1 Lecture 1 – Introduction

1.1 Course Introduction and Examples with Derivative Pricing

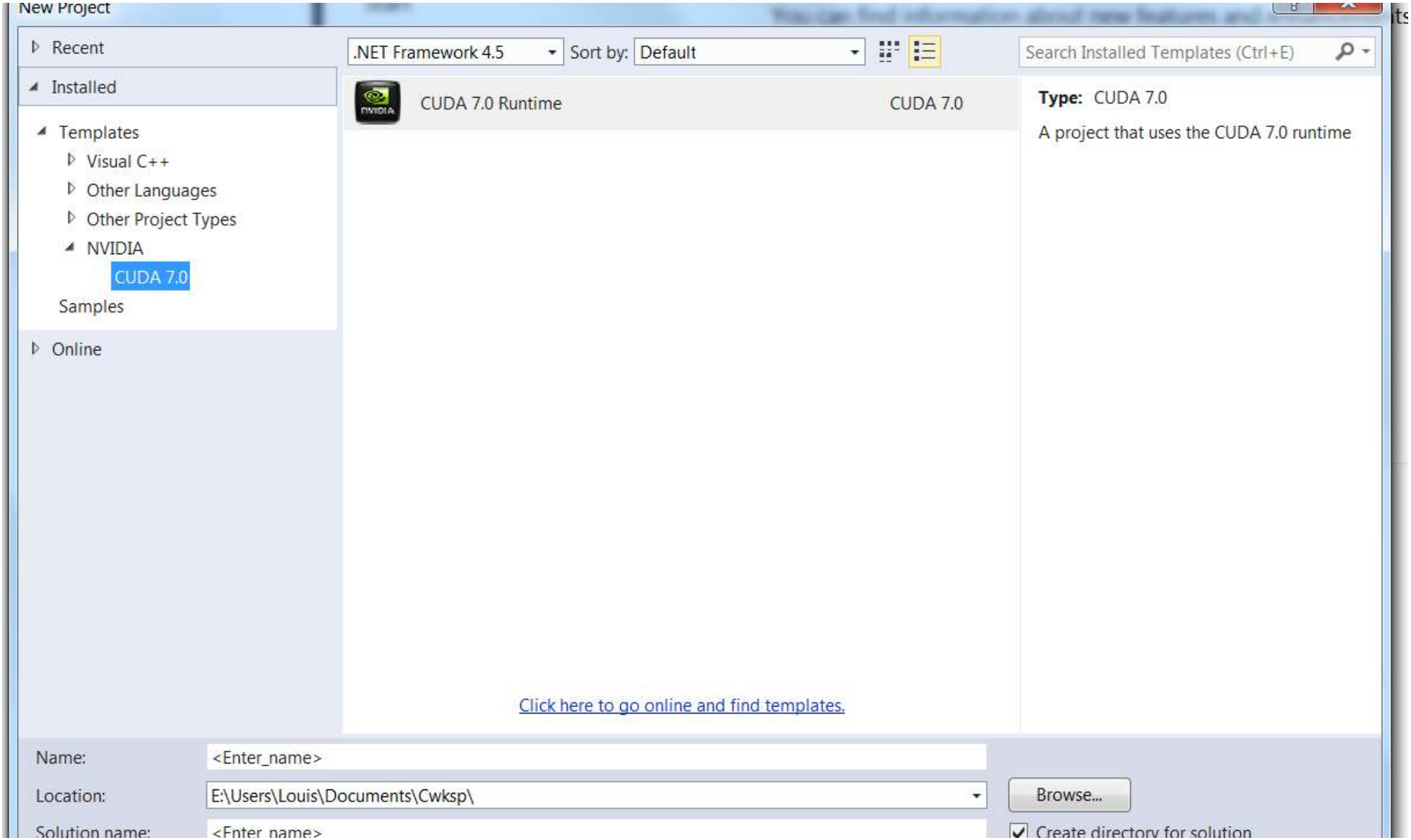
1.2 Lab Examples – Introduction to CUDA C Programming

