ValueFunctionIterationCodeComparison

Generated by Doxygen 1.6.1

Thu Oct 25 15:21:27 2012

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 parameters Class Reference

Object to store parameter values for VFI problem.

`#include <global.h>`

Public Member Functions

- void load (const char *)

  Function to load VFI parameter values to parameters object.

Public Attributes

- REAL eta

  Coefficient of relative risk aversion.

- REAL beta

  Time discount factor.

- REAL alpha

  Share of capital in the production function.

- REAL delta

  Rate of capital depreciation.

- REAL mu

  TFP mean.

- REAL rho

  TFP persistence.

- REAL sigma

  TFP volatility.

- REAL lambda

  Number of standard deviations for AR1 approximation.

- int nk

  Number of values in capital grid.

- int nz

  Number of values in TFP grid.

- REAL tol

  Tolerance for convergence.

- char maxtype

  Maximization method - choices are 'g' (grid) and 'b' (binary search).

- int howard

  Number of howard steps to perform between maximizations - set howard = 1 if max = 'b'.

3.1.1 Member Function Documentation

3.1.1.1 void parameters::load (const char * *fileName*)

This function is a parameters class method which loads parameter values from a text file for storage in the object. The input file must have 13 lines, each line beginning with a parameter value, followed by a comma and a character string describing the parameter. The order of the parameters must correspond to the order in the parameters class description.

Parameters:

← ***fileName*** Name of file storing parameter values.

Returns:

Void.

The documentation for this class was generated from the following files:

- global.h
- parameters.cpp

Chapter 4

File Documentation

4.1 ar1.cpp File Reference

File containing AR1 function for the VFI problem. `#include "global.h"`

`#include <math.h>`

`#include <Eigen/Dense>`

Functions

- void ar1 (const parameters &param, VectorXR &Z, MatrixXR &P)

  Function to compute discrete AR1 approximation values and transition matrix.

4.1.1 Detailed Description

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012

4.1.2 Function Documentation

4.1.2.1 void ar1 (const parameters & *param*, VectorXR & *Z*, MatrixXR & *P*)

This function that computes a discrete AR1 approximation and transition matrix using the method of Tauchen (1986).

Parameters:

← ***param*** Object of class parameters.

→ ***Z*** Grid of AR1 values.

→ ***P*** AR1 transition matrix values.

Returns:

Void.

4.2 binaryMax.cpp File Reference

File containing binary search maximization function. `#include "global.h"`

`#include <Eigen/Dense>`

`#include <math.h>`

Functions

- void binaryMax (const int &klo, const int &nksub, const REAL &ydepK, const REAL eta, const REAL beta, const VectorXR &K, const VectorXR &Exp, REAL &V, int &G)

  Function to compute maximum of Bellman objective via binary search.

4.2.1 Detailed Description

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012

4.2.2 Function Documentation

4.2.2.1 void binaryMax (const int & *klo*, const int & *nksub*, const REAL & *ydepK*, const REAL *eta*, const REAL *beta*, const VectorXR & *K*, const VectorXR & *Exp*, REAL & *V*, int & *G*)

This function finds the maximum and argmax of the Bellman objective over a specified subgrid of capital by using a binary search algorithm. The algorithm requires concavity and cannot be used with the howard improvement method.

Parameters:

← ***klo*** Lower index of the capital grid to begin search.

← ***nksub*** Number of points in the capital grid to include in search.

← ***ydepK*** value of output plus depreciated capital.

← ***eta*** Coefficient of relative risk aversion.

← ***beta*** Time discount factor.

← ***K*** Grid of capital values.

← ***Exp*** Expected value function continuation values.

→ ***V*** Updated value function.

→ ***G*** Updated policy function.

Returns:

Void.

4.3 binaryVal.cpp File Reference

File containing a function which finds the approximate location of a value in a vector with monotonically increasing values. `#include "global.h"`

Functions

- int binaryVal (const REAL &x, const VectorXR &X)

  Function to find the location of a value in a monotonic grid.

4.3.1 Detailed Description

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012

4.3.2 Function Documentation

4.3.2.1 int binaryVal (const REAL & *x*, const VectorXR & *X*)

This function finds the first value X[ix] such that x <= X[ix], where x is a scalar value, X is a monotonic array, and ix is the index of X.

