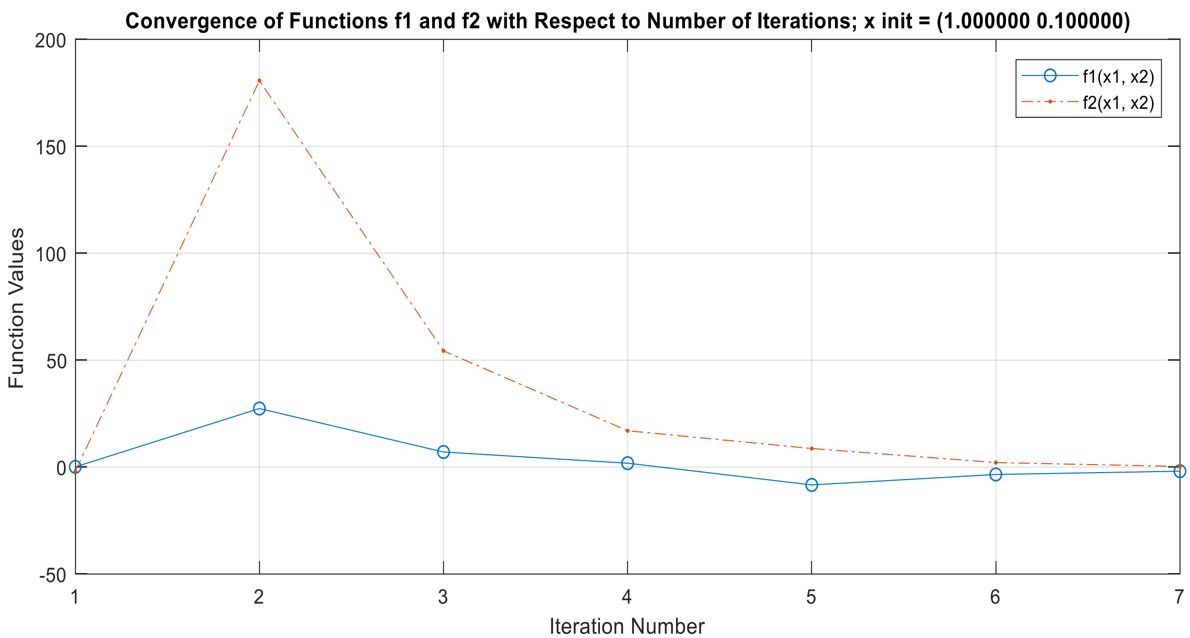
CS3200 – Assignment 4 Written Report
Skylar Shyu

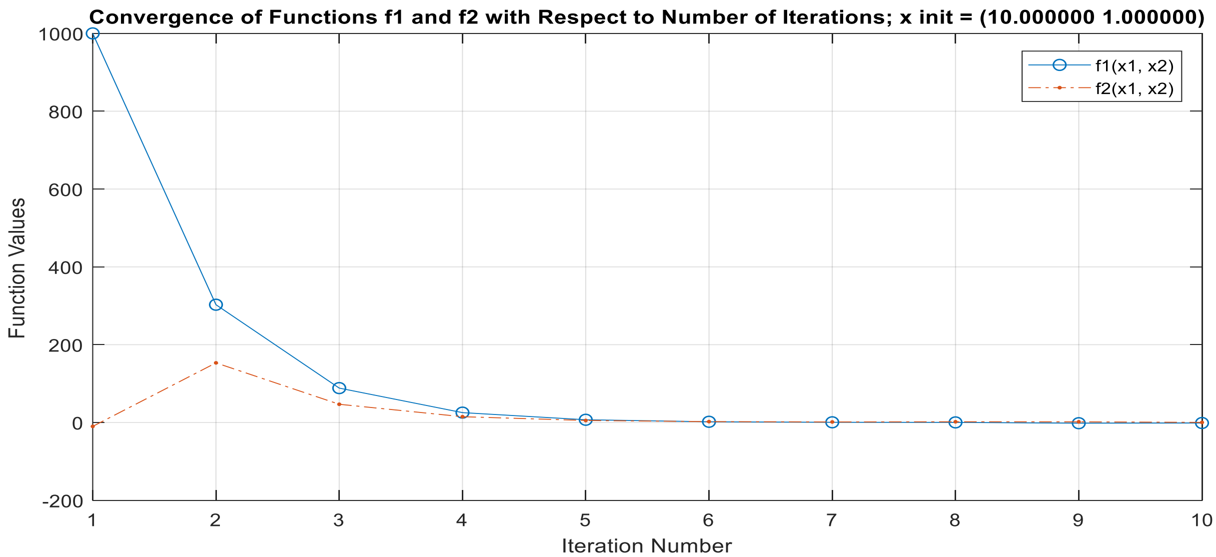
Question 1:

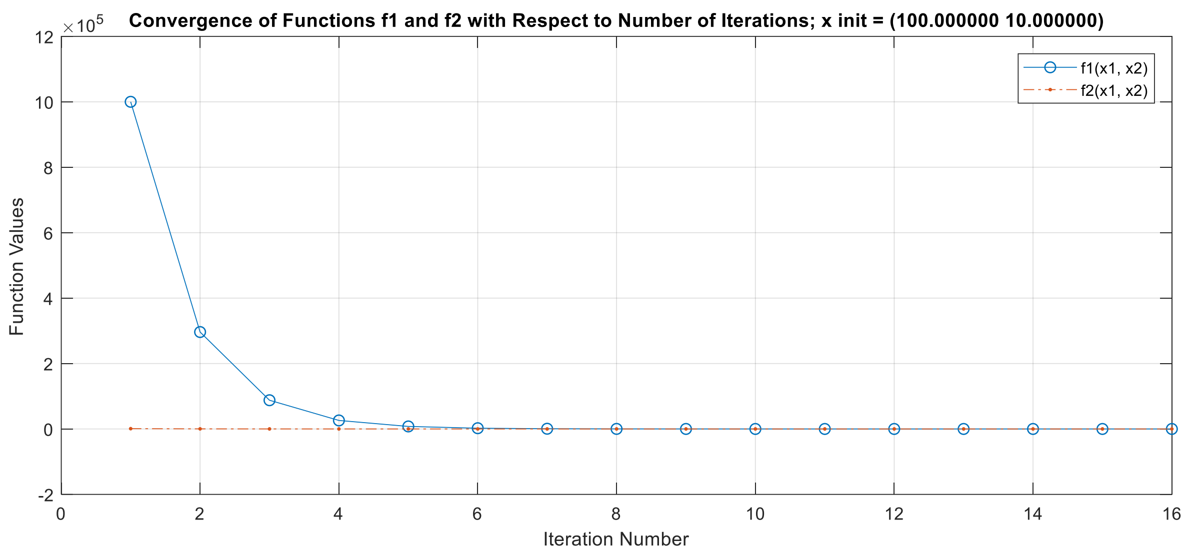
The first question involved a straight forward systems of equations, f_1 and f_2, which were implemented as anonymous function handlers featuring two arguments, along with their necessary derivatives to create a Jacobian matrix. I also assumed a large number of for the number of iterations to begin with: 10^5. (1.0, 0.1) were the starting values for x.

