MTMM.00.005 Numerical Methods
Practical lesson 6: Newton's method for solving systems of equations.

March 19, 2019

To get a positive result, student has to solve all of the exercises given in practical lessons, explain the programming code and answer the questions.

The deadline for the sixth set of exercises is March 26th.

NB! In order to use the nonstandard packages as *numpy* and *matplotlib* one could use Command Prompt and type

```
H:\nm> "C:\Program Files\Anaconda3\python" <filename>.py
```

Here `H:\nm` is the folder for `<filename>.py`.

Introduction

Consider the system

$$\begin{cases} f_1(x_1,\dots,x_n)=0, \\ \dots \\ f_n(x_1,\dots,x_n)=0, \end{cases}$$

which can be written as $F(x)=0$, where $x=(x_1,\dots,x_n)$ is the vector of unknowns, $F:\mathbb{R}^n\to\mathbb{R}^n$ and $F(x)=(f_1(x),\dots,f_n(x))$. Let x^* be the solution of $F(x)=0$, and $x^m=(x_1^m,\dots,x_n^m)$, $m=0,1,\dots$ the sequence of consecutive approximations. Convergence $x^m\to x^*$ means that $\|x^m-x^*\|\to 0$, where $\|\cdot\|$ is any norm in $\mathbb{R}^n$. Most common norms are

$\|x\|_\infty=\max\{|x_1|,\dots,|x_n|\}$ (∞-norm)

$\|x\|_1=|x_1|+\dots+|x_n|$ (1-norm)

$\|x\|_2=\left(|x_1|^2+\dots+|x_n|^2\right)^{1/2}$ (2-norm)

$\|x\|_p=\left(|x_1|^p+\dots+|x_n|^p\right)^{1/p}, 1\le p<\infty$ (p-norm)

Newton's method

Consider the system

$$F(x)=0.$$

Let f_i, $i=1,\dots,n$, be differentiable, i.e., F is differentiable. Denote

$$F'(x)=\begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \dots & \frac{\partial f_1}{\partial x_n}(x) \\ \dots & \dots & \dots \\ \frac{\partial f_n}{\partial x_1}(x) & \dots & \frac{\partial f_n}{\partial x_n}(x) \end{pmatrix}.$$

According to initial approximation $x^0\in\mathbb{R}^n$ the approximations

$$x^{m+1}=x^m-\left(F'(x^m)\right)^{-1}F(x^m),\quad m=0,1,2,\dots,$$

are computed. Here $\left(F'(x^m)\right)^{-1}$ is the inverse matrix.

The step can be performed, if $(F'(x^m))^{-1}$ exists, i.e., $\det F'(x^m) \neq 0$.

In the case of modified Newton's method

$$x^{m+1} = x^m - \left(F'(x^0)\right)^{-1} F(x^m), \quad m = 0, 1, 2, \ldots.$$

Inverse of a matrix

In case of small systems the inverse of A could be found by using *Numpy* packages *linalg* or *matrix*. If A is of type numpy array, then use `inv(A)`, if A is of type numpy matrix, then use `A.I`. See `https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.linalg.inv.html` and `https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.matrix.html`.

In case of bigger systems finding the inverse matrices is quite time-consuming, thus the step

$$x^{m+1} = x^m - \left(F'(x^m)\right)^{-1} F(x^m)$$

should be replaced by solving the linear system

$$F'(x^m)x^{m+1} = F'(x^m)x^m - F(x^m),$$

with respect to x^{m+1}. Here, in order to avoid the multiplication of matrix $F'(x^m)$ and vector x^m (which takes n^2 operations), one could solve the linear system

$$F'(x^m)z = F(x^m)$$

with respect to z and find

$$x^{m+1} = x^m - z = x^m - (F'(x^m))^{-1}F(x^m).$$

Linear systems could be solved with ordinary iteration method or by using `solve` from the package *linalg*. See `https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.linalg.solve.html`

Exercise. Use Newton's method and modified Newton's method to solve

$$\begin{aligned} 12x_1 - 3x_2^2 - 4x_3 &= 7.17, \\ x_1^2 + 10x_2 - x_3 &= 11.54, \\ x_2^3 + 7x_3 &= 7.631. \end{aligned}$$

First, write the system in form $F(x) = 0$ and find $F'(x)$. Terminate the iteration process if $\|x^m - x^{m-1}\| \leq 10^{-5}$. Print out all approximations, the norms $\|x^m - x^{m-1}\|$ and the number of iterations needed.