# Ordinary Differential Equations II: Advanced Methods

Even a simple two-body problem in gravitation is difficult to do with prior methods.

Different pieces of information help us to properly constrain motion:

$\boldsymbol{F} = -\,G\,M\,m\,/\,|\boldsymbol{r}|^3\,\boldsymbol{r}$ [ conserved? ]

$E = K + U = \tfrac{1}{2}\,m\,v^2 - G\,M\,m\,/\,r$ [ conserved ]

$\boldsymbol{L} = \boldsymbol{r} \times \boldsymbol{p}$ [ conserved ]

There is an analytic solution to the simple two-body problem, but if we add drag or a third mass, the problem is much more difficult

$\boldsymbol{F} = -\,G\,M\,m\,/\,|\boldsymbol{r}|^3\,\boldsymbol{r}$ + $-\,G\,M'\,m\,/\,|\boldsymbol{r}'|^3\,\boldsymbol{r}'$ => numerical approach!

One can use the Euler method to compute the trajectory, but energy is not conserved, hinting that the approach is incorrect. Another clue is that a supposedly circular orbit scenario results in the orbiter wandering away when the Euler method is used (Fig 3.2).

<u>Runge-Kutta</u>

The 4th order Runge-Kutta approach to numerically solving ODEs may be the most popular – it is both accurate and fast. The text nicely outlines the approach. To summarize in brief, the Runge-Kutta method utilizes Taylor series expansions done about the midpoint of the time interval, and uses the Euler method to approximate the (position) evaluated at the midpoint.

$d\boldsymbol{x}/dt = f(\boldsymbol{x}(t),t)$ $\quad \boldsymbol{x}(t) = \mathbf{x}(\boldsymbol{r},t)$

The function varies over the time step, so we estimate and average $\boldsymbol{x}^*$ at the midpoint by extrapolating forward:

$\boldsymbol{x}(t+\delta t) = \boldsymbol{x}(t) + \delta t\,\boldsymbol{f}[\boldsymbol{x}^*(t+\delta t/2),t+\delta t/2]$ $\quad \boldsymbol{x}^*(t+\delta t/2)$ = some order of Taylor series extrapolation from $x(t)$

Example: $dy/dt = t + y$, $y(t=0) = 1$ *solution*: $y(t=0.1) = 1.1103418$

**Euler** $y_{n+1} = y_n + \delta t\, y'_n + O(\delta t^2)$

| $\delta t$=0.02 | $t_n$ | $y_n$ | $y'_n$ | $\delta t\, y'_n$ |
|---|---|---|---|---|
| | 0 | | 1.0000 | |
| | 0.02 | 1.0200 | 1.0400 | 0.0208 |
| | 0.04 | 1.0408 | | 0.0216 |
| | | 1.0624 | 1.1224 | 0.0224 |
| | 0.08 | | 1.1648 | 0.0233 |
| | 0.10 | 1.1081 | | |

**Euler Predictor-Corrector** $y_{n+1} = y_n + \delta t\, [y'_n + y'_{n+1}]/2 + O(\delta t^3)$ where the Euler method is used to compute the "intermediate" $y_{n+1}$

| | | *improved* | *improved* | | *intermediate* | *intermediate* |
|---|---|---|---|---|---|---|
| $\delta t$=0.02 | $x_n$ | $y_n$ | $y'_n$ | $\delta t\, y'_n$ | $y_{n+1}$ | $y'_{n+1}$ |
| | 0 | 1.0000 | 1.0000 | 0.0200 | 1.0200 | |
| | 0.02 | 1.0204 | | 0.0208 | 1.0412 | 1.0812 |
| | | 1.0416 | 1.0816 | 0.0216 | | 1.1232 |
| | 0.06 | | 1.1237 | 0.0225 | 1.0862 | 1.1662 |
| | 0.08 | 1.0866 | 1.1666 | 0.0233 | 1.1099 | 1.2099 |
| | 0.10 | 1.1104 | | | | |

