Introduction to computer physics (Spurzem, Klessen)

Robin Heinemann

6. Mai 2018

Inhaltsverzeichnis

1 ODE's Ordinary differential equations

1.1 Gravitational 2-body problem

Newton's equation for the relative motion of two bodies under their mutual gravitational force is given by

$$\vec{r} = \vec{x}_1 - \vec{x}_2 \ddot{\vec{r}} = -\frac{GM}{r^2}\frac{\vec{r}}{r}$$
$$M = m_1 + m_2$$
$$G = 6.67 \times 10^{-8}\,\mathrm{cm}^3\,\mathrm{g}^{-1}\,\mathrm{s}^{-2}$$

m_1 gets accelerated by m_2, respectivly m_2 by m_1:

$$\vec{F}_{12} = m_1\vec{a}_1$$
$$\vec{F}_{21} = m_2\vec{a}_2$$

Newton 3 $\implies$

$$\vec{F}_{12} = -\vec{F}_{21}$$
$$\implies m_1\vec{a}_1 = -m_2\vec{a}_2 \implies \vec{a}_2 = \frac{m_1}{m_2}\vec{a}_1$$
$$|F_{21}| = |F_{12}| = \frac{Gm_1m_2}{r^2}$$
$$\implies \ddot{\vec{r}} = \vec{a} = \frac{m_1+m_2}{m_1m_2}m_1\vec{a}_1$$
$$\implies \mu\ddot{\vec{r}} = -\frac{Gm_1m_2}{r^2}\frac{\vec{r}}{r}$$

Simplification 1: 2. order ordinary differential equation $\rightarrow 2\times$ first order differential equation.
Construct a system of first order ordinary differential equations:

$$\begin{aligned}\dot{\vec{r}} &= \vec{v} \qquad \vec{v} = \text{ velocity}\\ \dot{\vec{v}} &= -\frac{GM}{r^2}\frac{\vec{r}}{r}\end{aligned}$$

Now look for „symmetries" $\rightarrow$ conserved quatities / properties

1. energy: $E = T + U = \frac{\mu v^2}{2} - \frac{GM}{r}$
2. angular momentum $\vec{L} = \vec{r} \times \vec{p} = \vec{r} \times \mu\vec{v}, \dot{\vec{j}}$
3. Lenz vector (Laplace-Runge-Lenz)

$$\vec{e} = \frac{\vec{v} \times \vec{j}}{GM} - \frac{\vec{r}}{r}$$

related to eccentricity

$$\begin{aligned}\vec{e} \cdot \vec{r} &= \frac{\vec{r}\left(\vec{v} \times \vec{j}\right)}{GM} - \frac{\vec{r}\vec{r}}{r}\\ &= \frac{\vec{j} \cdot (\vec{r} \times \vec{v})}{GM} - r\\ &= j^2 - r\\ \vec{e}\vec{r} &= er\cos(\varphi)\\ \Longrightarrow \quad r(\varphi) &= \frac{j^2/(GM)}{1 + e\cos(\varphi)}\end{aligned}$$

for (closed) elliptical orbits

$$e = \frac{r_a - r_p}{r_a + r_p} \qquad \text{(eccentricity)}$$

second check: look for conserved quantities.
Simplification 3: search for non-dimensional represenation.

$$\begin{aligned}\vec{s} &= \frac{\vec{r}}{R_0}\\ \vec{\omega} &= \frac{\vec{v}}{V_0}\\ \tau &= \frac{t}{T}\\ R_0 &= \text{ arbitrayry radius (say initial separation)}\\ V_0 &= \sqrt{\frac{GM}{R_0}}\\ T_0 &= \frac{R_0}{V_0}\end{aligned}$$

Our set of equations then reads

$$\begin{aligned}\frac{\mathrm{d}\vec{s}}{\mathrm{d}\tau} &= \vec{\omega}\\ \frac{\mathrm{d}\vec{\omega}}{\mathrm{d}\tau} &= -\frac{\vec{s}}{s^3}\end{aligned}$$

For a bound system $(E = T + U < 0)$, we know that the sollution is an ellipse around the coordintae center. This can be used to test the validity and accuracy of the numerical solution.

1.1.1 Elementary numerical solution: Euler Method

The simplest approach to solve this system is to replace the derivatives by first order differential quotients. We discretize the time evolution ito discrete steps i of af fixed width $h = \tau_i - \tau_{i-1}$. We get

$$\frac{\mathrm{d}\vec{s}_i}{\mathrm{d}\tau} = \frac{\vec{s}_i - \vec{s}_{i-1}}{\tau_i - \tau_{i-1}} \mathcal{O}(h)$$

with $\vec{s}_i = \vec{s}(\tau_i)$ desribing the state of the system (location of the body) at step i. The symbol $\mathcal{O}(h)$ denotes that the error we make by this approximation is of linear order fo the step size h. Multiplying with h leads to

$$h\frac{\mathrm{d}\vec{s}_i}{\mathrm{d}\tau} = \vec{s}_i - \vec{s}_{i-1} + \mathcal{O}\big(h^2\big)$$

