SHANGHAI JIAO TONG UNIVERSITY, UM-SJTU JOINT INSTITUTE

Introduction to Numerical Analysis Project2

Yu Cang
018370210001

August 2, 2018

1 TASK 1

Note the symmetry of x_1, for any solution (a,b), $(-a,b)$ is also valid.
If no special claim, the norm function for a vector is taken as the 2-norm.

(a) For fixed-point iteration, the iteration function G is taken as

$$G \triangleq \begin{bmatrix} \sqrt{x_2} \\ \sqrt{1-x_1^2} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x \tag{1.1}$$

The iteration converges to $(0.7862, 0.6180)$ with intial guess $x_{init} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$ and tolerance set to 10^{-10}.

(b) The newton iteration is implemented as mentioned in the project sheet, each time the linear system

$$J_F(x_k) w_k = y_k \tag{1.2}$$

is solved with the built-in function 'linsolve' inside matlab.
The iteration converges to $(0.7862, 0.6180)$ with intial guess $x_{init} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$ and tolerance set to 10^{-10}.

(c) The input of the Broyden's method is a little bit different from that in newton or fixed-point. The initial Jacobi matrix should be provided, denoted as A_0. In my implementation, it is given as

$$A_0 = J_F(x_0) \tag{1.3}$$

An extra calculation on x_1 should be done before the iteration loop as the iteration process requires both x_0 and x_1. It is approximated through newton's method in my implementation as

$$x_1 \approxeq x_0 - A_0^{-1} F(x_0) \tag{1.4}$$

After all the preparation work, the iteration loop can be carried out as mentioned in the project sheet, and the key part is the application of Sherman-Morrison's formula to calculated the inverse of A_k iteratively as

$$A_k^{-1} = A_{k-1}^{-1} + \frac{(s_k^T A_{k-1}^{-1} y_k) s_k^T A_{k-1}^{-1}}{s_k^T A_{k-1}^{-1} y_k} \tag{1.5}$$

where $s_k = x_k - x_{k-1}$ and $y_k = F(x_k) - F(x_{k-1})$. And the iteration of x_k is given as

$$x_{k+1} = x_k - A_k^{-1} F(x_k) \tag{1.6}$$

The iteration converges to $(0.7862, 0.6180)$ with intial guess $x_{init} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$ and tolerance set to 10^{-10}.

(d) With numerical tests, newton's method out-performs the other two in terms of time cost.

2 TASK 2

With trival inequality analysis, the exact solution is given as $x = [0.5, 0, -\frac{\pi}{6}]$, which can be used for verification.

2.1 SOLVE DIRECTLY

Applying the fixed-point iteration, newton's method and the broyden's method to this nonlinear system respectively, with initial guess given as $x_0 = [0.10.10.1]^T$, and tolerance set to 10^{-10}, the iteration process all converge to the exact solution as

$$\begin{aligned} x_1 &= 0.5 \\ x_2 &= 3.5440 \times 10^{-11} \\ x_3 &= -0.5236 \end{aligned} \tag{2.1}$$

The iteration function for fixed-point iteration is given as

$$G = \begin{bmatrix} \frac{0.5+cos(x_2 x_3)}{3} \\ \frac{\sqrt{(0.25-x_1^2)}}{25} \\ \frac{\frac{3-10\pi}{3} - e^{-x_1 x_2}}{20} \end{bmatrix} = x \tag{2.2}$$

2.2 DOWNHILL SIMPLEX ALGORITHM

In general, the initial value for such a non-linear system is hard to know, even the range of each component is hard to estimate. In this case, the downhill simplex method can be applied to produce a initial guess for further calculation.
The downhill simplex algorithm was implemented in my code. The kernel idea of this algorithm is to minimize the performance of the worst vertex gradually.
Inside the iteration loop, serval operations guide the simplex to move towards the minimum:

1. reflection
2. expansion
3. contraction
4. shrink

These operations are designed for different ranges respectively. The whole range of the function is divided by 3 points:

1. the best
2. the last-but-one
3. the worst

Details of the implement will not be discussed here and can be referenced clearly from the code.

2.3 SOLVE WITH INITIALIZATION

Before using the downhill simplex to find a fine initial guess, a targeting function $f : \mathbb{R} \to \mathbb{R}$ is taken as

$$f(X) = ||F(x)||_2 \tag{2.3}$$

inside the matlab code as the evaluating function for the algorithm.
Numerical experiments were done to identify the difference in terms of time cost.