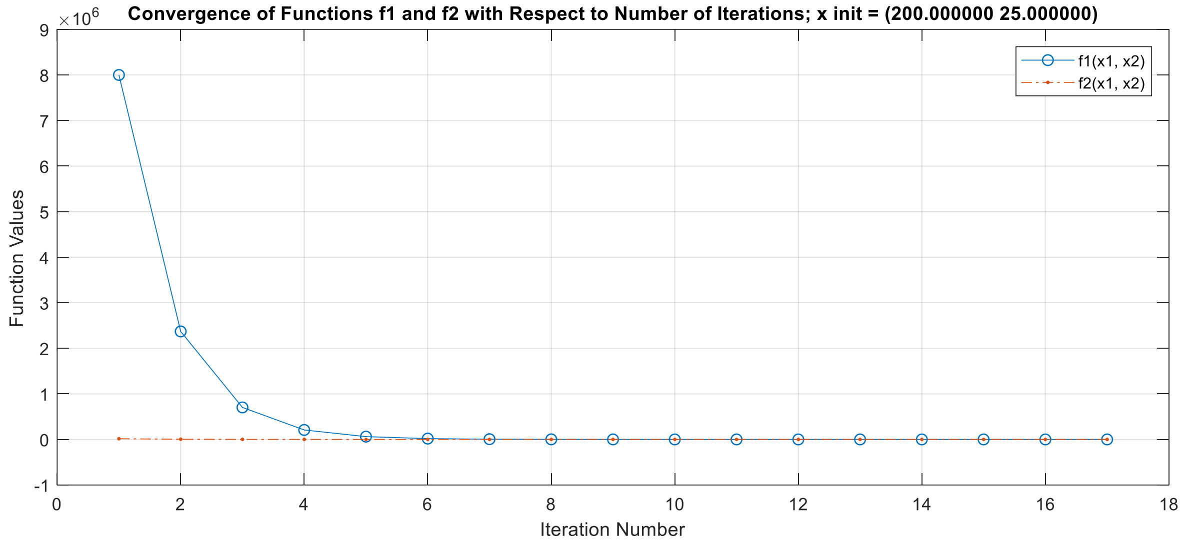
Each iteration began with gathering the highest iterated values of x. Using x, the Jacobian Matrix was calculated, then inverted and multiplied with the values of f(x); this represented the change in x itself from one iteration to the next. The next set of x values would be x less the change in x calculated with the Jacobian. This newly calculated x would be used as an argument in f(x), where x was compared with the allowable convergence error ε. To measure convergence and properly terminate the iteration, I arbitrarily chose $\varepsilon = 10^{-15}$. Thus, the iteration would break and terminate once $||f(x)|| < \varepsilon$, returning the number of iterations it took to achieve convergence. Below are a few plots demonstrating how the number of iterations it took to achieve convergence changed as the initial values of x were manipulated:



CS3200 – Assignment 4 Written Report
Skylar Shyu







As we increased the initial values for x, we can see that it takes more iterations for the systems of equations to reach convergence.

Question 2:

This question also dealt with solving a system of two equations. Using the suggested first approach, x_2 was eliminated from the bottom equation, resulting in one equation: $f(x_1) = e^{(x_1-1)}+x_1^2-4$. The same steps from Question 1 were taken, from setting up $\varepsilon = 10^{-15}$ to creating an anonymous function handler for f(x), iterating through until the $||f(x)|| < \varepsilon$ and terminating the iteration. The solution was determined to be $x_1 = 1.5216$ through this approach.

Trying to repeat the same steps while maintaining 2 separate equations in the system proved to be difficult. Using Newton's method did not yield a valid solution for any of them; in fact, errors were generated during the iteration. The systems of equations was wrapped into a single function called *fsolver.m*, and was passed in as the function handle with the 3 designated initial x-values; all 3 did not yield a solution to the systems of equation by the solver, giving the following result in the terminal for all 3:

```
No solution found.

fsolve stopped because the problem appears regular as measured
by the gradient, but the vector of function values is not near
zero as measured by the default value of the function
tolerance.
```

Question 3:

This question was similar to question 1 in terms of the approached used to solve it. Instead of having only one initial x, 25 iterations must be made for each x initialization beginning from (400,400) to (600,600), inclusively. While the first question only had one for loop to handle the iteration, an outer loop had to be created for this question to iterate through all the initial x-values. Another notable difference is that iteration didn't terminated upon immediately hitting the approved convergence error threshold. 25 iterations were made for each initial starting value of x. This resulted in a large (2 x 25 x 201)—2 representing x_1 and x_2, 25 for the number of iterations per initial starting value, and 201 for account all the initial starting values between 400 and 600. Otherwise, the methodology remained the same.

A table was created in the end, to display the values where the system converged toward with respect to the initial starting values.

