Implicit_Lab

December 6, 2023

Massive parallel programming on GPUs and applications, by Lokman ABBAS TURKI

1 7 Implicit simulation scheme for Black and Scholes Partial Differential Equation

1.1 7.1 Objective

This is the second lab of a series of four labs dedicated to the simulation of Parabolic Partial Differential equations using discretization schemes. We continue with the implicit scheme which is more difficult to make parallel than the explicit scheme but has the property to be stable with respect to time discretization. The parallelization strategy is uniquely performed with respect to . Indeed, we will be implementing Thomas's method which is very serial. This method is needed to solve the tridiagonal systems induced by the discretization scheme at each time step. The NP function is used to check the simulation results and thus it should not be modified.

As usual, do not forget to use CUDA documentation, especially:

1) the specifications of CUDA API functions within the CUDA_Runtime_API.
2) the examples of how to use the CUDA API functions in CUDA_C_Programming_Guide

1.2 7.2 Content

Compile PDE.cu using

[ ]:
```
!nvcc PDE.cu -o PDE
```

Execute PDE using (on Microsoft Windows OS ./ is not needed)

[ ]:
```
!./PDE
```

As long as you did not include any additional instruction in the file PDE.cu, the execution above is supposed to return incorrect values on the left column. The right column is supposed to contain the true results that we should approximate with the discretization scheme.

1.2.1 7.2.1 Thomas's method

We ask here to implement this method specifically to the studied problem

a) What should be the input arguments of the device function Thomas_d.

b) Complete the syntax of the device function Thomas_d.

1.2.2 7.2.2 PDE_diff_k4 and memory copy

In a lecture video and slides, we showed how we obtained the tridiagonal system associated with the implicit scheme. The time loop is better placed in the kernel PDE_diff_k4.

a) Justify the allocation of 2*sizeof(MyTab) for the array on the device.

b) Write the necessary code for CPU2GPU and GPU2CPU memory copy.

c) Complete the syntax of the kernel PDE_diff_k4.

d) Compare the execution time of the solution involving PDE_diff_k4 to the one involving PDE_diff_k3 using !nvprof ./PDE.

e) What can be done to make PDE_diff_k4 competitive when compared to PDE_diff_k3?

[ ]: