Value Function Iteration

Numerical Methods

Cezar Santos

FGV/EPGE

Acknowledgements

- This set of slides is heavily based on notes by Georg Dürnecker and Grey Gordon.

Value Function Iteration

- Well known, basic algorithm of dynamic programming
- Well suited for parallelization
- It will always work (perhaps quite slowly)
- Subject to the curse of dimensionality

Our Example

Remember our RBC example:

$$V(k,z) = \max_{c,k'} u(c) + E\beta V(k',z')$$

s.t.

$$c + k' = zf(k) + (1-\delta)k$$

The Basic Idea

- Start with an initial guess for the value function, $V^0(k)$
- Maximize the right hand side and compute $V^1(k)$
- Repeat procedure until $||V^j(k) - V^{j-1}(k)|| < \varepsilon$, where ε is a small number

The Basic Idea

- Start with an initial guess for the value function, $V^0(k)$
- Maximize the right hand side and compute $V^1(k)$
- Repeat procedure until $||V^j(k) - V^{j-1}(k)|| < \varepsilon$, where ε is a small number

What's the simplest way to solve this?

- Discretize everything!
- Create grids for the states and for the controls

A "Similar" Problem

With discretized states and controls:

$$V(k,z) = \max_{c,k'} u(c) + \beta \sum_{z'} \pi(z'|z) V(k',z')$$

s.t.

$$c + k' = zf(k) + (1-\delta)k$$

for all $k \in \mathscr{K}$ and $z \in \mathscr{Z}$.

Questions:

1. What are the relevant k and z? $\mathscr{K} \times \mathscr{Z}$
2. How can V be represented on a computer? As an array
3. How does one compute Ez? Markov chain
4. How does one compute the optimal c and k'? Checking all k'

New Questions

- What is an appropriate/optimal discretization of $\mathscr{K}$ and $\mathscr{Z}$?
- How does one discretize the AR1 into a Markov chain?
- How does one assess the quality of the approximation?

Discretizing the Grid for k

Questions:

- What are appropriate lower and upper bounds?
- How many points should there be and where?

Bounds for the k Grid

Upper and lower bounds $[\underline{k}, \overline{k}]$ for $\mathscr{K}$:

- Many models have an explicit lower bound. There may also be an endogenous lower bound.
- Most models do not have an explicit upper bound instead relying on an endogenous upper bound.

Bounds for the k Grid

Upper and lower bounds $[\underline{k}, \overline{k}]$ for $\mathscr{K}$:

- Many models have an explicit lower bound. There may also be an endogenous lower bound.
- Most models do not have an explicit upper bound instead relying on an endogenous upper bound.

Guesses for endogenous bounds should be taken from real-life:

- For instance, even if exogenous lower bound $\underline{k} = 0$, one should not use this as that is pre-industrial revolution.
- $\underline{k} = 0.5k_{ss}$ would also generally be too low because k is closely related to wealth and we have probably never seen aggregate wealth be 50% below "average". Note that k_{ss} is generally a good proxy for Ek.
- $\overline{k} = 1.5k_{ss}$ would likewise probably be too high.
- A reasonable guess would be $[\underline{k}, \overline{k}] = [0.7k_{ss}, 1.3k_{ss}]$, but this should be verified once the solution is computed.

Quantity and Location of the k Grid Points

Where should the points be?

- Linearly spaced (most common): often good starting point
- Concentrated, usually around $\underline{k}$ (close second)
- Stochastic (distant third, only potentially useful in more than one dimension)

Useful if we need extra precision near some points

Quantity and Location of the k Grid Points

Where should the points be?

- Linearly spaced (most common): often good starting point
- Concentrated, usually around $\underline{k}$ (close second)
- Stochastic (distant third, only potentially useful in more than one dimension)

How many points should there be?

- As many as you can handle? Maybe too much.
- Ideally, adding points wouldn't change your results.
- Maybe, try to start with 100. But, remember, you need to check.

Discretizing an AR1 into a Markov Chain

- Consider the following AR1 process:

$$\theta_t = (1-\rho)\mu + \rho\theta_{t-1} + \varepsilon_t$$

- θ is a continuous variable and continuous variables are hard to work with numerically
- Task: Approximate the continuous AR(1) process with a discrete first-order Markov chain
- Aim: Discretize θ so that the resulting process resembles the continuous AR(1) process

Discretizing an AR1 into a Markov Chain

- Consider the following AR1 process:

$$\theta_t = (1-\rho)\mu + \rho\theta_{t-1} + \varepsilon_t$$

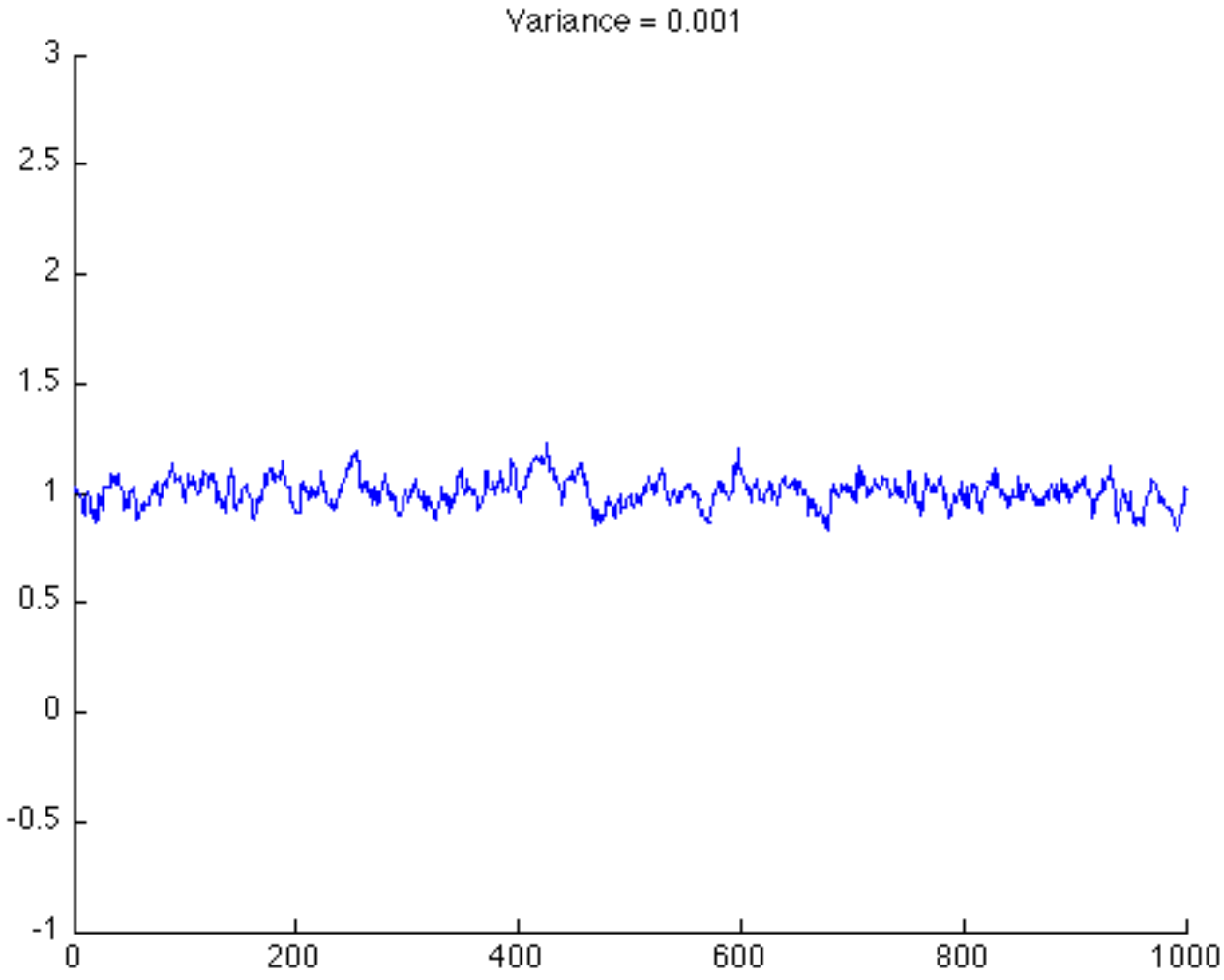
- θ is a continuous variable and continuous variables are hard to work with numerically
- Task: Approximate the continuous AR(1) process with a discrete first-order Markov chain
- Aim: Discretize θ so that the resulting process resembles the continuous AR(1) process

- We'll see a couple of methods:
  - Tauchen (1986)
  - Rouwenhorst (Kopecky and Suen, 2010)

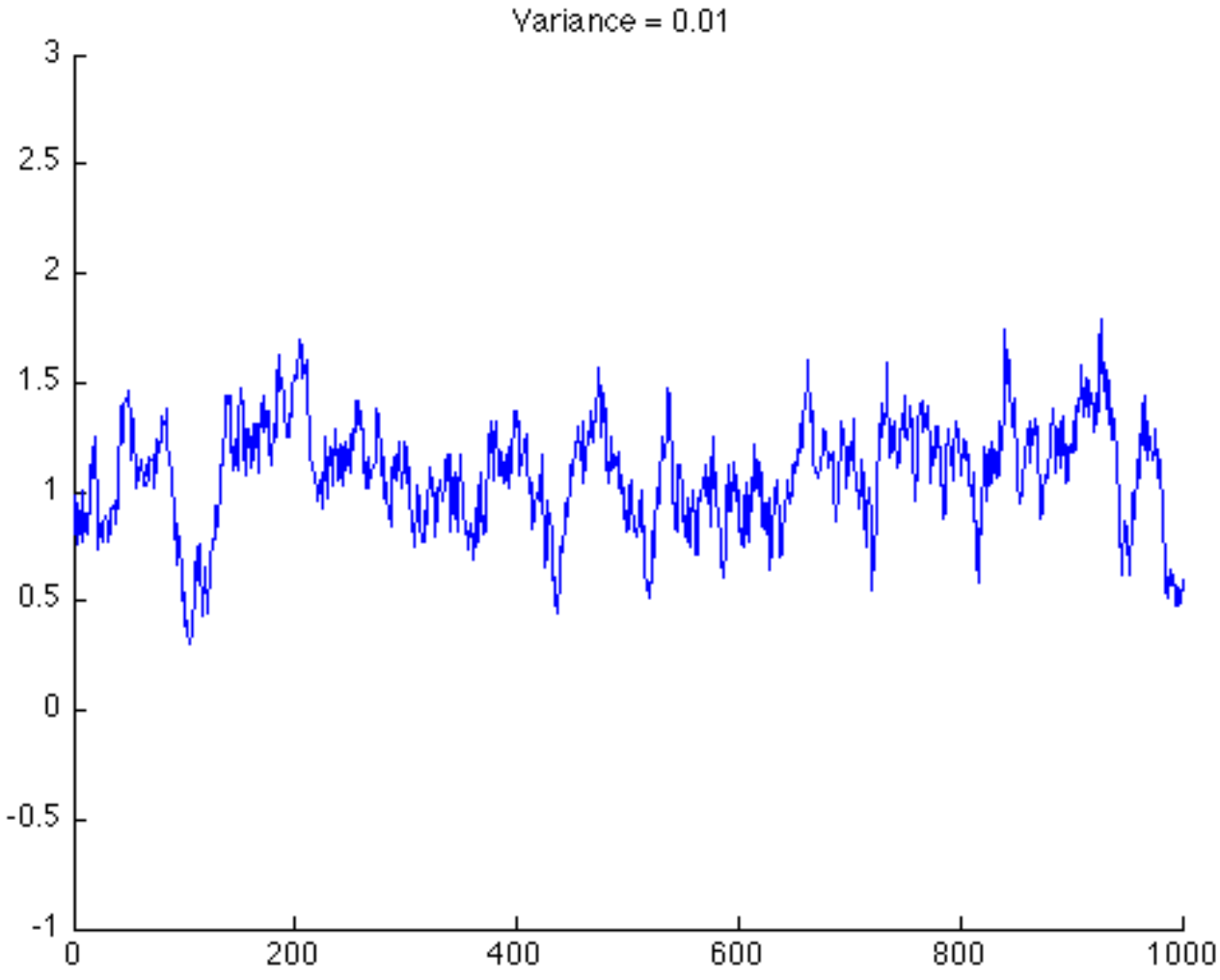
Tauchen's Method

- First issue: Determine the range of the grid and the location and the number of the grid points
- Choice of grid size depends on the persistence and the variance of the underlying continuous process
- Higher variance $\sigma^2 \rightarrow$ a larger range of the state space needs to be covered by the grid

