Math 128a - Program 3 - ANSWERS

Here is the output of the test driver tstGE.m, which we will use to answer the questions in the assignment.

Example #	1	2	3	4	5
Case	1	1	1	1	1
Dimension	5	10	25	50	1e+02
Err_NP	9.5e-16	4.9e-16	4.2e-16	1.5e-15	2e-15
Err_PP	1.9e-16	1.2e-16	2.8e-16	2.3e-16	3.3e-16
Err_CP	7.7e-17	9.7e-17	2.8e-16	1.8e-16	3e-16
PG_NP	53	1.3e+02	1.5e+02	2.3e+03	2.4e+03
PG_PP	1.3	1.3	1.5	1.6	1.8
PG_CP	1.3	1.3	1.4	1.5	1.5
Ebnd	2.2e-16	2.2e-16	2.2e-16	2.2e-16	2.2e-16
R_NP	4.3	2.2	1.9	6.9	8.8
R_PP	0.83	0.52	1.3	1	1.5
R_CP	0.35	0.44	1.3	0.83	1.4
R_PG_NP	0.081	0.017	0.013	0.003	0.0037
R_PG_PP	0.65	0.41	0.88	0.66	0.82
R_PG_CP	0.27	0.34	0.88	0.57	0.94

Example #	6	7	8	9	10
Case	2	2	2	2	2
Dimension	5	10	25	50	1e+02
Err_NP	3.6e-16	1.9e-15	3.4e-14	9.9e-13	7.4e-13
Err_PP	1.2e-15	1.1e-15	1.5e-15	7.1e-15	1.2e-14
Err_CP	1.4e-16	5.4e-16	1.2e-15	4.5e-15	1.1e-14
PG_NP	22	9.9	5.8e+03	1.1e+06	8.5e+05
PG_PP	1.3	1.8	6.4	9.3	9.7
PG_CP	1.6	1.8	2.9	4.4	5.3
Ebnd	4.8e-15	7e-15	6.1e-15	2.5e-14	7.8e-14
R_NP	0.075	0.27	5.5	40	9.5
R_PP	0.25	0.15	0.24	0.29	0.16
R_CP	0.029	0.077	0.2	0.18	0.15
R_PG_NP	0.0034	0.027	0.00095	3.6e-05	1.1e-05
R_PG_PP	0.19	0.084	0.038	0.031	0.016
R_PG_CP	0.019	0.043	0.069	0.041	0.027

Example #	11	12	13	14	15
Case	3	3	3	3	3
Dimension	5	10	25	50	1e+02
Err_NP	6.8e-13	5.4e-05	15	1.4e+02	11
Err_PP	2e-12	0.00013	3.5e+02	64	79
Err_CP	5.4e-12	5.8e-06	49	2.1e+02	21
PG_NP	2	13	2.7e+03	7.3e+04	9.7e+05
PG_PP	1	1	1	1	1
PG_CP	1	1	1	1	1
Ebnd	1.1e-10	0.0036	7.8e+02	4.6e+03	3.8e+04
R_NP	0.0064	0.015	0.019	0.03	0.0003
R_PP	0.019	0.036	0.45	0.014	0.0021
R_CP	0.051	0.0016	0.062	0.046	0.00054
R_PG_NP	0.0032	0.0012	6.9e-06	4.2e-07	3.1e-10
R_PG_PP	0.019	0.036	0.45	0.014	0.0021
R_PG_CP	0.051	0.0016	0.062	0.046	0.00054

Example #	16	17	18	19	20
Case	4	4	4	4	4
Dimension	5	10	25	50	1e+02
Err_NP	6e-17	7e-16	6.1e-11	0.015	0.67
Err_PP	6e-17	7e-16	6.1e-11	0.015	0.67
Err_CP	1.4e-16	2e-16	3.6e-16	1.9e-16	1.7e-16
PG_NP	16	5.1e+02	1.7e+07	5.6e+14	6.3e+29
PG_PP	16	5.1e+02	1.7e+07	5.6e+14	6.3e+29
PG_CP	2	2	2	2	2
Ebnd	4.9e-16	9.7e-16	2.5e-15	5e-15	9.9e-15
R_NP	0.12	0.72	2.5e+04	3.1e+12	6.7e+13
R_PP	0.12	0.72	2.5e+04	3.1e+12	6.7e+13
R_CP	0.28	0.21	0.15	0.039	0.017
R_PG_NP	0.0076	0.0014	0.0015	0.0055	1.1e-16
R_PG_PP	0.0076	0.0014	0.0015	0.0055	1.1e-16
R_PG_CP	0.14	0.1	0.073	0.02	0.0085

Example #	21	22	23	24	25
Case	5	5	5	5	5
Dimension	5	10	25	50	1e+02
Err_NP	0.012	0.19	0.48	2.2e+02	31
Err_PP	3e-16	1.8e-15	8.3e-16	1.1e-14	1e-14
Err_CP	1.3e-16	2.3e-15	3.3e-15	3.1e-15	2.4e-14
PG_NP	6.6e+28	2.2e+29	4.9e+30	7.8e+32	6.2e+31
PG_PP	1	1.5	2.1	2.4	4.4
PG_CP	1	1.3	1.2	1.5	1.9
Ebnd	1.6e-15	4.4e-14	1.2e-14	6.4e-14	1.4e-13
R_NP	7.7e+12	4.2e+12	4.2e+13	3.3e+15	2.2e+14
R_PP	0.19	0.04	0.072	0.17	0.072
R_CP	0.081	0.051	0.28	0.049	0.17
R_PG_NP	1.2e-16	1.9e-17	8.5e-18	4.3e-18	3.5e-18
R_PG_PP	0.18	0.027	0.035	0.069	0.017
R_PG_CP	0.077	0.041	0.23	0.033	0.087

1. Which examples (numbered 1 through 25) had error bounds Ebnd bigger than .001, which means Ebnd predicts 3 or fewer correct digits of accuracy in the computed solutions? These matrices are called "ill-conditioned", and it is inherently difficult to solve with them accurately.

