Economics 714 - Computational Methods in Economics
University of Pennsylvania
Fall 2021

Problem Set 1

Javier Tasso

Question 1

See this link.

Question 2

Figure (1) plots the area we want to calculate and table (1) shows the results using different methods. The code *econ714_homework1_question2.m* produces these outputs.

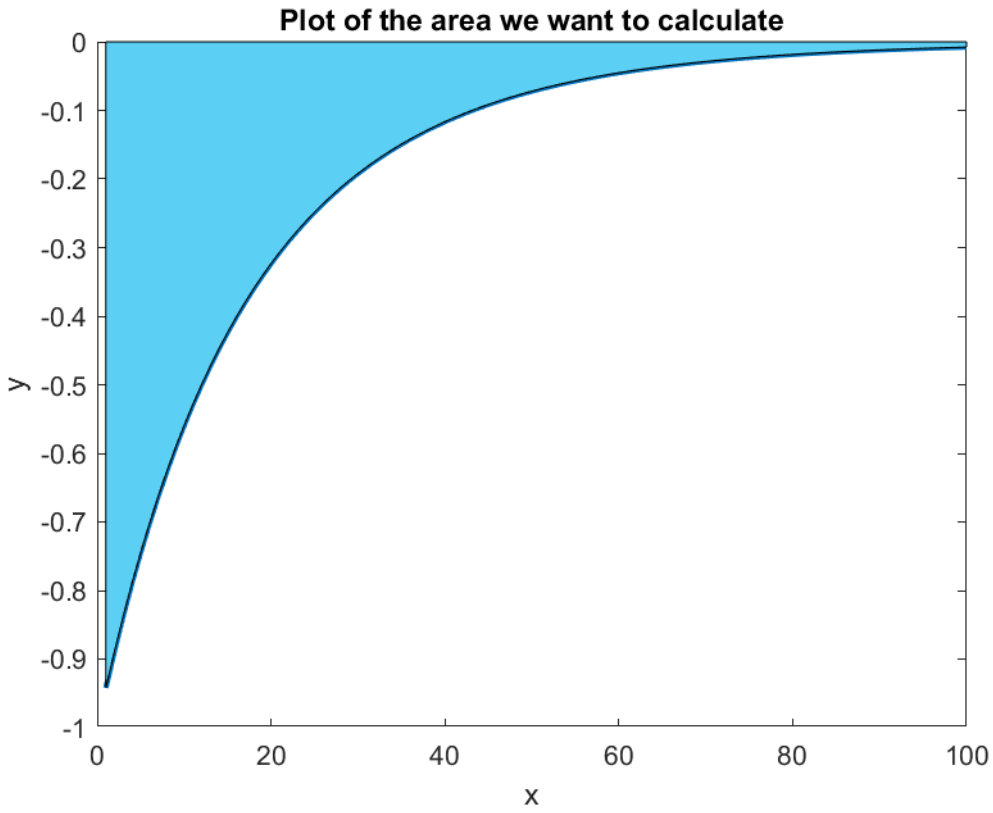

Figure 1: Area - Question 2

Table 1: Comparison of Different Methods - Integration

Method	N100	N1000	N10000	N100000
Midpoint	-18.207	-18.2095	-18.2095	-18.2095
Time (in seconds)	0.0018	0.0007	0.0018	0.0123
Trapezoid	-18.0402	-18.1921	-18.2078	-18.2094
Time (in seconds)	0.0012	0.0007	0.0014	0.0114
Simpson	-17.8836	-18.176	-18.2062	-18.2092
Time (in seconds)	0.0009	0.0007	0.0015	0.0114
Montecarlo Crude	-18.2349	-18.6906	-17.9991	-18.2703
Time (in seconds)	0.001	0.0006	0.0022	0.0129
Montecarlo Importance	-17.7369	-18.1602	-18.2046	-18.209
Time (in seconds)	0.0055	0.0042	0.1626	6.1616

Question 3

The code *econ714_homework1_question3.m* generates the output of question 3. The function attains a minimum at the point $(1, 1)$ and its value is $f(1, 1) = 0$. We use four initial guesses $(2, 3)$, $(2, -3)$, $(-2, 3)$ and $(-2, -3)$.

- Newton-Raphson. This algorithm converges in a small number of iterations, but it requires the computation of the inverse of the Hessian, which can be slow as we increase the number of dimensions of the problem. Table (2) shows the number of iterations and time elapsed.

