# Policy Function Iteration, Parallel Processing, & an Introduction to MEX

Alexander W. Richter
Nathaniel A. Throckmorton

Indiana University

Computational Mini-Course
March 21-22, 2012

# Benchmark Model

Representative household chooses $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ to maximize expected lifetime utility,

$$E_t \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma}}{1-\sigma},$$

subject to

$$c_t + k_{t+1} = z_t k_t^{\alpha} + (1-\delta)k_t$$
$$z_{t+1} = (1-\rho)\bar{z} + \rho z_t + \varepsilon_{t+1},$$

where $k_0, z_0$ are given.

# Discretized State Space

- State variables: $k_t$, $z_t$
- Number of grid points: $N_k$, $N_z$
- Given grid boundaries: $[k_{\min}\ k_{\max}]$ and $[z_{\min}\ z_{\max}]$, set grids
- For evenly spaced grids use:

$$x_{grid} = \texttt{linspace}(x_{\min}, x_{\max}, N_x), \quad x \in \{k, z\}$$

- State space: contains $N = N_k \times N_z$ independent nodes
- To create an array for each state variable where every position is a unique permutation of the state space use:

$$[k_{gr}\ z_{gr}] = \texttt{ndgrid}(k_{grid}, z_{grid})$$

# Functional Approximation

- Goal: Find a function that maps the state space to the policy function

$$c_t = \underbrace{c(k_t, z_t)}_{\text{True RE Solution}} \approx \underbrace{\mathcal{P}(k_t, z_t, \eta)}_{\text{Approximating Function}}$$

  where $\eta$ is matrix of polynomial coefficients with $M$ elements

- Collocation Method: $M = N$
- Galerkin Method: $M < N$

# Fixed-Point Iteration

1. Given $\mathcal{P}(k_t, z_t, \eta^{q-1})$, solve for $k_{t+1}$ using the budget constraint
2. Given the updated state, evaluate $\mathcal{P}(k_{t+1}, z_{t+1}, \eta^{q-1})$
3. Use Euler equation to back out the implied policy function
4. Use a routine to minimize the approximating function errors over the entire state space (implies $\hat{\eta}^q$)
5. Set $\eta^q = (1-\lambda)\eta^{q-1} + \lambda\hat{\eta}^q$, $\lambda \in [0, 1]$
6. Follow iteration procedure until convergence

# Time Iteration

1. Use $\mathcal{P}(k_t, z_t, \eta^{q-1})$ as a guess for the current policy function, $c_t$
2. Solve for $k_{t+1}$ using the budget constraint
3. Given the updated state, evaluate $\mathcal{P}(k_{t+1}, z_{t+1}, \eta^{q-1})$
4. Use a nonlinear solver so $c_t$ satisfies the equilibrium system
5. Update approximating function
6. Follow iteration procedure until convergence

# Global Approximation: Monomial Basis

- Choose the set of monomials corresponding to an $n^{th}$ order Taylor approximation of the policy function
- Complete set of basis functions is:

$$\mathcal{B}_n^s = \left\{ \left(x_1^{k_1} \cdots x_s^{k_s}\right) \middle| \sum_{i=1}^{s} k_i = j, k_i \geq 0, j = 0, 1, \ldots, n \right\},$$

  where $s$ is the number of state variables
- The monomials are evaluated at each of the $N$ nodes in the discretized state-space and stored column-wise in $X$
- Obtain updated coefficients using OLS:

$$\hat{\eta}^q = (X'X)^{-1}X'\Lambda_t,$$

  where $\Lambda_t$ is a vector of policy function values.

# Monomial Basis Example

If $n = 2$, 2 state variables ($k$ and $z$) imply

- Complete Basis:

$$\mathcal{P}(k, z, \eta) = \eta_0 + \eta_k k + \eta_z z + \eta_{kk} k^2 + \eta_{kz} kz + \eta_{zz} z^2$$

- Regressor matrix (subscripts denote grid indices)

$$X = \begin{bmatrix} 1 & k_1 & z_1 & k_1^2 & k_1 z_1 & z_1^2 \\ 1 & k_2 & z_2 & k_2^2 & k_2 z_2 & z_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & k_N & z_N & k_N^2 & k_N z_N & z_N^2 \end{bmatrix}$$

- Obtain updated coefficients using OLS:

$$\hat{\eta}_c^q = (X'X)^{-1} X'\mathbf{c},$$

# Local Approximation

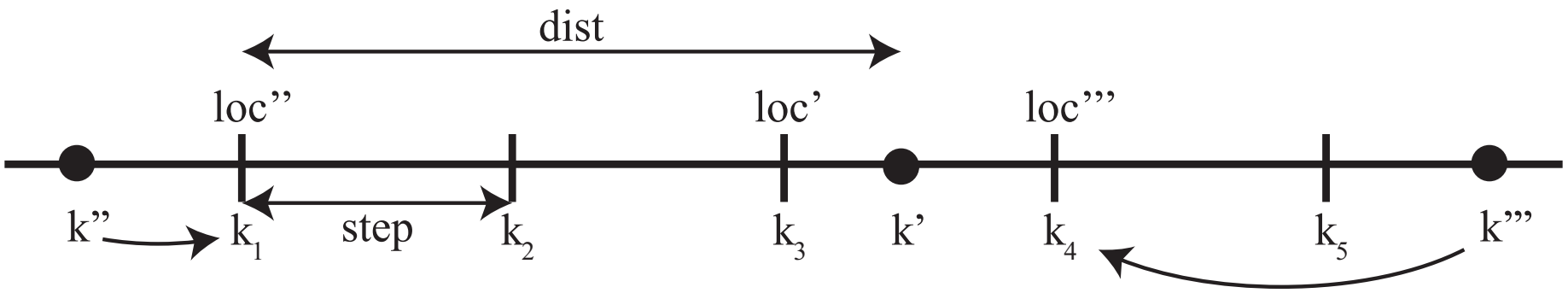
- Linear Interpolation: 2 state variables $(k, z)$
- Goal: Find the policy value $c(k', z')$
- We have policy function values on nearest nodes,

$$[c(k_i, z_j), c(k_i, z_{j+1}), c(k_{i+1}, z_j), c(k_{i+1}, z_{j+1})],$$

  once we determine the grid indices, $i, j$
- Locate grid point to left of $k'$ (evenly spaced nodes)

$$\text{step} = k_2 - k_1, \qquad \text{dist} = k' - k_1$$

$$\text{loc} = \min(N_k - 1, \max(1, \text{floor}(\text{dist}/\text{step}) + 1))$$



# Local Approximation

- Interpolate in the $k$ direction to obtain

$$\begin{aligned} c(k', z_j) &= c(k_i, z_j) + (k' - k_i)\frac{c(k_{i+1}, z_j) - c(k_i, z_j)}{k_{i+1} - k_i} \\ &= \underbrace{\frac{k_{i+1} - k'}{k_{i+1} - k_i}}_{\omega_{k_i}} c(k_i, z_j) + \underbrace{\frac{k' - k_i}{k_{i+1} - k_i}}_{\omega_{k_{i+1}}} c(k_{i+1}, z_j) \end{aligned}$$

- Then interpolate in the $z$ direction to obtain

$$c(k', z') = \underbrace{\frac{z_{j+1} - z'}{z_{j+1} - z_j}}_{\omega_{z_j}} c(k', z_j) + \underbrace{\frac{z' - z_j}{z_{j+1} - z_j}}_{\omega_{z_{j+1}}} c(k', z_{j+1}).$$

