EP1 - Análise numérica

André Camargo Perello - 9912403

September 2017

Contents

1 Apresentação

Relatório realizado para a disciplina MAP2220 - Fundamentos da análise numérica, ministrada pelo professor Luis Carlos dos Santos.

2 Introdução

Para organizar o trabalho desenvolvido neste exercício, foram criadas três fases. São elas:

1. Escolha dos arquivos e estudo dos dados.
2. Criação da spline e comparação com as imagens fornecidas.
3. Calculo da espessura máxima e de sua posição.

Primeiro iremos discutir sobre como foi feita a compilação do programa. Depois, iremos apresentar os scripts e funções que foram utilizadas no trabalho. Finalmente, apresentamos as fases expostas acima.

3 Informações de compilação

Para o desenvolvimento do EP, foi utilizado o compilador g++ com a versão C++11. Para compilar o EP, basta entrar na pasta "numerical", executar o comando "make" e depois disso executar "./numerical" mais o nome do tipo de asa que deseja utilizar. O programa foi testado em três diferentes sistemas operacionais (windows, linux, macOS), entretanto foi completamente desenvolvido no macOS Sierra.

4 Scripts

Ambos os scripts utilizados no projeto foram desenvolvidos por mim. Eles serviram como apoio para o programa em c++. Sao eles:

4.1 plot.R

O script foi o mecanismo utilizado para plotar todos os gráficos que serao utilizados neste relatorio. Ele recebe como entrada a localização de um arquivo no formato .csv (Comma Separated Values) e o nome da imagem de saída desejada. A chamada do script atraves do programa em c++ é feita através de uma system call, na forma:

```
p << string("Rscript ./../scripts/plot.R ") << string(path) << " " << name;
system(p.str().c_str());
```

4.2 readInput.py

O script recebe como entrada o nome de um arquivo no formato explicado acima e cria uma pasta em /arrays com o nome do arquivo lido e dois arquivos de texto, os pontos da parte inferior ($y < 0$) guardados em low.txt e os da parte superior ($y > 0$) em up.txt. Este script deve ser invocado antes da main, uma vez que gera os arquivos necessários para a execução da mesma.

5 Classes

5.1 CoordList

auxilia na manipulação dos dados, servindo como uma lista para guardar as coordenadas. Ela possui os seguintes métodos:

5.1.1 Construtor

Inicializa os vetores onde guardaremos nossas coordenadas.

5.1.2 addCoord

Adiciona um ponto no vetor de coordenadas

5.1.3 makePairs

Ordena o vetor de coordenadas para que fique em ordem crescente em relação à coordenada x.

5.2 SplineDiff

A classe serve para a segunda parte do programa. Apos serem criadas as splines, a classe SplineDiff cria a função altura. Ela possui os seguintes métodos:

5.2.1 Construtor

Recebe 3 vetores de tipo double (pontos no eixo X, pontos na spline superior e inferior) e cria um quarto vetor que guarda a altura (diferença entre os pontos superiores e inferiores). O chamador precisa se assegurar de que haja dois Y's associados a cada X.

5.2.2 getDerivate

Inicialmente, criamos um vetor de tamanho $n = 10^6$ e $h = 10^{-6}$ espaçados uniformemente, onde:

$$x_i = x_{i-1} + h$$

Depois, criamos uma spline baseada na função $e(x)$. Utilizando este spline e os pontos gerados acima, calculamos a derivada da função altura. Para isso, foi escolhido o algoritmo apresentado no livro denominado 3 point midpoint formula. O algoritmo foi escolhido pois como tratamos de n pontos, os pontos das extremidades são irrelevantes, uma vez que a diferença é degenerada perto do um e muito pequena perto do zero. Portanto, tiramos a derivada apenas nos n-2 pontos centrais. O algoritmo implementa a seguinte formula:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

Além disso, ela encontra o ponto onde a derivada troca de sinal e o imprime na saída padrão.

Finalmente, plota a derivada chamando a função plot.

5.2.3 plotHeight

O método apenas plota a função altura chamando a função plot.

6 Funções

6.1 genPoints

A função gera cem pontos distribuidos no intervalo $[0, 1]$, seguindo a distribuição **D**. Para isso, geramos 100 pontos distribuídos uniformemente no intervalo $[0, \pi]$ e aplicamos à distribuição **D**. A função retorna estes 100 pontos.