Parameters:

← ***x*** Value to search for in vector X.

← ***nx*** Length of array X.

← ***X*** Vector of data to search.

Returns:

imax Integer ix (<= nx) such that x <= X[ix].

4.4 global.h File Reference

Global header file. `#include <Eigen/Dense>`

Classes

- class parameters

  Object to store parameter values for VFI problem.

Typedefs

- typedef double **REAL**
- typedef Eigen::Matrix< REAL, Eigen::Dynamic, 1 > **VectorXR**
- typedef Eigen::Matrix< REAL, 1, Eigen::Dynamic > **RowVectorXR**
- typedef Eigen::Matrix< REAL, Eigen::Dynamic, Eigen::Dynamic > **MatrixXR**
- typedef Eigen::Array< REAL, Eigen::Dynamic, 1 > **ArrayXR**
- typedef Eigen::Array< REAL, Eigen::Dynamic, Eigen::Dynamic > **ArrayXXR**

Functions

- double curr_second (void)

  Basic timer function.

- void ar1 (const parameters &param, VectorXR &Z, MatrixXR &P)

  Function to compute discrete AR1 approximation values and transition matrix.

- void kGrid (const parameters &param, const VectorXR &Z, VectorXR &K)

  Function to compute the values of an equally spaced capital grid.

- void vfInit (const parameters &param, const VectorXR &Z, MatrixXR &V)

  Function to initialize value function.

- void vfStep (const parameters &param, const bool &howard, const VectorXR &K, const VectorXR &Z, const MatrixXR &P, const MatrixXR &V0, MatrixXR &V, MatrixXi &G)

  Function to update value function.

- int binaryVal (const REAL &x, const VectorXR &X)

  Function to find the location of a value in a monotonic grid.

- void gridMax (const int &klo, const int &nksub, const REAL &ydepK, const REAL eta, const REAL beta, const VectorXR &K, const VectorXR &Exp, REAL &V, int &G)

  Function to compute maximum of Bellman objective via grid search.

- void binaryMax (const int &klo, const int &nksub, const REAL &ydepK, const REAL eta, const REAL beta, const VectorXR &K, const VectorXR &Exp, REAL &V, int &G)

  Function to compute maximum of Bellman objective via binary search.

4.4.1 Detailed Description

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012

4.4.2 Function Documentation

4.4.2.1 void ar1 (const parameters & *param*, VectorXR & *Z*, MatrixXR & *P*)

This function that computes a discrete AR1 approximation and transition matrix using the method of Tauchen (1986).

Parameters:

← ***param*** Object of class parameters.

→ ***Z*** Grid of AR1 values.

→ ***P*** AR1 transition matrix values.

Returns:

Void.

4.4.2.2 void binaryMax (const int & *klo*, const int & *nksub*, const REAL & *ydepK*, const REAL *eta*, const REAL *beta*, const VectorXR & *K*, const VectorXR & *Exp*, REAL & *V*, int & *G*)

This function finds the maximum and argmax of the Bellman objective over a specified subgrid of capital by using a binary search algorithm. The algorithm requires concavity and cannot be used with the howard improvement method.

Parameters:

← ***klo*** Lower index of the capital grid to begin search.

← ***nksub*** Number of points in the capital grid to include in search.

← ***ydepK*** value of output plus depreciated capital.

← ***eta*** Coefficient of relative risk aversion.

← ***beta*** Time discount factor.

← ***K*** Grid of capital values.
← ***Exp*** Expected value function continuation values.
→ ***V*** Updated value function.
→ ***G*** Updated policy function.

Returns:

Void.

4.4.2.3 int binaryVal (const REAL & *x*, const VectorXR & *X*)

This function finds the first value X[ix] such that x <= X[ix], where x is a scalar value, X is a monotonic array, and ix is the index of X.

Parameters:

← ***x*** Value to search for in vector X.
← ***nx*** Length of array X.
← ***X*** Vector of data to search.

Returns:

imax Integer ix (<= nx) such that x <= X[ix].

4.4.2.4 curr_second (void)

Returns:

Double precision value representing time.

4.4.2.5 void gridMax (const int & *klo*, const int & *nksub*, const REAL & *ydepK*, const REAL *eta*, const REAL *beta*, const VectorXR & *K*, const VectorXR & *Exp*, REAL & *V*, int & *G*)

This function finds the maximum and argmax of the Bellman objective function by using a naive grid search: computing the utility at each value of the grid.

