Simulation and High-Performance Computing

University of Kiel, September 28th to Oktober 9th, 2020.

Exercises: Day Four

CSC Computational Science Center

General Remark

On day four, we will look at iterative methods. Those methods aim to approximate the solution to *one* difficult problem by *repeatedly* solving a related but simpler problem. The solution of one simple problem is used as starting data for the next one. The solutions to these simpler problems are expected to *converge* towards the solution of the original difficult problem.

In the first part of the exercises, we examine iterative methods for solving nonlinear equations. Newton's method is a widely used approach in that case. Eigenvalue problems also lead to nonlinear equations, however, there are more efficient methods tailored specifically for the task of finding eigenvalues.

In the second part of the exercises, we look at an iterative method that is effective at solving linear systems that are so large that direct solvers - such as an LU decomposition - are too costly and require too much memory.

Main Exercises: Applications of Lecture 07

a) Computing Lagrange Points With Newton's Method

For two planets at position $y_1, y_2 \in \mathbb{R}^2$ with masses $m_1, m_2 \in \mathbb{R}_{>0}$ we would like to compute the so-called *Lagrange points*.

The force that acts on a given point $x \in \mathbb{R}^2$ can be computed by

$$f : \mathbb{R}^2 \to \mathbb{R}^2, \quad x \mapsto \gamma \left(m_1 \frac{y_1 - x}{\|y_1 - x\|^3} + m_2 \frac{y_2 - x}{\|y_2 - x\|^3} \right).$$

A Lagrange point $x^* \in \mathbb{R}^2$ is characterized by $f(x^*) = 0$.

Implement the Newton-iteration in the file **exercise_lagrange_point.c** in order to solve for the Lagrange point x^* of the above system.

In every step of the Newton-iteration solving a linear system with the Jacobian $Df(x)$ is needed. In order to accomplish that in an easy manner, use *Cramer's rule.*

Cramer's rule for a 2D system of equations looks as follows:

For

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad \text{with} \quad Ax = b$$

the solution vector x is defined as:

$$x_1 = \left| \begin{pmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{pmatrix} \right| \Big/ \left| \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \right|,$$

$$x_2 = \left| \begin{pmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{pmatrix} \right| \Big/ \left| \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \right|.$$

b) The 1D Model Problem and the Power Iteration

For a given $n \in \mathbb{N}$, we define $h := \frac{1}{n+1}$ and the matrix

$$A \; := \; \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

One can show that A has the eigenvalues

$$\lambda_j \; = \; \frac{2}{h^2}\left(1 - \cos\left(\frac{\pi j}{n+1}\right)\right), \qquad j \in \{1, ..., n\}.$$

The matrix A appears when discretizing (using finite differences) the Poisson model problem not in 2D on the unit square as it was shown in lecture 05, but when discretizing that problem merely in 1D. We are using this simpler version of the model problem here, because it has everything we need to test the numerical methods of these exercises and because it is a bit easier to handle than the 2D version.

Obviously, A has a very simple structure - only two different values appear in it and all values not on the main diagonal or directly next to it are zero. Similarly to the exercises from yesterday, we want to make use of this simple structure by implementing - *without storing A as a matrix* - a multiplication of a given vector x with A and adding the result to a given vector y. **Implement** that multiplication in the function `addeval_1d_problem` in the file `exercise_eigensolver.c`.

Now **implement** the power iteration from lecture 07 in the function `power_iteration` in the file `exercise_eigensolver.c` and use it to find the largest eigenvalue of A. Compare it to the given analytical solution.

Additional Exercises: Applications of Lecture 08

a) The Conjugate Gradient Method

Here, we look at a very important and widely used method for solving large systems of linear equations. That so-called *conjugate gradient (CG) method* is only applicable if the matrix A in the linear system $Ax = b$ is *symmetric and positive-definite.* **Implement** the CG method in the function **cg_solve** in the file **linalg.c** and test it on the matrix from the 1D model problem introduced above. Think about why that matrix is positive-definite.

b) The Inverse Iteration Using the CG Method

Now **implement** the *inverse* iteration from lecture 07 in the function `inverse_iteration` in the file `exercise_eigensolver.c` and use it to find the smallest eigenvalue of the matrix from the 1D model problem introduced above. Your newly implemented CG method can be utilized to solve the arising systems of linear equations. Compare your solution for the smallest eigenvalue to the given analytical solution.

Please provide feedback about the format/difficulty/length of the exercises so that we can adjust accordingly. Contact us via Email or on the OpenOLAT forum.