Exercise: Interpolation and Optimization in Matlab

Stockholm Doctoral Program in Economics
Computational Bootcamp

Kathrin Schlafmann

Questions

1. 1-D interpolation:
   Take the following function:
   $$v(x) = 0.25x^5 - 1.2x^4 \tag{1}$$
   We will use this function as an example. In practice, you would obviously not know the function that generates your sample, otherwise you would not need to interpolate in the first place.

   (a) You have a sample of 15 linearly spaced points between -2 and 5 (for example because this is the grid of state variables you have solved your maximization problem for). But now you need the value of this function for a finer grid.
   Complete the provided code fragment to interpolate using 4 different methods of interpolation: linear, spline, nearest neighbor and pchip. Look at the plot: which one do you think performs best?

   (b) Now imagine you only had a sample of 5 linearly spaced points in the same range. Repeat the interpolation and look at the resulting interpolation. What do see?

   (c) Thinking about nearest neighbor interpolation: Do you think it is good? In which situation do you think it is good? Why?

   (d) Now you realize that you need values for the function also for points outside of the range of the sample: you need points in the range -4 to +8. You think you can save computation time by extrapolating from the existing sample (after all, it might be very costly to solve your numerical problem for more points). Allow the interpolation to extrapolate and compute the approximated values.

   (e) Look at the plot: What do you think? Which method performs best? Is that a general result?

2. 2-D interpolation:
   For this example we will work with the following function:
   $$v(x, y) = x \cdot e^{-x^2-y^2} \tag{2}$$
   To use `interp2()` we need to obtain inputs x and y in a format that provides all possible combinations of these variables and the corresponding values $v(x, y)$ with it.

(a) You are interested in the range of $x, y \in [-2, 2]$. Use `meshgrid()` to construct matrices with all possible combinations for the two variables, where each variable is supposed to be 16 equally spaced points in this interval. (Note: you can also use `ndgrid()` to obtain the same matrices) Look up the `meshgrid()`-function in the Matlab help files to learn how the function works.

(b) Construct the values $v(x, y)$ that correspond to these matrices.

(c) Interpolate between the points to obtain 40 equally spaced points using `interp2()`.

(d) Compare the resulting plots. What do you observe?

(e) Repeat the exercise for the case where you only have a grid available with 6 linearly spaced points in each direction. What do you see now?

3. Repeated interpolation of the same set of points:
For this exercise we will work with the following function:

$$V(x, y) = \frac{\sin\left(\sqrt{x^2 + y^2}\right)}{\sqrt{x^2 + y^2}} \tag{3}$$

(a) Imagine you have sample of all possible combinations of coordinates between -2 and 2 (equally spaced, 10 points in each direction. Construct this sample.

(b) Your code needs to evaluate this function at several points in the algorithm. We represent this by a loop over query points. (note: If this was literally what you need to do then doing one interpolation for all points at once would be much more efficient than looping over the points!)
Use `interp2()` to interpolate in each iteration of the loop with spline interpolation.

(c) Now you realize that repeatedly interpolating interp2 is fairly time-consuming. Rewrite to use `griddedInterpolant()` instead.

(d) Compare the time needed to run the code with the two interpolation functions. Use the function `tic` and `toc` to measure the time spent in the code.

4. Optimization: grid search:
You want to solve the following optimization problem:

$$\max_{c1,c2} \; u(c_1) + \beta u(c_2) \tag{4}$$

$$\text{s.t.} \quad c_2 = (x - c_1)R \tag{5}$$

$$u(c) = \frac{c^{1-\sigma}}{1-\sigma} \tag{6}$$

$$0 < c_1 \leq x \tag{7}$$

$$x \text{ given} \tag{8}$$

You decide to solve it via grid search. To do this you implement the following steps:

(a) Set parameter values and grids for states and controls. What are the parameters here and what are the states and controls? In your opinion, what is a good grid for states and controls if you are interested in solving the problem up to $x = 10$?

(b) Compute the utility of all possible consumption choices for all values of the state variables. Complete the provided function `CRRAutility` and use it. Make sure to exclude choices for consumption that are infeasible, ie satisfy equation 7.

(c) Determine for each value of the state variable the choice of control variable that delivers the highest utility.

(d) Use the functions `tic` and `toc` to measure how the solution takes.

(e) Play around with the number of grid points for the state and the control variables. What do you see?

5. Optimization: `fmincon`:
You are still interested in the same optimization problem as in question 4, but now you want to solve it using fmincon. You do this in the following steps:

(a) Set parameter values and grids for states. Why don't you need to set a grid for controls in this case? Use the same grid for states that you used in question 4.

(b) Optimize using `fmincon`:

i. Set options

ii. Set lower and upper bounds for the control variable. Why do you need to do this?

iii. Set initial values for the optimizer to start searching. What is a good value?

iv. Call `fmincon` using the same function as you used in question 4 (`CRRAutility`).

(c) Compare the resulting solution to the one you obtained in question 4. What do you see?

(d) Measure the computing time and compare to the grid search.

6. Optimization: `fsolve`:
Again, we are solving the optimization problem from question 4, but this time we want to use `fsolve`.

(a) `fsolve`solves an equation, but the problem we have is a maximization problem. So how can we use it?

(b) Derive the FOC of the problem analytically.

(c) Complete the function `EulerEquationDifference` that computes the difference between the LHS and RHS of the Euler Equation.

(d) Follow similar steps as before to solve numerically:

i. Set parameter values and grids. Which ones do you need?

ii. Set initial values for `fsolve`to start searching.

iii. Call `fsolve`to set `EulerEquationDifference` equal to zero.

(e) Compare the resulting solution to the one you obtained in question 4. What do you see?

(f) Measure the computing time and compare to the previous two methods.