6.2 plot

A função plot é a interface do programa com o script plot.R, gerando os argumentos necessários para a execução de tal. A função recebe como entrada:

- Dois vetores (X, Y) que serão os pontos a serem plotados.
- O tamanho dos vetores.
- O nome dos arquivos de dados iniciais.
- O local para salvar os plots gerados.

A função plotSpline é análoga, apenas recebendo dois vetores Y, ao invés de um.

7 Bibliotecas

7.1 Spline.h

Para a criançāo da spline, foi utilizada uma biblioteca de código livre desenvolvida pelo usuário do github ttk592. A biblioteca tem uma interface muito simples. Ressaltaremos os métodos utilizados na construção da spline:

7.1.1 set_points

Recebe dois vetores, X e Y, ordenados e cria um polinômio de grau três totalmente diferenciável.

7.1.2 Operador ()

O operador () é a interface do usuário com a spline criada. Um exemplo seria: A chamada spline(x) retornaria o valor da função no ponto x.

Mais informações da biblioteca podem ser encontradas **aqui**.

8 Terminologia e constantes

8.1 Distribuição D

D se refere a distribuição fornecida no enunciado:

$$x = \frac{1}{2}(1 - cos\theta), \forall\theta \in [0, \pi]$$

8.2 plotSize

plotSize é o numero de pontos utilizados para plotar a imagem de maneira "contínua". No caso, 10^5.

8.2.1 Função espessura / altura

A função altura, definida como a diferença entre as imagens dos pontos da distribuição é:

$$e(x) = upSpline(x) - downSpline(x), \forall x \in D$$

Ambos os splines serão explicados na Fase I

8.2.2 Paço

A diferença h entre os pontos no eixo x. Ou seja:

$$h = x_i - x_{i-1}$$

9 Fase I

9.1 Escolha dos arquivos

Como determinado na proposta, foram utilizadas as iniciais do meu nome **A**ndré **C**amargo **P**erello. Portanto: **a,c,p**. Alem disso, foi imposta a restrição do arquivo explicitar a quantidade de pontos e quais fazem parte do spline superior e inferior. Dessa forma, foram escolhidos os seguintes arquivos:

- arad20
- clarky
- p51hroot

9.2 Estudo dos dados

Inicialmente, podemos observar que todos os arquivos possuem formas diferentes. A quantidade de pontos em cada um também não é uniforme, uma vez que arad20 possui 52 pontos, clarky possui 122 e p51hroot possui 98. Para a "limpeza" dos dados foi utilizado o script readInput.py. Os arquivos gerados nesta fase se encontram na pasta arrays. Já os arquivos originais se encontram na pasta files.

10 Fase II

10.1 Criação dos splines inicias

A fim de obter uma precisão melhor na criação das splines, foram criados um spline para os pontos de cima e uma para os pontos de baixo. no código, são chamados de **UpSpline** e **DownSpline**. A criação é feita através da biblioteca spline.h, onde fornecemos os pontos obtidos na fase 1.

10.2 Geração dos pontos

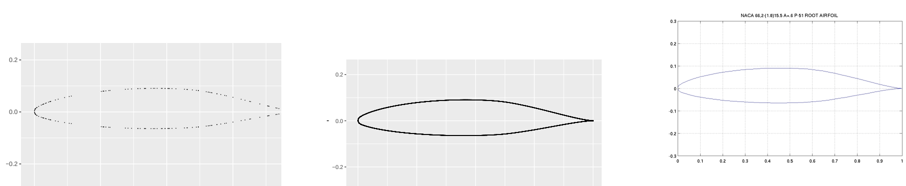
Conforme pedido no enunciado, geramos 100 pontos seguindo a distribuição **D**. Estes pontos terão suas imagens calculadas em relação ao spline construído na secção anterior.

10.3 Plot e resultados

Apos a criação dos splines, plotamos dois graficos. o primeiro se refere aos 100 pontos gerados em genPoints. Já no segundo gráfico, geramos **plotSize** pontos, a fim de mostrar os splines criados como funções continuas. Infelizmente não foi possível padronizar o tamanho e posições das imagens, uma vez que todas possuem formatos e resoluções diferentes. Dessa forma, adotei os tamanhos para preservar a resolução de cada uma. Seguem abaixo os plots:

10.3.1 p51hroot

Como podemos ver abaixo, os plots gerados (a), (b) possuem o formato do gráfico inicial, comprovando que a interpolação foi bem sucedida. Como esperado, os pontos na figura (a) estão mais concentrados nas extremidades do intervalo $[0, 1]$.

