Numerical Optimal Control Course at IfA, ETH Zurich
on Feb-May 2011, by Prof. M. Diehl (K.U. Leuven)

# Exercise 2: Runge-Kutta Integrator and Gauss-Newton Algorithm

Moritz Diehl, Peter Hokayem, Robin Vujanic

In this second exercise we compare two methods for numerical integration of ordinary differential equations (ODE) and program our first own optimization code, a Gauss-Newton algorithm.

1. ODE Simulation: throughout this exercise sheet, we regard again the controlled harmonic oscillator from the first sheet described by

$$\frac{d}{dt}\begin{bmatrix} p(t) \\ v(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} p(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \quad t \in [0, T]. \tag{1}$$

   We abbreviate this ODE as $\dot{x} = f(x, u)$ with $x = (p, v)^T$. Let us choose again the fixed initial value $x_0 = (10, 0)^T$ and $T = 10$.

   In the next questions we are interested in comparing the simulation results for $u(t) = 0$ that are obtained by two different integration schemes, namely the Euler integrator from the last exercise, and a Runge-Kutta integrator. We regard in particular the value $p(10)$, and as the ODE is explicitly solvable, we know it exactly, which is useful for comparisons. What is the analytical expression for $p(10)$?

2. First run again your explicit Euler method from the last time, e.g. again with $N = 50$ integrator steps, i.e. with a stepsize of $\Delta t = 10/50 = 0.2$. The central line in the Euler code reads

$$x_{k+1} \quad = \quad x_k + \Delta t \cdot f(x_k, u_k) \tag{2}$$

   Plot your trajectories $\{(t_k, x_k)\}_1^{N+1}$ for $u_k = 0$.

3. Now exchange in your Euler simulation code the line that generates the step (2) by the following five lines:

$$\begin{aligned} k_1 &= f(x_k, u_k) && (3)\\ k_2 &= f(x_k + \frac{1}{2}\Delta t \cdot k_1, u_k) && (4)\\ k_3 &= f(x_k + \frac{1}{2}\Delta t \cdot k_2, u_k) && (5)\\ k_4 &= f(x_k + \Delta t \cdot k_3, u_k) && (6)\\ x_{k+1} &= x_k + \Delta t \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) && (7) \end{aligned}$$

This is the classical Runge Kutta method of order four (RK4). Note that each integrator step is four times as expensive as an Euler step. What is the advantage of this extra effort? To get an idea, plot your trajectories $\{(t_k, x_k)\}_1^{N+1}$ for the same number $N$ of integrator steps.

4. To make the comparison of Euler and RK4 quantitative, regard the different approximations of $p(10)$ that you obtain for different stepsizes, e.g. $\Delta t = 10^{-k}$ with $k = 0, \ldots, 5$. We call these approximations $\tilde{p}(10; \Delta t)$. Compute the errors $|p(10) - \tilde{p}(10; \Delta t)|$ and plot them doubly logarithmic, i.e. plot $\log_{10}(|p(10) - \tilde{p}(10; \Delta t)|)$ on the $y$-axis and $\log_{10} \Delta t$ on the $x$-axis, for each of the integrators. You should see a line for each integrator. Can you explain the different slopes?

5. We regard again the optimal control problem from the last exercise sheet. We had previously used the Euler integrator, but today we use our new RK4 integrator because it is more accurate. We do again $N = 50$ integrator steps to obtain the terminal state as a function of the controls $u_0, \ldots, u_{N-1}$, and denote the function today by $g_{\text{sim}} : \mathbb{R}^N \to \mathbb{R}^2$. The nonlinear program we want to solve is again

$$\min_{U \in \mathbb{R}^N} \|U\|_2^2 \quad \text{subject to} \qquad g_{\text{sim}}(U) = 0 \tag{8}$$

Today, we do not solve this problem with `fmincon`, but we write our first Newton-type optimization method. To prepare the ground, make sure you have the routine $g_{\text{sim}}$ as a function of the 50 controls with the two terminal states as outputs.

6. Motivating background information (you might skip to the next question): The necessary optimality conditions (KKT conditions) for the above problem are

$$\begin{aligned} 2U^* + \frac{\partial g_{\text{sim}}}{\partial U}(U^*)^T \lambda^* &= 0 && (9)\\ g_{\text{sim}}(U^*) &= 0. && (10) \end{aligned}$$

Let us introduce a shorthand for the Jacobian matrix

$$J_{\text{sim}}(U) := \frac{\partial g_{\text{sim}}}{\partial U}(U)$$

By linearization of the constraint at some given iterate $(U_k, \lambda_k)$ and neglecting its second order derivatives, we get the following (Gauss-Newton) approximation of the KKT conditions:

$$\begin{bmatrix} 2U_k \\ g_{\text{sim}}(U_k) \end{bmatrix} + \begin{bmatrix} 2\mathbb{I} & J_{\text{sim}}(U_K)^T \\ J_{\text{sim}}(U_k) & 0 \end{bmatrix} \begin{bmatrix} U_{k+1} - U_k \\ \lambda_{k+1} \end{bmatrix} = 0$$

This system can be solved easily by a linear solve in order to obtain a new iterate $U_{k+1}$. But in order to do this, we need first to compute the Jacobian $J_{\text{sim}}(U)$.

7. Implement a routine that uses finite differences, i.e. calls the function $g_{\text{sim}}$ $(N+1)$ times, once at the nominal value and then with each component slightly perturbed by e.g. $\delta = 10^{-4}$ in the direction of each unit vector $e_k$, so that we get the approximations

$$\frac{\partial g_{\text{sim}}}{\partial u_k}(U) \approx \frac{g_{\text{sim}}(U + \delta e_k) - g_{\text{sim}}(U)}{\delta}.$$

We denote the resulting function that gives the full Jacobian matrix of $g_{\text{sim}}$ by $J_{\text{sim}} : \mathbb{R}^N \to \mathbb{R}^{2\times N}$.

8. Now, we implement the Gauss-Newton scheme from above, but as we are not interested in the multipliers we just implement it as follows:

$$U_{k+1} = U_k - \begin{bmatrix} \mathbb{I} & 0 \end{bmatrix} \begin{bmatrix} 2\mathbb{I} & J_{\text{sim}}(U_k)^T \\ J_{\text{sim}}(U_k) & 0 \end{bmatrix}^{-1} \begin{bmatrix} 2U_k \\ g_{\text{sim}}(U_k) \end{bmatrix}$$

Choose an initial guess for the controls, e.g. $U = 0$, and start your iteration and stop when $\|U_{k+1} - U_k\|$ is very small. How many iterations do you need to converge? Do you have an idea why?