INSTITUTO FEDERAL DO ESPÍRITO SANTO

CAMPUS VITÓRIA

MESTRADO PROFISSIONAL EM TECNOLOGIAS SUSTENTÁVEIS

**ÁLVARO LUIZ LAGO DE MENEZES**

**OTIMIZAÇÃO DE VIGAS DE CONCRETO ARMADO DE SEÇÃO RETANGULAR E ARMADURA SIMPLES**

Vitória

2019

ÁLVARO LUIZ LAGO DE MENEZES

**OTIMIZAÇÃO DE VIGAS DE CONCRETO ARMADO DE SEÇÃO RETANGULAR E ARMADURA SIMPLES**

Trabalho apresentado durante a disciplina Métodos de Otimização: Matemáticos e Heurísticos do Programa de Pós-graduação em Tecnologias Sustentáveis do Instituto Federal do Espírito Santo.

Área de Concentração: Desenvolvimento de Produtos e Processos Sustentáveis.

Linha de pesquisa: Linha 1 – Otimização de Serviços, Sistemas e Processos.

Professor: Mário Mestria

Vitória

2019

SUMÁRIO

SUMÁRIO 2

1 INTRODUÇÃO 3

2 DESENVOLVIMENTO 3

2.1 FUNÇÃO EXEMPLO 4

2.2 TAREFA 1 5

2.2.1 Tarefa 1 – função a 5

2.2.2 Tarefa 1 – função b 6

2.2.3 Tarefa 1 – função c 7

2.3 TAREFA 2 8

2.3.1 Tarefa 2 – função I 8

2.3.2 Tarefa 2 – função II 9

2.3.3 Tarefa 2 – função III 10

2.3.4 Tarefa 2 – função IV 11

3 CONCLUSÃO 12

REFERÊNCIA BIBLIOGRÁFICA 13

ANEXO I 14

1 INTRODUÇÃO

O concreto é o material construtivo mais utilizado do mundo, devido a possibilidade de modelagem em diversas formas e tamanhos, por ser resistente à água, pelo seu menor custo e pela sua menor produção de poluentes em relação a outros materiais utilizados na construção civil (Pedroso, 2009).

A Engenharia busca um dimensionamento estrutural correto do concreto armado, tendo em vista a preocupação com a relação custo/benefício e com a utilização sustentável do material. Por se tratar de um processo baseado em experiências, tentativas e erros, o auxílio de métodos computacionais é indicado para obtenção de dimensionamentos com o menor custo possível.

O dimensionamento tradicional do concreto armado é realizado através de iterações onde já se indica uma seção transversal, aproximada por experiências e projetos previamente realizados.

De acordo com as restrições do dimensionamento, uma nova seção transversal é adotada ou não, com a finalidade de atender as restrições ou de diminuir os custos. Por se tratar de um processo de tentativa e erro, o processo torna-se demorado se for feito sem a utilização de métodos computacionais.

Durante o trabalho serão apresentadas a função objetivo de custo e as restrições relativas à otimização de vigas de concreto armado bi apoiadas, de seção retangular e armadura simples. Será utilizado o *solver* *fmincon*, presente no *Optimization Toolbox* do Matlab, para otimização do custo da viga, variando os possíveis algoritmos e comparando seus resultados e performance em termos de iterações.

Como a execução das tarefas era repetitiva, foi criada uma função auxiliar que recebe como parâmetros o comprimento L da viga em metros e executa o *solver fmincon* variando os algoritmos. O código da função está disponível no Anexo I.

Para a execução das tarefas, foi utilizado o Matlab R2018b, versão 9.5.0 em ambiente *Windows*.

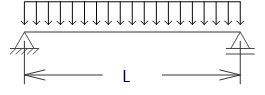
O código e os resultados obtidos nas execuções das tarefas também estão disponíveis no *Github* em https://github.com/alvarollmenezes/ifes-otimizacao-final.

2 conceitos

O concreto armado é a associação entre o concreto simples e a armadura passiva convenientemente colocada, de tal maneira que ambos resistam aos esforços solicitantes (Carvalho e Figueiredo, 2014).

O trabalho busca dimensionar vigas biapoiadas com carregamentos distribuídos e comprimento L, conforme a Figura 1.

Figura 1 – Viga biapoiada com carregamento distribuído



Fonte: Acervo pessoal

Vigas de armadura simples possuem dois tipos de armadura: armadura longitudinal, responsável pela resistência ao momento fletor e armadura transversal, responsável pela resistência ao esforço cortante.