- BGFS. This algorithm requires more iterations and more times because we are approximating the inverse of the Hessian (without calculating it directly). Table (2) shows the results. We used the following initial guess for the inverse of the Hessian.

$$H_0^{-1} = \begin{pmatrix} \frac{1}{f''_{xx}(1,1)} & 0 \\ 0 & \frac{1}{f''_{yy}(1,1)} \end{pmatrix} = \begin{pmatrix} \frac{1}{802} & 0 \\ 0 & \frac{1}{200} \end{pmatrix}$$

Table 2: Comparison of Different Methods - Minimization

Method	GuessI	GuessII	GuessIII	GuessIV
Newton Raphson	5	4	6	5
Time (in seconds)	0.0052	0.0004	0.0002	0.0002
BFGS	32	23	73	53
Time (in seconds)	0.0078	0.0017	0.0009	0.0012

- Steepest descent. Figure (2) plots the trajectories for different initial guesses and has the number of iterations and time elapsed included. The algorithm is pretty fast, but since the function is very flat near the minimum the trajectories are not optimal. Circles are the initial guesses and the star is the solution.

- Conjugate descent. Figure (3) shows the results. This one is slower but we see that the trajectories seem smoother. We implemented the non-linear version and as step

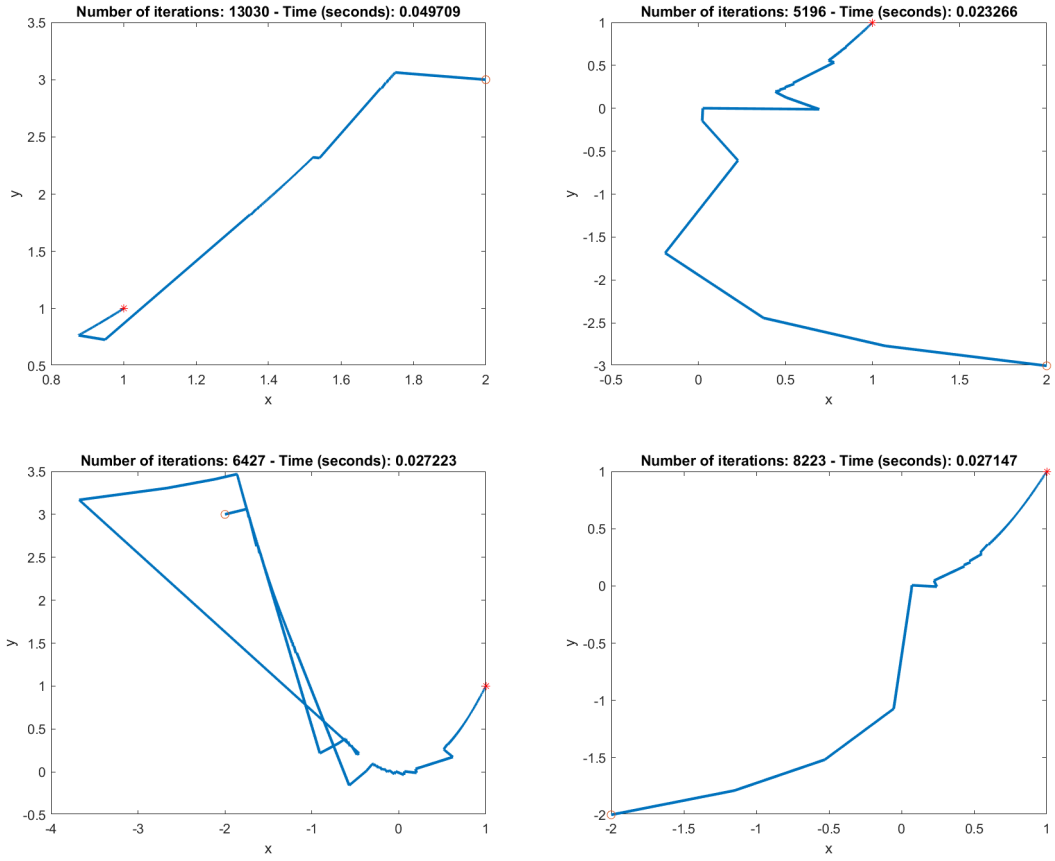

Figure 2: Steepest Descent Method

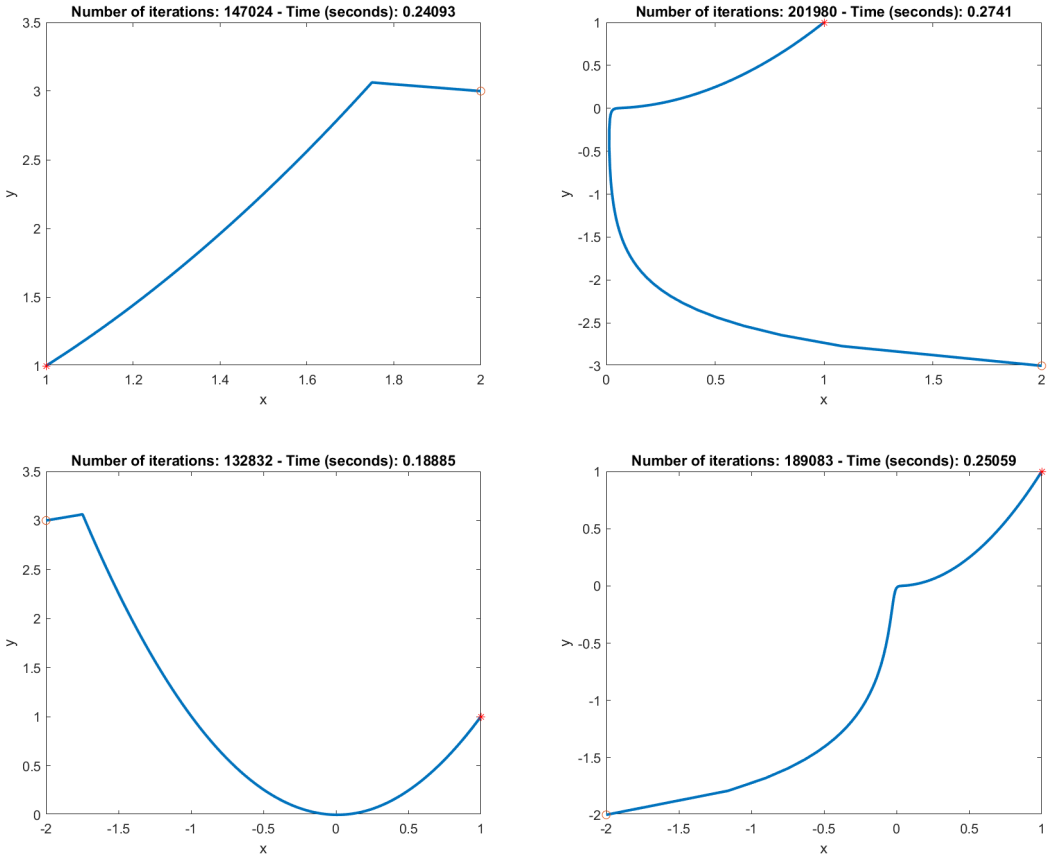

Figure 3: Conjugate Descent Method

size we used the inverse of the maximum eigenvalue of the Hessian matrix at the initial guess. As before, circles are the initial guesses and the star is the solution.

Question 4

The code *econ714_homework1_question4_question5.m* generates the output of this problem. This code uses functions *sp_objective_function.m* and *spp_solution.m*. The first function returns the objective function of the social planner and the second function implements the algorithm that solves the social planner problem. Before showing the results, we describe what the algorithm does.