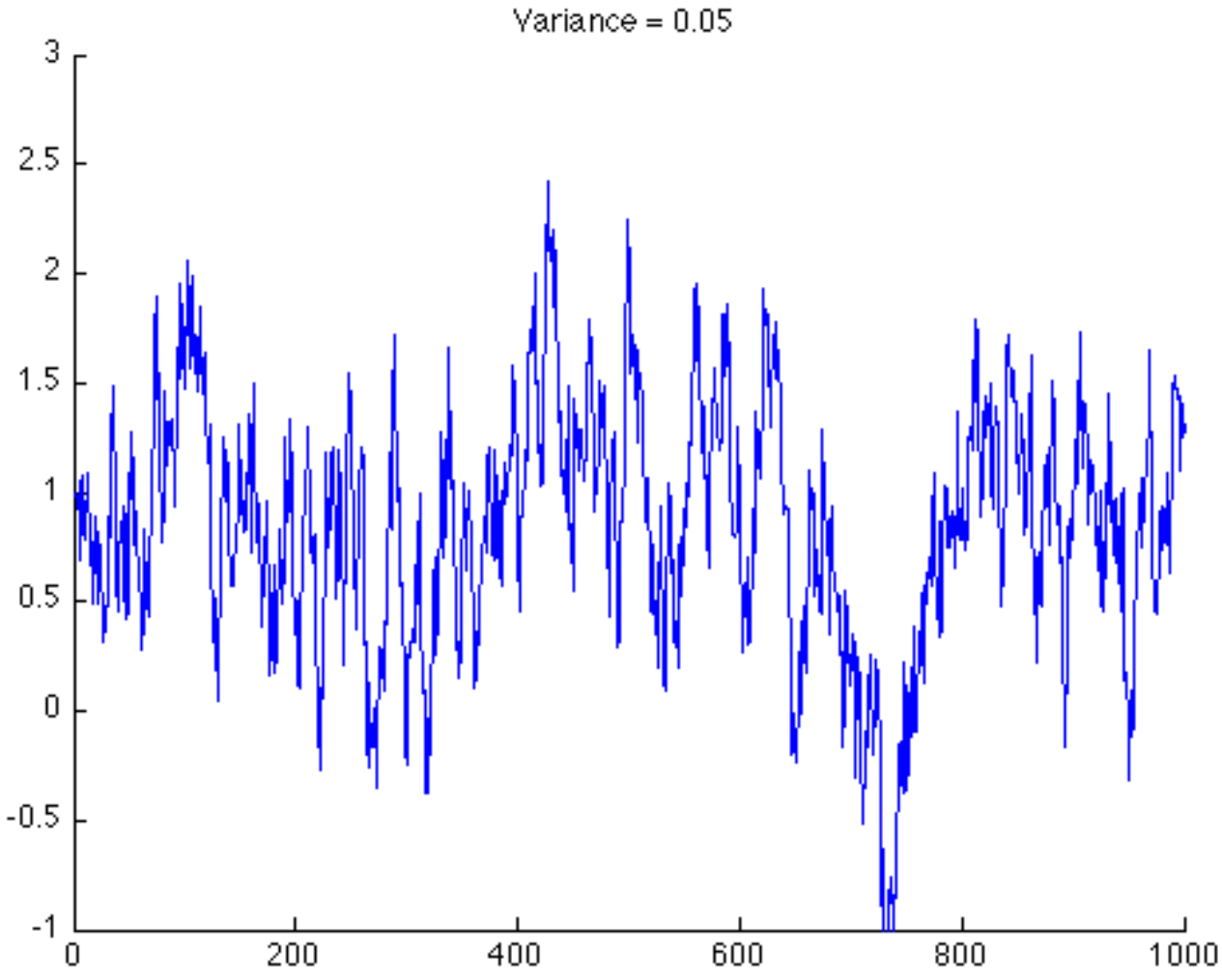
AR1 Simulation



AR1 Simulation



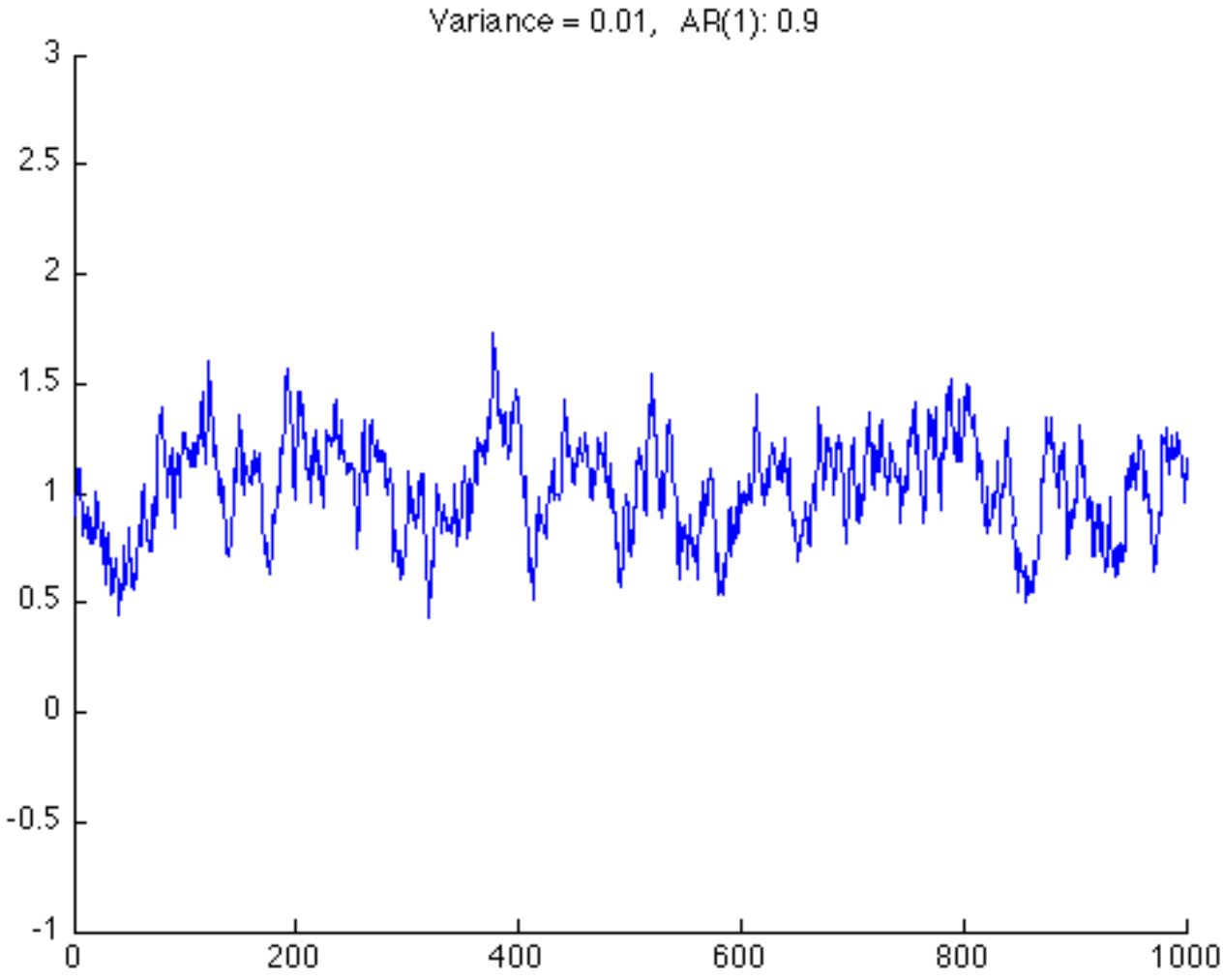
AR1 Simulation



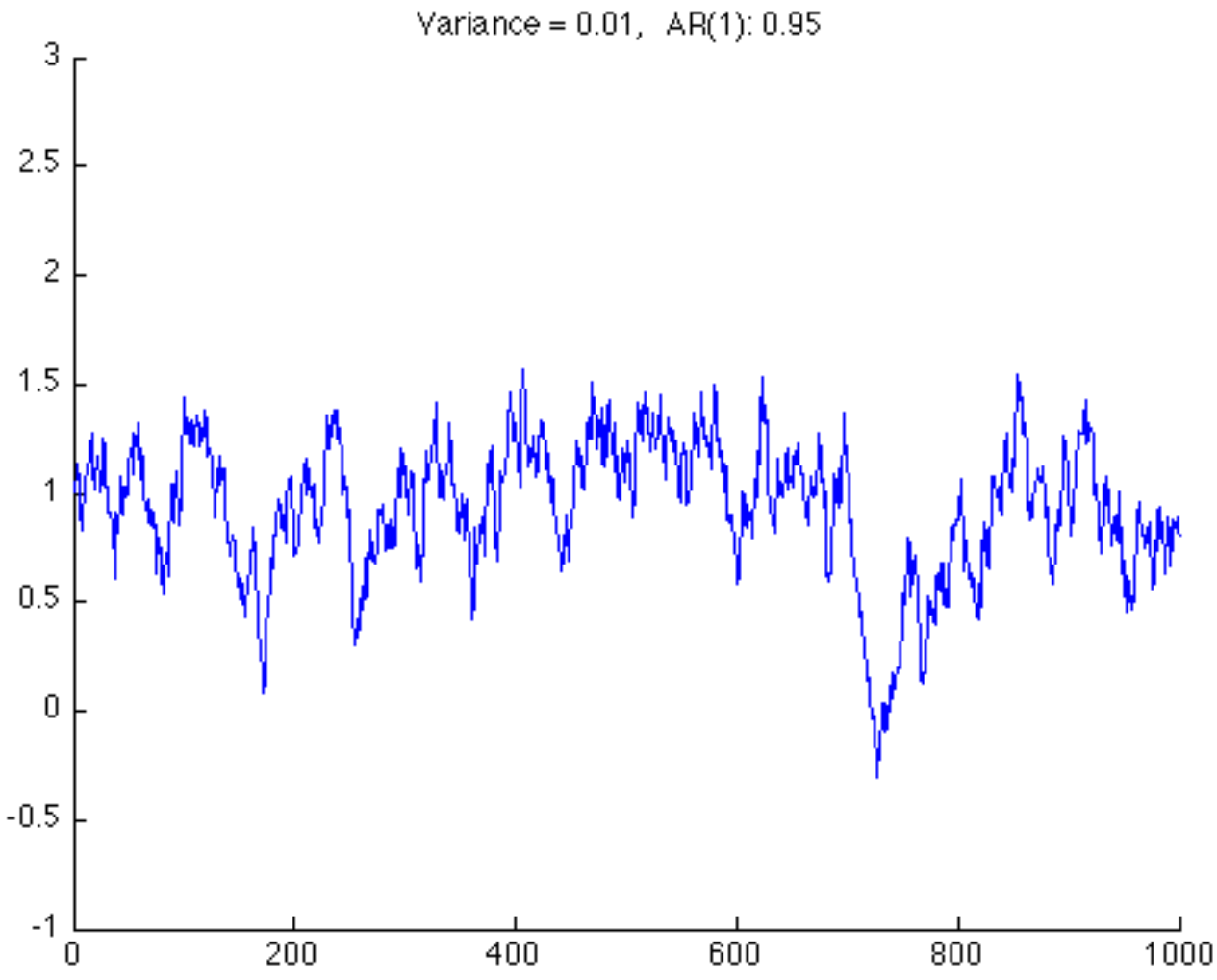
Tauchen's Method

- First issue: Determine the range of the grid and the location and the number of the grid points
- Choice of grid size depends on the persistence and the variance of the underlying continuous process
- Higher variance $\sigma^2 \rightarrow$ a larger range of the state space needs to be covered by the grid
- The same for persistence, ρ

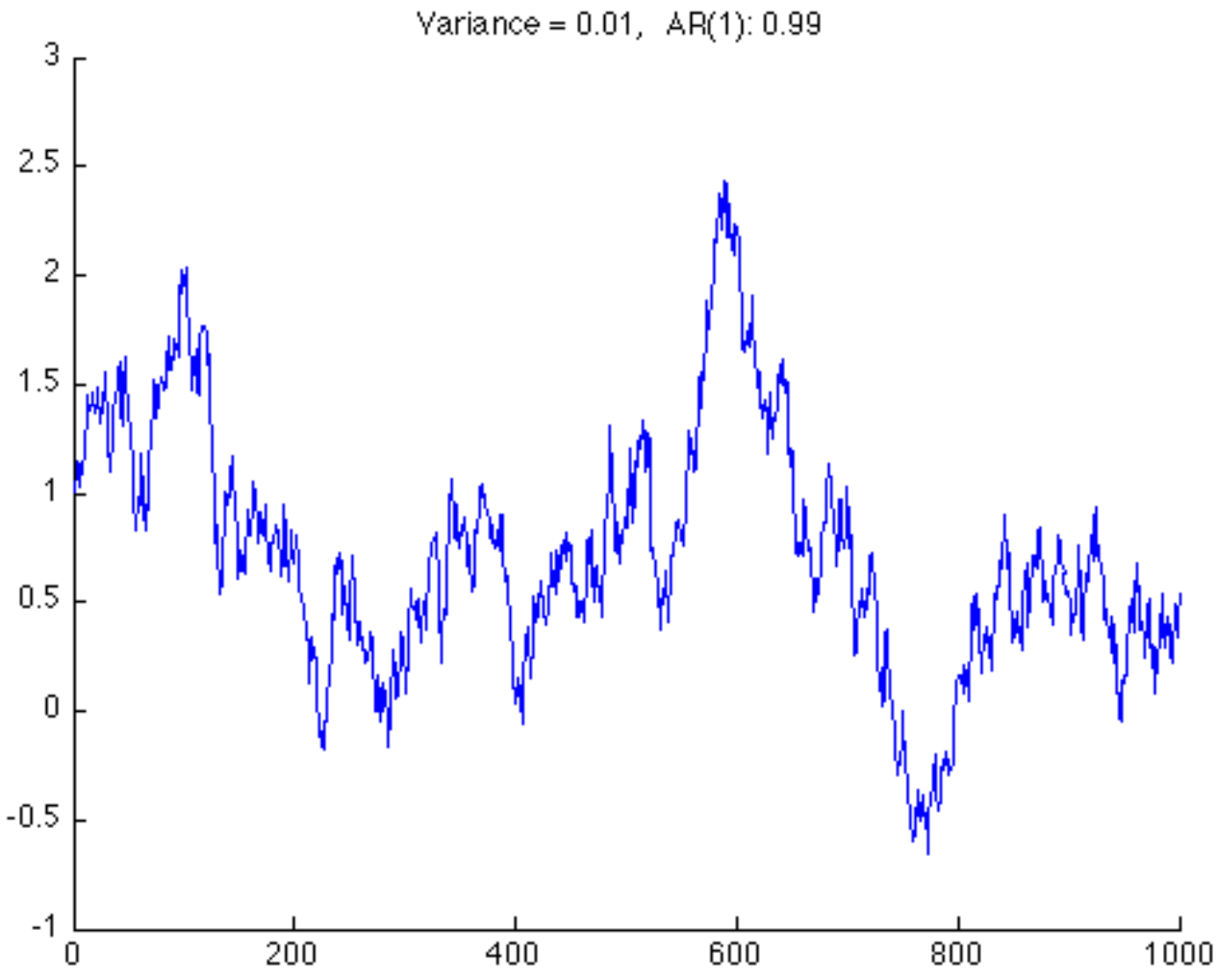
AR1 Simulation



AR1 Simulation



AR1 Simulation



Specifying the Grid

- Set the upper bound of the grid, θ_N, equal to

$$\theta_N = m\frac{\sigma}{\sqrt{1-\rho^2}}$$

where $\sigma(\sqrt{1-\rho^2})^{-1}$ is the unconditional standard deviation of θ and m is a scaling parameter

Specifying the Grid

- Set the upper bound of the grid, θ_N, equal to

$$\theta_N = m\frac{\sigma}{\sqrt{1-\rho^2}}$$

  where $\sigma(\sqrt{1-\rho^2})^{-1}$ is the unconditional standard deviation of θ and m is a scaling parameter

- The lower bound, θ_1, is set equal to $\theta_1 = -\theta_N$

Specifying the Grid

READ THIS STUFF

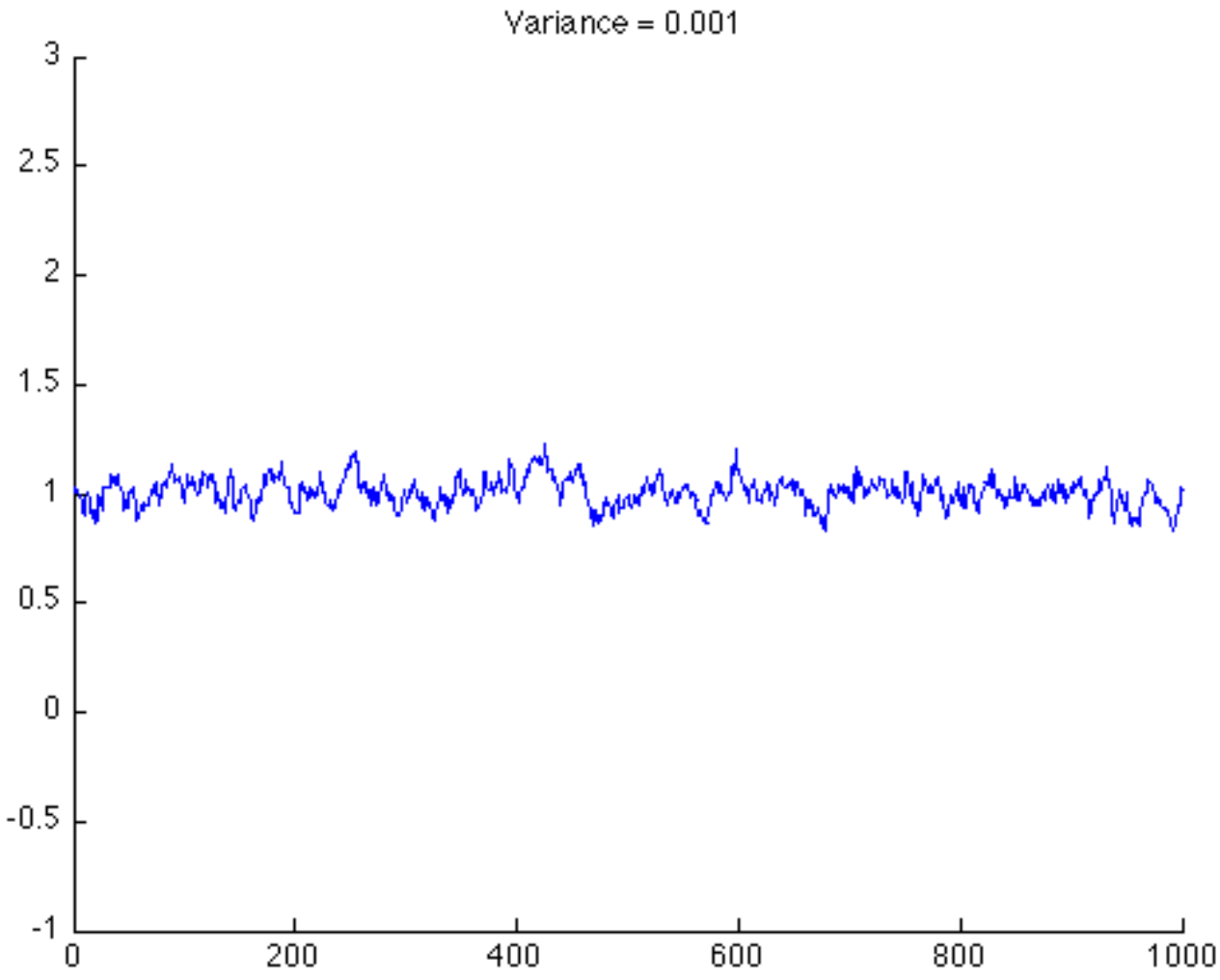
- Set the upper bound of the grid, θ_N, equal to

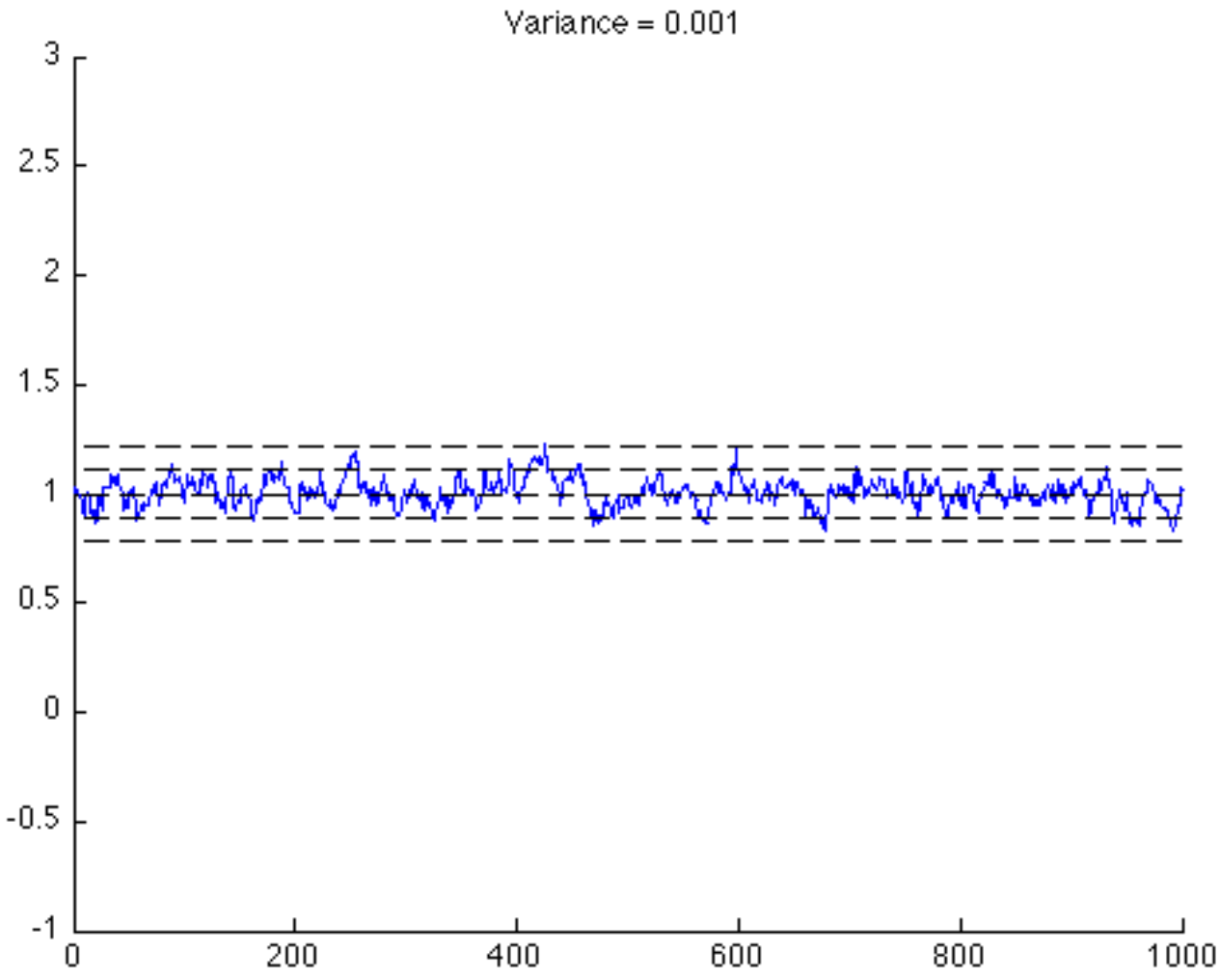
$$\theta_N = m\frac{\sigma}{\sqrt{1-\rho^2}}$$

