MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA CARTOGRÁFICA

1 TEN KESSY JHONNES MATHEUS MARQUES MAGALHÃES
1 TEN ROMÁRIO FREIRE DA SILVA

REQUISITOS PARA A CONSTRUÇÃO DE UMA PLATAFORMA DE AJUSTAMENTO E INTEGRAÇÃO COM A PLATAFORMA DE SÉRIES TEMPORAIS

RIO DE JANEIRO
2020

1 TEN KESSY JHONNES MATHEUS MARQUES MAGALHÃES
1 TEN ROMÁRIO FREIRE DA SILVA

REQUISITOS PARA A CONSTRUÇÃO DE UMA PLATAFORMA DE AJUSTAMENTO E INTEGRAÇÃO COM A PLATAFORMA DE SÉRIES TEMPORAIS

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia Cartográfica do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia Cartográfica.

Orientador(es): TC Francisco Roberto da Rocha Gomes, Professor do IME

Rio de Janeiro

2020

©2020
INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia (IME), que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

Marques Magalhães, 1 Ten Kessy Jhonnes Matheus; Freire da Silva, 1 Ten Romário.

Requisitos para a Construção de uma Plataforma de Ajustamento e Integração com a Plataforma de Séries Temporais / 1 Ten Kessy Jhonnes Matheus Marques Magalhães e 1 Ten Romário Freire da Silva. – Rio de Janeiro, 2020.

47 f.

Orientador(es): TC Francisco Roberto da Rocha Gomes.

Projeto de Final de Curso (graduação) – Instituto Militar de Engenharia, Engenharia Cartográfica, 2020.

1. Direcionamento. 2. Construção. 3. Plataforma Online. 4. Problemas de Ajustamentos. 5. Integração. 6. Séries Temporais. 7. Procedimentos e Ferramentas. 8. Python. i. da Rocha Gomes, TC Francisco Roberto (orient.) ii. Título

1 TEN KESSY JHONNES MATHEUS MARQUES MAGALHÃES
1 TEN ROMÁRIO FREIRE DA SILVA

Requisitos para a Construção de uma Plataforma de Ajustamento e Integração com a Plataforma de Séries Temporais

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia Cartográfica do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia Cartográfica.

Orientador(es): TC Francisco Roberto da Rocha Gomes.

Aprovado em Rio de Janeiro, de de 2020, pela seguinte banca examinadora:

TC. Francisco **Roberto** da Rocha **Gomes** - Professor do IME - Presidente

---

Maj. **Stefano** Sampaio Suraci - Professor do IME

TC. Alexandre Dantas Soares **Coutinho** - Professor do IME

Rio de Janeiro
2020

"Sê humilde para evitar o orgulho, mas voa alto para alcançar a sabedoria."

(Santo Agostinho)

RESUMO

O presente trabalho tem por fim gerar uma análise de requisitos para a construção de uma plataforma online, gratuita e código aberto para soluções de problemas de ajustamentos e integração desta com uma outra plataforma já existente de Séries Temporais. Esse relatório apresenta as etapas do desenvolvimento do trabalho visando relatar todos os procedimentos e ferramentas utilizadas.

Palavras-chave: Direcionamento. Construção. Plataforma Online. Problemas de Ajustamentos. Integração. Séries Temporais. Procedimentos e Ferramentas. Python.

ABSTRACT

The purpose of this work is to generate a direction for the construction of an online, free and open source platform for solving adjustment problems and its integration with another existing Time Series platform. This report presents the stages in the development of the work in order to report all the procedures and tools used.

Keywords: Direction. Construction. Online Platform. Adjustment Problems. Integration. Time Series. Procedures and Tools. Python.

LISTA DE ILUSTRAÇÕES

Figura 2.1 – Matriz $\sum$ da Decomposição SVD. (ÅKE BJÖRCK, 1996) 17
Figura 2.2 – Decomposição QR.(ÅKE BJÖRCK, 1996) 18
Figura 3.1 – Fluxograma de atividades. 24
Figura 3.2 – Códigos para inserção de botões. 26
Figura 3.3 – Estrutura para inserção de novos algoritmos. (MONTEIRO e MARCHIORE, 2018) 28
Figura A.1 – Configuração de instalação Anaconda. 33
Figura A.2 – Configuração de variáveis de ambiente. 34
Figura A.3 – Integração com o terminal de comando. 34
Figura A.4 – Configurações do pacote de integração com o terminal. 35
Figura A.5 – Integração com o Python. 35
Figura A.6 – Bibliotecas Python. 36
Figura A.7 – Construção do ambiente virtual. 36
Figura A.8 – Instalação de bibliotecas. 36
Figura A.9 – Conclusão da instalação das bibliotecas. 36
Figura B.1 – Janela para a criação de novo aplicativo. 37
Figura B.2 – Criação de novo aplicativo. 38
Figura B.3 – Plataforma Heroku. 38
Figura B.4 – Instalação Heroku CLI. 39
Figura B.5 – Instalação Git SCM. 40
Figura B.6 – Conexão entre o terminal e a plataforma Heroku. 41
Figura A.1 – Exemplo presente na plataforma. 42

LISTA DE TABELAS

Tabela 2.1 – Comparação Entre as Normas Vetoriais (SURACI, 2018) 20

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
GPS	Sistema de Posicionamento Global
MMQ	Método dos Mínimos Quadrados
ML1	Minimização da norma L1
ML∞	Minimização da norma ML_∞
SVD	Decomposição de Valor Singular
IME	Instituto Militar de Engenharia
PFC	Projeto de Final de Curso

LISTA DE SÍMBOLOS

$\Re$	Conjunto dos Números Reais
$\in$	Pertence
$\sum$	Somatório
A^T	Matriz Transposta de A
$A^{m \times n}$	Matriz com m linhas e n colunas
λ	Valor próprio (autovalor)
σ	Letra grega minúscula Sigma
α	Letra grega minúscula Alfa
β	Letra grega minúscula Beta
I	Matriz Identidade

SUMÁRIO

1 INTRODUÇÃO . . . 13
1.1 MOTIVAÇÃO . . . 13
1.2 OBJETIVO . . . 13
1.3 JUSTIFICATIVA . . . 14

2 FUNDAMENTAÇÃO TEÓRICA . . . 15
2.1 CARACTERÍSTICAS DO AJUSTAMENTO . . . 15
2.2 RESOLUÇÕES PARA AJUSTAMENTOS . . . 15
2.2.1 MÉTODO DOS MÍNIMOS QUADRADOS (MMQ) PONDERADOS . . . 16
2.2.1.1 DECOMPOSIÇÃO DE VALOR SINGULAR (SVD) . . . 16
2.2.1.2 DECOMPOSIÇÃO QR . . . 18
2.2.2 MINIMIZAÇÃO DA NORMA L1 (ML1) . . . 18
2.2.3 MINIMIZAÇÃO DA NORMA L_∞ (ML∞) . . . 19
2.3 COMPARAÇÃO ENTRE MMQ, ML1 E ML∞ . . . 20
2.4 FERRAMENTAS . . . 21
2.4.1 GITHUB . . . 21
2.4.2 PYTHON ANYWHERE . . . 21
2.4.3 LINGUAGENS DE PROGRAMAÇÃO . . . 21
2.4.4 BIBLIOTECAS DO PYTHON . . . 22
2.5 REQUISITOS DO PROJETO . . . 23
2.5.1 ESTRUTURA DA PLATAFORMA . . . 23
2.5.2 ALGORITMOS . . . 23

3 METODOLOGIA . . . 24
3.1 DESENVOLVIMENTO DO PROJETO . . . 24
3.2 EMPREGO DOS ALGORITMOS . . . 25

4 CONCLUSÃO . . . 29
4.1 SUGESTÃO PARA TRABALHOS FUTUROS . . . 30

REFERÊNCIAS . . . 31

APÊNDICE A – SETUP PARA DESENVOLVIMENTO . . . 33

APÊNDICE B – DEPLOYMENT NA PLATAFORMA HEROKU . . . 37

ANEXO A – EXEMPLOS PRESENTES NA PLATAFORMA . . . 42

A.0.1 MODELO MMQ 42
A.0.2 CÓDIGO EXEMPLO MMQ 43
A.0.3 MODELO ML1 43
A.0.4 MODELO ML∞ 43

ANEXO B – CÓDIGO EXEMPLO ML1 44

ANEXO C – CÓDIGO EXEMPLO ML∞ 46

1 INTRODUÇÃO

1.1 Motivação

No âmbito do aperfeiçoamento da engenharia, o desenvolvimento tecnológico tem se mostrado característica fundamental para o desenvolvimento científico desta área. A criação de ferramentas digitais como aplicativos, softwares, sites e outros, têm sido de extrema relevância para esse desenvolvimento. Isso se deve principalmente ao fato de que após realizado um determinado trabalho, o mesmo não precisará ser replicado, podendo assim desenvolver um projeto a partir do que já foi finalizado ou utilizar-lo para a solução de um problema como uma ferramenta digital.

A Engenharia Cartográfica, nesse ponto, também se desenvolveu devido a sincronização de novas tecnologias com o uso de dados do Sistema de Posicionamento Global (GPS) e imagens de satélite, por exemplo. Tendo vista o atual cenário globalizado, o aprimoramento cartográfico no cenário digital se faz uma necessidade.

Atualmente existem diversos programas e softwares que solucionam o problemas atuais relacionados ao ajustamento de dados. No entanto, muitos deles não apresentam uma solução online e gratuita para a realização dessas análises.

Quando o assunto é a criação de uma determinada plataforma online, não se encontra referências concretas e confiáveis que auxiliem de forma simples e direta a maneira correta de se realizar tal criação. Com isso, essa é uma área que ainda visa amadurecimento, mesmo frente a todo desenvolvimento tecnológico, e enriquecimento de fontes para auxílio de trabalhos futuros.

O ensino de Engenharia Cartográfica muitas vezes utiliza-se de diversos softwares dos mais diferentes fins. Frente a isso, uma plataforma que unifique possíveis soluções online para resolução de problemas enfrentados no ensino de tal engenharia seria uma saída para concentrar-se em um só local algumas das ferramentas mais importantes e recorrentes, tanto para o ensino quanto para o desenvolvimento de projetos cartográficos.

1.2 Objetivo

Com base nas motivações apresentadas, o presente trabalho objetiva não só a realizar o levantamento de requisitos para a construção de uma plataforma para solução de ajustamento, como também verificar as ferramentas necessárias para integrar tal plataforma com a solução *web* realizada em um Projeto de Final de Curso (PFC) de 2018 [1] do curso de Engenharia Cartográfica do Instituto Militar de Engenharia (IME), sendo esta

última também uma solução online, porém para a realização de análises de séries de dados temporais.

Concomitantemente, obtem-se ferramentas necessárias para a contrução de uma plataforma online facilmente aplicada, *Open-source* visando a introdução de algorítmos e também e contribuições de projetos futuros, além de ser totalmente gratuita, podendo contribuir com a formação de diversas turmas de engenheiros cartógrafos.

1.3 Justificativa

Tal projeto nasceu da necessidade de um direcionamento para a construção de uma determinada plataforma online e integração da mesma com uma outra, além da falta de refências concretas que auxiliem de forma simples tal criação.