- Let the social planner know the vector of parameters α a vector of weights λ and a matrix of preference parameters called ω. Additionally, she knows the total endowment of each good.
- She takes the total endowment of each good and divides it between the number of agents of the economy. If the problem was symmetric (including the weights the social planner gives to each agent) we know that the solution would be to divide the total endowment of each good evenly. We are interested in the cases where the problem is not fully symmetric. Divide the total endowment of each good by the number of people in the economy. This is the initial guess of the algorithm.
- Start with good 1. Choose two people at random: we will reallocate a random amount of good 1 between them. Call x_{star} the bundle after reallocation. For this we need to tell the social planer the standard deviation of the random variable we are using to make the reallocation. Check that the consumption is not negative and make sure that we are dividing the total endowment of the good (the social planner does not want to waste a single unit).
- Once we determine the random amount we want to reallocate, define the next bundle as a Bernoulli random trial between the previous bundle and the one with the reallocation. Let $p = \min\{1, f(x_{\text{star}})/f(x_{\text{prev}})\}$ be the parameter of the Bernoulli random trail where getting x_{star} is a success.
- If there was an improvement in the objective function, keep the draw. If there was no improvement, disregard it.
- Repeat this process for goods $j = 2, \ldots, m$.
- If we see 100000 iterations without improvements, stop. This number could be modified to address the fact that having more goods automatically increases the number of iterations.

