Numerical Methods
Assignment 04

Georgios Christos Chouliaras (2592496)

April 23, 2017

Exercise 1

The function

The function `Chouliaras_assignment4_exercise1(f, df, x0, h0, N, minh, maxh)` continues the solution of an equation $f(x) = 0$ where $f : \mathbb{R}^n \to \mathbb{R}^{n-1}$. Given a function f, its derivative and a starting point, the function calculates the solution curve of the function.
The inputs are the function `f`, its derivative `df`, a starting point `x0`, an initial step size `h0`, the maximum number of steps `N` and optionally the minimum step size `minh` and the maximum stepsize `maxh`. If the stepsize limits are not given, the function assigns default values $minh = 10^{-5}$ and $maxh = 2$. Also,if an empty matrix `[]` is provided as the derivative, the function calculates the derivative numerically using `numjac`.
The output of the function is a matrix of zeros of f which contains the points on the solution curve.
As an example, in order to calculate the solution curve of $f = x(1)^4 + x(2)^4 - 1$ with $x_0 = [1; 0]$, initial step size $h_0 = 0.01$ and maximum 5000 number of steps one should type:

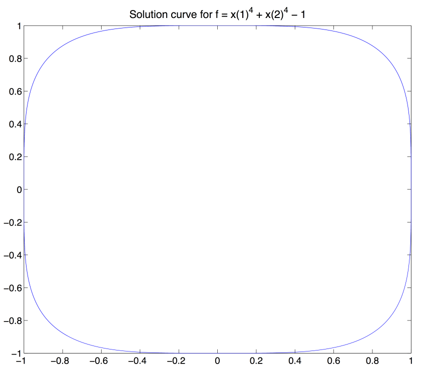
```
[thecurve] = Chouliaras_assignment4_exercise1(@(x) x(1)^4 + x(2)^4 -1, [], [1;0], 0.01, 5000)
```

The method

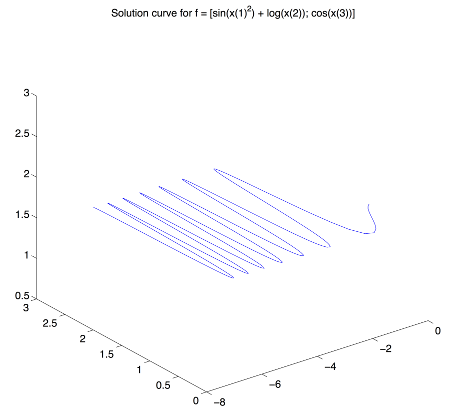
In order to calculate the solution curve of a function $f : \mathbb{R}^n \to \mathbb{R}^{n-1}$ given a starting point $x_0 \in \mathbb{R}^n$ such that $f(x_0) \approx 0 \in \mathbb{R}^{n-1}$ the first step is to "walk" a bit on the direction of the tangent vector v which is the kernel of the derivative matrix and is perpendicular to the gradient. We walk on the direction by taking a step of size h_0 in that direction: $\tilde{x}_1 = x_0 + h_0 v$. In order to find again a point on the solution curve we put a hyperplane and we want the next point to be on that hyperplane and also on the solution curve. This can be translated into two conditions: $f(x) = 0$ near $\tilde{x}_1$ and $v^T(x - \tilde{x}_1) = 0$. Thus, we define a function g which contains these two conditions and we apply Newton's method on that function and its derivative with starting point $\tilde{x}_1$. This is repeated at most N times. When $\tilde{x}$ goes to the wrong direction we decrease the step size, while if it goes to the right direction we increase somewhat the step size. If the step size becomes too small the function stops before reaching the maximum number of iterations.