Os métodos de ajustamento foram escolhidos frente à incidência com que os mesmos são deparados durante a graduação em engenharia cartográfica, visando auxiliar nos possíveis problemas enfrentados pelos alunos e contribuir para a formação de futuros engenheiros cartógrafos.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Características do Ajustamento

Os problemas lineares de mínimos quadrados são problemas computacionais de vital importância para o ajustamento de modelos matemáticos de determinadas observações. Para reduzir a influência de erros nas observações, busca-se adquirir um número maior de informações do que o respectivo número de incógnitas em um determinado modelo. Resumindo, visa-se resolver um sistema superdeterminado de equações lineares. Em termos matriciais, dado um vetor $b \in \Re^m$ e uma matriz $A \in \Re^{m \times n}$, com $m > n$, desejamos encontrar um vetor $x \in \Re^n$ em que Ax é a melhor aproximação para b.

Em outras palavras, deseja-se obter o $min||Ax - b||_2$, $A \in \Re^{m \times n}$, $b \in \Re^m$ onde $||.||$ detona a norma euclideana de um vetor. Chamamos isso de um problema linear de míimos quadrados sendo x uma solução para o sistema $Ax = b$, sendo $r = b - Ax$ o vetor resíduo das observações. Caso $m < n$, o problema não terá solução única. Por outro lado, ao fazermos $m > n$, garantimos que com este método obtenha uma solução única para o ajustamento [2], contanto que a matriz dos coeficientes das incógnitas possuir posto completo [3].

2.2 Resoluções para Ajustamentos

Para realizar o ajustamento de determinadas observações, podemos seguir diversos métodos dependendo da problemática e da abordagem que se deve dar ao mesmo. Inicialmente, para a plataforma, focaremos nos seguintes métodos que serão aprofundados mais a frente:

a) Método dos Mínimos Quadrados (MMQ);

   - Decomposição de Valor Singular (SVD);

   - Decomposição QR;

b) Minimização da norma L1 (ML1); e

c) Minimização da noma L∞ (ML_∞).

2.2.1 Método dos Mínimos Quadrados (MMQ) Ponderados

Tal método dá como solução para o ajustamento os valores que tornam mínima a soma dos quadrados dos resíduos ponderados, conforme a Eq. 2.1.

$$\sum_{i+1}^{n} W_i v_i^2 = v^T W v = min \tag{2.1}$$

sendo W_i os pesos e v_i os resíduos.

Utilizando o método paramétrico, tem-se o que se segue em Eq. 2.2, Eq. 2.3 e Eq. 2.4:

$$Ax = L + v, N = A^T W A, U = A^T W L \tag{2.2}$$

Logo:

$$Nx - U = 0 \rightarrow x = N^{-1} U \tag{2.3}$$

Assim:

$$x = (A^T W A)^{-1} A^T W L \tag{2.4}$$

2.2.1.1 Decomposição de Valor Singular (SVD)

A Decomposição de Valor Singular (SVD) de uma matriz $A^{m \times n}$ é uma matriz de decomposição de grande importância no tratamento de problemas de mínimos quadrados. Tal composição fornece uma forma diagonal de A utilizando uma transformação de equivalência ortogonal. Tal decomposição é um dos resultados mais importantes em álgebra linear, seja ela teórica ou computacional.

Dada uma matriz $\mathbf{A}$, $m \times n$, a matriz $\mathbf{A}^T\mathbf{A}$, $n \times n$, é uma matriz simétrica com n valores próprios reais não negativos, igual ou distintos, λ_1, λ_1, λ_2, $\ldots$, λ_n.

Designam-se por **valores singulares** da matriz A as n raízes quadradas dos valores próprios da matriz $\mathbf{A}^T\mathbf{A}$, $\sigma_i = \sqrt{\lambda_i}$.

A SVD decompõe a matriz em um produto de outras três, sendo que duas são matrizes ortogonais ($MM^T = I$) e a terceira é diagonal, em que os valores que compõem tal diagonal são os valores singulares, ou seja, os autovalores, sendo isso fator determinante para o nome da decomposição.

Toda matriz $A^{m \times n}$ pode ser fatorada conforme a Eq. 2.5 abaixo:

$$A = U \sum V^T \tag{2.5}$$

sendo:

a) V é uma matriz ortogonal $n \times n$, construída a partir de um conjunto ortonormal de autovetores da matriz $A^T A$, $(v_1, v_2, ..., v_n)$

b) U é uma matriz ortogonal $m \times m$, cujos elementos são determinados seguindo a Eq. 2.6:

$$u_i = Av_i/\sigma_i \tag{2.6}$$

Se A possui r autovalores não nulos, $\sum$ (Figura 2.1) é uma matriz da seguinte forma:

$$\Sigma = \begin{array}{c} \begin{array}{cc} \overbrace{\qquad\qquad}^{r} & \overbrace{\qquad\qquad}^{n-r} \end{array} \\ \left[\begin{array}{ccc|ccc} & & & 0 & \cdots & 0 \\ & \mathbf{D} & & \vdots & \ddots & \vdots \\ & & & 0 & \cdots & 0 \\ \hline 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right] \begin{array}{l} \left.\vphantom{\begin{array}{c}0\\ \vdots\\ 0\end{array}}\right\} r \\ \left.\vphantom{\begin{array}{c}0\\ \vdots\\ 0\end{array}}\right\} m-r \end{array} \end{array}, \text{ em que } \mathbf{D} = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{bmatrix}$$

Figura 2.1 – Matriz $\sum$ da Decomposição SVD. (ÅKE BJÖRCK, 1996)

A matriz A, $m \times n$, com r valores singulares não nulos, pode ser escrita conforme a Eq. 2.7:

$$A = \sigma_1 u_1 v_1^T + \cdots + \sigma_r u_r v_r^T = \sum \sigma_i u_i v_i^T, i = 1, \ldots, r \tag{2.7}$$

em que u_i são colunas de U, ditos vetores singulares à esquerda, e v_i são colunas de V, ditos vetores singulares à direita.

Caso **A** possua r autovalores não nulos, tais valores compõem as colunas da matriz **U**. Caso $r < m$, será necessário determinar os $m - r$ vetores de $\Re^m$ convenientes (podendo, por exemplo, utilizar a ortogonalização de Gram-Schmidt tendo como base os versores canônicos).[4]

Para o processo de ortogonalização de Gram-Schmidt, dado um espaço vetorial euclidiano V e uma base qualquer $B = (v_1, v_2, ..., v_n)$ desse espaço,pode-se determinar uma base ortogonal de V.

Supondo $v_1, v_2, ..., v_n$ não ortogonais e fazendo $w_1 = v_1$, determina-se $w_2 = v_2 - \alpha w_1$ ortogonal a w_1 fazendo o que se encontra nas equações Eq. 2.8, Eq. 2.9, Eq. 2.10 e Eq. 2.11, que seguem abaixo:

$$\alpha = \frac{v_2 \cdot w_1}{w_1 \cdot w_1} \tag{2.8}$$

e

$$w_2 = v_2 - (\frac{v_2 \cdot w_1}{w_1 \cdot w_1})w_1 \tag{2.9}$$

de forma geral:

$$w_n = v_n - \alpha_{n-1} w_{n-1} - \cdots - \alpha_2 w_2 - \alpha_1 w_1 \tag{2.10}$$

em que

$$\alpha_1 = \frac{v_n w_1}{w_1 \cdot w_1}, \ldots, \alpha_{n-1} = \frac{v_n \cdot w_{n-1}}{w_{n-1} \cdot w_{n-1}} \tag{2.11}$$

Para obter uma base ortonormal, basta dividir cada w_i pelo seu respectivo módulo.[4]

2.2.1.2 Decomposição QR

A fatoração QR é uma alternativa muito importante para resolução de sistemas lineares por mínimos quadrados, principalmente em âmbito computacional, podendo, inclusive, ser utilizado para encontrar autovalores de uma matriz quadrada.

Qualquer matriz A cujas colunas são linearmente independentes pode ser fatorada na forma $A = QR$, onde Q é uma matriz ortogonal e R é uma matriz triangular superior invertível. O espaço coluna das matrizes A e Q é o mesmo, ou seja, as colunas de Q formam uma base ortonormal para o subespaço gerado pelas colunas de A.[5]

Além disso, tem-se também por teorema o fato de que uma matriz ortogonal é sempre invertível e sua inversa é dada pela sua transposta, ou seja, $Q^{-1} = Q^T$.[5]

Frente a isso, ao obter-se a matriz Q, que também pode ser obtida através do processo de Gram-Schmidt, a matriz R pode ser obtida da seguinte forma:

$$A = QR \Rightarrow Q^T A = \underbrace{Q^T Q}_{I} R \therefore R = Q^T A = Q^{-1} A$$

Figura 2.2 – Decomposição QR.(ÅKE BJÖRCK, 1996)

2.2.2 Minimização da norma L1 (ML1)

Para o ajustamento de redes geodésicas pela ML1, a soma dos módulos dos resíduos ponderados pelos respectivos pesos de suas observações independentes, por meio do vetor de pesos (composto pelos elementos da diagonal principal de W) deve ser mínima, conforme a Eq. 2.12 [3].

$$min(p^T|v|) \tag{2.12}$$

Para solução desse problema de minimização por programação linear, é necessário que todas variáveis x_i e v_i sejam não negativos. Frente a esse problema, são inseridas variáveis artificiais para x e v, como ilustram as equações Eq. 2.13 e Eq. 2.14 [3].

$$x = \alpha - \beta, v = u - w \tag{2.13}$$

$$Z_{ML1} = min(p^T|u - w|) \tag{2.14}$$

Adaptando o problema para resolução pelo SIMPLEx [3][6][7], sendo $I_{m\times m}$ uma matriz identidade, temos o que se segue na Eq. 2.15 [3]:

$$\begin{bmatrix} A & -A & -I & I \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ u \\ w \end{bmatrix} = [l] \tag{2.15}$$

Voltando à função objetivo, para um determinado resíduo v_i, além das respectivas variáveis artificiais u_i e w_i serem não negativas, pode-se garantir que no mínimo uma delas será nula, por ambas jamais farão parte de uma solução básica simultaneamente pelo método SIMPLEx [3][6][7]. Com isso, suas colunas serão sempre linearmmente dependentes, por serem correspondentes em $I_{m\times m}$ e $-I_{m\times m}$ [3].

Com isso, sendo Z_{ML1} a função objetivo a ser minimizada conforme Eq. 2.16, sendo 0_{mx1} um vetor de zeros, temos o que se segue na Eq. 2.17 [3]:

$$Z_{ML1} = min\{p^T(u+w)\} \tag{2.16}$$

$$Z_{ML1} = \begin{bmatrix} 0^T & 0^T & p^T & p^T \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ u \\ w \end{bmatrix} \rightarrow min \tag{2.17}$$

2.2.3 Minimização da norma L_∞ (ML∞)

Já para os ajustamentos de redes geodésicas pela ML_∞, o maior valor absoluto dentre os resíduos ponderados deve ser minimizado. Para solução utilizando programação linear, assim como para ML_1, deve-se inserir variáveis artificiais para x e v, visando permitir valores negativos para seus elementos. Após, cria-se uma variável unidimensional s, a qual deve ser minimizada e caracteriza a função objetivo $Z_{ML\infty}$, conforme Eq. 2.18 [3].

$$Z_{ML\infty} = \begin{bmatrix} 0^T & [1] \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ u \\ w \\ s \end{bmatrix} \rightarrow min \tag{2.18}$$

A estratégia é impor que o módulo de cada um dos resíduos v_i (substituídos por $u_i - w_i$) multiplicados pelos pesos das respectivas observações p_i (dados pelos elementos $P_{i,i}$ de P) sejam menores ou iguais a s, sendo, assim, restrições do problema de programação

linear (Eq. 2.19) [3].

$$\begin{bmatrix} 0 & 0 & -P & P & [-1] \\ 0 & 0 & P & -P & [-1] \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ u \\ w \\ s \end{bmatrix} \leq 0 \tag{2.19}$$

Utilizando a mesma restrição utilizada para o ML_1 adicionando a variável s, conclui-se a Eq. 2.20 [3]:

$$\begin{bmatrix} A & -A & -I & I & 0_s \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ u \\ w \\ s \end{bmatrix} = [l] \tag{2.20}$$

2.3 Comparação entre MMQ, ML1 e ML∞

Levando em consideração os métodos MMQ, ML1 e ML∞, temos a seguinte tabela de comparação:

Tabela 2.1 – Comparação Entre as Normas Vetoriais (SURACI, 2018)

Norma Min	**Outliers?**	**Solução usual**	**Sempre unica?**
L1 (ML_1)	Sim	Prog. linear	Não
L2 (MMQ)	Não	Operações "simples"	Sim
L∞ (ML∞)	Não	Prog. Linear	Não

Com Base na Tabela 2.1, obseva-se que enquanto os métodos ML_1 e ML_∞ são normalmente solucionados por programação linear, o método MMQ pode ser solucionado utilizando operações básicas de matrizes, fazendo com que o MMQ seja, comparativamente, mais simples de se revolver. Outro ponto positivo para o Método dos Mínimos Quadrados é que o mesmo possui solução única quando a matriz dos coeficientes das incógnitas possuir posto completo [3], enquanto que os demais métodos estudados nesse exemplo não.