Although the algorithm is slow because it performs a lot of iterations, it can handle asymmetries and a higher number of agents and goods. Consider a total endowment of 100 for each good, let's see how the social planner reallocates it under different parameter values for the case $m = n = 3$.

- Let $\omega_j^i = \omega = -2$ for every agent and good. Include heterogeneity in the weights and the parameter α_j. In particular, $\lambda = (1, 2, 3)$ and $\alpha = (3, 2, 1)$. Table (3) shows the efficient allocation. These results make sense because the problem is fairly symmetric and the social planner values the utility of individual 3 the most. The last column of the table has the equilibrium prices.

Table 3: Allocations for $\omega_i^j = -2$, $\alpha = (3, 2, 1)$ and $\lambda = (1, 2, 3)$

Goods	AgentI	AgentII	AgentIII	Price
GI	31.7	36.6	31.7	1
GII	31.7	36.6	31.7	0.6667
GIII	31.7	36.6	31.7	0.3333

- Now let $\omega_j^i < 0$ be random, $\alpha = (1, 1, 1)$ and $\lambda = (1, 2, 3)$. Table (4) shows the results: the left panel includes the values of ω_j^i and the right panel the allocation. We see that the higher the value of ω_j^i (the closer it is to zero) the more consumption agent i gets of good j. This makes sense because higher values of ω_j^i are associated with higher marginal utility to the right of the point $x = 1$. Also remember that this social planer likes person $i = 3$ the most so she is willing to give person $i = 3$ more units.

Table 4: Allocations for random ω_i^j, $\alpha = (1, 1, 1)$ and $\lambda = (1, 2, 3)$

Goods	AgentI	AgentII	AgentIII
GI	-1.3155	-1.0113	-1.2541
GII	-1.1034	-1.0221	-1.1435
GIII	-1.3643	-1.4835	-1.1703

Goods	AgentI	AgentII	AgentIII	Price
GI	12.56	53.33	34.12	1
GII	17.14	42.32	40.54	1.2794
GIII	15.82	20.22	63.96	0.4766

Now let's jump to the case $m = n = 10$. Again there are 100 units of each good to allocate.

- Let $\omega_j^i = \omega = -2$ for everyone and every good. Here α is a vector of ones and $\lambda = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$. So the social planner values agent $i = 10$ utility the most. In table (5) we see that the algorithm can handle this case but we lose precision. Maybe we should increase the number of iteration before stopping, provided that we have more iterations because there are more goods now. Every individual consume the same of each good because the problem is symmetric except for the weights of the social planner. Note that in the competitive equilibrium we get that all the relative prices are the same. This makes sense because the problem is symmetric, the social planner is the one who introduces the asymmetry having different weights.

- Finally let $\omega_j^i < 0$ be random, α be a vector of ones and again let the weights differ $\lambda = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$. Table (6) shows the results: the first table has the values of ω_i^j and the second table has the solution of the social planner problem. For a given agent, the social planner allocates more units if the ω_j is higher (closer to zero). And again we see a preference towards individuals to the right because we are giving them more weight.

Table 5: Allocations for $\omega_i^j = -2$, α symmetric and $\lambda = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$

Goods	AgntI	AgntII	AgntIII	AgntIV	AgntV	AgntVI	AgntVII	AgntVIII	AgntXI	AgntX	Price
GI	4.44	6.29	7.7	8.91	9.95	10.91	11.79	12.59	13.35	14.07	1
GII	4.46	6.29	7.69	8.9	9.96	10.92	11.79	12.58	13.35	14.07	1
GIII	4.44	6.3	7.72	8.89	9.96	10.92	11.78	12.58	13.33	14.07	1
GIV	4.46	6.3	7.71	8.9	9.96	10.88	11.8	12.57	13.33	14.08	1
GV	4.45	6.3	7.71	8.91	9.94	10.88	11.79	12.6	13.35	14.07	1
GVI	4.45	6.29	7.7	8.9	9.94	10.91	11.77	12.6	13.35	14.1	1
GVII	4.45	6.3	7.71	8.88	9.93	10.93	11.77	12.58	13.35	14.08	1
GVIII	4.45	6.3	7.7	8.9	9.96	10.89	11.8	12.61	13.36	14.04	1
GIX	4.46	6.28	7.71	8.91	9.94	10.9	11.77	12.62	13.36	14.06	1
GX	4.44	6.3	7.73	8.9	9.95	10.9	11.78	12.59	13.34	14.07	1

