Project1

September 15, 2021

0.0.1 TMA4215 Numerisk Matematikk

Høst 2021 – September 16, 2021

1 Project 1: Newton Iteration

1.0.1 Notes

This project is an individual project and should be solved that way. You can discuss the problems, but you should write the solutions *in your own words*.

All code – also the tests – should be in individual cells that can just be run (as soon as the necessary functions are defined). Functions should only be used in cells *after* their definition, such that an evaltuation in order of the notebook does not yield errors.

The notebook with the answers should be uploaded in inspera.

The project is obligatory and counts 10% on the final grade.

1.0.2 Submission Deadline

Tuesday, October 5, 2021, 23:59.

1.1 Introduction

This project considers the Newton iteration for the n-dimensional nonlinear equation

$$\mathbf{F}(\mathbf{x}) = \mathbf{0},$$

i.e. $\mathbf{F}\colon \mathbb{R}^n \to \mathbb{R}^n$. For the first two problems, we will consider the multivariate case. For the implementation, we stay with $n = 1$.

1.2 Problem 1

1. Compute the gradient ∇F and the Hessian H_F of the Rosenbrock function $F\colon \mathbb{R}^2 \to \mathbb{R}$ defined by $F(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$. Show that $\mathbf{x}^* = (1, 1)^{\mathrm{T}}$ is the only local minimizer of this function and that the Hessian matrix at this point is positive definite.
2. Show that the function $G\colon \mathbb{R}^2 \to \mathbb{R}$ defined by $G(\mathbf{x}) = 8x_1 + 12x_2 + x_1^2 - 2x_2^2$ has one stationary point which is neither a maximum nor a minimum.

3. Use `matplotlibs pyplot` to plot `contour` plots on the unit square $\Omega_1 = [-1, 1]^2$ for F_1 and on the unit square $\Omega_2 = [-10, 10]^2$ for G.

1.3 Problem 2

Let $A \in \mathbb{R}^{n\times n}$ be symmetric. The Rayleigh quotient $F\colon \mathbb{R}^n \to \mathbb{R}$ for A is defined as

$$F(\mathbf{x}) = \frac{\mathbf{x}^T A\mathbf{x}}{\|\mathbf{x}\|_2^2}.$$

The goal of this problem is to analyze Newton's method for finding minimizer (or stationary points) $\mathbf{x}^*$ of F or in other words zeros of the gradient $\nabla F(\mathbf{x}) = \mathbf{0}$.

To this end:

1. Compute the gradient ∇F.
2. What are the stationary points $\mathbf{x}^*$ of F? What is the value of $F(\mathbf{x}^*)$ then?
3. Show that the Hessian matrix $\nabla^2 F$ (i.e. the Jacobian of the Gradient ∇F) satisfies

$$\nabla^2 F(\mathbf{x}) = \frac{2}{\|\mathbf{x}\|_2^2}\Big(I - 2\frac{\mathbf{x}\mathbf{x}^T}{\|\mathbf{x}\|_2^2}\Big)\Big(A - F(\mathbf{x})I\Big)\Big(I - 2\frac{\mathbf{x}\mathbf{x}^T}{\|\mathbf{x}\|_2^2}\Big)$$

   Hint: Start with the directional derivative into a direction z, i.e. $(\nabla^2 F(x))z$.

4. Assume during the iterations of a the *Newton Method* descent we reach nearly an Eigenvector of A. Then for $\mathbf{x}$ an Eigenvector we are looking for a descent direction by solving the Newton equation

$$\nabla^2 F(\mathbf{x})\, d = -\nabla F(\mathbf{x})$$

   To what does the solution of this equation simplify in this case?

5. What does the Newton iteration look like for the situation in 4?
6. Interpret the result from 5 and analyze what the reason for this behavior is.

1.4 Problem 3

1. Implement a generic (simple) Newton iteration function of the form

   ```
   def my_newton(f, Df, x0, **kwargs)
   ```

   To look for a zero of the univariate function $f\colon \mathbb{R} \to \mathbb{R}$. Hence the function performs the newton iteration for `f` and its derivative `Df` starting from `x0` performing `maxiter` steps, where `maxiter` is a keyword argument from `**kwargs`. Provide a suitable default if no `maxiter` keyword is given.

2. Extend the function from the first part of this problem with an additional keyword `tol` to stop, if $|f(x_k)|$ is smaller in absolute value than the tolerance `tol`. What is a good default here to make the function work the same as in 1, i.e. that the default behavior (from 1) is not changed by this additional keyword?

3. Document the function properly.
4. Write at least two tests with simple functions. One should end with `maxiter` iterations, the other with a tolerance stop.

[ ]:

[ ]:

[ ]:

[ ]:

1.5 Problem 4

1. Write a function that uses `my_newton(f, Df, x0)` from Problem 3 to run the newton iteration on a grid of possible starting values from the complex square

$$\{x + \mathrm{i}y \mid -1 \leq x, y \leq 1\}$$

sampled equidistantly on `n` points. For example for $n = 3$ you get $\{-1+\mathrm{i}, \mathrm{i}, 1+\mathrm{i}, -1, 0, 1, -1-\mathrm{i}, -\mathrm{i}, -\mathrm{i}-1\}$ which are arranged in a mesh $x_{0,0}, ..., x_{2,2}$.

Obs! Python uses `j` for the complex unit.

In total, your function should be of the form

```
eval_newton(f,Df,n)
```

and should return an `n`-by-`n` array `A`, that is $A = (a_{ij})$, where $i, j = 1, \ldots, n$, of resulting values from the n^2 Newton iteration runs.

2. How can we formulate the solution of $z^3 = 1$, where $z \in \mathbb{C}$ as a problem that fits our setting here? State f and f' accordingly. If you now run `eval_newton(f,Df,n)` for example for `n=256`, how many different solutions do you expect? Let m be the answer to that question. Choose m favourite colors and create an image with $n \times n$ pixel, where a pixel gets the kth color if it converges to the kth of the solutions you have. Save/display this image in this notebook. If your code is fast enough, do this with `n=1024` instead.
3. Similarly to the last point, solve the equation $z^5 = 1$ using `eval_newton`, again with `n=256` (or `n=1024` if you dare) for a maximal number of iterations (a) of 5 and (b) of 15.

   From each of the resulting arrays A we again create an images. We proceed as follows: Since a trivial solution of $z^5 = 1$ is $z_0 = 1$, we want to see the difference in *angle* to this solution. Note that z_0 itself has an angle of 0 in polar coordinates in the complex plane.

   Compute and visualize the phase (angle) of a resulting value to z_0 (see `numpy.angle`) of each result $a_{ij} \in \mathbb{C}$ in both arrays and use this angle as a value in the image.

[ ]:

[ ]:

[ ]: