

Value Function Iteration in Matlab

Michael Kotrous

John Munro Godfrey, Sr. Department of Economics

University of Georgia

ECON 8040: Macroeconomic Theory I

Fall 2025

Matlab

- ▶ Popular programming language among economists
- ▶ Advantages:
 - Efficient matrix math
 - Built-in functionality
 - Simple syntax
 - Documentation
- ▶ Disadvantages:
 - Proprietary
 - Limited add-ons
 - Non-numeric data

Getting Started

- ▶ [UGA installation guide](#)
- ▶ Useful add-ons
 - Optimization
 - Parallel Computing
- ▶ Useful Links
 - [Matlab documentation](#)
 - eLC: Matlab > Example Code > `matlab_tutorial_files.zip` > `vfi`

Neoclassical Growth Model

- ▶ Infinite horizon planner's problem, written **recursively**

$$V(k) = \max_{k'} \{ \log(Ak^\alpha + (1 - \delta)k - k') + \beta V(k') \}$$

subject to

$$0 \leq k' \leq Ak^\alpha + (1 - \delta)k$$

- ▶ When $\delta = 1$, $k' = g(k) = \alpha\beta Ak^\alpha$
- ▶ When $\delta < 1$, solve numerically

Value Function Iteration

- ▶ Solve models numerically
- ▶ Solution: approximation of $V(k)$
- ▶ Method: Iterate on $V(k)$, reach fixed point
- ▶ Works b/c Bellman operator = **contraction mapping**
- ▶ Implement **grid search** in Matlab

Outline

Value Function Iteration

- Calibrate Parameters

- Discretize State Space

- Calculate Flow Utility

- Converge to Fixed Point

Policy Functions

Conclusion

VFI Algorithm

- 1 Calibrate parameters $(\alpha, \beta, \delta, \mathbf{A})$
- 2 Set tolerance $\varepsilon > 0$
- 3 Discretize state space

$$\mathcal{K} = \{k_1, k_2, \dots, k_n\}$$

- 4 Calculate flow utility $u(k, k')$ for $(k, k') \in \mathcal{K} \times \mathcal{K}$

VFI Algorithm (cont.)

- 5 Define initial guess $V_0(\mathbf{k}) = \{V_0(k_1), V_0(k_2), \dots, V_0(k_n)\}$
- 6 For each $\mathbf{k} \in \mathcal{K}$, solve

$$V_1(\mathbf{k}) = \max_{k'} \{ \log(Ak^\alpha + (1 - \delta)k - k') + \beta V_0(k') \}$$

subject to $0 \leq k' \leq Ak^\alpha + (1 - \delta)k$, $k' \in \mathcal{K}$

- 7 Calculate $\|V_1(\mathbf{k}) - V_0(\mathbf{k})\|$
- 8 Stopping criteria $\|V_{n+1}(\mathbf{k}) - V_n(\mathbf{k})\| < \varepsilon$

Calibration

Table: Calibrated Parameters, Tolerance

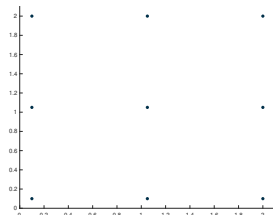
Parameter	Value(s)
α	0.39
β	0.95
δ	1
A	274

Tolerance	Value
ϵ	10^{-8}

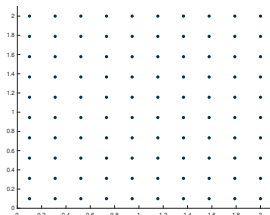
Calibration (in code)

```
1 a = 0.39;  
2 b = 0.95;  
3 d = 1;  
4 A = 274;  
5  
6 tol = 1e-8;
```

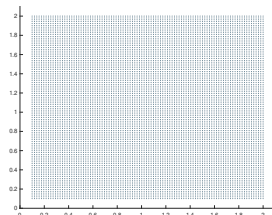
Discretize State Space



(a) 3 nodes



(b) 10 nodes



(c) 100 nodes

Increasing nodes

- ▶ ↑ precision
- ▶ ↑ compute time

Discretize State Space (in code)

```
1 n = 5;  
2 kss = ((A*a)./(1/b - 1 + d))^(1/(1-a));  
3 kgrid = griddle(0.1*kss, 2*kss, n, 1.5);
```

▶ [griddle.m](#)

Flow Utility

$$u(k, k') = \begin{cases} \log(Ak^\alpha + (1 - \delta)k - k') & \text{if } Ak^\alpha + (1 - \delta)k - k' > 0 \\ -10^{20} & \text{otherwise} \end{cases}$$

Table: Flow Utility for all $(k, k') \in \mathcal{K} \times \mathcal{K}$

		k'				
		k_1	k_2	k_3	k_4	k_5
k	k_1	7.5737	7.3024	6.4588	-1e20	-1e20
	k_2	8.0852	7.9315	7.5694	6.7369	-1e20
	k_3	8.4241	8.3171	8.0857	7.6745	6.7524
	k_4	8.6458	8.5610	8.3844	8.0966	7.5941
	k_5	8.8087	8.7371	8.5912	8.3638	8.0039

Flow Utility (in code)

```
1 ucgrid = zeros(n,n);
2 for i = 1:n
3     for j = 1:n
4         c = A*kgrid(i)^a + (1-d)*kgrid(i) - kgrid(j);
5         if c > 0
6             ucgrid(i,j) = log(c);
7         else
8             % exclude infeasible choices
9             ucgrid(i,j) = -1e20;
10        end
11    end
12 end
```

Define Initial Guess

- ▶ Solution: fixed point
- ▶ Blackwell sufficient conditions ▶ Theorem
 - Discounting
 - Monotonicity
- ▶ Bellman operator = contraction mapping
- ▶ Any initial guess works!

```
1 V = linspace(0, 1, n);  
2 Tv = zeros(1, n);  
3 g = zeros(1, n);
```

Update Guess

```
1 for i = 1:n
2     [vmax, kmax] = max(ucgrid(i,:) + b*V);
3     Tv(i) = vmax;
4     g(i) = kgrid(kmax);
5 end
```

Update Guess (cont.)

Table: Updating Value, Policy Functions

		k'					Tv	$g(k)$
		k_1	k_2	k_3	k_4	k_5		
k	k_1	7.5737	7.5399	6.9338	0.7125	0.9500	7.5737	k_1
	k_2	8.0852	8.1690	8.0444	7.4494	0.9500	8.1690	k_2
	k_3	8.4241	8.5546	8.5607	8.3870	7.7024	8.5607	k_3
	k_4	8.6458	8.7985	8.8594	8.8091	8.5441	8.8594	k_3
	k_5	8.8087	8.9746	9.0662	9.0763	8.9539	9.0763	k_4

Test for Convergence

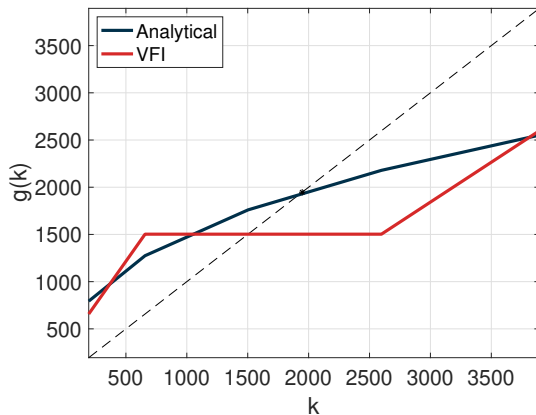
Table: Distance between Initial, Updated Guess

V	Tv	$V - Tv$
0	7.5737	-7.5737
0.25	8.1690	-7.9190
0.50	8.5607	-8.0607
0.75	8.8594	-8.1094
1	9.0763	-8.0763

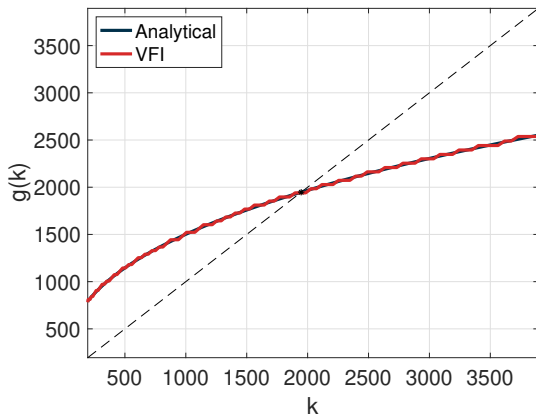
```
1 err1 = norm(V-Tv);           % 17.7774
2 err2 = abs(max(V-Tv));      % 8.1094
```

Iteration & Convergence

```
1 V = Tv; % update V
2 err = err1; % use Euclidean norm for stopping criteria
3 it = 0; % count iterations
4 while err >= tol && it < 500
5     for i = 1:n
6         [vmax, kmax] = max(ucgrid(i,:) + b*V);
7         Tv(i) = vmax;
8         g(i) = kgrid(kmax);
9     end
10    % check for convergence and update guess
11    err = norm(Tv-V);
12    V = Tv;
13    it = it+1;
14 end
```

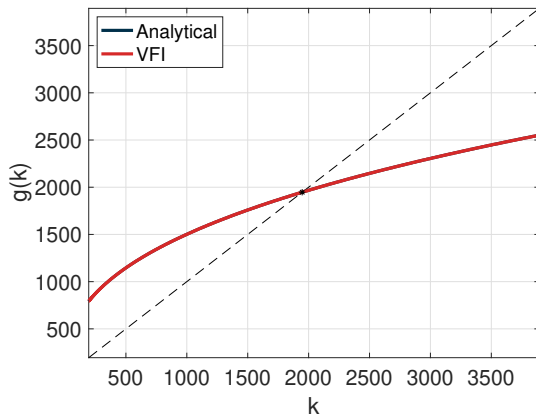
Figure: Policy Function, $\delta = 1, n = 5$ 

► Solved in 0.000481 seconds

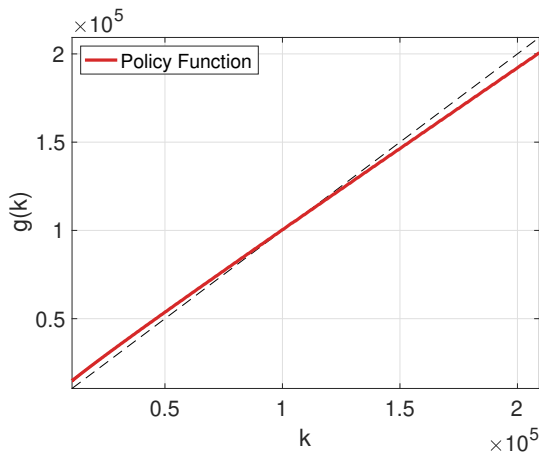
Figure: Policy Function, $\delta = 1, n = 100$ 

► Solved in 0.003190 seconds

Figure: Policy Function, $\delta = 1, n = 1,000$



- Solved in 0.003492 seconds

Figure: Policy Function, $\delta = 0.04$, $n = 1,000$ 

► Solved in 0.003404 seconds

Conclusion

- ▶ Many models can't be solved analytically
- ▶ VFI solves models numerically
- ▶ We implemented VFI in Matlab
- ▶ We restricted policy function to grid
 - Pros: simpler code, flow utility before iteration
 - Cons: inaccurate when n small, high n inefficient
- ▶ Alternative: interpolate between grid points
 - Accurate policy functions with small n
 - Useful for multi-dimensional state spaces
 - See [Karen Kopeccky](#) and [Eric Sim's](#) VFI notes

griddle.m

```
1 function g = griddle(a,b,n,p)
2     gr = zeros(1,n);
3     gr(1) = a;
4     gr(n) = b;
5     for k = 2:n-1
6         gr(k) = a + (b-a)*((k-1)/(n-1))^p;
7     end
8     g = gr;
9 end
```

Blackwell Sufficient Conditions

Let $X \subseteq \mathbb{R}^\ell$ and $B(X)$ be the space of bounded functions $f : X \rightarrow \mathbb{R}$ with d being the sup-norm. Let $T : B(X) \rightarrow B(X)$ be an operator satisfying

- 1 **Monotonicity:** If $f, g \in B(X)$ are such that $f(x) \leq g(x)$ for all $x \in X$, then $(Tf)(x) \leq (Tg)(x)$ for all $x \in X$.
- 2 **Discounting:** Let the function $f + a$, for $f \in B(X)$ and $a \in \mathbb{R}_+$, be defined as $(f + a)(x) = f(x) + a$. There exists $\beta \in (0, 1)$ such that for all $f \in B(X)$, all $a \geq 0$, and all $x \in X$

$$[T(f + a)(x)] \leq [Tf](x) + \beta a$$

Then the operator T is a contraction mapping with modulus β .