$$g_1 = f(x_0, y_0)$$

$$g_2 = f(x_0 + a_2 h, y_0 + h b_{2,1} g_1)$$

$$g_3 = f(x_0 + a_3 h, y_0 + h b_{3,2} g_2 + h b_{3,1} g_1)$$

The p -th Runge-Kutta formulas contain 3 types of coefficients ($c_i, a_i, b_{i,j}$), which are mutually related by recursion formulas. The total number of coefficients is $(p^2 + 3p - 2)/2$.

Popular Runge-Kutta formulas:

Euler's Method (1st order):

$$\hat{y}(x_0 + h) = y(x_0) + h \cdot f(x_0, y_0)$$

Modified Euler's Method (2nd order):

$$\hat{y}(x_0 + h) = y_0 + h \cdot f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} f(x_0, y_0)\right)$$

Improved Euler's Method (3rd order):

$$\hat{y}(x_0 + h) = y_0 + h \cdot \left\{ \frac{1}{2} f(x_0, y_0) + \frac{1}{2} f\left[x_0 + h, y_0 + hf(x_0, y_0)\right] \right\}$$

"Classical" Runge-Kutta Method (4th order):

$$\hat{y}(x_0 + h) = y(x_0) + h \left[\frac{1}{6} g_1 + \frac{1}{3} g_2 + \frac{1}{3} g_3 + \frac{1}{6} g_4 \right]$$

This week(10/12/2013)

Ordinary Differential Equations: Initial Value problems (part II)

- Practical use of Runge-Kutta methods.
- Why do we need an adaptive stepsize?
- Methods for error estimate.
- Implementing a simple Runge-Kutta method with **adaptive stepsize**.

Using Runge-Kutta Methods in Practice:

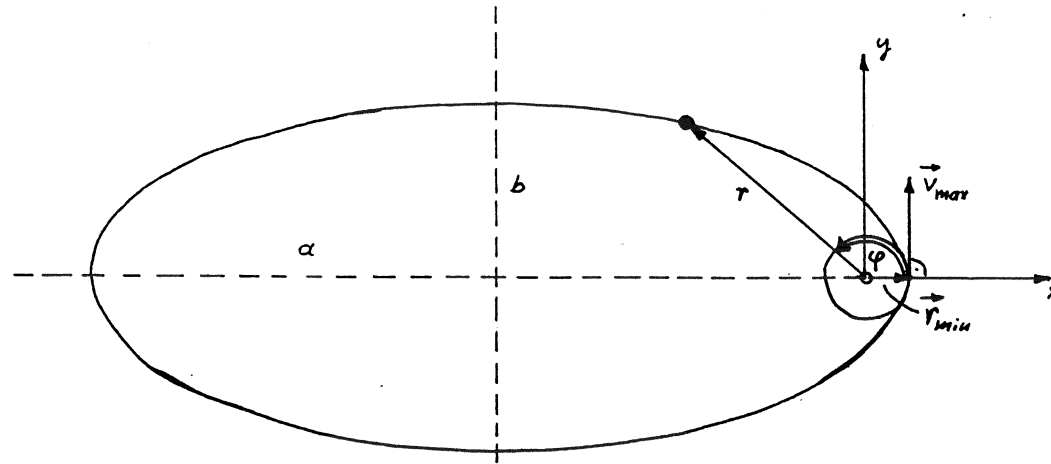
Calculating a single Runge-Kutta step is of not much help. In practice, if we want to integrate a differential system, we want to perform *several steps one* after the other:

$$\hat{y}_i(x_0 + h) \equiv \hat{y}_{i,1} = y_i(x_0) + h \cdot \sum_{j=1}^p c_j g_{i,j}(x_0; y_{1,0}, \dots, y_{n,0})$$

$$\hat{y}_i(x_0 + 2h) \equiv \hat{y}_{i,2} = \hat{y}_{i,1} + h \cdot \sum_{j=1}^p c_j g_{i,j}(x_0 + h; \hat{y}_{1,1}, \dots, \hat{y}_{n,1})$$

In the first step, the initial values are known exactly, and given by the boundary conditions. For all other steps, the initial values are given by previous R-K moves, and thus known only approximately.

A practical Example: trajectory of a satellite



Hypotheses:

- Starting velocity of the satellite: v_{max} .
- The earth is an homogeneous sphere of radius r_{min} .
- No influence of the atmosphere.
- Influence of other celestial body is negligible.

