Trabalho Final de Métodos Numéricos

Introdução - Objetivos

- Aplicar o conhecimento sobre sistemas de equações adquirido em sala para resolver um problema
- Deslocamento de partículas

Classes

- Vetor
  - Representa um vetor Nx1
- MatrizQuadrada
  - Representa uma matriz NxN
- Fazem parte da biblioteca "AlgebraLinear"

Vetor

- operator()
- operator+
- getTamanho
- toString

Métodos implementados

Método de Gauss-Jacobi

- Em uma iteração k, utiliza apenas os valores de k-1 para calcular as aproximações

$$x_1^{k+1} = \frac{1}{a_{11}}\left[b_1 - a_{12}x_2^k - a_{13}x_3^k - \cdots - a_{1n}x_n^k\right]$$
$$x_2^{k+1} = \frac{1}{a_{22}}\left[b_2 - a_{21}x_1^k - a_{23}x_3^k - \cdots - a_{2n}x_n^k\right]$$
$$x_3^{k+1} = \frac{1}{a_{33}}\left[b_3 - a_{31}x_1^k - a_{32}x_2^k - \cdots - a_{3n}x_n^k\right]$$
$$\vdots$$
$$x_n^{k+1} = \frac{1}{a_{nn}}\left[b_n - a_{n1}x_1^k - a_{n2}x_2^k - \cdots - a_{n,n-1}x_{n-1}^k\right]$$

Método de Gauss-Seidel

- Em uma iteração k, utiliza os valores de k que já foram calculados, e os valores de k-1

$$x_1^{k+1} = \frac{1}{a_{11}}\left[b_1 - a_{12}x_2^k - a_{13}x_3^k - \cdots - a_{1n}x_n^k\right]$$
$$x_2^{k+1} = \frac{1}{a_{22}}\left[b_2 - a_{21}x_1^{k+1} - a_{23}x_3^k - \cdots - a_{2n}x_n^k\right]$$
$$x_3^{k+1} = \frac{1}{a_{33}}\left[b_3 - a_{31}x_1^{k+1} - a_{32}x_2^{k+1} - \cdots - a_{3n}x_n^k\right]$$
$$\vdots$$
$$x_n^{k+1} = \frac{1}{a_{nn}}\left[b_n - a_{n1}x_1^{k+1} - a_{n2}x_2^{k+1} - \cdots - a_{n,n-1}x_{n-1}^{k+1}\right]$$

Inversão da Matriz

- Usando a dica do professor estamos encontrando a inversa resolvendo a equação $\mathrm{X} \cdot Y = Z$ dado X e Z.
- Isto é feito descobrindo cada coluna de Y resolvendo o sistema $X \cdot Y_i = Z_i$ sendo A_i a i-ésima coluna de A
  - Em particular estamos tomando Z como a matriz identidade.

Visualização dos Dados

Apresentação dos resultados

- Como estamos lidando com matrizes, mostrar todos os dados como antes ficava muito feio num .csv.
- Então decidimos serializar as matrizes e vetores usados em dois jeitos.
  - Para uso no Matlab e Octave podemos serializar num arquivo .csv
  - Para uso em outras linguagens como Python e Javascript estamos serializando em JSON

- O arquivo .csv seria uma forma não-proprietária de guardar essas matrizes para uso no Matlab e o no Octave
- Enquanto que o arquivo JSON pode ser lido quase todas as linguagens de programação em uso.

Octave/Matlab

```
> m = csvread(“matriz.csv”);
> sin(m)
ans =
   0.84147   0.90930   0.14112
  -0.75680  -0.95892  -0.27942
   0.65699   0.98936   0.41212
```

Python/NumPy

```
> import json
> from numpy import *
> m = array(json.load(open(“matriz.json”)))
> sin(m)

array([[ 0.84147098,  0.90929743,  0.14112001],
       [-0.7568025 , -0.95892427, -0.2794155 ],
       [ 0.6569866 ,  0.98935825,  0.41211849]])
```

Javascript/NumericJS

```
> var m = JSON.parse("[[1,2,3],[4,5,6],[7,8,9]]");
> numeric.prettyPrint(numeric.sin(m));

“[[     0.8415,     0.9093,     0.1411],
  [    -0.7568,    -0.9589,    -0.2794],
  [      0.657,     0.9894,     0.4121]]”
```

Conclusão

- Considerando os trabalhos anteriores este não teve muitos problemas de programação
- Não houve muita integração com o trabalho anterior, algo que poderia ter sido feito
  - Expansão na definição de função para poder ter variáveis livres na fórmula