* Answer to the questions mentioned in the previous email
  1. Prob3: *the solver was generating incorrect results on Std Prob 3. To elaborate, what happens on that test problem is that the magnetization starts in a random configuration. Then the spins smooth out to give a plausible magnetization configuration, so it appears that the exchange term is working properly. But over time a vortex configuration emerges with a head-to-head type wall in the middle of the cube. It looks like something is wrong with the demag. The simulation never completely converges (Max dm/dt decreases but doesn't go below 20 deg/ns or so), and eventually stops at 3 ns of sim time.*
     1. In fact the results that were outputted by GPU were correct. Here please note that we are using eulerevolver to test. Under this evolving method, the value of alpha will affect the equilibrium state. The default alpha = 0.5 will generate domain wall while alpha = 0.6 will not. This has been proved with my test of the original CPU OOMMF code.
     2. The value of dm/dt is indeed inconsistent with CPU results. And it turns out to be related to the precision we are using. When I configure the GPU computation to use double precision by defining oommf-1.2a5\pkg\oc\<platform>\CHOOSEDOUBLE the ocport.h. The dm/dt value will be almost the same as original CPU OOMMF code.
  2. Prob4: *Today I modified GPU\_eulerevolve.cu as per the instructions below and tried to run GPU\_prob4b.mif. There is something very wrong in this test. If I send the magnetization to mmDisp immediately after loading the problem I get the "S" magnetization pattern as expected. If I then take two steps and resend the magnetization I find that the magnetization is zero at every point. If I try to send GPU\_Demag::Field to mmDisp I get this error…*
     1. To make the code running with the fastest speed on GPU, the last version of GPU code was not transferring data (spin, field and energy) to CPU. I have been working on this and I have fixed almost all the outputs. With the newer version of GPU code, we have removed those errors. However, there are still some known bugs unfixed due to the limited time. I have summarized these bugs in the “Known Bugs” session.
* Working progress
  1. To make the outputs work, now we are transferring data from GPU to CPU. However this has made the code much slower than before. For example, the MuMag problem4b test is running almost 3x slower than the last version of code. This can be fixed by avoiding memory transfer when it is not requested by user. We are still working on it.
  2. Up to now we have made the GPU code working with only single threads configured with OOMMF settings. This can be extended to work with multi-threads.
* Known Bugs
  1. In mmDisp, the output of the initial iteration data (field, mxH, energy) are zeros. All the outputs following will be correct.
  2. In mmDataTable & mmGraph, the GPU\_eulerevolve::max\_spin\_agnle / max\_spin\_angle\_stage / max\_spin\_angle\_run are always zeros.
* Code changes

To let the code work with GPU, except for adding .cu files in the /app/oxs/loc/ directory, we also have to make modifications to some files in the /app/oxs/base/ directory. here are some major changes:

1. timeevolver.(cc/h) added Oxs\_TimeEvolver::GPU\_GetEnergyDensity();
2. chunckEnergy.(cc/h) added Oxs\_ComputeEnergiesChunkThread::GPU\_Cmd(); added void Oxs\_GPU\_ComputeEnergies(); added new constructor to accept GPU flag Oxs\_ChunkEnergy();
3. chunkEnergy.h added GPU\_ComputeEnergyAlt();
4. oxsthread.h added void Launch\_GPU();void LaunchRoot\_GPU();void LaunchTree\_GPU()
5. energy.(cc/h) modified Oxs\_Energy::UpdateStandardOutputs() to account for GPU outputs; added void Oxs\_Energy::GPU\_ComputeEnergy(); added new constructor to accept GPU flag: Oxs\_Energy::Oxs\_Energy();
6. A lot of debug output lines have been added to all these files, which will be removed after all the GPU development.