Parameters:

← ***klo*** Lower index of the capital grid to begin search.
← ***nksub*** Number of points in the capital grid to include in search.
← ***ydepK*** value of output plus depreciated capital.
← ***eta*** Coefficient of relative risk aversion.
← ***beta*** Time discount factor.
← ***K*** Grid of capital values.
← ***Exp*** Expected value function continuation values.
→ ***V*** Updated value function.
→ ***G*** Updated policy function.

Returns:

Void.

4.4.2.6 void kGrid (const parameters & *param*, const VectorXR & *Z*, VectorXR & *K*)

This function computes an equally spaced capital grid. The upper and lower bounds are the deterministic steady-state values of capital at the highest and lowest values of the TFP process (respectively), scaled by 0.95 and 1.05 (respectively).

Parameters:

← ***param*** Object of class parameters.

← ***Z*** Grid of TFP values.

→ ***K*** Grid of capital values.

Returns:

Void.

4.4.2.7 void vfInit (const parameters & *param*, const VectorXR & *Z*, MatrixXR & *V*)

This function initializes the value function at the deterministic steady state values for each level of TFP: conditional on a TFP level, the deterministic steady-state value of capital is computed, as well as the associated value function value.

Parameters:

← ***param*** Object of class parameters.

← ***Z*** Grid of TFP values.

→ ***V*** Matrix of value function values.

Returns:

Void.

4.4.2.8 void vfStep (const parameters & *param*, const bool & *howard*, const VectorXR & *K*, const VectorXR & *Z*, const MatrixXR & *P*, const MatrixXR & *V0*, MatrixXR & *V*, MatrixXi & *G*)

This function performs one iteration of the value function iteration algorithm, using V0 as the current value function and either maximizing the LHS of the Bellman if howard = false or using the concurrent policy function as the argmax if howard = true. Maximization is performed by either gridMax or binaryMax.

Parameters:

← ***param*** Object of class parameters.

← ***howard*** Indicates if the current iteration of the value function will perform a maximization (false) or if it will simply compute the new value function using the concurrent policy function (true).

← ***K*** Grid of capital values.

← ***Z*** Grid of TFP values.

← ***P*** TFP transition matrix.

← ***V0*** Matrix storing current value function.

→ ***V*** Matrix storing updated value function.

↔ ***G*** Matrix storing policy function (updated if howard = false).

Returns:

Void.

4.5 gridMax.cpp File Reference

File containing grid search maximization function. `#include "global.h"`

```
#include <Eigen/Dense>
#include <math.h>
#include <iostream>
```

Functions

- void gridMax (const int &klo, const int &nksub, const REAL &ydepK, const REAL eta, const REAL beta, const VectorXR &K, const VectorXR &Exp, REAL &V, int &G)

  Function to compute maximum of Bellman objective via grid search.

4.5.1 Detailed Description

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012

4.5.2 Function Documentation

4.5.2.1 void gridMax (const int & *klo*, const int & *nksub*, const REAL & *ydepK*, const REAL *eta*, const REAL *beta*, const VectorXR & *K*, const VectorXR & *Exp*, REAL & *V*, int & *G*)

This function finds the maximum and argmax of the Bellman objective function by using a naive grid search: computing the utility at each value of the grid.

Parameters:

← ***klo*** Lower index of the capital grid to begin search.

← ***nksub*** Number of points in the capital grid to include in search.

← ***ydepK*** value of output plus depreciated capital.

← ***eta*** Coefficient of relative risk aversion.

← ***beta*** Time discount factor.

← ***K*** Grid of capital values.

← ***Exp*** Expected value function continuation values.

→ ***V*** Updated value function.

→ ***G*** Updated policy function.

Returns:

Void.

4.6 kGrid.cpp File Reference

File containing function to create capital grid. `#include "global.h"`

`#include <math.h>`

`#include <Eigen/Dense>`

Functions

- void kGrid (const parameters &param, const VectorXR &Z, VectorXR &K)

  Function to compute the values of an equally spaced capital grid.

