Numerical Methods for Nonoptimal Economies

Troy Davig

Prepared for the
Center for Applied Economics and Policy Research
January 2007
Overview

- An applied overview of a particular method (‘monotone map’)

- Primary ingredients
  1. Interpolation / Extrapolation
  2. Numerical integration
  3. Non-linear equation solving

- Use a concrete example to highlight method, techniques and pitfalls
Disclaimer on Applicability

• Importantly, there is little reason to embark down the route of functional iteration described here if your model can be solved using methods with readily available software routines
  – e.g. perturbation methods, such as Dynare and the Schmitt-Grohe and Uribe routines

• When is this functional iteration method useful
  – Anytime you have a particular reason for solving the full nonlinear version of your model (and higher-order approximation methods won’t do)
  – Models with parameters subject to change
* Exogenously switching parameters governed by Markov chain (Svensson and Williams (2005) and Blake and Zampolli (2005) have methods in an LQ framework)

* Endogenously switching parameters (‘threshold-switching’)

* Switching between different ‘steady-states’
  – A particular nonlinearity is important in your analysis

* The zero nominal interest rate bound

• When is it not useful?
  – A large number of state variables
    * More than 4 probably requires an implementation in Fortran
– If your model has an indeterminacy

* Although ‘indeterminacy’ is unusual language when referring to nonlinear models, linear models without a determinate equilibrium fail to converge using the monotone map algorithm (unless starting guess is exactly the fundamental solution)

• Judd (1998) provides all the basic ingredients
Interpolation Overview

- Interpolation - a method for constructing a ‘nice’ functional approximation from a discrete set of data points

- Judd (1998), Chapter 6 gives details on the methods

- A simple example: Neoclassic production function

\[ Y = \sqrt{K} \]  

(1)

- In this example, we know the true function \( Y = f(K) \).

  - Later, we will ‘guess’ \( \hat{f}(K) \) and then apply an iterative method until we find a function that satisfies the first-order necessary conditions on every point of the state-space
• To approximate, define ‘grid’ or nodes consisting of \( n + 1 \) distinct numbers

\[ \Omega_K = \{K_0, K_1, K_2, ..., K_n\} \quad (2) \]

– this is the ‘state-space’, which will typically refer to the minimum set of state variables

• Define the data as

\[ D = \{(K_0, Y_0), (K_1, Y_1), (K_2, Y_2), ...\} \quad (3) \]

– also called Lagrange data

– The error bound is proportional to the squared distance between nodes

– A function with a lot of curvature requires a ‘fine’ grid
• Many economic variable must satisfy sign restrictions: capital stock, government purchases, price dispersion
  
  – For strictly positive state variables, use log transformation (i.e. \( \log(K_0), \log(K_1) \))
  
  – For state variables between 0 and 1: logistic transformation

• Various methods include least squares approximation, Lagrange polynomials, orthogonal polynomials,...
  
  – Judd (1998) is a key reference
  
  – de Boor (1978) and subsequent work to go in-depth
Piecewise linear interpolation

- ‘Connecting the dots’

- Benefit: easy to control when extrapolating off the grid

- Downside: can be very inaccurate for highly nonlinear models, especially when extrapolating

- Simple formula

\[
f(K) = Y_i + \frac{K - K_i}{K_{i+1} - K_i} (Y_{i+1} - Y_i) \quad \text{(4)}
\]

for \( K \in [K_i, K_{i+1}] \)
• Approximation error

\[
\left| \hat{f}(x) - f(x) \right| \leq M (x_b - x_a)^2
\]

(5)

\[
M = \frac{1}{8} \max_{x \in (x_a, x_b)} f''(x)
\]

(6)

• For example,

\[
\Omega_K = \{0, 2, 4, 6\}
\]

(7)

• Nodes should be monotonic, not necessarily evenly-spaced

• Must also supply Y values so

\[
\Omega_Y = \left\{ 0, \sqrt{2}, \sqrt{4}, \sqrt{6} \right\}
\]

(8)

• In Matlab :
  - Define nodes (i.e. state-space)

\[
k\_grid = \text{kmin} : \text{kstep} : \text{kmax}
\]

(9)
\begin{itemize}
  \item Provide values for $f(K)$
  \[ y_{\text{grid}} = \sqrt{\text{k}_{\text{grid}}} \] \hfill (10)
  \item Interpolation for any $K \in [K_0, K_n]$ is
  \[ Y = \text{interp1}(k_{\text{grid}}, y_{\text{grid}}, K) \] \hfill (11)
  \item Linear extrapolation for $K \notin [K_0, K_n]$ is
  \[ Y = \text{interp1}(k_{\text{grid}}, y_{\text{grid}}, K, 'linear', 'extrap') \] \hfill (12)
  \item ‘linear’ can be replaced with
    \begin{itemize}
      \item ‘cubic’ : piecewise cubic Hermite interpolation
      \item ‘spline’ : cubic spline interpolation
    \end{itemize}
\end{itemize}
Piecewise Cubic Hermite interpolation

- Uses level and slope information to formulate an approximation between each node

- ‘Pieces’ connect at the nodes and the approximation is $C^1$ everywhere (first derivatives agree at nodes)
Cubic splines

- Similar to piecewise cubic Hermite interpolation, but is smooth at the nodes where the pieces connect.