- Combine these two equations

$$c(k', z') = \sum_{a=0}^{1} \sum_{b=0}^{1} \omega_{k_{i+a}} \omega_{z_{j+b}} c\left(k_{i+a}, z_{j+b}\right),$$

# FORTRAN/MEX ADVANTAGE

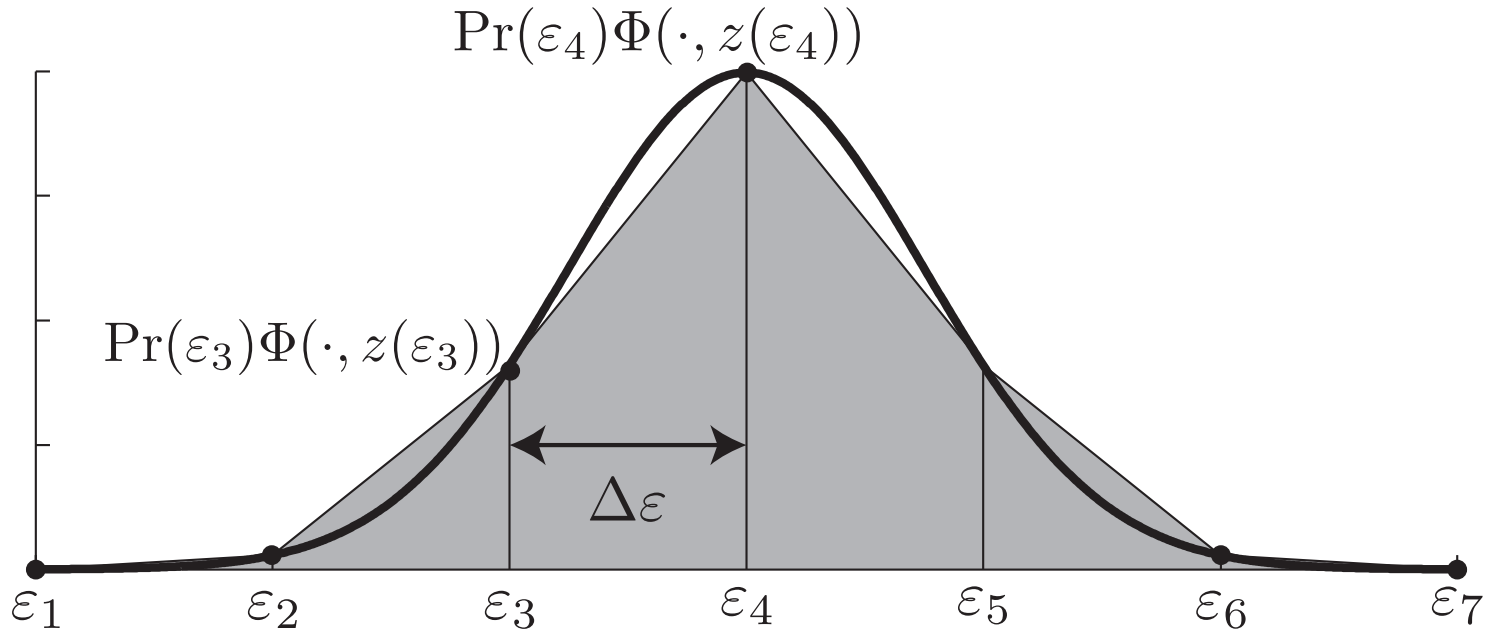
- Use a nested loop to calculate the nested sum

```
nestedsum = 0;  %initialize
for a = 0:1     %loop for k
  for b = 0:1   %loop for z
    nestedsum = nestedsum + ...
      wk(i+a)*wz(j+b)*c(i+a,j+b);
  end
end
```

- Number of nested loops corresponds to the number of state variables
- Since these are unavoidable, compiled code is faster

# Integration: Trapezoid Rule

$$
\begin{aligned}
E[\Phi(\cdot,z)] \approx & \frac{\Pr(\varepsilon_1)\Phi(\cdot,z(\varepsilon_1))+\Pr(\varepsilon_2)\Phi(\cdot,z(\varepsilon_2))}{2}\Delta\varepsilon \\
& + \frac{\Pr(\varepsilon_2)\Phi(\cdot,z(\varepsilon_2))+\Pr(\varepsilon_3)\Phi(\cdot,z(\varepsilon_3))}{2}\Delta\varepsilon + \cdots \\
& + \frac{\Pr(\varepsilon_{m-1})\Phi(\cdot,z(\varepsilon_{m-1}))+\Pr(\varepsilon_m)\Phi(\cdot,z(\varepsilon_m))}{2}\Delta\varepsilon \\
& = \frac{\Delta\varepsilon}{2}\left[2\sum_{i=1}^{m}\Pr(\varepsilon_i)\Phi(\cdot,z(\varepsilon_i)) - \Pr(\varepsilon_1)\Phi(\cdot,z(\varepsilon_1)) - \Pr(\varepsilon_m)\Phi(\cdot,z(\varepsilon_m))\right]
\end{aligned}
$$



# INTEGRATION: GAUSS-HERMITE

- Given shock, $\varepsilon \sim N(\mu, \sigma^2)$,

$$E[\Phi(\cdot, z(\varepsilon))] = (2\pi\sigma^2)^{-1/2} \int_{-\infty}^{\infty} \Phi(\cdot, z(\epsilon)) e^{-(\varepsilon-\mu)^2/(2\sigma^2)} d\epsilon$$

- Apply change of variables, $u = (\epsilon - \mu)/(\sqrt{2}\sigma)$,

$$E[\Phi(\cdot, z(u))] = \pi^{-1/2} \int_{-\infty}^{\infty} \Phi(\cdot, z(\sqrt{2}\sigma u + \mu)) e^{-u^2} du$$

