

UNIVERSIDADE FEDERAL DO PARÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM GEOFÍSICA

FORTRAN
Prof. Cícero Régis

Questão 1 – Função amostrada em um grid.

O arquivo `funcao.dat` contém valores de uma função discretizada em um grid de pontos do plano (x, z), juntamente com os valores das coordenadas dos pontos amostrados. O arquivo está organizado da seguinte maneira:

- A primeira linha contém dois números inteiros que são as quantidades de coordenadas dos pontos do grid: N_x, N_z .
- A segunda linha contém um zero na primeira coluna (apenas para completar a linha) seguido dos valores das N_x abscissas dos pontos.
- A partir da terceira linha do arquivo, o primeiro valor é a ordenada z dos pontos de cada linha e os demais são os valores da função.

Desta maneira, o arquivo tem a seguinte estrutura:

N_x	N_z					
0.	x_1	x_2	x_3	...		
z_1	$f(x_1, z_1)$	$f(x_2, z_1)$	$f(x_3, z_1)$...		
z_2	$f(x_1, z_2)$	$f(x_2, z_2)$	$f(x_3, z_2)$...		
z_3	$f(x_1, z_3)$	$f(x_2, z_3)$	$f(x_3, z_3)$...		
:	:	:	:	:		

Escreva um programa para ler as informações do arquivo `funcao.dat` e dar como saída, no terminal, qual o maior valor da função e em que coordenadas está este valor. Repita para o menor valor e suas coordenadas.

5	4					
0.	0.20	0.40	0.60	0.80	1.00	
0.20	0.25	0.11	-0.02	-0.18	-0.33	
0.40	0.03	-0.12	-0.28	-0.42	-0.52	
0.60	-0.21	-0.37	-0.49	-0.55	-0.52	
0.80	-0.44	-0.53	-0.54	-0.45	-0.26	

Por exemplo, se o arquivo fosse formado apenas com as linhas e colunas mostradas no quadro abaixo

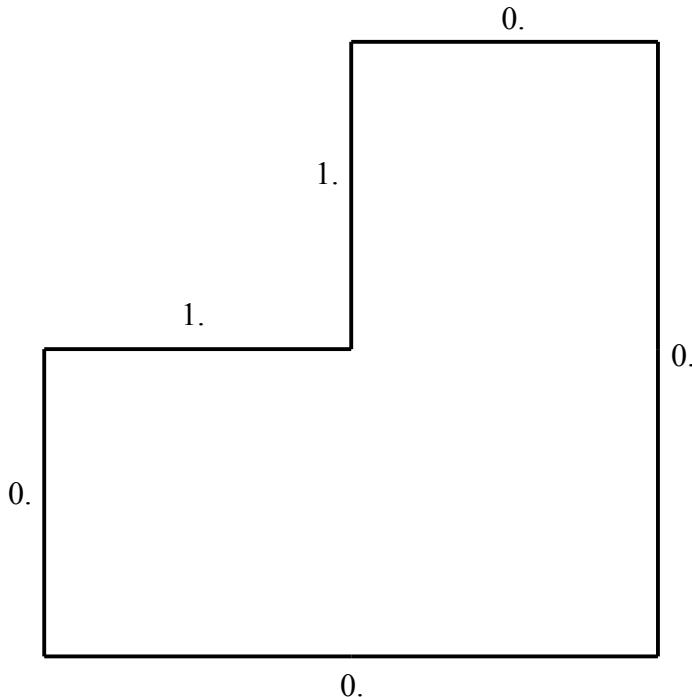
então, a saída do seu programa deveria ser:

O maior valor: 0.25000
Nas coordenadas: $x = 0.20000, z = 0.20000$

O menor valor: -0.55000
Nas coordenadas: $x = 0.80000, z = 0.60000$

Questão 2 – Equação de Laplace.

Modifique seu programa para calcular a solução numérica da equação de Laplace, de modo a calcular os valores da função nos pontos interiores da região ilustrada na figura abaixo. Construa um grid com 100 pontos na direção horizontal e 100 na direção vertical. Os valores nas fronteiras estão indicados na figura. Armazene os valores calculados em uma matriz 100 x 100, preenchendo os pontos no quadrado fora da área da função com valores iguais. Salve esta matriz em um arquivo, carregue-o no MATLAB e construa uma figura com as curvas de isovalores, usando o comando `contour`.