Portanto, na formação do custo, que será utilizado como função objetivo na otimização, deve ser calculado o custo do concreto e do aço das armaduras. Também deve ser levado em conta o custo da fôrma de madeira utilizada na construção da viga.

Para o dimensionamento da seção retangular, com vigas de armadura simples, foram admitidas as hipóteses de acordo com a NBR 6118 (2004). As restrições consideradas nas vigas são: a armadura máxima, a capacidade de resistência à flexão, a capacidade de resistência ao cisalhamento, a flecha limite e normas de segurança à instabilidade da viga.

3 DESENVOLVIMENTO

3.1 FUNÇÃO OBJETIVO

A função objetivo é de minimizar o custo, os custos do concreto, aço e forma estão descritos abaixo, retirados da tabela SINAPI da Caixa Econômica Federal de 2017, referentes ao estado do Espírito Santo.

Os valores de cada material são calculados abaixo:

* Valor total do concreto:

Onde: = Valor total do concreto;

= área da seção transversal do concreto;

= Custo do concreto.

* Valor total do aço:

Onde: = Valor total do aço;

= área da seção longitudinal do aço;

= área da seção transversal do aço;

= Peso específico do aço;

= custo do aço.

* Valor total da forma:

Onde: = Valor total da forma;

= altura da viga;

= base da viga;

= custo da forma.

Assim, tem-se o custo total:

3.2 FUNÇÕES DE RESTRIÇÃO

As restrições consideradas nas vigas são: a armadura máxima, a capacidade de resistência à flexão, a capacidade de resistência ao cisalhamento, a flecha limite e normas de segurança à instabilidade da viga. Seus cálculos são descritos a seguir.

3.2.1 Capacidade de resistência à flexão

Entendidas as tarefas do laboratório, foi percebida a necessidade da execução de tarefas similares, variando-se as funções objetivo e o ponto inicial e utilizando-se os três algoritmos possíveis para os métodos de escolha de direção de busca. Sendo assim, foi criado o arquivo *executar.m* (Anexo I), contendo uma função que recebe como parâmetros a função a ser minimizada, o ponto inicial e retorna um *struct* com os resultados dos métodos *bfgs, dfp* e *steepdesc*.

Durante as execuções, foi utilizada a opção *Display* com o valor *iter*, desta forma a cada iteração, o *solver* exibe uma saída com as informações dos valores calculados nesta iteração.

Nenhuma opção além de *Algorithm, hessupdate* e *Display* foi alterada, mantendo os padrões do Matlab no restante. Destaque para as opções *MaxIterations* que determina o número máximo de iterações permitido e tem como padrão 400 e *MaxFunctionsEvaluation* que determina o número máximo permitido de avaliações da função a ser minimizada, tendo como padrão 100 \* número de variáveis. Este último foi fator limitador em algumas execuções durante os testes. Todos os valores padrão utilizados pelo *solver* estão disponíveis na documentação do Matlab.

2.1 FUNÇÃO EXEMPLO

A primeira execução foi feita com a função f(x1,x2)=3(x1)2 + 2x1x2 + (x2)2, do texto explicativo, para isso foi criado o arquivo *fexemplo.m* (Anexo I), para ser utilizado na função *executar* com o ponto inicial em [1, 1], código executado:

>> [ex\_bfgs, ex\_dfp, ex\_steepdesc] = executar(@fexemplo, [1,1]);

A saída completa desta tarefa está disponível no arquivo *output\_fexemplo.txt* no *Github.*

* **Resultados com o método BFGS**

Mínimo local encontrado em 8 iterações.

x1 = 2.5408e-07, x2 = -2.0293e-07.

fval = 1.3173e-13

* **Resultados com o método DFP**

Mínimo local encontrado em 8 iterações.

x1 = 2.5408e-07, x2 = -2.0293e-07.

fval = 1.3173e-13

* **Resultados com o método steepdesc**

Mínimo local encontrado em 22 iterações.

x1 = -2.8500e-06, x2 = 5.6879e-06.

fval = 2.4299e-11

2.2 TAREFA 1

As execuções das minimizações desta tarefa estão descritas a seguir.