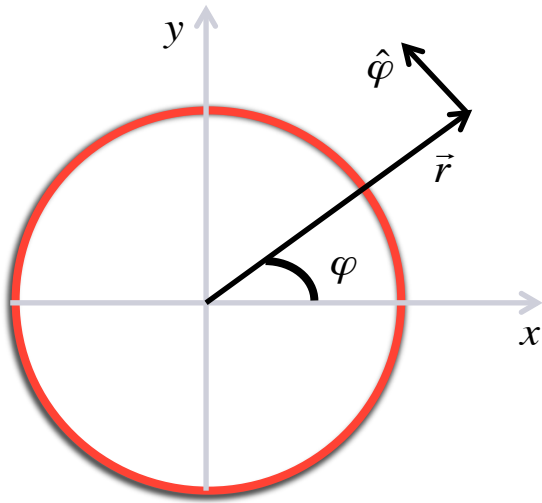
$$\mathbf{F} = -\frac{\gamma M}{r^3} \mathbf{r}$$

Table 1: Comparison of relative acceleration (in g 's) for a satellite orbiting the Earth at an altitude of 500 km.

	Mean distance ($\times 10^9$ m)	Mass ($\times 10^{24}$ kg)	Mass (Earth = 1)	Acceleration in g 's on 500 km satellite
Sun		1989100.0	332948.0	6.1×10^{-4}
Mercury	57.9	0.3302	0.055	2.7×10^{-10}
Venus	108.2	4.8690	0.815	1.9×10^{-8}
Earth	149.6	5.9742	1.000	0.86
Mars	227.9	0.64191	0.107	7.1×10^{-10}
Jupiter	778.3	1898.8	317.833	3.3×10^{-8}
Saturn	1429.4	568.5	95.159	2.4×10^{-9}
Uranus	2875.0	86.625	14.500	7.9×10^{-11}
Neptune	4504.4	102.78	17.204	3.7×10^{-11}
Moon	0.3844 (Earth-Moon)	0.073483	0.012	3.5×10^{-6}
Earth Oblateness				1.0×10^{-3}

- Notes: 1. Table 1 follows Table 1.2-1 in *Fundamentals of Astrodynamics* by Bate, Mueller and White, 1971. The values in columns 2 and 3 (mean distances and masses) are taken from the *Explanatory Supplement to the Astronomical Almanac*, edited by P. K. Seidelmann, U.S. Naval Observatory, Washington, D.C., 1992.
2. The Earth is not a spherical body with homogeneous mass density; it is actually slightly pear shaped with an equatorial bulge and variable mass density. Treating the Earth as an homogeneous spherical body will induce small errors in calculated accelerations due to gravitational attraction and this error is modelled by the Earth Oblateness value shown in column 5 of Table 1.

Polar coordinates:



$$\mathbf{r} = (x, y) = r(\cos \varphi, \sin \varphi) = r\hat{\mathbf{r}},$$

$$\dot{\mathbf{r}} = (\dot{x}, \dot{y}) = \dot{r}(\cos \varphi, \sin \varphi) + r\dot{\varphi}(-\sin \varphi, \cos \varphi) = \dot{r}\hat{\mathbf{r}} + r\dot{\varphi}\hat{\boldsymbol{\varphi}},$$

$$\ddot{\mathbf{r}} = (\ddot{x}, \ddot{y}) = \ddot{r}(\cos \varphi, \sin \varphi) + 2\dot{r}\dot{\varphi}(-\sin \varphi, \cos \varphi) + r\ddot{\varphi}(-\sin \varphi, \cos \varphi) - r\dot{\varphi}^2(\cos \varphi, \sin \varphi) =$$
$$(\ddot{r} - r\dot{\varphi}^2)\hat{\mathbf{r}} + (r\ddot{\varphi} + 2\dot{r}\dot{\varphi})\hat{\boldsymbol{\varphi}} = (\ddot{r} - r\dot{\varphi}^2)\hat{\mathbf{r}} + \frac{1}{r} \frac{d}{dt} (r^2\dot{\varphi}) \hat{\boldsymbol{\varphi}}$$

Equations of motion:

$$\mathbf{F} = -\frac{\gamma M}{r^3} \mathbf{r}$$

$$\ddot{\mathbf{r}} = -\frac{\gamma M_{\text{earth}}}{r^3} \mathbf{r}$$

$$\gamma = 6.67 \cdot 10^{-11} \text{ m}^3 / \text{kg s}^2$$

$$M_{\text{earth}} = 5.977 \cdot 10^{24} \text{ kg}$$

Expressing \mathbf{r} in **polar coordinates** we get:

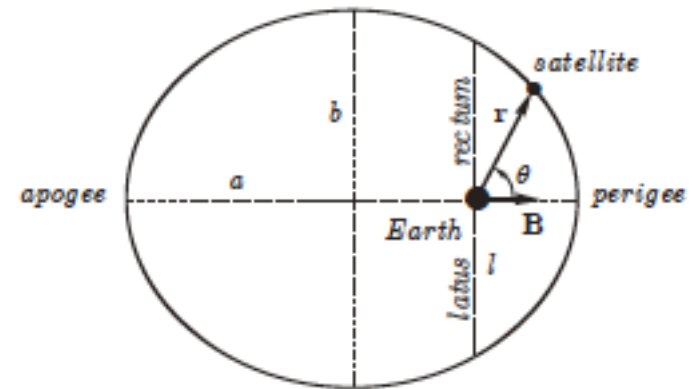
$$\ddot{r} = r\dot{\varphi}^2 - \frac{\gamma M_{\text{earth}}}{r^2}$$

Centrifugal acceleration.

$$\ddot{\varphi} = -2\dot{r}\dot{\varphi}$$

Coriolis acceleration

$$T = 2\pi \sqrt{\frac{a^3}{\gamma M}}$$

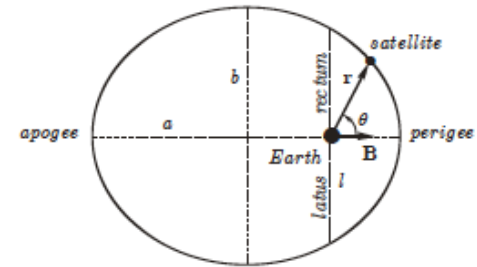


We now want to reduce a 2x2 2nd order system into a 4x4 1st order system, using the substitutions:

$$r \rightarrow y_1 \quad \varphi \rightarrow y_2 \quad \dot{r}=y_3 \quad \dot{\varphi}=y_4$$

We obtain:

$$\begin{cases} \dot{y}_1 = y_3 \\ \dot{y}_2 = y_4 \\ \dot{y}_3 = y_1 y_4^2 - \frac{\gamma M}{y_1^2} \\ \dot{y}_4 = -\frac{2y_3 y_4}{y_1} \end{cases}$$



With the initial conditions (velocity and distance from the earth at the *perigee*):