Table 6: Allocations for random ω_j^i, α symmetric and $\lambda = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$

Goods	AgntI	AgntII	AgntIII	AgntIV	AgntV	AgntVI	AgntVII	AgntVIII	AgntXI	AgntX
GI	-1.0649	-1.203	-1.1377	-1.0004	-1.5101	-1.1552	-1.264	-1.2185	-1.0696	-1.0986
GII	-1.0983	-1.5908	-1.0079	-1.2276	-1.382	-1.3208	-1.2064	-1.0905	-1.6704	-1.0623
GIII	-1.4879	-1.0204	-1.1805	-1.4035	-1.3234	-1.2779	-1.31	-1.451	-1.5806	-1.1777
GIV	-1.3756	-1.3956	-1.0986	-1.2156	-1.1914	-1.1027	-1.0859	-1.1985	-1.0149	-1.396
GV	-1.258	-1.1303	-1.0773	-1.0131	-1.5188	-1.669	-1.0279	-1.2068	-1.0266	-1.0354
GVI	-1.0213	-1.1791	-1.1459	-1.3779	-1.2195	-1.0268	-1.3945	-1.1155	-1.1438	-1.4064
GVII	-1.3102	-1.2443	-1.2617	-1.0095	-1.0736	-1.0016	-1.1075	-1.0855	-1.0245	-1.4576
GVIII	-1.3279	-1.1801	-1.441	-1.0307	-1.055	-1.1698	-1.0342	-1.0117	-1.0874	-1.1047
GIX	-1.2549	-1.3699	-1.3785	-1.0065	-1.054	-1.2385	-1.0389	-1.1098	-1.1403	-1.4322
GX	-1.1623	-1.0626	-1.1174	-1.4374	-1.2479	-1.1987	-1.0269	-1.1677	-1.1846	-1.0081

Goods	AgntI	AgntII	AgntIII	AgntIV	AgntV	AgntVI	AgntVII	AgntVIII	AgntXI	AgntX	Price
GI	2.46	3.95	6.12	10.42	5.49	10.82	9.99	12.1	19.14	19.51	1
GII	2.69	3.05	8.71	7.51	7.04	8.87	12.38	18.28	7.13	24.34	0.8842
GIII	2.47	7.37	7.89	6.98	9.29	11.62	12.31	10.58	9.39	22.11	0.7398
GIV	2.02	3.3	6.55	6.91	8.68	12.18	14.64	12.73	22.58	10.41	0.925
GV	2	4	6.23	9.29	5.12	4.93	15.5	11.53	19.87	21.52	1.0086
GVI	2.86	4.51	6.68	5.98	9.08	16.44	8.78	17.02	17.54	11.12	0.9626
GVII	1.94	3.5	4.75	9.3	10.03	14.2	12.62	15.04	19.83	8.79	1.0632
GVIII	1.83	3.55	3.73	8.31	9.81	9.15	14.17	17.2	15.71	16.54	1.0992
GIX	2.18	3.39	4.5	10.47	11.65	9.35	16.66	15.71	16.2	9.88	0.975
GX	2.16	4.44	5.92	4.89	7.4	9.34	15.82	12.73	13.54	23.77	1.035

Question 5

The equilibrium prices are computed using the function *eq_prices.m*. The same file *econ714_homework1_question4_question5.m* generates the output. Tables (3), (4), (5) and (6) show the equilibrium prices for the cases we presented in question 4, but here we include some additional cases that are interesting. Before, the initial endowment was the same for all goods and also sometimes the value of α was unchanged. Here it will be interesting to vary the initial endowment to make some good more or less scarce and see how that affects the equilibrium prices. We describe what the algorithm does to compute the equilibrium prices. We use the Newton-Raphson method for nonlinear systems of equations.

