# Projeto 1: C++ e SIMD

### Filipe F. Borba

### Insper

### Super Computação, Prof. Igor Montagner

## Introdução ¶

O problema explorado nesse projeto é o da simulação de física 2D. Foi feita uma simulação de corpos retangulares num plano. Com isso, para aumentar o número de corpos na simulação ou melhorar a performance dos cálculos, podemos usar técnicas de Super Computação.

O modelo a ser simulado supõe que todos os corpos são retângulos que possuem massa, posição, velocidade e aceleração, porém sem velocidade angular (para simplificar). Seguindo essa linha, as colisões no modelo seguem duas simplificações. A primeira é que elas são sempre totalmente elásticas. A segunda simplificação será na detecção de colisões. A cada passo da simulação a posição do corpo no próximo instante de tempo é calculada. Se houver uma colisão, os vetores de velocidade mudarão, mas eles não serão movidos. Ou seja, em cada iteração a posição de um corpo só é atualizada se ele não colidir com outro corpo.

O ambiente de simulação possui uma largura e altura dada pela entrada do programa, juntamente com um coeficiente de atrito. As colisões com as bordas apenas "rebatem" o retângulo na direção oposta. No fim, a simulação acaba quando o módulo da velocidade de todos os corpos for menor que 0,0001 m/s ou o número máximo de iterações é alcançado.

Finalmente, os objetivos deste projeto são

- implementar um projeto de média complexidade (em termos de especificações técnicas) em C++;
- estudar efeito de opções de compilação e de vetorização em um projeto mais complexo que as atividades de sala de aula;
- montar uma comparação de desempenho reprodutível e descritiva.

** Como descrito em https://github.com/Insper/supercomp/wiki/Projeto1 (https://github.com/Insper/supercomp/wiki/Projeto1)

# Organização do Projeto

O projeto foi realizado utilizando a linguagem C++ e o compilador g++ do Ubuntu Linux com opções de otimização de código e vetorização diferentes. Basicamente temos duas classes: Rectangle e Simulator separadas nos arquivos rectangle.cpp/hpp e simulator.cpp/hpp.

A classe Rectangle possui toda a lógica do corpo de retângulo, com atributos como massa, largura, altura, posicao e velocidade. A segunda classe possui toda a lógica do simulador, com funções específicas para manipular os retângulos, printar relatórios e etc. além de guardar atributos importantes como a área e parâmetros (atrito, iterações, número de retângulos) da simulação. É nessa classe que a simulação ocorre de fato.

Além disso, o projeto possui um CMakeLists.txt que possibilita a compilação de diferentes executáveis com diferentes opções de otimização e vetorização. São eles:

- simulator
- simulator_O0
- simulator_O1
- simulator_O2
- simulator_O3
- simulator_O3_SIMD

Para compilar todos os executáveis, basta configurar usar os seguintes comandos na pasta raíz do projeto: `mkdir build; cd build; cmake ..; make` . O comando `make` é responsável por compilar os executáveis. Após isso, para iniciar cada executável, pasta utilizar o comando `./build/[nome_do_arquivo] < entrada.txt` .

# Resultados

## Testes Básicos

Nesta seção, vamos testar os diferentes executáveis com alguns testes. Os primeiros são bem básicos servem apenas para verificar que os resultados estão de acordo com o esperado. As saídas podem ser verificadas na pasta do projeto.

In [1]:

```
%matplotlib inline
import os
import subprocess

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:

```
# Pegar o nome dos executaveis
executables = sorted([n for n in os.listdir("./build/") if n.startswith('simulat
or') and n != 'simulator'])
executables
```

Out[2]:

```
['simulator_O0',
 'simulator_O1',
 'simulator_O2',
 'simulator_O3',
 'simulator_O3_SIMD']
```

In [3]:

```
# Pegar o nome das entradas
inputs = sorted([n for n in os.listdir("./") if n.startswith('supercomp')])
inputs
```

Out[3]:

```
['supercomp_test100.txt',
 'supercomp_test1000.txt',
 'supercomp_test300.txt',
 'supercomp_test500.txt']
```

In [4]:

```
def run_basic_test(executable, input_file):
    with open('./' + input_file, 'rb', 0) as f:
        output = subprocess.check_output(['./build/' + str(executable)], stdin=f
)
        output = output.decode("utf-8").splitlines()

    # Se for um arquivo com nome "entrada" vamos printar o resultado
    print("-------Basic Test------------")
    print("Executable: " + executable)
    print("Input File: " + input_file)
    print("\n".join(output))
    print("-----------------------------")
```

In [5]:

```
# Vamos rodar apenas um teste básico para verificar que o programa funciona.
# run_basic_test(executables[-1], inputs[0])

# Caso queira rodar todos os testes básicos, use:
inputs = sorted([n for n in os.listdir("./") if n.startswith('entrada')])
for i in inputs:
    run_basic_test(executables[-1], i)
```

```
-------Basic Test------------
Executable: simulator_O3_SIMD
Input File: entrada1
----------Parameters----------
Simulation Grid
Width: 800 Height: 600 Friction: 0.1
Steps (dt): 1 Print Frequency: 1 Max Iterations: 1

10 10 100 0
--------End Parameters--------

0
110 10 99.02 0
-------------------
--------Final Report-------
Maximum Iterations: 1
Is Stable? No

1
110 10 99.02 0
--------End Simulation--------
Total time taken in seconds:
1.9093e-05
Number of rectangles:
1
-----------------------------
-------Basic Test------------
Executable: simulator_O3_SIMD
Input File: entrada2
----------Parameters----------
Simulation Grid
Width: 800 Height: 600 Friction: 0.1
Steps (dt): 1 Print Frequency: 1 Max Iterations: 1

10 10 0 100
--------End Parameters--------

0
10 110 6.06323e-15 99.02
-------------------
--------Final Report-------
Maximum Iterations: 1
Is Stable? No

1
10 110 6.06323e-15 99.02
--------End Simulation--------
Total time taken in seconds:
2.8727e-05
Number of rectangles:
1
-----------------------------
-------Basic Test------------
Executable: simulator_O3_SIMD
Input File: entrada3
----------Parameters----------
Simulation Grid
Width: 800 Height: 600 Friction: 0.1
Steps (dt): 1 Print Frequency: 1 Max Iterations: 1

10 10 100 100
```

```
--------End Parameters--------

0
110 110 99.307 99.307
--------------------
--------Final Report-------
Maximum Iterations: 1
Is Stable? No

1
110 110 99.307 99.307
--------End Simulation--------
Total time taken in seconds:
2.3719e-05
Number of rectangles:
1
-----------------------------
-------Basic Test------------
Executable: simulator_O3_SIMD
Input File: entrada4
----------Parameters----------
Simulation Grid
Width: 20 Height: 20 Friction: 0
Steps (dt): 1 Print Frequency: 1 Max Iterations: 4

13 13 1 0
--------End Parameters--------

0
14 13 1 0
--------------------
1
14 13 -1 0
--------------------
2
13 13 1 -0
--------------------
3
14 13 1 -0
--------------------
--------Final Report-------
Maximum Iterations: 4
Is Stable? No

4
14 13 1 -0
--------End Simulation--------
Total time taken in seconds:
3.6668e-05
Number of rectangles:
1
-----------------------------
-------Basic Test------------
Executable: simulator_O3_SIMD
Input File: entrada5
----------Parameters----------
Simulation Grid
Width: 20 Height: 20 Friction: 0
Steps (dt): 1 Print Frequency: 1 Max Iterations: 4

13 18 0 1
```

```
--------End Parameters--------

0
13 19 6.12323e-17 1
-------------------
1
13 19 6.12323e-17 -1
-------------------
2
13 18 6.12323e-17 -1
-------------------
3
13 17 6.12323e-17 -1
-------------------
--------Final Report-------
Maximum Iterations: 4
Is Stable? No

4
13 17 6.12323e-17 -1
--------End Simulation--------
Total time taken in seconds:
4.3576e-05
Number of rectangles:
1
------------------------------
-------Basic Test------------
Executable: simulator_O3_SIMD
Input File: entrada6
----------Parameters----------
Simulation Grid
Width: 20 Height: 20 Friction: 0
Steps (dt): 1 Print Frequency: 1 Max Iterations: 5

