BEPP 931 - Solution to Problem Set 2

Shasha Wang, Cung Truong Hoang, Jose Sidaoui

March 13, 2020

Question 1

We reproduce Table 3.2 by employing the Gauss-Jacobi and the Gauss-Seidel iteration process to the demand and supply problem. The initial guess is $p_0 = 4$ and $q_0 = 1$.

The Gauss-Jacobi iteration process is governed by the following equations:

$$q_{n+1} = 1 + 0.5p_n$$
$$p_{n+1} = 10 - q_n$$

The Gauss-Seidel iteration process updates p using the most current guess for q. It is governed by the following equations:

$$q_{n+1} = 1 + 0.5p_n$$
$$p_{n+1} = 10 - q_{n+1}$$

We then obtain the following results as in Table 3.2:

Table 1. Gaussian Methods

Iteration	Gauss-Jacobi		Gauss-Seidel	
n	p_n	q_n	p_n	q_n
0	4	1	4	1
1	9	3	7	3
2	7	5.5	5.5	4.5
3	4.5	4.5	6.25	3.75
4	5.5	3.25	5.875	4.125
5	6.75	3.75	6.0625	3.9375
7	5.625	4.125	6.0156	3.9844
10	6.0625	4.0938	5.9980	4.0020
15	5.9766	4.0078	6.0001	3.9999
20	5.9980	3.9971	6.0000	4.0000

Question 2

We reproduce Figure 3.4 by using Gauss-Seidel and the SOR method with dampening. The Gauss-Seidel iteration process is governed by the equations:

$$
\begin{aligned}
p_{n+1} &= 21 - 3q_n \\
q_{n+1} &= 0.5p_{n+1} - 3
\end{aligned}
$$

The SOR process is implemented with a dampening factor of $\omega = 0.75$:

$$
\begin{aligned}
p_{n+1} &= 0.75(21 - 3q_n) + 0.25p_n \\
q_{n+1} &= 0.75(0.5p_{n+1} - 3) + 0.25q_n
\end{aligned}
$$

The table below depicts the values for p and q at each iteration and shows how dampening can lead to convergence to the true solution whereas the Gauss-Seidel process diverges.

Table 2. Figure 3.4: Dampening an Unstable Hog Cycle

Iteration	Gauss-Seidel		SOR ($\omega = 0.75$)	
n	p_n	q_n	p_n	q_n
0	10	2	10	2
1	15	4.5	13.75	3.4062
2	7.5	.75	11.523	2.9229
3	18.75	6.375	12.054	3.0011
4	1.875	-2.0625	12.011	3.0044
5	27.188	10.594	11.993	2.9984
6	-10.781	-8.3906	12.002	3.0003
7	46.172	20.086	12	3
8	-39.258	-22.629	12	3
9	88.887	41.443	12	3
10	-103.33	-54.665	12	3
15	887.79	440.89	12	3
20	-6638.5	-3322.3	12	3

Similarly as before, we reproduce Figure 3.5 by using Gauss-Seidel and the SOR method but with acceleration. The Gauss-Seidel process is given by:

$$
\begin{aligned}
p_1^{n+1} &= 1 + 0.75p_2^n \\
p_2^{n+1} &= 2 + 0.80p_1^{n+1}
\end{aligned}
$$

For the SOR process we set $\omega = 1.5$:

$$p_1^{n+1} = 1.5(1 + 0.75p_2^n) - 0.5p_1^n$$
$$p_2^{n+1} = 1.5(2 + 0.80p_1^{n+1}) - 0.5p_2^n$$

The table below shows the solutions at each iteration. The acceleration approach yields faster convergence.

Table 3. Figure 3.5: Accelerating a Nash Equilibrium Computation

Iteration	Gauss-Seidel		SOR ($\omega = 1.5$)	
n	p_n	q_n	p_n	q_n
0	.1	.1	.1	.1
1	1.075	2.86	1.5625	4.825
2	3.145	4.516	6.1469	7.9638
3	4.387	5.5096	7.3858	7.8811
4	5.1322	6.1058	6.6733	7.0674
5	5.5793	6.4635	6.1142	6.8033
6	5.8476	6.6781	6.0966	6.9143
7	6.0086	6.8068	6.2303	7.0192
8	6.1051	6.8841	6.2814	7.0281
9	6.1631	6.9305	6.2659	7.0051
10	6.1978	6.9583	6.2477	6.9947
15	6.2459	6.9968	6.25	6.9999
20	6.2497	6.9997	6.25	7

Question 3

We replicate Table 4.1 by applying Newton's method to 4.3.5. We compare the solution path to the true solution of $x^* = [y^*, z^*] = [-0.5625, 1.0769]$ which we obtain with the fsolve command in Matlab (obtain the FOCs and find numerically the solution).

To employ Newton's method, we use Maple/Mathematica to obtain the analytical partial derivatives, from which we construct the gradient and the Hessian matrix. Then we compute the step k by solving:

$$s^k = -H\left(x^k\right)^{-1} \nabla f\left(x^k\right)'$$

.

We set the initial guess to $x = [1.5, 2]$ and update our guess in the following way:

$$x^{k+1} = x^k + s^k = x^k - H\left(x^k\right)^{-1} \nabla f\left(x^k\right)'$$

The following table displays the results for the first 10 iterations. Our results depart from the book in column 2 and for the last 3 iterations.

Table 4. Newton's Methods

Iteration	Error		Function Value Diff	Step-size		Normed Gradient	
k	$x^k - x^*$		$f\left(x^k\right) - f\left(x^*\right)$	$x^k - x^{k-1}$		$\nabla f\left(x^k\right) / \left(1 + \left\lvert f\left(x^k\right)\right\rvert\right)$	
0	2.0625	0.92306	-0.54281	0	0	-0.43139	-0.6168
1	1.4549	0.36589	-0.1102	-0.60757	-0.55717	-0.12842	-0.17596
2	0.97732	-0.0029202	-0.015821	-0.47759	-0.36881	-0.034273	-0.03977
3	0.61337	-0.11562	-0.0025609	-0.36396	-0.1127	-0.0087454	-0.004529
4	0.27872	-0.040975	-0.00041457	-0.33465	0.074643	-0.0027644	-0.0021743
5	0.075258	-0.0092594	-2.6531e-05	-0.20346	0.031716	-0.00061047	-0.00064996
6	0.0068103	-0.00077064	-2.1326e-07	-0.068448	0.0084888	-5.2588e-05	-6.4114e-05
7	-6.9201e-09	1.8281e-09	2.8866e-15	-0.0068103	0.00077065	-4.8196e-07	-6.0453e-07
8	-6.3099e-05	7.0091e-06	1.7973e-11	-6.3092e-05	7.0073e-06	-4.1159e-11	-5.1492e-11
9	-6.3105e-05	7.0097e-06	1.7973e-11	-5.3928e-09	5.9999e-10	1.0106e-17	4.0425e-16