Accelerated flow visualization of advective–diffusive mixing processes using GPUs

Visualización acelerada de flujos de mezclado advectivo-difusivo usando GPUs

Alejandro Aguilar Sierra*, Luis M. de la Cruz Salas**

ABSTRACT

In this article a strategy to accelerate the simulation and visualization of combined advective–diffusive mixing of a contaminant inside a square cavity with time-dependent boundary–conditions is presented. No moving walls are required to mix the fluid, but natural convection by periodic temperatures on opposite walls. A contaminant will diffuse asymptotically to uniform concentration. Advective mixing is evaluated with Lagrangian tracking and diffusive mixing is calculated with the Diffusive Strip method. That calculation requires hours of CPU time due to the large amount of numerical operations and the precision requested but it can be directly translated to a Graphics Processors Unit (GPU), improving the performance by orders of magnitude. The algorithms implemented in Compute Unified Device Architecture (CUDA) and an analysis of the speed achieved are presented. The visualization of the diffusion process is done simultaneously using the data stored in the GPU memory, which allows to make a real-time analysis of the mixing.

RESUMEN

Se presenta una estrategia para acelerar la simulación y visualización de mezclado advectivo–difusivo de un contaminante dentro de una cavidad cuadrada con condiciones de frontera dependientes del tiempo. No se requieren paredes móviles para mezclar el fluido, sino la convección natural por temperaturas periódicas en paredes opuestas. Un contaminante se diluye asintóticamente hasta una concentración uniforme. El mezclado advectivo es evaluado con rastreo lagrangiano y el mezclado difusivo con el método *Diffusive Strip*. Este cálculo requiere horas de tiempo CPU debido a la gran cantidad de operaciones numéricas y la precisión requerida, pero puede ser directamente transportado a una Unidad de Procesamiento Gráfico (GPU, por su acrónimo en inglés), mejorando así el rendimiento en órdenes de magnitud. Se presentan los algoritmos implementados en la Arquitectura de Cálculo Paralelo (CUDA, en inglés) y un análisis de la velocidad lograda. La visualización del proceso de difusión es simultánea usando los datos almacenados en la memoria de la GPU, lo que permite hacer un análisis en tiempo real del mezclado.

INTRODUCTION

Mixing has been studied during centuries and, in particular in the past few decades, there has been a recent surge of studies of fundamental properties of this study area due to its applications in manufacturing, food, pharmacology and other industries. The literature reporting progress in the field is large and the interested reader is referred to monographs and review articles for a comprehensive survey, for instance [2] and [3]. Many of the recent advances are the result of computerized measurement and simulation techniques that are now ubiquitous throughout the world. A pioneer investigation on the subject is the blinking vortices flow proposed by Aref [4]. This mixing flow is the two dimensional motion of an incompressible inviscid fluid, generated by two corrotating point vortices fixed in space that are switched alternatively on and off. This flow can be described analytically and mixing can be illustrated by Lagrangian tracking of markers, whose position of markers at all times

Recibido: 12 de abril de 2012
Aceptado: 10 de mayo de 2012

Keywords:
Difussive mixing; GPU accelerated simulation; real-time visualization.

Palabras clave:
Mezclado difusivo; simulación acelerada con GPU; visualización en tiempo real.

*Grupo de Modelos Climáticos. Centro de Ciencias de la Atmósfera. Universidad Nacional Autónoma de México. Ciudad Universitaria, Coyoacán, México, D. F., C. P. 04510. E-mail: asierra@unam.mx
**Departamento de Recursos Naturales. Instituto de Geofísica. Universidad Nacional Autónoma de México. Ciudad Universitaria, Coyoacán, México, D. F., C. P. 04510. E-mail: luiggi@unam.mx

can be determined numerically with a high degree of accuracy. Another mixing study that is particularly relevant for our analysis is the translating–rotating mixer introduced by Finn and Cox [5]. This mixer consists of a circular cylindrical vat inside which the fluid is stirred by a rod with circular cross section. The rod can be moved across the fluid and also can be rotated around its own axis. The dynamics of this system is similar to that of a vortex that moves describing a prescribed orbit inside the container.

In the present work, it is studied a mixing flow produced by a time dependent wall temperature in presence of a body force. These conditions generate a vortex of variable strength whose center moves around the container. This mixing protocol can be interpreted as a combination of a translating rotating mixer with blinking vortices. These conditions generate chaotic mixing flows where no moving walls are required. Lagrangian tracking of particles along with the diffusion of a concentration field are used to visualize and analize the mixing properties of the flow. To handle scalar diffusion on a moving substrate, the diffusive strip method introduced by Meunier *et al.*, [1] is used. Several authors have addresed this problem with different point of view, see for example [6–9]. The diffusive strip method requires too much time to calculate the concentration diffusion. In this work, that time is reduced using a Compute Unified Device Architecture (CUDA) kernel and displaying the diffused scalar field in real-time.

MATERIALS AND METHODS

Natural convection in a square cavity

Consider natural convection in a two dimensional square, with the temperature of the left half of the upper wall cyclically reduced and the right half of the bottom wall cyclically increased. Temperature changes are out-phased. The two vertical walls are adiabatic. An schematic representation of the geometry of the cavity, the coordinate system and the boundary conditions are shown in figure 1. The Boussinesq approximation is assumed to be valid and the state equation considered is $\rho = \rho_0(1 - \beta(T - T_0))$, where β is the volumetric expansion coefficient and the subindex o denotes a reference state. The governing equations in terms of non-dimensional variables are:

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \nabla\cdot(\mathrm{Pr}\,\nabla\mathbf{u}) + \mathbf{g}, \tag{2}$$

$$\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T = \mathrm{Pr}\nabla^2 T. \tag{3}$$

where g = (0. RaPrT). The spatial coordinates and time are scaled with the length of the enclosure side, L and L^2/α respectively, where α is the thermal diffusivity. The velocity vector $\mathbf{u} = (u, v)$ is scaled with α/L and the pressure p is scaled with $\rho\alpha^2/L^2$. Here ρ represents the fluid density. The temperature T is non-dimensionalized by $(T - T_m)/\Delta T$, $\Delta T = T_H - T_C$ and $T_m = (T_H + T_C)/2$. T_H and T_C are the maximum and minimum wall temperatures; $T_H = - T_C$. The Prandtl number is defined by $\mathrm{Pr} = \nu/\alpha$, while the Rayleigh (Ra) numbers is $\mathrm{Ra} = g\beta\Delta T L^3/\nu\alpha$. Here, the acceleration of gravity is g and ν is the kinematic viscosity.

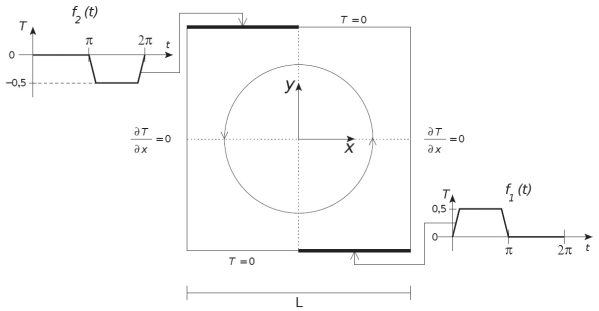

Figure 1. Square cavity geometry and boundary conditions

The origin of the coordinate system is taken at the geometrical center of the cavity and the boundary conditions are -see figure 1-:

$$\mathbf{u} = 0,\ \frac{\partial T}{\partial x} = 0 \quad \text{for } x = -0,5,\ 0,5 \text{ and } -0,5 \le y \le 0,5,$$
$$\mathbf{u} = 0,\ T = 0 \quad \text{for } -0,5 \le x \le 0 \text{ and } y = -0,5,$$
$$\mathbf{u} = 0,\ T = f_1(t) \text{ for } 0 \le x \le 0,5 \text{ and } y = -0,5,$$
$$\mathbf{u} = 0,\ T = f_2(t) \text{ for } -0,5 \le x \le 0 \text{ and } y = 0,5,$$
$$\mathbf{u} = 0,\ T = 0 \quad \text{for } 0 \le x \le 0,5 \text{ and } y = 0,5.$$

