Assignments IV

Exercise 1 *Write a* MATLAB*-function which* continues *the solution of an equation* $f(x) = 0$*, where* $f : \mathbb{R}^n \to \mathbb{R}^{n-1}$.

A system of $n-1$ (non-linear) equations with n unknowns has generically a one-dimensional set of solutions (a curve in $\mathbb{R}^n$). The aim is to calculate this curve, given a starting point $x_0 \in \mathbb{R}^n$ such that $f(x_0) \approx 0 \in \mathbb{R}^{n-1}$. According to the implicit function theorem the curve is smooth as long as the matrix of partial derivatives $(Df)_{ij} = \frac{\partial f_i}{\partial x_j}$ has maximal rank (i.e. $n-1$).

The input variables are the function f, its derivative Df, a starting point x_0, an initial step size h_0, and the maximum number of steps N. The output is a matrix of zeros of f (a "vector" of points on the solution curve).

1. Basic idea: determine in which direction v the curve extends. Take a step of size h_0 in that direction: $\tilde{x}_1 = x_0 + h_0 v$. Calculate (using Newton's method) a new point x_1 close to $\tilde{x}_1$ on the curve. Repeat N times (check that the direction does not flip).
2. Of course, first determine a point close to x_0 lying on the curve.
3. By changing the sign of h_0 the curve should be followed in the opposite direction.
4. When x_1 is not close enough to $\tilde{x}_1$ or when Newton's method fails to converge, decrease the step size h. On the other hand, when a step is successful you can increase the step size somewhat. When the step size becomes too small, the function terminates before the maximum number of steps is reached.
5. Make sure that the minimum and maximum step size are optional input parameters.
6. If an empty matrix `[]` is provided as the (input) derivative Df, then the function should calculate this derivative numerically using `numjac`.

Exercise 2 *Write a* MATLAB*-function which distills a phone number from a sound signal.*

A phone number is converted by a phone into a signal via a method that is based on the Dual Tone Multi-Frequency system. In this system every digit is associated with two frequencies f_1 and f_2, which can be found in the table below (frequencies are in Hertz)

$f_1 = 697$	1	2	3
$f_1 = 770$	4	5	6
$f_1 = 852$	7	8	9
$f_1 = 941$	$*$	0	#
	$f_2 = 1209$	$f_2 = 1336$	$f_2 = 1477$

Ideally the signal is zero when no key is pressed, while for the duration of the key-press the signal is a linear combination of two sines with the frequencies provided above. Furthermore, there is noise in the signal. Have a look at the file `signal.mat` for an example signal. Also zoom in on a

piece where a key is pressed to get an idea about of the signal. You can listen to the signal with `sound(signal,8192)`, where 8192 is the sampling rate of the signal. You will notice that not all keys are pressed equally long and also the time between two digits is not uniform. Moreover, the signal is a bit noisy and not all digits have equal amplitude. For this test signal the digits are 15086477001.

The input variables are a vector with the signal, and the sampling rate. The output is a row vector with the digits of the phone number.

1. The first step is to find the pieces in the signal which represent the digits (and to determine the number of digits).
2. From the piece of signal representing one digit, the function extracts the frequencies using the Fourier transform.
3. Finally, the function determines the digit associated with the frequencies.
4. The combination of these steps leads to the phone number.
5. You can download the file `phonenumbers.mat` from the website, which contains signals `number`n, for $n = 1, 2, 3, 4, 5$. They all have a sampling rate of 4096 samples per second (i.e. less than in the file `signal.mat`). The challenge is to extract the phone numbers from these signals using your function.

Exercise 3 *Write a* MATLAB*-function which makes a page ranking of webpages.*

To determine which websites are important GOOGLE uses roughly the following algorithm. First a connection matrix is made, where $A_{ij} = 1$ if page j contains a link to page i; otherwise $A_{ij} = 0$. Next, a Markov process is considered, where you follow with probability p an arbitrary link on the current page, whereas with probability $1 - p$ you jump to a completely arbitrary page (usually $p = 0.85$). Let A be an $(n \times n)$ matrix, where the j^{th} column contains c_j 1-s (i.e. page j contains c_j links). Then this Markov process is described by the matrix

$$B_{ij} = \begin{cases} \frac{p}{c_j} A_{ij} + \frac{1-p}{n} & \text{if } c_j \neq 0, \\ \frac{1}{n} & \text{if } c_j = 0. \end{cases}$$

Such a matrix (with $B_{ij} > 0$ for all i, j) has a positive dominant eigenvalue, i.e., the eigenvalue with largest absolute value is unique, real, and positive. Moreover, the corresponding eigenvector v has strictly positive elements ($v_j > 0$). This eigenvector represents the limiting long term distribution of this Markov process. The element v_j of this eigenvector is the page rank of page j.

The input variables are an $(n \times n)$ sparse connection matrix A, and optionally a list of associated websites (a cell array with n strings). The output is a vector with page rankings. It is **important** that the input matrix can be very large (GOOGLE has $n > 10^{10}$). Therefore, your function can *not* use matrix operations with full matrices. In particular, the matrix B can not even be formed.

1. Use the power method to determine the page rankings.
2. The function should also show (with `bar`) a picture of the result.
3. If the optional lists of websites is given as an input variable, the function should produce a list of the top sites (and their page rank).
4. On the course website you find the file `wwwdata.mat` containing the sparse $(n \times n)$ matrices `sp`n for $n = 100, 500, 2000, 10000, 50000, 200000$, with associated lists of websites `url`n. Depending on the speed and memory of your computer you can test your function on these matrices.