	Without initialization	With initialization
fixed-point	$4.033ms$	$3.970ms$
neweon	$4.661ms$	$0.679ms$
broyden	$4.447ms$	$0.799ms$

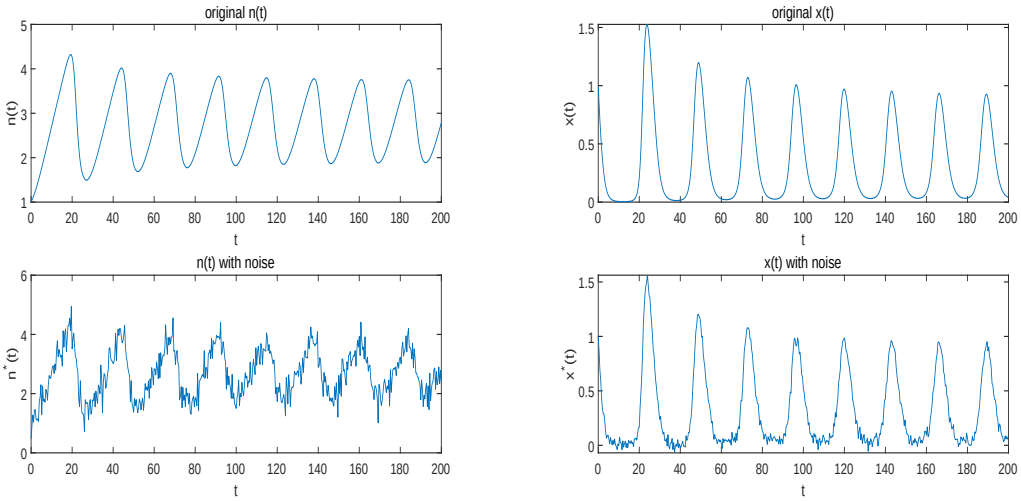
It can be observed from the table that the downhill simplex initialization process does help a lot to reduce time cost for further newton or broyden calculation as it provide a better initial guess.
However, the time cost is almost equivalent for the fixed-point iteration, which indicates that it does not benefit from the downhill simplex initialization.

3 TASK 3

(a) Denote $y_1(t) \triangleq n(t)$, $y_2(t) \triangleq x(t)$ and

$$\Phi \triangleq \begin{bmatrix} \dot{y_1} \\ \dot{y_2} \end{bmatrix} = \begin{bmatrix} \gamma - \alpha y_2^2 y_1 \\ \alpha y_2^2 y_1 - \frac{y_1}{1+y_1} \end{bmatrix} \tag{3.1}$$

Then, the synthetic data for n and x can be calculated iteratively. To make it more accurate, the Runge-Kutta 4-stage method was adopted. The matlab built-in function 'ode45' was used in my implementation.
Plot of $n(t)$ and $x(t)$ were generated as follows



Plot of $n(t)$ and $x(t)$ with noise were also generated here to make a clear comparsion.

(b) At each point (except the two boundary points), the 1st-order derivative of $n(t)$ and $x(t)$ can be approximated using central difference as

$$\dot{n_k} = \frac{n_{k+1} - n_{k-1}}{2\Delta t} \tag{3.2}$$

$$\dot{x_k} = \frac{x_{k+1} - x_{k-1}}{2\Delta t} \tag{3.3}$$

Thus a pair of α and γ can be solved at each point as

$$\alpha_k = \frac{\dot{x_k} + \frac{x_k}{1+x_k}}{n_k^2 x_k} \tag{3.4}$$

$$\gamma_k = \dot{x_k} + \dot{n_k} + \frac{x_k}{1+x_k} \tag{3.5}$$

for $k = 2,3,...,N-1$.
Finally, the estimated α and γ can be calculated as

$$\hat{\alpha} = \frac{\sum_{k=2}^{N-1} \alpha_k}{N-2} \tag{3.6}$$

$$\hat{\gamma} = \frac{\sum_{k=2}^{N-1} \gamma_k}{N-2} \tag{3.7}$$

Numerical experiment shows that $\hat{\alpha} = 0.099824$ and $\hat{\gamma} = 0.199699$, which implies that $\hat{\alpha}$ and $\hat{\gamma}$ agree well with α and γ.

(c) The noise is generated with the built-in function 'normrnd' of matlab in my code, plots are given in part(a).