The first cycle of the periodic functions f_1 and f_2 is defined by the following expressions:

$$f_1(t) = \begin{cases} 0,5\sin^2(4t) & \text{for } 0 \le t < \pi/8 \\ 1 & \text{for } \pi/8 \le t < 7\pi/8 \\ 0,5\sin^2(4t - 3\pi) & \text{for } 7\pi/8 \le t < \pi \\ 0 & \text{for } \pi \le t < 2\pi \end{cases}$$

and

$$f_2(t) = \begin{cases} 0 & \text{for } 0 \le t < \pi \\ -0,5\sin^2(4t - 3\pi) & \text{for } \pi \le t < 9\pi/8 \\ 1 & \text{for } 9\pi/8 \le t < 15\pi/8 \\ -0,5\sin^2(4t - 6\pi) & \text{for } 15\pi/8 \le t < 2\pi \end{cases}$$

- Numerical solution

The conservation equations (1–3) were solved using the software TUNAM [10–11], which is a C++ library that implements the control volume method [12]. The QUICK [13] and SIMPLEC [14] techniques were

used to deal with the advective terms and to solve the coupled equations, respectively. The time integration was accomplished with a backward Euler scheme and the uniform discretization mesh contains 256^2 control volumes. In the examples presented here, they were used Pr = 5 and Ra = 10^5. All results presented below refer to the periodic motion that occurs once the initial transient has died out.

The oscillatory wall temperature imposed using the functions f_1 and f_2 yields the formation of alternating ascending and descending thermal plumes in regions close to the vertical walls. These structures induce a global motion that displays an initial transient followed by a periodic motion. The evolution of the temperature field inside the cavity is illustrated in figures 2 (a–d) where the upward and downward thermal plumes are shown at $\phi = \pi/2$ and $\phi = 3\pi/2$ respectively. Here ϕ represents the phase angle of the cycle.

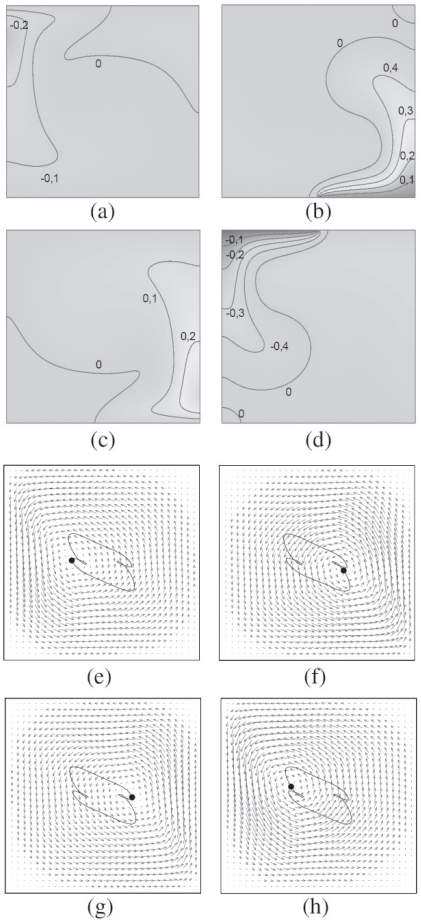

Figure 2. Temperature and velocity fields for: (a) and (e) $\phi = 0$; (b) and (f) $\phi = \pi/2$; (c) and (g) $\phi = \pi$; and (d) and (h) $\phi = 3\pi/2$. The line is the trajectory of the center of the vortex and the dot is the instantaneous position of the center of the vortex.

The ascending plume of hot fluid, and its interaction with the top wall, generates a vortical structure in the core of the cavity and, due to the periodic nature of the formation of the plumes and their asymmetric position, the center of the vortex defined by the point with zero velocity describes a curve that roughly encircles the center of the cavity -see figure 2 (e–h). The trajectory of the center of the vortex is also shown in figure 2 and its instantaneous position is denoted by the dots. The vortex moves with short spells of large velocities followed by long spells of small velocities. In the loop eyes, the vortex moves with the lowest speeds.

In summary, the overall effect of the alternating cold and hot plumes, produced with the protocol displayed in figure 1 in one full cycle, is the generation of a counterclockwise vortex with its center rotating around the geometrical center of the cavity in an orbit having more than two characteristic frequencies and maximum intensity occurring twice each cycle. This is the combination of blinking vortices with the translating–rotating mixer.

Advective–diffusive Mixing

The mixing efficiency of the flow described in section **Natural convection in a square cavity** can be qualitatively assessed from the simultaneous Lagrangian tracking of a set of N passive tracers in the flow, by integrating the equation of motion

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{u}(\mathbf{x}_i, t), \ \forall i = 1, \ldots, N, \tag{4}$$

where $\mathbf{x}_i$ is the position of the tracer i at instant t, which has the velocity $\mathbf{u}(\mathbf{x}_i, t)$ given by the numerical solution of the equations (1–3). Several integration methods can be used to find the position of each tracer. In this work, it is used the fourth order Runge-Kutta method.

