Projeto 1 de Física Computacional

Pedro Bicudo e Nuno Cardoso , IST,

04 de Setembro de 2017

Resumo

Este projeto deve ser submetido como um único ficheiro comprimido, na página pessoal de aluno no fénix:

- no menu superior devem selecionar estudante
- no menu lateral devem selecionar submeter, projetos;
- deve surgir o grupo desta disciplina de FC, e lá podem submeter projetos e visualizar os projetos submetidos.

Este ficheiro comprimido, por exemplo .zip, deve incluir os vários ficheiros que criados para o projeto, usando as extensões .cpp e .h para os códigos de g++ e .nb para os programas de mathematica, resultados na forma de gráficos em formato .pdf e de ficheiros de dados (ou de texto) .txt. ão devem ser incluídos os ficheiros executáveis.

É importante que enviem uma pequena memória descritiva, em texto simples, chamado readme_g#p#.txt, que inclua apenas

- o número e nome dos alunos do grupo
- o número do grupo
- a lista de ficheiros submetidos, e para cada ficheiro algumas palavras (no máximo meia dúzia) a descreve-lo,
- e finalmente, apenas caso hajam vários ficheiros a linkar em conjunto (por exemplo várias funções e um header), a linha de comandos para os linkar, por exemplo algo do tipo g++ -o g13p1c1.o g13p1c1.cpp g13p1c2.cpp g13p1c3.cpp g13p1c1.h

Para sua própria organização os alunos/grupos devem designar os códigos que forem construindo por g#p1c#.cpp devendo substituir os # por números, sendo o primeiro # o número do seu grupo, o 1 refere-se ao projeto 1 e o segundo # o número do código (ex: g13p1c2.cpp será o segundo código que o grupo 13 realiza para este projeto. Deve usar uma designação semelhante para os outros ficheiros com outras extensões.

O próprio ficheiro zip deve ser denominado g#p1.zip . Isto é válido para o 1o projeto, para o segundo projeto será g#p2.zip, etc.

Serão avaliados,

- o valor dos resultados dos resultados,
- a clareza dos resultados,
- se o código compila,
- a ausência de erros e de fugas de memória nos executáveis,
- a simplicidade e economia de recursos computacionais do código,
- a indentação e os comentários que ajudem ao entendimento do código.

1 Pseudo Gerador de Números Aleatórios (PRNG)

1.a

(1pt) Copie para um editor o seguinte código com erros, que pode chamar g#p1c0.cpp,

```
#include <iostream>
int main(){ int n_max=10;
float r=.7, x=.5; // r eh o parametro geomehtrico, x eh o valor inicial
while( int n < n_max ) { /* ciclo das iteradas */  x *= a;
std::cout << "x(" << ++n << ")= " << x << std::endl; }
return 0; }
```

corrija-o para ele compilar bem, e indente-o e melhore os comentários, mude o ciclo do while para um for, para uma nova versão g#p1c1.cpp.

Lembre-se, o # não é para ficar no nome do seu ficheiro, deve substituí-lo pelo número do seu grupo.

1.b

(2pt) Melhore o input e o output do código, sendo os códigos designados de agora em diante como referirmos anteriormente g#p1c2.cpp, g#p1c3.cpp, etc. Pretendemos que ele receba, a partir do teclado, todos os dados a utilizar (número de iteradas, fator r e variável x inicial); e ainda que escreva os resultados (número de iterada i e valor da sequência x_i) num ficheiro de output.

Pretendemos ainda que, com o *Mathematica* (ou com o *gnuplot*, mas neste caso deve referir isso na memória descritiva), crie um gráfico com os resultados presentes no ficheiro. Note que o gráfico deve sempre incluir nos eixos um label, neste caso i para as abcissas e x_i para as ordenadas. No *Mathematica* pode usar comandos como Import ou ListPlot, e opções como Axeslabel e PlotRange.

1.c

(4pt) Re-aproveite o código de 1.b) para uma nova sucessão. Trata-se de um mapa do intervalo usado para um PRNG simples de programar, com um período elevado. Em particular vamos utilizar um gerador linear congruencial, conforme o "website"de FC, dado pela sequência

$$x_{i+1} = (a\, x_i + c) \mod m$$

onde $m = 2^{32}, a = 1664525, c = 1013904223$. Note que mod m significa o resto da divisão por m. Pense bem se é mais prático trabalhar com variáveis int, ou directamente com variáveis double.

Neste caso pretendemos um plot quadrado, para despistar alguma correlação entre x_i e x_{i+1}, no *Mathematica* pode em usar a opção AspectRatio . Mas os números a guardar no ficheiro devem estar normalizados. ou seja não deve guardar (x_i, x_{i+1}) mas sim deve guardar $(x_i/2^{32}, x_{i+1}/2^{32})$. Faça um plot dos dados no ficheiro utilizando o *Mathemática*, com os labels corretos nos eixos.

2 Calcular o valor de π com um integral de Monte Carlo

Partindo dos códigos do problema anterior vamos calcular o valor de π numericamente com um técnica standard de Monte Carlo. Pretendemos calcular a área quarto de círculo de raio 1 que é $\pi/4$. Para tal vamos gerar pares de pontos aleatórios no intervalo $[0, 1]$.

Vamos aproveitar para trabalhar com ponteiros para vectores, matrizes e funções em c++. .

2.a

(1pt) Crie um novo código, dividindo o código anterior em dois, ficando o pseudo gerador de números aleatórios numa função, e ficando o output, input, ciclos, etc na função main. O mais simples será esta função, por exemplo prng, ter como um argumento um double (que a cada ordem é $x_i \in [0, 1[$, e retornar um double, o x_{i+1}. A função main chama a função geradora de números. Tal como o código anterior, deve ainda escrever a matriz num ficheiro, e produzir um gráfico, que deverão ser do mesmo tipo que os anteriores.

(1pt) O código deve também trabalhar com matrizes na memória dinâmica usando ponteiros. A matriz será uma matriz de dimensão $2 \times n$, devendo em cada linha escrever dois números aleatórios sucessivos, e sendo nt o número total de iteradas. A matriz terá a mesma informação que escrevemos no ficheiro, enquanto o código não a apagar.

(1pt) E ainda, melhore o código para que ele trabalhe com uma função na memória dinâmica usando ponteiros.

2.b

(5pt) Vamos então calcular π.

Para tal deve juntar ao código um if, afim de distinguir o sub-conjunto de pontos de coordenadas (x_i, x_{i+1}) com raio $r = \sqrt{{x_i}^2 + {x_{i+1}}^2} < 1$ do sub-conjunto de pontos com $r > 1$. Defina mais duas variáveis ni e ne, afim de contar o número de pontos interiores e exteriores ao círculo unitário.

Faça um plot para um número de iteradas suficientemente elevado (por exemplo $n = 10000$) , com os pontos interiores e exteriores ao círculo decores diferentes (por exemplo vermelho e verde), afim de visualizarmos esta técnica de Monte Carlo.

Calcule então π, que numericamente é dado pelo valor de $4\frac{ni}{nt}$, sendo que esse valor é uma função do número total de iteradas nt.

Sugestão: *se não tiver tempo para terminar a alínea anterior 2.a) use apenas o início dessa alínea que já inclui a função pseudo geradora de números aleatórios.*

2.c

(5pt) Finalmente vamos estudar a precisão deste método, e a sua convergência em função do número total de iteradas nt.

Para tal, pode incluir um novo ciclo no seu código anterior, que varra o número de iteradas. Pode também optar por outra solução que prefira e que produza os mesmos resultados.

Faça um novo plot, a partir dum ficheiro onde o programa escreve os pares $\left(nt,\, 4\frac{ni}{nt}\right)$, e estude a convergência para π do valor numérico de $4\frac{ni}{nt}$ comparando graficamente o resultado numérico com $pi = 3.14159....$

Comente muito brevemente (uma frase apenas) a convergência deste método de Monte Carlo, no fim do ficheiro de texto da memória descritiva.