- Guess a price vector. Start with one person and calculate her income using the guess. The initial guess for $x^i = (x_1^i, \ldots, x_m^i)$ is the income divided by $m \cdot p_j$. This would be the solution if the utility function was Cobb Douglas.
- Now calculate the first order condition at this value of x^i. This a vector valued function $F(\cdot)$ (with m dimensions) where the first equation is the budget constraint and we have eliminated the Lagrange multiplier. Calculate the norm of $F(\cdot)$: as long as the norm is higher than our tolerance level, we continue iterating.
- Calculate the Jacobian matrix J of the system and define a possible step as $-J^{-1} \cdot F(\cdot)$. Change the step if it is too high: we don't want to move too fast. See equation (1) for more details about the system.
- Update the point x^i and start again. After iterating we get the individual demand of person i.

- Repeat for everyone and get the aggregate demand.
- If the aggregate demand of a good is greater than the aggregate endowment, randomly increase the price of that good. Similarly, if the aggregate demand of a good is smaller than its aggregate endowment, randomly decrease the price. Here we use the absolute value of a zero mean normal random variable. We let the standard deviation depend on the excess demand so that the price changes makes sense in every iteration (they get smaller as we approach the solution).
- Start the process again and repeat until the aggregate demand and aggregate endowment are close to each other.

The algorithm is slow, but it seems to work at least when the heterogeneity is not too high. When the values of ω_i^j are very different from each other, the algorithm has troubles inverting the Jacobian. We tried to address this by forcing the method to take smaller steps so that the matrix does not change too much from iteration to iteration. In particular, the maximum step in a given direction cannot exceed 5% of the maximum amount the agent could buy of that good, that is, $0.05 \cdot m/p_j$, where m is the income. Table (7) presents the equilibrium prices of seven possible scenarios. In all of them there are 3 people and 3 goods.

1. First $\alpha = (3, 2, 1)$, there are 100 units of each good. Person i owns all the endowment of good $j = i$. Here we see that good 1 is relatively more expensive because it is more desirable. We use $\omega_j^i = -2$ for all i and j.
2. Redistributing the initial endowment from the first scenario does not change the equilibrium price vector. Here everything is the same as before but now the initial endowment is 100/3 for each good and each person.
3. Now there are 90 units of good 1, 100 units of good 2 and 120 units of good 3 owned by agents 1, 2 and 3 respectively.
4. Same as the previous one but the initial endowment is shared equally between everyone. we see that the equilibrium prices do not change.
5. Now we go back to scenario 1 but we vary ω. Person i has the same $\omega_j^1 = -2$ for every good, person 2 has $\omega_j^2 = -2.05$ for every good and person 3 has $\omega_j^3 = -2.1$ for every good. We see that the equilibrium prices do not change when comparing to the first scenario. This makes sense because the preferences are homothetic.
6. Now good 1 has $\omega_1^i = -2$ for every individual, good 2 has $\omega_2^i = -2.05$ for everyone and good 3 has $\omega_3^i = -2.1$ for everyone. Now we see differences in the equilibrium prices.
7. Same as scenario 1 but now we use $\alpha = (1, 2, 3)$. Note that $p^* = \alpha$.

For each individual $i = 1, \ldots, n$ we solve the following system. We omit the superscripts that refer to the agent. $F(x)$ is $m \times 1$. We eliminate the Lagrange multiplier by using the first order condition of the agent.

Table 7: Equilibrium prices of 6 possible scenarios

Scenarios	GoodI	GoodII	GoodIII
Scenario I	1	0.6667	0.3333
Scenario II	1	0.6667	0.3333
Scenario III	1	0.54	0.1875
Scenario VI	1	0.54	0.1875
Scenario V	1	0.6667	0.3333
Scenario VI	1	0.5554	0.2314
Scenario VII	1	2	2.9999

$$\begin{cases} F_1(x) & = p_1x_1 + \cdots + p_mx_m - p_1e_1 - \cdots - p_me_m = 0 \\ F_j(x) & = \frac{\alpha_j}{\alpha_1} \cdot \frac{x_j^{\omega_j}}{x_1^{\omega_1}} - \frac{p_j}{p_1} = 0 \quad \text{for} \quad j = 2, \ldots, m \end{cases} \tag{1}$$

The Jacobian of the system looks like this. we express it in terms of $F(x)$ to help with the computation.

$$J = \begin{pmatrix} p_1 & p_2 & \cdots & p_m \\ -\frac{\omega_1}{x_1}F_2(x) & \frac{\omega_2}{x_2}F_2(x) & \ldots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ -\frac{\omega_1}{x_1}F_m(x) & 0 & \ldots & \frac{\omega_m}{x_m}F_m(x) \end{pmatrix}$$