$$y1(0) = r_{\min} = 6.37 \cdot 10^6 m$$

$$y2(0) = 0$$

$$y3(0) = 0$$

$$y4(0) = \frac{v_{\max}}{r_{\min}} = 58.29527 \text{rads}^{-2}$$

$$v_{\max} = 10.4 \cdot 10^3 \text{ms}^{-1}$$

Numerical Solution (Runge-Kutta): effect of the stepsize

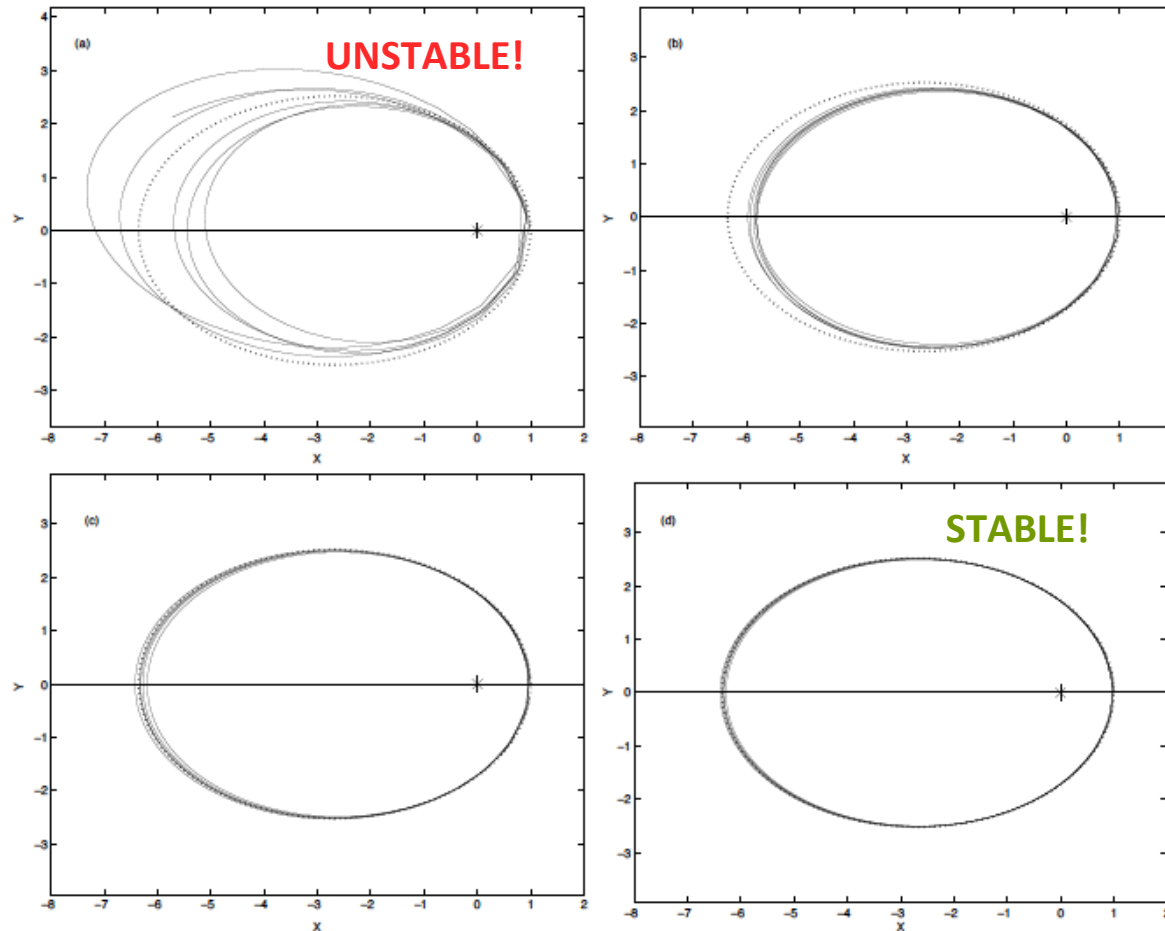


Figure 5.5: Stability test for RUNGETEST. The constant stepsizes have been chosen as follows: (a) $h = 1/50$, (b) $h = 1/60$, (c) $h = 1/70$, (d) $h = 1/80$ of the revolution period. The star indicates the centre of the earth, and the dotted line the exact analytical trajectory of the satellite.

How to estimate the error of a Runge-Kutta step

(and choose the optimal step size):

Using the Runge-Kutta method with arbitrary step size can lead to severe divergences (and repeating the whole calculation of the trajectory with different stepsizes is not an economical solution). To **estimate the error** connected with a single Runge-Kutta step, we can use the formula for the Lagrange remainder of a fourth-order Taylor series:

$$E_V(h) = \frac{h^5}{5!} \left[y_i^{(5)}(x) \right]_{x=\xi}, \quad x_0 \leq \xi \leq x_0 + h$$

$$\text{i.e. } E_V(h) = C(h) \cdot h^5$$

$E_V(h)$ is by definition the error between the exact value of the solution and the corresponding approximate solution in x_0+h :

$$E_V(h) = y_i(x_0 + h) - \hat{y}_i(x_0 + h)$$

A good estimate of $E_V(h)$ is given in practice between the value of y obtained with a single R-K step of size h , and two steps of size $h/2$:

$$E_V(h) \approx \hat{y}(x_0 + h) - \hat{y}(x_0 + 2 \frac{h}{2})$$

PROOF

$$y(x_0+h) - \hat{y}(x_0+h) = c(h) \cdot h^5$$

3

$$y(x_0+2 \cdot \frac{h}{2}) - \hat{y}(x_0+2 \cdot \frac{h}{2}) = 2 \cdot c(\frac{h}{2}) \cdot (\frac{h}{2})^5$$

HYP: $c(h) = \text{const} = c$

$$\Rightarrow \text{not } \hat{y}(x_0+2 \cdot \frac{h}{2}) - \hat{y}(x_0+h) = c \cdot h^5 \left[1 - 2 \cdot \frac{h^5}{32} \right] = c \cdot h^5 \cdot \frac{15}{16}$$

$$\Rightarrow \hat{y}\left(\frac{h}{2}\right) - \hat{y}(h) = c \cdot h^5 \cdot \frac{15}{16} = \varepsilon_v(h) \Rightarrow \boxed{\varepsilon_v(h) = \frac{16}{15} \cdot \left[\hat{y}\left(\frac{h}{2}\right) - \hat{y}(h) \right]}$$

DEF


$$\varepsilon_v(h_{\text{ideal}}) = \varepsilon^*$$

$$c \cdot h_{\text{ideal}}^5 \cdot \frac{15}{16} = \varepsilon^*$$

given an arbitrary h : $\varepsilon_v(h) = \frac{15}{16} \cdot c \cdot h^5$

$$\frac{\varepsilon_v(h)}{\varepsilon^*} = \frac{h^5}{h_{\text{ideal}}^5}$$

$$\Rightarrow \boxed{h_{\text{ideal}} = h \cdot \left(\frac{\hat{y}\left(\frac{h}{2}\right) - \hat{y}(h)}{\varepsilon} \right)^{\frac{1}{5}}}$$



This also gives a practical strategy to estimate the ideal Runge-Kutta *stepsize*, given a required accuracy threshold ε :

$$h_{ideal} = h \cdot \left(\frac{\hat{y}(h/2) - \hat{y}(h)}{\varepsilon} \right)^{-1/5}, \quad \hat{y}(h) = y(x_0 + h) \quad \text{and} \quad \hat{y}(h/2) = y(x_0 + h/2)$$

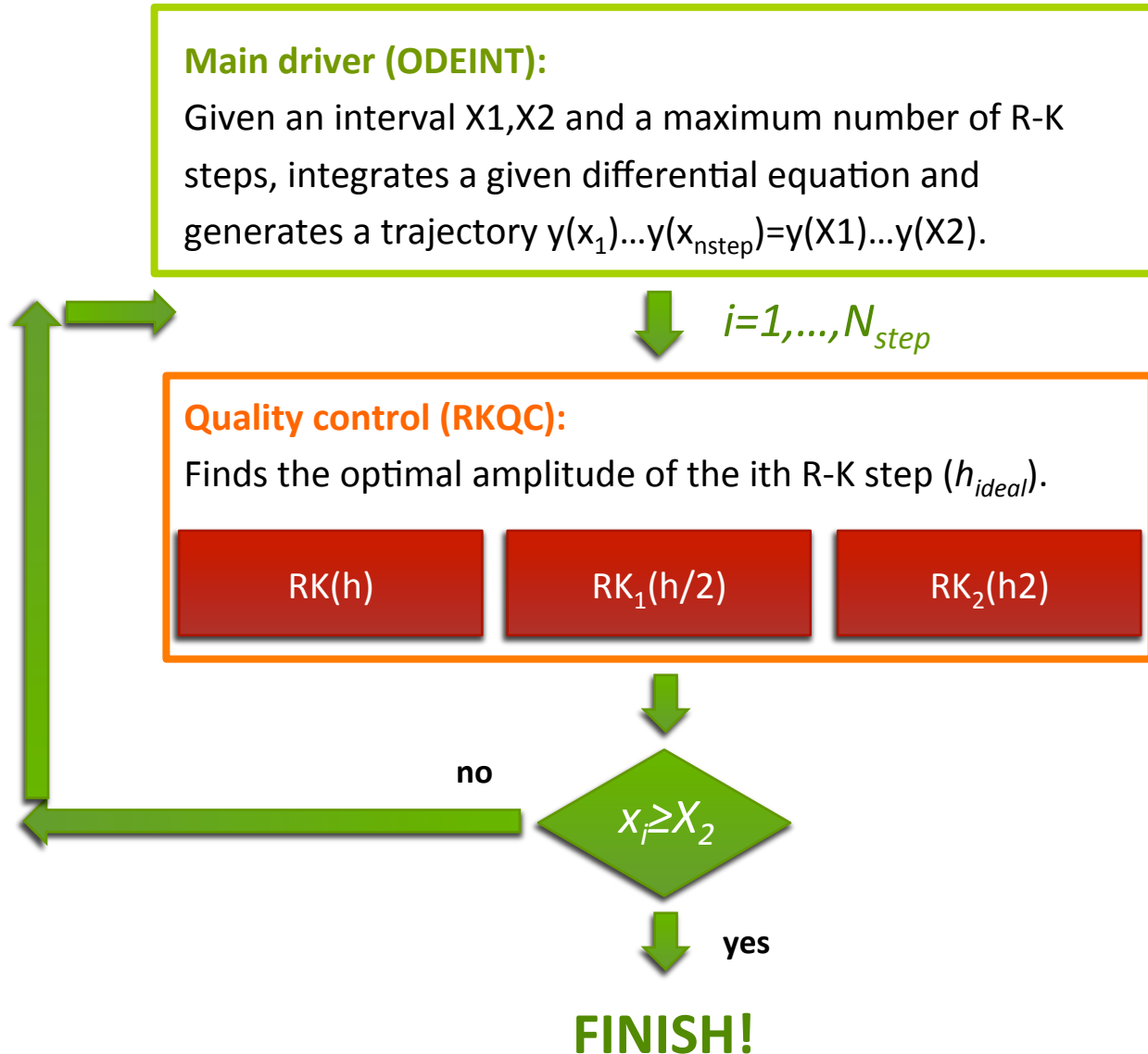
To implement this method in practice, one chooses h and calculates $E_V(h)$ with a standard R-K routine:

$$E_V(h) \approx \hat{y}(h) - \hat{y}\left(\frac{h}{2}\right)$$

If $E_V(h)$ is smaller than ε , the R-K move is accepted with h .

If $E_V(h)$ is larger than ε , the R-K move is refused, h is reduced, and $E_V(h)$ is evaluated again.

A suite for integrating differential equations using the R-K method:



Basic Building Block RK4: single Runge-Kutta move

RK(h)

Structure chart 28 — RK4(Y,G1,N,X,H,YOUT)

I=1(1)N
Y1(I):=Y(I)+H/2.0*G1(I)
XTEMP:=X+H/2.0
DERIVS(XTEMP,Y1,G2)
I=1(1)N
Y2(I):=Y(I)+H/2.0*G2(I)
DERIVS(XTEMP,Y2,G3)
I=1(1)N
Y3(I):=Y(I)+H*G3(I)
XTEMP:=X+H
DERIVS(XTEMP,Y3,G4)
I=1(1)N
YOUT(I):=Y(I)+H/6.0*(G1(I)+2*(G2(I)+G3(I))+G4(I))

$$\hat{y}(x_0 + h) = y(x_0) + h \left[\frac{1}{6} g_1 + \frac{1}{3} g_2 + \frac{1}{3} g_3 + \frac{1}{6} g_4 \right]$$

$$g_1 = f(x_0, y_0)$$

$$g_2 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} g_1\right)$$

$$g_3 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} g_2\right)$$

$$g_4 = f(x_0 + h, y_0 + h g_3)$$

5.5.2 The program RKQC.

RKQC (Runge-Kutta Quality Control): This program performs the quality control of a single Runge-Kutta move and the corresponding stepsize adaptation.

INPUT parameters:

Y(): \hat{y}_i 's of the last Runge-Kutta move =
initial values for the next step.

F(): Array of the corresponding f_i -values.

N: Number of the equations of the system.

X: Abscissas of the current initial value.

HTRY: Amplitude of the interval for the next step.

EPS: Required *relative* precision.

YSCAL(): Scaling factors for the next precision evaluation.

OUTPUT parameters:

Y(): New \hat{y}_i 's.