Discussion & Illustration

In figure 1a can be seen the solution curve for the function $f(x_1, x_2) = x_1^4 + x_2^4 - 1$ with $x_0 = [1; 0]$, $df(x) = [4x_1^3\ 4x_2^3]$, maximum number of steps $N = 5000$ and initial step size $h_0 = 0.01$. It must be noted that every function might require a different number of maximum steps in order to produce an appropriate solution curve. For example, if the maximum number of steps was set to 1000 we would not see the whole solution curve but almost the half of it. The necessary maximum number of steps also depends on the initial step size, since by choosing a very small initial step size we need more number of steps in order to get the complete solution curve. It must be also mentioned that at some point the new tangent vector v is almost opposite to the old v and when this is the case we must flip v. This can be checked by looking at the point when the inner product of these tangent vectors is negative, since in the case that it is zero, we know that they are perpendicular. In figure 1b can be seen an example with two equations and three unknowns: $f(x_1, x_2, x_3) = [sin(x_1)^2 + log(x_2); cos(x_3)]$ with vector of initial guesses $x_0 = [0; 1; \pi/2]$, $h_0 = 0.1$ and $N = 5000$. As can be seen by the solution curve which is not closed in this case, the direction changes very often, and due to this the minimum step size must be set very low ($\approx 10^{-10}$), otherwise the functions stops without reaching the maximum iterations since the step size cannot decrease any further. Additionally, it is noted that in order to get a precise solution curve and thus, a precise picture of the solution curve, in the condition $f(x) \leq threshold$ after the Newton method, the *threshold* must be very small ($\approx 10^{-10}$). If it is larger, the picture that is produced is very distorted, due to the poor approximations.



(a)



(b)

.

Figure 1: (a) The solution curve for the function $f(x_1, x_2) = x_1^4 + x_2^4 - 1$ with $x_0 = [1; 0]$. (b) The solution curve for $f(x_1, x_2, x_3) = [sin(x_1)^2 + log(x_2); cos(x_3)]$ with $x_0 = [0; 1; \pi/2]$

Exercise 2

The function

The function `Chouliaras_assignment4_exercise2(thesignal,rate)` extracts a phone number from a sound signal. Given a sound signal and its sampling rate, this function identifies the digits in the signal and finds the low and high frequency (in Hertz) associated with each digit. Then, it uses these frequencies to extract the digits based on the Dual Tone Multi-Frequency system. The input of the function is a vector containing the sound signal (`thesignal`) and the sampling rate of the signal (`rate`).
The output of the function is a row vector (`digits`) containing the digits of the phone number. As an example, in order to find the digits associated with the signal `number2` from the file `phonenumbers.mat` with a sampling rate of 4096, one should type:

```
 [digits] = Chouliaras_assignment4_exercise2(number2,4096)
```

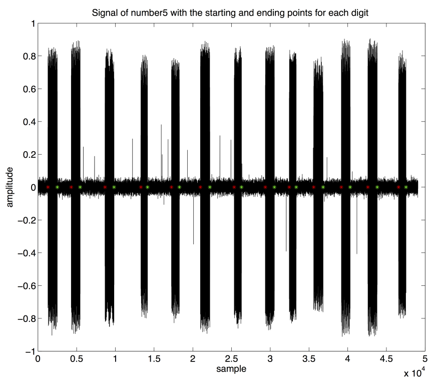
The method

Since the signal contains noise, the first thing that needs to be done is to isolate the digits in the sound signal, in other words to find the pieces in the signal that represent the digits without including noise. To determine the starting point of a digit in the signal we set a threshold of 0.6 (0.6 is chosen arbitrarily, 0.55 could also work) and we find the point in which the absolute value of the signal is greater than this threshold. The ending point of a digit in the signal is found by checking when the maximum of the absolute value of 50 consecutive values of the signal is less than the threshold. In this way we find the digits in the signal and then we use the Fourier transform to extract the frequencies of each digit. The first two peaks on the magnitude of the FFT are the low and high frequency respectively. In order to find the digits associated with these frequencies we look at the differences between the frequencies found and the frequencies associated with each digit from the Dual Tone Multi-Frequency system. To find the correct digit we check in which cases the 2-norm of the differences is the minimum and we store these digits in a vector. This vector `digits` consists the output of the function.