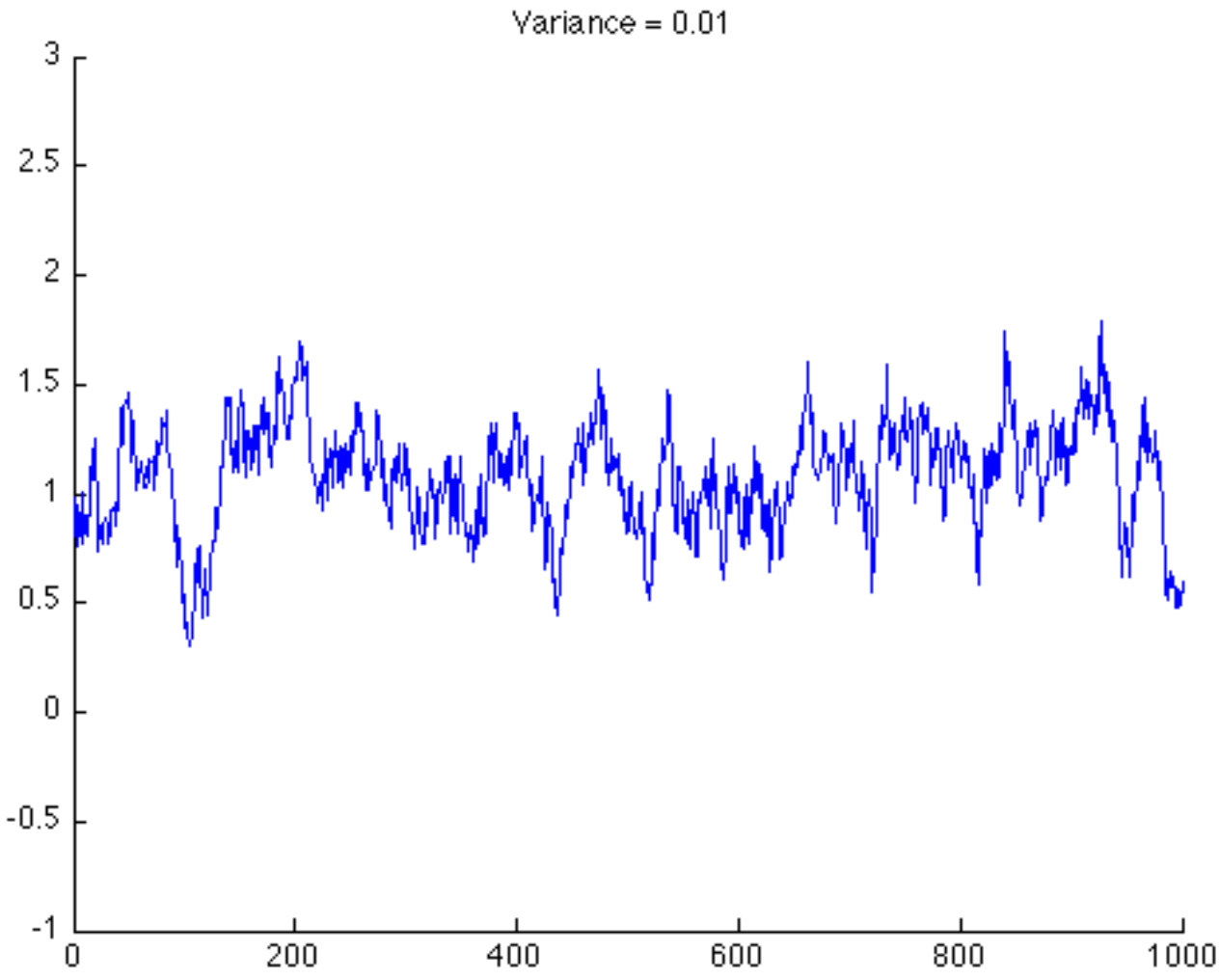
  where $\sigma(\sqrt{1-\rho^2})^{-1}$ is the unconditional standard deviation of θ and m is a scaling parameter

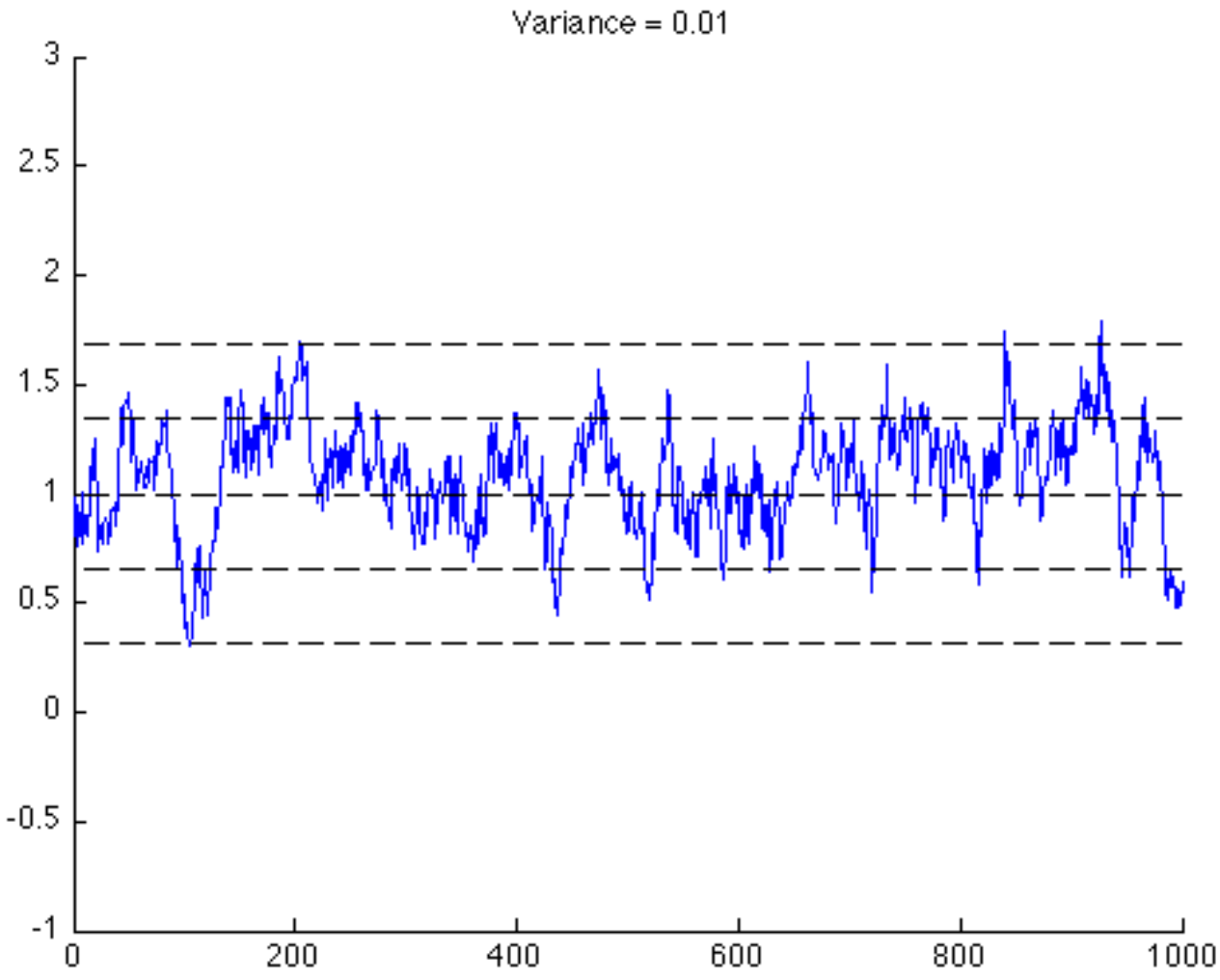
- The lower bound, θ_1, is set equal to $\theta_1 = -\theta_N$

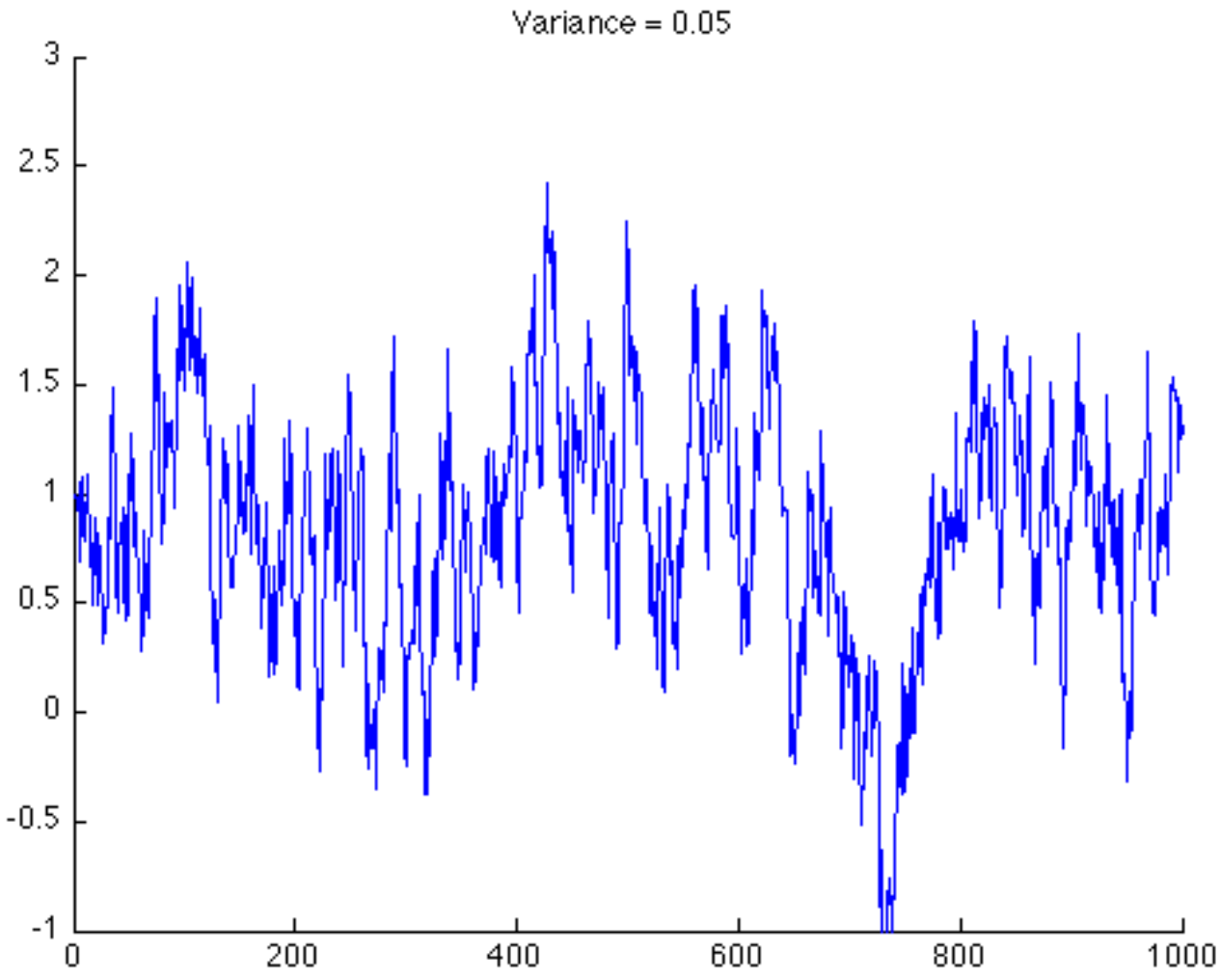
- The remaining $N-2$ grid points are equidistantly distributed between θ_1 and θ_N