7 7 0 -1
--------End Parameters--------

0
7 6 6.12323e-17 -1
-------------------
1
7 5 6.12323e-17 -1
-------------------
2
7 4 6.12323e-17 -1
-------------------
3
7 4 6.12323e-17 1
-------------------
4
7 5 6.12323e-17 1
-------------------
--------Final Report-------
Maximum Iterations: 5
Is Stable? No

5
7 5 6.12323e-17 1
--------End Simulation--------
Total time taken in seconds:
5.1873e-05
```

```
Number of rectangles:
1
-----------------------------
-------Basic Test------------
Executable: simulator_O3_SIMD
Input File: entrada7
----------Parameters----------
Simulation Grid
Width: 20 Height: 20 Friction: 0
Steps (dt): 1 Print Frequency: 1 Max Iterations: 5

2 7 -1 0
--------End Parameters--------

0
1 7 1 -0
-------------------
1
2 7 1 -0
-------------------
2
3 7 1 -0
-------------------
3
4 7 1 -0
-------------------
4
5 7 1 -0
-------------------
--------Final Report-------
Maximum Iterations: 5
Is Stable? No

5
5 7 1 -0
--------End Simulation--------
Total time taken in seconds:
3.7288e-05
Number of rectangles:
1
-----------------------------
```

## Testes de Desempenho

Já os arquivos com nome supercomp_testxxx.txt possuem numeros aleatórios para que a máquina trabalhe ao máximo calculando os números. Aqui estamos preocupados com a diferença de desempenho, então o tamanho das entradas é o número do arquivo e o tempo de execução foi medido a partir da biblioteca e o high_resolution_clock. Elas possuem um dt de 0.001 e máximo número de iterações de 100000.

Para criar as próprias bases de testes, basta usar e modificar o arquivo `test_generator.py` .

In [6]:

```
def run_test(executable, input_file):
    with open('./' + input_file, 'rb', 0) as f:
        output = subprocess.check_output(['./build/' + str(executable)], stdin=f
)
        output = output.decode("utf-8").splitlines()
    print("-------SuperComp Test----------")
    print("Executable: " + executable)
    print("Input File: " + input_file)
    print("Numero de Retangulos: " + output[-1])
    print("Tempo de Execucao: " + output[-3])
    print("-------------------------------")
    return [executable, input_file, output[-1], output[-3]]
```

In [7]:

```
# Vamos rodar os testes pesados e colher dados de desempenho
inputs = sorted([n for n in os.listdir("./") if n.startswith('supercomp')])
data = []
for e in executables:
    for i in inputs:
        data.append(run_test(e, i))
```

```
-------SuperComp Test----------
Executable: simulator_O0
Input File: supercomp_test100.txt
Numero de Retangulos: 100
Tempo de Execucao: 1.47774
------------------------------
-------SuperComp Test----------
Executable: simulator_O0
Input File: supercomp_test1000.txt
Numero de Retangulos: 1000
Tempo de Execucao: 174.683
------------------------------
-------SuperComp Test----------
Executable: simulator_O0
Input File: supercomp_test300.txt
Numero de Retangulos: 300
Tempo de Execucao: 13.5415
------------------------------
-------SuperComp Test----------
Executable: simulator_O0
Input File: supercomp_test500.txt
Numero de Retangulos: 500
Tempo de Execucao: 40.3967
------------------------------
-------SuperComp Test----------
Executable: simulator_O1
Input File: supercomp_test100.txt
Numero de Retangulos: 100
Tempo de Execucao: 6.68717
------------------------------
-------SuperComp Test----------
Executable: simulator_O1
Input File: supercomp_test1000.txt
Numero de Retangulos: 1000
Tempo de Execucao: 535.744
------------------------------
-------SuperComp Test----------
Executable: simulator_O1
Input File: supercomp_test300.txt
Numero de Retangulos: 300
Tempo de Execucao: 48.465
------------------------------
-------SuperComp Test----------
Executable: simulator_O1
Input File: supercomp_test500.txt
Numero de Retangulos: 500
Tempo de Execucao: 130.303
------------------------------
-------SuperComp Test----------
Executable: simulator_O2
Input File: supercomp_test100.txt
Numero de Retangulos: 100
Tempo de Execucao: 5.31049
------------------------------
-------SuperComp Test----------
Executable: simulator_O2
Input File: supercomp_test1000.txt
Numero de Retangulos: 1000
Tempo de Execucao: 523.576
------------------------------
-------SuperComp Test----------
```