   ANSWER: Only case=3 matrices of dimension 10 or greater (examples 12 – 15) had Ebnd exceeding .001. In fact Ebnd was larger than 1 for examples 13 – 15 (dimension $\geq$ 25), which means that no correct digits in the solution can be expected. Case=3 matrices are called Hilbert matrices, and are given by the simple formula $H_{ij} = 1/(i + j - 1)$; they are notoriously ill-conditioned, with Ebnd growing exponentially with dimension. Note that since Ebnd measures relative error, once it is 1 or greater, its exact magnitude is irrelevant, it just means that the answer can be completely wrong.

2. Were there any examples where Ebnd was much smaller than Err_NP, i.e. where GENP was much less accurate than predicted by Ebnd? Were any of these "easy" examples, in the sense that Ebnd was quite small, and either GEPP or GECP got them right? On which examples was there a lot of pivot growth for GENP? Is there a correspondence between these examples? How did Err_NP compare to Ebnd*PG_NP on these examples? Is GENP a generally reliable algorithm? Is Ebnd a generally reliable error bound for GENP (i.e. never much smaller than the true error, and rarely much larger? Is Ebnd*PG_NP a generally reliable error bound, in the same sense?

   ANSWER: We look for examples where R_NP = Err_NP/Ebnd is much larger than 1. This occurs for case=4 and dimension $\geq$ 25 (examples 18 – 20) and for case = 5 (examples 21 – 25) Case = 4 is specifically designed to cause large pivot growth and so inaccurate answers when n is large for both GENP and GEPP, even though these problems have small Ebnd (and so are "easy") and GECP solves them accurately. Case = 5 is designed to cause large pivot growth for GENP for all dimensions, even though they have small Ebnd and both GEPP and GECP solve them accurately. GENP had very large pivot growth for these examples (PG_NP $\geq 10^{14}$ and usually $\geq 10^{27}$). GENP had more modest pivot growth (up to 10^5) for other examples as well, which did indeed hurt the accuracy Err_NP, but not nearly as much as these examples. The alternative error bound Ebnd*PG_NP (measured by whether R_PG_NP = Err_NP/(Ebnd*PG_NP) is near or far from 1, the nearer the better) is indeed always less than 1, which means that Ebnd*PG_NP does always bound the error Err_NP, but it can be much less than 1, so that it is often a pessimistic error bound. In summary, GENP is not a generally reliable algorithm, because it can give inaccurate answers on common "easy" problems like case 5 (where we claim without proof that case 5 is "common"). Ebnd is correspondingly not a reliable error bound for GENP either (because it is too small on the easy case = 5). Ebnd*PG_NP is better, because it is never too small, but may be too large.

3. Were there any examples where Ebnd was much smaller than Err_PP, i.e. where GEPP was much less accurate than predicted by Ebnd? Were any of these "easy" examples, in the sense that Ebnd was quite small, and GECP got them right? On which examples was there a lot of pivot growth for GEPP? Is there a correspondence between these examples? How did Err_PP compare to Ebnd*PG_PP on these examples? Is GEPP a generally reliable algorithm? (Note: case=4 is essentially the only known example where GEPP has large pivot growth). Is Ebnd a generally reliable error bound for GEPP (i.e. never much smaller than the true error, and rarely much larger? Is Ebnd*PG_NP a generally reliable error bound, in the same sense?

   ANSWER: We look for examples where R_PP = Err_PP/Ebnd is much larger than 1. This occurs only for case = 4 and dimension at least 50, although dimension 25 is already starting to lose accuracy. Otherwise, GEPP works very well. It turns that that the matrix in case = 4 is one of the very few examples ever found where GEPP has large pivot growth (the same as GENP, 2^{n-1}) and so gets an inaccurate answer, even when the problem is easy (Ebnd is small). On other problems, either R_PP is close to 1, or the problem is so hard (case=3) that both Err_PP and Ebnd are much larger than 1, and their exact ratio is irrelevant. Since case = 4 is considered rare, GEPP is considered a reliable algorithm, and indeed is the standard version of Gaussian elimination in use. Both Ebnd and Ebnd*PG_PP are use in practice as error bounds (Ebnd*PG_PP is slightly more reliable in the sense of dealing with case = 4).

4. Were there any examples where Ebnd was much smaller than Err_CP, i.e. where GECP was much less accurate than predicted by Ebnd? On which examples was there a lot of pivot growth for GECP? Is GECP a generally reliable algorithm? Is Ebnd a generally reliable error bound for GECP (i.e. never much smaller than the true error, and rarely much larger? Is Ebnd*PG_CP a generally reliable error bound, in the same sense?

   ANSWER: GECP is the most reliable algorithm of the three studied here, handling case = 4 without problem. R_CP and R_PG_CP were always reasonably close to 1, except for the hard problems (case = 3 and dimension at least 25) where both Ebnd and Err_CP were much larger than 1, and their exact ratio did not matter.