Applying the same procedure to $\mathrm{d}\vec{\omega}_i/\mathrm{d}\tau$ results in

$$h\frac{\mathrm{d}\vec{\omega}_i}{\mathrm{d}\tau} = -\frac{\vec{s}_i}{s_i^3} + \frac{\vec{s}_{i-1}}{\vec{s}_{i-1}^2} + \mathcal{O}\big(h^2\big)$$

Rearaanging givess

$$\begin{aligned} \vec{s}_i &= \vec{s}_{i-1} + \vec{\omega}_{i-1}h + \mathcal{O}\big(h^2\big) \\ \vec{\omega}_i &= \vec{\omega}_{i-1} + \frac{\vec{s}_{i-1}}{s_{i-1}^3}h + \mathcal{O}\big(h^2\big) \end{aligned}$$

This is the **forward Euler** method. It is an explicit method, because all quantities required to advance the system from time τ_{i-1}, by a discrete step of size h are know at the start of the step. To test the accuracy and validity of the method, we can look at the evolution of the three conserved parameters

- total energy: $E_i = \frac{\omega_i^2}{2} - \frac{1}{s_i}$
- angular momentum: $j_i = \vec{s}_i \times \vec{\omega}_i$
- Laplace-Runge-Lenz vector $\vec{e}_i = \vec{\omega}_i \times (\vec{s}_i \times \vec{\omega}_i) - \vec{s}_i$

The question we ask is whether these quantities remain constant or evolve with time. Or phrasing this diffrently: does our algorithm prevent the accumulation of truncation and integration errors, or do these unavoidable discretization errors add up without bounds. The quantity that is usually lokked at the characterzie this behavior is the relative error at step i compared to the the initial value at 0. For the 2-body problem, the easiest quantity to consider is the relative error in the total energy of the system

$$\varepsilon_i(h) = \frac{|E_i - E_0|}{|E_0|}$$

Question 1: How does the forward euler Method behave? We can speculate about the answer along the following line of reasoning: We need roughly $1/h$ steps to cover one orbit. So despite the fact that each step is second order in h, the global error for one full orbit is expected to be $\mathcal{O}\big(h^2\big) \cdot \mathcal{O}\big(h^{-1}\big) = \mathcal{O}(h)$, that is only linear in step size h.

1.1.2 Higher-Order Schemes: Verlet

From the previous section, we conclude that a first order algorithm usually performes insufficiently. One way to remedy this problem is to develop higer-order integration schemes. The idea of the Verlet method is to Taylor expand the solution and keep terms up to third order, meaning that the error is of forth order. These are acceleration $\vec{a}$ and jerk $\vec{b} = \dot{\vec{a}}$, the change of the acceleration. Let us adopt the notation $h = \Delta t$ for the step size and Taylorexpand the system around the time t forwards by Δt and backwards by $-\Delta t$:

$$\vec{s}(t+\Delta t) = \vec{s}(t) + \vec{\omega}(t)\Delta t + \frac{1}{2}\vec{a}(t)\Delta t^2 + \frac{1}{6}\vec{b}(t)\Delta t^2 + \mathcal{O}\big(\Delta t^4\big)$$
$$\vec{s}(t-\Delta t) = \vec{s}(t) - \vec{\omega}(t)\Delta t + \frac{1}{2}\vec{a}(t)\Delta t^2 - \frac{1}{6}\vec{b}(t)\Delta t^2 + \mathcal{O}\big(\Delta t^4\big)$$

adding these both together leaves us with

$$\vec{s}(t+\Delta t) = 2\,\vec{s}(t) - \vec{s}(t-\Delta t) + \vec{a}(t)\Delta t^2 + \mathcal{O}\big(\Delta t^4\big)$$

Which is a numerical scheme of forth order in space. Note that this only works, because the acceleration only depends on the position $\vec{s}(t)$. The situation becomes more complicated when $\vec{a}(t)$ depends on other parameters, such as the velocity $\vec{\omega}(t)$ as in the case of Lorentz forces for charged particeles in a magnetic field. In general, the order in which we evaluate the different terms becomes important. Note also, that the accuracy in the velocity $\vec{\omega}$ is only $\mathcal{O}\big(\Delta t^3\big)$. More severly, the equation becomes implicit, meaning that information of a future timestep is needed. Implicit schemes typically require an iterative approach to solve.

$$\vec{\omega}(t) = \frac{\vec{s}(t+\Delta t) - \vec{s}(t-\Delta t)}{2\Delta t} - \frac{1}{6}\vec{b}(t)\Delta t^2 + \mathcal{O}\big(\Delta t^3\big)$$

