CATEGORY: COMPUTATIONAL PHYSICS - CP13

POSTER P5226

CONTACT NAME David Claudio-Gonzalez: dclaudio@ugto.mx

GPU TECHNOLOGY CONFERENCE

Optimization of an Explicit Finite Differences Solver for Enabling Faster Studies of Spintronic Effects

David Claudio-Gonzalez*, Thomas Sanchez-Lengeling*, José F. Ramos-Ortega*, André Thiaville+, and Jacques Miltat+

*Engineering Division Campus Irapuato-Salamanca, University of Guanajuato, 38940 Yuriria, Gto. Mexico

+Laboratoire de Physique des Solides, CNRS UMR 8502, Universite Paris-Sud XI, 91405 Orsay, France

1. Abstract

+ The acceleration of spintronic simulations in double precision based on the implementation of an explicit finite differences solver by factors of 1.6 to 13x is reported.

+ The smaller factor was observed when comparing a single thread implementation running in a Intel Xeon E5620 @ 2.4 GHz and a Nvidia GeForce GTX 670M. The highest value was observed when comparing an Intel i7-2760QM @ 2.4 GHz. and a Nvidia Tesla M2070.

+ Optimizations consisted of the reduction of access to the global device memory by the increased usage of registers and shared memory.

2. The Zhang and Li model

+ Interaction between spins of itinerant and localized electrons in an “sd” Hamiltonian:

$$H_{sd} = -J_{ex}\mathbf{s}\cdot\mathbf{S}$$

+ Localized electrons approximated as a classical magnetization vector

$$\mathbf{m}(\mathbf{r},t) = \mathbf{m}_0(\mathbf{r},t) + \delta\mathbf{m}_0(\mathbf{r},t) = n_0\frac{\mathbf{M}(\mathbf{r},t)}{M_s} + \delta\mathbf{m}(\mathbf{r},t)$$

+ Induced spin density consisting of adiabatic plus deviation terms:

$$J(\mathbf{r},t) = J_0(\mathbf{r},t) + \delta J(\mathbf{r},t)$$

+ Non adiabatic spin current density, from spin parallel to local magnetisation plus out of equilibrium spin density

$$= -(\mu_B P/e)\mathbf{j}_e \otimes \frac{\mathbf{M}(\mathbf{r},t)}{M_s} + \delta\mathbf{J}(\mathbf{r},t)$$