Mesmo não sendo um dos mais indicados nesse caso, o método ML_1 é o melhor para observação de outliers, ou seja, pontos que estão fora do padrão das observações.

O ML_∞ é o mais indicado quando se deseja distribuir melhor os erros pois, ao minimizar o máximo resíduo, os erros do levantamento são automaticamente distribuidos no ajustamento, sendo muito importante quando se visa obter resultados mais homogêneos.

2.4 Ferramentas

Para o desenvolvimento de uma plataforma, podem ser utilizadas as mesmas ferramentas empregadas por Monteiro e Marchiore [1] em seu projeto. Nesse contexto, o principal objetivo é manter a simplicidade e homogeneidade da plataforma no que tange ao âmbito da programação, utilizando mesma linguagem e utilizando recursos já disponíveis visando uma simplicidade na integração entre as plataformas. As principais ferramentas que serão empregadas estarão listadas nesse tópico.

2.4.1 GitHub

O GitHub [8] é uma plataforma online que permite a construção de projetos de software. A principal vantagem do GitHub é a integração dessa plataforma com outras plataformas oferecendo o serviço de hospedagem web. Sendo asim, o GitHub é responsável pelo armazenamento e estruturação do projeto enquanto que outra plataforma, como o Python Anywhere ou a Heroku, será o responsável por disponibilizar o resultado do processamento online.

A integração ocorre através de uma chave SSH, que é um método de identificação entre o servidor local, onde é desenvolvido o projeto, e o servidor onde será hospedada a plataforma utilizando um método de autenticação. Primeiramente há uma integração entre o servidor local e o GitHub, e posteriormente, é realizada a integração do GitHub com o Python Anywhere também por intermédio da chave SSH. A partir disso, a plataforma é disponibilizada de forma online e atualizada após atualização dos códigos no GitHub.

2.4.2 Python Anywhere

O Python Anywhere [9] é um ambiente de desenvolvimento integrado (IDE) e serviço de hospedagem web baseado em Python [9]. Enquanto a programação, o desenvolvimento da plataforma pode ser realizado em ambiente local e posteriormente realizar um *upload* do projeto. Esse carregamento dos dados ocorrerá por intermédio do GitHub, que armazenará os códigos do projeto. A partir disso, é realizada uma integração entre o Python Anywhere e o GitHub, onde o GitHub armazena os dados, enquanto que o Python Anywhere processa e cria a plataforma.

2.4.3 Linguagens de Programação

No desenvolvimento desse modelo de plataforma web proposto, as principais linguagens de programação que devem ser implementadas são o Python 3.8 e o HTML. O Python deve ser utilizado tanto no desenvolvimento dos códigos dos algoritmos adotados, como os que foram apresentados na seção 2.2, quanto na integração com o HTML através do micro framework Flask [10]. A utilização do Flask nesse modelo de projeto destaca os

pontos principais desse micro framework, que são: a rapidez no desenvolvimento, devido a simplicidade que a estrutura aborda; e ser normalmente atribuido a projetos menores, não tendo a necessidade de uma arquitetura complexa para o desenvolvimento do projeto.

Nesse ponto, o micro framework Flask se destaca em relação ao framework Django, que possui uma arquitetura e estrutura de projeto um pouco mais complexa, não sendo necessária sua abordagem nesse tipo projeto uma vez que as especificações do Flask já atendem as demandas. Dessa forma, o desenvolvimento em Python está relacionado tanto com a sua, quando relacionado com o HTML através do Flask, quanto com o desenvolvimento dos códigos referente aos algoritmos que serão implementados.

2.4.4 Bibliotecas do Python

Na construção de uma plataforma, pode-se utilizar as bibliotecas que estão disponíveis no Python. Nessa seção são listadas alguams bibliotecas de modo a direcionar uma pesquisa dependendo da necessidades do projeto. Todas essas bibliotecas podem ser instaladas através do comando *pip install "nome da biblioteca"* de forma direta, isto é, inserindo cada uma no prompt de comando ou através de um arquivo de texto chamado *requirements.txt* que cria uma lista dessas bibliotecas relacionando também as versões das bibliotecas, como pode ser visualizado no APÊNDICE A.

a) *alembic*: biblioteca que tem como ferramentas a migração de um banco de dados; [11]

b) *chardet*: biblioteca que reconhece a codificação que o texto está escrito; [11]

c) *click*: pacote para criar interfaces de linhas de comando simplificando sua digitação; [11]

d) *flask*: micro framework para aplicações web; [10]

e) *httplib2*: pacote para reconhecimento das bibliotecas HTML;[11]

f) *idna*: pacote para o protocolo de nomes de domínio internacionalizados especificados no RFC 5891; [11]

g) *itsdangerous*: pacote para a segurança de informações transmitidas; [11]

h) *Jinja2*: pacote que auxilia na digitação dos códigos em HTML permintindo que os modelos sejam escritos de forma semelhante à sintaxe do Python; [12]

i) *lazy*: pacote para a atributos computados de forma lenta; [11]

j) *mako*: pacote que fornece uma sintaxe familiar não semelhante a sintaxe XML e que compila em módulos de Python; [11]

k) *SQLAlchemy*: pacote com um kit de ferramentas SQL que podem ser empregados dentro de uma aplicação Python; [11]

l) *urllib3*: cliente HTTP que possuir recursos importantes e que não estão contidas nas bibliotecas padrão do Python; [11]

m) *Werkzeug*: pacote que contem uma biblioteca de aplicativos web WSGI; e [13]

n) *WTForms*: pacote que realiza a validação e renderização de formulários para o desenvolvimento web em Python. [11]

2.5 Requisitos do Projeto

O projeto deve manter a estrutura da plataforma desenvolvida por Monteiro e Marchiore [1], isto é, manter a mesma linguagem de programação e ferramentas que foram empregadas. Desse modo, a integração da plataforma de ajustamento poderá ser realizada com a de séries temporais, haja vista que ambas estarão estruturadas de forma similiar. As premissas básicas para o desenvolvimento da plataforma estão listadas a seguir.

2.5.1 Estrutura da Plataforma

Dentro da plataforma, o usuário deve poder realizar as seguintes tarefas:

a) Escolher o algoritmo que será utilizado;

b) Fazer o carregamento dos dados; e

c) Extrair os resultados do processamento.

Nesse ponto, a plataforma irá conter um sumário descrevendo como os algoritmos funcionam e como realizar o carregamento dos dados. Após o carregamento e processamento dos dados, os resultados estarão disponíveis para o *download* contendo os dados ajustados e testes de significância de modo a verificar a acurácia dos dados.

2.5.2 Algoritmos

A implementação dos algoritmos será realizada em *Python* utilizando de funções já programadas da biblioteca *statsmodel* [14] e do modelo proposto por Monteiro e Marchiore [1] para a inserção de novos algoritmos na plataforma.

O template e a estrutura da plataforma são desenvolvidos em *HTML* e o projeto é estruturado de forma que os algoritmos sejam adicionados em *Python*. Nesse ponto, é importante destacar que adicionar novos algoritmos será algo possível dentro da plataforma e a linguagem escolhida foi o *Python* devido a familiaridade dos autores com a mesma.

3 METODOLOGIA

3.1 Desenvolvimento do projeto

O projeto de desenvolvimento de uma plataforma segue as etapas abaixo explicitadas no fluxograma aprensatado Figura 3.1.

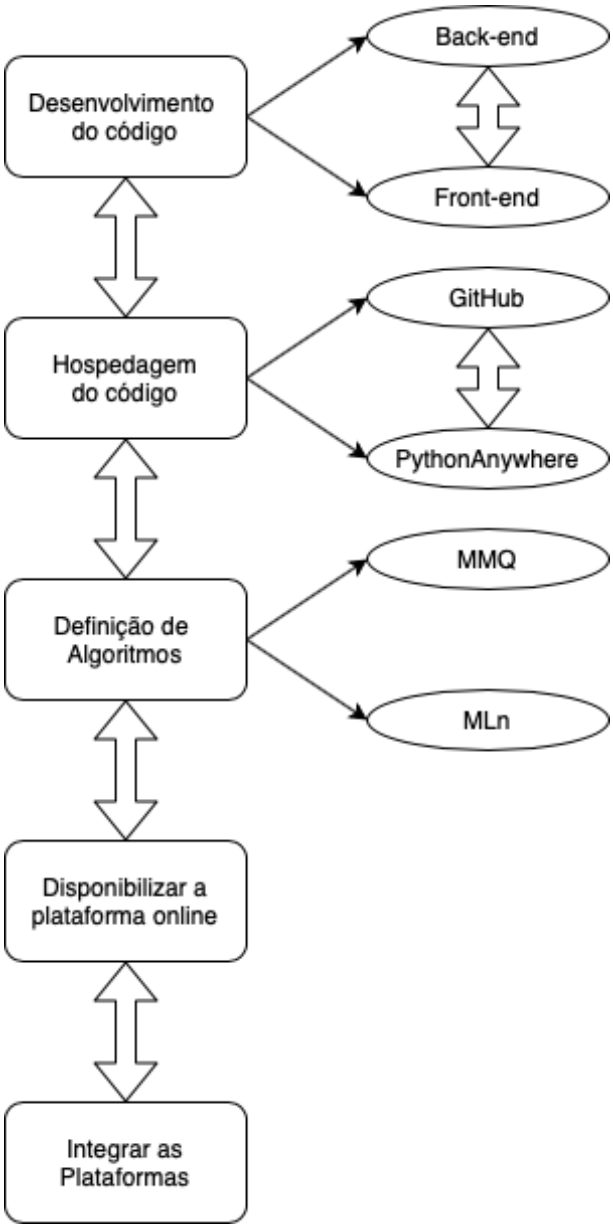


Figura 3.1 – Fluxograma de atividades.

O desenvolvimento de um projeto pode ser compreendido como a execução dessas 5 etapas: Desenvolvimento do código; Hospedagem do código; Definição de Algoritmos; Diponibilizar a plataforma de forma online; e integrar as plataformas. A primeira etapa tem como objetivo a construção da plataforma e de seu *layout*. Essa etapa pode ser dividida em

duas subetapas o *back-end* e o *front-end*, sendo o *back-end* responsável pela estruturação da plataforma e o *front-end* pelo *layout*.

