Exercise #4

13. September 2022

Exercises marked with a (J) should be handed in as a Jupyter notebook.

Problem 1. (Newton's method for systems)
(J) We are given the system of nonlinear equations

$$\begin{aligned} x_1^3 + x_1^2 x_2 - x_1 x_3 &= -6, \\ e^{x_1} + e^{x_2} - x_3 &= 0, \\ x_2^2 - 2x_1 x_3 &= 4. \end{aligned}$$

Write a python program for solving this system by Newton's method. Use $\mathbf{x}_0 = (-1, -2, 1)$ as a starting value, and iterate until $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_\infty < 10^{-6}$. How many iterations are needed in this case?

You are encouraged to experiment a bit with different starting values.

Problem 2. (Periodic functions)

Recall that a function $f : \mathbb{R} \to \mathbb{R}$ is called periodic, if there exists $p > 0$ (a period of f) such that

$$f(x + p) = f(x) \text{ for all } x \in \mathbb{R}.$$

Moreover, the smallest positive number for which this statement holds (if it exists), is called the fundamental period of f.

a) Decide whether the following statement is true or false, and then find either a proof or a counterexample: Every periodic function has a fundamental period.

b) What is the fundamental period of the following functions:

- $f(x) = \cos(x)$
- $f(x) = \sin(\pi x)$
- $f(x) = \cos(\frac{2\pi}{m}x) + \sin(\frac{2\pi}{n}x), \quad n, m \in \mathbb{N}.$

Problem 3. (Fourier series)

For each of the 2π periodic functions below, sketch the function over $-3\pi < x < 3\pi$ and find their Fourier series. In each case, plot the truncated series

$$S_N(x) = a_0 + \sum_{n=1}^{N}(a_n \cos(nx) + b_n \sin(nx))$$

for $N = 5$, $N = 20$ and $N = 100$.

a) $f(x) = \begin{cases} 0 & \text{if } -\pi < x < 0 \text{ or } \frac{\pi}{2} < x \leq \pi, \\ x & \text{if } 0 \leq x \leq \frac{\pi}{2}. \end{cases}$

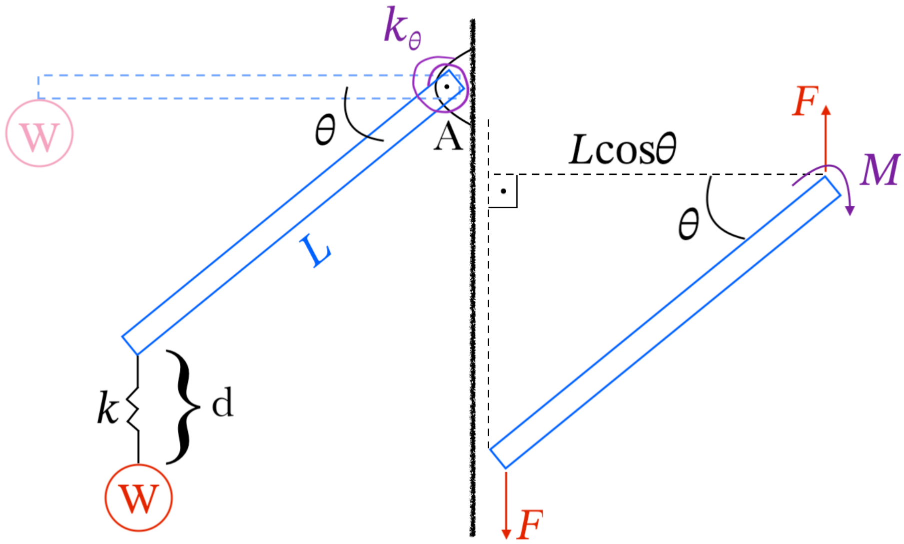
b) $f(x) = \begin{cases} 0 & \text{if } -\pi < x < 0, \\ x & \text{if } 0 < x < \frac{\pi}{2}, \\ \pi - x & \text{if } \frac{\pi}{2} < x \leq \pi. \end{cases}$

c) $f(x) = \begin{cases} -\pi - x & \text{if } -\pi < x < -\frac{\pi}{2}, \\ x & \text{if } -\frac{\pi}{2} < x < \frac{\pi}{2}, \\ \pi - x & \text{if } \frac{\pi}{2} < x \leq \pi. \end{cases}$

The next exercises are optional and should not be handed in!

