**Matthew Feldman (868354) – Numerical methods and computing**

**Standard root function analysis:**

1. First two roots for sin(x) = cos(x^2) are 0.6708 and 1.7474 found in 9 and 7 iterations respectively. This depends where you start the iteration.

|  |
| --- |
| C:\Users\mrfel\Documents\Uni & Life documents\Study\UoM Study - Sem2-2019\Numerical methods & Scientific computing\Assignment2\Q1 - sin(x) + cos(2x^2).jpg |

1. The root is apparently at pi, which is found after 65 iterations. This is misleading as when you look at the graph of the function no root can be seen. This is because as the function approaches pi from the right, it is approaching positive infinity. As it approaches the left it is approaching negative infinity. Therefore as the signs are opposite it converges to what it thinks is the root in the middle; however pi is not actually a root of the function. Here is a graph of the function.

|  |
| --- |
| C:\Users\mrfel\Documents\Uni & Life documents\Study\UoM Study - Sem2-2019\Numerical methods & Scientific computing\Assignment2\Q1 - pi.jpg |

1. A root is not found as the function does not give opposite signs, so the code will not run. However, when you plug in x=0 you will see that the function is equal to 0. This is the only root for the function; however the hybrid algorithm cannot find it due to the sign problem.

|  |
| --- |
|  |

**Q1) c)**

The stopping criteria used for fzero is:

m = 0.5\*(a - b);

tol = 2.0\*eps\*max(abs(b),1.0);

if (abs(m) <= tol) || (fb == 0.0)

break

end

This is an absolute value test of the new midpoint that is derived from the current two interval bounds. Each a and b is updated through each iteration, and the new calculated absolute value of the midpoint is tested to see if it is below the required absolute tolerance.

**Matrix error analysis:**

Table 1: Normwise relative forward error

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Technique | | N | Condition number or Gfpp matrix | | | | |
|  | | | 1 | 10^4 | 10^8 | 10^12 | Gfpp |
| LU Factorization | N = 25 | | 5.4182e-16 | 9.9001e-14 | 2.6529e-10 | 2.4356e-06 | 2.1709e-10 |
| N = 50 | | 1.2091e-15 | 1.1539e-13 | 5.4862e-10 | 6.9134e-06 | 1.4680e-03 |
|  |  | |  |  |  |  |  |
| Inverse | N = 25 | | 5.8209e-16 | 1.7998e-13 | 5.6357e-10 | 2.0535e-06 | 4.2432e-16 |
|  | N = 50 | | 1.1332e-15 | 1.5547e-13 | 8.8005e-10 | 9.7475e-06 | 1.2631e-15 |
|  |  | |  |  |  |  |  |
| QR | N = 25 | | 4.7443e-16 | 1.6024e-13 | 6.6946e-10 | 4.3786e-06 | 1.3731e-15 |
|  | N = 50 | | 7.1015e-16 | 1.6144e-13 | 6.7795e-10 | 1.3232e-05 | 5.2456e-15 |

Table 2: Relative residual eta

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Technique | | N | Condition number or Gfpp matrix | | | | |
|  | | | 1 | 10^4 | 10^8 | 10^12 | Gfpp |
| LU Factorization | N = 25 | | 5.8969e-16 | 6.4348e-17 | 3.2097e-17 | 2.2821e-17 | 2.2788e-11 |
| N = 50 | | 1.2143e-15 | 8.3390e-17 | 1.0335e-16 | 5.7950e-17 | 7.6508e-05 |
|  |  | |  |  |  |  |  |
| Inverse | N = 25 | | 6.0513e-16 | 6.1806e-14 | 1.1440e-10 | 1.4069e-07 | 1.6165e-17 |
|  | N = 50 | | 1.0879e-15 | 2.1490e-14 | 9.9202e-11 | 1.4267e-06 | 5.5268e-17 |
|  |  | |  |  |  |  |  |
| QR | N = 25 | | 4.1862e-16 | 7.8698e-17 | 3.9802e-17 | 3.1800e-17 | 1.9681e-16 |
|  | N = 50 | | 6.6463e-16 | 1.1831e-16 | 8.6844e-17 | 7.7821e-17 | 3.7102e-16 |

Across both n = 25, 50 types of matrices it can be seen that the relative forward error increases as the matrices become progressively more ill-conditioned, across all three techniques. That is the difference between the estimated and true values of our solutions becomes progressively larger as the condition number becomes larger. This is because the condition number acts as an upper bound for the relative forward error. The errors that are being introduced by the solving algorithm (truncation, rounding) are causing larger and larger perturbations in the ill-conditioned matrices (as k gets bigger).

Regarding the relative residual eta it can be seen that LU factorization and QR factorization keep eta relatively constant across the differently conditioned matrices. Because of the link between the relative residual eta, and the backward error (if we assume error is coming from the design matrix A only) then there is a proportional relationship. QR and LU factorization show that low eta infers low backward error; independent of the condition number of the matrices; which is in contrast with the solution error in the first table. Interestingly, solving by the inverse method yields larger backward error as the condition number gets larger. The inverse method residual eta, and it’s backward stability and accuracy are a function of the condition number.