A etapa de hospedagem do código é importante para que se haja a contribuição de usuários a partir do conteúdo que será disponibilizado pelo GitHub. Tendo acesso a documentação da plataforma, o usuário poderá inserir novos algoritmos realizando algumas alterações conforme será citado em um exemplo no Seção 3.2.

A etapa de definição de algoritmos tange ao fato de se escolher quais serão os algoritmos que serão implementados na plataforma, como por exemplo o MMQ e o MLn, apresentados como exemplos durante esse projeto. Em seguida, a dispobinibilização de forma online se dá através do *deploy* desses códigos no GitHub e, assim, realizando a disponibilização através de uma plataforma que realiza essas tarefa, como é o caso das plataformas PythonAnywhere e Heroku. Por fim, a integração das plataformas pode ser feita criando uma nova plataforma e disponibilizando os links das plataformas que já foram desenvolvidas.

No desenvolvimento da plataforma, as principais ferramentas empregadas são o Python 3.8 e o HTML. O HTML é utilizado para a estrutura da plataforma, sendo a linguagem predominante do desenvolvimento das etapas de *Back-end*, Implementação dos algoritmos, *Front-end* e Carregamento online (*Deploy*).

3.2 Emprego dos algoritmos

Para a criação de uma plataforma *online* que utilize o Python como uma das linguagens empregadas, deve-se primeiramente definir o *setup* do ambiente de desenvolvimento. Nesse projeto, foi utilizado com base para o desenvolvimento as seguintes interfaces: Anaconda [15] para interpretar as linguagens de programação; e o Atom como IDE para desenvolvimento do código. É importante ressaltar que na configuração de instalação da Anaconda, deve-se optar pela opção que faça a conexão com o *PATH*, como apresentando no APÊNDICE A, o que permite que comandos sejam implementados de forma mais simples no *prompt* de comando.

O Atom [16] é uma IDE gratuita que pode ser utilizada para a construção de aplicativos e plataformas web. Visando um projeto que utilize principalmente das linguagens HTML e Python, existem alguns pacotes que devem ser instalados de modo que essa IDE realize a interpretação dessas linguagens, e que pode ser encontrada no APÊNDICE A desse projeto.

Em termos de desenvolvimento do projeto, após as etapas iniciais de pesquisa bibliográfica, a primeira etapa é a definição do *setup* apresentada no Apêndice B, onde é criado um ambiente virtual onde será desenvolvida a plataforma. Isso significa que em uma

primeira etapa a plataforma é desenvolvida em ambiente local e em uma etapa posterior, o *deploy*, é colocada de forma online. Após estruturado o ambiente virtual onde serão desenvolvidos os códigos, a primeira etapa é chamada de *back-end*. O *back-end* é focado na estruturação do projeto e após determinada etapa ocorre de forma paralela ao *front-end*, que por sua vez tem seu direcionamento na apresentação da plataforma.

A construção da plataforma ocorre de maneira significativa na etapa de *back-end*. Nessa etapa ocorre a criação dos códigos em HTML e Python e todas as conexões estabelicidas na definição do projeto. Sendo assim, é principalmente nessa etapa onde se é dedicado o tempo de desenvolvimento da aplicação. A complexidade do projeto não está concentrada na estruturação básica da plataforma, mas sim em como realizar as relações no código de modo que se consiga fazer a leituras de arquivos de entrada, processamento dos dados e, assim, gerar os resultados.

Em paralelo com as etapas de *back-end* tem-se a etapa de *front-end*. O *front-end* é onde será desenvolvida a forma em que a plataforma será apresentada, ou seja, essa etapa é dependente do desenvolvimento da etapa de *back-end*, porém é importante o desenvolvimento de ambas ocorrer em paralelo, pois assim já se pode visualizar os resultados durante a construção da plataforma. Existem ferramentas online que auxiliam no desenvolvimento do *front-end*, como é o caso do Bootstrap[17]. Essa ferramenta fornece códigos e estruturas já prontas de funções para apresentação no site, como pode ser visualizado na Figura 3.2.

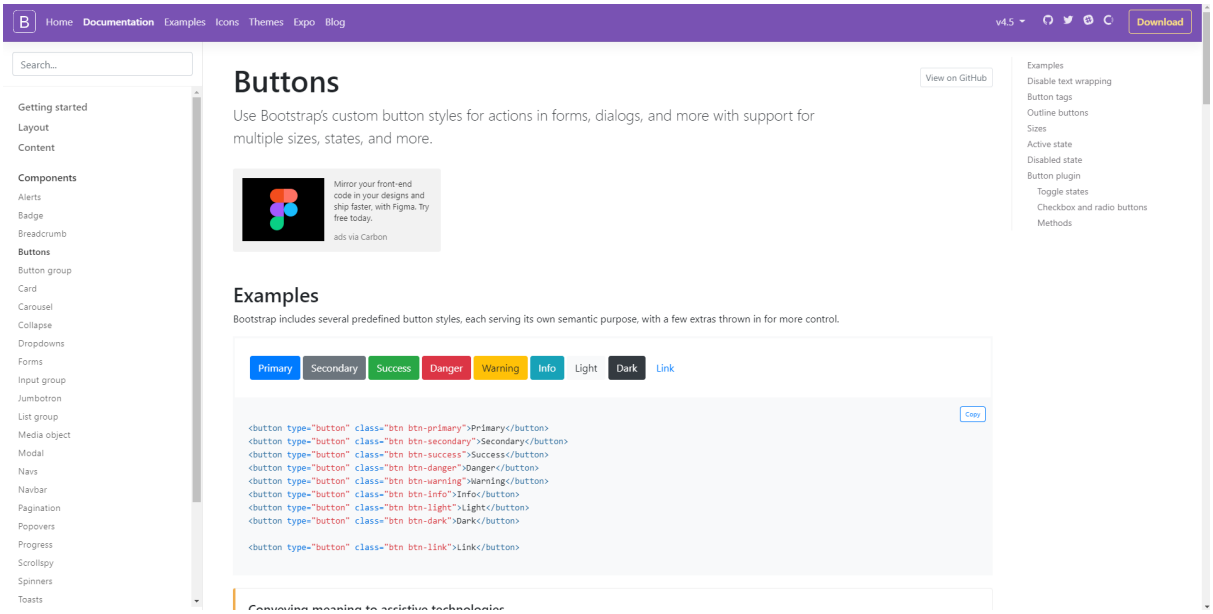


Figura 3.2 – Códigos para inserção de botões.

A última etapa para desenvolvimento da plataforma é a disponibilização da plataforma de forma online (*deploy*). Para realizar o *deploy* deve-se primeiro possuir uma aplicação Flask, ou seja, as etapas de *back-end* e *front-end* já devem possuir um nível de

desenvolvimento de modo que os resultados que são gerados pelos aplicativos possam ser exibidos. Apesar do *deploy* ocorrer como uma das últimas atividades do ciclo do projeto, é importante que exista um tempo para que se teste as funcionalidades e operação dos aplicativos disponibilizados.

O *deploy* pode ser realizado de forma direta em algumas plataformas de disponibilização online, ou pode ser realizado diretamente através do Atom, como apresentando no Apêndice C para o exemplo da plataforma Heroku[18]. O site Heroku é uma plataforma de disponibilzação de aplicações web em Python. Outra plataforma disponível é o PythonAnywhere[9] e ambas realizam a disponibilização das plataformas de forma gratuita. No trabalho de Monteiro e Marchiore [1], a escolha de plataforma foi o PythonAnywhere, pois após o teste em ambas plataformas, para aquele projeto o PythonAnywhere apresentou um processamento mais rápido das rotinas que haviam sido implementadas. No entanto, ambas plataformas possuem essa disponibilidade e dependendo do projeto uma pode apresentar um desempenho superior a outra podendo ser verificada a partir do teste em ambas plataformas. Nesse projeto é apresentado os passos de como se realizar o *deploy* de uma aplicação Flask na plataforma Heroku e no trabalho de Monteiro e Marchiore [1] é citado o processo referente a plataforma PythonAnywhere.

Por fim, é importante destacar que o projeto desenvolvido tem como um dos objetivos permitir a contribuição de colaboradores externos ao projeto no que tange à inserção de novos algoritmos. Essa contribuição poderá ser realizada através do GitHub, que é uma plataforma para hospedagem dos códigos. Dessa forma, o projeto deve ser desenvolvido de modo que se possibilite a inserção de novos códigos. A inserção desses novos algoritmos depende do modelo de desenvolvimento que o projeto foi construido. Nesse ponto, o projeto de Monteiro e Marchiore [1] cita uma forma de se realizar essa condição. Além da existência do código em Python contendo o algoritmo, no código em HTML esse novo algoritmo deve ser adicionado criando a página onde será compilado e uma porta para entrada dos dados. Na Figura 3.3 é apresentado uma fórmula genérica sobre a adição de algoritmos proposta por Monteiro e Marchiore [1].

No ANEXO A tem-se um exemplo de rede de nivelamento que pode ser ajustada pelos métodos MMQ, ML1 e ML_∞ (como exemplo) e que pode estar presente na plataforma de ajustamento visando auxiliar o usuário da mesma a entender como se dá a entrada de dados para utilizar algum determinado método.

Código 4.1: Adição de nova página de algoritmo

```
1  @app.route('/algorithms/acf', methods= ['GET', 'POST'])
2  def algorithms_acf():
3      text = desc_list['01acf']
4      form = FormACF()
5
6      if form.validate_on_submit():
7          (...)
8          return render_template('algorithms_nome_output.html',
               title='Titulo', ...)
9
10     else:
11         file_url = None
12
13     return render_template('algorithms_nome.html', title='Titulo
           ', text=text, form= form, file_url=file_url)
```

Figura 3.3 – Estrutura para inserção de novos algoritmos. (MONTEIRO e MARCHIORE, 2018)

4 CONCLUSÃO

Torna-se, portanto, evidente que o desenvolvimento de uma plataforma web é dependende de mais de uma linguagem de programação, sendo necessário reservar um tempo no início do projeto de direcionamento para aprendizado dessas liguagens.

O projeto possui uma vertente colaborativa no que tange a inserção de novos algoritmos na linguagem Python, isto é, um usuário pode contribuir com um algoritmo que seja compilado em Python e esse se tornará mais um aplicativo na plataforma web. Dessa forma, a plataforma desve ser desenvolvida de modo que possa compilar o Python, o que justifica a utilização do Flask, que será uma forma de realizar essa conexão com o HTML.

A plataforma deve ser desenvolvida utilizando principalmente HTML. O Flask, sendo uma biblioteca do Python, tem a principal funcionalidade de realizar uma conexão entre ambas linguagens e permitir que a plataforma web possa compilar programas utilizando Python. Sendo assim, bibliotecas e funcionalidades que são empregadas em Python, isto é, ferramentas, bibliotecas e outras, podem ser utilizadas no projeto devido a utilização do Flask.

Existem outras bibliotecas contidas no Python que também podem realizar essa conexão com o HTML. O Django [referencia] é uma outra biblioteca que possui essa funcionalidade. Ambas as bibliotecas, o Flask e o Django, conseguem entregar a mesma operação, porém o Flask possui uma abordagem mais simples de ser compreendida sem que se perca nenhuma das principais funcionalidades da aplicação web, sendo portanto, uma boa opção para o desenvolvimento desse modelo de projeto onde minimizar o custo de tempo é importante para que se possa atingir de forma esperada os resultados almejados.