Problem 4. (Multivariate Newton's Method)

Figure 1 illustrates the static equilibrium problem of a structural system. At the tip of a weightless bar with length L lies a mass whose weight W pulls the system down. The rigid bar can rotate about its support point A by an angle $0 \leq \theta \leq \pi/2$. Then, an angular spring responds with an opposing moment $M(\theta) = k_\theta \theta$, with $k_\theta > 0$ being a given constant. The mass and the bar are connected by a linear spring that responds to any separation d with a force $F(d) = kd$, with $k > 0$ being another given constant. Thus, equilibrium of forces for the mass yields $F(d) = W$, while equilibrium of moments for the bar gives $F(d)L\cos\theta = M(\theta)$.



(a) General setup. (b) Free body diagram for the bar.

Figure 1: Static equilibrium problem for a bar-mass system.

To find the *unknown* equilibrium configuration (θ, d), we have to solve a non-linear system:

$$\begin{aligned} kLd\cos\theta - k_\theta\theta &= 0 \\ kd - W &= 0\,, \end{aligned}$$

in which (W, L, k, k_θ) are all considered as *given* constants.

This is what engineers normally call a *geometric non-linearity*, since the nonlinear behaviour comes entirely from the trigonometry of the problem. For small displacements, it is common to use the "small-angle approximation" $\cos\theta \approx 1$. In this exercise, we will compare this linearised approach to the non-linear one.

a) Using the simplification $\cos\theta \approx 1$, find θ and d in terms of the constants (W, L, k, k_θ).

b) Derive the Jacobian matrix $J(\theta, d)$ for system (4), also in terms of (W, L, k, k_θ).

c) Let's now assign values to the parameters: $L = 1$ m, $k = 2$ N/m, $k_\theta = 3$ Nm/rad and $W = 4$ N. Using these values, evaluate the d and θ obtained in task a). Then, using them as initial guesses, compute *by hand* the first iteration of Newton's method.

d) Now, compute a few more iterations until meeting a tolerance of 10^{-6} (you can use the Jupyter notebook 04-*Nonlinear-eqs.ipynb* to make your life easier). Then, compare the linearised θ calculated previously to the one obtained iteratively: which one is larger?

Problem 5. (Newtons method and nonlinear BVPs)

In the note on boundary value problems, a chemical reactor example was discussed. The problem describing the reactant's concentration is

$$\alpha u_{xx} - v u_x - \kappa u^\gamma = 0, \qquad x \in [0, L],$$
$$u(0) = u_0, \qquad u(L) = u_L$$

with $\gamma \in \mathbb{N}$ and (α, v, κ) being positive constants.

For a given mesh size $h = L/N$, a central finite difference scheme will result in $N-1$ nonlinear equations

$$\alpha\left(\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2}\right) - v\left(\frac{U_{i+1} - U_{i-1}}{2h}\right) - \kappa U_i^\gamma = 0\,, \quad i = 1, \dots, N-1\,,$$

in which $U_0 = u_0$ and $U_N = u_L$. Multiplying each equation by $2h^2$ leads to

$$f_i(\mathbf{U}) := (2\alpha + vh)U_{i-1} - (4\alpha + 2\kappa h^2 U_i^{\gamma-1})U_i + (2\alpha - vh)U_{i+1} = 0\,, \quad i = 2, \dots, N-2\,.$$