Related to the above approach, and more commonly used, is the velocity Verlet algorithm. It reads

$$\vec{s}(t+\Delta t) = \vec{s}(t) + \vec{\omega}(t)\Delta t + \frac{1}{2}\vec{a}(t)\Delta t^2 + (\mathcal{O})\big(\Delta t^3\big)$$
$$\vec{\omega}(t+\Delta t) = \vec{\omega}(t) + \frac{\vec{a}(t) + \vec{a}(t+\Delta t)}{2}\Delta t + \mathcal{O}\big(\Delta t^2\big)$$

where we Taylor expand the velocity equation to second order and replace the acceleration at t by the average of the acceleration and t and $t + \Delta t$. If the acceleration is a function of position $\vec{s}$ only, this can be solved by adjusting the order with wich the system is integrated:

1. calculate new position
2. update acceleration
3. calculate new velocity

We understand the underlying structur of the Verlet scheme better as combination of diffrent first oder steps, if we introduce half time intervals

1. $\vec{\omega}\left(t + \frac{1}{2}\Delta t\right) = \vec{s}(t) + \frac{1}{2}\vec{a}(t)\Delta t$
2. $\vec{s}(t+\Delta t) = \vec{s}(t) + \vec{\omega}\left(t + \frac{1}{2}\Delta t\right)\Delta t$
3. $\vec{a}(t+\Delta t) = \vec{F}(\vec{s}(t+\Delta t))$
4. $\vec{\omega}(t+\Delta t) = \vec{\omega}\left(t + \frac{1}{2}\Delta t\right) + \frac{1}{2}\vec{a}(t+\Delta t)\Delta t$

The Verlet scheme therefore belongs to the group of semi-implicit Euler methods. In this implemenation it is also closely related to the leap-frog integration scheme that we discuss in the next section, however unlike the leap-frog, it hat $\vec{s}$ and $\vec{\omega}$ defined at the same time.

1.1.3 Higher-Order Schemes: Leap-Frog

Closely related to the Verlet scheme is the leap-frog method. Unlike Verlet, we are giving up on the requivement that location $\vec{s}$ and velocity $\vec{\omega}$ are known at the same time. Instead they are considered perfectly interlaced, hence the name leap frop. Consequently, the method requives infromation from two timesteps for the integration. Introducing agian the notion of half timesteps, the method reads:

1. update location: $\vec{s}\left(t+\frac{1}{2}\Delta t\right) = \vec{s}\left(t-\frac{1}{2}\Delta t\right) + \vec{\omega}(t)\Delta t + \mathcal{O}\left(\Delta t^2\right)$
2. update velocity: $\vec{\omega}(t+\Delta t) + \vec{a}\left(t+\frac{1}{2}\Delta t\right)\Delta t + \mathcal{O}\left(\Delta t^3\right)$

The positions are always updated at half timesteps, while the velocity lives on full timesteps. This approach is called 'drift-kick-drift' implemenation. The order of integration could also be interchanged, then we speak of the 'kick-drift-kick' scheme. Because $\vec{s}$ and $\vec{\omega}$ are never synchronized, the scheme needs to be started and ended at times when input is received or output is written, for which positions and velocities need to be in sync. To achive that, the system is typically initiated with a half step using the forward Euler method (or any other suitable approach). The reduced accuracy for this half step is typically not an issue, because most of the time integration is then achieved with second order leap-frog. The fact that leap-frog integrators are of second order accuracy may seem surprising, given that they at first sight correspond to a Taylor expansion to first order. However the highr order is achived by the asynchronicity of the integraiton. The reason it works lies in the symplectic nature of the integration scheme, that reflects the symplectic symmetry of Hamiltonian mechanics. Symplectic integration schemes conserve the total energy of the system extremely well, because they are fully time reversible. To see that, integrate the sysetm from state $\left(\vec{s}, \vec{\omega}_{i-\frac{1}{2}}\right)$ to $\left(\vec{s}_{i+1}, \vec{\omega}_{i+\frac{1}{2}}\right)$, and return:

$$
\begin{aligned}
\vec{s}_{\text{final}} &= \vec{s}_{i+1} - \vec{\omega}_{i+\frac{1}{2}}\Delta t \\
&= \left(\vec{s}_i + \vec{\omega}_{i+\frac{1}{2}}\Delta t\right) - \vec{\omega}_{i+\frac{1}{2}}\Delta t = \vec{s}_i \\
\vec{\omega}_{\text{final}} &= \vec{\omega}_{i+\frac{1}{2}} - \vec{a}_i\Delta t \\
&= \left(\vec{\omega}_{i-\frac{1}{2}} + \vec{a}_i\Delta t\right) - \vec{a}_i\Delta t = \vec{\omega}_i
\end{aligned}
$$