O desenvolvimento do código, como mencionado na metodologia, pode ser subdividido em *back-end* e *front-end*. Essa divisão permite o aumentar o foco em diferentes partes do projeto, sendo o *back-end* responsável pela estruturação da plataforma e o *front-end* responsável pela apresentação da plataforma, isto é, as funções que são visualizadas na aplicação. Portanto, recomenda-se que o desenvolvimento da plataforma seja realizado de forma separada, pois dividir o foco nessas duas grandes etapas permitirá um melhor fluxo no desenvolvimento do projeto.

Por fim, após desenvolvidas as etapas de *back-end* e *front-end* é realizado o carregamento dos dados na etapa denominada *deploy*. As plataformas Heroku e PythonAnywhere são dois exemplos de plataformas em que se é realizado o *deploy* do projeto. A plataforma PythonAnywhere, no entanto, apresenta um melhor desempenho que a plataforma Heroku em termos de processamento das aplicações processadas em Python, conforme citada

por Monteiro e Marchiore [1]. No entanto, ambas plataformas possuem capacidade de suportar projetos de aplicação web utilizando Python, mas a plataforma PythonAnywhere demonstrou um processamento mais rápido das rotinas empregas pelo trabalho de Monteiro e Marchiore [1].

4.1 Sugestão para trabalhos futuros

Como sugestão para trabalhos futuros que tenham como objetivo a construção de uma plataforma web, deve-se primeiramente atentar-se os requisitos necessários para o desenvolvimento da mesma: HTML, CSS, Python, e outras ferramentas apresentadas durante esse projeto, que de forma geral se relacionam para o funcionamento dessa aplicação.

O Python possui diversas bibliotecas que realizam diferentes operações. Para o desenvolvimento web, utilizando como base o Flask, destacam-se principalmente as bibliotecas WTForms que permite a utilização de formulários, e Jinja2 que é um mecanismo que facilita a aplicação HTML dentro de uma aplicação Python, sendo uma forma relevante de se aproveitar o conhecimento de Python, isto é, não dependendo apenas do HTML para criação da plataforma.

REFERÊNCIAS

1 MONTEIRO, M. N.; MARCHIORE, T. N. *Plataforma WEB para anélise de séries temporais*. 85 p. Monografia (Graduação em Engenharia Cartográfica) — Instituto Militar de Engenharia, Rio de Janeiro, 2018.

2 BJöRCK Åke. *Numerical Methods for Least Squares Problems*. 1. ed. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 1996. 404 p.

3 SURACI, M. S. S. *Aplicação das Normas L1 e L∞ em Redes Altimétricas: Identificação de Outliers e Construção do Modelo Estocástico*. 91 p. Mestrado em Ciências em Engenharia Cartográfica — Instituto Militar de Engenharia (IME), Rio de Janeiro, 2018. 10 de ago. de 2020. Disponível em: <https://www.researchgate.net/publication/342357822_Aplicacao_das_normas_L1_e_L_em_redes_altimetricas_identificacao_de_outliers_e_construcao_do_modelo_estocastico>.

4 NEPOMUCENO, P. E. G. *Métodos Numéricos: Sistema de Equações Lineares*. 2016. 12 ago. 2020. Disponível em: <https://www.ufsj.edu.br/portal2-repositorio/File/nepomuceno/mn/08MN_SL6.pdf>.

5 MATEMáTICA, U. I. R. E. A. de. *Fatoração QR e matrizes ortogonais*. 2020. 8 ago. 2020. Disponível em: <https://www.ufrgs.br/reamat/AlgebraLinear/livro/s13-fatorax00e7x00e3o_qr_e_matrizes_ortogonais.html>.

6 UNESP. *O Método Simplex*. 2020. 23 set. 2020. Disponível em: <http://wwwp.fc.unesp.br/~arbalbo/Iniciacao_Cientifica/simplex/teoria/3_simplex.pdf>.

7 PHPSIMPLEX. *Teoria do Método Simplex*. 2020. 23 set. 2020. Disponível em: <http://www.phpsimplex.com/pt/teoria_metodo_simplex.htm>.

8 GITHUB. *GitHub, Built for developers*. 2020. 10 ago. de 2020. Disponível em: <https://github.com/>.

9 PYTHONANYWHERE. *Host, run, and code Python in the cloud*. 2020. 10 ago. de 2020. Disponível em: <https://www.pythonanywhere.com>.

10 RONACHER, A. *Flask, web development, one drop at a time*. 2020. 04 nov. 2020. Disponível em: <https://flask.palletsprojects.com/en/1.1.x/>.

11 PYPI. *Python Package Index*. 2020. 04 nov. 2020. Disponível em: <https://pypi.org/>.

12 RONACHER, A. *Jinja*. 2020. 04 nov. 2020. Disponível em: <https://jinja.palletsprojects.com/en/2.11.x/>.

13 RONACHER, A. *Werkzeug*. 2020. 04 nov. 2020. Disponível em: <https://werkzeug.palletsprojects.com/en/1.0.x/>.

14 PERKTOLD, J.; SEABOLD, S.; TAYLOR, J. *StatsModels Statistics in Python*. 2020. 10 ago. de 2020. Disponível em: <https://www.statsmodels.org/stable/index.html>.

15 ANACONDA. *Data science technology for humam sensemaking*. 2020. 04 nov. 2020. Disponível em: <https://atom.io/>.

16 ATOM. *Atom*. 2020. 04 nov. 2020. Disponível em: <https://atom.io/>.

17 BOOTSTRAP. *Bootstrap*. 2020. 04 nov. 2020. Disponível em: <https://getbootstrap.com/>.

18 LINDENBAUM, J.; WIGGINS, A.; HENRY, O. *Heroku*. 2020. 04 nov. 2020. Disponível em: <https://www.heroku.com/>.

19 CENTER, H. D. *Heroku CLI*. 2020. 04 nov. 2020. Disponível em: <https://git-scm.com/doc>.

20 SCM, G. *Git SCM*. 2020. 04 nov. 2020. Disponível em: <https://git-scm.com/doc>.

21 AGENCY, N. G.-I. *Office of Geomatics*. 2020. 03 out. 2020. Disponível em: <http://earth-info.nga.mil/GandG/update/index.php?dir=gnss&action=gnss>.

APÊNDICE A – SETUP PARA DESENVOLVIMENTO

Para a confuração do *setup* primeiramente deve-se instalar o Python, podendo ser realizado através da instalação da plataforma Anaconda. Nesse ponto, apenas deve-se atentar para a inclusão da Anaconda como uma variável de ambiente do Windows. A instalação pode ser realizada diretamente no site da plataforma e essa inclusão como variável de ambiente pode ser realizada durante a instalação como apresentado no guia de instalação disponível pela plataforma e apresentado na Figura A.1.

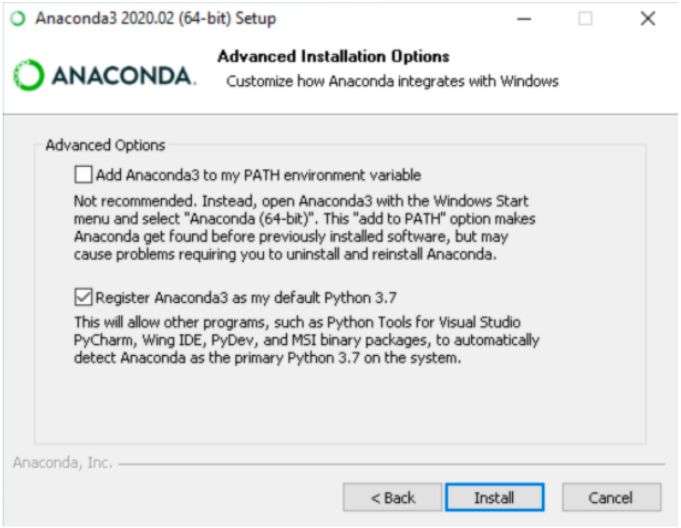


Figura A.1 – Configuração de instalação Anaconda.

A etapa de definição do Python através da Anaconda como uma variável de ambiente é importante para o desenvolvimento do projeto e demais configurações de *setup* apresentadas nesse tópico. Sendo assim, é possível verificar se foi configurado corretamente através das configurações do sistema, seguindo o caminho apontado pela Figura A.2.

Em seguida, deve-se instalar a plataforma onde serão desenvolvidos os código. O Atom, além de ser uma IDE de fácil acesso e gratuita, integra facilmente as configurações da plataforma Anaconda, tornando assim uma opção que viabiliza o desenvolvimento do projeto. Após a instalação do Atom, disponível em seu site, deve-se realizar algumas configurações para que o sistema realize o projeto. A primeira configuração é enquanto ao terminal de comando. Através da aba de configurações pode instalar o pacote: *plataformio-ide-terminal*, como apresentado na Figura A.3. A principal vantagem dessa configuração é realizar as rotinas necessárias no prompt de comando dentro da própia IDE. Após a intalação é importante verificar se a plataforma encaminhou a rota correta, como apresenta a Figura A.4.

Além disso, deve-se realizar a integração com o Python através do pacote: *autocomplete-*

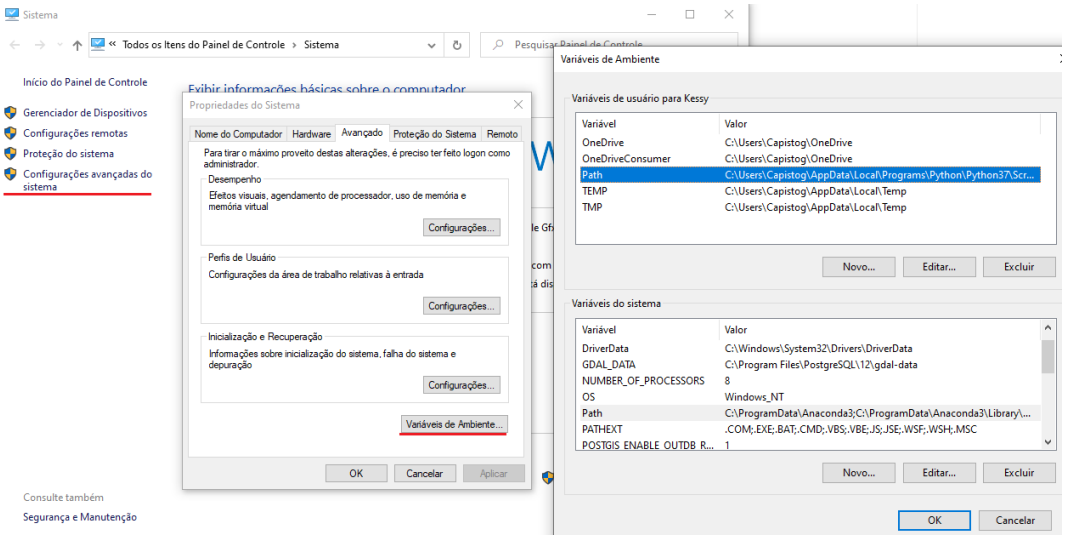


Figura A.2 – Configuração de variáveis de ambiente.