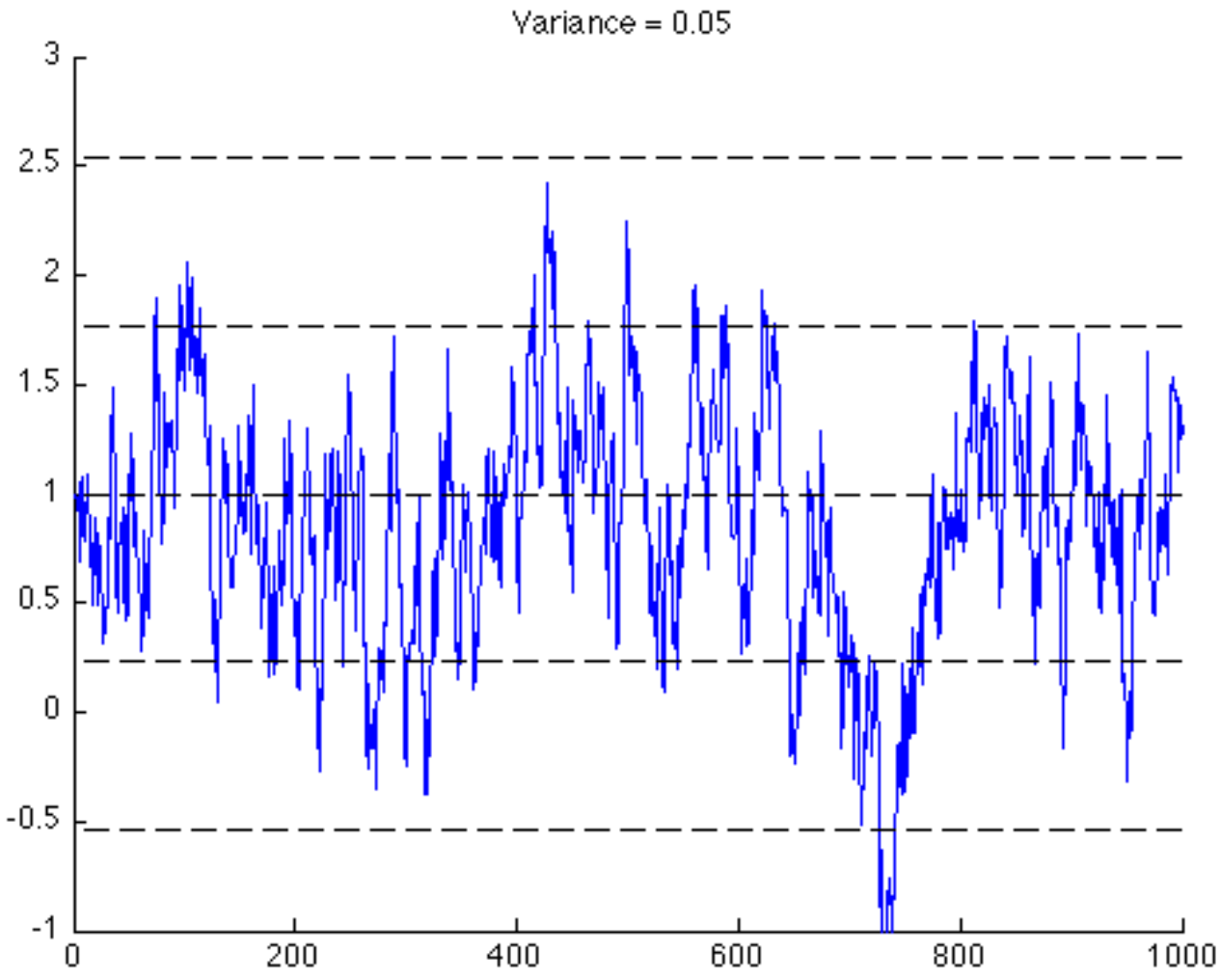

Variance = 0.001
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000


Variance = 0.001
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000


Variance = 0.01
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000


Variance = 0.01
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000


Variance = 0.05
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000


Variance = 0.05
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000

Computing the Transition Probabilities

- Next we need to know what is the probability of moving to state θ_j conditional on being in state θ_i
- Denote this probability by $p_{i,j} = P(\theta_{t+1} = \theta_j | \theta_t = \theta_i)$ and let P be the matrix of $\{p_{i,j}\}_{i=1,j=1}^{N,N}$

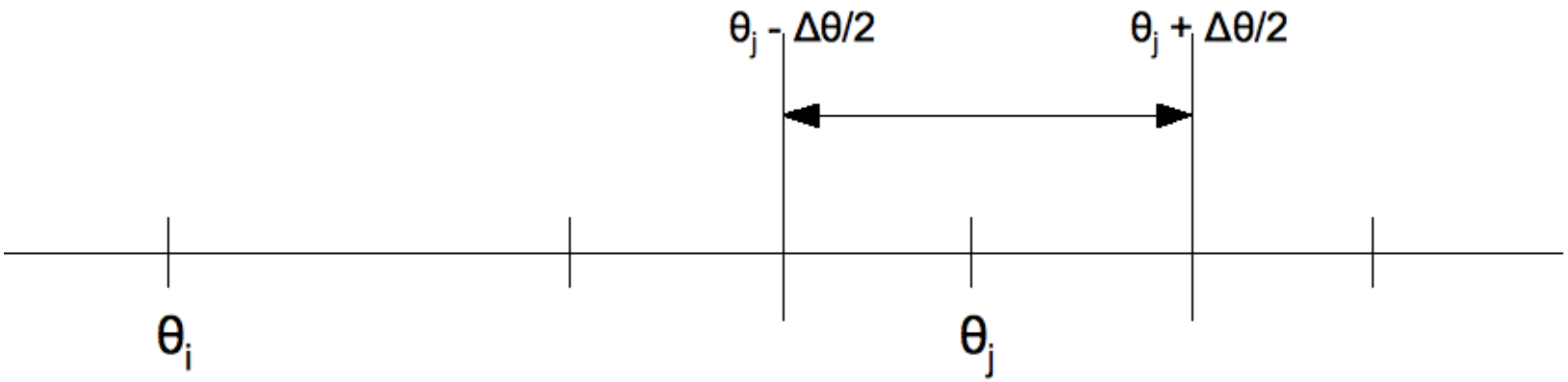
Computing the Transition Probabilities

- Next we need to know what is the probability of moving to state θ_j conditional on being in state θ_i
- Denote this probability by $p_{i,j} = P(\theta_{t+1} = \theta_j | \theta_t = \theta_i)$ and let P be the matrix of $\{p_{i,j}\}_{i=1,j=1}^{N,N}$
- Tauchen's approach: Discretize the conditional probability distribution

Computing the Transition Probabilities

- Next we need to know what is the probability of moving to state θ_j conditional on being in state θ_i
- Denote this probability by $p_{i,j} = P(\theta_{t+1} = \theta_j | \theta_t = \theta_i)$ and let P be the matrix of $\{p_{i,j}\}_{i=1,j=1}^{N,N}$
- Tauchen's approach: Discretize the conditional probability distribution
- Suppose, the current state is θ_i. What is the probability of moving to θ_j?

Computing the Transition Probabilities



The realization θ' lies within the specified interval with probability[1]

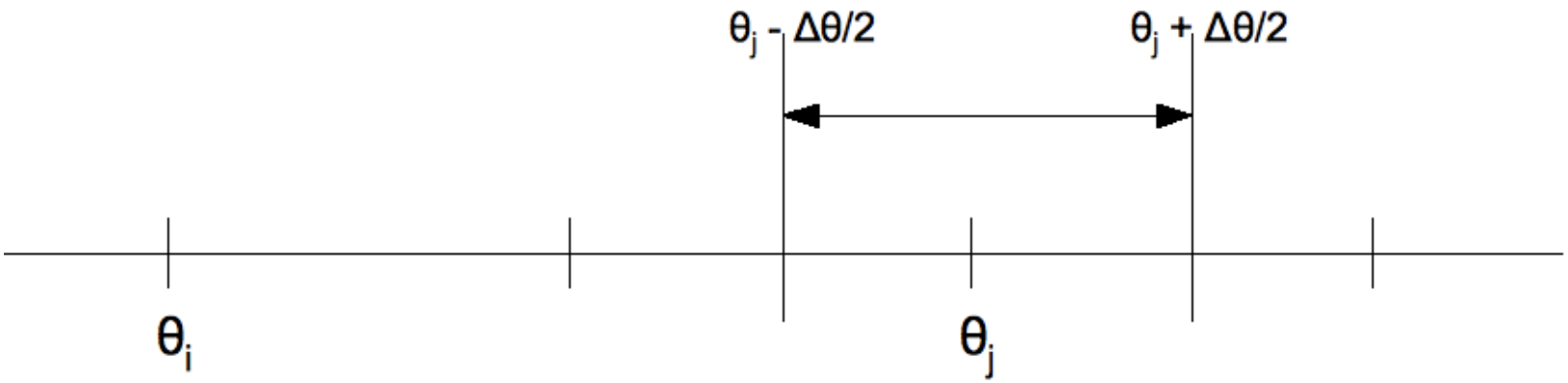
$$p_{i,j} = Prob\left(\quad \theta_j - \frac{\Delta\theta}{2} \quad \leq \quad (1-\rho)\mu + \rho\theta_i + \varepsilon' \quad \leq \quad \theta_j + \frac{\Delta\theta}{2} \quad \right)$$

rearranging yields the expression

$$Prob\left(\quad \theta_j - \frac{\Delta\theta}{2} - (1-\rho)\mu - \rho\theta_i \leq \quad \varepsilon' \quad \leq \theta_j + \frac{\Delta\theta}{2} - (1-\rho)\mu - \rho\theta_i \quad \right)$$

[1] $\Delta\theta = (\theta_N - \theta_1)(N-1)^{-1}$ is the width between two grid points.