(d) Considering the mathmatical expect of $Eq(3.2)$ and $Eq(3.3)$ as

$$\begin{aligned} E(\dot{n}) &= \gamma - \alpha E(n^2 x) \\ E(\dot{x}) &= \alpha E(n^2 x) - E(\frac{x}{1+x}) \end{aligned} \tag{3.8}$$

Then α and γ can be resolved as

$$\begin{aligned} \hat{\alpha} &= \frac{E(\dot{x}) + E(\frac{x}{1+x})}{E(n^2 x)} \\ \hat{\gamma} &= E(\dot{x}) + E(\dot{n}) + E(\frac{x}{1+x}) \end{aligned} \tag{3.9}$$

But the signal we have is not clean, what we can obtain from the data are quantities with noise like $E(\dot{n^*})$, $E(\dot{x^*})$, $E(n^{*2}x^*)$, and $E(\frac{x^*}{1+x^*})$. These clean quantities like $E(\dot{n})$, $E(\dot{x})$, $E(n^2 x)$, and $E(\frac{x}{1+x})$ should be approximated from these polluted quantities so that $\hat{\alpha}$ and $\hat{\gamma}$ can be resolved.
As $\epsilon \sim N(0, 0.01)$ and $\nu \sim N(0, 0.001)$, then $E(\epsilon) = 0$, $E(\nu) = 0$ and $E(\epsilon^2) = 0.01$, $E(\nu^2) = 0.001$. Since $n^* = n + \epsilon$ and $x^* = x + \nu$, then

$$\begin{aligned} E(n^*) &= E(n) + E(\epsilon) = E(n) \\ E(x^*) &= E(x) + E(\nu) = E(x) \end{aligned} \tag{3.10}$$

As ϵ and ν are independent, thus

$$\begin{aligned} E(n^{*2}x^*) &= E(n^2 x + 2nx\epsilon + x\epsilon^2 + n^2\nu + 2n\epsilon\nu + \epsilon^2\nu) \\ &= E(n^2 x) + E(x)E(\epsilon^2) \end{aligned} \tag{3.11}$$

and

$$E(\frac{x^*}{1+x^*}) = E(\frac{x}{1+x^*}) \approxeq E(\frac{x}{1+x}) \tag{3.12}$$

For derivatives, similiar relation can be concluded as

$$\begin{aligned} E(\dot{n_k^*}) &= \frac{E(n_{k+1}^*) - E(n_{k-1}^*)}{2\Delta t} = \frac{E(n_{k+1}) - E(n_{k-1})}{2\Delta t} = E(\dot{n_k}) \\ E(\dot{x_k^*}) &= \frac{E(x_{k+1}^*) - E(x_{k-1}^*)}{2\Delta t} = \frac{E(x_{k+1}) - E(x_{k-1})}{2\Delta t} = E(\dot{x_k}) \end{aligned} \tag{3.13}$$

Hence these clean quantities can be approximated as

$$\begin{aligned} E(n) &= E(n^*) \\ E(x) &= E(x^*) \\ E(\dot{n}) &= E(\dot{n^*}) = \frac{\sum_{k=2}^{N-1} \dot{n_k^*}}{N-2} \\ E(\dot{x}) &= E(\dot{x^*}) = \frac{\sum_{k=2}^{N-1} \dot{x_k^*}}{N-2} \\ E(n^2 x) &= E(n^{*2} x^*) - E(x)E(\epsilon^2) \\ E(\frac{x}{1+x}) &= E(\frac{x^*}{1+x^*}) \end{aligned} \tag{3.14}$$

where the properties of the noise is utilized.
In this way, $\hat{\alpha}$ and $\hat{\gamma}$ were resolved in numerical experiment as

$$\begin{aligned} \hat{\alpha} &= 0.102744 \\ \hat{\gamma} &= 0.201082 \end{aligned} \tag{3.15}$$

The result has been validated for about 20 times, although the noise is different at each time, as well as $\hat{\alpha}$ and $\hat{\gamma}$, but each time the result was quite stable.

(e) If the properties of noise are not available, some approximations or assumptions can not be made finely.
At first, a distinct attempt is to treat n^* and x^* like what we have done in part(b).

$$\hat{\alpha} = \frac{\sum_{k=2}^{N-1} \alpha_k^*}{N-2} \tag{3.16}$$

$$\hat{\gamma} = \frac{\sum_{k=2}^{N-1} \gamma_k^*}{N-2} \tag{3.17}$$

In this way, the only difference is the input data sequence, as the input in part(b) is clean but here is polluted.
Numerical experiments were done for 200 times to illustrate the initial guess(Fig3.1) and the variance(Fig3.2) of $\hat{\alpha}$ and $\hat{\gamma}$.

