1. Course Purpose

"Our task, as I see it, is to write a FORTRAN program that will accept specific economic policy rules as 'input' and will generate as 'output' statistics describing the operating characteristics of time series we care about, which are predicted to result from these policies."

Robert E. Lucas, Jr.

Economic theory, as defined by Bob Lucas, amounts to writing down a mathematical model that represents the optimization problems of all actors, as well as any consistency conditions for the aggregates, and then working out their implications. Modern applied macroeconomic theory has recognized that these problems, whether they belong to households, firms, governments, or social planners, are inherently dynamic; for example, investment, which is essentially the choice of a society to consume in the future rather than today, is by far the most volatile aggregate over the cycle. Thus, solving dynamic problems is critical for working out what our theories mean.

The language of dynamic programming (recursive methods) has been indispensable in permitting researchers to formulate and solve these problems, and as a result essentially all of modern macroeconomics employs this formulation. Unfortunately, most interesting macroeconomic models do not have analytical solutions, so the field has by necessity moved into computational methods. But if one is to really get to the implications of the models one must solve the models accurately; there are a number of examples in the literature of "implications" that turned out to be artifacts of
inadequate solution methods. And of course we also need to take into consideration time – if your model takes 10 years to solve, tenure is not likely to be in your future; as with accuracy, though, often slowness is merely a result of using bad methods and not an inherent feature of the problem. Getting the mix of speed and accuracy right is critical, and being aware of and comfortable with the best available methods is the first step toward that goal.

The purpose of this course is to introduce you to the tools of quantitative economic theory, namely recursive methods for describing problems and numerical methods for solving them. I hope to get you far enough that you are comfortable pushing further, since we lack the time to bring you all the way to the frontier; a lot of numerical stuff looks intimidating at first, but with a little familiarity you can work through a lot of it.

2. Text

The textbook is

   
   This book is an indispensable reference guide for a lot of the tools used today (although it is getting a bit outdated). For additional references you can consult


   I will post class notes for most of the topics, as well as a comprehensive set of notes on mathematical programming that will fill in a lot of the gaps in the lectures.

3. Coursework

   "If you have $N$ errors in your code and you fix one, you have $N$ errors left in your code." — Dan Bernhardt

   There will be graded homeworks (one every two weeks or so), which will be a mix of pencil-and-paper problems and computer coding. As the course progresses the assignments will weight more heavily toward the programming side. We will use Matlab for our programming environment, since
it is easy to learn and sufficiently powerful to do what we need it to do. In the long run you will benefit from learning a "lower level" language, like Fortran or C, or at least to learn how to compile Matlab code. Programming assignments tend to take a lot of time, as code rarely works right the first time (or the ninth time) you run it, so be prepared to dedicate some time. Working in groups is encouraged, but everyone should turn in their own assignment; keep in mind that programming is something you only learn by doing it.

4. Topics

The following outline is tentative; think of it as an aspirational list of topics. We will not try to move quickly and get to everything, but instead will make sure that what we do cover is absorbed completely.

1. A Simple Routing Problem

2. Basics of Deterministic Dynamic Programming
   1. Recursive vs. Sequential Representations of Dynamic Optimization Problems
   2. The Theorem of the Maximum and the Contraction Mapping Theorem
   3. The Bellman Equation with Example
      1. The Cass-Koopmans Optimal Growth Model
   4. Euler Equations

   1. The Linear-Quadratic Regulator Problem
   2. Discrete Dynamic Programming
      1. Basic Grid Search
      2. Howard’s Improvement
      3. Polynomial and Spline Approximation
         1. Solving Nonlinear Optimization Problems
         2. Solving Nonlinear Equations

4. Basics of Stochastic Dynamic Programming
1. Rudiments of Measure Theory
2. The Bellman Equation

5. Basics of Numerical Stochastic Dynamic Programming

1. The Linear-Quadratic-Gaussian Regulator Problem
2. Discrete Stochastic Dynamic Programming
   1. Markov Chains
   2. Numerical Integration
3. Example: The Real Business Cycle Model

6. Recursive Competitive Equilibrium

1. The 'big K, little k' Problem
2. Formal Definition and Example
   1. Decentralizing the Optimal Growth Model

7. Solving Numerically for Recursive Competitive Equilibrium

1. Local Methods
   1. Linear-Quadratic Recursive Competitive Equilibrium
   2. Perturbation
2. Global Methods
   1. Projection
   2. Finite Element Methods


1. Theory
2. Computation
   1. Steady State
   2. Deterministic Dynamics
   3. Stochastic Dynamics