Lagrangian tracking does not incorporate explicitly diffusion effects. On the other hand, Eulerian methods deal with the markers concentration field c by solving the partial differential equation:

$$\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c = \Gamma \nabla^2 c, \tag{5}$$

in this latter equation Γ represents the tracers diffusivity. In this work, the Diffusive Strip method introduced by Meunier *et al.* is applied[1]: the equation (5) is reduced to a simpler and analytically tractable diffusion equation. Figure 3 shows and schematic description of an initial strip and its deformation after k time steps. The striation thickness s_i^k, defined as the width of the strip at tracer $\mathbf{x}_i^k$, decreases with time and can be calculated numerically by applying the conservation of areas

$$s_i^k = s^0 \Delta x_i^0 / \Delta x_i^k. \quad (6)$$

The time can be counted in units of the current diffusion time $(s_i^k)^2/\Gamma$ in such a way that the next equation is valid

$$\frac{\partial \tau_i}{\partial t} = \frac{\Gamma}{(s_i^k)^2}. \quad (7)$$

Now, using τ_i and $\tilde{n}$ as new variables, equation 5 can be simplified to

$$\frac{\partial c}{\partial \tau} = \frac{\partial^2 c}{\partial \tilde{n}^2}. \quad (8)$$

Both τ and $\tilde{n}$ will be functions of space, time, scalar diffusivity and of the structure of the velocity field.

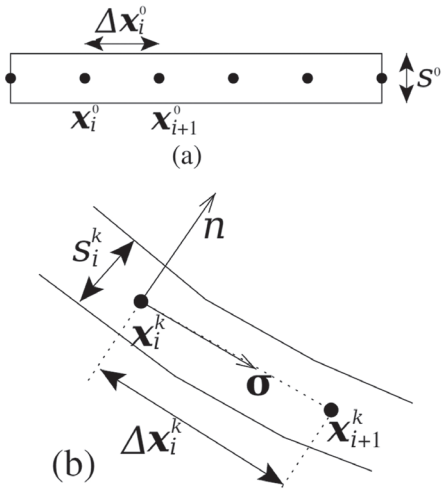

Figure 3. (a) A schematic strip defined by a set of tracers located at x_i for $i = 1,...,N$ with a striation thickness s^0. (b) A section of the strip deformed by the flow after k time steps.

Now, assuming that the initial concentration distribution on the strip has the Gaussian transverse profile $c(n) = c_0 e^{-n^2/(s^0)^2}$ for $t = 0$, then the solution to the equation (8) written in terms of n is

$$c^k(n) = \frac{c_0}{\sqrt{1+4\tau_i^k}} \exp\left(-\frac{n^2/(s_i^k)^2}{1+4\tau_i^k}\right). \quad (9)$$

The calculation of the scalar field c can be done by adding small Gaussian ellipses, of the form presented in equation (9), centered on each tracer. It is convenient that the tracers be equally spaced. The final formula to calculate c on a given position x of the domain at the instant k is

$$c^k(\mathbf{x}) = \sum_{i=1}^{N} \frac{c_0 / 1{,}7726}{\sqrt{1+4\tau_i^k}} \exp\left(-\frac{[(\mathbf{x}-\mathbf{x}_i^k)\hat{\sigma}_i^k]^2}{\Delta l^2} - \frac{[(\mathbf{x}-\mathbf{x}_i^k)\hat{n}_i^k]^2}{(s_i^k)^2(1+4\tau_i^k)}\right), \quad (10)$$

where $\hat{\sigma}_i^k$ and $\hat{n}_i^k$ are the unit vectors tangent and normal to the strip, which define the orientation of the major and minor axis of the Gaussian ellipses at instant k -see figure (3). The constant 1,7726 is due to the overlapping of the ellipses -see [1] for more details about the diffusive strip method.

▪ Algorithm for mixing analysis

The temperature T, pressure p, and velocity **u** fields are obtained numerically solving the governing equations (1–3), as it was described in section **Numerical solutions**. Those fields were obtained on mesh for each time step. In order to visualize and analize the advective–diffusive mixing of an strip immersed in the flow, the **u** vector field as input to the next algorithm was used.