This would not work for other methods, such as the forward Euler, as this uses quantities that are only known at the curretn step. For example, fo find the position at $(i+1)$ the velocity $\vec{\omega}_i$ is used, however when going back from $(i+1)$ to i then $\vec{\omega}_{i+1}$ is taken. Because typically $\vec{\omega}_{i+1} \neq \vec{\omega}_i$, we never precisely come back to the original startic position. The leap-frog integrator is related to the Verlet method. Compare the following two approaches:
Verlet:

$$
\begin{aligned}
\vec{\omega}_{i+\frac{1}{2}} &= \vec{\omega}_i + \vec{a}_i\frac{\Delta t}{2} \\
\vec{s}_{i+1} &= \vec{s}_i + \vec{\omega}_{i+\frac{1}{2}}\Delta t \\
&= \vec{s}_i + \vec{\omega}_i\Delta t + \frac{1}{2}\vec{a}_i\Delta t^2 \\
\vec{\omega}_{i+1} &= \vec{\omega}_{i+\frac{1}{2}} + \vec{a}_i\frac{\Delta t}{2} \\
&= \vec{s}_i + \frac{1}{2}(\vec{a}_i + \vec{a}_{i+1})\Delta t
\end{aligned}
$$

Leap-frog:

$$
\begin{aligned}
\vec{\omega}_{i+\frac{1}{2}} &= \vec{\omega}_{i-\frac{1}{2}} + \vec{a}_i\Delta t \\
\vec{s}_{i+1} &= \vec{s}_i + \vec{v}_{i+\frac{1}{2}}\Delta t
\end{aligned}
$$

1.2 Runge-Kutta Integration

Ordinary differential equations are a very common way to mathematically formulate the dynamical evolution of physical systems. Frequently the problem at hand corresponds to an initial value problem, such as the gravitational dynamics discussend above. Once the inital state of the system is fully specified, by providing initial values for the functions to solve for as well as for their derivatives, we can integrate to obtain a unique solution. Other typical situations require us to provide boundary conditions, for example when we want to calculate the potential that corresponds to specific mass or charge distributions, or when we need to compute eigenvalues and eigenfunctions of differential operators. As mentioned earlier we note that every system of ordinary differential equations of higher order can be reduced to a system of first order ordinary differential equations. Consider the following ordinary differential equation of order n

$$y^{(n)}(x) = f\left(y^{(n-1)}, y^{(n-2)}, \dots, y^{(1)}, y, x\right)$$

where $y(x)$ is a function of x and $y^{(n)}(x)$ is its n-th derivative with respect to x. The function f describes, how $y^{(n)}(x)$ depends on the lower-oder derivatives of $y(x)$ as wenn as on x explicitly. Introduce the following definitons and abbreviations:

$$\begin{aligned} y^{(n)}(x) &= f\left(y^{(n-1)}, y^{(n-2)}, \dots, y'', y', y, x\right) \\ y^{(n)}(x) &= \frac{\mathrm{d}^n}{\mathrm{d}x^n} y(x) \end{aligned}$$

define

$$\begin{aligned} y_0 &= y \\ y_1 &= y' \\ &\vdots \\ y_k &= y^{(k)} = y'_{k-1} \forall k = 0, \dots, n-1 \end{aligned}$$

Then we obtain the following set of first order ordinary differential equations that is equivalent

$$\begin{aligned} y'_0 &= y_1 \\ y'_1 &= y_2 \\ &\vdots \\ y'_{n-2} &= y'_{n-1} \\ y'n - 1 &= f(y_{n-1}, y_{n-2}, \dots, y_2, y_1, y_0, x) \end{aligned}$$

Introducing vector notation, we can write in short

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

Existence: Lipschitz Condition The existence and uniqueness of a solution of the initial value problem $\vec{y}(x_0) = \vec{y}_0$ in the vicinity of $(\vec{y}_0, x_0)$ is guaranteed, if

$$\left\| \vec{f}(\vec{y}, x) - \vec{f}(\vec{z}, x) \right\| \leq \lambda \left\| \vec{y} - \vec{k} \right\|$$

for all $\vec{y}$, $\vec{z}, x$ in vicinity of $(\vec{y}_0, x_0)$ and where $\lambda > 0$ is a real number and $\|\dots\|$ is an arbitrary vector norm. Intuitively the Lipschitz contiuity means that the absolute value of the slope of the line connecting the two points is bounded by $\lambda > 0$. A stronger (more restrictive) condition is that $\vec{f}$ is continuous in the region of interest and sufficiently often differentiable. Even stronger: $\vec{f}$ is analytic, meaning that it is infinitely differentiable.

1.2.1 Integration through Taylor Expansion

In the following, let us consider the simple one-dimensional function $y(x)$, which gives rise to the first order ordinary differential equation

$$y' = f(y,x) \qquad \text{with initial value} \quad y(x_0) = y_0$$

Its solutions are trajectories in the two-dimensional space (x, y). We consider the integration along discrete coordinate values $x_n = x_0 + nh$

$$\begin{aligned} x_{n+1} &= x_n + h = x_0 + nh \\ y(x_{n+1}) &= y_{n+1} = ? \end{aligned}$$