Suponha que a função que você calculou representa o potencial eletrostático ϕ na área da figura. Agora, calcule as componentes do campo elétrico associado a este campo potencial, utilizando a equação

$$\vec{E} = -\nabla \phi$$

Aproxime as derivadas parciais pelas diferenças entre os valores calculados nos pontos do grid na horizontal e depois na vertical:

$$\frac{\partial \phi}{\partial x} \approx (\phi_{i,j} - \phi_{i,j-1}) \quad \frac{\partial \phi}{\partial y} \approx (\phi_{i,j} - \phi_{i-1,j})$$

Construa duas matrizes de dimensões 99 x 99, com as componentes do campo, E_x e E_y , aproximadas, normalizadas pelo módulo do campo em cada ponto do grid:

$$E_x = \frac{Ex}{|\vec{E}|} \quad E_y = \frac{Ey}{|\vec{E}|}$$

Salve estes valores nos arquivos `Ex.dat` e `Ey.dat`. Carregue estes arquivos no MATLAB e use o comando `quiver(Ex,Ey)` para visualizar a direção do campo elétrico na área com vetores unitários.

Questão 3 – Reconstrução de uma imagem.

Uma imagem como esta



pode ser construída a partir da distribuição de intensidades de três cores básicas: vermelho (*red*), verde (*green*) e azul (*blue*). Um programa como o MATLAB, por exemplo, pode ler a informação sobre as cores em três matrizes e compor a imagem. Neste problema você irá trabalhar com estas informações.

No arquivo **fontes.dat** está toda a informação para construir a imagem da foto. Na primeira linha do arquivo estão dois números inteiros (N_x e N_y) que correspondem às quantidades de pixels que formam a imagem, na horizontal e na vertical, nesta ordem. Em seguida vem uma única coluna de dados com $3 \times N_x \times N_y$ valores. Estes valores correspondem aos elementos de três matrizes de tamanhos iguais, com as informações das três cores, respectivamente: *red*, *green*, *blue*. Os valores estão organizados por coluna, de maneira que os primeiros N_y números formam a primeira coluna da matriz Red, os próximos N_y valores formam a segunda coluna da mesma matriz e assim por diante. Na sequência vem as colunas da matriz Green e depois as da Blue. Seu programa deve dimensionar as matrizes a partir da leitura dos valores de N_x e N_y .

Você deve construir um programa em FORTRAN para ler a informação contida no arquivo **fontes.dat** e a partir dela montar três matrizes para serem importadas para o MATLAB e exibir a imagem. Salve as matrizes em três arquivos chamados **Red.dat**, **Green.dat** e **Blue.dat**. Depois, para visualizar a imagem final no MATLAB, simplesmente use o script **monta_imagem.m**.

Questão 4 – Compressão de imagem

No arquivo `Beni.dat` está toda a informação para construir esta imagem:



Na primeira linha do arquivo estão dois números inteiros (N_x e N_y) que correspondem às quantidades de pixels que formam a imagem, na horizontal e na vertical, nesta ordem. Em seguida vem uma única coluna de dados com $3 * N_x * N_y$ valores. Estes valores correspondem aos elementos de três matrizes de tamanhos iguais, com as informações das três cores que irão compor a imagem, respectivamente: vermelho, verde e azul (red, green, blue). Os valores estão organizados por coluna, de maneira que os primeiros N_y números formam a primeira coluna da matriz Red, os próximos N_y valores formam a segunda coluna da mesma matriz e assim por diante. Na sequência vem as colunas da matriz Green e depois as da Blue.

Esta imagem tem 2.359.296 pixels dispostos em 1.152 linhas e 2.048 colunas. O objetivo desta tarefa é reduzir o número de pixels da imagem, de maneira a gerar matrizes menores. Uma maneira simples de criar uma imagem com um quarto do número de pixels (perdendo bastante informação!) é construir matrizes das intensidades de cores de modo que cada pixel seja a média de 4 pixels da imagem original. Por exemplo, se a imagem tivesse apenas 4 por 8 pixels, cada matriz reduzida seria 2 por 4, sendo que cada elemento desta seria a média de 4 elementos da matriz original:

The diagram illustrates a reduction process. On the left, a 4x8 matrix is shown with values ranging from 1 to 4. An arrow points to the right, leading to a smaller 2x4 matrix where each element is the average of a 2x2 block from the original matrix. For example, the top-left element of the reduced matrix is the average of the top-left 2x2 block of the original matrix.