It can be seen from Fig3.2 that the error of $\hat{\gamma}$ is about 3% relative to $\bar{\gamma}$, which is quite nice. But the error of $\hat{\alpha}$ is about 200% relative to $\bar{\alpha}$, which means that the estimation of α fails in this manner. Hence, more sophisticated estimation need to be adopted, although this coarse approach gives relative precise estimation of γ.
Considering the approach in part(e), where $E(n^2 x)$ is estimated utilizing the info of the noise, similiar approximations can be made as

$$E(n^2 x) = E(n^{*2} x^*) = \frac{\sum_{k=1}^{N} n_k^{*2} x_k^*}{N} \tag{3.18}$$

and

$$E(\dot{n}) = E(\dot{n^*}) = \frac{\sum_{k=2}^{N-1} \dot{n_k^*}}{N-2} \tag{3.19}$$

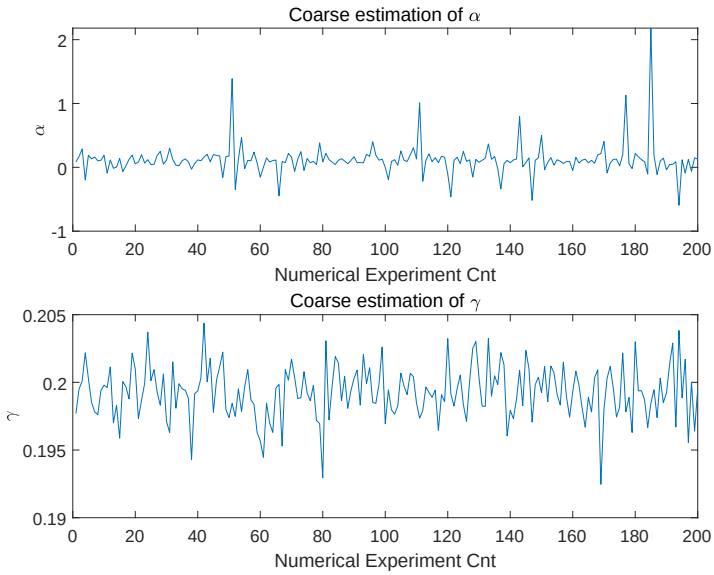


Figure 3.1: Coarse estimation

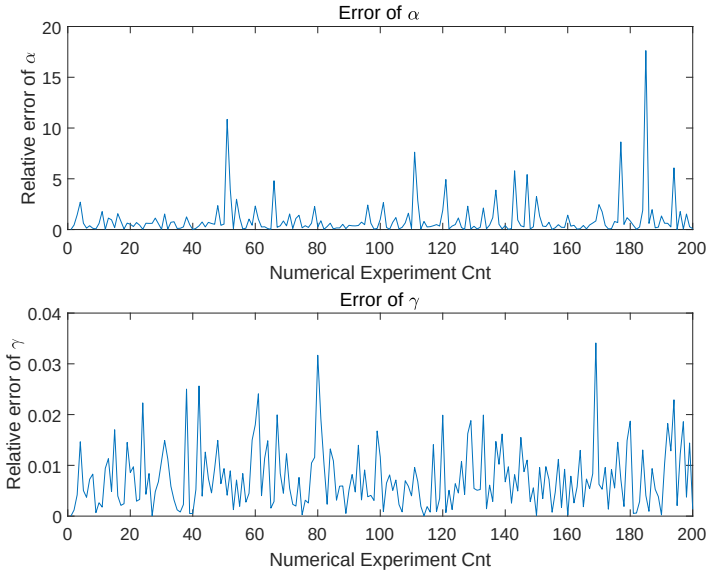


Figure 3.2: Error of estimation

Here, no assumption of the distribution of the noise was made. With $\hat{\gamma}$ given by the coarse estimation, a relative finer estimation of $\hat{\alpha}$ can be calculated as

$$\hat{\alpha} = \frac{\hat{\gamma} - E(\dot{n^*})}{E(n^{*2}x^*)} \tag{3.20}$$

Numerical experiment was carried to verifiy the approximations. This time, the relative error of $\hat{\alpha}$ is satisfying, and the estimation result is

$$\begin{aligned} \hat{\alpha} &= 0.100805 \\ \hat{\gamma} &= 0.198847 \end{aligned} \tag{3.21}$$

Further, we can apply the scheme described in part(e) without tuning $E(n^2x)$ finely, just take it as $E(n^{*2}x^*)$, which also gives similiar result. But these trival stuff will not be discussed here as it has been clear enough for this problem...