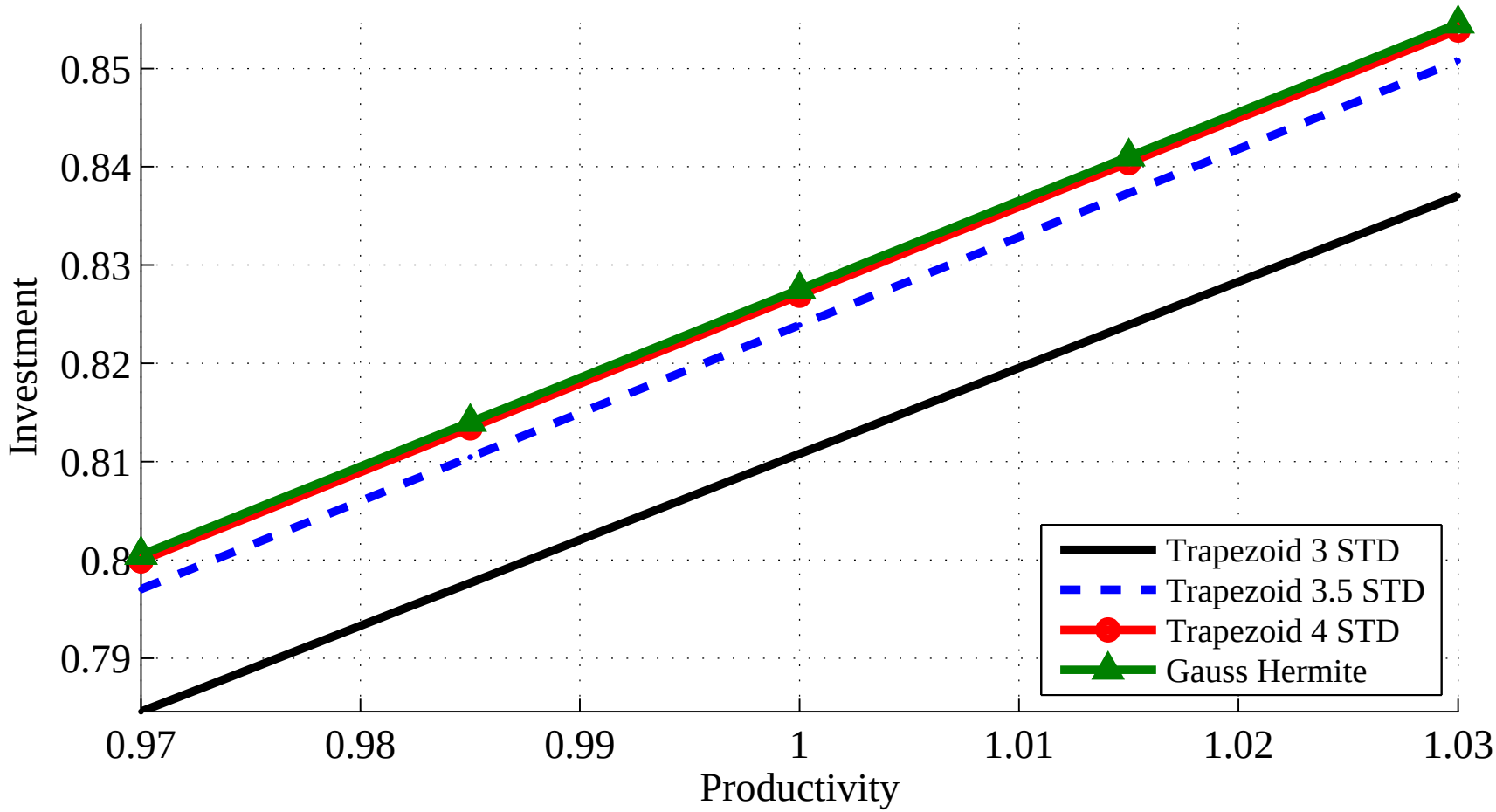
$$\approx \pi^{-1/2} \sum_{i=1}^{n} \omega_i \Phi(\cdot, z(\sqrt{2}\sigma u_i + \mu)),$$

- $\omega_i$ are Gauss-Hermite weights

$$\omega_i = 2^{n+1} n! \sqrt{\pi} [H_{n+1}(u_i)]^{-2}$$

$H_{n+1}$ is the physicists' Hermite polynomial of order $n + 1$.

# Integration Comparison



# Obtaining a Guess

Find log-linear solution on the state space as a guess for $c(k, z)$

- Log-linear solution from `gensys` takes the form,

$$\hat{Y}_{t+1} = T\hat{Y}_t + M\varepsilon_{t+1},$$

  where $\hat{Y}_t = [\hat{k}_t, \hat{z}_t, \hat{c}_{t-1}]'$ and $\varepsilon \sim N(0, \sigma^2)$.
- Convert state space in levels to percent deviations from steady state
- Compute solution for $i = 1, \ldots, N$

$$\hat{c}_i = T(c_{loc}, [k_{loc}\ z_{loc}])[\hat{k}_{gr}(i)\ \hat{z}_{gr}(i)]'$$

- Convert $\hat{\mathbf{c}}$ to levels

# Time Iteration Example

Solve for an updated $c(k, z)$ that *satisfies* equilibrium system

1. Use log-linear solution as a guess on each node
   - If global approximation, then calculate $\hat{\eta}^0 = (X'X)^{-1}X'\mathbf{c}$
2. Solve for $k_{t+1}, z_{t+1}$
3. Find $c_{t+1}$ given the updated state
   - If global approximation, update the basis. Then $c_{t+1} = X_{t+1}\eta^{q-1}$
   - If local approximation, then use linear interpolation
4. Evaluate the expectation (Trapezoid rule or GH), where

$$\Phi(\varepsilon_i) = \beta(c_{t+1}(\varepsilon_i))^{-\sigma}(\alpha z_{t+1}(\varepsilon_i)k_{t+1}^{\alpha-1} + 1 - \delta)$$

# Time Iteration Example

5. Use nonlinear solver to find a $c(k, z)$ that satisfies the consumption Euler equation, $c_t^{-\sigma} = E[\Phi]$.
6. Update policy function
   - Global: $\hat{\eta}^q = (X'X)^{-1}X'\mathbf{c}$.
   - Local: use outputted policy function
7. Calculate distance between updates
   - Global: $\text{dist} = \eta^q - \eta^{q-1}$
   - Local: $\text{dist} = c^q(k, z) - c^{q-1}(k, z)$
8. If $|\text{dist}| < \text{tol}$, then stop. If not, repeat algorithm using the new policy function as the starting guess.

Advantages: Satisfies the equilibrium system on each node and nodes can be run in parallel.

Disadvantage: Non-linear solver must execute on each node.

# Fixed-Point Iteration Example

Solve for an updated $c(k,z)$ *implied* by the equilibrium system

1. Use log-linear solution as a guess on each node
   - If global approximation, then calculate $\hat{\eta}^0 = (X'X)^{-1}X'\mathbf{c}$. Calculate $c(k,z) = X\hat{\eta}^0$.
2. Calculate updated variables and the expectation (steps 2-4 in the time iteration algorithm)
3. Solve for the implied policy, $c_t = (E[\Phi])^{-1/\sigma}$, on each node
4. Execute steps 6-8 in the time iteration algorithm

Advantage: Does not require a loop, since all of the nodes are evaluated simultaneously.

Disadvantage: Since resulting $c(k,z)$ does not satisfy equilibrium system, algorithm is less stable.

# Extension 1: Elastic Labor Supply

Household now chooses $\{c_t, n_t, k_{t+1}\}_{t=0}^{\infty}$ to maximize:

$$E_t \sum_{t=0}^{\infty} \beta^t \left\{ \frac{c_t^{1-\sigma}}{1-\sigma} - \chi \frac{n_t^{1+\eta}}{1+\eta} \right\},$$