1	1	2	2	3	3	4	4
1	1	2	2	3	3	4	4
2	2	3	3	4	4	2	2
2	2	3	3	4	4	2	2

1	2	3	4
2	3	4	2

Para gerar as matrizes reduzidas, você deve construir um programa em FORTRAN para realizar os seguintes passos:

- 1- Ler a informação contida no arquivo `Beni.dat` e a partir dela montar três matrizes, R, G e B, com o tamanho original;
- 2- Dimensionar outras três matrizes (R2, G2, B2) com metade do número de linhas e metade do

- número de colunas das matrizes originais;
- 3- Calcular o valor de cada elemento das matrizes reduzidas como a média dos valores de 4 elementos adjacentes das matrizes originais;
- 4- Salvar as matrizes em seis arquivos chamados `Red.dat`, `Green.dat`, `Blue.dat`, com os valores originais, e `Red2.dat`, `Green2.dat`, `Blue2.dat`, com os valores das matrizes reduzidas.

Seu programa deve declarar matrizes com atributo `allocatable` e dimensioná-las a partir da leitura dos valores de `Nx` e `. Você pode considerar que os números de linhas e de colunas são sempre pares. Ao final, para visualizar e comparar as imagens no MATLAB, use o script monta_imagens.m.`

Para a avaliação, você deve entregar uma pasta com o código em FORTRAN e os resultados que obteve na forma de imagens salvas do MATLAB. O nome da pasta deve ser seu nome completo.

- Para se divertir em casa:

Se queremos apenas gerar as matrizes reduzidas, sem necessidade de salvar as matrizes originais, este problema pode ser resolvido de forma mais econômica, sem alojar memória para as matrizes `R`, `G` e `B`. Tente construir uma segunda versão de seu programa que seja a mais econômica possível, em termos de memória.

Questão 5 – Crivo de Eratóstenes

Este problema trata de um método para encontrar números primos. Funciona da seguinte maneira: crie um vetor de números inteiros com todos os elementos iniciados com **1**. Elementos desse vetor cujas posições no vetor são números primos permanecerão valendo **1**. Todos os demais elementos serão convertidos para zero.

Iniciando na segunda posição, toda vez que um elemento for encontrado cujo valor seja **1**, percorra o vetor até o final e mude para zero todos os valores cujas posições sejam múltiplos daquela do elemento de valor **1**.

Quando esse processo estiver completo, os elementos do vetor que ainda valem **1** indicam que suas posições são números primos. Então, imprima essas posições.

Teste seu programa para encontrar os números primos até 1000. Otimize ao máximo o programa e tente determinar quantos números primos existem até 1 milhão.

Questão 6 – Ordenamento de vetores

O arquivo coordenadas.dat contém três colunas contendo as coordenadas (x, y) de 500 pontos dispostos de forma aleatória:

	x	y
500		
1	5.9853573	-2.9344454
2	3.4009480	-2.5645757
3	5.7954922	-4.6858921
4	4.4875660	-9.7483358
5	2.2043452	8.0593157
6	2.8838215	2.5492599
7	0.44252551	4.2822309
8	3.21310759E-02	0.32125548
9	2.0824876	8.7191763
10	2.0534627	8.8576050
.	.	.
.	.	.
.	.	.

Escreva um programa para realizar as seguintes tarefas:

- 1- Ler os dados do arquivo **coordenadas.dat**;
- 2- Criar um segundo arquivo contendo as mesmas coordenadas, agora ordenadas de acordo com os valores da coluna das coordenadas x .
- 3- Construa o gráfico da função $y=f(x)$ no MATLAB.

Questão 7 – Passeio do cavalo.

Um dos mais interessantes quebra-cabeças para aficionados do xadrez é o problema chamado de passeio do cavalo. A questão é esta: pode o cavalo se mover em um tabuleiro vazio e tocar cada uma das 64 casas uma única vez? Para analisar este problema, vamos escrever um programa para realizar os movimentos do cavalo no tabuleiro.

