

VFI

Generated by Doxygen 1.8.1.1

Tue Sep 4 2012 13:10:00

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	abs_diff< T > Struct Template Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Function Documentation	5
3.1.2.1	operator()	5
3.2	ar1Vals< T > Struct Template Reference	6
3.2.1	Detailed Description	6
3.2.2	Member Function Documentation	6
3.2.2.1	operator()	6
3.3	kGrid< T > Struct Template Reference	7
3.3.1	Detailed Description	7
3.3.2	Member Function Documentation	7
3.3.2.1	operator()	7
3.4	transMat< T > Struct Template Reference	8
3.4.1	Detailed Description	8
3.4.2	Member Function Documentation	8
3.4.2.1	operator()	8
3.5	vfInIt< T > Struct Template Reference	9
3.5.1	Detailed Description	9
3.5.2	Member Function Documentation	9
3.5.2.1	operator()	9
3.6	vfStep< T > Struct Template Reference	10
3.6.1	Detailed Description	10
3.6.2	Member Function Documentation	11
3.6.2.1	operator()	11

4	File Documentation	13
4.1	auxFuncs.h File Reference	13
4.1.1	Detailed Description	13
4.1.2	Function Documentation	14
4.1.2.1	printMatrix	14
4.1.2.2	printVector	14
4.2	functors.hpp File Reference	14
4.2.1	Detailed Description	15
4.2.2	Function Documentation	15
4.2.2.1	binary_max	15
4.2.2.2	binary_val	16
4.2.2.3	grid_max	16
4.3	global.cpp File Reference	17
4.3.1	Detailed Description	17
4.4	global.h File Reference	18
4.4.1	Detailed Description	19
4.4.2	Function Documentation	19
4.4.2.1	curr_second	19
4.5	main.cu File Reference	19
4.5.1	Detailed Description	19
4.5.2	Function Documentation	20
4.5.2.1	main	20
4.6	timer.cpp File Reference	20
4.6.1	Detailed Description	20
4.6.2	Function Documentation	20
4.6.2.1	curr_second	20

Chapter 1

Class Index

1.1 Class List

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

abs_diff< T >	Functor to compute the absolute difference between elements of two vectors	5
ar1Vals< T >	Functor to compute discrete AR1 approximation	6
kGrid< T >	Functor to compute grid values for capital	7
transMat< T >	Functor to compute transition matrix for discrete AR1 approximation	8
vfInit< T >	Functor to initialize the value function	9
vfStep< T >	Functor to update the value function	10

Chapter 2

File Index

2.1 File List

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

auxFuncs.h	Simple auxiliary functions	13
functors.hpp	File of Thrust functors and functions	14
global.cpp	Global variables for the value function iteration problem	17
global.h	Global header file	18
main.cu	File containing main main function for the VFI problem	19
timer.cpp	File containing timer function	20

Chapter 3

Class Documentation

3.1 `abs_diff< T >` Struct Template Reference

Functor to compute the absolute difference between elements of two vectors.

```
#include <functors.hpp>
```

Public Member Functions

- `__host__ __device__ T operator() (const T &x, const T &y) const`

3.1.1 Detailed Description

```
template<typename T> struct abs_diff< T >
```

Functor to compute the absolute difference between elements of two vectors.

3.1.2 Member Function Documentation

3.1.2.1 `template<typename T > __host__ __device__ T abs_diff< T >::operator() (const T & x, const T & y) const`
`[inline]`

Kernel to compute the absolute difference between elements.

Parameters

<code>x</code>	value of first vector element.
<code>y</code>	value of second vector element.

Returns

absolute difference between elements.

The documentation for this struct was generated from the following file:

- [functors.hpp](#)

3.2 ar1Vals< T > Struct Template Reference

Functor to compute discrete AR1 approximation.

```
#include <functors.hpp>
```

Public Member Functions

- [ar1Vals](#) (int _nz, T _lambda, T _mu, T _sigma, T _rho)
Constructor.
- `__host__ __device__ T operator\(\) (const int &ix) const`

Public Attributes

- const int [nz](#)
Number of values in AR1 grid.
- const T [lambda](#)
Number of standard deviations for AR1 approximation.
- const T [mu](#)
AR1 mean.
- const T [sigma](#)
AR1 standard deviation.
- const T [rho](#)
AR1 persistence.

3.2.1 Detailed Description

```
template<typename T>struct ar1Vals< T >
```

Functor to compute discrete AR1 approximation.

This functor provides the kernel to compute a discrete AR1 approximation using the method of Tauchen (1986).

3.2.2 Member Function Documentation

3.2.2.1 `template<typename T > __host__ __device__ T ar1Vals< T >::operator() (const int & ix) const` `[inline]`

Kernel to compute discrete AR1 approximation (using Tauchen's method).

Parameters

<code>ix</code>	index of the AR1 grid.
-----------------	------------------------

Returns

Value of the AR1 process at position ix in the grid.

The documentation for this struct was generated from the following file:

- [functors.hpp](#)

3.3 kGrid< T > Struct Template Reference

Functor to compute grid values for capital.

```
#include <functors.hpp>
```

Public Member Functions

- [kGrid](#) (int _nk, int _nz, T _beta, T _alpha, T _delta, T *_Z)
Constructor.
- `__host__ __device__ T operator\(\) (const int &ix) const`

Public Attributes

- const int [nk](#)
Number of values in capital grid.
- const int [nz](#)
Number of values in AR1 (TFP) grid.
- const T [beta](#)
Time discount factor.
- const T [alpha](#)
Capital share in production function.
- const T [delta](#)
Depreciation rate.
- const T * [Z](#)
Pointer to AR1 (TFP) grid.

3.3.1 Detailed Description

```
template<typename T>struct kGrid< T >
```

Functor to compute grid values for capital.

This functor provides the kernel to compute an equally spaced grid for capital.

3.3.2 Member Function Documentation

3.3.2.1 `template<typename T > __host__ __device__ T kGrid< T >::operator() (const int & ix) const` `[inline]`

Kernel to compute each K value in grid.

Parameters

<code>ix</code>	index of the AR1 grid.
-----------------	------------------------

Returns

Value of capital at position ix in the grid.

The documentation for this struct was generated from the following file:

- [functors.hpp](#)

3.4 transMat< T > Struct Template Reference

Functor to compute transition matrix for discrete AR1 approximation.

```
#include <functors.hpp>
```

Public Member Functions

- [transMat](#) (int _nz, T _mu, T _sigma, T _rho, T *_Z, T *_P)
Constructor.
- `__host__ __device__ void operator\(\) (const int &ix) const`

Public Attributes

- const int [nz](#)
Number of values in AR1 grid.
- const T [mu](#)
AR1 mean.
- const T [sigma](#)
AR1 standard deviation.
- const T [rho](#)
AR1 persistence.
- T * [Z](#)
Pointer to AR1 grid.
- T * [P](#)
Pointer to transition matrix.

3.4.1 Detailed Description

```
template<typename T>struct transMat< T >
```

Functor to compute transition matrix for discrete AR1 approximation.

This functor provides the kernel to compute the transition matrix for a a discrete AR1 approximation using the method of Tauchen (1986).

3.4.2 Member Function Documentation

3.4.2.1 `template<typename T > __host__ __device__ void transMat< T >::operator() (const int & ix) const` `[inline]`

Kernel to compute transition matrix for discrete AR1 approximation (using Tauchen's method).

Parameters

<code>ix</code>	index of the AR1 grid.
-----------------	------------------------

Returns

Void.

The documentation for this struct was generated from the following file:

- [functors.hpp](#)

3.5 vflnit< T > Struct Template Reference

Functor to initialize the value function.

```
#include <functors.hpp>
```

Public Member Functions

- [vflnit](#) (int _nk, T _eta, T _beta, T _alpha, T _delta, T *_Z, T *_V)
Constructor.
- `__host__ __device__ void operator\(\) (const int &jx) const`

Public Attributes

- const int [nk](#)
Number of values in capital grid.
- const T [eta](#)
Coefficient of relative risk aversion.
- const T [beta](#)
Time discount factor.
- const T [alpha](#)
Capital share in production function.
- const T [delta](#)
Depreciation rate.
- const T * [Z](#)
Pointer to AR1 (TFP) grid.
- T * [V](#)
Pointer to current iteration of the value function.

3.5.1 Detailed Description

```
template<typename T>struct vflnit< T >
```

Functor to initialize the value function.

This functor initializes the value function at the deterministic steady state, for each value of the AR1 (TFP) grid.

3.5.2 Member Function Documentation

3.5.2.1 `template<typename T > __host__ __device__ void vflnit< T >::operator() (const int &jx) const` `[inline]`

Kernel to initialize value function.