By setting $\varepsilon = 10^{-3}$, the following result was obtained. Please see below the table to understand how to interpret it:

(x1,x2)	$f_1(x)$	$f_2(x)$	Conv.
400	0.000246521398035806	-0.00541032309189982	1
401	0.000254210870953482	-0.00532933447459416	1
402	0.000262033117868478	-0.00523785582044556	1
403	0.000269988033078334	-0.00513540082624098	1
404	0.000278075470284402	-0.00502146968570538	1
405	0.000286295243087614	-0.00489554909398227	1
406	0.000294647125502740	-0.00475711226888120	1
407	0.000303130852490117	-0.00460561898888634	1
408	0.000311746120504440	-0.00444051564780545	1
409	0.000320492588059722	-0.00426123532591693	1
410	0.000329369876309829	-0.00406719787741361	1
411	0.000338377569643925	-0.00385781003392172	1
412	0.000347515216296158	-0.00363246552378138	1
413	0.000356782328969019	-0.00339054520676818	1
414	0.000366178385469633	-0.00313141722391419	1
415	0.000375702829358522	-0.00285443716198608	1
416	0.000385355070610098	-0.00255894823214664	1
417	0.000395134486284363	-0.00224428146239619	1
418	0.000405040421209202	-0.00190975590320830	1
419	0.000415072188672691	-0.00155467884584182	1
420	0.000425229071124768	-0.00117834605274969	1
421	0.000435510320887801	-0.000780041999483228	1
422	0.000445915160875333	-0.000359040127465793	1
423	0.000456442785318557	8.53968929912519e-05	1
424	0.000467092360499769	0.000554016890137898	1
425	0.000477863025492404	0.00104757790929821	0
426	0.000488753892906934	0.00156684792322581	0
427	0.000499764049642079	0.00211260453494266	0
428	0.000510892557640774	0.00268563467380645	0

429	0.000522138454650283	0.00328673428547233	0
430	0.000533500754985882	0.00391670801651678	0
431	0.000544978450297471	0.00457636889443269	0
432	0.000556570510338682	0.00526653800367027	0
433	0.000568275883737675	0.00598804415847409	0
434	0.000580093498769258	0.00674172357322145	0
435	0.000592022264127582	0.00752841953092442	0
436	0.000604061069698946	0.00834898205063306	0
437	0.000616208787334070	0.00920426755436266	0
438	0.000628464271619287	0.0100951385342338	0
439	0.000640826360646082	0.0110224632204790	0
440	0.000653293876778441	0.0119871152509132	0
441	0.000665865627417394	0.0129899733424912	0
442	0.000678540405762263	0.0140319209655724	0
443	0.000691316991568067	0.0151138460214117	0
444	0.000704194151898509	0.0162366405234420	0
445	0.000717170641874041	0.0174012002828925	0
446	0.000730245205414542	0.0186084245992266	0
447	0.000743416575975997	0.0198592159558622	0
448	0.000756683477280845	0.0211544797216499	0
449	0.000770044624041316	0.0224951238585185	0
450	0.000783498722675481	0.0238820586357047	0
451	0.000797044472015445	0.0253161963509014	0
452	0.000810680564007277	0.0267984510587460	0
453	0.000824405684402277	0.0283297383068488	0
454	0.000838218513439114	0.0299109748798132	0
455	0.000852117726516498	0.0315430785513702	0
456	0.000866101994856031	0.0332269678449375	0
457	0.000880169986154718	0.0349635618028645	0
458	0.000894320365227052	0.0367537797644051	0
459	0.000908551794636115	0.0385985411527929	0
460	0.000922862935313518	0.0404987652713507	0
461	0.000937252447167857	0.0424553711089044	0
462	0.000951718989681373	0.0444692771545279	0
463	0.000966261222494661	0.0465414012216814	0
464	0.000980877805979070	0.0486726602818390	0
465	0.000995567401796715	0.0508639703075866	0
466	0.00101032867344773	0.0531162461252190	0
467	0.00102516028680477	0.0554304012768463	0