- Initialization of the geometry of the strip -see figure 3-
  - Determine the initial position x_i^0 of the tracers, for $i = 1,...,N$ and define the lentgh Δx_i^0 as the initial separation between them.
  - The shape of the original strip is defined by connecting the N tracers.
  - Define the initial width of the strip s^0.
  - Define a mesh of $Nxf \times NyF$ nodes on the domain of study to calculate the concentration c.

- **FOR** $k = 0$ **TO** N_{MAX} **DO**

  1. Read the velocity field $\mathbf{u}^k$ defined on the solution mesh.
  2. **FOR** $i = 1$ **TO** N **DO**
     i. Interpolate $\mathbf{u}^k$ from the mesh to the tracer position x_i.
     ii. Integrate equation (4) to obtain the new position x_i^{k+1}.
     iii. Calculate s_i^k using (6).
     iv. Integrate equation (7) using s_i^k as follows: $\tau_i^k = \tau_i^{k-1} + \Gamma\delta t / s_i^k$.

     END FOR
  3. Draw the deformed strip by joining all the positions x_i^{k+1}.

4. **FOR** $ix = 1$ **TO** Nyf **DO**

. **FOR** $jy = 1$ **TO** Nxf **DO**

.. **FOR** $i = 1$ **TO** N **DO**

i. Calculate $(x_{ix,iy} - x_i^k)$, $\hat{o}_i^k$ and $\hat{n}_i^k$.

ii. Evaluate $c^k(x_{ix,iy})$ using (10).

.. **END FOR**

. **END FOR**

END FOR

END FOR

In the above algorithm, the calculation of the concentration, the **FOR** loops in line 4, take the major time of the complete process, due to the fact that Nxf and Nyf need to be 256 or greater to obtain good accuracy. It can be observed that the calculation of c for every point on the mesh, defined by Nxf and Nyf, is independent of each other. Therefore, these loops and the visualization can be done in the Graphics Processors Units (GPU) directly. Results of this algorithm for both CPU and GPU versions are shown in figure 4.

RESULTS

Concentration calculation on the GPU

The conentration calculation exhibits rich amount of data parallelism in such a way that it can be obtained a good speedup doing this process in the GPU. It is used *CUDA C*, the computer language used to program NVIDIA's high performance GPU, to implement a special kernel to calculate c. In order to represent the 2D grid of dimension $Nxf \times Nyf = 512 \times 512$, a grid of 32 by 32 blocks, each block of 16 by 16 threads, is executed. Each thread computes results for a single point in the grid. This thread configuration gives a high-performance balance between the number of threads per block, the maximum number of blocks per multiprocessor in the chip and the number of per thread resources available.

The calculation of c^k requires x_i^k, s_i^k and τ_i^k. These quantities are evaluated in lines 2(i), 2(ii) and 2(ii) respectively in the CPU. They were transfered these values to the GPU own Dynamic Random Access Memory (DRAM) using the typical commands `cudaMalloc` and `cudaMemcpy`. No other variables are required, however, this data transfer is done each time step. The next extract of the code shows the basics to construct the arrays, transfer the data to the GPU and to call the CUDA kernel

```
cudaMalloc((void**) &linec, 2*size);
cudaMalloc((void**) &s, size);
cudaMalloc((void**) &tao, size);
cudaMalloc((void**) &c, sizeC);
cudaMemcpy(linec, linec_h, 2*size, cudaMemcpyHostToDevice);
cudaMemcpy(s, s_h, size, cudaMemcpyHostToDevice);
cudaMemcpy(tao, tao_h, size, cudaMemcpyHostToDevice);
dim3 DimGrid(32, 32, 1);
dim3 DimBlock(16, 16, 1);
concentracion<<<DimGrid, DimBlock>>>(pixels, c, linec, s, tao,
                                     line_size, deltaL0, Nxf);
```

Notice that arrays for storing x_i^k, s_i^k and τ_i^k (`linec`, `s` and `tao` respectively), are inputs for the CUDA kernel and do not require to be transfered back to CPU. The array `c`, which store the concentration, will be used for the visualization in real-time.

