University of Mannheim
Scientific Computing, B6 26, C306, 68131 Mannheim
Dr. Andreas Sommer (ansommer@mail.uni-mannheim.de)

UNIVERSITY OF MANNHEIM

Nonlinear Optimization (FSS 2023)

Easter Challenge

Due on 13.04.2023

1. Programming: Parameter Estimation in Lotka-Volterra [10 bonus points]

The Lotka-Volterra equations

$$\begin{aligned} \dot{x}_1 &= p_1 x_1 - p_3 x_1 x_2 \\ \dot{x}_2 &= -p_2 x_2 + p_4 x_1 x_2; \end{aligned} \tag{1}$$

are an ODE system that is frequently used as a simple predatory-prey model. The parameter vector $p = (p_1, p_2, p_3, p_4)^\top \in \mathbb{R}^4$ describes the birth rates, death rates, and interactions between the two species (see any textbook for details).

While the prey and predator densities can be determined experimentally (by counting), the values of the parameters are not directly accessible and must be estimated from the data. The data consists of N pairs $(t_i, \eta_i), (i = 1...N)$ where $t_i \in [0, T]$ are time points and $\eta_i \in \mathbb{R}^2$ are observations of the prey and predator counts.

The file `LVdata.mat` contains a structure with fields `t` and `eta` with population observations. We want to determine a parameter vector p that reproduces these measurements [1].

A classical approach of parameter estimation tries to determine a parameter vector p that minimizes the sum of squared differences between model output and observed data ("least squares problems"):

$$\min_p g(p) := \sum_{i=1}^{N} \|x(t_i, p) - \eta_i\|^2 \tag{2}$$

$$\text{s.t. } \dot{x} = f(t, x, p) \quad x(0) = x_0 \quad t \in [0, T] \tag{3}$$

Here, $x(t_i, p)$ is the *model response* at time t_i for parameters p (here: full state measurements), and $f(t, x, p)$ is the right hand side of the ODE as in eq. (1).

The minimization is "constrained" by an *initial value problem* that consists of a parameter-dependent system of ordinary differential equations (ODEs) $\dot{x} = f(t, x, p)$, (t: time, x: states, p: parameters), integrated over a time horizon $[0, T]$, starting with an *initial value* x_0.

a) Visualize the time course of the data

b) Write a function `dx = LVrhs(t,x,p)` that implements the Lotka-Volterra ODE. Determine a solution of the ODE, using the initial values $x_0 = (6, 5)^\top$, and two distinct parameter vectors $p_0^{(1)} = (3, 2, 1, 1)^\top$ and $p_0^{(2)} = (0.1, 0.1, 0.1, 0.1)^\top$
Visualize these simulations (use Matlab's `ode45` for integration).

c) Implement the objective function in eq. (2).
Determine the objective function's values for $p_0^{(1)}$ and $p_0^{(2)}$.

[1] For comparison: The solution is `p = [2.2 2.5 0.85 1.5];`

d) Use one of Matlab's built-in optimization routines to solve the parameter estimation problem p. Determine the number of needed integrations, and measure the required time using `tic()` and `toc()`. Compare the results using both $p_0^{(1)}$ and $p_0^{(2)}$ as initial guesses.

e) Use one of the first-order algorithms (e.g. steepest descent) from the lecure to solve the parameter estimation problem. Compare the results using both $p_0^{(1)}$ and $p_0^{(2)}$ as initial guesses. Determine the number of needed integrations and measure the required time.

f) Use a second-order method (e.g. Newton) from the lecture to solve the parameter estimation problem. Compare the results using both $p_0^{(1)}$ and $p_0^{(2)}$ as initial guesses. Determine the number of needed integrations and measure the required time.

Always visualize the solutions and the data in the same plot.

Note that for e) and f), the gradient of the objective w.r.t. the parameters p is needed.
A simple approach is to use *finite differences* (FD):

$$\frac{\mathrm{d}g}{\mathrm{d}p_i}(p) \approx \frac{g(p + h \cdot e_i) - g(p)}{h}$$

with e_i being the i-th unit vector. A suitable *FD step size* h is 10^{-6}. This approach can also be used to approximate the Hessian, though this requires more accurate (and thus more costly) integration. This can be done setting by setting `RelTol` to, e.g. 10^{-10} (see `doc ode45` for details).

[As a remark: A much more sophisticated approach would implement the so-called *sensitivity equations* or *variational differential equations* for the ODE system to determine the parameter sensitivities.]

Put all files into a single zip-archive, named by the lexicographically ordered family names of all group members separated by a hyphen, and send the zip-file to ansommer@mail.uni-mannheim.de.
Please add PDF files containing code and output (text and visual) to your submissions.
Comment your code intensely. Use a complete header that describes input and output arguments and also comment the implementation where appropriate (see the examples at the course web site). Avoid obvious inefficiencies like repeated evaluation of identical/unchanged expressions.