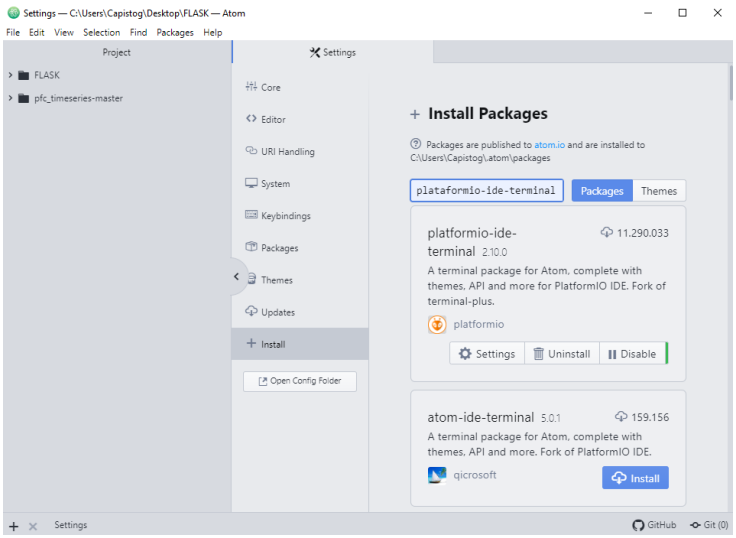


Figura A.3 – Integração com o terminal de comando.

python, como apresentado na Figura A.5. Esse pacote permite realizar rotinas no Python dentro dessa IDE.

Após essas definições, pode-se criar um ambiente virtual em Python utilizando o prompt de comando. O ambiente virtual será utilizado para o desenvolvimento da plataforma através do Flask. Por fim, serão instalados algumas bibliotecas que auxiliam nesse desenvolvimento da plataforma, apresetada na Figura A.6. As Figuras A.7 e A.8 apresentam, respectivamente, a forma de se criar uma ambiente virtual em Python e como instalar as bibliotecas através de um arquivo de texto contendo essas bibliotecas.

Após a conclusão da instalação das bibliotecas no ambiente virtual, o prompt de comando indicará as bibliotecas instaladas, conforme apresentado na Figura A.9.

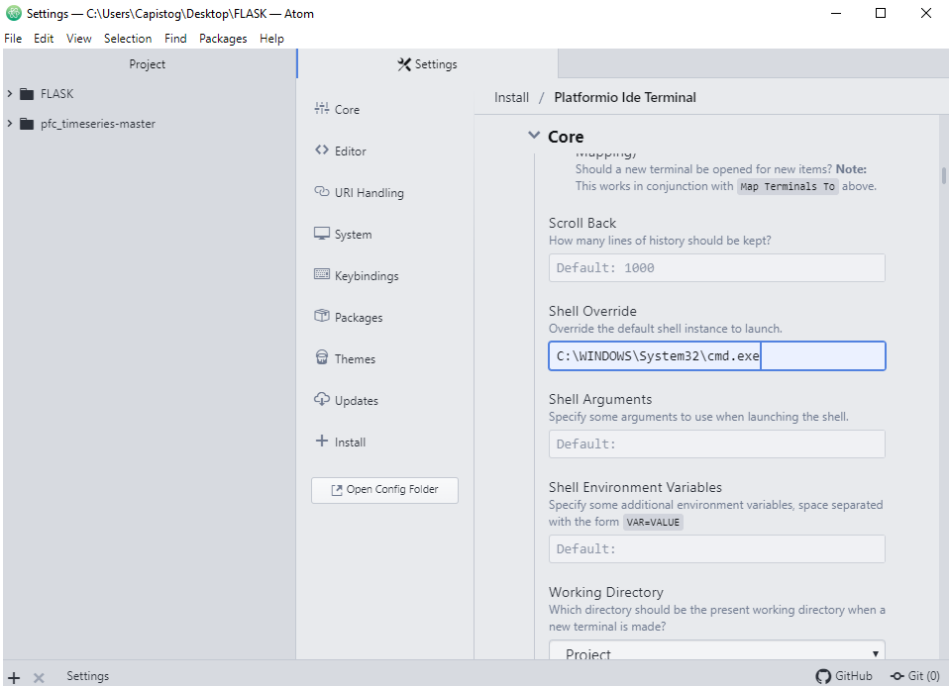


Figura A.4 – Configurações do pacote de integração com o terminal.

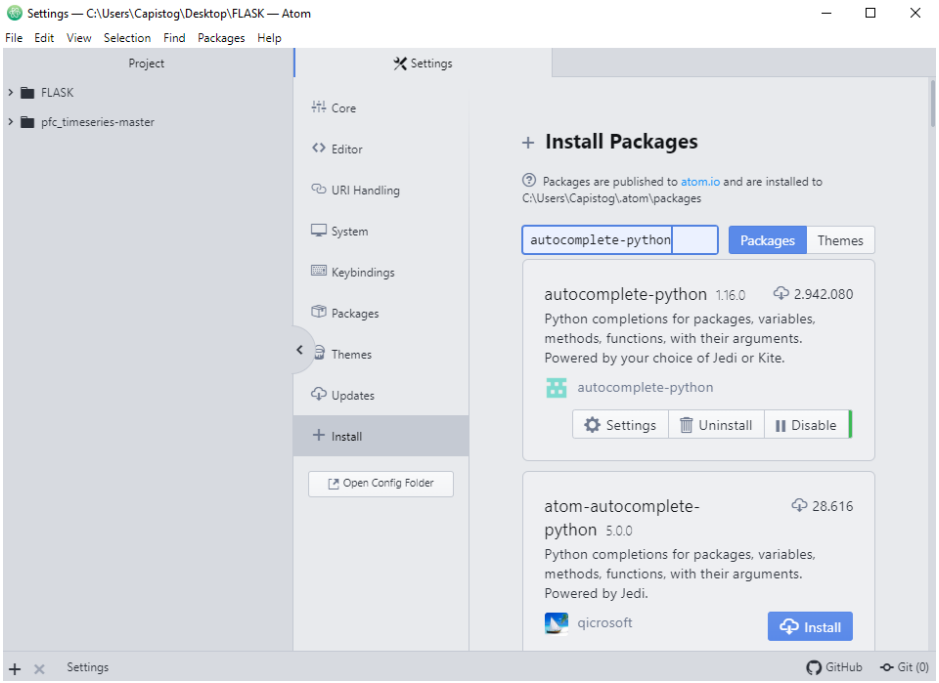


Figura A.5 – Integração com o Python.

requirements.txt - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
alembic==0.9.9
blinker==1.4
chardet==3.0.4
click==6.7
Flask==1.0.2
Flask-Dance==0.14.0
Flask-DebugToolbar==0.10.1
Flask-Login==0.4.1
Flask-Migrate==2.1.1
Flask-OAuth==0.12
Flask-OAuthlib==0.9.4
Flask-SQLAlchemy==2.3.2
Flask-WTF==0.14.2
httplib2==0.11.3
idna==2.6
itsdangerous==0.24
Jinja2==2.10
lazy==1.3
Mako==1.0.7
oauth2==1.9.0.post1
oauthlib==2.0.7
python-dateutil==2.7.2
python-editor==1.0.3
requests==2.18.4
requests-oauthlib==0.8.0
six==1.11.0
SQLAlchemy==1.2.6
SQLAlchemy-Utils==0.33.2
urllib3==1.22
URLObject==2.4.3
Werkzeug==0.14.1
wincertstore==0.2
WTForms==2.1
```

Ln 34, Col 1 100% Unix (LF) UTF-8

Figura A.6 – Bibliotecas Python.

Prompt de Comando - python app.py

```
C:\Users\Capistog\Desktop\PFC>conda create -n flaskdeploy flask
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.8.3
  latest version: 4.9.1

Please update conda by running

    $ conda update -n base -c defaults conda
```

Figura A.7 – Construção do ambiente virtual.

Prompt de Comando - conda deactivate - activate flaskdeploy

```
C:\Users\Capistog\Desktop\PFC>activate flaskdeploy

C:\Users\Capistog\Desktop\PFC>conda.bat activate flaskdeploy

(flaskdeploy) C:\Users\Capistog\Desktop\PFC>pip install -r requirements.txt
Processing c:\users\capistog\appdata\local\pip\cache\wheels\38\eb\46\936f2c130d99c643a
0403c9555ef850a5a1a39f84a7c00a65d\alembic-0.9.9-py2.py3-none-any.whl
Processing c:\users\capistog\appdata\local\pip\cache\wheels\b7\a5\68\fe632054a5eadd531
c7a49d740c50eb6adfbeca822b4eab8d4\blinker-1.4-py3-none-any.whl
Requirement already satisfied: certifi==2020.6.20 in c:\users\capistog\.conda\envs\fla
skdeploy\lib\site-packages (from -r requirements.txt (line 3)) (2020.6.20)
```

Figura A.8 – Instalação de bibliotecas.

Prompt de Comando - conda deactivate - activate flaskdeploy

```
Successfully installed Flask-1.0.2 Flask-Dance-0.14.0 Flask-DebugToolbar-0.10.1 Flask-
Login-0.4.1 Flask-Migrate-2.1.1 Flask-OAuth-0.12 Flask-OAuthlib-0.9.4 Flask-SQLAlchemy
-2.3.2 Flask-WTF-0.14.2 Jinja2-2.10 Mako-1.0.7 SQLAlchemy-1.2.6 SQLAlchemy-Utils-0.33.
2 URLObject-2.4.3 WTForms-2.1 Werkzeug-0.14.1 alembic-0.9.9 blinker-1.4 chardet-3.0.4
click-6.7 httplib2-0.11.3 idna-2.6 itsdangerous-0.24 lazy-1.3 oauth2-1.9.0.post1 oauth
lib-2.0.7 python-dateutil-2.7.2 python-editor-1.0.3 requests-2.18.4 requests-oauthlib-
0.8.0 six-1.11.0 urllib3-1.22

