Reference Manual

Generated by Doxygen 1.6.1

Thu Oct 25 15:07:48 2012

Contents

Chapter 1

Example Code for Basic Value Function Iteration Problem

1.1 Introduction

The software in this archive solves a basic neoclassical growth model using value function iteration, as outlined in 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. Multiple parallel implementations are included for the purpose of comparison, including massively parallel GPU software (CUDA C and Thrust), multi-core CPU C++ software (Thrust OpenMP), as well as single-core C++ and Matlab software.

1.2 Use

1.2.1 File

The file 'parameters.txt' contains the software inputs. The file must contain 13 lines, each line beginning with a parameter value, followed by a comma, follow by a line of text describing the parameter. The order of the parameters can be found in the 'parameters' class description in CPP/global.h.

1.2.2 Implemntation

Except for the Matlab code, individual software implementations can be run from individual directories by typing 'make; ./main' at the command line. The Matlab code can be run via 'main.m' either interactively or in batch.

The 'Thrust' directory contains two makefiles, one corresponding to a GPU implementation (makefile_-device) and one corresponding to an OpenMP CPU implementation (makefile_host). When using the OpenMP implementation it is important to set the environment variable 'OMP_NUM_THREADS=N', where 'N' is the number of CPU cores available on the system.

1.2.3 Output

When each softare implementation is run, it loads the parameter values in 'parameters.txt' and returns the value function, policy function and total solution time in files 'valFuncMethod.dat', 'polFuncMethod.dat' and 'SolTimeMethod.dat', respectively, where 'Method' is a string that corresponds to the implementation, and which is equivalent to one of the command line arguments described in the next section.

1.2.4 Comparison

To run multiple software implementations in sequence and compare their results, simply run the shell script 'compareMethods.sh'. The script takes potentially multiple arguments, which must correspond to one of software directory names, or alternatively 'ThrustGPU' or 'ThrustOMP'. The first argument serves as the baseline implementation against which other methods are compared. The script utilizes either 'solutionDiff.m' or 'solutionDiff.R' to compare the output reported in the data files described above - the user must comment the appropriate line to choose among the Matlab or R scripts.

1.3 Dependencies

The Thrust and CUDA-C implementations have been successfully built and run under CUDA Toolkit 4.2 on a CentOS 6 Linux operating system. The C++ implementation utilitizes the GNU g++ compiler and the Eigen template library for linear algebra (eigen.tuxfamily.org). The makefiles in the respective directories point to headers and libraries for CUDA and Eigen - these directory paths may need to be changed when running on different systems.

Author:

Eric M. Aldrich
ealdrich@ucsc.edu

Version:

1.0

Date:

23 Oct 2012