**4th order Runge-Kutta** $y_{n+1} = y_n + 1/6\, \delta t\, [k_1 + 2k_2 + 2k_3 + k_4] + O(\delta t^5)$

$k_1 = f(t_n, y_n)$ $k_2 = f(t_n + \tfrac{1}{2}\delta t, y_n + \tfrac{1}{2}\delta t\, k_1)$ $k_3 = f(t_n + \tfrac{1}{2}\delta t, y_n + \tfrac{1}{2}\delta t\, k_2)$ $k_4 = f(t_n + \delta t, y_n + \delta t\, k_3)$

$\delta t$=0.1 $k_1$=1 $k_2$=1.1 $k_3$=1.1050 $k_4$=1.2105

$\rightarrow y(0.1)$=1+0.1103417

| Method Name | Estimate of slope over time interval | Global Error | Local Error | Evaluations of $f(x,y)$ per step |
|---|---|---|---|---|
| Euler | Initial value | $O(\delta t)$ | $O(\delta t^2)$ | 1 |
| Euler Predictor-Corrector | Arithmetic average of initial and final predicted slope | $O(\delta t^2)$ | $O(\delta t^3)$ | 2 |
| 4th order Runge-Kutta | Weighted average of four values | $O(\delta t^4)$ | $O(\delta t^5)$ | 4 |

| Method | Step size | Result | Error | # function evaluations |
|---|---|---|---|---|
| Euler | 0.02 | 1.1081 | 0.0022 | 5 |
| Euler Predictor-Corrector | 0.02 | 1.1104 | 0.0001 | 12 |
| 4th order Runge-Kutta | 0.1 | 1.11034 | 0.000001 | 4 |

**Integration** (a.k.a. "quadrature")

Evaluating

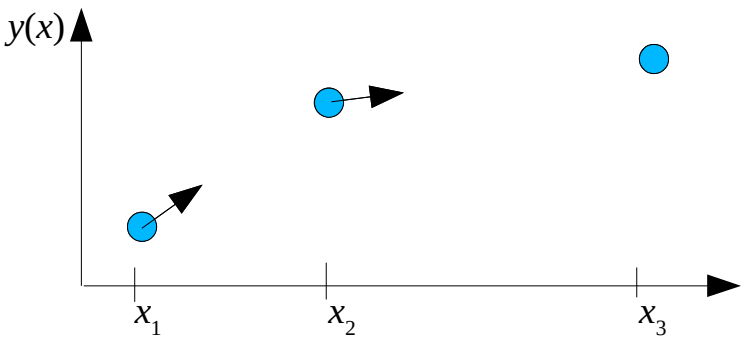
$I = \int f(x)\, dx$ from $x=a$ to $x=b$

is equivalent to solving for

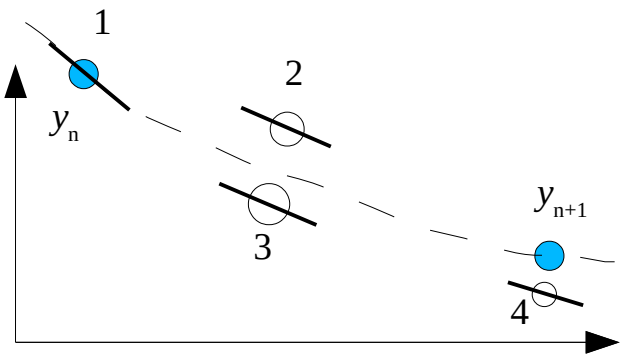
$I \equiv y(b)$ where $dy/dx = f(x)$ and $y(a)=0$.

Two techniques we've covered for solving these types of problems are the Euler method and the more accurate and faster 4th order Runge-Kutta approach.

*Quick Review of integrating ordinary differential equations*




*Euler* method: extrapolate the derivative at the start of each interval to find the next function value.