subject to

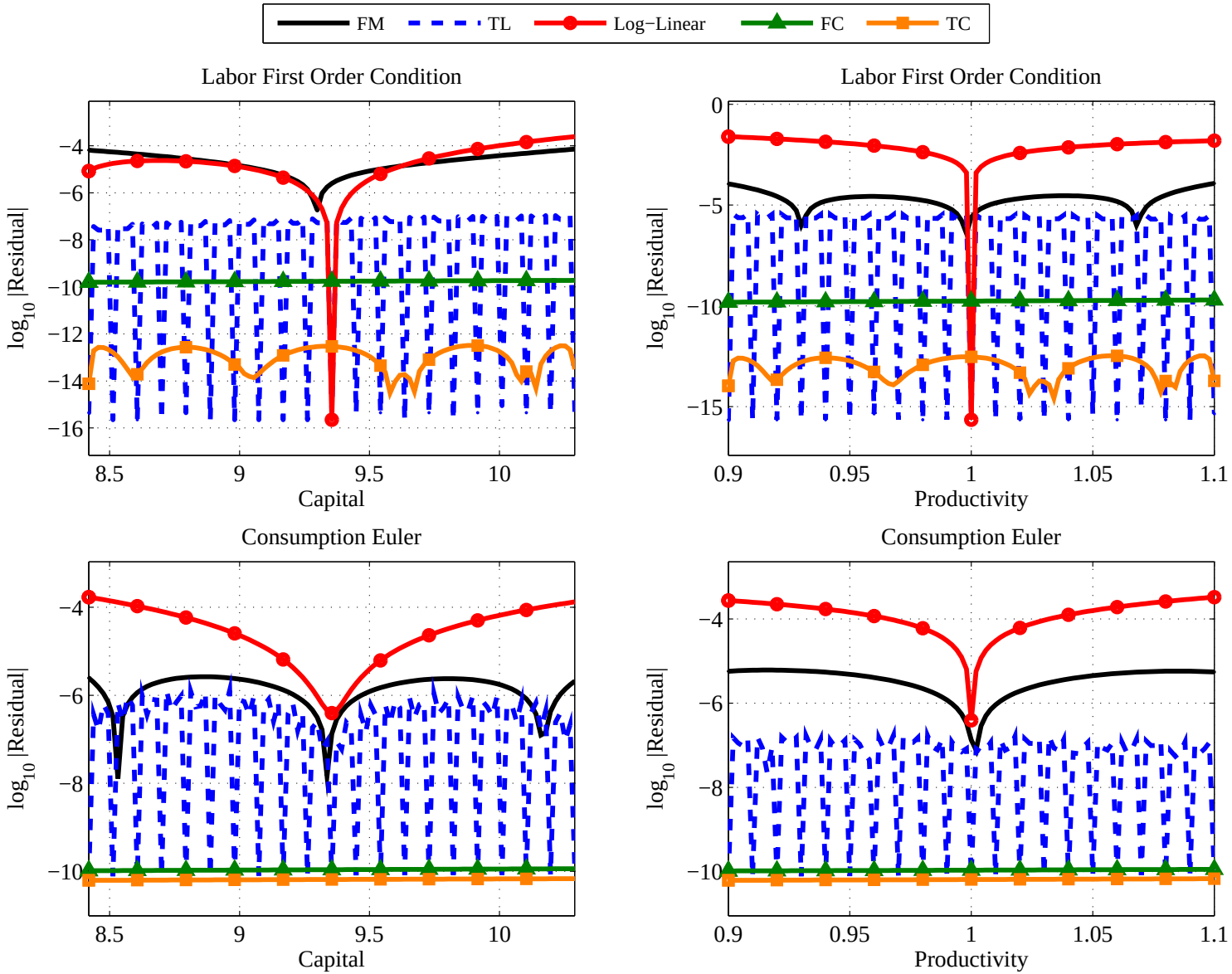
$$c_t + k_{t+1} = z_t k_t^{\alpha} n_t^{1-\alpha} + (1-\delta)k_t$$
$$z_{t+1} = (1-\rho)\bar{z} + \rho z_t + \varepsilon_{t+1}$$

Optimality conditions now includes:

$$(1-\alpha)y_t/n_t = \chi n_t^{\eta} c_t^{\sigma}$$

New optimality condition is a static relation and does not add a state variable, but may require a new policy function: $n(k_t, z_t)$

# Euler Equation Errors



# Extension 2: Productivity Switching

Productivity evolves according to:

$$z_{t+1} = (1-\rho)\bar{z}(s_{t+1}) + \rho z_t + \varepsilon_t,$$

where $s_t \in \{1, 2\}$, $z(1) < z(2)$, and

$$\begin{bmatrix} \Pr[s_{t+1}=1|s_t=1] & \Pr[s_{t+1}=2|s_t=1] \\ \Pr[s_{t+1}=2|s_t=1] & \Pr[s_{t+1}=2|s_t=2] \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix},$$

where $0 \leq p_{ij} \leq 1$ and $\sum_{j=1}^{2} p_{ij} = 1$ for all $i \in \{1, 2\}$

Key Changes:

- Policy functions are dependent on a discrete state variable
- Policy functions account for expectational effects of alternative regimes

# SOLVING THE MODEL

- Calculate $z_{t+1}(s_{t+1})$ for each realization of the productivity state
- Find $c_{t+1}(s_{t+1})$ and $n_{t+1}(s_{t+1})$ based on each realization of the productivity state
- Numerical Integration:
  - First integrate across the continuous random variable, $\varepsilon$, to obtain expectations, conditional on the realizations of the discrete stochastic variable at time $t+1$. This yields:

$$E[\Phi|s_{t+1}=1], \quad E[\Phi|s_{t+1}=2]$$

  - Then weight each of these values by their corresponding likelihood. Conditional expectations are:

$$E_t\left[\Phi|s_t=i\right] = p_{i1}E[\Phi|s_{t+1}=1] + p_{i2}E[\Phi|s_{t+1}=2]$$

# Introduction to MEX

- Advantages of MATLAB:
  - Many built-in functions with good documentation
  - Parallel computing toolbox (PCT) makes parallel processing easy to implement
  - Very easy to debug code
- Main drawback: Interpretive language is very slow at evaluating loops
- MEX (MATLAB Executable) functions: Allow programmers to write sections of the code using a compiled language and import it into MATLAB
- Challenge: Users must write a “Gateway” function that allows MATLAB to communicate with compiled code

# FORTRAN 90 MEX REQUIREMENTS

- Intel Visual Fortran Composer XE 2011
  - Basic compiler: very few intrinsic functions
  - IMSL Library: Provides hundreds of additional functions
- Microsoft Visual Studio 2008
- MATLAB default: Fixed-format (f77) Fortran code.
- Our code: Free-format (f90), which is similar to MATLAB.
- To change the default settings, modify the batch files (...\bin\win64\mexopts) by deleting the '/fixed' flag
- Use `mex -setup` to select IVF as the compiler in MATLAB

# Writing a Gateway Subroutine

- Every Fortran MEX-file gateway routine begins like this:

```
1 #include "fintrf.h"
2 subroutine mexFunction(nlhs,plhs,nrhs,prhs)
  ! Declarations
3 implicit none
  ! mexFunction arguments
4 mwPointer plhs(*), prhs(*)
5 integer*4 nlhs,nrhs
```

- Line 1 defines pointer types in the MATLAB interface
- Function arguments:
  - prhs: Pointer to an array which holds the input data
  - plhs: Pointer to an array which will hold the output data
  - nrhs: number of right-hand (input) arguments
  - nlhs: number of left-hand (output) arguments
- Line 3 avoids Fortran's implicit type definitions

# KEY MATLAB INTERFACE FUNCTIONS

- `mxGetPr`: Accesses the real data in an mxArray
- `mxGetScalar`: Grabs the value of the first real element of the mxArray (often one element)
- `mxGetM`/`mxGetN`: Determines the number of rows/columns in a specified mxArray
- `mxGetDimensions`: Determines how many elements are in each dimension of the mxArray
- `mxClassIDFromClassName`: Obtains an identifier for any MATLAB class (e.g. Double)
- `mxCreateNumericArray`: Creates an $N$-dimensional mxArray in which all data elements have the numeric data type specified by ClassID.
- `mxCreateDoubleMatrix`: Creates an $m$-by-$n$ mxArray

# Inputting a Policy Function

- Declare variable types and sizes (lines 1-3). If the dimension lengths are variable, use allocatable memory:

```
1 mwpointer c_pr
2 mwSize nk,nz
3 real*8, allocatable, dimension(:,:) :: c
  ! Load Inputs
4 nk = mxGetN(prhs(1))
5 nz = mxGetN(prhs(2))
6 allocate(c(nk,nz))
7 c_pr = mxGetPr(prhs(3))
8 call mxCopyPtrToReal8(c_pr,c,nk*nz)
```

- Load the dimensions of `pf` from inputs (lines 4 and 5) and allocate the memory (line 6)
- Grab the address of the `pf` (input 3), store in `c_pr` (line 7), and copy to Fortran variable `c` (line 8)

# Create Output Matrix

- Load output size: stochastic realizations (lines 1-2)

```
1 e_pr = mxGetDimensions(prhs(4))
2 call mxCopyPtrToInteger4(e_pr,e,1)
  !Create array for return argument
3 cid = mxClassIDFromClassName('double')
4 allocate(o(e))
5 plhs(1) = mxCreateNumericArray(1,e,cid,0)
6 o_pr = mxGetPr(plhs(1))
  ! Call subroutine and load Fortran array
7 call interpfnc(required inputs)
8 call mxCopyReal8ToPtr(o,o_pr,e)
```

- Create $1 \times e$ output vector of type double (lines 3-5) and assign address (line 6)
- Call interpolation subroutine (line 7) and copy the data to the output address (line 8)

# Introduction to Parallel Processing

- Any calculations that are not dependent on the results of other calculations can be performed in parallel (e.g. solving for policy values at each node in the state space)

- With the PCT, a maximum of 12 processors may be used locally with any version of MATLAB 2011a or later (8 processors with 2009a or later, 4 processors with 2007a or later). Additional processors require the MATLAB Distributed Computing Server (MDCS).