Computing the Transition Probabilities



which is equivalent to[2]

$$p_{i,j} = F\left(\frac{\theta_j + \Delta\theta/2 - (1-\rho)\mu - \rho\theta_i}{\sigma}\right) - F\left(\frac{\theta_j - \Delta\theta/2 - (1-\rho)\mu - \rho\theta_i}{\sigma}\right)$$

The transition to the corner points have to be treated differently

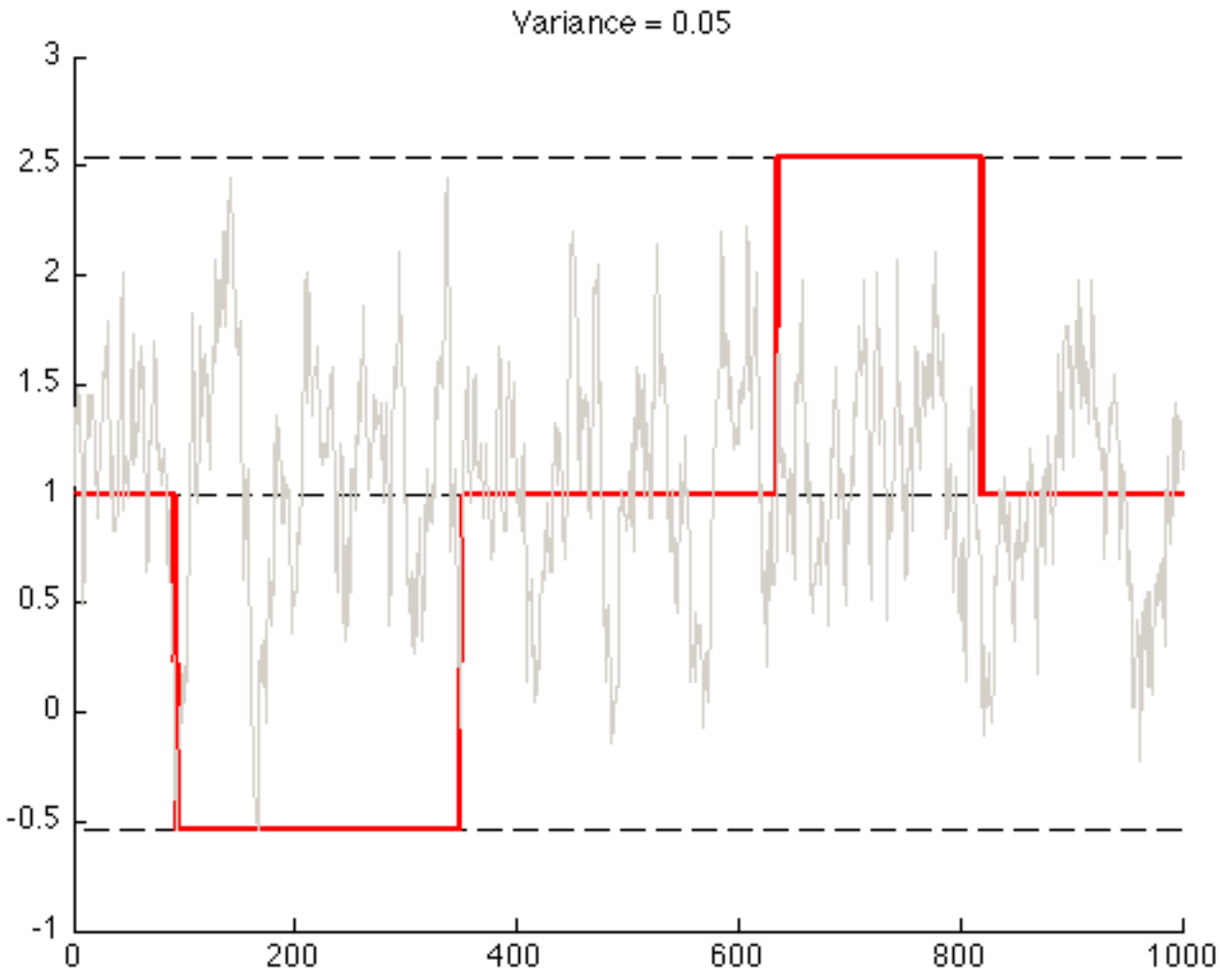
$$p_{i,1} = F\left(\frac{\theta_1 - (1-\rho)\mu - \rho\theta_i + \Delta\theta/2}{\sigma}\right) \qquad p_{i,N} = 1 - F\left(\frac{\theta_N - (1-\rho)\mu - \rho\theta_i - \Delta\theta/2}{\sigma}\right)$$

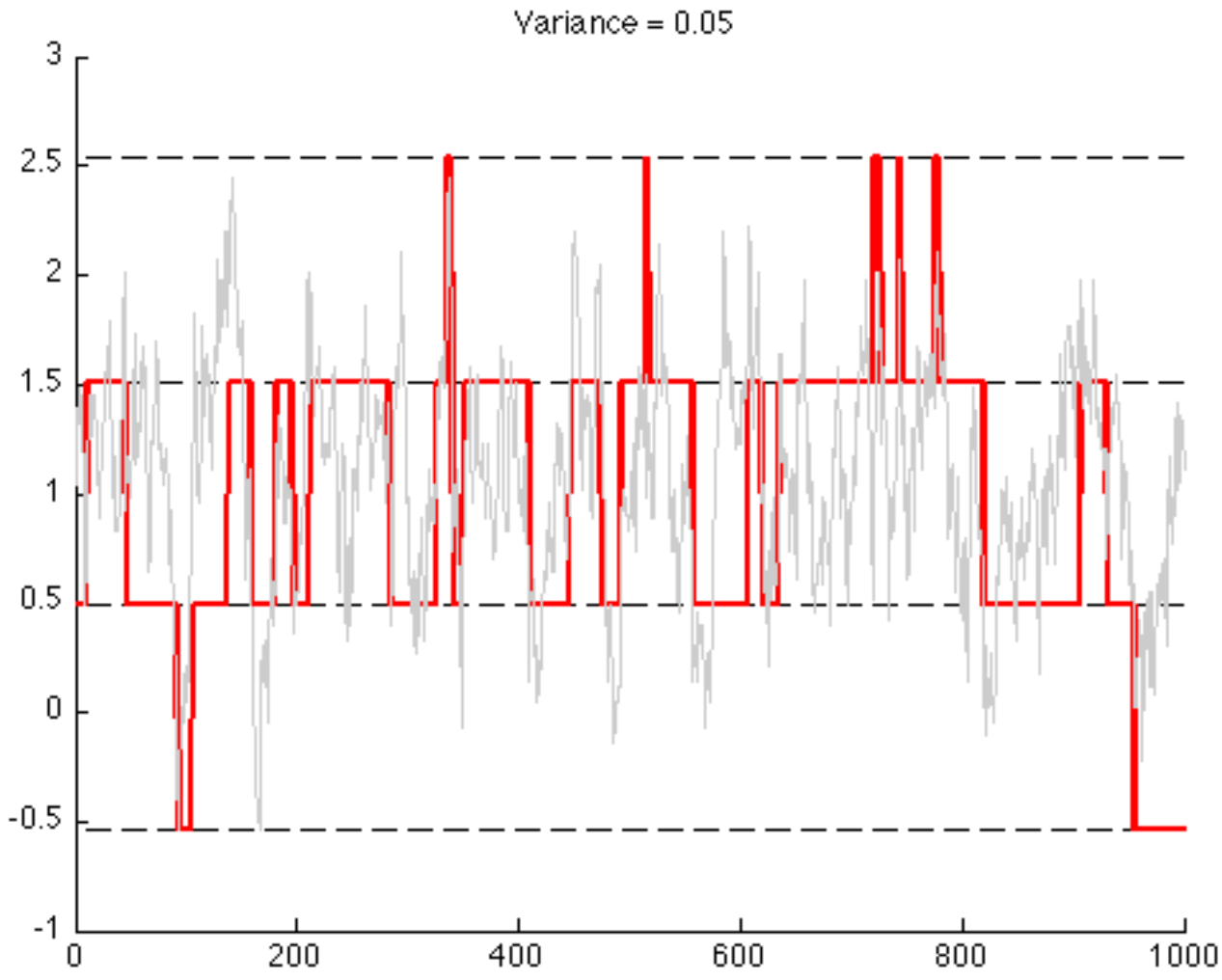
[2]Where F is the cumulative density of the standard normal distribution.

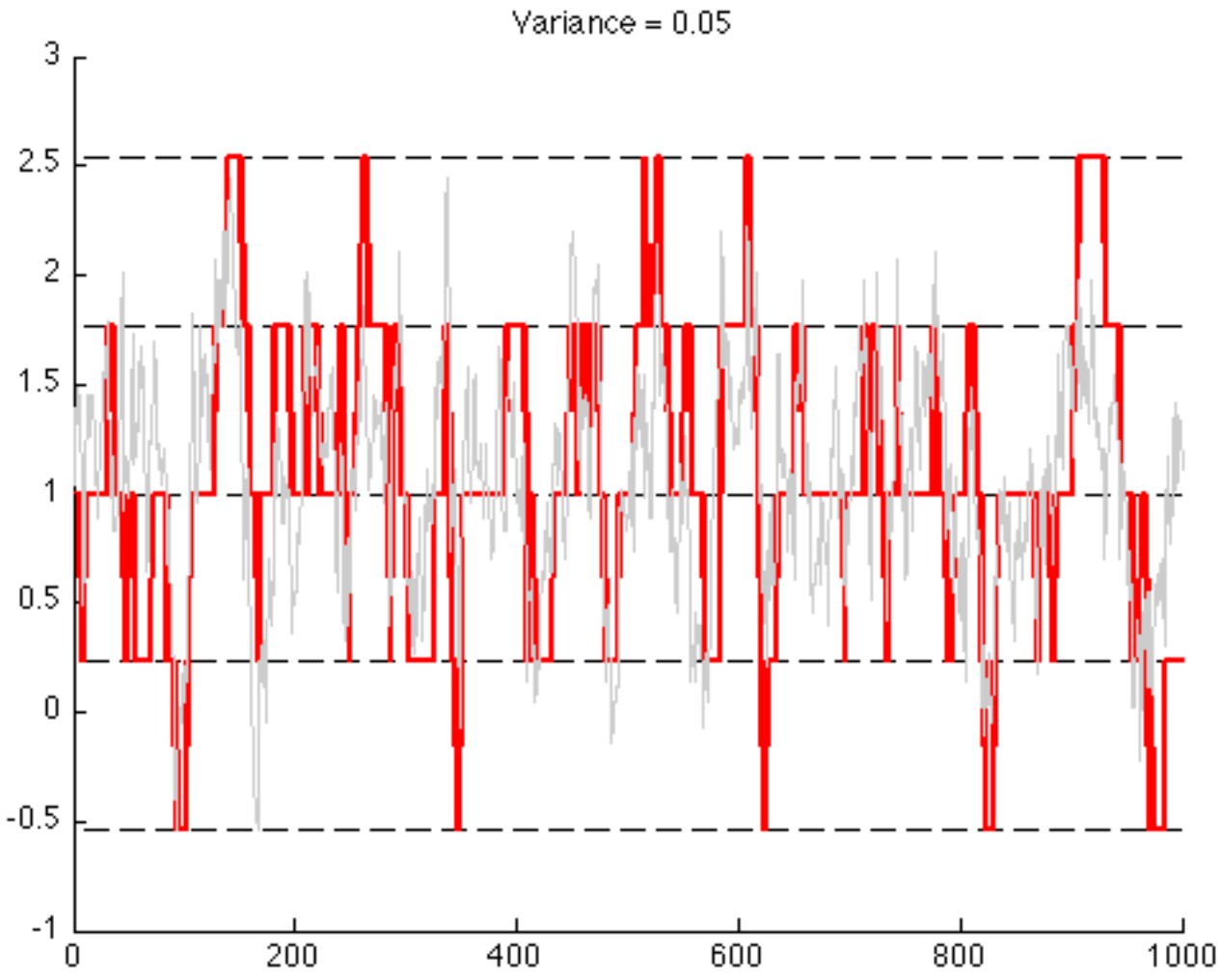
- Compute the transition probabilities $p_{i,j}$ for all i and j
- The result of this discretization is a grid $\{\theta_i\}_{i=1}^N$ and the transition probability matrix P
- Example: $N = m = 3$, $\mu = 1$, $\rho = 0.9$, $\sigma^2 = 0.05$