Parameters

<code>jx</code>	index of the AR1 (TFP) grid.
-----------------	------------------------------

Returns

Void.

The documentation for this struct was generated from the following file:

- [functors.hpp](#)

3.6 vfStep< T > Struct Template Reference

Functor to update the value function.

```
#include <functors.hpp>
```

Public Member Functions

- [vfStep](#) (int [_nk](#), int [_nz](#), T [_eta](#), T [_beta](#), T [_alpha](#), T [_delta](#), char [_maxtype](#), bool [_howard](#), T * [_K](#), T * [_Z](#), T * [_P](#), T * [_V0](#), T * [_V](#), T * [_G](#))
Constructor.
- `__host__ __device__ void operator\(\) (const int &hx) const`

Public Attributes

- const int [nk](#)
Number of values in capital grid.
- const int [nz](#)
Number of values in AR1 (TFP) grid.
- const T [eta](#)
Coefficient of relative risk aversion.
- const T [beta](#)
Time discount factor.
- const T [alpha](#)
Capital share in production function.
- const T [delta](#)
Depreciation rate.
- const char [maxtype](#)
Flag to indicate maximization method.
- const bool [howard](#)
Flag to indicate use of Howard improvement.
- const T * [K](#)
Pointer to capital grid.
- const T * [Z](#)
Pointer to AR1 (TFP) grid.
- const T * [P](#)
Pointer to transition matrix.
- const T * [V0](#)
Pointer to current iteration of the value function.
- T * [V](#)
Pointer to the updated value function.
- T * [G](#)
Pointer to current iteration of the capital policy function.

3.6.1 Detailed Description

```
template<typename T>struct vfStep< T >
```

Functor to update the value function.

This functor updates the value function. If [howard](#) is FALSE, the value and capital policy functions are updated by maximizing the Belman objective function using the current value function. Maximization is either performed by [binary_max](#) or [grid_max](#). If [howard](#) is TRUE, the value function is updated without maximization, by simply iterating the Belman with the current capital policy function.

3.6.2 Member Function Documentation

3.6.2.1 `template<typename T> __host__ __device__ void vfStep< T >::operator() (const int & hx) const` `[inline]`

Kernel to update the value function.

Parameters

<i>hx</i>	index of V0 (stored as a flat array).
-----------	---------------------------------------

Returns

Void.

The documentation for this struct was generated from the following file:

- [functors.hpp](#)

Chapter 4

File Documentation

4.1 auxFuncs.h File Reference

Simple auxiliary functions.

```
#include <iostream>
#include <iomanip>
```

Functions

- `template<class T >`
void `printMatrix` (const bool colMaj, const int M, const int N, T *X, const int printRows, const int printCols, const int digits)
Function to print the elements of a matrix.
- `template<class T >`
void `printVector` (const int N, T *X, const int digits)
Function to print the elements of a vector.

4.1.1 Detailed Description

Simple auxiliary functions.

Author

Eric M. Aldrich
ealdrich@ucsc.edu

Version

1.0

Date

18 July 2012

Copyright

Copyright Eric M. Aldrich 2012
Distributed under the Boost Software License, Version 1.0 (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)

4.1.2 Function Documentation

4.1.2.1 `template<class T > void printMatrix (const bool colMaj, const int M, const int N, T * X, const int printRows, const int printCols, const int digits)`

Function to print the elements of a matrix.

This functions prints a subset of the elements of a matrix to the screen.

Parameters

<i>colMaj</i>	boolean indicating if the matrix is stored in column-major format.
<i>M</i>	number of rows in the data matrix.
<i>N</i>	number of columns in the data matrix.
<i>X</i>	pointer to data matrix.
<i>printRows</i>	number of rows to print.
<i>printCols</i>	number of columns to print.
<i>precision</i>	number of significant digits to print.

Returns

Void.

4.1.2.2 `template<class T > void printVector (const int N, T * X, const int digits)`

Function to print the elements of a vector.

This functions prints a subset of the elements of a vector to the screen.

Parameters

<i>N</i>	number of elements in the data matrix.
<i>X</i>	pointer to data matrix.
<i>precision</i>	number of significant digits to print.

Returns

Void.

4.2 functors.hpp File Reference

File of Thrust functors and functions.

```
#include <thrust/iterator/zip_iterator.h>
#include <thrust/for_each.h>
#include <thrust/device_vector.h>
#include <cmath>
```

Classes

- struct [ar1Vals< T >](#)
Functor to compute discrete AR1 approximation.
- struct [transMat< T >](#)
Functor to compute transition matrix for discrete AR1 approximation.