- All available processors are initialized with the `matlabpool` command (use `matlabpool open X` for a subset of the available processors) and terminated with `matlabpool close`

# Introduction to Parallel Processing

The PCT requires the following alterations to the code:

- Replace `for` loops with `parfor` loops where applicable. This tells MATLAB to distribute each step in the loop across the available processors

- If a nested `for` loop is used to update policy functions across dimensions, reduce it to a single loop by changing to a single index (as opposed to specifying coordinates)

- Remove all global variables and instead use structures/parameter lists and variable arrays as direct inputs into functions called within the `parfor` loop

# Advantage of MEX and Parallelization

RBC Solution Times–MEX, Parallelization Comparison* (in seconds)

| | MATLAB | | MEX | |
|---|---|---|---|---|
| Processors | Structures | No Structures | Structures | No Structures |
| 1 | 317 (1.0) | 254 (1.2) | 191 (1.7) | 132 (2.4) |
| 2 | 164 (1.9) | 132 (2.4) | 101 (3.1) | 71 (4.5) |
| 6 | 65 (4.9) | 53 (6.0) | 41 (7.8) | 32 (10.0) |

* Based on a state space comprised of 1,681 nodes (41 points on $k_t$ and $z_t$). We specify 10 realizations of $\varepsilon_{t+1}$. The routines were computed with an Intel W3670 6-core processor (3.20GHz) operating 64-bit Windows 7.

# Conventional New Keynesian Model

The representative household chooses $\{c_{t+i}, n_{t+i}, M_{t+i}, k_{t+i}\}_{i=0}^{\infty}$ to maximize expected lifetime utility,

$$E_t \sum_{i=0}^{\infty} \beta^i \left\{ \frac{c_{t+i}^{1-\sigma}}{1-\sigma} - \chi \frac{n_{t+i}^{1+\eta}}{1+\eta} + \nu \frac{\left(M_{t+i}/P_{t+i}\right)^{1-\kappa}}{1-\kappa} \right\}, \qquad \chi, \nu > 0$$

subject to

$$c_t + m_t + i_t + b_t = (1-\tau_t)(w_t n_t + r_t^k k_{t-1}) + (m_{t-1} + r_{t-1} b_{t-1})/\pi_t + d_t$$
$$k_t = (1-\delta)k_{t-1} + i_t$$

Optimality implies

$$\chi n_t^{\eta} c_t^{\sigma} = (1-\tau_t) w_t \qquad \text{(FOC n)}$$
$$\nu m_t^{-\kappa} = (1 - 1/r_t) c_t^{-\sigma} \qquad \text{(FOC m)}$$
$$1 = \beta r_t E_t\{(c_t/c_{t+1})^{\sigma}/\pi_{t+1}\} \qquad \text{(FOC b)}$$
$$1 = \beta E_t\left\{(c_t/c_{t+1})^{\sigma}\left(r_{t+1}^k + 1 - \delta\right)\right\} \qquad \text{(FOC k)}$$

# Conventional New Keynesian Model

- Firm optimality conditions:

$$y_t = k_{t-1}^{\alpha} n_t^{1-\alpha}$$

$$r_t^k = \alpha \Psi_t y_t / k_{t-1}$$

$$w_t = (1-\alpha)\Psi_t y_t / n_t$$

$$\varphi\left(\frac{\pi_t}{\bar{\pi}} - 1\right)\frac{\pi_t}{\bar{\pi}} = (1-\theta) + \theta\Psi_t + \varphi E_t\left[q_{t,t+1}\left(\frac{\pi_{t+1}}{\bar{\pi}} - 1\right)\frac{\pi_{t+1}}{\bar{\pi}}\frac{Y_{t+1}}{Y_t}\right]$$

- Government Budget Constraint:

$$m_t + b_t + \tau_t(w_t n_t + r_t^k k_{t-1}) = \bar{g} + \underbrace{(m_{t-1} + r_{t-1} b_{t-1})}_{\equiv a_t}/\pi_t$$

- Monetary and Fiscal Policy Rules:

$$r_t = \bar{r}(\pi_t/\pi^*)^{\phi(s_t)}\exp(\varepsilon_{r,t}), \qquad \tau_t = \bar{\tau}(b_{t-1}/b^*)^{\gamma(s_t)}\exp(\varepsilon_{\tau,t})$$

- Aggregate Resource Constraint:

$$c_t + i_t + \bar{g} = [1 - \varphi(\pi_t/\bar{\pi} - 1)^2/2]y_t$$

# SOLVING THE MODEL: GUESS

- Minimum State Nonlinear Model: $a_t$, $b_{t-1}$, $k_{t-1}$, $\varepsilon_{r,t}$, $\varepsilon_{\tau,t}$, $s_t$
- Minimum State Linear Model: $m_{t-1}$, $r_{t-1}$, $b_{t-1}$, and $k_{t-1}$. Must average across one of the components of $a_t = m_{t-1} + r_{t-1}b_{t-1}$ to make these sets consistent and replicate across the shocks
- Use a convex combination of the linear solution for each policy mix with weights equal to the transition probabilities to form a guess for the nonlinear policy functions
  - Weighting the linear solutions approximates the expectational effect of switching between regimes

# SOLVING THE MODEL: POLICIES

- Policy Functions: $c_t$, $n_t$, $\pi_t$, $\Psi_t$. Note: $\Psi_t$ can be omitted if the firm's optimality conditions are rewritten as

$$\frac{k_{t-1}}{n_t} = \frac{\alpha}{1-\alpha}\frac{w_t}{r_t^k}$$

$$\Psi_t = w_t^{1-\alpha}(r_t^k)^{\alpha}(1-\alpha)^{-(1-\alpha)}\alpha^{-\alpha}$$

- Given the state and policies, solve for all time $t$ variables
- Interpolate values for the updated policies across both shocks and solve for variables inside expectations
  - With two continuous stochastic components, interpolated policies have $N_{\varepsilon_\tau} \times N_{\varepsilon_r}$ realizations (a matrix of values). Requires a nested loop in the interpolation function.