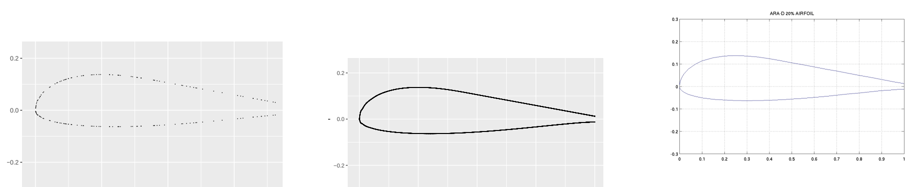


(a) (b) (c)

Figure 1: Plots da p51hroot. (a) plot da distribuição pedida. (b) plot das splines. (c) gráfico original

10.3.2 arad20

Como podemos ver abaixo, os plots gerados (a), (b) possuem o formato do gráfico inicial, comprovando que a interpolação foi bem sucedida. Como esperado, os pontos na figura (a) estão mais concentrados nas extremidades do intervalo $[0, 1]$.

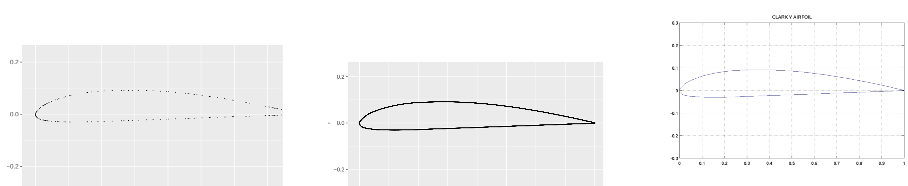


(a) (b) (c)

Figure 2: Plots da arad20. (a) plot da distribuição pedida. (b) plot das splines. (c) gráfico original

10.3.3 clarky

Como podemos ver abaixo, os plots gerados (a), (b) possuem o formato do grafico inicial, comprovando que a interpolação foi bem sucedida. Como esperado, os pontos na figura (a) estão mais concentrados nas extremidades do intervalo $[0, 1]$.



(a) (b) (c)

Figure 3: Plots da clarky. (a) plot da distribuição pedida. (b) plot das splines. (c) gráfico original

11 Fase III

11.1 Métodos

A fim de encontrar a espessura máxima, obtemos a diferença de altura em todos os pontos da distribuição gerada. Para encontrarmos a diferença máxima, foram testados diferentes métodos que serão discutidos abaixo. No fim, decidi interpolar a função altura através de um spline e encontrar a derivada máxima deste. Abaixo, estão todos os métodos testados, assim como o porque de suas falhas/sucessos:

11.1.1 Forca bruta

Inicialmente foi testado encontrar o máximo através da forca bruta, ou seja, iterar os 100 pontos gerados previamente e encontrar a maior espessura ($max\{e(x)\}, \forall x \in [0, 1]$). O método apresentou diversos problemas. Como os pontos são concentrados nas extremidades e o máximo se encontra perto do centro, os pontos a serem avaliados no centro são baixos, diminuindo a precisão do método. Dessa forma, o método foi rejeitado. Entretanto, o método não foi em vão, uma vez que me fez realizar que seria muito difícil encontrar uma resposta precisa devido a distribuição fornecida.

11.1.2 Derivada da função altura

Após isto, foi testada apenas encontrar x_0 tal que:

$$e'(x_0) = 0$$

Isto apresentou alguns problemas. Primeiro, devido a característica da distribuição dada, não foi possível encontrar h tal que:

$$h = x_i - x_{i-1}, \forall i \in \{0, 1, \cdots, n\}$$

Desta forma, os algoritmos usuais de calculo de derivadas foram inviabilizados. Portanto, o método foi rejeitado. Mesmo rejeitado, formou a base para o método final, uma vez que seu maior problema foi o h variável.

11.1.3 Derivada da spline da função altura

Este é o método implementado no EP. Visto todos os problemas gerados pelos métodos anteriores, busquei algo que iria elimina-los. Para isso, foi criada uma spline, **sHeight**, baseada na função altura, $e(x)$. Depois, são gerados pontos espaçados igualmente e calculadas suas imagens no spline **sHeight**, nos permitindo encontrar a derivada de **sHeight** numericamente. Dessa forma, eliminamos o problema do h variável, assim como podemos calcular a derivada em mais de cem pontos, aumentando nossa precisão. Entretanto, encontrar o zero numericamente sem um algoritmo se torna um problema. Como zero de funções ainda não foi tratado em aula, preferi utilizar um método mais simples.Assim, guardamos o ponto em que a derivada troca de sinal, sinalizando que ultrapassamos o máximo. Este será nossa posição de espessura máxima.

