Scientific Computing I

Winter semester 2022/2023
Prof. Dr. Carsten Burstedde
Uta Seidler and Denis Düsseldorf

Exercise sheet 6.

Submission on **01.12.2022**.

Exercise 16. (Weak derivatives I)

(6 Points)

a) Compute the weak derivative of $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto |x|$. Use directly the definition of weak differentiability.

b) The Heaviside step function is defined by

$$H(x) = \begin{cases} 0 & \text{if } x < 0, \\ 1/2 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Show that H is not weakly differentiable.
Bonus question: Can you think of a weaker notion of derivatives such that H is differentiable?

c) Let $\Omega = (0, 2)$. Consider the functions

$$f(x) = \begin{cases} x & \text{if } 0 < x < 1, \\ 1 & \text{if } 1 \le x < 2, \end{cases} \qquad \text{and} \qquad g(x) = \begin{cases} x & \text{if } 0 < x < 1, \\ 2 & \text{if } 1 \le x < 2. \end{cases}$$

Show that $f \in H^1(\Omega)$ by finding its weak derivative, and that $f \notin H^2(\Omega)$. Is $g \in H^1(\Omega)$?

Exercise 17. (Weak derivatives II)

(6 Points)

Similar to the spaces $H^m(\Omega)$, we can also define function spaces where the weak derivatives are functions in $L^p(\Omega)$:

Definition. Let k be a nonnegative integer and $1 \le p \le \infty$. The Sobolev spaces $W^{k,p}(\Omega)$ consists of functions $u \colon \Omega \to \mathbb{R}$ such that for each multiindex α with $|\alpha| \le k$, the weak derivative $\partial^\alpha u$ exists and is a function in $L^p(\Omega)$.

Note that for $p = 2$, the notation

$$W^{k,2}(\Omega) = H^k(\Omega)$$

is often used since $H^k(\Omega)$ is a Hilbert space.

Now let $d \in \mathbb{N}$ and $\Omega = B_1(0) \subset \mathbb{R}^d$. For $\alpha > 0$ consider the function

$$u\colon \Omega \to \mathbb{R}, \qquad x \mapsto |x|^{-\alpha}.$$

Determine the range of the parameter α in dependence on d and p such that

a) $u \in L^p(\Omega)$

b) $u \in W^{1,p}(\Omega)$.

Programming exercise 1. (The CG method)

(0 Points)

Let us prepare the first real programming exercise that will be posted next week and get warmed up in style. Remind yourself of the conjugate gradient method to iteratively solve symmetric positive definite linear systems. We will want to use this method to solve several example systems with the added requirement that the matrix-vector multiplication shall be defined as a function that is understood as a parameter. We will not be creating explicit matrix objects in this exercise.

1. Given a variable but fixed natural number N, choose a representation for a vector of this length and populate a non-zero right hand side vector of your choice. Prepare a solution vector of the same length and the temporary vectors needed in the conjugate gradient method.

2. Program the conjugate gradient method as a function, to which the matrix-vector product operation is passed as a function or closure parameter from the outside, and the right hand side and other required data are passed as well (such as the iteration count). Do not rely on any global variables except N.

3. Consider a function that applies a linear operator to a given vector, writing into a second vector provided as output space. Implement three such functions: applying the Hilbert matrix, the identity matrix, and our favorite finite element matrix for homogeneous boundary conditions,

$$\begin{pmatrix} 2 & -1 & \dots & & \\ -1 & 2 & -1 & \dots & \\ & & \dots & & \\ \dots & -1 & 2 & -1 & \\ & \dots & -1 & 2 & \end{pmatrix} \in \mathbb{R}^{N \times N}. \tag{1}$$

4. Compare the iteration numbers of the conjugate gradient solver at a given tolerance for the various matrices depending on N. Output your numerical solution and compute its error against the exact solution with an external linear algebra tool of your choice. For a fixed number of iterations, how does the error depend on N?

You may use any of the C, C++, C# lua, rust, haskell, or zig languages. In either case, it is not allowed to use predefined vector or matrix objects, and neither is it allowed to use iterators. Loops shall be written by explicitly iterating over an index variable. With object oriented languages, the use of classes or overloading shall be kept to a reasonable minimum.

You have two weeks for the programming exercises. There will be a folder on the ecampus site for the submission of the graded programming exercises. In what form you will have to present them will be discussed in the lecture next Tuesday.