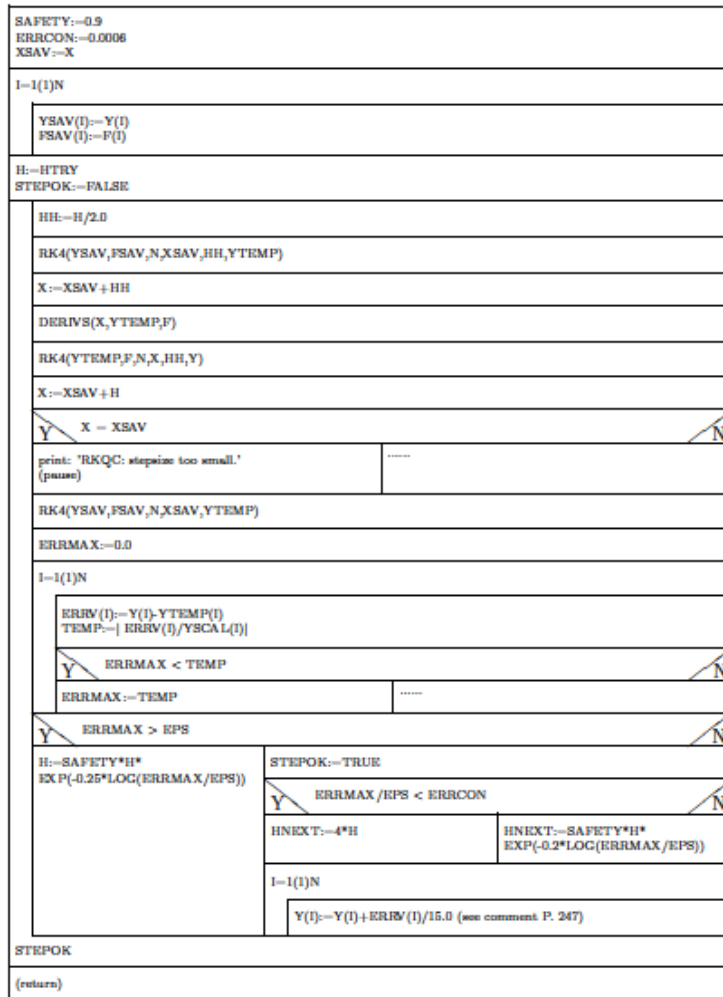
F(): Array of the corresponding f_i -values.

X: New abscissas.

HNEXT: Proposed stepsize for the next iteration.

Quality control of Runge-Kutta step: RKQC

Structure chart 27 — RKQC(Y,F,N,X,HTRY,EPS,YSCAL,HNEXT)



Quality control (RKQC):

Finds the optimal amplitude of the ith R-K step

RK(h)

RK₁(h/2)

RK₂(h/2)

$$E_v(h) \approx \hat{y}(h) - \hat{y}\left(\frac{h}{2}\right)$$

5.5.3 The programs RK4 and DERIVS.

RK4 (Runge-Kutta 4) performs the evaluation of a single Runge-Kutta move according to equations (5.18) and (5.19):

INPUT parameters:

Y(): Initial value for the current Runge-Kutta move.

G1(): Array of the f_i values at the current initial point.

N: Number n of equations in the differential system of equations.

X: Abscissa of the initial point.

H: Stepsize for the current Runge-Kutta move.


OUTPUT parameters:

YOUT(): Runge-Kutta approximate value \hat{y}_i in the point $X+H$.

DERIVS:

This program, which is called from all three programs in the system, is provided by the user and contains the definition of the functions f_i . In C it would read for example:

```
void derivs(double x, double y[], double f[])
{
    f[1]= .....; // entspricht f_{1}(x,y1,y2,...)
    f[2]= .....; // entspricht f_{2}(x,y1,y2,...)
    .
    .
    f[n]= .....; // entspricht f_{n}(x,y1,y2,...)
}
```

ODEINT (Ordinary Differential Equations INTegrator) is the 'driver program' of the set of programs:

INPUT parameters:

YSTART(): Vector y_0 with the initial values of the system of differential equations.

N: Number n of equations in the system.

X1, X2: Start and end point of the integration interval.

EPS: Required *relative* precision (see the following remarks).

HSTART: Guessed value for the stepsize of the Runge-Kutta process.

HMIN: Minimum value for the stepsize.

NPTMAX: Maximum number of points that can be saved in the arrays **XX** and **YY**.

OUTPUT parameters:

NWERTE: Number of stored points.

XX(): Abscissas of the points.

YY(,): Ordinates of the solution:
first index = index of the function,
second index = label of the abscissas.

Main driver: integrate the differential equation over the X1,X2 interval

Structure chart 26 — ODEINT(YSTART,N,X1,X2,EPS,HANF,HMIN,NSTMAX,NWERTE,XX,YY)

TINY:=-1.0E-30	
X:=-X1	
XX(1):-X	
H:-HANF	
I=-1(1)N	
Y(I):-YSTART(I)	
YY(1,1):-Y(I)	
NSTP=-1(1)NSTMAX-1	
DERIVS(X,Y,F)	
I=-1(1)N	
YSCAL(I):- Y(I) + F(I)*H + TINY	
Y	N
X+H > X2	
H:-X2-X	-----
RKQC(Y,F,N,X,H,EPS,YSCAL,HNEXT)	
XX(NSTP+1):-X	
I=-1(1)N	
YY(1,NSTP+1):-Y(I)	
Y	N
X ≥ X2	
I=-1(1)N	-----
YSTART(I):-Y(I)	
NWERTE:-NSTP+1 (return)	
Y	N
HNEXT < HMIN	
print: 'stepsize smaller than HMIN' (pause)	H:-HNEXT
print: 'ODEINT: more than NSTMAX points' NWERTE:-NSTMAX (return)	

Main driver (ODEINT):

Given an interval X1,X2 and a maximum number of R-K steps, integrates a given differential equation and generates a trajectory $y(x_1) \dots y(x_{nstep}) = y(X1) \dots y(X2)$.