2.2.1 Tarefa 1 – função a

A execução desta tarefa foi feita com a função f(x1,x2) = 2(x1 - 2x2)2 + ½(x2 -(x1)2 )2, para isso foi criado o arquivo *ft1a.m* (Anexo I), para ser utilizado na função *executar* com o ponto inicial em [2, 6], código executado:

>> [ft1a\_bfgs, ft1a\_dfp, ft1a\_steepdesc] = executar(@ft1a, [2,6]);

A saída completa desta tarefa está disponível no arquivo *output\_ft1a.txt* no *Github.*

* **Resultados com o método BFGS**

Mínimo local encontrado em 15 iterações.

x1 = 0.5000, x2 = 0.2500.

fval = 2.1236e-12

* **Resultados com o método DFP**

Mínimo local encontrado em 15 iterações.

x1 = 0.5000, x2 = 0.2500.

fval = 2.1236e-12

* **Resultados com o método steepdesc**

Mínimo local não encontrado, foram executadas 31 iterações e o *solver* parou prematuramente por extrapolar o número máximo de avaliações da função (200).

x1 = 0.5047, x2 = 0.2525.

fval = 2.6207e-06

2.2.2 Tarefa 1 – função b

A execução desta tarefa foi feita com a função f(x1,x2) = ½( (x1-1)2 + (x2-(x1)2 )2 ), para isso foi criado o arquivo *ft1b.m* (Anexo I), para ser utilizado na função *executar* com o ponto inicial em [-1, 0], código executado:

>> [ft1b\_bfgs, ft1b\_dfp, ft1b\_steepdesc] = executar(@ft1b, [-1,0]);

A saída completa desta tarefa está disponível no arquivo *output\_ft1b.txt* no *Github.*

* **Resultados com o método BFGS**

Mínimo local encontrado em 10 iterações.

x1 = 1.0000, x2 = 1.0000.

fval = 5.6093e-13

* **Resultados com o método DFP**

Mínimo local encontrado em 10 iterações.

x1 = 1.0000, x2 = 1.0000.

fval = 5.6093e-13

* **Resultados com o método steepdesc**

Mínimo local não encontrado, foram executadas 35 iterações e o *solver* parou prematuramente por extrapolar o número máximo de avaliações da função (200).

x1 = 0.9780, x2 = 0.9430.

fval = 3.3330e-04

2.2.3 Tarefa 1 – função c

A execução desta tarefa foi feita com a função f(x1,x2) = -24x1x2 + (x1)2 x2 + x1(x2)2, para isso foi criado o arquivo *ft1c.m* (Anexo I), para ser utilizado na função *executar* com o ponto inicial em [7, 9], código executado:

>> [ft1c\_bfgs, ft1c\_dfp, ft1c\_steepdesc] = executar(@ft1c, [7,9]);

A saída completa desta tarefa está disponível no arquivo *output\_ft1c.txt* no *Github.*

* **Resultados com o método BFGS**

Mínimo local encontrado em 5 iterações.

x1 = 8.0000, x2 = 8.0000.

fval = -512

* **Resultados com o método DFP**

Mínimo local encontrado em 5 iterações.

x1 = 8.0000, x2 = 8.0000.

fval = -512

* **Resultados com o método steepdesc**

Mínimo local possível encontrado em 11 iterações. O *solver* parou a execução pois o passo da iteração era menor que a tolerância configurada.

x1 = 8.0000, x2 = 8.0000.

fval = -512

2.3 TAREFA 2

Para a tarefa 2, além da execução do *solver* utilizando os três algoritmos de métodos de escolha de direção de busca, foi solicitado o preenchimento da tabela abaixo:

Tabela 1 Número de iterações de cada algoritmo na Tarefa 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Número de iterações dos algoritmos** | | |
| **Problema** | **Ponto Inicial** | **bfgs** | **dfp** | **steepdesc** |
| I | (-2, 1) | 2 | 2 | 2 |
| II | (-1, 1) | 7 | 7 | 2 |
| III | (3, -1) | 13 | 13 | 14 |
| IV | (-7, 8) | 48 | 48 | 12 |

2.3.1 Tarefa 2 – função I

A execução desta tarefa foi feita com a função f(x1,x2) = (x1)2 + (x2)2 -1, para isso foi criado o arquivo *ft2i.m* (Anexo I), para ser utilizado na função *executar* com o ponto inicial em [-2, 1], código executado:

>> [ft2i\_bfgs, ft2i\_dfp, ft2i\_steepdesc] = executar(@ft2i, [-2,1]);

A saída completa desta tarefa está disponível no arquivo *output\_ft2i.txt* no *Github.*

* **Resultados com o método BFGS**

Mínimo local encontrado em 2 iterações.

x1 = -1.1102e-16, x2 = -2.2352e-08.

fval = -1.0000

* **Resultados com o método DFP**

Mínimo local encontrado em 2 iterações.

x1 = -1.1102e-16, x2 = -2.2352e-08.

fval = -1.0000

* **Resultados com o método steepdesc**

Mínimo local encontrado em 2 iterações.