- The approximation is $C^2$ at the interior nodes (first and second derivatives agree at nodes)

- Must impose conditions on behavior near the boundaries of the grid

- Monotone data can lead to nonmonotone approximations
Multidimensional Approximation

- Add a productivity shock,

\[ Y = Z \sqrt{K} \quad (13) \]

now \( f : \mathbb{R}^2 \rightarrow \mathbb{R}^+ \)

- Must define grid for the productivity shock

\[ \Omega_Z = \{Z_0, Z_1, Z_2, ..., Z_m\} \quad (14) \]

- Need to define the state as

\[ \Omega = \{(K_0, Z_0), (K_0, Z_1), (K_0, Z_2), ... \]
\[ (K_1, Z_0), (K_1, Z_1), (K_1, Z_2), ... \]
\[ : \]
\[ (K_n, Z_0), (K_n, Z_1), (K_n, Z_2), ... \} \]
In Matlab

- nested for loops is the most straight-forward way

- interpolation for any $K$ in 2-dimensions
  \[
  Y = \text{interp2}(k\_grid, z\_grid, y\_grid, K, Z)
  \]

- generalizing to n-dimensional interpolation
  \[
  Y = \text{interpn}(k\_grid, z\_grid, x1\_grid, x2\_grid, ... , y\_grid, K, Z, x1, x2, ...)
  \]
Extrapolation

• Extrapolation approximates a function outside the range of the state space (i.e. ‘outside of the grid’)

• Extrapolation is necessary when evaluating expectations
  
  – Cannot restrict endogenous state variables (e.g. capital stock, price dispersion, debt, real balances, habit persistence etc.) to lie exactly on the grid

• Specify grid to minimize the need for extrapolation

• Poor extrapolation grind iterative algorithms to a halt
• The extent to which extrapolation is needed depends on the method of numerical integration used to evaluate expectations

• A well-specified state-space requires extrapolation on few relatively ‘unlikely’ states
Numerical Integration

• Numerical Integration (basically synonymous with quadrature)

• Judd (1998), Chapter 7 gives details on the methods

• Numerical integration necessary to solve for expectations

• General approach

\[
\int_{a}^{b} f(x) \, dx \approx \sum_{i=1}^{m} \omega_i f(x_i)
\]  

(15)

• The various methods provide ways of selecting
  
  - \( \omega_i \) : quadrature weights
  
  - \( x_i \in [a, b] \) : quadrature nodes
• Approximate the value of an integral using

1. Newton-Cotes formulas - evaluate the integrand over uniformly spaced nodes on \([a, b]\)

2. Gaussian formulas - nodes and weights are more carefully chosen

• Change of variable methods can increase speed and accuracy
  – However, a change of variable can result in a lot of extrapolation

• Working on 2 different grids

  1. State-space (capital, technology, price dispersion, etc.)

  2. Quadrature nodes
Newton-Cotes Approach

• Trapezoid rule
  
  – Uses uniformly spaced nodes
  
  – Example: Compute $E[Y_{t+1}|Z_t]$ where

  $Y_t = Z_t \sqrt{K},$ \hspace{1cm} (16)

  $\log Z_t = \rho \log Z_{t-1} + e_t,$ \hspace{1cm} (17)

  where $e_t \sim TN(0, \sigma_e^2)$ and $e_t \in [-\lambda \sigma_e, \lambda \sigma_e]$

  – Choice of $\lambda$ important

  – ‘Large’ $\lambda$ will result in larger extrapolations when we move to the case with endogenous $K$
– Uses a linear piecewise approximation

\[ E [Y_{t+1}|Z_t] = \int_{-\lambda \sigma_e}^{\lambda \sigma_e} \phi (e) f(\rho \log Z_t + e)de, \quad (18) \]

\[ \approx \frac{h}{2} \left[ \phi (e_0) f (\rho \log Z_t + e_0) + 2\phi (e_1) f (\rho \log Z_t + e_1) + \cdots + 2\phi (e_{m-1}) f (\rho \log Z_t + e_{m-1}) + \phi (e_m) f (\rho \log Z_t + e_m) \right] \quad (19) \]

where

\[ h = \frac{2\lambda \sigma_e}{m} \quad (20) \]