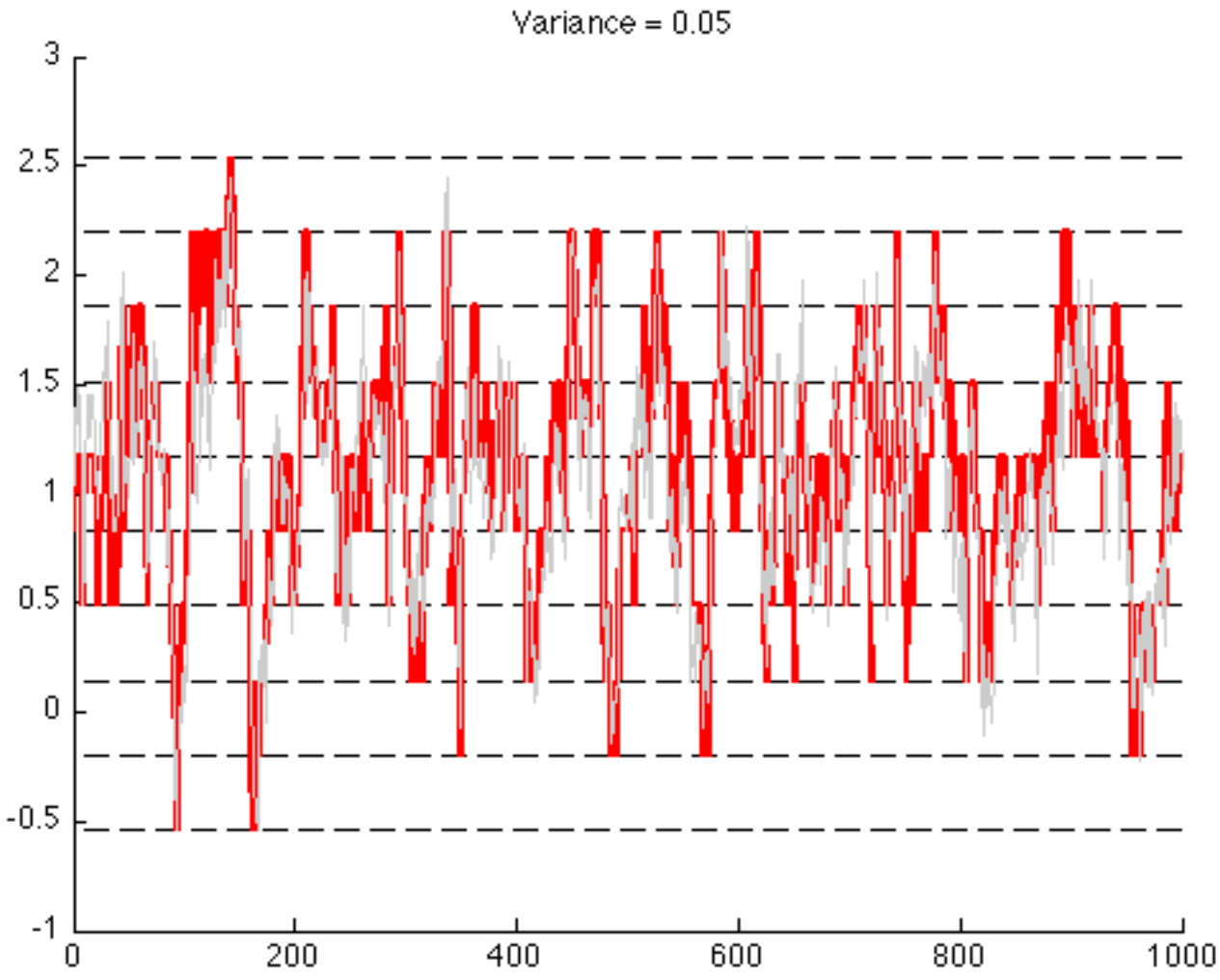
$$\{\theta_i\}_{i=1}^N = \begin{bmatrix} 2.54 \\ 1 \\ -2.54 \end{bmatrix} \qquad P = \begin{pmatrix} 0.9970 & 0.0030 & 0 \\ 0.0003 & 0.9994 & 0.0003 \\ 0.0000 & 0.0030 & 0.9970 \end{pmatrix}$$

- Let's look at the time path of the simulated Markov chain (for $N = 3, 4, 5, 10$)


Variance = 0.05
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000


Variance = 0.05
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000


Variance = 0.05
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000


Variance = 0.05
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
200
400
600
800
1000

Rouwenhorst's Method

- Recently re-introduced by Kopecky and Suen (RED 2010)
- Does not rely on the discretization of the conditional distribution of θ
- Step 1: Choose grid size N and compute the end points of the grid

$$\theta_N = \sigma_\theta \sqrt{N-1} \qquad \theta_1 = -\theta_N \qquad \text{where} \quad \sigma_\theta^2 = \frac{\sigma^2}{1-\rho^2}$$

- Step 2: Compute the transition matrix P recursively

$$P_N = p \begin{bmatrix} P_{N-1} & \mathbf{0} \\ \mathbf{0}' & 0 \end{bmatrix} + (1-p) \begin{bmatrix} \mathbf{0} & P_{N-1} \\ 0 & \mathbf{0}' \end{bmatrix} + (1-p) \begin{bmatrix} \mathbf{0}' & 0 \\ P_{N-1} & \mathbf{0} \end{bmatrix} + p \begin{bmatrix} 0 & \mathbf{0}' \\ \mathbf{0} & P_{N-1} \end{bmatrix}$$

where $p = \frac{1+\rho}{2}$, $P_2 = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix}$ and $\mathbf{0}$ is an $(N-1) \times 1$ column vector of zeros

Rouwenhorst's Method

- Tauchen's method might not be the best choice when the underlying stochastic process is very persistent, i.e. $|\rho|$ is close to 1.
- Issue: When ρ is close to 1 $\rightarrow$ process is almost a random walk.
- Rouwenhorst's methods turns out to be very reliable in that case
- Kopecky and Suen (2010) compare the performance of existing methods

Back to Our Example

Remeber our example:

$$V(k,z) = \max_{c,k'} u(c) + \beta \sum_{z'} \pi(z'|z) V(k',z')$$

s.t.

$$c + k' = zf(k) + (1-\delta)k$$

with $k, z \in \mathscr{K} \times \mathscr{Z}$

How to solve this with the discretized grids?

A Typical Solution

Algorithm for solving this Bellman equation:

1. Make a guess on $V_0(k,z)\ \forall\ k,z$ (on the grid). Typically, $V_0 = 0$ since increasing and concave (important if exploiting monotonicity or concavity below). Also, this gives a finite-time interpretation. Let i = 0.
2. For each k,z, compute $V_{i+1}(k,z)$ by either
   - 2.1 Brute-force grid search, or
   - 2.2 Exploiting monotonicity, or
   - 2.3 Exploiting monotonicity and concavity
3. Check whether the supnorm ε,

$$\varepsilon = \max_{k,z} |V_i(k,z) - V_{i+1}(k,z)|,$$

   has $\varepsilon < \mathit{toler}$. Typically, $\mathit{toler} \approx 10^{-5}$.
   - 3.1 If it is, stop and treat V_{i+1} as V.
   - 3.2 If not, increment i and go to 2.

Brute Force Grid Search

Brute force grid search is checking every $k' \in \mathscr{K}$ value and taking the maximum. (Be careful with feasibility!)

Advantages

1. Always works — can handle discontinuities, non-convex budget constraints, non-concave objective functions.
2. Some problems are naturally discrete choice.
3. Easy for compilers to optimize and easy to make parallel.

Disadvantages

1. Very slow relatively.
2. Suffers greatly from the "curse of dimensionality."
   - E.g., here, doubling $\#\mathscr{K}$ and $\#\mathscr{Z}$ requires an eight-fold increase in computational cost.
3. Most problems allow faster algorithms.

Exploiting Monotonicity

Very often, one has policy functions that are monotone, e.g.,

$$k'(k_2, z) \geq k'(k_1, z) \quad \forall\, k_2 \geq k_1 \text{ and } z$$

is monotone increasing in capital. Why?

- In macro, we usually ask how much is being saved.
- Wealthier people tend to save more than poorer people (in levels is all that's required).

This can exploited when finding the optimal $k'(k, z)$:

1. Compute $k'(k_1, z)$ by checking all $k_1, \ldots, k_N$. Set $i = 1$.
2. Compute $k'(k_{i+1}, z)$ by checking all $k'(k_i, z), \ldots, k_N$.
3. If $i + 1 = N$, stop. O/w, increment i and go to 2.

Exploiting Monotonicity

Advantages

1. Typically much faster than brute-force
2. Applies to many models
3. Can still parallelize easily, e.g., over the `iZ` values

Disadvantages

1. Dangerous if not sure that monotonicity holds. If it doesn't hold, then VFI may not converge or may converge to something wrong. This danger can be greatly mitigated by either
   - assume it holds but check the supnorm after a final brute-force search
   - explore monotonicity every, say, 9 out of 10 iterations
   - randomly check (say $\approx$ every 1000th loop over states) that the policy found using monotonicity agrees with brute force.

Note: if monotonicity almost holds, you can "relax" the monotonicity assumption—e.g., set `iKp_lb=kpi(iK-1,iZ)-nK/5`. The warning still applies.

Exploiting Concavity

In many applications one can also exploit concavity of the objective function. That is, if

$$H(k') := u(c(k,z,k')) + \beta E_z V(k',z')$$

is concave in k' (for all k,z), then the following picture is typical:

[INSERT CONCAVE FUNCTION / USE YOUR IMAGINATION]

Exploiting Concavity

How can this be exploited?

1. Compute $H(k' = k_1)$ be a known value. Let $i = 1$.
2. If $i = N$ or $H(k_{i+1}) < H(k_i)$, then stop. The optimum is $k' = k_i$. O/w, increment i and go to 2.

Note: this can be easily paired with exploiting monotonicity (which essentially gives you an initial i).

Exploiting Concavity

Disadvantages

1. Not as generally applicable as monotonicity.
2. Like monotonicity, dangerous if not sure it holds. If it doesn't hold, then VFI may not converge or may converge to something wrong. You can mitigate the danger with previously mentioned techniques.
3. When working with a concave function, can use FOCs instead which may be even faster. However, grid search w/ speedups is as reliable as it gets.

Hint: ensure your V_0 guess is consistent with your assumptions!

Using the FOC

- If you have concavity ***and*** the problem is smooth, we can use the FOC to find the policy function:

$$u_c(f(k)+(1-\delta)k-k') = \beta E\left\{V_k(k',z')\right\}$$

or

$$0 = -u_c(f(k)+(1-\delta)k-k') + \beta E\left\{V_k(k',z')\right\}$$

That is, instead of maximizing, we find the zero of the FOC.

- Since there's no maximization to be performed, this may be much faster to compute.

Using the FOC

Issues with using the FOC:

- You of course must know whether the FOC characterizes the solution (e.g. concavity, etc)
- Since you most likely don't have a closed form solution for k', you'll need to find the zero of the FOC numerically (root-finding). We'll talk about this soon.
- Since the optimal k' will most likely not fall on the grid, we need to find a way of computing the value function at points that are not on the grid (interpolation). We'll talk about this soon.
- Since you don't know what the value function is (I mean, what the hell are we doing here?), you need to come up with a way of finding its derivative. That is, we must compute the derivative, possibly numerically. We'll talk about this soon.

Another Trick: Multigrid

- Basic idea: solve first a problem in a coarser grid and use it as a guess for more refined solution
- It works like such:

1. Let there be an initial grid $\mathscr{K}_0$. Let $i = 0$.
2. Compute the optimal value V on $\mathscr{K}_i$. If the grid $\mathscr{K}_i$ is "fine enough," stop.
3. Define a finer grid $\mathscr{K}_{i+1}$ and interpolate V from $\mathscr{K}_i$ onto $\mathscr{K}_{i+1}$. (We will discuss interpolation soon).
4. Go to 2.

Yet Another Trick: Accelerator

- Maximization is the most expensive part of value function iteration
- Often, while we update the value function, optimal choices are not changing
- This suggests a simple strategy: apply the max operator only from time to time
- How do we choose the optimal timing of the max operator?

Endogenous Grid Method

One more trick: Carroll's (2006) Endogenous Grid Method (EGM).

- Very fast.
- Very accurate (much more accurate than grid search for the same number of points).
- Can be used in life-cycle models.
- Fairly easy if labor is not a choice variable.