x1 = 7.4506e-09, x2 = -1.1176e-08.

fval = -1.0000

2.3.2 Tarefa 2 – função II

A execução desta tarefa foi feita com a função f(x1,x2) = (x1)4 + (x1)2 + (x2)2, para isso foi criado o arquivo *ft2ii.m* (Anexo I), para ser utilizado na função *executar* com o ponto inicial em [-1, 1], código executado:

>> [ft2ii\_bfgs, ft2ii\_dfp, ft2ii\_steepdesc] = executar(@ft2ii, [-1,1]);

A saída completa desta tarefa está disponível no arquivo *output\_ft2ii.txt* no *Github.*

* **Resultados com o método BFGS**

Mínimo local encontrado em 7 iterações.

x1 = -4.3061e-07, x2 = -8.8616e-07.

fval = 9.7071e-13

* **Resultados com o método DFP**

Mínimo local encontrado em 7 iterações.

x1 = -4.3061e-07, x2 = -8.8616e-07.

fval = 9.7071e-13

* **Resultados com o método steepdesc**

Mínimo local encontrado em 2 iterações.

x1 = 7.4506e-09, x2 = 1.0348e-10.

fval = 5.5522e-17

2.3.3 Tarefa 2 – função III

A execução desta tarefa foi feita com a função f(x1,x2) = (-13 + x1 + 5(x2)2 - (x2)3 - 2x2)2 + (-29 + x1 + (x2)3 + (x2)2 - 14x2)2, para isso foi criado o arquivo *ft2iii.m* (Anexo I), para ser utilizado na função *executar* com o ponto inicial em [3, -1], código executado:

>> [ft2iii\_bfgs, ft2iii\_dfp, ft2iii\_steepdesc] = executar(@ft2iii, [3,-1]);

A saída completa desta tarefa está disponível no arquivo *output\_ft2iii.txt* no *Github.*

* **Resultados com o método BFGS**

Mínimo local encontrado em 13 iterações.

x1 = 11.4128, x2 = -0.8968.

fval = 48.9843

* **Resultados com o método DFP**

Mínimo local encontrado em 13 iterações.

x1 = 11.4128, x2 = -0.8968.

fval = 48.9843

* **Resultados com o método steepdesc**

Mínimo local não encontrado, foram executadas 14 iterações e o solver parou prematuramente por extrapolar o número máximo de avaliações da função (200).

x1 = 3.2866, x2 = -1.3455.

fval = 76.2182

2.3.4 Tarefa 2 – função IV

A execução desta tarefa foi feita com a função f(x1,x2) = 100(x2 – (x1)2 )2 + (1 - x1)2, para isso foi criado o arquivo *ft2iv.m* (Anexo I), para ser utilizado na função *executar* com o ponto inicial em [-7, 8], código executado:

>> [ft2iv\_bfgs, ft2iv\_dfp, ft2iv\_steepdesc] = executar(@ft2iv, [-7,8]);

A saída completa desta tarefa está disponível no arquivo *output\_ft2iv.txt* no *Github.*

* **Resultados com o método BFGS**

Mínimo local encontrado em 48 iterações.

x1 = 0.9199, x2 = 0.8457.

fval = 0.0064

* **Resultados com o método DFP**

Mínimo local encontrado em 48 iterações.

x1 = 0.9199, x2 = 0.8457.

fval = 0.0064

* **Resultados com o método steepdesc**

Mínimo local não encontrado, foram executadas 12 iterações e o solver parou prematuramente por extrapolar o número máximo de avaliações da função (200).

x1 = 2.9036, x2 = 8.4330.

fval = 3.6241

3 CONCLUSÃO

Foi possível perceber uma tendência a convergir mais rapidamente utilizando-se os algoritmos *bfgs* e *dfp,* para o método de escolha de direção de busca. Porém há casos e que o método *steepdesc* se comporta não apenas convergindo mais rapidamente, como trazendo uma solução melhor, como foi o caso da função II da tarefa 2.

REFERÊNCIA

BELLOC, M.; VELASTIN, S.; FERNANDEZ, R.; JARA, M. Detection of People Boarding/Alighting a Metropolitan Train using Computer Vision. 9th International Conference on Pattern Recognition Systems (ICPRS 2018), pp. 5-11. 2018.

BRASIL. Ministério das Cidades. Política Nacional de Mobilidade Urbana Sustentável: Princípios e Diretrizes. Brasília. 2004.