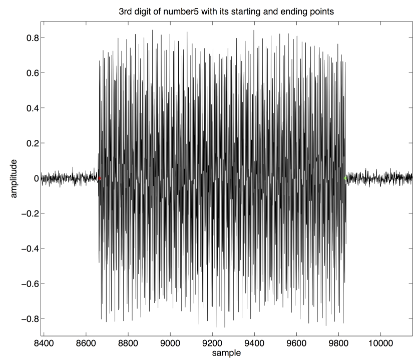
Discussion & Illustration

In order to provide a better view of the extraction of the digits in the signal, the signal of `number5` is plotted in figure 2a along with marks that indicate the starting and ending points of each digit. The 3rd digit is zoomed in figure 2b in order to see the starting and ending points in greater detail. As it can be seen the start and the end of the digit are accurate, since noise is not included. In figure 2c is plotted the magnitude of the Fourier transform against the frequency for the 3rd digit of `number5`. The first peak from the left corresponds to the low frequency while the second peak to the high frequency. It can be observed that the low frequency in this case is around 700 while the high frequency around 1200 and it corresponds to the digit 1, something that it is verified from the numbers provided below. Moreover, from figure 2c can be observed that most of the noise has been excluded, something that verifies the efficiency of the starting and ending points.
We extract the phone numbers from the signals in `phonenumbers.mat` using the function: **Number 1:** 6241111, **Number 2:** 0205258780, **Number 3:** 0657877307, **Number 4:** 0031595526526, **Number 5:** 0017035456700.

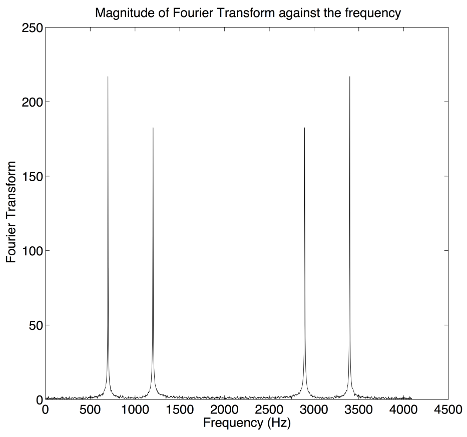
It must be mentioned that a limitation of the function is that it might not work with signals different than those in the `phonenumbers.mat` and the `signal.mat` since the threshold of 0.6 was chosen based on the given signals. Thus, this threshold might need to be altered in order for the function to work also with other signals.
In order to evaluate the precision of the algorithm, the minimum values of $(f_{low} - \tilde{f}_{low})^2 + (f_{high} - \tilde{f}_{high})^2$ were stored in a vector for testing purposes, where f_{low} is the low frequency from the Dual Tone Multi-Frequency system, while $\tilde{f}_{low}$ is the extracted frequency from the function. It was observed that in most cases this function resulted to a value 0, while the highest value is 434 for the first and last digit of `number5`. The very low values of these differences verify the precision of the algorithm.



(a) *The signal of `number5`.*



(b) *The signal of the 3rd digit.*



(c) *Magnitude of the Fourier transform against the frequency for the 3rd digit.*

Figure 2: (a) The signal of `number5` with the starting points (red star) and the ending points (green star) for each digit. (b) Zoom in the 3rd digit of `number5` with its starting point (red star) and the ending points (green star) for this digit. (c) Fourier Transform against the Frequency (in Hertz) for the 3rd digit of the `number5`. The low frequency is around 700 and the high frequency around 1200 which corresponds to the digit 1.

Exercise 3

The function

The function `Chouliaras_assignment4_exercise3(A,websites)` makes a page ranking of webpages. Given a $n\ x\ n$ sparse connection matrix A which contains the links between the websites ($A_{ij} = 1$ if page j contains a link to page i and 0 otherwise), the function uses the power method in order to compute the largest eigenvalue of a matrix B which describes a Markov process. The eigenvector which correspond to this largest eigenvalue is the page rank.
The input of the function is a $n\ x\ n$ sparse connection matrix `A` and optionally, a cell array `websites` with n strings that represent the associated websites .
The output is a vector `rankings` containing the page rankings. The function also produces a bar chart with the page rankings. Additionally, if `websites` is given as input, the function also produces a list with the top 5 websites and their page rank.
As an example, in order to find the page ranks of the sparse connection matrix `sp500` from the file `wwwdata.mat` with the associated list of websites `url500`, one should type:
`[rankings] = Chouliaras_assignment4_exercise3(sp500,url500)`