Let us now Taylor expand this to find a suitable integration scheme

$$\begin{aligned}
y(x+h) &= y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \frac{3}{8}y^{(3)}(x) + \wr(h^4) \\
y'(x) &= f(y(x), x) \\
y''(x) &= \frac{\mathrm{d}^2}{\mathrm{d}x^2}(y) & = \frac{\mathrm{d}}{\mathrm{d}x}f(y(x),x) = \frac{\mathrm{d}y}{\mathrm{d}x}\frac{\mathrm{d}f}{\mathrm{d}y}\Big|_{(} \\
y^{(3)}(x) &= \frac{\mathrm{d}^3}{\mathrm{d}x^3}y(x) = \frac{\mathrm{d}}{\mathrm{d}x}(ff_y + f_x) \\
&= \frac{\mathrm{d}}{\mathrm{d}y}(ff_y + f_x)\frac{\mathrm{d}y}{\mathrm{d}x} + \frac{\mathrm{d}}{\mathrm{d}x}(ff_y + f_x) = (f_yf_y + ff_{yy} + f_{yx})f + f_xf_yff_{xy}f_{xx} \\
f_x &= \frac{\mathrm{d}f}{\mathrm{d}x} \\
f_y &= \frac{\mathrm{d}f}{\mathrm{d}y} \\
f_{xy} &= \frac{\mathrm{d}^2 f}{\mathrm{d}x\mathrm{d}y} = \frac{\mathrm{d}^2 f}{\mathrm{d}y\mathrm{d}x} = f_{yx} \\
y(x+h) &= y(x) + hf + \frac{h^2}{2}(ff_y + f_x) + \frac{h^3}{6}\left(f_{xx} + 2ff_{xy} + f^2f_{yy}ff_y^2 + f_xf_y\right) + \wr(h^4)
\end{aligned}$$

note

$$\begin{aligned}
y^{(n)}(x) &= \frac{\mathrm{d}^n}{\mathrm{d}x^n}y(x) = \frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}}f(y(x),x) = \left(f\frac{\partial}{\partial y} + \frac{\partial}{\partial x}\right)^{n-1} f = \left(f\frac{\mathrm{d}}{\mathrm{d}y} + \frac{\mathrm{d}}{\mathrm{d}x}\right)^{n-1} f \\
&= ff_y^{(n-1)} + f_x^{(n-1)}
\end{aligned}$$

This aproach can be used to construct methods of arbitrary high accuracy and order. However, the problem ist that the higher order derivatives need to be constructed recursively. This is normally very complicated and slow. Note: The Euler scheme, simply corresponds to the first order Taylor expansion:

$$y_{i+1} = y_i + h \cdot f(y_i, x_i)$$

These schemes are so-called one-step methods, because they propagate the solution through the integration interval h in one single step. Because it is often complicated to construct higher-order derivatives, it is often a better idea to approximate or substitute the derivatives through a smart combination of function evaluation at specific locations within one integration interval h. This is the central idea of the Runge-Kutta integration method. To see how this may work, let us consider

$$y(x_{i+1}) = y_{i+1} = y_i + hf(y_i, x_i)\frac{h^2}{2}(f_x(y_i,x_i) + f_y(y_i,x_i)f(y_i,x_i)) + \mathcal{O}(h^3)$$

which we use to approximate $y(x_{i+1})$. Alternatively to the Taylor expansion, we could try to obtain an estimate for $y(x_{i+1})$ by direct integration:

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(y(x), x)\mathrm{d}x$$

This integral can be solved by numerical quadrature, in which the function is evaluated at a finite set of integration points within the interval $[x_i, x_{i+1}]$. Let N be the number of integration points $\xi_1, \ldots, \xi_N \in [x_i, x_{i+1}]$, for example $\xi_{ij} = x_i + a_j h$ with $0 \leq a_j \leq 1$, then we obtain

$$y_{i+1} = y_i + h\sum_{j=1}^{N} c_j f(y(\xi_{ij}), \xi_{ij})$$

The coefficient c_j are weight functions that determine the contribution of the function evaluation at x_{ij} to the quadrature integration. The values of c_j as well as the best choice of a_j can be obtained by direct comparison with the Taylor expansion. Let us demonstrate this procedure by looking at the example of the derivation of the second Runge-Kutta integration scheme. For that consider again

$$y_{i+1} = y_i + hf + \frac{h^2}{2}(f_x + f_y f) + \mathcal{O}(h^3)$$

where we evaluate f and the derivatives f_x and f_y at the position (y_i, x_i). For the comparison, we adopt the following approach

$$y(x_{i+1}) = y_{i+1} + hb_i f(y_i, x_i) + hb_2 f(y_i + ha_{21} f(y_i, x_i), x_i + c_2 h)$$

We Taylor expand the last term to first order,

$$f(y_i + ha_{21} f, x_i + c_2 h) = f + c_2 h f_x + ha_{21} f f_y$$

where again all terms are computed at (y_i, x_i) unless otherwise stated. Put together, we obtain

$$\begin{aligned} y_{i+1} &= y_i + hb_1 f + hb_2(f + hc_2 f_x + ha_{21} f f_y) \\ &= y_i + h(b_1 + b_2)f + hc_2 a_{21}(f_x + f f_y) \quad \text{if we set} \quad c_2 = a_{21} \end{aligned}$$