Endogenous Grid Method

To simplify life, let's assume a problem with an easier FOC:

$$V(a,e) = \max_{a'} u(c) + \beta \mathbb{E}_e V(a',e')$$
$$\text{s.t. } c + qa' = we + a$$

This is a very common heterogeneous-agent problem.
Note: a slight modification could have e be a vector including, among other things, an age component.

Endogenous Grid Method

If u is concave, we have the following FOC:

$$u'(c)q = \beta E u'(c')$$

Endogenous Grid Method

If u is concave, we have the following FOC:

$$u'(c)q = \beta E u'(c')$$

EGM asks the following:

- Suppose a' is in some discrete grid $\mathscr{A} = \{a_1 = \underline{A}, \ldots, a_n = \overline{A}\}$.
- What must a be for this a' to be optimal?

Endogenous Grid Method

First, take a' in $\mathscr{A}$. Question:
What must a be for this to be optimal? From the FOC, we can solve for a directly:

$$
\begin{aligned}
u'(c)q &= \beta \mathbb{E} u'(c(a', e'), e') \\
u'(we + a - qa')q &= \beta \mathbb{E} u'(c(a', e'), e') \\
we + a - qa' &= u'^{-1}\left(\beta / q \mathbb{E} u'(c(a', e'), e')\right) \\
a &= -we + qa' + u'^{-1}\left(\beta / q \mathbb{E} u'(c(a', e'), e')\right)
\end{aligned}
$$

Let this a be denoted $a_{i,egm}$.

Endogenous Grid Method

Repeating this for every a' in the grid gives:

$$a'(a_{i,egm}, e) = a_i$$

for $i = 1, \ldots, n$. These are points on the asset policy function defined on an endogenous grid $\mathscr{A}_{egm} = \{a_{1,egm}, \ldots, a_{n,egm}\}$.

Note: this grid should be strictly increasing! If it is not, you probably gave a bad guess on the consumption policy.

Endogenous Grid Method

Repeating this for every a' in the grid gives:

$$a'(a_{i,egm}, e) = a_i$$

for $i = 1, \ldots, n$. These are points on the asset policy function defined on an endogenous grid $\mathscr{A}_{egm} = \{a_{1,egm}, \ldots, a_{n,egm}\}$.

Note: this grid should be strictly increasing! If it is not, you probably gave a bad guess on the consumption policy.

What about policy function off the endogenous grid? Interpolate!

Endogenous Grid Method

The full EGM can be outlined as follows:

1. Make a guess on the consumption policy $c_0(a, e)$ for all values on $\mathscr{A}$ (and the e grid). Let $t = 0$.
2. Compute the endogenous grid $\mathscr{A}_{egm} = \{a_{i,egm}\}$ giving $a'(a_{i,egm}, e) = a_i$ for all $a_i \in \mathscr{A}$.
3. Recover the asset policy on the exogenous grid $\mathscr{A}$ getting $a'(a, e)$ for all $a \in \mathscr{A}$ (using the previous slide).
4. Compute an update on the consumption policy using $c_{t+1}(a, e) = we + a - a'(a, e)q$ for all $a \in \mathscr{A}$.
5. Check the difference between c_{t+1} and c_t. If sufficiently small, stop. O/w, increment i and go to 2.

Endogenous Grid Method

Advantages

1. Faster than any grid search because it directly exploits the optimality condition.
2. Can handle occasionally binding constraints.
3. Much more accurate than grid search because (1) the asset policy is not restricted to take values on a grid and (2) directly works with Euler equation.

Endogenous Grid Method

Disadvantages

1. Requires concavity and monotonicity so other fast methods are available.
2. Doesn't generalize naturally to endogenous labor supply, but can be done (see Barillas and Fernandez-Villaverde 2007).
3. A decent initial guess is required. Generally, an increasing, linear policy works if the slope is not crazy.
4. No theorems about convergence.

Changing the Problem: Cash-at-hand Formulations

Many times the largest speed improvements come not from different algorithms but from a reformulated problem, the most common of which is a cash-at-hand. Consider the following setup:

- Infinitely-lived households
- TFP shock z follows Markov-chain, output is zk^{α}.
- Save in a discount bond b' with price q.
- Save also in capital k'.
- Period utility function u, discount β.

Cash-at-hand Formulations

To solve this directly, one has 2 continuous state variables and 1 discrete state variable.

$$V(b,k,z) = \max_{b',k'} u(c) + \beta \sum_{z'} V(b',k',z') F(z'|z)$$

$$\text{s.t. } c + qb' + k' = b + zk^{\alpha} + (1-\delta)k$$

$$c \geq 0, k' \geq 0, b' \geq -\underline{B}$$

Cash-at-hand Formulations

However, a cash-at-hand reformulation is a much easier problem:

$$\tilde{V}(x,z) = \max_{b',k'} u(c) + \beta \sum_{z'} \tilde{V}(x'(b',k',z'),z')F(z'|z)$$

$$\text{s.t. } c + qb' + k' = x$$

$$x'(b',k',z') = b' + z'k'^{\alpha} + (1-\delta)k'$$

$$c \geq 0, k' \geq 0, b' \geq -\underline{B}$$

Now there is only 1 continuous state variable and 1 discrete state. This is a massive improvement:

- If you would have had 250 points for b and k, this code would be around 250 times faster!

Cash-at-hand Formulations

Very easy to go back to the original problem once one has the value $\tilde{V}$ and policies $\tilde{b}', \tilde{k}', \tilde{c}$. Define

$$x(b, k, z) = b + zk^{\alpha} + (1 - \delta)k.$$

Then

- $V(b, k, z) = \tilde{V}(x(b, k, z), z)$
- $c(b, k, z) = \tilde{c}(x(b, k, z), z)$
- $k'(b, k, z) = \tilde{k}'(x(b, k, z), z)$
- $b'(b, k, z) = \tilde{b}'(x(b, k, z), z)$

Note: this step usually requires interpolation.

Cash-at-hand Formulations

Advantages

- Can provide massive speedups (100-250+).
- Very often applicable.

Disadvantages

- Computing expectation becomes more costly.
- Have to do interpolation (extra coding step).
- Have to think about an appropriate grid for x.

Euler Equation Errors

A valid criticism of computational economics is that it is often a black box:

- Code produces some results which cannot be verified theoretically.
- Often the model mechanics are not clear (often because the paper is poorly written).
- Often not clear that a good solution has been found.

Euler equation errors address the last point by assessing the accuracy of an approximation. Since almost all macro models have some sort of Euler equation, these are very useful.

Euler Equation Errors

In our simplified model, the Euler equation reads

$$0 = u'(c(a,e))q - \beta \mathbb{E}_e u'(c(a'(a,e),e')).$$

For the true optimal policies, this would hold exactly. However, our computed optimal policies will result in this not holding exactly. Consequently, we can use the Euler equation to assess the error. Define the Euler equation error (EEE) in log10 as

$$EEE(a,e) = \log_{10} \left| 1 - \frac{u'^{-1}(\frac{\beta}{q}\mathbb{E}_e u'(c(a'(a,e),e')))}{c(a,e)} \right|.$$

Note: some authors don't use the log10 formulation.

Euler Equation Errors

$$EEE(a,e) = \log_{10}\left|1 - \frac{u'^{-1}(\frac{\beta}{q}\mathbb{E}_e u'(c(a'(a,e),e')))}{c(a,e)}\right|.$$

Why write it in this way?

- Unit-free in that consumption and assets could be scaled up by a constant factor without changing the measure.
- Nice interpretation: $EEE(a,e) = x$ means a \$1 mistake is made for every $\$10^{-x}$ spent. So, $EEE = -6$ means a \$1 dollar mistake is made for every \$1,000,000 spent, which in my book is very accurate.

Euler Equation Errors

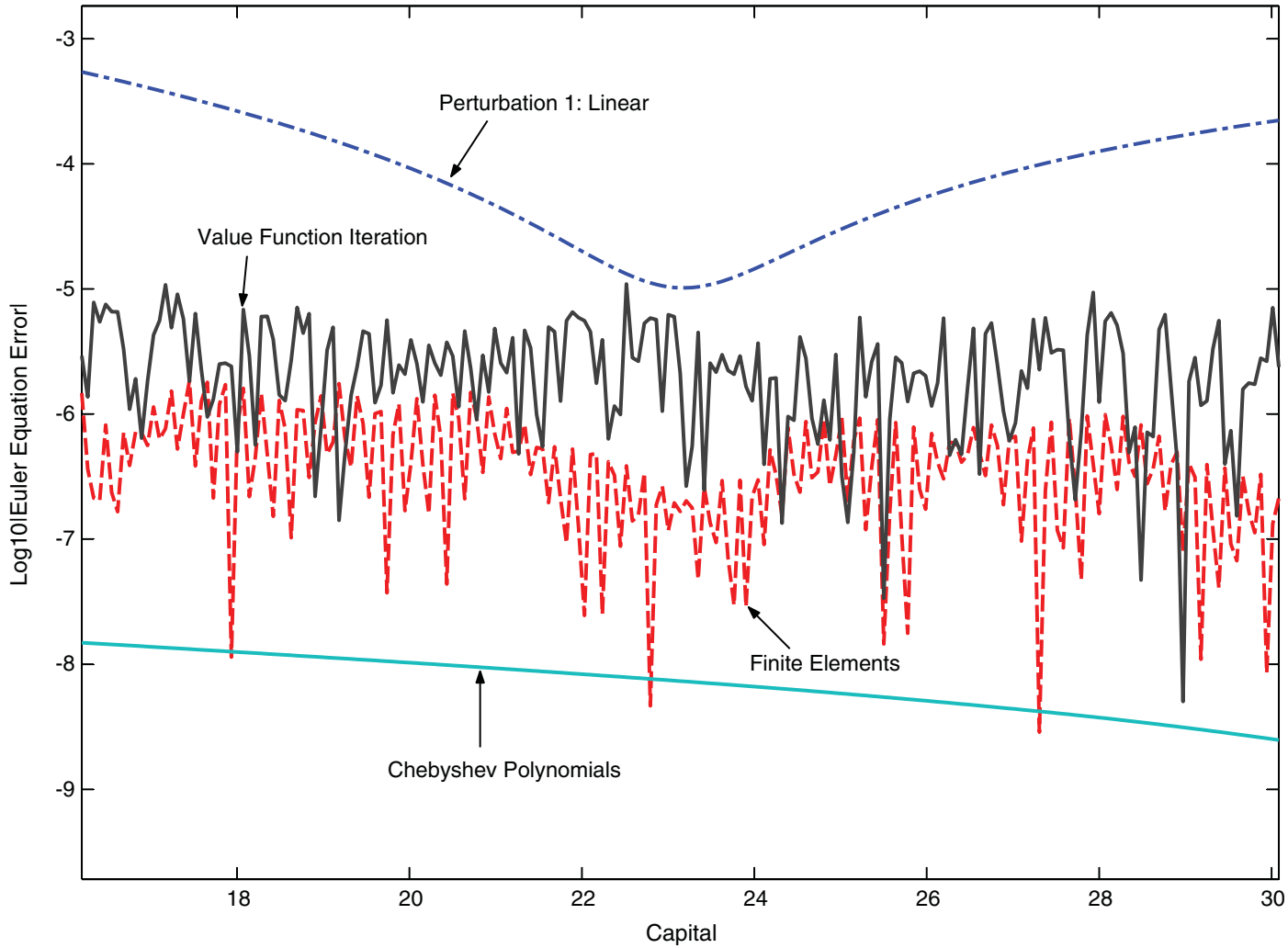


Fig. 9. Euler equation errors at $z = 0$, $\tau = 2/\sigma = 0.007$.

Reference: Aruoba, Fernandez-Villaverde, Rubio-Ramirez (2006).

Euler Equation Errors

Final comments about EEE:

- Best measure of accuracy we have.
- Requires extra work to compute these things, but will reveal flaws in your code.
- Only applies when the optimal policy is not at a binding constraint.
- The patterns should look like in the preceding picture (note: their VFI had something like million points, so your levels will be higher).