(flaskdeploy) C:\Users\Capistog\Desktop\PFC>
```

Figura A.9 – Conclusão da instalação das bibliotecas.

APÊNDICE B – DEPLOYMENT NA PLATAFORMA HEROKU

Para realizar a disponibilização da plataforma em ambiente online é necessario que seja realizado o *deploy*. Dessa forma, são necessárias as seguinte etapas:

a) Instalar a extensão *Heroku CLI [19]*;

b) Instalar a extensão *Git SCM [20]*;

c) Possuir uma aplicação Flask;

d) Utilizar ou criar um arquivo de texto com as bibliotecas necessárias para o processamento (*requirements.txt*); e

e) Possuir uma conta na plataforma Heroku.

Na primeira etapa deve ser realizado o cadastro na plataforma Heroku e, em seguida, deve-se selecionar a opção de criar um novo aplicativo, como apresentado nas Figura B.1 e B.2, onde nesta última aparece opção que irá conter o nome da plataforma, como atribuido no exemplo: ime-flask-deploy.

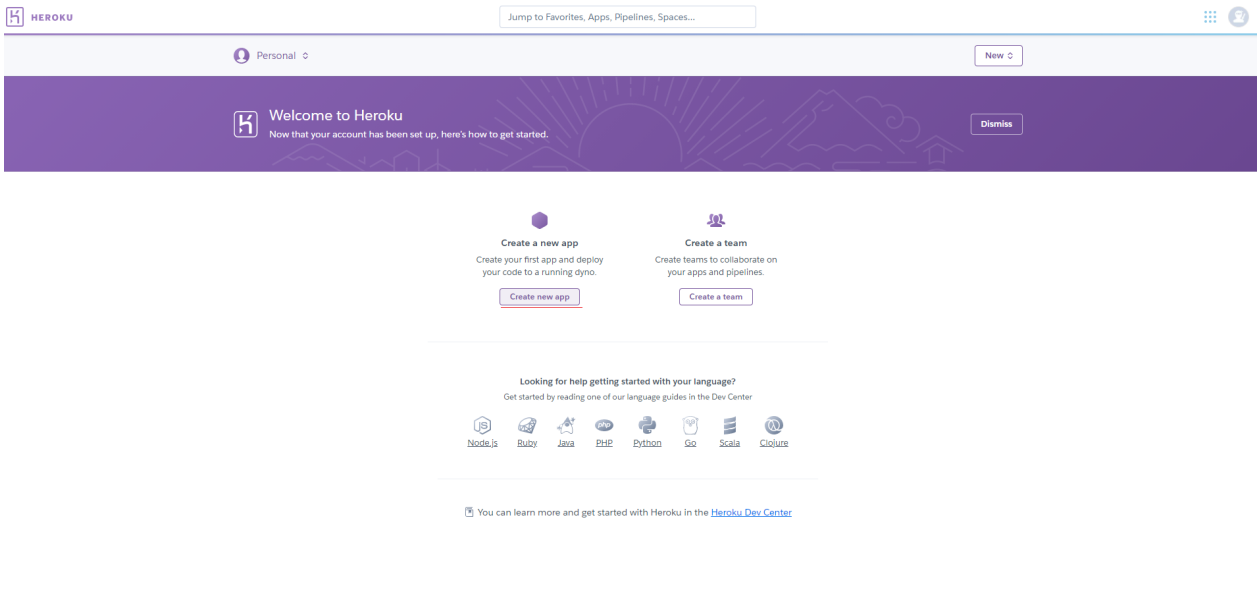


Figura B.1 – Janela para a criação de novo aplicativo.

Logo após ser criado o link que será descarregado o projeto, a página encaminha para uma aba contendo um método para se realizar o deploy via terminal, como apresentado na Figura B.3. Para que essa condição possa ocorrer deve-se ter instalados os pacotes Heroku CLI (Figura B.4) e Git SCM (Figura B.5), onde o primeiro é um pacote para

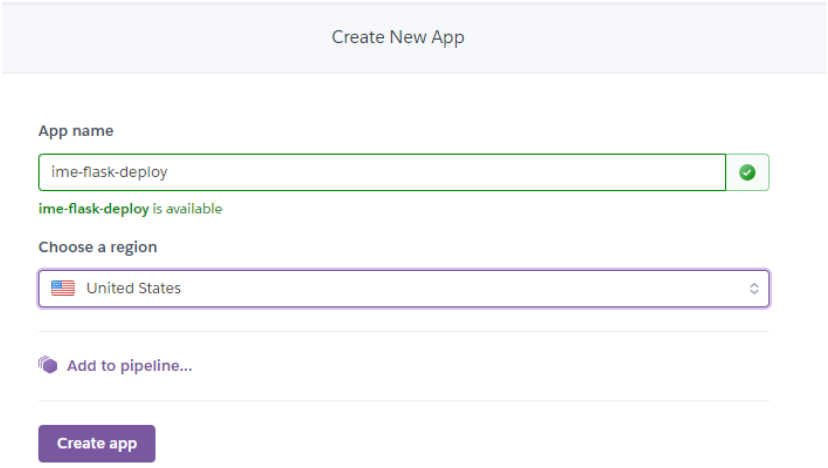


Figura B.2 – Criação de novo aplicativo.

plataformas Git, como o caso do GitHub, e servirá para a conexão com o GitHub. O pacote Heroku CLI, , de forma similar, realizará uma conexão entre o servidor local e a plataforma Heroku através da do prompt de comando.

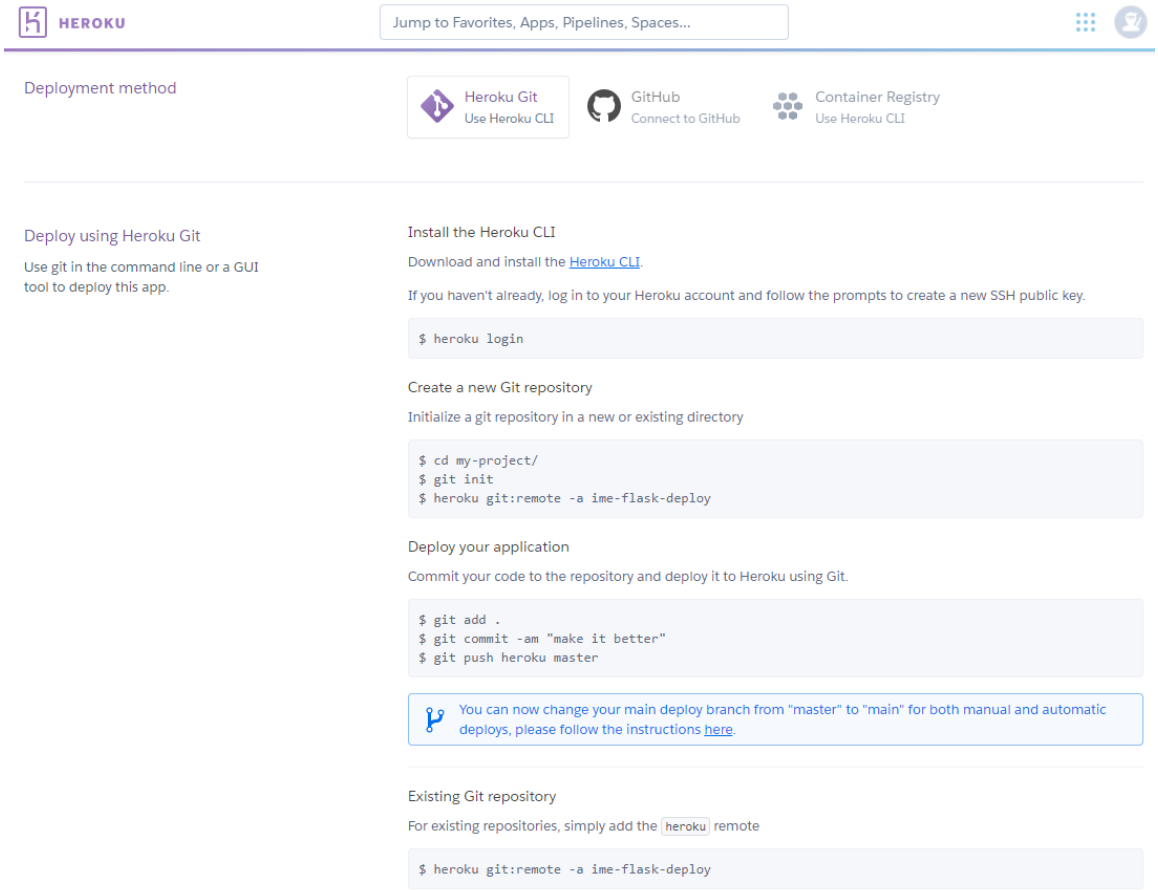


Figura B.3 – Plataforma Heroku.

Por fim, a plataforma é disponibilizada de forma online seguindo os códigos apre-

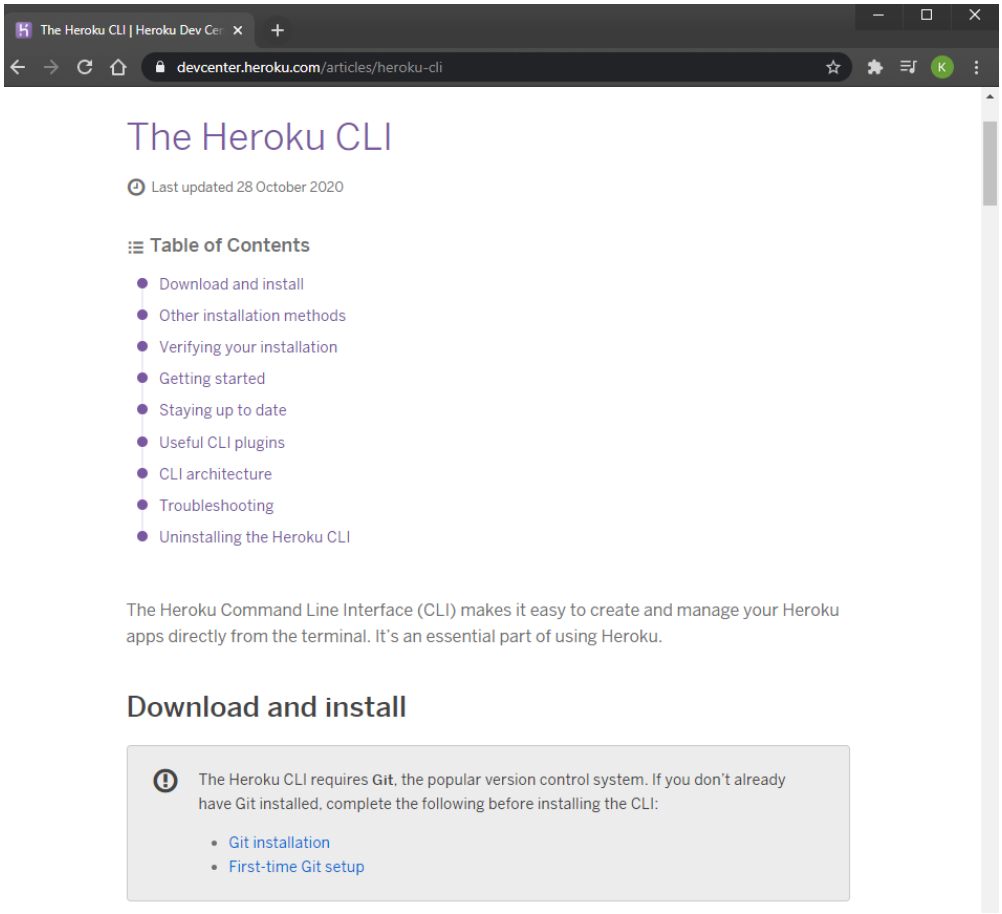


Figura B.4 – Instalação Heroku CLI.

sentados na Figura B.3 e que estão disponíveis na plataforma do Heroku após a criação de uma aplicação web. Existem dois métodos principais citados, um se baseia nos arquivos já disponíveis no GitHub, e dessa maneira será realizada a conexão entre ambos através de uma chave SSH, ou o método que realiza isso diretamente no terminal, seguindo as linhas de códigos apresentadas na Figura B.3.

A partir da instação dessas duas últimas ferramentas apresentadas, ao acessar o ambiente virtual pelo terminal será possível realizar uma conexão com a plataforma Heroku utilizando o comando *heroku login*, que é o comando incial do procedimento apresentado na Figura B.3. Dado que a conexão foi estabelecida será possível realizar o carregamento do site de forma online atráves da plataforma Heroku. A Figura B.6 demonstra a conexão entre a plataforma Heroku e o terminal apresentando uma conexão entre o ambiente virtual e a Heroku.

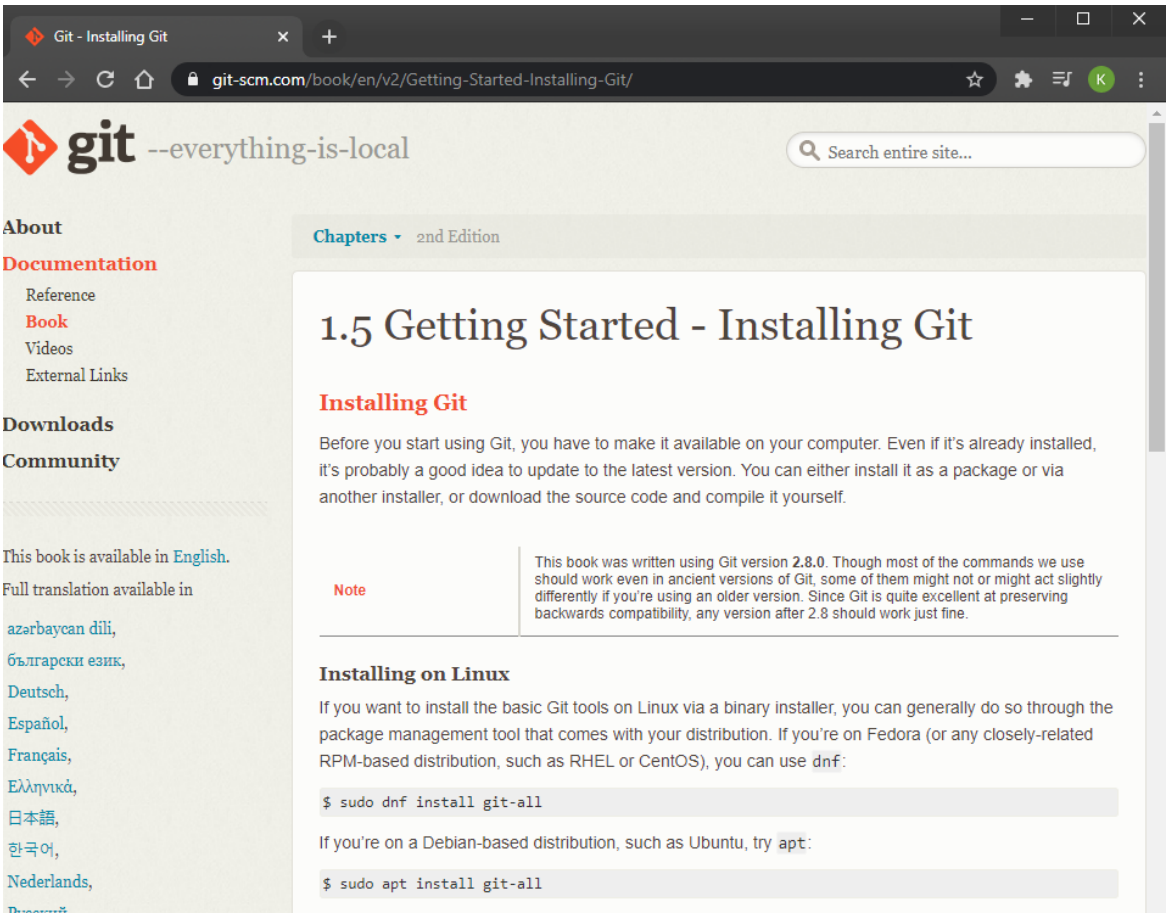


Figura B.5 – Instalação Git SCM.

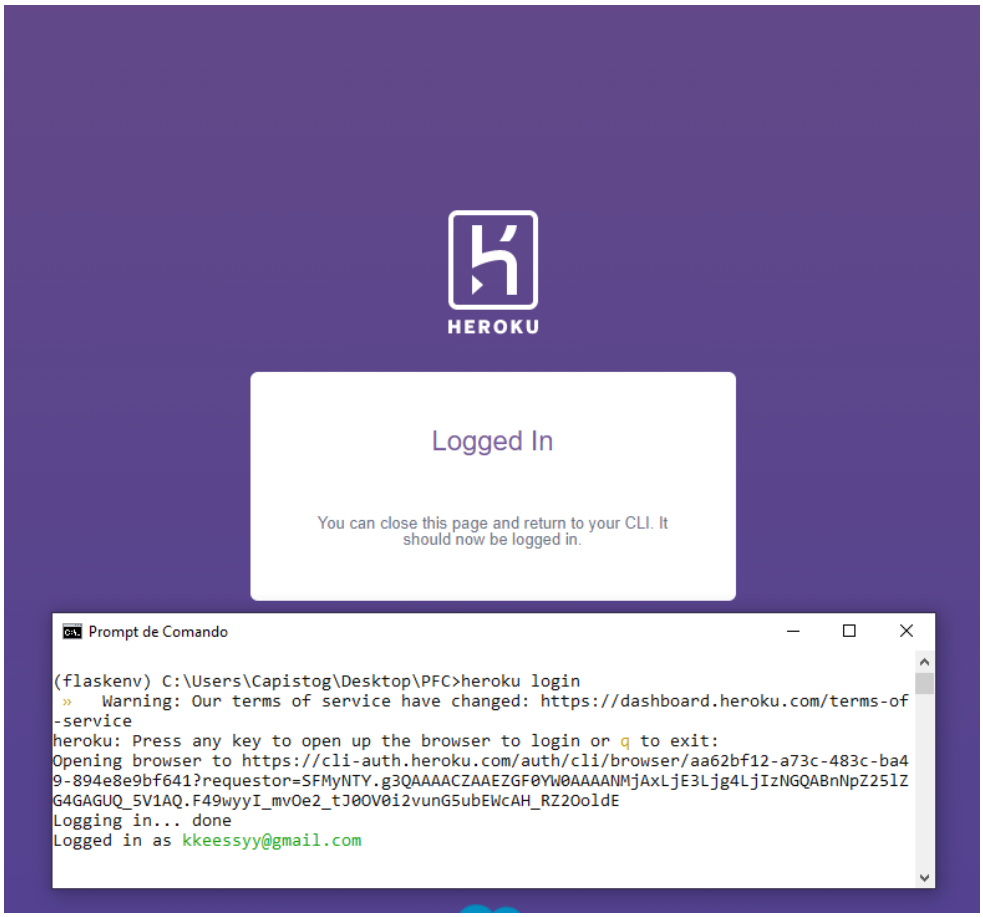
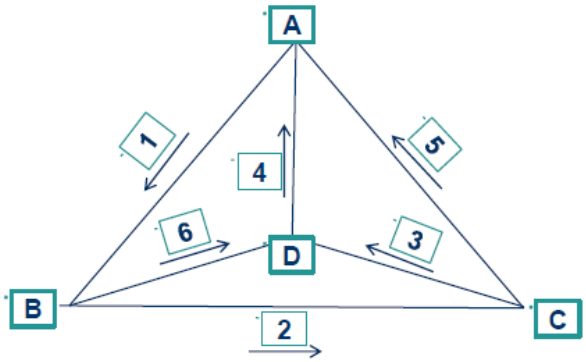


Figura B.6 – Conexão entre o terminal e a plataforma Heroku.

ANEXO A – EXEMPLOS PRESENTES NA PLATAFORMA

Tendo-se a rede de nivelamento abaixo, pode-se utiliza-la como exemplo de ajustamento para os métodos MMQ, ML1 e ML∞.



observação	nós	Valor (mm)
1	AB	1003,1
2	BC	1014,2
3	CD	1003,3
4	DA	-2995,2
5	CA	-1946,0
6	BD	1999,3

- A é RN com h=0 (injunção fixa).
- Todas observações com mesmo desvio padrão.

Figura A.1 – Exemplo presente na plataforma.

A.0.1 Modelo MMQ

Para o Método dos Mínimos Quadrados foi utilizado, inicialmente, o software MatLab para implementar o código, que se encontra na subseção A.0.2, e obtiveram-se os seguintes resultados, em que x representa os valores obitidos dos desníveis de B, C e D, respectivamente, enquanto que v representa os resíduos de cada observação:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad L = \begin{bmatrix} 1003.1 \\ 1014.2 \\ 1003.3 \\ -2995.2 \\ -1946.0 \\ 1999.3 \end{bmatrix}$$

Logo:

$$x = \begin{bmatrix} 0.9835 \\ 1.9753 \\ 2.9855 \end{bmatrix} \times 10^3 \; v = \begin{bmatrix} -19.625 \\ -22.370 \\ 6.925 \\ 9.675 \\ -29.300 \\ 2.7500 \end{bmatrix}$$

A.0.2 Código Exemplo MMQ