The Gfpp function returns lower triangular matrices populated with -1’s below the diagonal and 1’s in every row of the last column of the matrix. Inverse and QR methods produce low forward and eta error, and hence also can be considered backward stable. LU factorization produces differences between n = 25 and n = 50 for both forward error and eta. The condition number increasing from n =25, and n = 50 may explain partially why the forward error is increasing between the two matrices. This however does not explain why the eta is.

|  |
| --- |
|  |

**Root finder for n functions in n dimensions:**

Output for b was exactly:  
“After 5 iterations, the roots are [1.6180340 1.6180340]”.

Un-comment the relevant part of example2.m function to test if you would like. Here is the graph’s also for part b.

|  |
| --- |
| C:\Users\mrfel\Documents\Uni & Life documents\Study\UoM Study - Sem2-2019\Numerical methods & Scientific computing\Assignment2\Q3b.jpg |

**Q3c)**

For x0 = [1.2, 2.5]:

"After 5 iterations, the roots are [1.336355 , 1.754235]"

|  |
| --- |
|  |

For x0 = [-2, 2.5]

"After 9 iterations, the roots are [-0.901266 , -2.086588]"

|  |
| --- |
|  |

For x0 = [-1.2, -2.5]

"After 5 iterations, the roots are [-0.901266 , -2.086588]"

|  |
| --- |
|  |

For x0 = [2, -2.5]

"After 19 iterations, the roots are [-3.001625 , 0.148108]"

|  |
| --- |
|  |

It goes to (-5,0) then actually goes back the other way to (-3, 0.14) just very hard to see. Have to zoom in on the original matlab plot.

**Comments:**

For 4 different initial starting points, the Newtons method finds 3 unique roots in 2-dimensions (x1,x2) where f1(x1,x2) = 0 and f2(x1,x2) = 0. The following figure depicts the initial starting points in blue and the unique roots in red.

|  |
| --- |
|  |

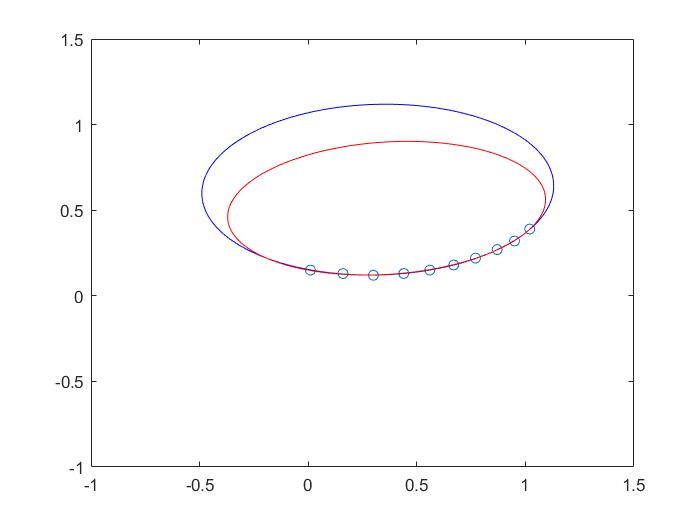
The two blue points to the left of the (0,0) point both converge to [-0.901266 , -2.086588]. The point on the far right converges to the root on the far left. Where the only other point converges to the root closest to it. From this, we can deduce that even though you can start close to a root, it doesn’t necessarily mean this algorithm will converge to that root which is closest.

What can be observed from point 4 is an example of how the algorithm corrects its path to an extent. From an initial starting point of [2, -2.5] it proceeds to go approximately point [-20,22] – very far away from any of the observed roots. The residual norm then spikes, and causes behaviour which forces the (x1,x2) to go in the opposite directions. This is also why the convergence takes longer than the other points (5/9 iterations as compared to 19).

**Linear model to orbit data:**

Parameter estimates for original data = [ -2.6356, 0.1436, 0.5514, 3.2229, -0.4329]

Parameter estimates for perturbed data = [ -3.5140, 0.4862, 0.4733, 3.4204, -0.4340]



Blue is the unperturbed data. Red is perturbed. Points are the original data given in question.

From the singular value decomposition we return a matrix S that has condition number:

K = S(1,1)/S(5,5) = 688.4294

This is a K much larger than 1, so we can consider this an ill-conditioned matrix. For ill-conditioned matrices a small perturbation in the data, or an ‘error’, can cause large fluctuations in the estimates of our solution. If we think of the geometric interpretation of hyperplanes, we can think of an ill-conditioned matrix as hyperplanes that are nearly parallel (singular); where there would be an infinite amount of solutions (e.g. radically different parameters). As our matrix hyperplanes are nearly singular, we can achieve radically different parameter estimates.

As we saw in Q2 the relative forward error increases dramatically as K does. So for a small ‘error’ introduced into our data we are seeing that the parameters that are given (x\_hat) are quite different to what was calculated previously. When you plot multiple orbits with randomly generated small errors, we can witness large deviations in the predicted orbit. When it comes to trusting the analysis, this makes it difficult to do so.