# Solving the Model: Integration

- Must numerically integrate across both random variables:
  - Replicate $\Pr(\varepsilon_{\tau,i})$ (the row weights) across the $\varepsilon_{r,j}$-dimension (the columns) to create an $N_{\varepsilon_\tau} \times N_{\varepsilon_r}$ weighting matrix
  - Use the weighting matrix to weight each element of $[\Phi_{i,j}]$
  - Use either the Trapezoid rule or GH to integrate across the $\varepsilon_\tau$-dimension (the rows)—collapses the matrix of realizations to a row vector
  - Weight each of the outcomes in the row vector by its respective $\Pr(\varepsilon_{r,j})$
  - Use either the Trapezoid rule or GH to integrate across the $\varepsilon_r$-dimension—reduces to a single value (state dependent)
- Use transition probabilities to integrate across the discrete random variable (policy regime state)

# Advantage of MEX and Parallelization

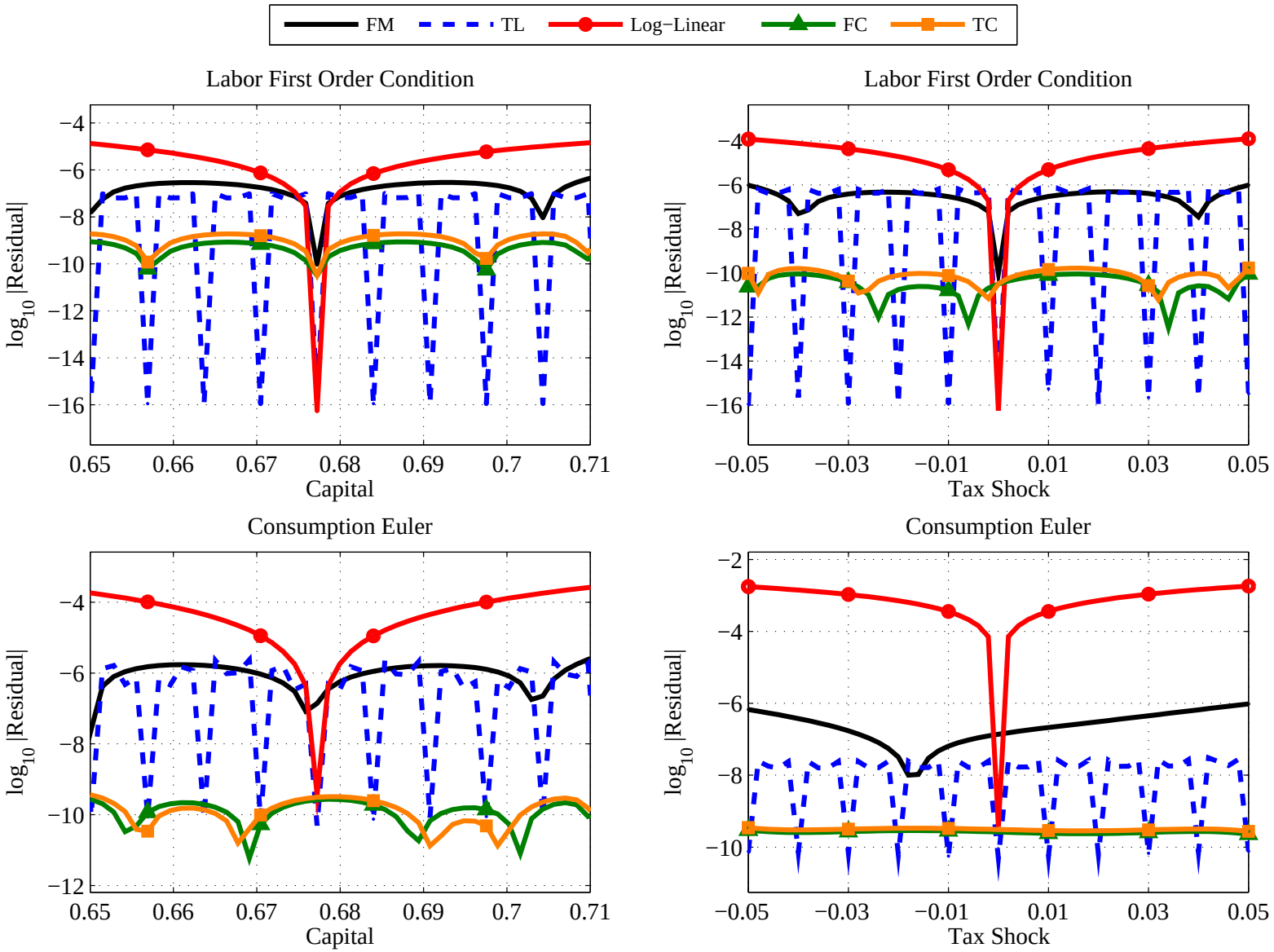
- Runs are based on 1 continuous stochastic variable ($\varepsilon_\tau$) and fixed AM/PF policy

NK Model Solution Times* (in seconds)

| Processors | No MEX | Partial MEX | All MEX |
|---|---|---|---|
| 1 | 288.2 (1.0) | 131.5 (2.2) | 32.0 (9.0) |
| 2 | 148.7 (1.9) | 68.6 (4.2) | 16.2 (16.2) |
| 6 | 54.1 (5.3) | 27.0 (10.7) | 8.2 (35.1) |

* Based on a state space comprised of 2,401 nodes (7 points on each continuous state variable). We specify 10 realizations of $\varepsilon_\tau$. The routines were computed with an Intel W3670 6-core processor (3.20GHz) operating 64-bit Windows 7. The value in parenthesis represents that speed factor increase from the slowest setup.

# Euler Equation Errors (fixed AM/PF)



# SPEED COMPARISON

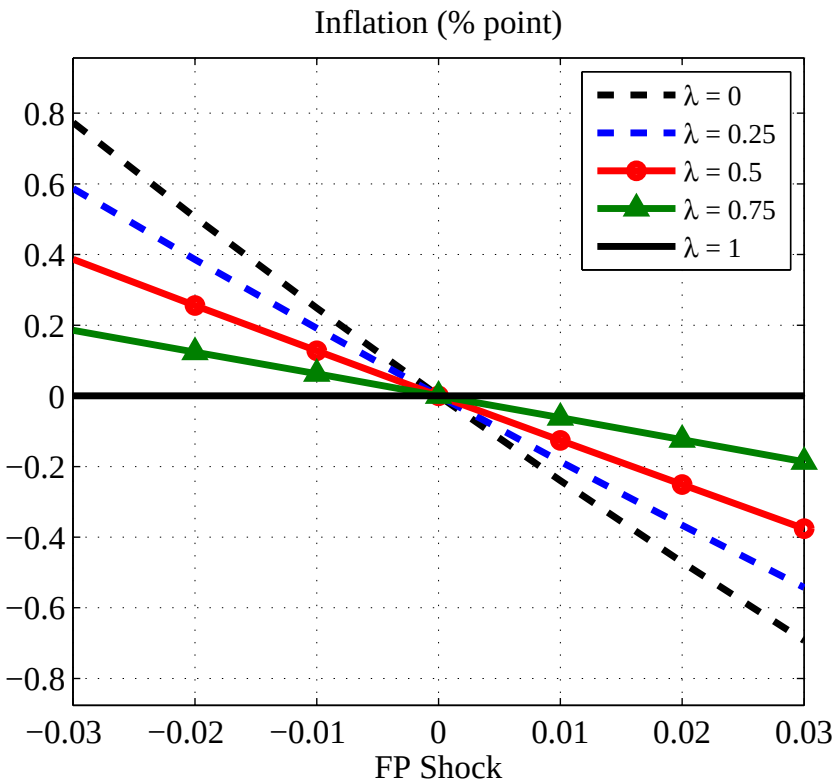
Solution Times–Solution Methods Comparison* (in seconds)