- struct `kGrid< T >`
Functor to compute grid values for capital.
- struct `vflnit< T >`
Functor to initialize the value function.
- struct `vfStep< T >`
Functor to update the value function.
- struct `abs_diff< T >`
Functor to compute the absolute difference between elements of two vectors.

Functions

- `template<typename T >`
`__host__ __device__ int binary_val (const T x, const int n, const T *X)`
Function to find the location of a value in a monotonic grid.
- `template<typename T >`
`__host__ __device__ void grid_max (const int klo, const int nksub, const int nk, const int nz, const T ydepK, const T eta, const T beta, const T *K, const T *P, const T *V0, T *V, T *G)`
Function to maximize Belman objective function with naive grid search.
- `template<typename T >`
`__host__ __device__ void binary_max (const int klo, const int nksub, const int nk, const int nz, const T ydepK, const T eta, const T beta, const T *K, const T *P, const T *V0, T *V, T *G)`
Function to maximize Belman objective function with binary search.

4.2.1 Detailed Description

File of Thrust functors and functions.

Author

Eric M. Aldrich
ealdrich@ucsc.edu

Version

1.0

Date

12 July 2012

Copyright

Copyright Eric M. Aldrich 2012
Distributed under the Boost Software License, Version 1.0 (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)

4.2.2 Function Documentation

4.2.2.1 `template<typename T > __host__ __device__ void binary_max (const int klo, const int nksub, const int nk, const int nz, const T ydepK, const T eta, const T beta, const T * K, const T * P, const T * V0, T * V, T * G)`

Function to maximize Belman objective function with binary search.

This function computes the maximum of the Belman objective function for a given pair of state values by using a binary search algorithm, as outlined in Heer and Maussner (2005, p.26).

Parameters

<i>klo</i>	lower index of the capital grid to begin search.
<i>nksub</i>	number of points in the capital grid to include in search.
<i>nk</i>	number of points in the capital grid.
<i>nz</i>	number of points in the AR1 (TFP) grid.
<i>ydepK</i>	value of output plus depreciated capital.
<i>eta</i>	coefficient of relative risk aversion.
<i>beta</i>	time discount factor.
<i>*K</i>	pointer to capital grid.
<i>*P</i>	pointer to AR1 (TFP) transition matrix.
<i>*V0</i>	pointer to current iterate of value function.
<i>*V</i>	pointer to updated value function.
<i>*G</i>	pointer to updated capital policy function.

Returns

Void.

4.2.2.2 `template<typename T> __host__ __device__ int binary_val (const T x, const int n, const T * X)`

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

This function finds the first value $X[ix]$ such that $X[ix] \geq x$, where x is a scalar value, X is a monotonic grid, and ix is the index of X .

Parameters

<i>x</i>	value to search for in grid X .
<i>n</i>	number of values in grid X .
<i>*X</i>	pointer to grid X .

Returns

$imax$ first integer ($\leq n$) such that $X[ix] \geq x$.

4.2.2.3 `template<typename T> __host__ __device__ void grid_max (const int klo, const int nksub, const int nk, const int nz, const T ydepK, const T eta, const T beta, const T * K, const T * P, const T * V0, T * V, T * G)`

Function to maximize Belman objective function with naive grid search.

This function computes the maximum of the Belman objective function for a given pair of state values by searching over each possible value of future capital in the grid for capital.

Parameters

<i>klo</i>	lower index of the capital grid to begin search.
<i>nksub</i>	number of points in the capital grid to include in search.
<i>nk</i>	number of points in the capital grid.
<i>nz</i>	number of points in the AR1 (TFP) grid.
<i>ydepK</i>	value of output plus depreciated capital.
<i>eta</i>	coefficient of relative risk aversion.
<i>beta</i>	time discount factor.
<i>*K</i>	pointer to capital grid.
<i>*P</i>	pointer to AR1 (TFP) transition matrix.
<i>*V0</i>	pointer to current iterate of value function.
<i>*V</i>	pointer to updated value function.
<i>*G</i>	pointer to updated capital policy function.

Returns

Void.