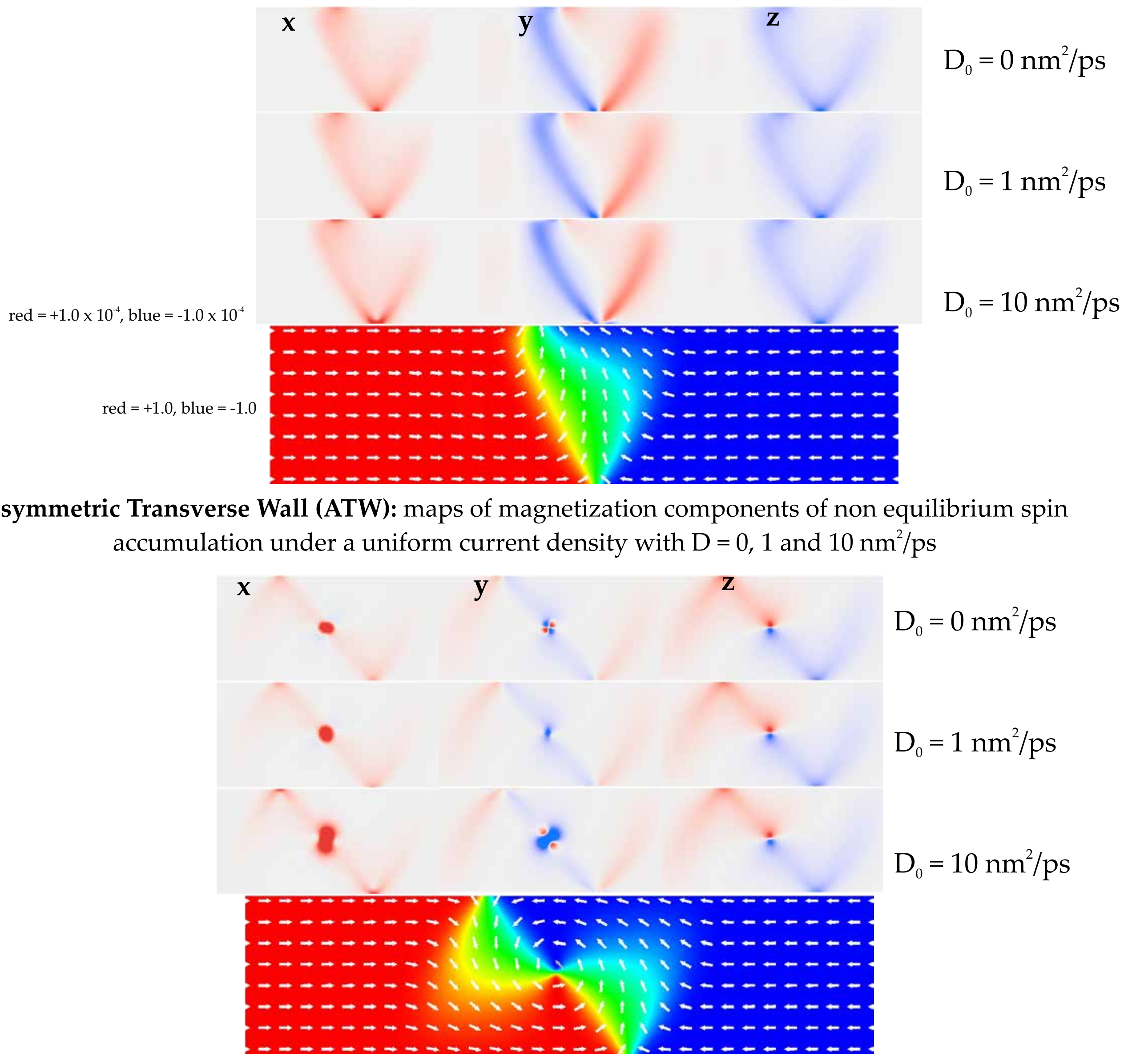
Closed form for non equilibrium spin density

$$D_0\nabla^2\delta\mathbf{m} - \frac{1}{\tau_{sd}}\delta\mathbf{m}\times\mathbf{M} - \frac{1}{\tau_{sf}}\delta\mathbf{m} = -\frac{\mu_B P}{e}(\mathbf{j_e}\cdot\nabla)\mathbf{M}$$

+ Physically realistic but computationally unfriendly (i.e. looooooooooooooooooooooooong computation times -> unpractical)

3. Numerical solution of spin accumulation

$$\frac{d\delta\vec{m}}{dt} = D_0\nabla^2\delta\vec{m} - \frac{1}{\tau_{sd}}\delta\vec{m}\times\vec{M} - \frac{1}{\tau_{sf}}\delta m + (\vec{u}\cdot\vec{\nabla})\vec{M}$$



Asymmetric Transverse Wall (ATW): maps of magnetization components of non equilibrium spin accumulation under a uniform current density with D = 0, 1 and 10 nm^2/ps

Vortex Wall (VW): Same as for ATW, we point out the noticeable effect of the diffusion constant around the vortex core, which is the smallest feature of the wall.

Figure 2

4. Advantages of using GPU computing

+ Current research in the field of spintronics relies heavily upon the use of numerical simulations.

+ Some simulations in the field of spintronics were unfeasible due to the long running times required sometimes months.

+In Figure 1 we present the results of running our finite differences solver in various CUDA-capable devices.

