**Entry Point:**

Compile\_mexSolveModelGivenParms.m

* For compilation we will need the following files:
  + mexSolveGivenParms.cu
  + liquidity\_mex\_defs.cu
  + normaldist\_mex.cu
  + tauchen\_mex.cu
  + liquidity\_vfi\_mex.cu

mexSolveGivenParms.cu

* This is a mexfile and serves as the main link between the C++/CUDA and MATLAB.
* How this function is called in MATLAB to solve the model given the parameters:
  + For us to run this function, we need to have arguments, the arguments are given in:
    - Fit\_params\_10302020\_turnover\_1998\_2004\_run2.m
      * This file does not run mexSolveGivenParms.cu directly.
      * It calls FitParmsExactID\_Parallel\_5moments.m
        + This is the function that calls mexSolveModelGivenParms
        + Then computes moments using SimSinglePath\_mex\_output.m

Does this function use other functions?

* How is this function constructed in terms of the C++ code?
  + Once parameters are read into the model, the mexfunction calls the function SolveModel
  + This function is defined in liquidity\_vfi\_mex.cu along with other functions.
* What other things does this file do?
  + Read parameters and translate them to C.
  + Store the parameters in a class defined in the host.
  + Calls SolveModel and stores all the results in host memory.
  + Creates an output structure in MATLAB and exports the results to MALTAB.

liquidity\_vfi\_mex.cu

1. This file creates several functions:
   * Ggq\_topdown
   * Vfi\_iterate\_policy
   * Vfi\_iterate
   * Vfi\_interpolate
   * Vfi\_update1
   * Vfi\_update2
   * Update\_compute\_errors
   * Vfi
   * SolveModel
     + This is the first function that is called in from the Mex file.
     + This function oversees allocation of device memory.
     + Initializing device memory.
     + Initializing value functions.
     + Running vfi
     + Running vfi\_iterate\_policy
     + Copying memory to host
     + Freeing memory.
   * initValueFuns

liquidity\_mex\_def.cu

* This File oversees reading the parameters and storing them in host memory.
* It creates grids and performs memory allocation and elimination of dynamic memory.
* The file also defines arrays that will be useful in the future and are constant across iterations.
* In the header file there are also class definitions for storing memory in the host.

**Detailed information about some functions:**

SolveModel: Seems to use a constant grid of threads for all kernel launches.

* initValueFuns: This kernel sets all vectors to zero.
* Vfi: This is a normal function that runs multiple kernels.
  + Vfi\_iterate:
    - Ggq\_topdown algorithm:
  + Vfi\_update1: Use the result from ggq\_topdown and integrate over output to store the results.
  + Vfi\_interpolate: Working with R(b) and R(b)/b might demand interpolation.
  + Vfi\_update2: Updates continuation value of default and default prices.
* Vfi\_iterate\_policy: After all the updates are performed this function finds the optimal policy at E[m]

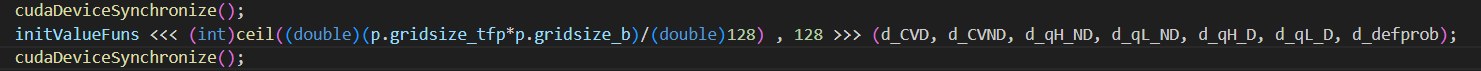
**Questions for Yu:**

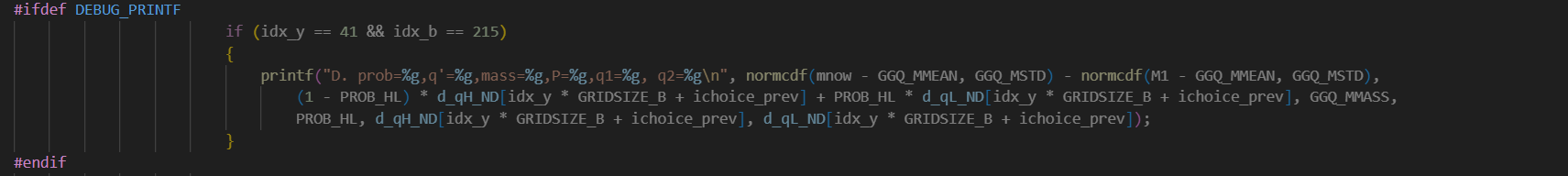
1. What is the -dynamic term doing during compilation of the mex file?
2. Why are you using the term?

A screenshot of a computer program

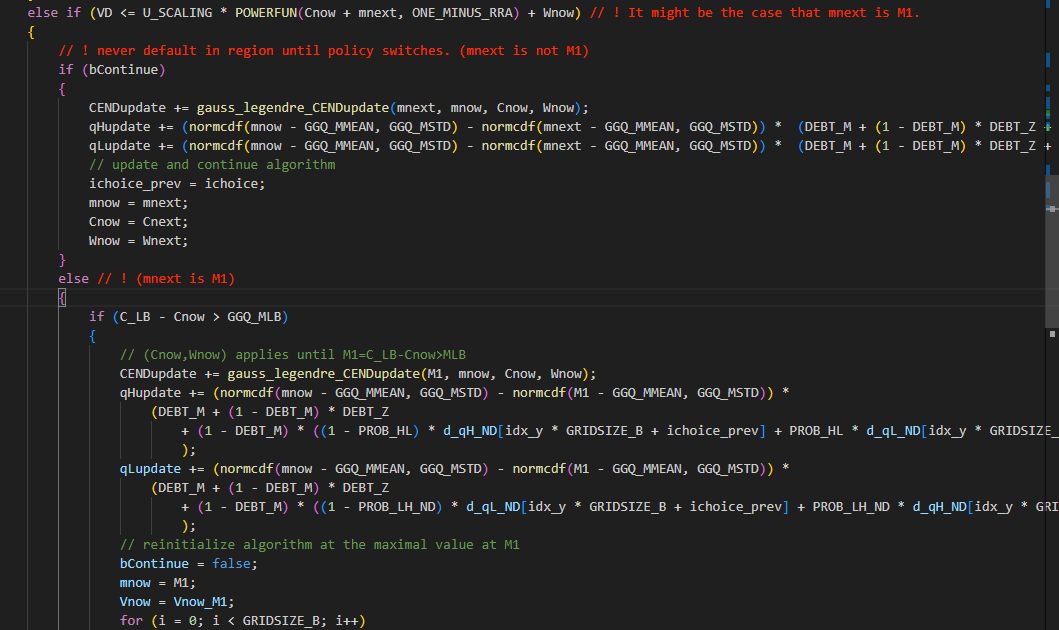
Description automatically generated

1. Why are you synchronizing the threads? Are they supposed to be synchronized after the kernel is executed?
2. Why are you fixing the number of threads per block to 128?



1. Why are you using volatile integers?
2. #ifdef debug statements: 
3. Why we allow for issuances that exceed probability of default?



1. How can this be a relevant region? Near to the lower bound of consumption the value function becomes –infinity. How can this be preferred to the value of default?
2. What is the reason for Vfi\_iterate\_policy?

**Questions to Answer:**

1. What are the convergence criteria?