Non-Linear Approximation of the One-Tree Model

David N. DeJong
University of Pittsburgh

Spring 2008
Overview

We seek a policy function of the form

\[ p_t = \hat{p}(d_t, q_t) \]

corresponding to the functional equation

\[ p_t = \beta e^{(1-\gamma)g} E_t \left[ \left( \frac{c_{t+1}}{c_t} \right)^{-\gamma} (d_{t+1} + p_{t+1}) \right], \]

where

\[ c_t = d_t + q_t \]
\[ d_t = \bar{d} e^{u_{dt}}, \quad u_{dt} = \rho_d u_{dt-1} + \varepsilon_{dt} \]
\[ q_t = \bar{q} e^{u_{qt}}, \quad u_{qt} = \rho_q u_{qt-1} + \varepsilon_{qt}. \]
Specifically, the policy function we seek is a Chebyshev polynomial of the form

\[
\hat{p}(d_t, q_t) = \sum_{i_d=0}^{r_d} \sum_{i_q=0}^{r_q} \chi_{i_d i_q} P_{i_d i_q}(\tilde{d}, \tilde{q}),
\]

\[
P_{i_d i_q}(\tilde{d}, \tilde{q}) = p_{i_d}(\tilde{d}) p_{i_q}(\tilde{q}),
\]

\[
\tilde{\chi} = \frac{x - x^*}{\omega_x x^*}, \quad x = d, q.
\]
Overview, cont.

We will select the $\chi's$ as the unique values that satisfy

$$F(p(\tilde{d}_i, \tilde{q}_j, \chi)) = 0, \quad i = 1, ..., r_d, \quad j = 1, ..., r_q,$$

where

$$\tilde{d}_i = -\cos\left(\frac{(2i - 1) \pi}{r_d} \right), \quad i = 1, 2, ..., r_d,$$

and likewise for $\tilde{q}_j$. 
Initialization

nstates = 2;  // # of state variables
let ord[2,1] = 4 3;  // order of polys specified for d, q
gams = prodc(ord);
Initialization, cont.

```matlab
zeros = zeros(maxc(ord),nstates);
// locations of zeros of the poly for each state variable

iii = 1; do while iii <= nstates;
    zeros[1:ord[iii],iii] = zeropoly(ord[iii]);
    iii = iii+1; endo;
```
Initialization, cont.

proc zeropoly(order);

   // locate zeros of chebyshev polynomials of order 'order'
   local ord, zeros, iter;
   ord = order;
   zeros=zeros(ord,1);
   iter=1;do while iter<=ord;
      zeros[iter]=-1.0*cos((2*iter-1)*PI/(2*ord));
   iter=iter+1;endo;
retp(zeros);
endp;
Initialization, cont.

With ord $= [4, 3]'$, zeros is

-0.92387953    -0.86602540  
-0.38268343   -6.1230e-017  
0.38268343     0.86602540  
0.92387953     0.00000000  

We will construct $\chi$ to satisfy the functional equation at the 12 possible combinations of $[\wedge d, \wedge q]$ contained in zeros.
Establish Starting Values for Optimization Routine

Recall that from our log-linear approximation, we can construct an approximation in levels of the form

\[
p \approx p^* + \frac{p^*}{d^*} \sigma_d (d - d^*) + \frac{p^*}{q^*} \sigma_q (q - q^*) + \frac{1}{2} \left( \frac{p^*}{d^*} \sigma_d \right) \left( \frac{p^*}{q^*} \sigma_q \right) (d - d^*)(q - q^*).
\]
Starting Values, cont.