Effect of the stepsize adaptation:

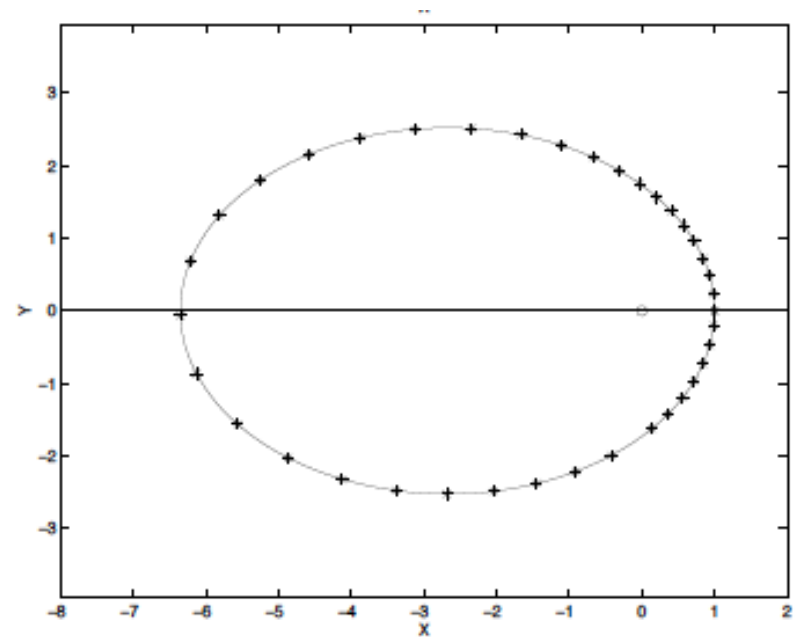
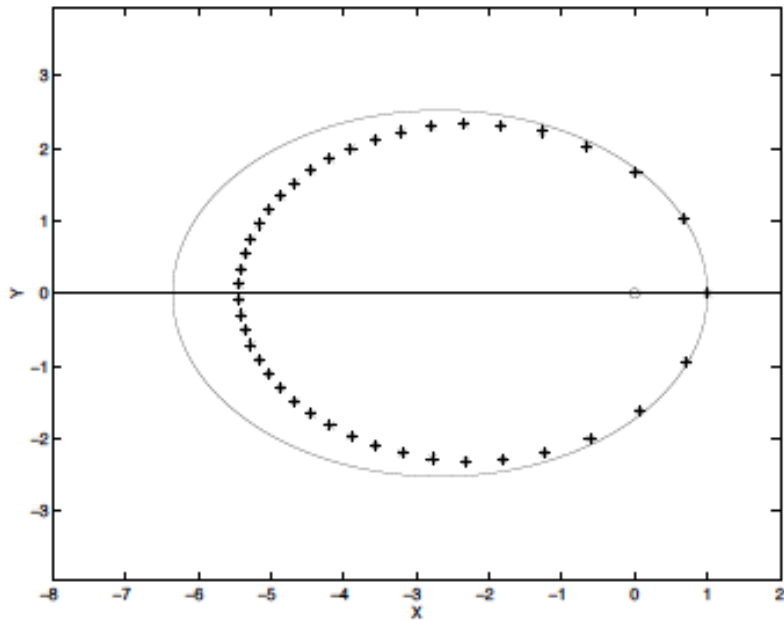


Figure 5.6: Efficiency of the stepsize adaptation in the 'satellite problem'. Comparison of the exact elliptical trajectory (full line) with the numerical values (stars). Above: Runge-Kutta *without* stepsize adaptation; Below: Runge-Kutta *with* stepsize adaptation.

This week(10/12/2013)

Ordinary Differential Equations: Initial Value problems (part II)

- Practical use of Runge-Kutta methods.
- Why do we need an adaptive stepsize?
- Methods for error estimate.
- Implementing a simple Runge-Kutta method with **adaptive stepsize**.