If we compare the coefficients, then we obtain

$$b_1 + b_2 = 1, \qquad b_2 a_{21} = \frac{1}{2}$$

recalling again that we have already set $c_2 = a_{21}$. Because we have two equations for three unknowns, we have one degree of freedomand choose $b_2 = \gamma$. Finally, we obtain the following integration formula

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h\left((1-\gamma)f(y_i, x_i) + \gamma f\left(y_i + \frac{h}{2\gamma} f(y_i, x_i), x_i + \frac{h}{2\gamma}\right)\right) \end{aligned}$$

Depending on the choice of γ, we obtain several integration schemes.
$\gamma = 0$: This is the standard forward Euler scheme. The parameters are $b_1 = 1$ and $b_2 = 0$ a_{21} and c_2 are not needed. It simply corresponds to a Taylor expansion to first order

$$\begin{aligned} k_1 &= hf(y_i, x_i) \\ x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + k_1 + \mathcal{O}(h^2) \end{aligned}$$

$\gamma = 1/2$: This is the classical second-order Runge-Kutta method, sometimes simply called RK2. It hat $b_1 = b_2 = 1/2$ and $a_{21} = c_2 = 1$, and reads:

$$\begin{aligned}
k_1 &= hf(y_i, x_i)\\
k_2 &= hf\left(y_i + \frac{k_1}{2}, x_i + \frac{h}{2}\right)\\
x_{i+1} &= x_i + h\\
y_{i+1} &= y_i + \frac{1}{2}(k_1 + k_2) + \mathcal{O}(h^3)
\end{aligned}$$

The RK2 method evaluates the integration interval h at two locations. First, it uses the slope obtained at the starting point (y_i, x_i) to obtain an estimate for the function the midpoint. Finally, the arithmetic mean of both slopes is used to actually advance the system across the entire interval h.
$\gamma = 1$: This is the classical midpoint rule, also called Euler-Cauchy method. The parameters are $b_1 = 0, b_2 = 1$ and $a_{21} = c_2 = 1/2$. It takes the prediction of the slope f at the midpoint of the integration interval to advance the system across the ful interval:

$$\begin{aligned}
k_1 &= hf(y_i, x_i)\\
k_2 &= kf\left(y_i + \frac{k_1}{2}, x_i + \frac{h}{2}\right)\\
x_{i+1} &= x_i + h\\
y_{i+1} &= y_i + k_2 + \mathcal{O}(h^3)
\end{aligned}$$

Sometimes, this method is called RK2. The trial step at the midpoint helps to cancle higher order terms. The RK2 and midpoint rule use two function evaluations in the interval h to advance the solution.

1.2.2 Forth Order Runge-Kutta Method

The most commonly used method is RK4, the forth order Runge-Kutta scheme. It reaches forth order accuracy by evaluating the function $f(u, x)$ four times in within one step. It is defined in the following way:

$$\begin{aligned}
k_1 &= hf(y_i, x_i)\\
k_2 &= hf\left(y_i + \frac{k_1}{2}, x_i + \frac{h}{2}\right)\\
k_3 &= hf\left(y_i + \frac{k_2}{2}, x_i + \frac{h}{2}\right)\\
k_3 &= hf(y_i + k_3, x_i + h)\\
x_{i+1} &= x_i + h\\
y_{i+1} &= y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(h^5)
\end{aligned}$$

RK4 evaluates the function at four diffrent positions:

- $k_1 =$ slope at the start of the integration interval, using the endpoints of the previous step x_i and y_i to evaluate $f(y_i, x_i)$.
- $k_2 =$ slope in the middle of the integration interval, at $x_i = x_i + h/2$. We use Euler with k_1 to obtain an estimate for y' at this position. That is, we evaluate $f(y_i + k_1/2, x_i + h/2)$.
- $k_3 =$ we again obtain an estimate for the slope at $h/2$, however, this time we use the previous estimate k_2 to get an estimate for y'. We compute $f(y_i + k_2/2, x_i + h/2)$.

- $k_3 =$ slope at the end of the interval, $x_i + h$. This time we use k_3 to evaluate $f(y_i + k_3, x_i + h)$.

The combination
$$y_{i+1} = y_i + (k_1 + 2k_2 + 2k_3 + k_4)/6$$
reaches fourth order accuracy.

1.2.3 Generalized Runge-Kutta algorithms

The examples above motivate the folowing general formula to construct Runge-Kutta integration schemes of arbitrary high order: Let us define s positions within the integration interval $h, \tilde{x}_j = x_i + hc_j$ as well as the same number of function estimates $\tilde{y}_j$ with $j = 1, 2, \ldots, s$ to define coordinates for the evaluation of the slope at $f(\tilde{y}_j, \tilde{x}_j)$. We always start with the evaluation at the start of the interval, and so $c_1 = 0$. The general Runge-Kutta formula to integrate the function $y(x)$ accross the interval h reads as follows
$$\begin{aligned}
\tilde{x}_s &= x_i + hc_s \\
\tilde{y}_s &= y_i + h\sum_{j=1}^{s-1} a_{ij} f(\tilde{y}_j, \tilde{x}_j) \\
x_{i+1} &= x_i + h \\
y_{i+1} &= y_i + h\sum_{j=1}^{s} b_j f(\tilde{y}_j, \tilde{x}_j)
\end{aligned}$$
This gives rise to a very compact notation. The Matrix $A = A_{jl}$ with $j = 1, \ldots, s$ and $l = 1, \ldots, s$ is called Runge-Kutta matrix, while the vectors $\vec{b} = b_j$ and $\vec{c} = c_j$ are called Runge-Kutta weights and Runge-Kutta nodes respectively. As in the above example, all equations can be explititly calculated recursively. A is a lower triangular matrix with zero on the diagonal. Note that if A is fully occupied, the numerical scheme represents an implicit Runge-Kutta method, wich can only be solved interatively. Any Runge-Kutta scheme can be described in a very compact form via a Runge-Kutta tableau as
$$\left[\begin{array}{c|c} \vec{c} & A \\ \hline & \vec{b}^T \end{array}\right]$$
The standard Runge-Kutta method of fourth order is
$$\left[\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}\right]$$
The other methods, mentioned thus far, can be described by the following Runge-Kutta tableaus Euler:
$$\left[\begin{array}{c|cc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{array}\right]$$
RK2
$$\left[\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & \frac{1}{2} & \frac{1}{2} \end{array}\right]$$
midpoint rule
$$\left[\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{array}\right]$$

1.2.4 Adaptive Step Size

One key advantage of Runge-Kutta algorithms is that they allow for adaptive step sizes. Because they are not time-symmetric and because they only requive information available at the beginning of the integration interval, it is possible to change the step size as the system is integrated. This feature is a prerequisite for many effective integration algorithms in computational physics. For example, consider the calculation of the absorption cross section of electromagnetic waves for bound elections in a simple two-level system. The relevant equation is

$$\ddot{\vec{x}} + \gamma\dot{\vec{x}} + \omega^2\vec{x} = \frac{e}{m}\vec{E}e^{i\omega t}$$

describing the response (amplitude $\vec{x}$) of a damped harmonic oscillator with resonance frequency ω_0 to an external perturbation with frequency ω. Furthermore,
$\gamma = \dfrac{2e^2}{3mc^3}\omega_0^2$ is the damping term
$\omega_0 =$ resonance frequency of the bound orbin, corresponding to the energy difference $\Delta E = \hbar\omega_0$ between the energy states of the two-level system
$\omega =$ frequency of the incident electromagnetic wave with polarization $\vec{E}$
e, m are electron charge and mass
$\hbar, c$ are the Planck constant and the speed of light.
We obtain the cross section as function of frequency as

$$\sigma(\omega) = \frac{8\pi}{3}\frac{\omega^4}{\left(\omega^2 - \omega_0^2\right)^2 + \gamma^2\omega^2}$$

which relates to the classical Thomson cross section

$$\sigma_T = \frac{8\pi}{3}r_0^2 \quad \text{with the classical electron radius} \quad r_0 = \frac{e^2}{mc^2}$$

For a detailed derivation, see J.D. Jackson „Classical Electrodynamics". The resulting curve exhibits three regimes:
$\omega \gg \omega_0$: In the high energy limit, $E_{\text{photon}} \gg \Delta E$, the oscillator responds to the high-frequency forcing by adopting the forced frequency. The photons of the electromagnetic wave essentially see free electrons. This is the regime of Thomson scattering and $\sigma(\omega) \approx \sigma_T$
$\omega \approx \omega_0$: This is the regime of resonant scattering. The absorption cross section reaches a maximum of

$$\begin{aligned}
\sigma(\omega) &\approx \sigma_T\frac{\omega^4}{\left((\omega-\omega_0)(\omega+\omega_0)\right)^2 + \gamma^2\omega_0^2} \\
&\approx \frac{8\pi}{3}\left(\frac{e^2}{mc^2}\right)^2\frac{\omega_0^2}{4(\omega-\omega_0)^2 + \gamma^2} \\
&\approx \frac{8\pi}{3}\left(\frac{e^2}{mc^2}\right)^2\frac{3mc^3}{2e^2}\frac{1}{2}\frac{\gamma/2}{(\omega-\omega_0)^2 + (\gamma/2)^2} \\
&\approx 2\pi\frac{e^2}{mc}\frac{\gamma/2}{(\omega-\omega_0)^2 + (\gamma/2)^2}
\end{aligned}$$