A. Students should check PC's to determine if PC has a Cuda capable GPU
   Go to https://developer.nvidia.com/cuda-gpus

B. Vector addition

   Open Microsoft Visual Studio and start a New Project using
   Nvidia Cuda

   → TestGPU example


New Project
Recent
Installed
Templates
Visual C++
Other Languages
Other Project Types
NVIDIA
CUDA 7.0
Samples
Online
.NET Framework 4.5
Sort by:
Default
Search Installed Templates (Ctrl+E)
CUDA 7.0 Runtime
CUDA 7.0
Type: CUDA 7.0
A project that uses the CUDA 7.0 runtime
Click here to go online and find templates.
Name:
<Enter_name>
Location:
E:\Users\Louis\Documents\Cwksp\
Browse...
Solution name:
<Enter_name>
Create directory for solution

```cuda
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <stdio.h>

cudaError_t addWithCuda(int *c, const int *a, const int *b, unsigned int size);

__global__ void addKernel(int *c, const int *a, const int *b)
{
    int i = threadIdx.x;
    c[i] = a[i] + b[i];
}

int main()
{
    const int arraySize = 5;
    const int a[arraySize] = { 1, 2, 3, 4, 5 };
    const int b[arraySize] = { 10, 20, 30, 40, 50 };
    int c[arraySize] = { 0 };

    // Add vectors in parallel.
    cudaError_t cudaStatus = addWithCuda(c, a, b, arraySize);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "addWithCuda failed!");
        return 1;
    }

    printf("{1,2,3,4,5} + {10,20,30,40,50} = {%d,%d,%d,%d,%d}\n",
        c[0], c[1], c[2], c[3], c[4]);
```

Lecture 1 – Introduction (continued)

function_kernel <<< (blocksize), (threads per block) >>> (arguments, ……) ;

The kernel calls a function that will execute on the GPU. The arguments of the function can be used to pass inputs and return results. Results are typically handled in pointers. Blocksize and threads per block are used on the GPU. Threads per block must not exceed 1024 (max = 1024). We will discuss different strategies for setting blocksize and threads per block.

One can also allocate memory on the GPU and copy input values or data to GPU device memory

```
int *a, Size = 512;
int *d_a;

a = (int *)malloc(Size * sizeof(int));

cudaMalloc((void **)&d_a, Size * sizeof(int));

cudaMemcpy(d_a, a, Size * sizeof(int), cudaMemcpyHostToDevice);

Do not forget to release memory (normally at end of program)
```

See Test_GPU.cu under NYU Classes → Resources → Sample Code

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <stdio.h>
void VectorAdd(int *a, int *b, int *c, int n) {
    int i;
    for (i = 0; i < n; i++)
        c[i] = a[i] + b[i];
}
__global__ void VectorAddKernel(int *a, int *b, int *c, int n)
{
    int i = threadIdx.x;
//    c[i] = a[i] + b[i];
    if (i < n) {
        c[i] = a[i] + b[i];
    }
}
int main()
{
    int i, Size = 1024;
    int *a, *b, *c;
    int *d_a, *d_b, *d_c;
    a = (int *)malloc(Size * sizeof(int));
    b = (int *)malloc(Size * sizeof(int));
    c = (int *)malloc(Size * sizeof(int));
    cudaMalloc((void **)&d_a, Size * sizeof(int));
    cudaMalloc((void **)&d_b, Size * sizeof(int));
    cudaMalloc((void **)&d_c, Size * sizeof(int));
```

```
    for (i = 0; i < Size; i++) {
        a[i] = i;
        b[i] = 2*i;
        c[i] = 0;
    }

    VectorAdd(a, b, c, Size);

    for (i = 0; i < 10; i++) printf(" a, b, c row %4i  %4d %4d %4d \n", i, a[i], b[i], c[i]);
    i = Size - 1;
    printf(" a, b, c row %4i  %4d %4d %4d \n", i, a[i], b[i], c[i]);

    printf("\n   Now rerun the calculations on GPU \n");

//  Copy vectors a and b to device memory d_a, d_b
    cudaMemcpy(d_a, a, Size * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, Size * sizeof(int), cudaMemcpyHostToDevice);

// Launch a kernel on the GPU with one thread for each element.
    VectorAddKernel <<<1, Size >>>(d_a, d_b, d_c, Size);

    cudaGetLastError();
    cudaDeviceSynchronize();
    cudaMemcpy(c, d_c, Size*sizeof(int), cudaMemcpyDeviceToHost);

    for (i = 0; i < 10;i++) printf(" a, b, c row %4i  %4d %4d %4d \n", i, a[i], b[i], c[i]);
    i = Size - 1;
    printf(" a, b, c row %4i  %4d %4d %4d \n", i, a[i], b[i], c[i]);
```

```
    free(a);
    free(b);
    free(c);

    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_c);

    cudaDeviceReset();

    return 0;
}
```

1 Lecture 1 – Introduction (continued)

1.2 Lab Examples – Introduction to CUDA C Programming

B. Matrix multiplication

See example Test_MatrixMultiplication.cu in NYU Classes → Resources → Sample Code