\( \phi \) is the normal density and \( e_i = -\lambda \sigma_e + ih \)

– Simpson’s rule is very similar, using a piecewise quadratic approximation
– Both the Trapezoid and Simpson method are slow, since $m$
needs to be large to obtain a satisfactory approximation

• Gaussian formulas are smart about specify nodes and weights
  – Gauss-Hermite is very useful for models with normally
distributed shocks
Monotone Map Overview

- Monotone map is a functional iteration method that computes the equilibrium (i.e. policy functions) as the limit of a monotone sequence of approximating functions

- Objective to approximate policy function(s) on a discrete grid
  - Policy functions map exogenous state variables to endogenous variables
  - Policy function(s) satisfy first-order necessary conditions
  - Transversality conditions can be checked, but not imposed
The Monotone Map Approach

1. Specify the state-space and an initial SWAG (i.e. scientific wild guess) for the solution
   - State space should bracket the deterministic steady state
   - Use a known analytic solution from a simpler version of your model
   - Use intuition on how variables relate to construct guess
   - Use simpler model in Dynare to get a starting guess

2. Calibrate
3. Apply functional iteration to generate a sequence of approximating functions, requires
   - Interpolation / Extrapolation
   - Numerical integration
   - Non-linear equation solving

4. Check sensitivity of solution to state-space and number of quadrature nodes

5. Compute impulse responses, moments, etc.
The Monotone Map: Example

- Note the model is solved in levels

- The basic neoclassical growth model

\[
\max E_t \sum_{t=0}^{\infty} \beta^t \frac{C_t^{1-\sigma}}{1-\sigma} \tag{21}
\]

subject to

\[
c_t + k_{t+1} = Z_t K_t^\alpha + (1 - \delta) K_t \tag{22}
\]

\[
\log Z_t = (1 - \rho) \log \bar{Z} + \rho \log Z_{t-1} + u_t \tag{23}
\]

where \(K_0 > 0\) given.
First-order necessary condition

\[ C_t^{-\sigma} = \beta E_t \left[ (1 + r_{t+1}) C_{t+1}^{-\sigma} \right], \tag{24} \]

where

\[ (1 + r_{t+1}) = Z_t \alpha K_{t+1}^{\alpha - 1} + 1 - \delta \tag{25} \]

can be expressed as the nonlinear expectational difference equation

\[
\begin{align*}
(K_{t}^{\alpha} + (1 - \delta) K_{t} - K_{t+1})^{-\sigma} &= \\
\beta E_t \left[ (Z_t \alpha K_{t+1}^{\alpha - 1} + 1 - \delta) (K_{t+1}^{\alpha} + (1 - \delta) K_{t+1} - K_{t+2})^{-\sigma} \right]
\end{align*}
\]
• What does a solution look like? A function mapping the state 
\((Z_t, K_t)\) into \(K_{t+1}\)

\[ K_{t+1} = h (Z_t, K_t) \]  \hspace{1cm} (26)

• Substitute into nonlinear expectational difference equation

\[
\left( K_t^\alpha + (1 - \delta) K_t - K_{t+1} \right)^{-\sigma} = \\
\beta E_t \left[ (Z_t^\alpha K_{t+1}^{\alpha-1} + 1 - \delta) \left( K_{t+1}^\alpha + (1 - \delta) K_{t+1} - h (Z_{t+1}, K_{t+1}) \right)^{-\sigma} \right]
\]

noting that

\[ K_{t+2} = h (Z_{t+1}, K_{t+1}) \]  \hspace{1cm} (27)

• Interpolation/extrapolation is necessary when evaluating

\( h (Z_{t+1}, K_{t+1}) \)

• Given \((K_t, Z_t)\) we have one equation in one unknown \(K_{t+1}\)
  
  – Use a nonlinear equation solver (Chris Sims’ csolve.m is very good)
• Handling the expectation operator with the trapezoid rule