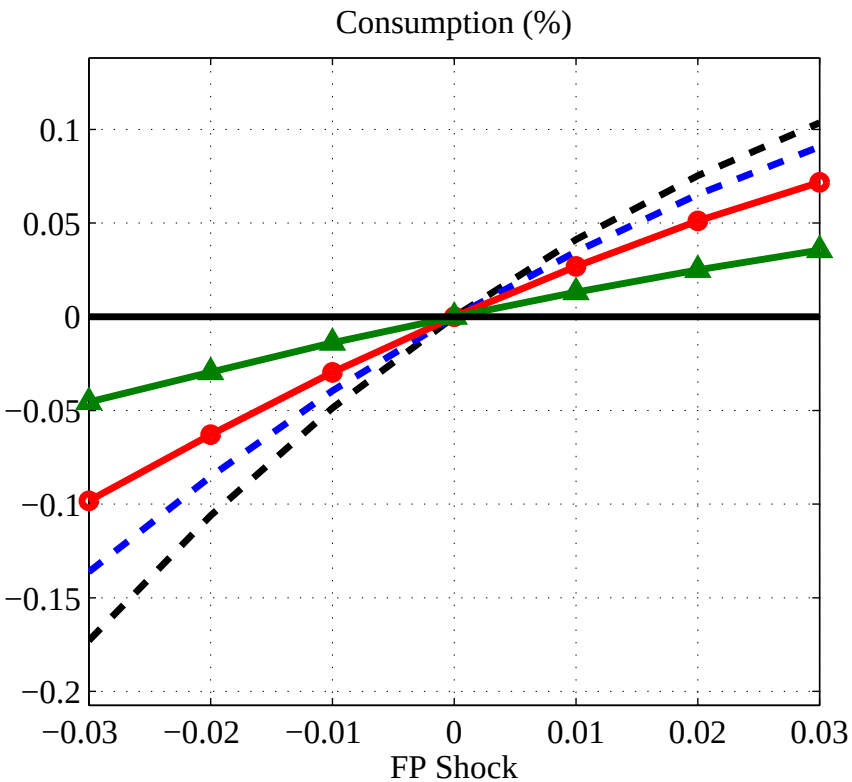
| | RBC | | NK | |
|---|---|---|---|---|
| Method | MATLAB | MEX | MATLAB | MEX |
| FM | 3 | N/A | 3 | N/A |
| FC | 139 | 6 | 948 | 19 |
| TL | 62 | 42 | 69 | 39 |
| TC | 106 | 44 | 258 | 40 |

* RBC solution is based on a state space comprised of 1,681 nodes (41 points on $k_{t-1}$, 41 points on $z_t$). We specify 10 realizations of $\varepsilon_{t+1}$. NK solution is based on a state space comprised of 2,401 nodes (7 points on $a_{t-1}$, $b_{t-1}$, $k_{t-1}$, and $\varepsilon_{\tau,t}$). We specify 10 realizations of $\varepsilon_{\tau,t+1}$. The routines were computed with an Intel W3670 6-core processor (3.20GHz) operating 64-bit Windows 7. All locally available processors were used.

# POLICY FUNCTIONS (LUMP-SUM TAXES)

- Average duration of time spent in the AM/PF regime in the ergodic distribution: $\lambda = (1 - p_{22})/(2 - p_{11} - p_{22})$
- When $\lambda = 1$, there is no chance of moving to the PM/AF

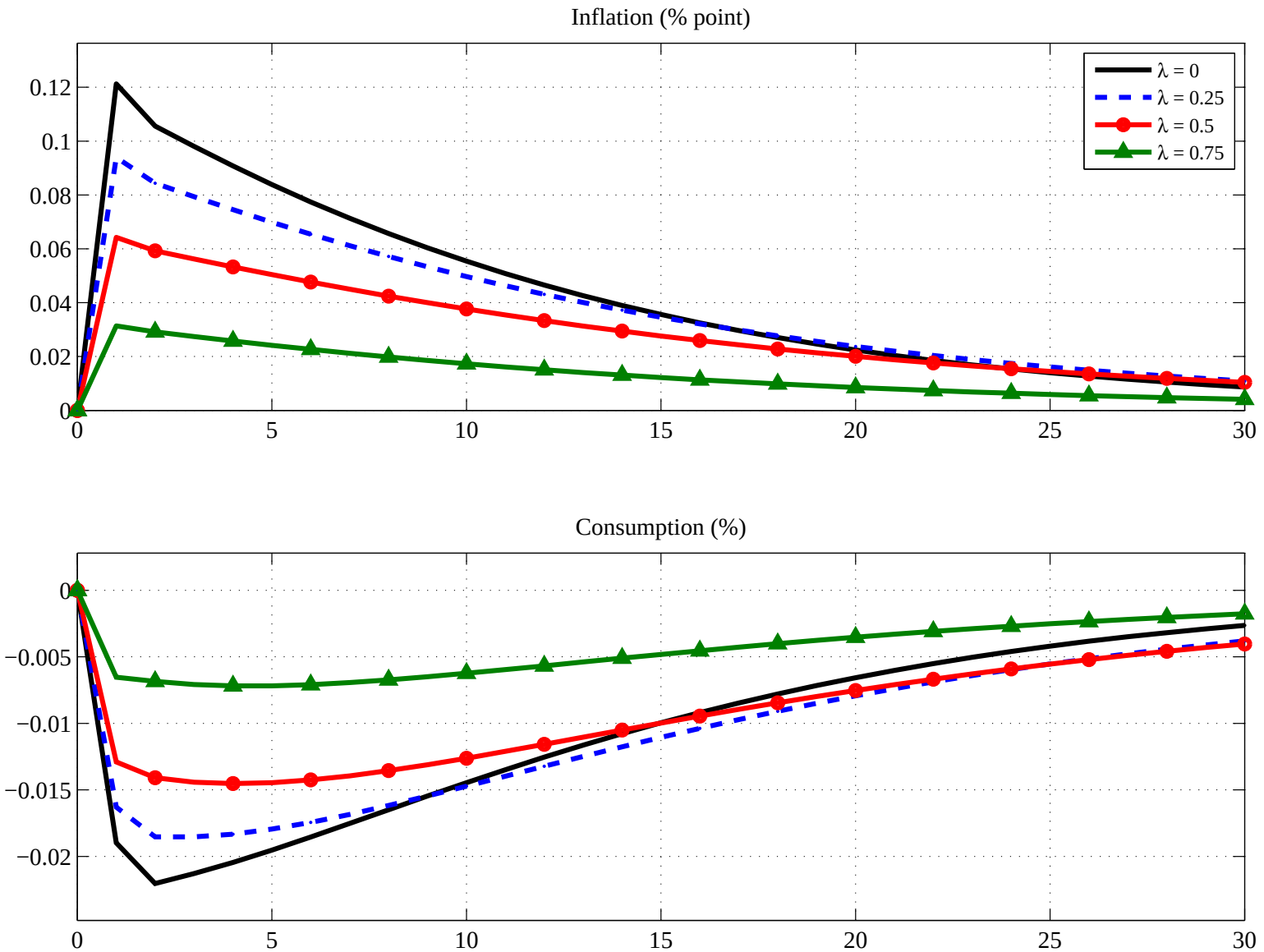


# Impulse Responses

- Nonlinear impulse responses differ from linear responses
- Linear impulse responses report how a shock makes each simulated variable differ from its *deterministic* (calculated) steady state
- Nonlinear impulse responses report how a shock makes each simulated path differ from the *stochastic* steady state, defined as a value for the state vector such that $|\Theta_t - \Theta_{t-1}| < \epsilon$, where $\Theta$ is a vector of policy functions
- To simulate the model, burn off the number of periods needed to ensure that the model reaches its *stochastic* steady state
- Although the transversality condition is not imposed, simulated paths in models that explicitly violate this condition will diverge even if the algorithm converges