The CUDA kernel comprises the two first **FOR** loops in line 4 of the pseudocode presented in section **Algorithm for mixing analysis**. It is just needed to determine the indexes for each thread as usual

```
int i = blockIdx.x * blockDim.x + threadIdx.x;
int j = blockIdx.y * blockDim.y + threadIdx.y;
```

Then, it is possible to update every position of the array `c`, which will contains the concentration, using `i` and `j` in a construction of the form `c[Nxf * j + i] += conc_eval(...)`.

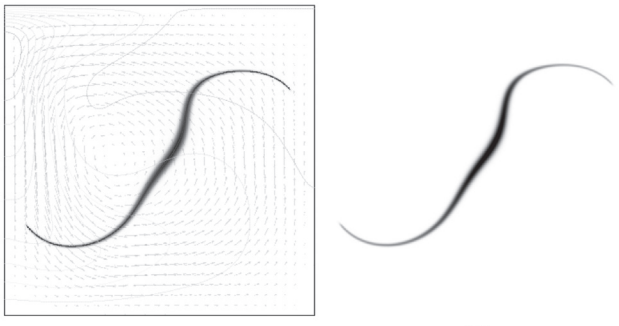

Figure 4. Concentration c displayed on gray levels after 1 cycle of heating-cooling as described in section **Natural convection in a square cavity**. (a) Result obtained with the CPU version of the algorithm: temperature contours and velocity displayed along the distribution of c. The blue line shows the actual position of the tracers. (b) Result obtained with the real-time visualization algorithm: distribution of c is displayed.

▪ Speedup

The algorithms developed in this work were tested on two different architectures: (a) PC with four processors to 2,00 GHz and 4 GB in RAM. This computer has a GPU QUADRO FX 3800M with 128 Cores and 1 024 MB in RAM; (b) Server with 8 processors to 3,47 GHz and 4GB in RAM. This server contains a TESLA C1060 with 240 Cores and 2 048 MB in RAM.

The data for the examples are: N = 1 000 (tracers), $Nxf = Nyf$ = 512, s^0 = 0,01. The tracers were uniformly located initially from x = 0,05 to x = 0,95, and y = 0,5. It was runned the code for several cycles of the functions f_1 and f_2 described in section **Natural convection in a square cavity**. However, only one cycle is required to make an speedup analysis.

Tables 1 and 2 show the times in seconds for different parts of the code. As can be seen, the concentration calculation takes more than the 95 % of the total computation. It is also constatable that the kernel implemented drastically reduce this time. For the QUADRO GPU, it was got an speedup of 14X, while for the TESLA GPU the speedup is roughly 62X. The time measurements includes the memory transfer between CPU and GPU.

Besides, in the CPU case, it was needed to store c in many files so the post-processing of the results can be done. On the other hand, the CUDA version use the c stored in the GPUs own RAM to visualize the results immediately after its calculation -see section **Real-time visualization**-, in such a way that it is not needed to store this information in the hard disk.

Table 1.
Results for the GPU FX QUADRO FX 3800M. Times in seconds for different parts of the code. One cycle of the process is measured.

	Average	Standard deviation	Total
Tracking	1,25e-05	3,535e-04	0,01
Concentration CPU	19,2922	5,379e-01	15 433,8
Concentration GPU	1,18278	3,868e-02	946,22
File reading	0,14240	9,939e-03	113,91
Total time CPU	15 741,883s		
Total time GPU	1 123,002s		
Total time CPU / Total time GPU = 14,017			

Table 2.
Results for the TESLA C1060. Times in seconds for different parts of the code. One cycle of the process is measured..

	Average	Standard deviation	Total
Tracking	2,05e-03	4,375e-04	0,07
Concentration CPU	15,8654	3,677e-01	2 538,47
Concentration GPU	0,14025	1,341e-02	22,44
File reading	0,11200	5,806e-03	17,92
Total time CPU	2 576,929s		
Total time GPU	41,448s		
Total time CPU / Total time GPU = 62,172			

The acceleration obtained using the GPU gives the opportunity to combine the algorithms with a real-time visualization process. Next section explain the combination of CUDA and OpenGL done for this purpose.

▪ Real-time visualization

Once the concentration has been calculated and stored in the array c in the GPU memory, the advantage of CUDA and OpenGL interoperability are taken to visualize and analyze the results in real-time. First, an OpenGL Pixel Buffer Object (PBO) is created and memory space for a texture with the same dimension, as the grid used to compute the concentration, is allocated. Then the PBO was mapped to a CUDA device memory pointer using `cudaGLMapBufferObject()`. This allows it to be passed to the CUDA kernel, which will create an image representing the concentration as gray levels (white representing zero concentration). The PBO is then unmapped from the device pointer so it can be used as a normal OpenGL texture using `glBindTexture()`.

For each iteration, the visualization is updated -see figure 4. To create and manage an OpenGL context, it has been using the modern GLFW [15] instead of GLUT, which actually is outdated. In this way, it is possible to have a better control over the graphics loop without many changes to the previous non graphics code.

CONCLUSIONS

It has been demonstrated with an example that mixing in cavities can be achieved by changing the temperature of vertical sidewalls in an appropriate manner, taking advantage of the physical properties of the fluid to generate the motion. This mixing method does not require moving external parts. A very important aspect is that this problem requires a thorough analysis of the dynamic behavior after hundreds of cycles. As it has be shown, the computing of the concentration could take several hours to complete a simple cycle. However, using a very simple CUDA kernel and a real-time visualization process, it is possible to study the problem for several cycles in relative short times. Particularly, in this problem it has been able to analyze 16 cycles in less time than that required for 2 cycles in the CPU version. This study is not simple, not only for the extremely long computing times required, but also because the error propagation may put very stringent demands on the accuracy of the calculations. At this time, it was used a simple precision version on GPUs, but any inconsistencies compared to the CPU

version are found. Of course, for hundreds of cycles it is needes to extrapolate our CUDA kernels to double precision. Hopefully, a better performance on the most actual GPUs (TESLA C2075 for example) is expected.

REFERENCES

[1] Meunier, P. and Villermaux, E. (2010). The diffusive strip method for scalar mixing in two dimensions. *J. Fluid Mech.* 662: pp. 134-172.

[2] Ottino, J. M. (1989). *The kinematics of mixing: stretching, chaos and transport*. Cambridge University Press.

[3] Ottino, J. M. (1990). Mixing, Chaotic Advection, and Turbulence. *Annu. Rev. Fluid Mech.* 22: pp. 207–253.

[4] Aref, H. (1984). Stirring by chaotic advection. *J. Fluid Mech.* 143: pp. 1–21.

[5] Finn, M. D. and Cox, S. M. (2001). *J. Engng Math.* 41: pp. 75–99.

[6] Sukhatme, J. and Pierrehumbert, R. T. (2002). Decay of passive scalars under the action of single scale smooth velocity fields in bounded two-dimensional domains: From non-selfsimilar probability distribution functions to self-similar eigenmodes. *Phys. Rev.* E 66.

[7] Fereday, D. R. and Haynes, P. H. (2004). Scalar decay in two-dimensional chaotic advection and batchelor-regime turbulence. *Phys. Fluids* 16(12): pp 4359-4370.

[8] Perugini, D., Ventura, G., Petrelli, M. and Poli, G. (2004). Kinematic significance of morphological structures generated by mixing of magmas: a case study from salina island (southern italy). *Earth and Planetary Sci.* Lett. 222: pp. 1051-1066.

[9] Shankar, P. N. and Kidambi, R. (2009). Mixing in internally stirred flows. *Proc. Roy. Soc.* A 465: pp. 1271-1290.

[10] de la Cruz, L. M. (2005). *Parallel computing for solving the balance equations in turbulent flow.* PhD thesis. Universidad Nacional Autónoma de México (UNAM). Mexico.

[11] de la Cruz, L. M. and Ramos. E. (2012). *General Template Units for the Finite Volume Method in Box-shaped Domains.* Trans. on Math. Software. To be printed.

[12] Versteeg, H. and Malalasekera, W. (1995). *An introduction to computational fluid dynamics*: The Finite volume method. Longman.

[13] Leonard, B. P. (1979). A stable and accurate conevctive modelling procedure based on quadratic upstream interpolation. *Comp. Meth. in App. Mech. and Engineering* 19: pp. 59–98.

[14]. Doormal, J. V and Raithby, G. D. (1984). *Enhancements of the simple method for predicting incompressible fuid flows.* Num. Heat Transfer 7: pp. 147–163

[15] Geelnard, Marcus and Berglund, Camilla GLFW: *User Guide, API version 2.7.3*, http://www.glfw.org/, consulted: 13 February 2012