468	0.00104006091063449	0.0578073478918919	0
469	0.00105502921710582	0.0602479965680716	0
470	0.00107006388228509	0.0627532562616571	0
471	0.00108516358661768	0.0653240341869916	0
472	0.00110032701539621	0.0679612357252026	0
473	0.00111555285921527	0.0706657643419004	0
474	0.00113083981441245	0.0734385215138336	0
475	0.00114618658349588	0.0762804066643099	0
476	0.00116159187555799	0.0791923171072249	0
477	0.00117705440667565	0.0821751479995734	0
478	0.00119257290029669	0.0852297923022318	0
479	0.00120814608761279	0.0883571407488701	0
480	0.00122377270791879	0.0915580818227666	0
481	0.00123945150895840	0.0948335017413360	0
482	0.00125518124725653	0.0981842844481868	0
483	0.00127096068843830	0.101611311612507	0
484	0.00128678860753451	0.105115462635463	0
485	0.00130266378927424	0.108697614663565	0
486	0.00131858502836415	0.112358642608608	0
487	0.00133455112975498	0.116099419174062	0
488	0.00135056090889516	0.119920814887643	0
489	0.00136661319197184	0.123823698139847	0
490	0.00138270681613941	0.127808935228217	0
491	0.00139884062973566	0.131877390407078	0
492	0.00141501349248586	0.136029925942570	0
493	0.00143122427569481	0.140267402172655	0
494	0.00144747186242719	0.144590677571997	0
495	0.00146375514767624	0.149000608821328	0
496	0.00148007303852119	0.153498050881208	0
497	0.00149642445427344	0.158083857069860	0
498	0.00151280832661182	0.162758879144933	0
499	0.00152922359970716	0.167523967388878	0
500	0.00154566923033630	0.172379970697834	0
501	0.00156214418798594	0.177327736673711	0
502	0.00157864745494630	0.182368111719308	0
503	0.00159517802639509	0.187501941136257	0
504	0.00161173491047189	0.192730069225602	0
505	0.00162831712834319	0.198053339390760	0
506	0.00164492371425835	0.203472594242776	0

507	0.00166155371559670	0.208988675707543	0
508	0.00167820619290607	0.214602425134972	0
509	0.00169488021993295	0.220314683409767	0
510	0.00171157488364452	0.226126291063789	0
511	0.00172828928424284	0.232038088389686	0
512	0.00174502253517136	0.238050915555789	0
513	0.00176177376311413	0.244165612721978	0
514	0.00177854210798776	0.250383020156462	0
515	0.00179532672292658	0.256703978353286	0
516	0.00181212677426104	0.263129328150420	0
517	0.00182894144148972	0.269659910848331	0
518	0.00184576991724517	0.276296568328874	0
519	0.00186261140725371	0.283040143174404	0
520	0.00187946513028956	0.289891478786952	0
521	0.00189633031812341	0.296851419507431	0
522	0.00191320621546575	0.303920810734666	0
523	0.00193009207990504	0.311100499044211	0
524	0.00194698718184114	0.318391332306851	0
525	0.00196389080441401	0.325794159806654	0
526	0.00198080224342800	0.333309832358553	0
527	0.00199772080727197	0.340939202425312	0
528	0.00201464581683527	0.348683124233815	0
529	0.00203157660542003	0.356542453890664	0
530	0.00204851251864969	0.364518049496889	0
531	0.00206545291437418	0.372610771261854	0
532	0.00208239716257181	0.380821481616201	0
533	0.00209934464524809	0.389151045323785	0
534	0.00211629475633162	0.397600329592577	0
535	0.00213324690156732	0.406170204184516	0
536	0.00215020049840704	0.414861541524118	0
537	0.00216715497589786	0.423675216806058	0
538	0.00218410977456807	0.432612108101375	0
539	0.00220106434631111	0.441673096462562	0
540	0.00221801815426767	0.450859066027288	0
541	0.00223497067270587	0.460170904120853	0
542	0.00225192138689994	0.469609501357297	0
543	0.00226886979300731	0.479175751739145	0
544	0.00228581539794445	0.488870552755811	0
545	0.00230275771926136	0.498694805480557	0

546	0.00231969628501510	0.508649414666106	0
547	0.00233663063364223	0.518735288838795	0
548	0.00235356031383041	0.528953340391310	0
549	0.00237048488438936	0.539304485674023	0
550	0.00238740391412106	0.549789645084852	0
551	0.00240431698168945	0.560409743157672	0
552	0.00242122367548980	0.571165708649341	0
553	0.00243812359351764	0.582058474625225	0
554	0.00245501634323750	0.593088978543338	0
555	0.00247190154145156	0.604258162336992	0
556	0.00248877881416816	0.615566972496065	0
557	0.00250564779647045	0.627016360146840	0
558	0.00252250813238503	0.638607281130376	0
559	0.00253935947475090	0.650340696079607	0
560	0.00255620148508860	0.662217570494879	0
561	0.00257303383346973	0.674238874818230	0
562	0.00258985619838684	0.686405584506262	0
563	0.00260666826662380	0.698718680101672	0
564	0.00262346973312669	0.711179147303403	0
565	0.00264026030087540	0.723787977035528	0
566	0.00265703968075559	0.736546165514796	0
567	0.00267380759143168	0.749454714316888	0
568	0.00269056375922034	0.762514630441406	0
569	0.00270730791796491	0.775726926375611	0
570	0.00272403980891059	0.789092620156921	0
571	0.00274075918058056	0.802612735434201	0
572	0.00275746578865297	0.816288301527815	0
573	0.00277415939583895	0.830120353488589	0
574	0.00279083977176156	0.844109932155516	0
575	0.00280750669283577	0.858258084212390	0
576	0.00282415994214956	0.872565862243312	0
577	0.00284079930934601	0.887034324787081	0
578	0.00285742459050660	0.901664536390529	0
579	0.00287403558803557	0.916457567660801	0
580	0.00289063211054550	0.931414495316618	0
581	0.00290721397274400	0.946536402238492	0
582	0.00292378099532169	0.961824377517971	0
583	0.00294033300484135	0.977279516505969	0
584	0.00295686983362826	0.992902920860034	0

585	0.00297339131966189	1.00869569859081	0
586	0.00298989730646880	1.02465896410754	0
587	0.00300638764301675	1.04079383826268	0
588	0.00302286218361022	1.05710144839564	0
589	0.00303932078778714	1.07358292837576	0
590	0.00305576332021691	1.09023941864441	0
591	0.00307218965059979	1.10707206625622	0
592	0.00308859965356751	1.12408202491972	0
593	0.00310499320858532	1.14127045503703	0
594	0.00312137019985522	1.15863852374290	0
595	0.00313773051622061	1.17618740494304	0
596	0.00315407405107219	1.19391827935172	0
597	0.00317040070225526	1.21183233452868	0
598	0.00318671037197822	1.22993076491546	0
599	0.00320300296672252	1.24821477187092	0
600	0.00321927839715388	1.26668556370636	0

(x_1, x_2) – because $x_1 = x_2$ in this problem, the lone number in this column represents both integers, where (x_1, x_2) is the **initial value of x**.
f_1(x) and f_2(x) are placed side-by-side to see the value they are converging toward
Conv? column is a Boolean, where 1 = true (the system does converge) and 0 = false (the system does not converge)

Thus, viewing the values in the table where Conv? is 1 (true), it is plausible to assume that they are converging toward the value of 0. There are 25 different solutions where the system converges.

<u>Sources of Information:</u>

Sources of information include the slides posted on Canvas, as well as MatLab's documentation on routines such as zero() and fsolve(). Also, I did browse the Discussion forum on Canvas for tips and help, as well as going to TA hours to obtain assistance

I did not collaborate with any classmates on this assignment.

As far as source codes went, I implemented all the functions that didn't already come with MatLab; in this case, I implemented the function *fsolver.m*.