VFI-Thrust

Generated by Doxygen 1.8.1.1

Thu Jul 12 2012 13:59:47

Contents

Chapter 1

Class Documentation

1.1 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

1.1.1 Detailed Description

template<typename T>struct abs_diff< T >

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

1.1.2 Member Function Documentation

1.1.2.1 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

x	value of first vector element.
y	value of second vector element.

Returns

absolute difference between elements.

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

- functors.hpp

1.2 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.

1.2.1 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).

1.2.2 Member Function Documentation

1.2.2.1 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

ix	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

1.3 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.

1.3.1 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.

1.3.2 Member Function Documentation

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

Kernel to compute each K value in grid.

Parameters

ix	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

1.4 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.

1.4.1 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).

1.4.2 Member Function Documentation

1.4.2.1 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

ix	index of the AR1 grid.

Returns

Void.

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

- functors.hpp

1.5 vfInit< T > Struct Template Reference

Functor to initialize the value function.

`#include <functors.hpp>`

Public Member Functions

- vfInit (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.

1.5.1 Detailed Description

template<typename T>struct vfInit< T >

Functor to initialize the value function.

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

1.5.2 Member Function Documentation

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

Kernel to initialize value function.

Parameters

jx	index of the AR1 (TFP) grid.

Returns

Void.

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

- functors.hpp

1.6 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.

1.6.1 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.

1.6.2 Member Function Documentation

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

Kernel to update the value function.

Parameters

hx	index of V0 (stored as a flat array).

Returns

Void.

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

- functors.hpp

Chapter 2

File Documentation

2.1 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 vfInit< 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.

2.1.1 Detailed Description

File of Thrust functors and functions.

Author

Eric M. Aldrich
ealdrich@ucsc.edu

Version

1.0

Date

12 July 2012

Copyright

2.1.2 Function Documentation

2.1.2.1 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

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

Returns

Void.

2.1.2.2 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] >= x, where x is a scalar value, X is a monotonic grid, and ix is the index of X.

Parameters

x	value to search for in grid X.
n	number of values in grid X.
∗X	pointer to grid X.

Returns

imax first integer (<= n) such that X[ix] >= x.

2.1.2.3 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

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

Returns

Void.

2.2 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 = 256

  Number of values in capital grid.
- const int nz = 4

  Number of values TFP grid.
- const REAL tol = 0.00000001∗(1-beta)

  Tolerance for convergence.
- const char maxtype = ’g’

  Maximization method - choices are ’g’ (grid) and ’b’ (binary search).
- const int howard = 20

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

2.2.1 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

2.3 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 maximizations - set howard = 1 if max = 'b'.

2.3.1 Detailed Description

Global header file.

2.3.2 Function Documentation

2.3.2.1 REAL curr_second (void)

Basic timer method.

Author

Kyle Spafford

Version

1.0

Date

19 November 2010

Copyright

2.4 main.cu File Reference

File containing main main function for the VFI problem.

```
#include "global.h"
#include <iostream>
#include <ctime>
#include "functors.hpp"
#include <thrust/device_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.

2.4.1 Detailed Description

File containing main main function for the VFI problem.

2.4.2 Function Documentation

2.4.2.1 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, 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.

See also

functors.hpp

Returns

0 upon successful completion, 1 otherwise.

Author

Eric M. Aldrich
ealdrich@ucsc.edu

Version

1.0

Date

12 July 2012

Copyright

2.5 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.

2.5.1 Detailed Description

File containing timer function.

2.5.2 Function Documentation

2.5.2.1 REAL curr_second (void)

Basic timer method.

Author

Kyle Spafford

Version

1.0

Date

19 November 2010

Index