```
Executable: simulator_O2
Input File: supercomp_test300.txt
Numero de Retangulos: 300
Tempo de Execucao: 47.4278
------------------------------
-------SuperComp Test----------
Executable: simulator_O2
Input File: supercomp_test500.txt
Numero de Retangulos: 500
Tempo de Execucao: 129.909
------------------------------
-------SuperComp Test----------
Executable: simulator_O3
Input File: supercomp_test100.txt
Numero de Retangulos: 100
Tempo de Execucao: 5.58146
------------------------------
-------SuperComp Test----------
Executable: simulator_O3
Input File: supercomp_test1000.txt
Numero de Retangulos: 1000
Tempo de Execucao: 536.727
------------------------------
-------SuperComp Test----------
Executable: simulator_O3
Input File: supercomp_test300.txt
Numero de Retangulos: 300
Tempo de Execucao: 47.6271
------------------------------
-------SuperComp Test----------
Executable: simulator_O3
Input File: supercomp_test500.txt
Numero de Retangulos: 500
Tempo de Execucao: 130.437
------------------------------
-------SuperComp Test----------
Executable: simulator_O3_SIMD
Input File: supercomp_test100.txt
Numero de Retangulos: 100
Tempo de Execucao: 0.975204
------------------------------
-------SuperComp Test----------
Executable: simulator_O3_SIMD
Input File: supercomp_test1000.txt
Numero de Retangulos: 1000
Tempo de Execucao: 140.158
------------------------------
-------SuperComp Test----------
Executable: simulator_O3_SIMD
Input File: supercomp_test300.txt
Numero de Retangulos: 300
Tempo de Execucao: 11.8098
------------------------------
-------SuperComp Test----------
Executable: simulator_O3_SIMD
Input File: supercomp_test500.txt
Numero de Retangulos: 500
Tempo de Execucao: 32.6813
------------------------------
```

In [39]:

```
df = pd.DataFrame(data, dtype=np.float64)
df = df.sort_values([0, 2], ascending=False)
df
```

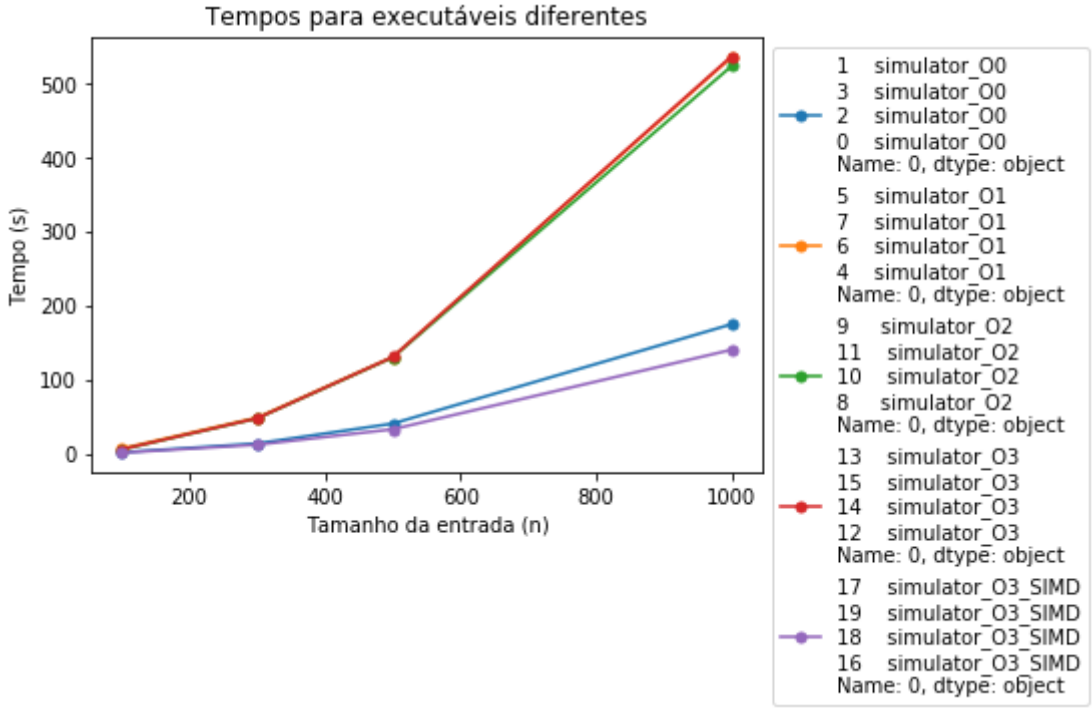
Out[39]:

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 17 | simulator_O3_SIMD | supercomp_test1000.txt | 1000.0 | 140.158000 |
| 19 | simulator_O3_SIMD | supercomp_test500.txt | 500.0 | 32.681300 |
| 18 | simulator_O3_SIMD | supercomp_test300.txt | 300.0 | 11.809800 |
| 16 | simulator_O3_SIMD | supercomp_test100.txt | 100.0 | 0.975204 |
| 13 | simulator_O3 | supercomp_test1000.txt | 1000.0 | 536.727000 |
| 15 | simulator_O3 | supercomp_test500.txt | 500.0 | 130.437000 |
| 14 | simulator_O3 | supercomp_test300.txt | 300.0 | 47.627100 |
| 12 | simulator_O3 | supercomp_test100.txt | 100.0 | 5.581460 |
| 9 | simulator_O2 | supercomp_test1000.txt | 1000.0 | 523.576000 |
| 11 | simulator_O2 | supercomp_test500.txt | 500.0 | 129.909000 |
| 10 | simulator_O2 | supercomp_test300.txt | 300.0 | 47.427800 |
| 8 | simulator_O2 | supercomp_test100.txt | 100.0 | 5.310490 |
| 5 | simulator_O1 | supercomp_test1000.txt | 1000.0 | 535.744000 |
| 7 | simulator_O1 | supercomp_test500.txt | 500.0 | 130.303000 |
| 6 | simulator_O1 | supercomp_test300.txt | 300.0 | 48.465000 |
| 4 | simulator_O1 | supercomp_test100.txt | 100.0 | 6.687170 |
| 1 | simulator_O0 | supercomp_test1000.txt | 1000.0 | 174.683000 |
| 3 | simulator_O0 | supercomp_test500.txt | 500.0 | 40.396700 |
| 2 | simulator_O0 | supercomp_test300.txt | 300.0 | 13.541500 |
| 0 | simulator_O0 | supercomp_test100.txt | 100.0 | 1.477740 |

In [42]:

```
groups = df.groupby(0)

fig, ax = plt.subplots()
for name, group in groups:
    ax.plot(group[2], group[3], marker='o', linestyle='-', ms=5, label=group[0])
plt.title('Tempos para executáveis diferentes')
plt.ylabel('Tempo (s)')
plt.xlabel('Tamanho da entrada (n)')
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
plt.show()
```




No gráfico acima, pode-se verificar que os melhores tempos de todos os testes de desempenho ficaram com as opções O0 e O3 com SIMD (vetorização). O executável com opções de SIMD ser o mais performático é esperado neste caso, porém a opção O0 deveria ser mais lenta, já que não realiza nenhuma otimização no código. Um motivo para esse resultado é que a opção O3 ativa um processamento mais agressivo e em alguns casos isso afeta negativamente o programa. Nem sempre as opções conseguem otimizar de fato seu código (https://stackoverflow.com/questions/34246954/are-there-any-drawbacks-to-using-o3-in-gcc). Por fim, pode-se concluir que o uso de vetorização para cálculos gera uma grande melhoria no desempenho dos programas, sendo ainda mais eficiente se realizada de forma manual.