O cavalo faz movimentos em forma de L (andando duas casas em uma direção e então uma casa em uma direção perpendicular). Então, de uma casa no meio de um tabuleiro vazio, o cavalo pode fazer oito movimentos diferentes como mostrado na figura:

	1	2	3	4	5	6	7	8
1								
2				3		2		
3			4				1	
4					C			
5			5				8	
6				6		7		
7								
8								

Em seu programa, o tabuleiro é representado por uma matriz 8 por 8 chamada **tab** que deve ser iniciada com zeros. Descreva cada um dos 8 movimentos possíveis em termos de suas componentes horizontal e vertical. Por exemplo, um movimento do tipo 1 como mostrado na figura consiste em mover duas casas horizontalmente para a direita e uma casa verticalmente para cima. O movimento 3 consiste em mover uma casa horizontalmente para a esquerda e duas casas verticalmente para cima. Os oito movimentos podem ser descritos por dois vetores, **h** e **v**, de números inteiros, com os movimentos para a direita e para baixo representados por números positivos e para esquerda e para cima, por números negativos, da seguinte maneira:

$$\begin{array}{ll} h(1) = 2 & v(1) = -1 \\ h(2) = 1 & v(2) = -2 \\ h(3) = -1 & v(3) = -2 \\ h(4) = -2 & v(4) = -1 \\ h(5) = -2 & v(5) = 1 \\ h(6) = -1 & v(6) = 2 \\ h(7) = 1 & v(7) = 2 \\ h(8) = 2 & v(8) = 1 \end{array}$$

Crie duas variáveis, **linha** e **coluna**, para indicar a posição do cavalo em cada momento. Para fazer um movimento do tipo **N**, em que **N** é um número entre 1 e 8, seu programa pode usar os comandos

$$\begin{aligned} \text{linha} &= \text{linha} + v(N) \\ \text{coluna} &= \text{coluna} + h(N) \end{aligned}$$

Iniciando em uma posição (**linha**, **coluna**) escolhida pelo usuário, escolha aleatoriamente os próximos passos do cavalo. Crie um contador que varia de 1 a 64. Registre na matriz **tab** o valor correto do contador em cada casa em que o cavalo chegar. Lembre-se de testar cada movimento em potencial para verificar se o cavalo já passou por aquela casa, e, claro, para se certificar de que ele não vai cair fora do tabuleiro. Escreva e execute seu programa, fazendo o cavalo se deslocar pelo tabuleiro. Inicie o movimento a partir de casas diferentes e veja quantos passos são dados em

cada vez. Você consegue algum passeio completo, ou seja, o cavalo tocar em cada uma das 64 casas?

Após analisar vários passeios do cavalo com seu programa, você pode perceber que as casas nas bordas são mais problemáticas do que as casas próximas ao centro do tabuleiro. De fato, as mais problemáticas, ou de difícil acesso, são os 4 vértices.

Tente, então, mover o cavalo para as casas mais complicadas primeiro e deixar abertas aquelas de acesso mais fácil, para o final, quando o tabuleiro vai ficando mais congestionado.

Escreva uma segunda versão do programa, na qual você vai classificar cada uma das casas de acordo com a facilidade de acesso e mover o cavalo sempre para a casa de menor acessibilidade. Crie uma matriz 8x8 chamada **acesso**, na qual cada número indica o número de casas a partir das quais cada casa pode ser atingida. Em um tabuleiro vazio, a matriz **acesso** terá os valores:

2	3	4	4	4	4	3	2
3	4	6	6	6	6	4	3
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
3	4	6	6	6	6	4	3
2	3	4	4	4	4	3	2

Agora, em seu programa, ao invés de andar aleatoriamente, o cavalo deve sempre se mover para a casa com o menor número de acessibilidade. Em caso de empate, o cavalo pode mover-se para qualquer uma das casas empatadas. Enquanto o cavalo se movimenta, seu programa deve diminuir os números de acessibilidade, conforme mais e mais casas ficam ocupadas. Dessa forma, em qualquer instante, o número de acessibilidade para cada casa vai sempre ser exatamente o número de casas a partir das quais ela pode ser atingida. Execute seu programa 64 vezes, iniciando em cada uma das casas. Quantos passeios completos você obteve?

