# exercises_4

February 5, 2021

# 1 Exercises 4: Nonlinear equations

**Alvar Wallström Lindell**

Date: **Feb 1, 2021** Submission deadline: **Feb 22, 2021**

If you want to have a nicer theme for your jupyter notebook, download the cascade stylesheet file tma4125.css and execute the next cell:

[5]:
```
from IPython.core.display import HTML
def css_styling():
    styles = open("tma4125.css", "r").read()
    return HTML(styles)

# Comment out next line and execute this cell to restore the default notebook
↪style
css_styling()
```

[5]:
```
<IPython.core.display.HTML object>
```

### 1.0.1 Problem 1

Consider the equation $x = f(x)$ where $f(x) = \frac{1}{2}x^2 + \frac{1}{2}\cos(x)$.

**a)** Show that $f$ has a unique fixed point $x^\star$ in $[0, 1]$.

*Solution:*

**b)** Take one step with the Fixed-point iteration method and $x^{(0)} = 0$. What is $x^{(1)}$?

*Solution:*

**c)** Estimate how many iterations you have to perform to ensure that the error $|x^{(k)} - x^\star| \leq 10^{-6}$.

*Solution:*

### 1.0.2 Problem 2

Recall that for systems of equations $\mathbf{x} = F(\mathbf{x})$ the Banach Fixed-point theorem says that if there is a closed subset $D \subseteq \mathbb{R}^2$ (note that we can have $D = \mathbb{R}^2$) such that $F(\mathbf{x}) \in D$ for every $\mathbf{x} \in D$ and $F \in C^1(D)$ with $\|J_F(\mathbf{x})\| \leq C < 1$ for every $\mathbf{x} \in D$ then

1. $F$ has a unique fixed point $\mathbf{x}^\star$ in $D$.

2. The Fixed-point iteration converges linearly to $\mathbf{x}^\star$ for every $\mathbf{x}^{(0)} \in D$.
3. The error after $k$ iterations is bounded by

$$\|x^{(k)} - x^\star\| \leq \tfrac{C^k}{1-C}\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|.$$

*Note: Here $\|\cdot\|$ is any norm (for example $\|\cdot\|_1$) and $J_F$ is measured in the corresponding matrix-norm (for example $\|\cdot\|_1$).*

Consider the equation $\mathbf{x} = F(\mathbf{x})$ where $F : \mathbb{R}^2 \to \mathbb{R}^2$ is defined as

$$F(\mathbf{x}) = \begin{pmatrix} \frac{1}{4}\sin(x_1 + x_2) \\ \frac{1}{4}\cos(x_1 - x_2) \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

**a)** Show that $F$ has a unique fixed point $\mathbf{x}^\star$ in $\mathbb{R}^2$.

*Solution:*

**b)** Take one step with the Fixed-point iteration method and $\mathbf{x}^{(0)} = (0,0)^T$. What is $\mathbf{x}^{(1)}$?

*Solution:*

**c)** Estimate how many iterations you have to perform to ensure that the error $\|\mathbf{x}^{(k)} - \mathbf{x}^\star\|_1 \leq 10^{-6}$.

*Solution:*

### 1.0.3 Problem 3

Consider the equation $F(\mathbf{x}) = 0$ where $F : \mathbb{R}^2 \to \mathbb{R}^2$ is defined as

$$F(\mathbf{x}) = \begin{pmatrix} 1 + x_1^2 - x_2^2 \\ 2x_1 - x_2. \end{pmatrix}$$

Take one step with Newton's method and $\mathbf{x}^{(0)} = (1/2, 1)^T$. What is $\mathbf{x}^{(1)}$?

*Solution:*

### 1.0.4 Problem 4

We will now attempt to solve a non-linear equation numerically. Consider the equation $f(x) = 0$ where $f(x) = \arctan x + 1$.

**a)** What is the root $x^\star$ of $f$?

*Solution:*

**b)** Implement Newton's method for the function $f$ in Python. The function should iterate until $|f(x^{(k)}| \leq$ `tol`. You should be able to change the tolerance `tol` and the initial guess $x^{(0)}$. If it converges, what is the error $|x^{(k)} - x^\star|$ after each iteration?

*Hint: It is recommended that you ensure that the program iterates at most some large number of times, say* 10000*. There is a skeleton of a code provided below.*

**c)** Try your method with tolerance `tol` $= 10^{-6}$ and the two initial guesses $x^{(0)} = 1.2, 1.4$. Does the iteration converge for both initial guesses?

*Solution:*

**d)** *Optional*

The root $x^\star$ of $f$ is a fixed point of the function $g(x) = x - \arctan(x) - 1$. Implement the Fixed-point method for the function $g$, with the ability to change $x^{(0)}$ and `tol`.

Does the iteration converge for `tol` $= 10^{-6}$ and the two inital guesses $x^{(0)} = 1.2, 1.4$? Feel free to test other values of $x^{(0)}$.

If it converges, what is the error $|x^{(k)} - x^\star|$ after each iteration?

How many iterations does it take to converge? Compare this to the number of iteration it takes for Newton's method to converge (if it converges).

*Solution:*

[7]:
```
##Post your code here##
```

**Prebuilt skeleton of a code**

[ ]:
```
import numpy as np
from math import atan

def newton(x):
    #Should return the next iterate after one Newton step.
    return x_new

def newton_iter(x, tol):

    while error>tol and iterations<10000:

        try:
            #Take one newton step
        except OverflowError: #This prints out a message if the numbers are too␣
↪large to handle.
            print("Did not converge.")
            break
```