4.6.1 Detailed Description

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012

4.6.2 Function Documentation

4.6.2.1 void kGrid (const parameters & *param*, const VectorXR & *Z*, VectorXR & *K*)

This function computes an equally spaced capital grid. The upper and lower bounds are the deterministic steady-state values of capital at the highest and lowest values of the TFP process (respectively), scaled by 0.95 and 1.05 (respectively).

Parameters:

← ***param*** Object of class parameters.

← ***Z*** Grid of TFP values.

→ ***K*** Grid of capital values.

Returns:

Void.

4.7 main.cpp File Reference

File containing main function for the VFI problem. `#include "global.h"`

```
#include <math.h>
#include <ctime>
#include <typeinfo>
#include <Eigen/Dense>
#include <iostream>
#include <fstream>
```

Functions

- int main ()

  Main function for the VFI problem.

4.7.1 Detailed Description

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012

4.7.2 Function Documentation

4.7.2.1 main ()

This function solves a standard neoclassical growth model with value function iteration on a CPU.

See Aldrich, Eric M., Jesus Fernandez-Villaverde, A. Ronald Gallant and Juan F. Rubio-Ramirez (2011), Tapping the supercomputer under your desk: Solving dynamic equilibrium models with graphics processors, Journal of Economic Dynamics & Control, 35, 386-393.

Returns:

0 upon successful completion, 1 otherwise.

4.8 parameters.cpp File Reference

File containing parameters class method for loading VFI parameter values. `#include "global.h"`

`#include <stdlib.h>`

`#include <vector>`

`#include <fstream>`

4.8.1 Detailed Description

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012

4.9 timer.cpp File Reference

File containing basic timer function. `#include <stddef.h>`

`#include <sys/time.h>`

Functions

- double curr_second (void)

  Basic timer function.

4.9.1 Detailed Description

Author:

Kyle Spafford

Date:

19 November 2010

Public domain.

4.9.2 Function Documentation

4.9.2.1 double curr_second (void)

Returns:

Double precision value representing time.

4.10 vfInit.cpp File Reference

File containing function to initialize the value function. `#include "global.h"`

```
#include <math.h>
#include <Eigen/Dense>
#include <iostream>
```

Functions

- void vfInit (const parameters &param, const VectorXR &Z, MatrixXR &V)

  Function to initialize value function.

4.10.1 Detailed Description

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012

4.10.2 Function Documentation

4.10.2.1 void vfInit (const parameters & *param*, const VectorXR & *Z*, MatrixXR & *V*)

This function initializes the value function at the deterministic steady state values for each level of TFP: conditional on a TFP level, the deterministic steady-state value of capital is computed, as well as the associated value function value.

Parameters:

← ***param*** Object of class parameters.

← ***Z*** Grid of TFP values.

→ ***V*** Matrix of value function values.

Returns:

Void.

4.11 vfStep.cpp File Reference

File containing main iterative step of the VFI problem. `#include "global.h"`

```
#include <math.h>
#include <iostream>
#include <typeinfo>
#include <Eigen/Dense>
#include <stdlib.h>
```

Functions

- void vfStep (const parameters &param, const bool &howard, const VectorXR &K, const VectorXR &Z, const MatrixXR &P, const MatrixXR &V0, MatrixXR &V, MatrixXi &G)

  Function to update value function.

4.11.1 Detailed Description

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012

4.11.2 Function Documentation

4.11.2.1 void vfStep (const parameters & *param*, const bool & *howard*, const VectorXR & *K*, const VectorXR & *Z*, const MatrixXR & *P*, const MatrixXR & *V0*, MatrixXR & *V*, MatrixXi & *G*)

This function performs one iteration of the value function iteration algorithm, using V0 as the current value function and either maximizing the LHS of the Bellman if howard = false or using the concurrent policy function as the argmax if howard = true. Maximization is performed by either gridMax or binaryMax.

Parameters:

← ***param*** Object of class parameters.

← ***howard*** Indicates if the current iteration of the value function will perform a maximization (false) or if it will simply compute the new value function using the concurrent policy function (true).

← ***K*** Grid of capital values.

← ***Z*** Grid of TFP values.

← ***P*** TFP transition matrix.

← ***V0*** Matrix storing current value function.

→ ***V*** Matrix storing updated value function.

↔ ***G*** Matrix storing policy function (updated if howard = false).

Returns:

Void.