

## Diffusion Equation Solver with CUDA Implementation

### Problem description:

Partial differential equations are commonly used to model the flow of material in and out of a given region. One commonly used in Computational Fluid Dynamics is the **diffusion equation**

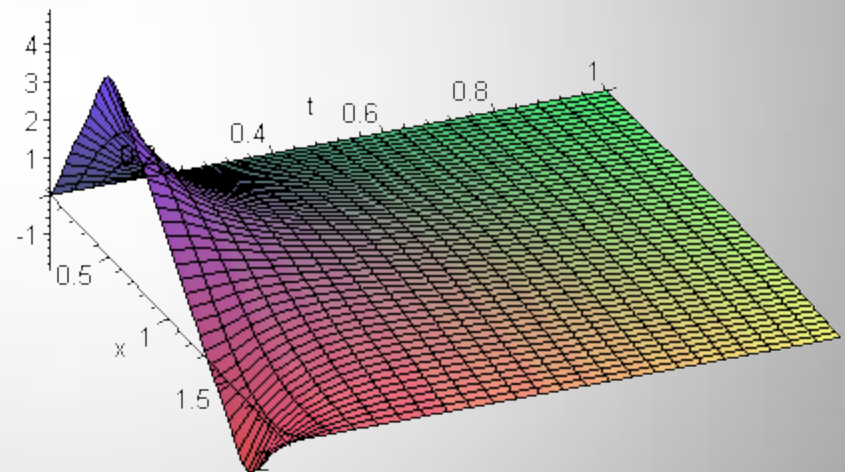
$$\frac{\partial \varphi(r, t)}{\partial t} = \nabla \cdot [D(\varphi, r) \nabla \varphi(r, t)]$$

Since real world problems are often too difficult to solve analytically, a common method used to approximate values is the finite difference method. Partial derivatives are estimated by finding the difference between values at neighboring points on the structured grid, then dividing by the spatial spacing or time step.

$$\frac{\partial \varphi}{\partial x} \approx \frac{\varphi_{i+1,j} - \varphi_{i-1,j}}{\delta x}$$

$$\frac{\partial \varphi}{\partial t} \approx \frac{\varphi_{i,j} - \varphi_{i,j-1}}{\delta t}$$

$$\frac{\partial \varphi^2}{\partial x^2} \approx \frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{(\delta x)^2}$$



## Basic serial approach:

The problem solution involves constant recalculation of all points until the desired time is reached. A serial Fortran77 code was used to calculate values for the 1D diffusion equation using various grid spacing to test for scalability. Below is the portion on the code which recalculates the point values at each time step:

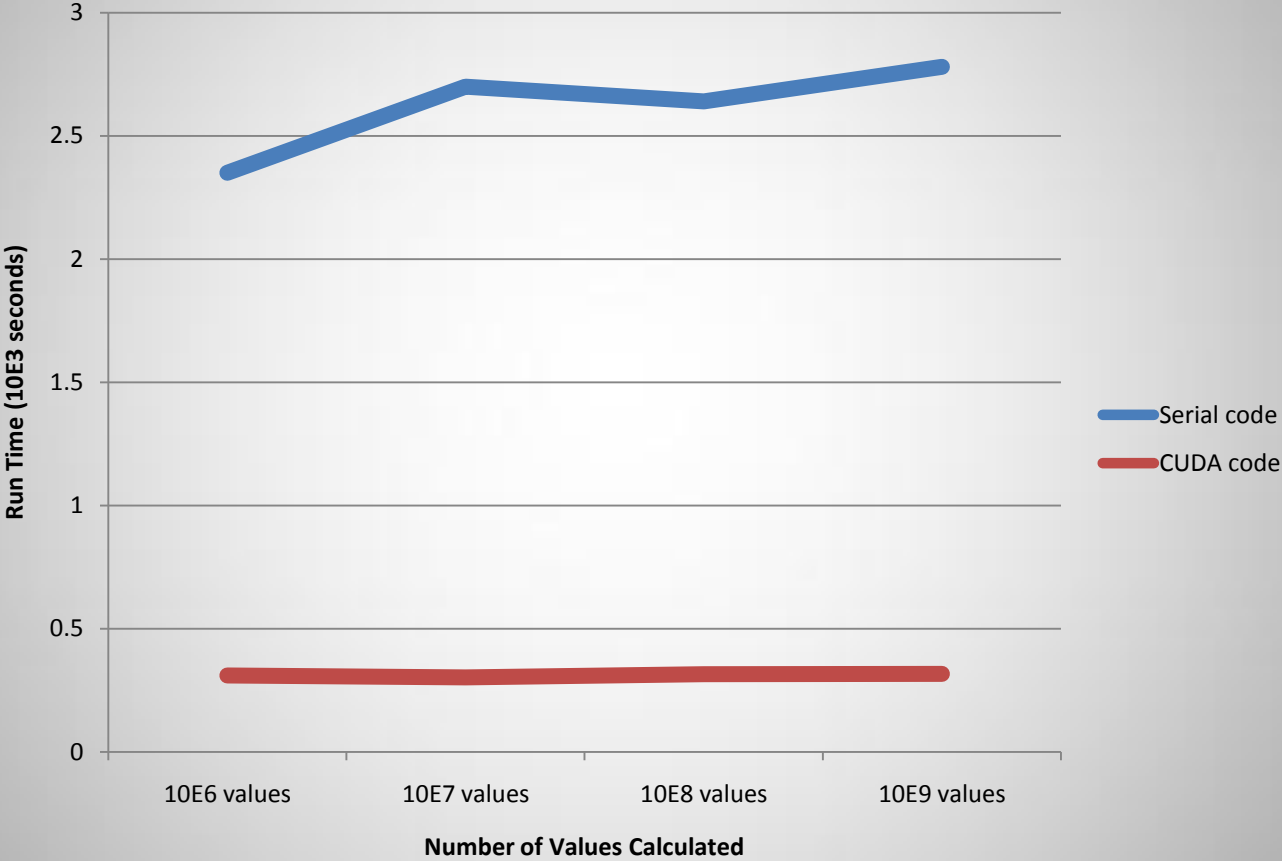
```
do i = 1, n
    h_new(i) = h(i) &
        + ( time_delt * k / x_delx / x_delx )
&
        * ( h(i-1) - 2.0 * h(i) + h(i+1) ) &
        + time_delt * BC ( x(i), time )
end do
```

## **Parallel approach and hardware considerations:**

One of the largest manufacturers of graphics cards, NVIDIA, has developed a library of commands to enable a programmer to access the parallel capability of their second generation graphics cards for mathematical calculations. Operations still need to go through the computer's CPU, and only those by Intel currently support this technology (AMD chips do not.) These graphics cards also use more power, so a minimum 600W power supply is needed. Since this problem is commonly solved, a very complete public domain solution already exists. Even just implementing this program took a considerable amount of work, but it was used to demonstrate the potential of CUDA. For this report, a GeForce GT 220 NVIDIA graphics card was used, on a computer with a dual core Intel Celeron processor 450.



# Performance:



## **Lessons learned, level of difficulty and future:**

- CUDA programming is definitely not for beginners
- The hardware requirements for this are not trivial, but thanks to the gaming industry, are on their way to becoming mainstream.
- In June 2009, the Portland Group and NVIDIA announced they were working together to create CUDA Fortran. It has since been released, and is now for purchase. This was created as a variation of the Fortran 2003 compiler.