After inserting the boundary values $U_0 = u_0$ and $U_L = u_L$, we obtain moreover for $i = 1$ and $i = N-1$ the equations

$$f_1(\mathbf{U}) := (2\alpha + vh)u_0 - (4\alpha + 2\kappa h^2 U_1^{\gamma-1})U_1 + (2\alpha - vh)U_2 = 0,$$
$$f_{N-1}(\mathbf{U}) := (2\alpha + vh)U_{N-2} - (4\alpha + 2\kappa h^2 U_{N-1}^{\gamma-1})U_{N-1} + (2\alpha - vh)u_L = 0.$$

Therefore, starting from an initial guess $\mathbf{U}_0$, we iterate by solving

$$J(\mathbf{U}_k)\boldsymbol{\Delta}_k = -\mathbf{f}(\mathbf{U}_k)$$

and updating $\mathbf{U}_{k+1} = \mathbf{U}_k + \boldsymbol{\Delta}_k$. Since each f_i depends only on U_{i-1}, U_i and U_{i+1}, most terms of the Jacobian $J(\mathbf{U})$ will be zero, except for

$$J_{i,i-1} = 2\alpha + vh\,, \quad J_{i,i} = -4\alpha - 2\gamma\kappa h^2 U_i^{\gamma-1} \;\text{ and }\; J_{i,i+1} = 2\alpha - vh\,.$$

Therefore, the $(N-1)\times(N-1)$ Jacobian matrix will look like

$$J(\mathbf{U}) = \begin{pmatrix} b+cU_1^{\gamma-1} & d & 0 & 0 & \cdots & 0 & 0 & 0 \\ a & b+cU_2^{\gamma-1} & d & 0 & \cdots & 0 & 0 & 0 \\ 0 & a & b+cU_3^{\gamma-1} & d & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a & b+cU_{N-2}^{\gamma-1} & d \\ 0 & 0 & 0 & 0 & \cdots & 0 & a & b+cU_{N-1}^{\gamma-1} \end{pmatrix},$$

in which

$$\begin{aligned} a &= 2\alpha + vh\,, \\ b &= -4\alpha\,, \\ c &= -2\gamma\kappa h^2\,, \\ d &= 2\alpha - vh\,. \end{aligned}$$

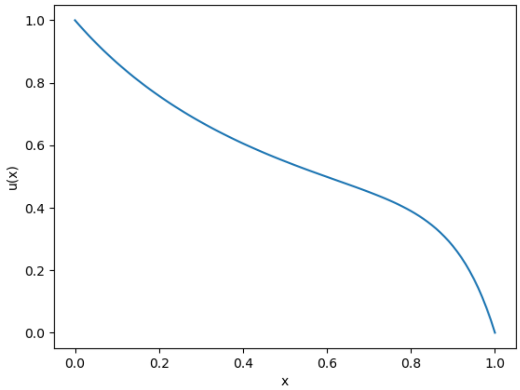
This problem illustrates how computationally demanding Newton's method can be, in practice.

In this exercise, you are supposed to write a python program for solving this problem with a general choice of parameters $\alpha, v, \kappa, \gamma$. The program will consist of the following elements:

- Set up the system of $N-1$ nonlinear equations to be solved.
- Set up the Jacobi-matrix for this system.
- Use Newton's method to solve the system of nonlinear equations you found in a). You may stop the iterations when $\max_i |\Delta_i| < \text{Tol}$, where Tol is some prescribed tolerance. As initial value for the iterations, use the straight line between $(0, u_0)$ and (L, u_L).

Test your program with the parameters $\alpha = 0.1$, $v = 0.5$, $\kappa = 10$, $n = 3$, $L = 1$, $u_0 = 1$ and $u_L = 0$. Choose different values of N, e.g. $N = 5$, 10 and 50. Let Tol=10^{-8}. Plot the numerical solution for $N = 50$. How many iterations are needed in each case?

Hint 1: The solution with the $\alpha = 0.1$, $v = 1$, $\kappa = 2$ and $n = 2$ will look something like



Hint 2: You may start with $n = 1$. In this case, your system of equations is linear, and the Newton iterations should converge in one iteration. (Why?)

Hint 3: In the Jacobian, notice that only the diagonal needs to be updated in each iteration.