+ Our simulation consisted of the integration of the equation known as the Zhang and Li model for **1 ns** in a grid with **57,600 cells**[1,2], an example of the same simulation but using **360,000 cells** for a more precise numerical integration is shown in Figure 2.

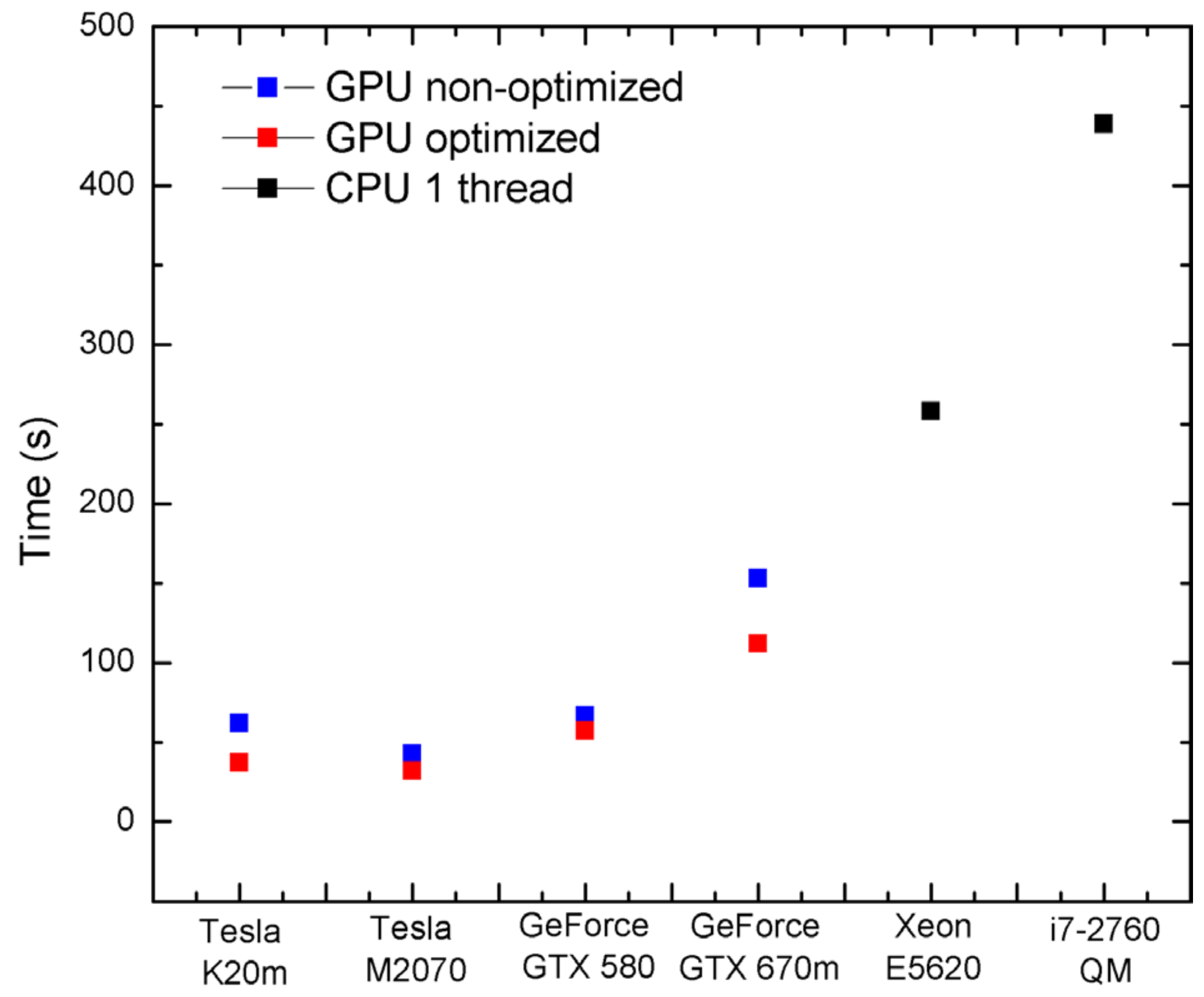


Figure 1

+ An equally realistic but larger simulation might require as much **16,000,000 cells**.