\[
\beta E_t \left[ (1 + r_{t+1}) (Z_{t+1} K_{t+1}^\alpha + (1 - \delta) K_{t+1} - h (Z_{t+1}, K_{t+1}))^{-\sigma} \right] \\
= \frac{h}{2} \left[ \phi(e_1) \left( (\rho Z_t + e_1) \alpha K_{t+1}^{\alpha-1} + 1 - \delta \right) \right. \\
\left. \left( (\rho Z_t + e_1) K_{t+1}^\alpha + (1 - \delta) K_{t+1} - h (\rho Z_t + e_1, K_{t+1}) \right)^{-\sigma} \right. \right. \\
\left. \left. \phi(e_2) \left( (\rho Z_t + e_2) \alpha K_{t+1}^{\alpha-1} + 1 - \delta \right) \right. \\
\left. \left. \left( (\rho Z_t + e_2) K_{t+1}^\alpha + (1 - \delta) K_{t+1} - h (\rho Z_t + e_2, K_{t+1}) \right)^{-\sigma} \right. \right. \right. \\
\left. \left. \vdots \right. \\
\left. \left. \phi(e_m) \left( (\rho Z_t + e_m) \alpha K_{t+1}^{\alpha-1} + 1 - \delta \right) \right. \\
\left. \left. \left( (\rho Z_t + e_m) K_{t+1}^\alpha + (1 - \delta) K_{t+1} - h (\rho Z_t + u_m, K_{t+1}) \right)^{-\sigma} \right. \right] \\
\]
• Convergence

  – Maximum update of the policy function at any grid point is $< \varepsilon$. That is

  $$|h^N(K_i, Z_j) - h^{N-1}(K_i, Z_j)| < \varepsilon$$

  for every $i \in \{1, 2, .., n\}$ and $j \in \{1, 2, .., m\}$.
• Adding elastic labor supply

\[
\max E_t \sum_{t=0}^{\infty} \beta^t \log C_t + \theta \log (1 - N_t)
\]

subject to

\[
c_t + k_{t+1} = Z_t K_t^\alpha N_t^{1-\alpha} + (1 - \delta) K_t
\]

\[
\log Z_t = (1 - \rho) \log \bar{Z} + \rho \log Z_{t-1} + u_t
\]

First-order conditions now include

\[
\frac{\theta C_t}{(1 - N_t)} = (1 - \alpha) \frac{Y_t}{N_t},
\]

which is a static relation, so \( N_t \) can be solved for given the state \((K_t, Z_t)\) and existing approximation \( h(Z_{t+1}, K_{t+1}) \)
Calibration

- Must specify functional forms and numerical values
  \[ \alpha = .33 \quad \delta = .025 \quad \beta = .99 \quad n = .2 \]

- Solve \( \theta \) and \( k \)

\[
(1 + \alpha K^{\alpha-1} N^{1-\alpha} - \delta) = \beta^{-1} \\
\frac{\theta}{1 - N} = \frac{(1 - \alpha) K^{\alpha} N^{-\alpha}}{K^{\alpha} N^{1-\alpha} - \delta K}
\]

Solution is: \( \{k = 5.6697, \theta = 3.5034\} \)
Extensions

- Switching levels of productivity

\[
\log Z_t = (1 - \rho) \log Z(s_t) + \rho \log Z_{t-1} + u_t
\]

where \( s_t \in \{0, 1\} \), \( Z(0) < Z(1) \) ans \( s_t \) evolves according to

\[
\Pi = \begin{bmatrix}
    p & 1 - p \\
    1 - q & q
\end{bmatrix}
\]
• Handling the expectation operator for $s_t = 0$

$$
\beta E_t \left[ R_{t+1} \left( Z_{t+1} K_{t+1}^{\alpha} N_{t+1}^{1-\alpha} + (1 - \delta) K_{t+1} - h (Z_{t+1}, K_{t+1}) \right)^{-\sigma} \right]
$$

$$
= \rho \beta \int_a^b \phi (e) R_{t+1} \left( \left( Z_0 + e \right) K_{t+1}^{\alpha} N_{t+1}^{1-\alpha} + \left( 1 - \delta \right) K_{t+1} - h \left( Z_0 + e, K_{t+1} \right) \right)^{-\sigma} de +
$$

$$
(1 - \rho) \beta \int_a^b \phi (e) R_{t+1} \left( \left( Z_1 + e \right) K_{t+1}^{\alpha} N_{t+1}^{1-\alpha} + \left( 1 - \delta \right) K_{t+1} - h \left( Z_1 + e, K_{t+1} \right) \right)^{-\sigma} de
$$

where $R_{t+1} = (1 + r_{t+1})$ and $\rho = 0$

• Can extend to additional number of regimes

• Can extend to allow other parameters to vary (i.e. preference shocks, degree of serial correlation, etc.)
Reducing Computation Time

- Efficiency of numerical routines
  - At first, best to use methods that you understand, even if they are slow
  - To speed things up
    * Economize on grid space
    * Use Gaussian quadrature
    * ‘Vectorize’ integration routine
    * Experiment with various interpolation methods
    * Use Fortran
• Speed is very important for estimation
  
  – Monotone map can be used for estimation by using the particle filter (hasn’t been done...)

  – See Villaverde and Rubio-Ramirez for details on estimation methods using the particle filter