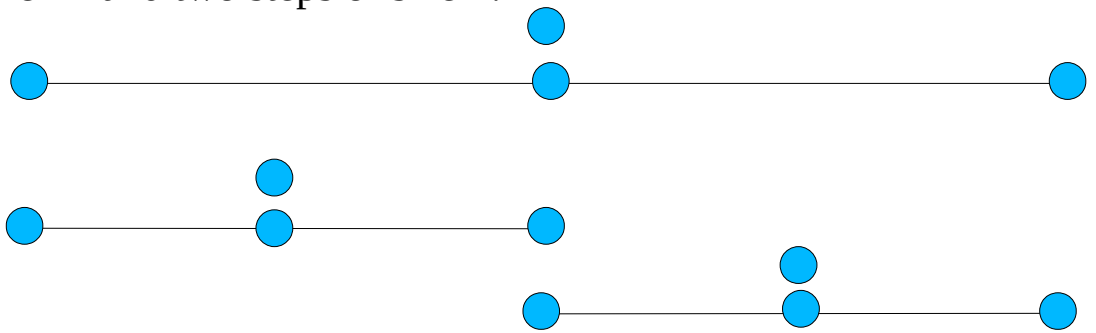



*RK*: the derivative is evaluated four times at each step, once initially, twice at trial midpoints, and at the trial endpoint. (Numerical Recipes)

**Adaptive Time Stepping**

Sometimes one can be computationally more efficient by actually incorporating additional calculations. One such example is adaptive time stepping, where additional computations dealing with smaller step sizes is performed. If smaller step sizes are warranted (e.g., improved accuracy), then the program automatically decreases the step size for ensuing computations. Conversely, if the canonical step performs just fine in terms of some metric like energy conservation, then the step size can be increased for ensuing computations. The savings can sometimes be orders

(continued)
of magnitude, not merely tens of percent. Take the example where a Runge-Kutta computation is carried out over both a step size $2h$ and two steps of size $h$.



If the exact solution in going from $x$ to $x+2h$ is $y(x+2h)$ and the approximate solutions are $y_1$ (one step of $2h$) and $y_2$ (two steps each of size $h$), then

$$y(x+2h) = y_1 + (2h)^5\phi + O(h^6)$$

$$y(x+2h) = y_2 + 2(h)^5\phi + O(h^6)$$

The difference between the two estimates is a measure of the truncation error:

$$\Delta \equiv y_2 - y_1$$

This degree of accuracy can be monitored by adjusting $h$. Since 4th order Runge-Kutta is accurate to fifth power, $\Delta$ scales as $h^5$. Therefore if we take step $h_1$ and produce error $\Delta_1$, then the step $h_0$ that would have introduced error $\Delta_0$ is found from

$$h_0 = h_1 \left| \Delta_0/\Delta_1 \right|^{0.2} \qquad (1)$$

In short, if $\Delta_1$ is larger than $\Delta_0$ in magnitude, then Equation 1 tells us how much to decrease the stepsize when we re-try the present (failed) step. On the other hand, if $\Delta_1$ is smaller than $\Delta_0$ in magnitude, then Equation 1 tells us how much to increase the stepsize for the next step (Numerical Recipes).

Question: Why does the first sentence in the previous paragraph say “re-try the present step” whereas the second sentence says “for the next step”?

Pendulum examples

Suppose we have a simple pendulum of length 2.0 m; the rod is massless and there is a ball of mass 0.5 kg at the end. We know that for small oscillations the period is _____.

How would the period change if it were a “physical” pendulum? (i.e., the mass is distributed throughout the rod, not just at the end)

Suppose we achieve a certain acceptable accuracy in our computations using timestep *h*. If the simple pendulum's length were doubled, what should we use for the timestep to achieve the same accuracy?

Spring example

Now suppose we have a spring of spring constant *k*, and we achieve a certain acceptable accuracy in the period using timestep *h*. How should our timestep change if the spring constant is quadrupled?