The corresponding approximation of $\hat{p}(\tilde{d}, \tilde{q}, \chi)$ we seek is of the form

$$
\hat{p}(d, q, \chi) \approx \chi_{11} + \chi_{12} \left( \frac{d - d^*}{\omega_d} \right) + \chi_{21} \left( \frac{q - q^*}{\omega_q} \right) \\
+ \chi_{22} \left( \frac{d - d^*}{\omega_d} \right) \left( \frac{q - q^*}{\omega_q} \right) + \ldots
$$

Matching terms yields the suggested starting values

$$
\chi_{11} = p^*, \quad \chi_{12} = \sigma_d \omega_d \frac{p^*}{d^*}, \quad \chi_{21} = \sigma_q \omega_q \frac{p^*}{q^*}, \\
\chi_{22} = \frac{1}{2} \left( \sigma_d \omega_d \frac{p^*}{d^*} \right) \left( \sigma \omega_q \frac{p^*}{q^*} \right).
$$
Non-Lin. Approx. of the One-Tree Model

Overview

Initialization of Chebyshev Polynomial

Establish Starting Values

Calculating Expectations

Programming the Functional Equation

Programming the Policy Function

Call the Procedure

Results

Starting Values, cont.

\[
\text{omegad} = 4 \times \text{stdx}[3,1] \times \text{ss}[3];
\]

// stdx is the stddev of logged dev. from ss. mult by xbar converts to levels

\[
\text{omegaq} = 4 \times \text{stdx}[4,1] \times \text{ss}[4];
\]

\[
\text{sigd} = (1/p[6]) \times f[1,3];
\]

// adjustment by 1/rhod converts from lagged to contemporaneous elasticity

\[
\text{sigq} = (1/p[7]) \times f[1,4];
\]
Starting Values, cont.

```matlab
startval = zeros(ngams,1);
//setup starting values for non-linear eqn solver
startval[1] = ss[1];
startval[2] = sigd*omegad*(ss[1]/ss[3]);
startval[ord[1]+2] = 0.5*omegad*(ss[1]/ss[3])*omegaq*(ss[1]/ss[4]);
```
Starting Values, cont.

```
startval

18.084250
1.8154179
0.00000000
0.00000000
1.8795104
2.7540218
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
```

Non-Lin. Approx. of the One-Tree Model

DND

Overview

Initialization of Chebyshev Polynomial

Establish Starting Values

Calculating Expectations

Programming the Functional Equation

Programming the Policy Function

Call the Procedure

Results
Given the presence of two correlated stochastic processes in the functional equation, we will approximate expectations via Monte Carlo integration. Critically, common random numbers are used for this purpose.
// prepare for mc integration: draw epsd epsq pairs

sqrtvmat = chol(vcvmat)';
ndraws = 10000;
draws = zeros(ndraws,2);
niii=1; do while iii<=ndraws;
    draws[iii,..] = (sqrtvmat*rndn(2,1))';
iii=iii+1; endo;
Programming the Functional Equation

```plaintext
proc feval(gam);
    // functional equation used to construct optimal gamma vector
    local blahblahblah;
    f=zeros(ngams,1);
    // will contain values of functional equation

    // statemat will contain all possible combinations of state variables
    iii = nstates; do while iii>=2;
        if iii==nstates;
            statemat = combinestates(zers[1:ord[iii-1],iii-1],zers[1:ord[iii],iii]);
        else;
            statemat = combinestates(zers[1:ord[iii-1],iii-1],statemat);
        endif;
    iii=iii-1; endo;
```
Functional Equation, cont.

proc combinestates(newstate,existingstates);

