Department of Mathematics and Computer Science
University of Southern Denmark, Odense

February 9, 2016
Marco Chiarandini

# **FF505** – Computational Science

## Week 5, Spring 2016

## MATLAB Exercises

Work in small groups at the computer to solve the following exercises.

1. Calculate $\pi$ using Monte Carlo simulation explained in class.

   - write the algorithm on paper
   - use first simple for-loop and scalar calculations then try to rewrite your algorithm using vectorisation.
   - Make your simulation exactly reproducible by setting the seed of the random number, such that everytime you execute your program you obtain precisely the same sequence of random numbers (hint: `help seed`).
   - read the slides on "Functions" from lecture three and make a function that takes as input the sample size (that is, the number of points to generate) and returns the value of $\pi$ calculated by Monte Carlo simulation.
   - compare the running time of a for-loop implementation against a vectorised implementation (hint: `help tic, toc`).

2. **Numerical methods for Ordinary Differential Equations** The study of population dynamics deals with the evolution of population of plants, animals or humans over time. The interest is answering questions about the *asymptotic stability* of the dynamic system, that is, for example, whether a specie will eventually become extinct.

   An important model in this field is the *logistic equation* or *Verhulst equation*

   $$y' = y - y^2, \quad y(0) = 0.2$$

   that expresses the rate of growth of the population $y$ over time $t$ ($dy/dt = y'$). The first term on the right hand side, $y$, determines a positive growth. If the model was just $y' = y$ then the growth would be exponential and unlimited (you can verify this claim by deriving the solution of the equation). The second term, $-y^2$, is a "braking term" that prevents the population from growing without bound. Finally, $y(0)$ is the initial value, needed to solve the equation.

   The logistic equation above is a *nonlinear nonhomogeneous ordinary differential equation (ODE)* (see Chapter 1 of [Kr] for an explanation of these terms). That specific equation can be reduced to a linear form and the solution written down as a formula (that is, in closed form). If you do not remember or do not know how to derive the closed form of that equation, you can read on page 30 of [Kr].

   However, often a closed form is not known. Computational methods have then been developed for a numerical solution of the equation. More specifically, in MatLab there is a set of functions for carrying out this task in the case of ODEs. To gain an idea of the algorithms they implement you can read Section 21.1 of [Kr]. A distinction is done between stiff and nonstiff functions. The name is taken from one of the main applications of ODEs, which is in the study of oscillations. It is related to the value of the constants involved in

the ODE. The term stiff recalls a mass-spring system with a stiff spring (large constant $k$). For stiff functions methods must be more accurate in order to avoid an increase in errors from the initial value.

- Search in the MatLab Help for "Ordinary Differential Equations".
- Retrieve from the source given above the closed form solution of the equation.
- Compare in a plot the closed form solution with the solution provided by some of the MatLab functions. In particular, try the `ode23` and `ode23s`. Which is giving the best solution?
- Provide an interpretation of the curve observed from a physical point of view.