Relatório EP3 MAC0300 - Métodos Numéricos de Álgebra Linear

nome: Ygor Tavela Alves

nUSP: 10687642

Introdução

O objetivo deste EP é o de implementar algoritmos para resolver o problema dos mínimos quadrados no $\mathbb{R}^2$, mantendo certos cuidados em relação à estabilidade dos algoritmos e eficiência de acordo com a forma de acesso da memória utilizado pela linguagem C.

Informações Gerais

Ajustes realizados

Para fins de simplificação foram tomados algumas convenções, como:

- A curva do polinômio que resolve a equação é construída de acordo com a base padrão de um polinômio de um dado grau, isto é, foi utilizado a base $\{1, x, x^2, x^3, \ldots, x^m\}$;
- No caso da resolução de sistemas que com rank incompleto dado uma solução permutada $\hat{x}^T = [\hat{x}_1 \ \hat{x}_2]$, pelo fato de $\hat{x}_2$ assumir infinitas soluções, foi tomado a solução trivial para ele, ou seja, $\hat{x}_2$ será nulo;
- É assumido que a precisão de máquina é 1×10^{-6} a partir da constante $EPSILON$ definida em `util.h`. Tal constante é importante para contornar problemas relacionados com as limitações de representação do sistema de ponto flutuante, por exemplo, em casos que é necessário verificar se um dado valor x é nulo, na verdade é feito uma comparação do tipo $|x| < EPSILON$.

Arquivos

Na pasta raiz encontra-se o diretório `src/` que contém:

- *header* e implementação dos algoritmos propostos pelo enunciado do EP (`ep3.c e ep3.h`);
- *header* e implementação de funções utilitárias (`util.c e util.h`);
- *Makefile* para compilação do programa;

Compilação do Programa

Como mencionado acima, na pasta `src/` há um ***Makefile*** que pode ser utilizado para compilação do programa. Abaixo listamos as opções para compilação com `make` :

- `make` : compilação padrão do programa gerando executável **ep3** no diretório;
- `make clean` : apaga objetos e programa.

Execução do Programa

Para executar o programa, após compilação, utilize o comando `./ep3 #numero_operacao` , sendo #numero_operacao dado por:

1. Resolução do problema dos mínimos quadrados para sistemas que possuem rank completo;

2. Resolução do problema dos mínimos quadrados para sistemas que possuem rank incompleto;

Entrada do programa

Para ambos os casos a entrada esperada será, o grau do polinômio **m** para o qual se deseja obter uma curva que minimiza os resíduos, a quantidade total de pontos **n**, seguida por **n** coordenadas de pontos *(x, y)*, um exemplo de entrada válida seria:

```
4
5
1 1.1
1.5 1.2
2 1.3
2.5 1.3
3 1.4
```

Será uma entrada que irá buscar um polinômio de grau $m = 4$, utilizando um total de $n = 5$ pontos cartesianos dados por *(1; 1.1), (1.5; 1.2), (2; 1.3), (2.5; 1.3), (3; 1.4)*.

Saída do programa

A saída esperada do programa será a curva de um polinômio que minimiza o quadrados dos residuos dos pontos em relação a tal curva. Para a entrada do programa apresentada acima, temos uma saída do tipo:

```
The polynomial - using the polynomial standard basis - that solves the leas
t squares problem is:
p(t) = 2.800000 + -4.516667t**1 + 4.150000t**2 + -1.533333t**3 + 0.200000t*
*4
```

Discussão

Apesar do enunciado sugerir a implementação de vários algoritmos que resolvem o problema dos mínimos quadrados, neste EP foi optado por implementar apenar o algoritmo de decomposição QR utilizando refletores de Householder para os casos em que um dado sistema A tenha posto completo ou incompleto. Tal fato se deve pelo motivo dele possuir um desempenho e/ou estabilidade numérica melhor se comparado as outras alternativas. Abaixo é listado algumas das principais diferenças entre as implementações sugeridas pelo capítulo 3 do livro:

- Algoritmo utilizando rotações de Givens: Em comparação com o algoritmo que utiliza refletores de Householder, não há muita distinção no que diz respeito à estabilidade numérica ou tempo de execução. No entanto, o algoritmo que utiliza rotações de Givens possui uma grande complexidade de implementação, o que pode implicar num *overhead* maior para entendimento e desenvolvimento do mesmo.
- Algoritmo de Gram-Schmidt clássico: O processo é instável, pequenos erros de arredondamentos podem levar à computação de vetores longe de serem ortogais. Além disso, durante o processo de ortogonalização de A (processo análogo ao de decomposição QR) ele realiza uma quantidade de flops na ordem de $\mathcal{O}(2nm^2)$, enquanto que, o algoritmo utilizando refletores realiza uma quantidade total de flops na ordem de $\mathcal{O}(nm^2 - \frac{2}{3}m^3)$, tal diferença pode fazer com que o algoritmo utilizando refletores de Householder seja mais performático para casos onde a diferença entre n e m não é muito grande.
- Algoritmo de Gram-Schmidt adaptado: Diferentemente do seu algoritmo clássico, ele possui uma estabilidade numérica boa. No entanto, ainda aprensenta a mesma diferença assintótica em relação à quantidade de flops realizados para ortogonalizar a matriz A.

Em relação à implementação em si do algoritmo, se pode destacar alguns pontos interessantes como:

- Para diminuir o *overhead* com acesso à memória, a matriz A foi interpretada como sua transposta e as operações que agem sobre vetores e matrizes foram pensadas de forma que haja prioridade para o percorrimento de linhas devido a forma de representação dos mesmos na linguagem C. Desta forma, em alguns casos como o da implementação para posto incompleto que necessita realizar trocas entre as colunas da matriz de A, na verdade são realizadas trocas de linhas da matriz transposta de A.
- Para manter a estabilidade numérica dos algoritmos, foram adotadas algumas medidas com o intuito de diminuir erros de arrendodamento ou overflow. Pode-se citar medidas como rescalonamento durante a computação de refletores no caso de rank completo, rescalonamento do sistema $Ax = b$ dividindo tanto A quanto b pelo maior elemento da matriz A no caso de rank incompleto, etc. Apesar de tais mudanças implicarem num leve aumento do tempo de execução o ganho com a estabilidade numérica é um fator que prevaleceu sobre a decisão, se comparado com o tempo de execução total a diferença será da ordem de $\mathcal{O}(nm)$, o que é infímo se comparado com o tempo total de execução da ordem de $\mathcal{O}(nm^2)$.
- Para fins de otimização, no caso do algoritmo para posto incompleto as normas das colunas (lembrando que na realidade são feitas operações sobre as linhas da matriz transposta) são cacheadas num vetor e atualizadas conforme as iterações do algoritmo. Tal mudança levou a uma mudança em relação a quantidade total de flops realizados, sem o cache a quantidade de flops seria da ordem de $\mathcal{O}(nm^2 - \frac{1}{3}m^3)$, enquanto que, com o cache o custo passou a ser da ordem de $\mathcal{O}(2nm)$.