4.3 global.cpp File Reference

Global variables for the value function iteration problem.

```
#include "global.h"
```

Variables

- const REAL [eta](#) = 2
Coefficient of relative risk aversion.
- const REAL [beta](#) = 0.984
Time discount factor.
- const REAL [alpha](#) = 0.35
Share of capital in the production function.
- const REAL [delta](#) = 0.01
Rate of capital depreciation.
- const REAL [mu](#) = 0.0
TFP mean.
- const REAL [rho](#) = 0.95
TFP persistence.
- const REAL [sigma](#) = 0.005
TFP volatility.
- const int [nk](#) = 1024
Number of values in capital grid.
- const int [nz](#) = 4
Number of values TFP grid.
- const REAL [tol](#) = 1e-8*(1-beta)
Tolerance for convergence.
- const char [maxtype](#) = 'b'
Maximization method - choices are 'g' (grid) and 'b' (binary search).
- const int [howard](#) = 1
Number of howard steps to perform between.

4.3.1 Detailed Description

Global variables for the value function iteration problem.

Author

Eric M. Aldrich
ealdrich@ucsc.edu

Version

1.0

Date

12 July 2012

Copyright

Copyright Eric M. Aldrich 2012

Distributed under the Boost Software License, Version 1.0 (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)

4.4 global.h File Reference

Global header file.

Typedefs

- typedef double **REAL**

Functions

- REAL [curr_second](#) (void)
Basic timer method.

Variables

- const REAL [eta](#)
Coefficient of relative risk aversion.
- const REAL [beta](#)
Time discount factor.
- const REAL [alpha](#)
Share of capital in the production function.
- const REAL [delta](#)
Rate of capital depreciation.
- const REAL [mu](#)
TFP mean.
- const REAL [rho](#)
TFP persistence.
- const REAL [sigma](#)
TFP volatility.
- const int [nk](#)
Number of values in capital grid.
- const int [nz](#)
Number of values TFP grid.
- const REAL [tol](#)
Tolerance for convergence.
- const char [maxtype](#)
Maximization method - choices are 'g' (grid) and 'b' (binary search).
- const int [howard](#)
Number of howard steps to perform between.

4.4.1 Detailed Description

Global header file.

4.4.2 Function Documentation

4.4.2.1 REAL curr_second (void)

Basic timer method.

Author

Kyle Spafford

Version

1.0

Date

19 November 2010

Copyright

Copyright Kyle Spafford 2012

Distributed under the Boost Software License, Version 1.0 (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)

4.5 main.cu File Reference

File containing main main function for the VFI problem.

```
#include "global.h"
#include "auxFuncs.h"
#include <iostream>
#include <ctime>
#include "functors.hpp"
#include <thrust/host_vector.h>
#include <thrust/transform.h>
#include <thrust/for_each.h>
#include <thrust/sequence.h>
#include <thrust/sort.h>
```

Functions

- int `main` ()

Main function for the VFI problem.

4.5.1 Detailed Description

File containing main main function for the VFI problem.

4.5.2 Function Documentation

4.5.2.1 main ()

Main function for the VFI problem.

This function solves a standard neoclassical growth model with value function iteration, using Thrust. Parallelization occurs at the grid of values for the state space, with each thread finding the maximum of the Bellman objective function for a pair of state values.

See [Aldrich et al. \(2011\)](#).

```
@see functors.hpp

@returns 0 upon successful completion, 1 otherwise.

@author Eric M. Aldrich \n
        ealdrich@ucsc.edu

@version 1.0

@date 12 July 2012

@copyright Copyright Eric M. Aldrich 2012 \n
        Distributed under the Boost Software License, Version 1.0
        (See accompanying file LICENSE_1_0.txt or copy at \n
        http://www.boost.org/LICENSE_1_0.txt)
```

4.6 timer.cpp File Reference

File containing timer function.

```
#include <stddef.h>
#include <sys/time.h>
#include "global.h"
```

Functions

- REAL [curr_second](#) (void)
Basic timer method.

4.6.1 Detailed Description

File containing timer function.

4.6.2 Function Documentation

4.6.2.1 REAL curr_second (void)

Basic timer method.

Author

Kyle Spafford

Version

1.0

Date

19 November 2010

Copyright

Copyright Kyle Spafford 2012

Distributed under the Boost Software License, Version 1.0 (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)

Bibliography

Aldrich, E. M., Fernández-Villaverde, J., Gallant, A. R., and Rubio-Ramírez, J. F. (2011), "Tapping the supercomputer under your desk: Solving dynamic equilibrium models with graphics processors," *Journal of Economic Dynamics and Control*, 35, 386–393.