Computational Methods in Macroeconomics

ECON 213, Winter 2007

Instructor: Irina A. Telyukova
E-mail: itelyukova@ucsd.edu
Office: Economics 224
Phone: 822-2097
Office hours: Wed 9:30-11:30 or by appointment
Class hours: Tue, Thu 9:30-10:50 in Economics 300

I. Description of the Course

Models in modern macroeconomics often are analytically intractable and have no closed-form solutions. In order to answer many interesting questions in the field, macroeconomists have been leaning more and more on numerical solutions of their models using powerful computers. In this course, we will learn many of the standard computational techniques required to solve such models. We will cover a little bit of macroeconomic theory by studying some standard macro models that require the computational approach, but unfortunately there is not enough time do go into economics in depth. This course is really about tools. We will begin by a brief overview of dynamic programming, followed by a survey of numerical methods required when analytical methods are not available for solving equations and systems of equations, for finding derivatives and integrals, and for solving optimization problems. After this, in the context of some standard models, we will look at methods that can be used to solve them. Finally, I hope to have time for at least a brief discussion of calibration - to give you a clearer idea of where the computation of a model fits in the overall progression of data-based economic research.

II. Requirements for the Course

1. Homework. By its nature, the learning of methods requires using them. So there will be a number of hands-on homeworks assigned, which you should do carefully. I encourage you to talk to each other as you do these to speed up the learning. Learning computation for the first time is quite time-consuming; be ready to spend the time! At each homework due date, I will ask one of you to talk about the homework briefly and describe your results and also any problems you have encountered along the way. On the same day, you will also need to hand in your homework to me, in hard copy, with a printout of your code and a write-up of the results. I will grade these informally to give you an idea of where you stand. Your "cumulative" homework performance (including the class presentation) will account for 60% your final course grade.

2. Project. By the last day of class, you are to submit a referee report on a computational paper from a list to be distributed at a later date. The referee report should consist of a summary of the paper, constructive commentary on the question and the execution of the solution - especially its computation and parameterization aspects - and some equally constructive suggestions for improvements or changes you would have liked to see. Slightly apart from a typical referee report, I ask you also to write a section at the end where you either propose an interesting

extension of the paper, or a related question on which you yourself might consider writing a paper, employing methods learned in class. This is thus a combination of a referee report and a brief research proposal inspired by the paper you review. The last section of this project is a crucial part to be taken very seriously. To do a good proposal, it may be necessary for you to review other literature related to the paper at hand, and I expect not only a write-up of an idea, but also a fairly clear plan on how you would pursue the execution of the idea, computation included. This project will account for 40% of your grade.

III. Programming Languages

This is not a computer science course, and I will not teach you programming - this part is up to you. So it helps if you have had previous programming experience in a language such as C, but it is not a prerequisite - you can learn as you go. If you intend to work with computation in your research, now is the time to invest in learning Fortran 90. It is much (I have heard numbers in the hundreds of times) faster than Matlab and other comparable languages, and is used most widely by the scientific community. The speed will really matter if you ever plan to attempt any serious computation-based research. Matlab can be a fine substitute if you are not planning on taking computation far beyond this course.

For both, there are lots of codes available online - which is a good way to learn. There is also a good Matlab primer available if you have never used it before. *Numerical Recipes* cited below are great for help when you are learning Fortran 90. Finally, there is a book by Ellis, Phillips and Lahey, *Fortran 90 Programming*, that is a good reference - I have vaguely heard that it is out of print, but you can probably locate it at half.com and other such places.

IV. Relevant Books

There is no one book that covers all of the material we will discuss, and there is a lot more relevant material out there than we will have time for. The following forms the beginnings of an excellent reference library for a macroeconomic theorist who is interested in computation. I will refer you to parts of these books when relevant. Some of them are available in the bookstore.

1. Stokey, N., R. Lucas with E. Prescott. *Recursive Methods in Economic Dynamics*. 1989.
2. Sargent, T. and L. Ljungqvist. *Recursive Macroeconomic Theory*. MIT Press, 1998.
3. Heer, B. and A. Maußner. *Dynamic General Equilibrium Modelling*. Springer, 2005.
4. Judd, K. *Numerical Methods in Economics*. MIT Press, 1998.
5. Marimon, R. and A. Scott, eds. *Computational Methods for the Study of Economic Dynamics*. Oxford University Press, 1999.
6. Cooley, T.F., ed. *Frontiers of Business Cycle Research*. Princeton University Press, 1995.
7. Press, W., S. Teukolsky, W. Vetterling and B. Flannery. *Numerical Recipes in Fortran77: The Art of Scientific Computing*. Cambridge University Press, 2001. *Available full-text online along with a Fortran 90 edition.*

V. Course Outline

1. **Dynamic Programming**

   (a) Finite horizon and the Theorem of the Maximum

   (b) Infinite horizon and the Contraction Mapping Theorem

   Reference: Stokey, Lucas, Prescott (1989), chapters 4,5.

2. **Numerical Methods**

   (a) Function Approximation Methods
   - Finite-elements methods: linear interpolation, splines, etc.
   - Weighted-residuals methods
   - Chebyshev polynomials

   (b) Numerical Differentiation

   (c) Numerical Integration
   - Newton-Coates quadrature
   - Gaussian quadrature
   - Monte-Carlo methods

   (d) Root Finding (Solving Systems of Equations)

   (e) Numerical Optimization

   Reference: Heer and Maußner (2005), chapter 8; Judd (1998).

3. **Representative Agent Models**

   (a) Value Function Methods
   - Discretization
   - Linear quadratic methods
   - Finite elements methods
   - Weighted residuals methods

   (b) Euler Equation Methods
   - Linearization
   - Perturbation
   - Policy function approximation
   - Parameterized expectation approach

   References: Heer and Maußner (2005), chapters 1-4. Marimon and Scott (1998), ch. 2-7.

4. **Bewley-Aiyagari Heterogeneous Agent Model**

   (a) Stationary equilibrium (Aiyagari, *QJE* 1994)

   (b) Comparison of steady states: transition paths

   (c) Aggregate uncertainty (Krusell-Smith, *Macro. Dyn.* 1998)

   Reference: Heer and Maußner (2005), chapters 5-6; Sargent and Ljungqvist (2004), chapters 16, 17; Rios-Rull (1999) in Marimon and Scott (chapter 11).

5. **Calibration**

   (a) Classic approach: Cooley and Prescott (1995), in Cooley (chapter 1).

   (b) A more involved approach: Chatterjee, Corbae, Nakajima and Rios-Rull (2002).

   (c) Estimation: Gourinchas and Parker (*Econometrica* 2002).