# Impulse Response: FP Shock



# Explosive Transfers and a Fiscal Limit

- State: $a_t$, $b_{t-1}$, $k_{t-1}$, $z_t$, $s_{z,t}$, $s_{p,t}$ (no shocks on MP or FP)
- $z$: lump-sum transfers to household, distributed AR(1)
  - $s_z = 1$: transfers process is stationary.
  - $s_z = 2$: transfers process is non-stationary.
- $s_p$: monetary/fiscal policy mix
  - $s_p = 1$: active MP, passive FP, and promised transfers (Before Fiscal Limit)
  - $s_p = 2$: passive MP, active FP, and promised transfers (After Fiscal Limit)
  - $s_p = 3$: active MP, active FP, and delivered transfers (After Fiscal Limit)
- The probability of hitting the fiscal limit—point where higher taxes are no longer feasible—is endogenous

$$p_{FL,t} = 1 - \frac{\exp(\eta_0 - \eta_1(\tau_{t-1} - \bar{\tau}))}{1 + \exp(\eta_0 - \eta_1(\tau_{t-1} - \bar{\tau}))}$$

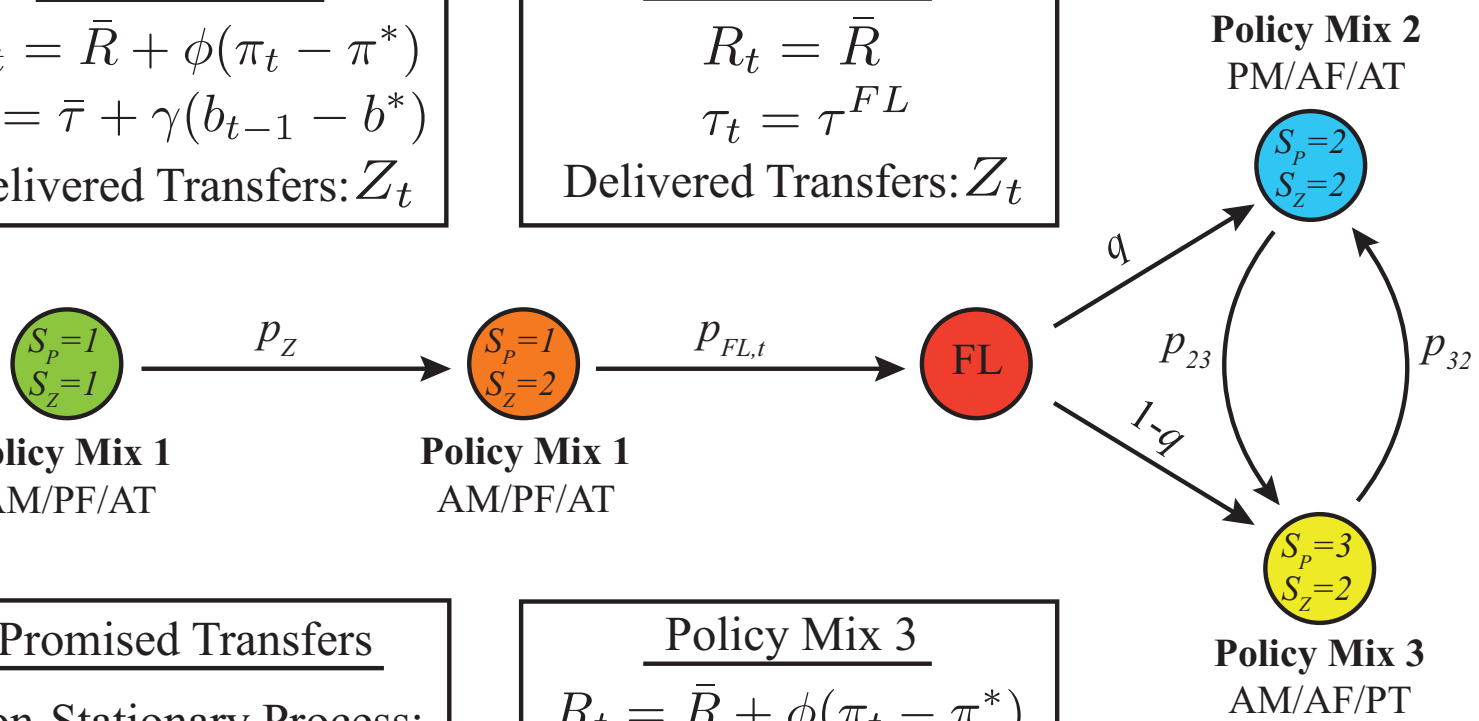
# Possible Policy Evolution

**Policy Mix 1**

$$R_t = \bar{R} + \phi(\pi_t - \pi^*)$$
$$\tau_t = \bar{\tau} + \gamma(b_{t-1} - b^*)$$
Delivered Transfers: $Z_t$

**Policy Mix 2**

$$R_t = \bar{R}$$
$$\tau_t = \tau^{FL}$$
Delivered Transfers: $Z_t$




**Policy Mix 1**
AM/PF/AT

**Policy Mix 1**
AM/PF/AT

**Policy Mix 2**
PM/AF/AT

**Policy Mix 3**
AM/AF/PT

**Promised Transfers**

Non-Stationary Process:

$$Z_t = \rho_Z^{NS} Z_{t-1} + \varepsilon_t$$

**Policy Mix 3**

$$R_t = \bar{R} + \phi(\pi_t - \pi^*)$$
$$\tau_t = \tau^{FL}$$
Delivered Transfers: $\lambda_t Z_t$

# Obtaining a Guess

To obtain initial conjectures for the policy functions:

- First solve the *linear* model under both the initial (AM/PF/AT) and debt revaluation (PM/AT/AT) regimes
- Use these linear solutions as initial guesses for the corresponding fixed-regime *non-linear* models, assuming there is no probability of hitting the fiscal limit
- Use weighted averages of these non-linear solutions to obtain initial conjectures for each regime combination (3 policy states and 2 transfers states form 6 combinations)

# SOLVING THE MODEL

- Given the state and numerical policies, calculate time $t$ variables (state dependent)
- Calculate updated (time $t+1$) values for each of the policy variables using piecewise linear interpolation/extrapolation. Note that $s_z$ and $s_p$ form 6 possible outcomes.
- Solve for the variables that are inside expectations (obtain all other $t+1$ values from the interpolated policies)
- Integrate across the continuous transfers shock. Each expectation will be conditional on the 6 possible outcomes
- Calculate $p_{FL}$ and integrate across the policy regimes ($s_p$)
- Integrate across the transfers regimes ($s_z$)
- Using Chris Sims' root finder, `csolve`, solve for the zeros of the equations with embedded expectations, subject to each of the remaining equilibrium conditions.

# Stability and Grid Adjustments

- Achieve stability by starting with a narrow grid near the initial steady state
- Simulate the model and identify areas of extrapolation
- Since non-stationary transfers cause drift, must make small adjustments to the grid using previous nonlinear solution as guess
- Continue adjusting the grids until they encompass all simulated paths of state variables

# Monte Carlo Simulations

- Pick a large number of times to simulate the model
- Begin all simulations from stochastic steady state
- Take draws from the distributions of the random variables using random number generators
  - $\varepsilon_{z,t} \sim i.i.d.N(0, \sigma_z^2)$
  - For Markov processes, draw from $U(0, 1)$ and create sequences for each discrete state variable (transfers and policy regimes) using the respective transition matrix
- Use the `quantile` function to calculate paths that characterize the tails of the distribution for variables of interest across time, e.g., 10th and 90th percentile of all realizations