```
A=[1 0 0;-1 1 0;0 -1 1;0 0 -1;0 -1 0;-1 0 1];
W=1;
L=[1003.1;1014.2;1003.3;-2995.2;-1946.0;1999.3];
N=A'*A;
U=A'*L;
x=pinv(N)*U;
v=A*x-L;
x
v
```

A.0.3 Modelo ML1

Para o ML1 foi utilizado, inicialmente, o software Octave para rodar o código, que se encontra no ANEXO B [3], e obtiveram-se os seguintes resultados, em que x representa os valores obitidos dos desníveis de B, C e D, respectivamente, enquanto que v representa os resíduos de cada observação [21]:

$$x = \begin{bmatrix} 0.9959 \\ 1.9919 \\ 2.9952 \end{bmatrix} \times 10^3 \; v = \begin{bmatrix} -7.2 \\ -18.2 \\ 0 \\ 0 \\ -45.9 \\ 0 \end{bmatrix}$$

A.0.4 Modelo ML∞

Para o ML∞ utilizamos o software Octave para rodar o código, que se encontra no ANEXO C [3], e obtiveram-se os seguintes resultados, em que x representa os valores obitidos dos desníveis de B, C e D, respectivamente, enquanto que v representa os resíduos de cada observação:

$$x = \begin{bmatrix} 0.97933 \\ 1.96977 \\ 2.97143 \end{bmatrix} \times 10^3 \; v = \begin{bmatrix} -23.7667 \\ -23.7667 \\ -1.6333 \\ 23.7667 \\ -23.7667 \\ -7.2000 \end{bmatrix}$$

ANEXO B – CÓDIGO EXEMPLO ML1

```
m=6; n=3;
L=[1003.1;1014.2;1003.3;-2995.2;-1946.0;1999.3];
A=[1 0 0;-1 1 0;0 -1 1;0 0 -1;0 -1 0;-1 0 1];
x=zeros([n 1]);
v=zeros([m 1]);
p=ones(1,m);
P=diag(p);
I = eye(m);
A1 = [A -A -I I];
c=zeros(1,2*n);
c = cat(2,c,p,p);
ctype=[];
for i=1:m
ctype = [ctype, "S"];
end
vartype=[];
for i=1:(2*(n+m))
vartype = [vartype, "C"];
end
param.dual=1;
param.lpsolver=1;
[xopt, fopt, erro, extra] = glpk (c, A1, L, lb=[], ub=[], ctype, vartype, s=1, param);
for i=1:m
v(i)=xopt(2*n+i)-xopt(2*n+i+m);
end
for i=1:n
x(i)=xopt(i)-xopt(n+i);
```

```
end
x
v
```

ANEXO C – CÓDIGO EXEMPLO ML∞

```
m=6; n=3;
L=[1003.1;1014.2;1003.3;-2995.2;-1946.0;1999.3];
A=[1 0 0;-1 1 0;0 -1 1;0 0 -1;0 -1 0;-1 0 1];
x=zeros([n 1]);
v=zeros([m 1]);
p=ones(1,m);
P=diag(p);
I = eye(m);
O = zeros(m,2*n);
A1 = [A -A -I I zeros(m,1)];
A2 = [O P -P (-1)*ones(m,1)];
A3 = [O -P P (-1)*ones(m,1)];
A = cat(1,A1, A2, A3);
L = cat(1,L,zeros(2*m,1));
c=zeros(1,2*(n+m)+1);c(1,2*(n+m)+1)=1;
ctype=[];
for i=1:(m)
ctype = [ctype, "S"];
end
for i=1:(2*m)
ctype = [ctype, "U"];
end vartype=[];
for i=1:(2*(n+m)+1)
vartype = [vartype, "C"];
end
[xopt, fopt, erro, extra] = glpk (c, A, L, lb=[], ub=[], ctype, vartype, s=1);
for i=1:n
```

```
x(i)=xopt(i)-xopt(n+i);
end
for i=1:m
v(i,1)=xopt(2*n+i)-xopt(2*n+i+m);
end
x
v
```