

Computational Methods in Economics and Finance

Exam 1

This test is due Wednesday Feb. 24 at 2:30 PM. You should follow the guidelines discussed in the syllabus carefully when submitting your answer. All work should be your own and you should not discuss the exam with anyone but me. Collaborating and/or giving or receiving help on this exam is a violation of the university's honor code and will result in sanctions (see syllabus for more information).

4.1. Consider the function

$$\Psi(x, y) = x + y - u \left(1 + \frac{1}{3} \left(\frac{v}{u} \right)^2 \right)$$

where $u = \max(|x|, |y|)$ and $v = \min(|x|, |y|)$ (define $\Psi(0, 0) = 0$). Discuss how this function should be evaluated to avoid numerical difficulties (note that x and y can take on any real values).

Determine the derivatives of the function with respect to x and y and develop a method by which they can be accurately evaluated (the derivatives are not defined at $(0, 0)$ so you will need to do something reasonable at this point).

Finally write a MATLAB function that accepts x and y and returns Ψ , Ψ_x and Ψ_y . The code should accept array inputs (i.e., x and y can be arrays of any size or scalars and the output should be an array of the same size). Test this code to ensure that it is accurate for any values of x and y .

It may be of interest that this function, like ϕ^- , has the same 0-contour curve as $\min(x, y)$.

4.2. Quasi-Newton methods for solving non-linear equations ($f(x) = 0$) are typically defined as iterative methods of the form

$$x_{k+1} \leftarrow x_k - \lambda_k B_k f(x_k)$$

where the sequence of matrices B_k satisfies the so-called "secant condition"

$$B_{k+1}(f(x_{k+1}) - f(x_k)) = x_{k+1} - x_k$$

You may recall that the Broyden method uses the B_{k+1} that satisfies the secant condition and is, in a certain sense, as close to B_k as possible (specifically the usual Broyden method minimizes the Frobenius norm

of $[B^{k+1}]^{-1} - [B^k]^{-1}$ subject to the secant condition). Suppose instead that you pick B_{k+1} so that B_{k+1} differs from B_k only in a single column. This can be written as

$$B_{k+1} \leftarrow B_k + ue_j^\top$$

where e_j is the j th column of an identity matrix and u is a vector to be determined. Find the u that satisfies the secant condition.

To define a fully constructive algorithm, one must have a method for picking j . Suppose j is picked to be the index of the largest absolute value of $f(x_{k+1}) - f(x_k)$, i.e., the column of B picked to change is associated with the largest change in the function value. Modify the `CompEcon` procedure `broyden` to use this updating rule for B (be sure to edit the documentation). Test your function and compare it to the performance of `broyden` (be sure to test it on a multivariate function).

4.3. Consider the function

$$f(\alpha, \beta, \gamma; x) = \alpha\beta \sum_{i=1}^n \frac{z_i^\beta (1 - z_i)}{x_i}$$

where x is an n -vector and

$$z_i = \frac{(x_i/\gamma)^\alpha}{1 + (x_i/\gamma)^\alpha}$$

Using the vector x provided, find the value of $\theta = [\alpha; \beta; \gamma]$ that maximizes f . Do this using a global optimizer and a quasi-Newton solver using closed form expression for the derivatives (you may use numerical derivatives but I will take points off). To answer this question you should write a function that accepts θ and x and returns the function and its gradient. This function should then be passed to the appropriate solvers.

4.4. An alternative global optimization method with a number of similarities to the genetic algorithm can be described as follows. Let $f(x) : \mathbf{R}^d \rightarrow R$ be the objective function. Suppose that one begins with an initial population of n points in \mathbf{R}^d . Associated with the j th point is the fitness value $f_j = f(x_j)$. At each iteration all of the points will move to new locations according to a specified rule, tracing out a trajectory. For each trajectory the best location (most fit) will be stored, as will the best location for all the trajectories so far found. The direction the j th trajectory moves at each iteration depends on its displacement

from its own best location and on the location of the overall best point so far found.

At the k th iteration, the algorithm updates each of the n points using the rule (for the j th trajectory)

$$v_j^{k+1} = wv_j^k + c_1u_1^k(\hat{x}_j - x_j^k) + c_2u_2^k(\hat{x} - x_j^k)$$

$$x_j^{k+1} = x_j^k + v_j^{k+1}$$

where w , c_1 and c_2 are constant parameters and u_1 and u_2 are random d -vectors (the multiplication should be interpreted as element by element multiplication). Also \hat{x}_j is the best location found so far for the j th trajectory and \hat{x} is the best overall location so far found.

You may initialize v to equal 0. The three parameters w , c_1 and c_2 should be controllable by the user. Default values might be 0.5, 1 and 1, although you may decide that other values are more appropriate. Other options include the size of the population, the number of iterations and the convergence criteria.

Write a MATLAB function that implements this algorithm and test it using the Jones suite of test functions (you might also use the banana function in the CompEcon Toolbox). Be sure to document your function. Ideally the function should have the same input/output format as genetic and DiRect to facilitate easy comparison.