+ So far the largest simulations performed with our code are on the order of **1,440,000 cells** with integration times of **10 ns** and execution times of up to **5 hours**

+ A simulation like the one in Figure 2 usually becomes a single data point in a typical numerical study.

+ A thorough study migh require as many as **1000 datapoints** or approximately **208 days** assuming that all simulations are executed one after another.

+ Interestingly, the best performance was NOT obtained in the nominally superior hardware of the **Tesla K20m** (2496 cores @ 706 Mhz, Memory clock)[3] but on the **Tesla M2070** (448 @ 1.15 GHz)[4].

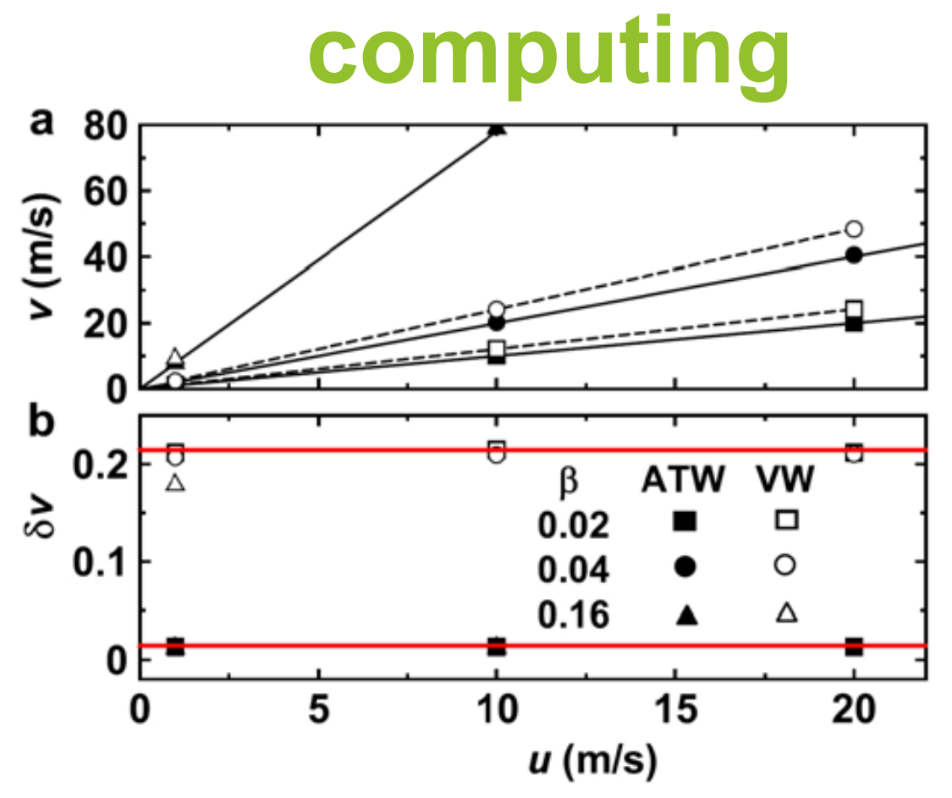
5. Code example before optimization

```
__global__ void gsource(double u, double *sm, double *m, int grid_width)
//Computation of source term using global memory
    {
    int i,j,index;
    double DELTAX;
    DELTAX = TX/NX;
//The last increment of two is due to the shifting of
//two array elements in the x direction in all arrays
    i = blockIdx.x * blockDim.x + threadIdx.x + 2;
    j = blockIdx.y * blockDim.y + threadIdx.y;
// map the two 2D indices to a single linear, 1D index
    index = j * grid_width + i;
    if (i > 1 && i < NX+2 && j >= 0 && j < NY)
        {
        sm[index] = u * (m[index - 2] - 8.0 * m[index - 1] + 8.0
            * m[index + 1] - m[index + 2]) / (12.0 * DELTAX);
        }
    }
```