The method

The page rank of a website can be described as the long-run probability that a user will reach this website. We consider a Markov process where with probability $p = 0.85$ we follow a link on the current page, and with probability $1 - p$ we jump to another page. This Markov process is described by a matrix $B_{ij} = \frac{p}{c_j}A_{ij} + \frac{1-p}{n}$ if $c_j \neq 0$ or $B_{ij} = \frac{1}{n}$ if $c_j = 0$, where c_j determines how many $1's$ contains the j^{th} column of matrix A. Matrix B is a stochastic matrix and in order to find the page rankings, we need to find the eigenvector v whose element v_j is the page rank of page j. To find this eigenvector, the Power Method is used, an iterative method that finds the eigenvalue with the largest magnitude (dominant eigenvalue). We start with a starting point y_0 and we compute $x_1 = By_0$. Then we define $y_1 = \frac{x_1}{||x_1||}$ and we iterate this procedure while $||y_k - y_{k-1}|| > eps$. The procedure converges to the largest eigenvalue and the corresponding eigenvector is the page rank. The issue here is that B is so large that cannot even be formed and hence we do the following procedure in order to use the power method without forming B: We create a vector of size n that contains the c_j's and using this we create an auxiliary matrix H (sparse and diagonal) which contains $\frac{1}{c_j}$ in the diagonal if $c_j \neq 0$ and 0 otherwise. Next, using the power method we compute $x_1 = pHAy_0 + \frac{1-p}{n}\mathbf{e}$, where $\mathbf{e}$ is a vector with n 1's. Then we normalize the vector by dividing with the sum of the components in order for the vectors to sum up to 1: $y_1 = \frac{x_1}{\sum_k^n x_1(k)}$. After that, we use y_1 as the initial guess and we iterate this procedure until convergence to the vector of page ranks.

Discussion & Illustration

An illustration of the bar-chart produced by the function for input the sparse matrix `sp2000` can be seen in figure 3. In this bar chart the websites with their page ranks are being presented using `[rankings] = Chouliaras_assignment4_exercise3(sp2000,url2000)`. Along with the chart, the function also produces a list with the top 5 websites and their ranks, which can be seen in table 1. These 5 websites correspond to the 5 highest peaks of figure 3. The function also performs well in terms of speed, more specifically given as input the large matrix `sp200000` which is a sparse matrix of size 200000x200000, it runs in almost 4 seconds. For the smaller matrices in `wwwdata.mat` the function runs in less than a second. Surely, the speed depends also on the computer power, however even in slower computer the function operates fast due to its efficient structure.

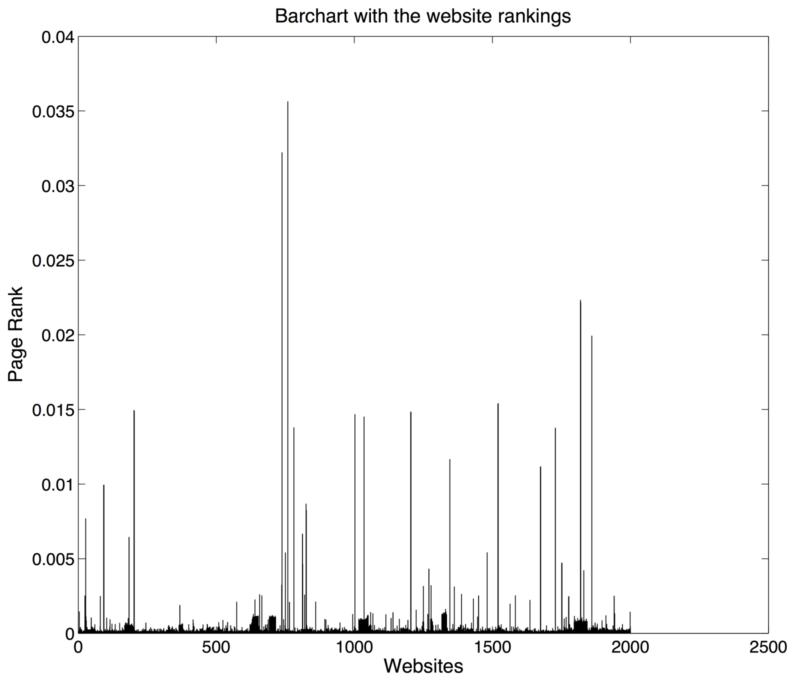


Figure 3: Bar-chart with the websites from the `sp2000` matrix with their page ranks.

Website	Page Rank
`www.top.ranking`	0.0356
`www.popular.but.boring`	0.0322
`www.small.world/site15601.html`	0.0223
`www.2016.nl`	0.0222
`www.top5.nl`	0.0199

Table 1: The top 5 websites with their rankings, as listed by the function for inputs [`sp2000`,`url2000`]