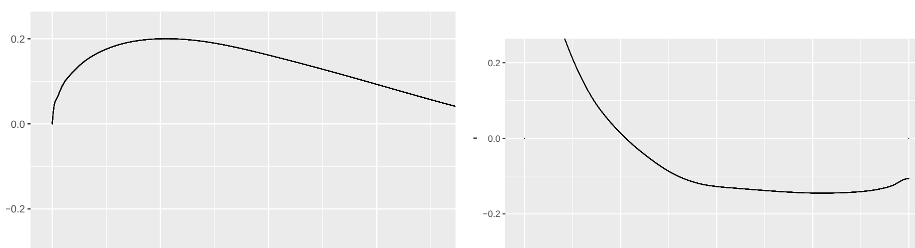
11.2 Plot e resultados

11.2.1 arad20

Temos que a posição de espessura máxima foi de 0.26344 e que a espessura máxima foi de 0.200178. Portanto, o ponto de máximo da função altura é:

$$(x, y) = (0.26344, 0.200178)$$

Podemos observar na figura abaixo que a função altura alcança o seu máximo no intervalo $[0.25, 0.30]$, corroborando com o ponto de espessura máxima obtido.



(a) (b)

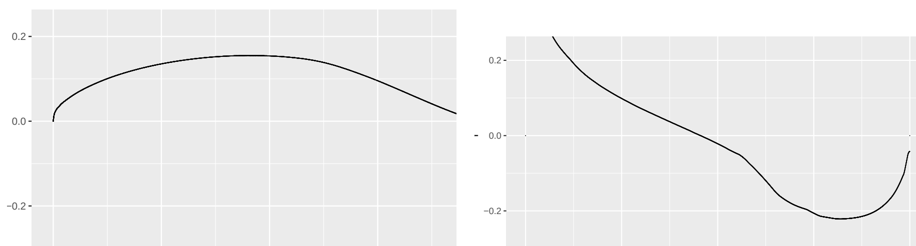
Figure 4: (a) plot da função altura. (b) plot da derivada da função altura.

11.2.2 p51hroot

Temos que a posição de espessura máxima foi de 0.45542 e que a espessura máxima foi de 0.155002. Portanto, o ponto de máximo da função altura é:

$$(x, y) = (0.45542, 0.155002)$$

Podemos observar na figura abaixo que a função altura alcança o seu máximo no intervalo $[0.40, 0.50]$, corroborando com o ponto de espessura máxima obtido.



(a) (b)

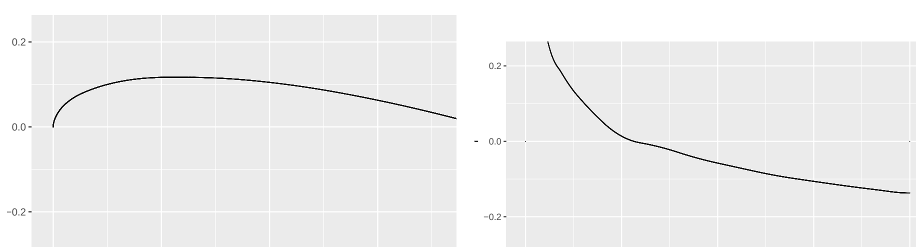
Figure 5: (a) plot da função altura. (b) plot da derivada da função altura.

11.2.3 clarky

Temos que a posição de espessura máxima foi de 0.28234 e que a espessura máxima foi de 0.117073. Portanto, o ponto de máximo da função altura é:

$$(x, y) = (0.28234, 0.117073)$$

Podemos observar na figura abaixo que a função altura alcança o seu máximo no intervalo $[0.25, 0.30]$, corroborando com o ponto de espessura maxima obtido.



(a) (b)

Figure 6: (a) plot da função altura. (b) plot da derivada da função altura.

12 Discussão

O programa implementa udo que foi requisitado no enunciado. Como observado nas secções anteriores, os resultados obtidos não possuem nenhum conflito com os gráficos das funções. O fato de não ser utilizado um algoritmo para encontrar o zero da função derivada pode ser um motivo para uma certa imprecisão nos resultados (muito pequena). Para balancear isto, procurei gerar o maior numero de pontos possíveis para o calculo da derivada, encontrando 10^6, uma vez que qualquer quantidade maior de pontos desestabilizaria o método de calcular a derivada (h ficaria muito pequeno).