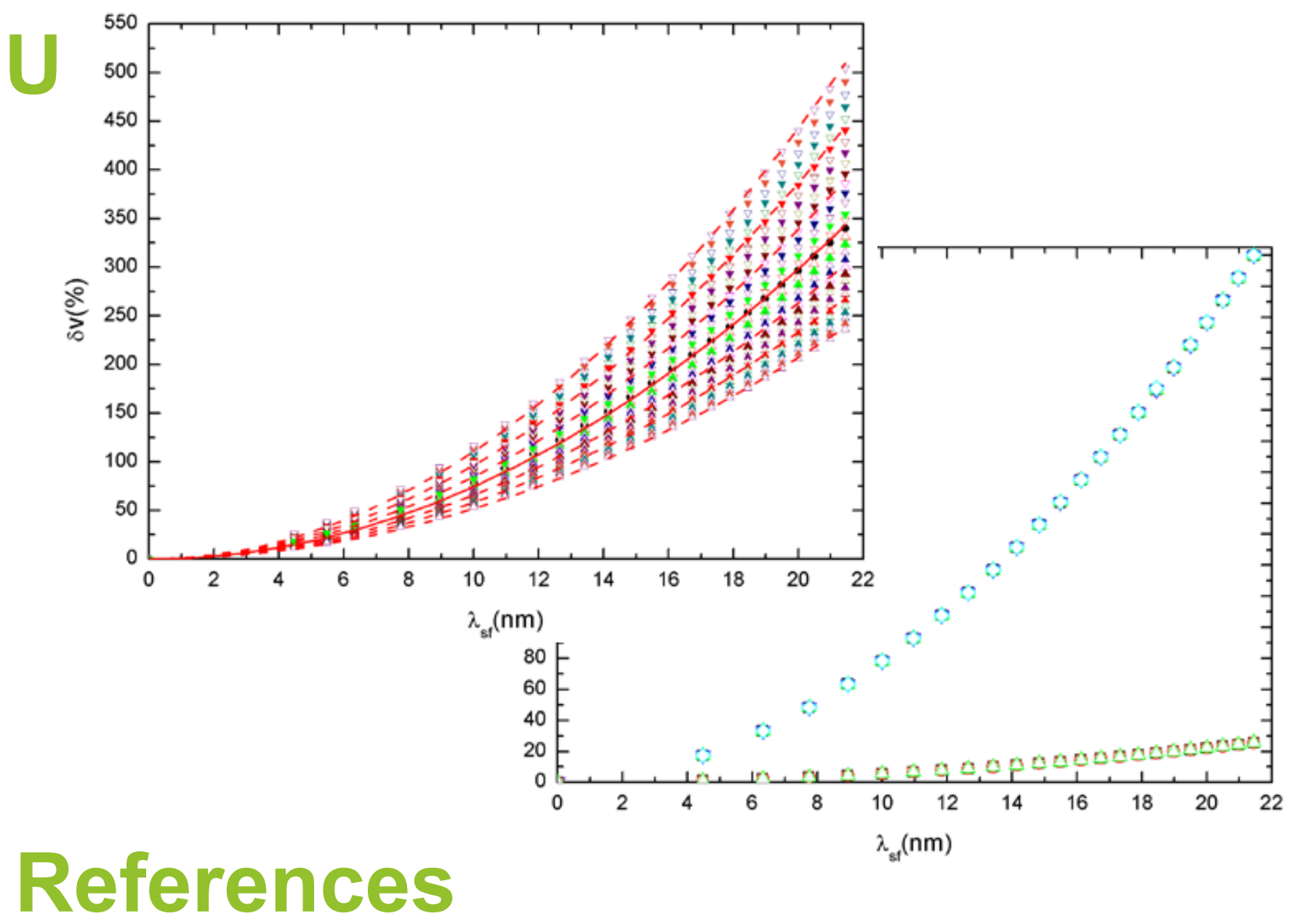
6. Code examples of optimization

```
__global__ //Constant Variables Increase performance with precalculation of values
    const int NXPLUS2 = NX + 2;
    const int NYMINUS2 = NY - 2;
    const int NYMINUS1 = NY - 1
...
__global__ void gsource(double u, double *sm, double *m, int grid_width)
//Computation of source term using global memory
    {
    int i, j, index;
//The last increment of two is due to the shifting of
//two array elements in the x direction in all arrays
    i = blockIdx.x * blockDim.x + threadIdx.x + 2;
    j = blockIdx.y * blockDim.y + threadIdx.y;
// map the two 2D indices to a single linear, 1D index
    index = j * grid_width + i;
    if (i > 1 && i < NXPLUS2 && j >= 0 && j < NY){
        sm[index] = u * (m[index - 2] - 8.0 * m[index - 1] + 8.0 * m[index + 1]
            - m[index + 2]) * DELTAX_TIMES_12_INV;
    }
}

void fillMatrix44Laplacian()
    {
    HANDLE_ERROR(cudaMalloc((void **)&Matrix44LaplacianB_01, sizeof(double)* 7));
...
    double * Matrix44LaplacianB_01_c = new double[16];
...
    double A11 = 2.0*DELTAY_CONS;
...
//Determinants of 2nd order
    DET2A = A33*A44 - A34*A43;
...
    Matrix44LaplacianB_01_c[0] = A22*A33 - A23*A32;  //2A
...
    Matrix44LaplacianB_01_c[6] = A11*DET2A - A21*DET2B + A31*DET2C; //YDENOM
...
    cudaMemcpy(Matrix44LaplacianB_01, Matrix44LaplacianB_01_c, 7 * sizeof(double),