   // combines an existing matrix of state combinations with a new state vector to get an expanded
   // group of outcomes. used to recursively construct all possible combinations.
   local blahblahblah;
   ns = newstate;
   ex = existingstates;
   ordnew = rows(ns);
   ordex = rows(ex);
   expandmat = zeros(ordnew*ordex,cols(ex)+1);
   counter = 1;
   iii=1; do while iii<=ordnew;
      jjj=1; do while jjj<=ordex;
         expandmat[counter,.]
            = ns[iii]~ex[jjj,.];
         counter = counter+1;
         jjj=jjj+1; endo;
   iii=iii+1; endo;
   retp(expandmat);
iii=1; do while iii<=rows(statemat);
    stilde = statemat[iii,:]
    d = ss[3] + omegad*stilde[1];
    q = ss[4] + omegaq*stilde[2];
    pc = pc_of_dq(stilde,gam);
    pee = pc[1,1];
    c = pc[1,2];
Functional Equation, cont.

```plaintext
rhsvec = zeros(ndraws,1);
jjj = 1; do while jjj<=ndraws;
    lndp = onemrhod + p[6]*ln(d) + draws[jjj,1];
    lnqp = onemrhoq + p[7]*ln(q) + draws[jjj,2];
    dp = exp(lndp);
    qp = exp(lnqp);
    dptilde = (dp - ss[3])/omegad;
    qptilde = (qp - ss[4])/omegaq;
    pcp = pc_of_dq(dptilde,qptilde,gam);
    pp = pcp[1,1];
    cp = pcp[1,2];
    rhsvec[jjj] = (dp+pp)/(cp^p[2]);
    j jj = j jj + 1; endo;
```
Functional Equation, cont.

\[
\text{rhs} = \text{meanc(rhsvec)} \times p[1] \times \exp((1 - p[2]) \times p[3]);
\]

\[
f[iii] = \frac{\text{pee}}{c^{p[2]}} - \text{rhs};
\]

\[
iii = iii + 1; \text{ endo;}
\]

\[
\text{retp}(f);
\]

\[
\text{endp;}
\]
Programming the Policy Function

```plaintext
proc pc_of_dq(s,gam);
    // calculates p, c as functions of state
    local blahblahblah;
    iii=1; do while iii<=nstates;
        ordiii = ord[iii];
        ntees = ordiii;
        tees = zeros(ntees,1);
        tees[1] = 1;
        tees[2] = s[iii];
        if ordiii > 2;
            j=3; do while j<=ntees;
                tees[j] =
                    2*s[iii]*tees[j-1]-tees[j-2];
                j=j+1; endo;
        endif;
    endo;
```
if iii==1;
    statepolyvec = tees;
else;
    statepolyvec = vec(statepolyvec*tees');
endif;
iii=iii+1; endo;
Policy Function, cont.

plev = sumc(gam’statepolyvec);
dlev = ss[3] + omegad*s[1];
qlev = ss[4] + omegaq*s[2];
clev = dlev + qlev;
retp(plev~clev);
endp;
Call the Procedure

\[
\{ \text{gamopt,fopt,gopt,retcode} \} = \text{nlsys}(&\text{feval,startval});
\]
## Results

### Optimized Versus Starting Values:

<table>
<thead>
<tr>
<th>Optimized Value</th>
<th>Starting Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.129067</td>
<td>18.084250</td>
</tr>
<tr>
<td>1.8187939</td>
<td>1.8154179</td>
</tr>
<tr>
<td>0.0020094885</td>
<td>0.0000000000</td>
</tr>
<tr>
<td>0.0006370966</td>
<td>0.0000000000</td>
</tr>
<tr>
<td>1.8800581</td>
<td>1.8795104</td>
</tr>
<tr>
<td>0.092788218</td>
<td>2.7540218</td>
</tr>
<tr>
<td>-0.0018554837</td>
<td>0.0000000000</td>
</tr>
<tr>
<td>8.5293814e-005</td>
<td>0.0000000000</td>
</tr>
<tr>
<td>0.022846908</td>
<td>0.0000000000</td>
</tr>
<tr>
<td>0.00031149878</td>
<td>0.0000000000</td>
</tr>
<tr>
<td>1.3264616e-006</td>
<td>0.0000000000</td>
</tr>
<tr>
<td>5.2236204e-006</td>
<td>0.0000000000</td>
</tr>
</tbody>
</table>
Results, cont.

Policy Functions and Slopes
Results, cont.

Fit

Non-Lin. Approx. of the One-Tree Model

DND

Overview

Initialization of Chebyshev Polynomial

Establish Starting Values

Calculating Expectations

Programming the Functional Equation

Programming the Policy Function

Call the Procedure

Results

Fit

Graphs showing the fit of the model with different curves.