The resonant lite follows a so-called Lorentz profile around the peak.
$\omega \ll \omega_0$: In the low energy regime, the cross section increases as

$$\sigma(\omega) = \sigma_T\left(\frac{\omega}{\omega_0}\right)^4$$

This is the limit of Rayleigh scattering, which is characterized by a strong frequency and wavelength dependence of the cross section. When looking at the frequency dependence of the cross section, it is clear that there is a large range $(\omega \gg \omega_0)$ in which $\sigma(\omega)$ is constant, whereas around the resonance peak $\sigma(\omega)$ changes rapidly with ω. When we scan or integrate along this curve with constant step size $\Delta\omega$, when we either waste computational resources at large frequencies when $\Delta\omega$ is small enough to well trace the resonance, or if we take large step sizes (adequate for large ω) we may well describe or even miss the resonance if γ is very small. Consequently, a method with adaptive step size is needed. With the Runge-Kutta schemes, there are several ways to achieve that. The most straight-forward one is to do every step twice, once with the normal stepsize h and once with two smaller steps $h/2$. Because each RK4 step requires 4 function evaluations, this in principle requires an effort of $3 \times 4 = 12$ evaluations. However, because the two steps h and $h/2$ have the same starting value, the real effort is just 11 evaluations. If you compare that with the number of evaluations for $h/2$ only, that is $2 \times x = 8$, then the overhead of $11/8 = 1.375$ is quite reasonable. We compare the results of the two steps for $y(x+h)$

$$
\begin{aligned}
y(x+h) &= y_1 + c_1 h^5 \frac{y^{(5)}(x)}{5!} + \mathcal{O}\big(h^6\big) \\
y(x+h) &= y_2 + 2c_2 \left(\frac{h}{2}\right)^5 \frac{y^{(5)}(x)}{5!} + \mathcal{O}\big(h^6\big)
\end{aligned}
$$

where y_1 and y_2 are the estimates for $y(x+h)$ for stepsize h and $h/2$, and where the second term on the right hand side describes the error to leading order. We know that the difference $\Delta = y_2 - y_1$ scales with h^5. If we want this difference to be smaller to some tolerance value Δ_0, we can obtain an estimate for the required stepsize as

$$h_0 = h \left| \frac{\Delta_0}{\Delta} \right|^{1/5}$$

That is if $\Delta > \Delta_0$ then we repeat the step with the new stepsize estimate h_0. If the difference is much smaller than the stepszie can actually be increased. In practice, we try to predict the stepsize for the next step based on previous values. Nevertheless, we still need to test the actual arror after each step. And so, a less timeconsuming alternative to the above approach of comparing two different values of h is to look at the difference between the previous and the current timestep. Other alternatives are to compare two diffrent RK schemes with the same stepsize h. For example RK2 and the midpoint rule. For the forth order RK method this is the base for the Runge-Kutta-Fehlberg algorithm that is commonly used in astrophysical hydrodynamics solver. It is frequently called RKF45 method, and provides stepping of order $\mathcal{O}\big(h^4\big)$ with an error estimate of order $\mathcal{O}\big(h^5\big)$. It requires only one extra calculation. The error can be estimated and controlled by using the higher order embedded method that allows for an adaptive stepsize to be determined automatically.

$$\left[\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{1}{4} & \frac{1}{4} & & & & & \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} & & & & \\
\frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & & & \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\
\frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & \\
\hline
 & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \\
\hline
 & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55}
\end{array}\right]$$

According to the RKF45 tableau the first function estimate is based on five states

$$y_{i+1,a} = y_i + h\left(\frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5\right)$$

and the second one is based on six evaluations and reads

$$y_{i+1,b} = y_i + h\left(\frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28516}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6\right)$$

The usefulness of the method is based on the fact that the same set of k_j is used for both estimates. According to the tableau, the k_j are computed as

$$\begin{aligned}
k_1 &= f(y_i, x_i) \\
k_2 &= f\left(y_i + \frac{1}{4}hk_1, x_i + \frac{1}{4}h\right) \\
k_3 &= f\left(y_i + \frac{3}{32}hk_1 + \frac{9}{32}hk_2, x_i\frac{3}{8}h\right) \\
k_4 &= f\left(y_i + \frac{1932}{2197}hk_1 - \frac{7200}{2197}hk_2 + \frac{7296}{2197}hk_3, x_i + \frac{12}{13}h\right) \\
k_5 &= f\left(y_i + \frac{439}{216}hk_1 - 8hk_2 + \frac{3680}{513}hk_3 - \frac{845}{4104}hk_4, x_i + h\right) \\
k_6 &= f\left(y_i + -\frac{8}{27}hk_1 + 2hk_2 - \frac{3544}{2565}hk_3 + \frac{1859}{4104}hk_4 - \frac{11}{40}hk_5, x_i + \frac{1}{2}h\right)
\end{aligned}$$

Subtracting the two function estimates as

$$\Delta = h\left|\frac{1}{360}k_1 + \frac{1}{33}k_3 - \frac{1}{34}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6\right|$$

If this value is acceptable, then go to the next step. If not, then reject and start again this step with smaller stepsize according to

$$h_0 = h\left|\frac{\Delta_0}{\Delta}\right|^{1/5}$$