Problem Set 2

AEM 7130

Problem Set 2: Due March 17 at 11:59PM.

Make sure your code is well-commented and reproducible. Unless stated in the problem, you can (and may need to) use some Google searching to find how to code parts of your answers.

Problem 1: Git

1. Create a new repository named `problem-set-1-q-3` on your GitHub account.
2. Put in a `README.md` with the following text: `Hello World!` .
3. Put in a .gitignore file, ignoring the Jupyter files .ipynb_checkpoints and the project files, .projects.
4. Create a new branch called `new-branch` .
5. Change the `README.md` text to `Goodbye World!` .
6. Create a pull request to merge `new-branch` back into `main/master` .
7. Merge the branch back in.

Problem 2: Memory location

Let's learn about some of the nuances of memory allocation.

1. Generate one $20,000 \times 20,000$ array of random numbers named `x` .
2. Make a function called `exp_cols` which exponentiates the elements of `x` column by column (i.e. by broadcasting `exp.()`) and returns the exponentiated array.
3. Make a function called `exp_rows` which exponentiates the elements of `x` row by row (i.e. by broadcasting `exp.()`) and returns the exponentiated array.
4. Call `exp_cols(x)` and `exp_rows(x)` twice and calculate the elapsed time and memory allocation on the second call (avoids fixed cost of initial compiliation).
5. Is one faster than the other?
6. If you exponentiate the full array instead of looping column-by-column or row-by-row how does it compare?

Problem 3: Newton

Consider a Cournot duopoly where inverse demand is given by $P(q) = q^{-1/\eta}$, and the firm cost functions are $C_1(q_1) = 0.5c_1 q_1^2, C_2(q_2) = 0.5c_2 q_2^2$.

1. Write your own Newton's method solver for this general problem: `cournot_newton(eta, c1, c2, initial_guess, tolerance)` .

Hint: think about the first-order conditions and best response functions.

2. Solve for the equilibrium outcome using Newton's method for $\eta = 1.6, c_1 = .15, c_2 = 0.2.$

Problem 4: Optimization packages

Suppose a consumer has the following CES utility:

$$u(C_1, C_2) = (\kappa C_1^{(\eta-1)/\eta} + (1-\kappa)C_2^{(\eta-1)/\eta})^{\eta/(\eta-1)}$$

and faces a budget of

$$p_1 C_1 + p_2 C_2 = W$$

1. Recast the problem as an unconstrained utility maximization problem.
2. Use the `Optim` or `JuMP` package and write a program `utility_maximizer(kappa, eta, p1, p2, w, initial_guess, solver)` to solve this utility maximization problem using one of the following solvers
    - Nelder-Mead
    - Newton
    - Conjugate gradient

Feel free to use the default settings (tolerances, etc) for the optimization package or change them. 3. Choose a set of parameters and solve it analytically. Also solve it numerically with the same set of parameters and report the algorithm results (e.g. the output shown here (https://julianlsolvers.github.io/Optim.jl/stable/#algo/cg/)). - Note: Let $0 < \kappa < 1$ and $\eta > 1$