        cudaMemcpyHostToDevice);
...
   }

//Computation of laplacian term using global memory
__global__ void glaplacianyboundaries(double *lapl_x, double *lapl_y, double *lapl_z,
    double *d2ady2, double *d2bdy2, double *d2gdy2,double *deltam_x, double *deltam_y,
    double *deltam_z,int grid_width, double DELTAY, double * Matrix44LaplacianB_01,
    double * Matrix44LaplacianB_02, double * Matrix44LaplacianB_03,
    double * Matrix44LaplacianB_04)
    {
...
// j = 0 mesh point after outmost down
    if (i > 1 && i < NXPLUS2 && j == 0)
        {
// d2deltam_x/dy2, (Lower Boundary)
        BFCT1 = deltam_x[frontneigh2] - deltam_x[index];
        BFCT2 = deltam_x[frontneigh1] - deltam_x[index];
...
    double YDENOM = Matrix44LaplacianB_01[6];
    double YNUM2 = -BFCT1 * Matrix44LaplacianB_01[3] + BFCT2 * Matrix44LaplacianB_01[4]
    - BFCT3 * Matrix44LaplacianB_01[5];
    D2FDL2 = YNUM2 / YDENOM;

    d2ady2[index] = D2FDL2;
...
    }
```

Science without GPU computing



Science with GPU computing



References

[1] S. Zhang and Z. Li Phy. Rev. Lett. 93, 127204 (2004)

[2] D. Claudio-Gonzalez, A. Thiaville and J. Miltat Phy. Rev. Lett. 108, 227208 (2012)

[3] http://www.nvidia.com/content/PDF/kepler/Tesla-K20-Passive-BD-06455-001-v07.pdf

[4] http://www.nvidia.com/docs/IO/43395/BD-05837-001_v01.pdf