CHATO, P.; CHIPANTASI, D.; VELASCO, N.; REA, S.; HALLO, V.; CONSTANTE, P. Image processing and artificial neural network for counting people inside public transport. IEEE Third Ecuador Technical Chapters Meeting (ETCM), pp. 1-5. 2018.

COCCHIA, A. Smart and digital city: a systematic literature review. Smart City: How to Create Public and Economic Value with High Technology in Urban Space, 13-43. 2014.

GUERRA, A. L.; BARBOSA, H. M.; OLIVEIRA, L. K. Estimativa de Matriz Origem/Destino Utilizando Dados do Sistema de Bilhetagem Eletrônica: Proposta Metodológica. Transportes, [s.l.], v 22, n. 3, p. 26-38, 2014.

PRADO, A.L.; DA COSTA, E.M.; FURLANI, T.Z.; YIGITCANLAR, T. Smartness that matters: Towards a comprehensive and human-centred characterisation of smart cities. Journal of Open Innovation Technology, Market and Complexity. 2016.

MOREIRA-MATIAS L.; MENDES-MOREIRA J.; DE SOUSA, J.; GAMA, J. Improving Mass Transit Operations by Using AVL-Based Systems: A Survey. IEEE Transactions on Intelligent Transportation Svstems, vol. 16, no. 4, pp. 1636-1653, 2015.

MOREIRA, Mauricio Renato Pina; SCHREINER, Sideney. Pesquisas de Origem e Destino na Região Metropolitana do Recife. In: XXXI Congresso Nacional de Pesquisa em Transportes da ANPET. Recife, 2017. p. 2039-2046.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You Only Look Once: Unified, Real-Time Object Detection. arXiv. 2016.

REDMON, J.; FARHADI, A. YOLOv3: An Incremental Improvement. arXiv. 2018.

SARAIVA, M. A Cidade e o Tráfego Uma Abordagem Estratégica. Recife. Ed. Universitária da UFPE. 2000.

ANEXO I

Arquivo executar.m:

function [bfgs, dfp, steep] = executar(func, x0)

options\_bfgs = optimoptions(@fminunc, 'Algorithm', 'quasi-newton',

'hessupdate', 'bfgs', 'Display', 'iter');

options\_dfp = optimoptions(@fminunc, 'Algorithm', 'quasi-newton',

'hessupdate', 'bfgs', 'Display', 'iter');

options\_steep = optimoptions(@fminunc, 'Algorithm', 'quasi-newton',

'hessupdate', 'steepdesc', 'Display', 'iter');

disp('Executando ''fminunc'' com o método ''bfgs''');

[bfgs.x, bfgs.fval, bfgs.exitflag, bfgs.output] = ...

fminunc(func, x0, options\_bfgs);

bfgs

bfgs\_output = bfgs.output

disp('Executando ''fminunc'' com o método ''dfp''');

[dfp.x, dfp.fval, dfp.exitflag, dfp.output] = ...

fminunc(func, x0, options\_dfp);

dfp

dfp\_output = dfp.output

disp('Executando ''fminunc'' com o método ''steepdesc''');

[steep.x, steep.fval, steep.exitflag, steep.output] = ...

fminunc(func, x0, options\_steep);

steep

steep\_output = steep.output

end

Arquivo fexemplo.m:

function f = fexemplo(x)

f = 3\*x(1)^2 + 2\*x(1)\*x(2) + x(2)^2;

end

Arquivo ft1a.m:

function f = ft1a(x)

f = 2 \* (x(1) - 2\*x(2))^2 + 1/2 \* (x(2) - x(1)^2)^2;

end

Arquivo ft1b.m:

function f = ft1b(x)

f = 1/2 \* ((x(1) - 1)^2 + (x(2) - x(1)^2)^2);

end

Arquivo ft1c.m:

function f = ft1c(x)

f = -24\*x(1)\*x(2) + x(1)^2\*x(2) + x(1)\*x(2)^2;

end

Arquivo ft2i.m:

function f = ft2i(x)

f = x(1)^2 + x(2)^2 - 1;

end

Arquivo ft2ii.m:

function f = ft2ii(x)

f = x(1)^4 + x(1)^2 + x(2)^2;

end

Arquivo ft2iii.m:

function f = ft2iii(x)

f = (-13 + x(1) + 5\*x(2)^2 - x(2)^3 - 2\*x(2))^2 + ...

(-29 + x(1) + x(2)^3 + x(2)^2 - 14\*x(2))^2;

end

Arquivo ft2iv.m:

function f = ft2iv(x)

f = 100 \* (x(2) - x(1)^2)^2 + (1 - x(1))^2;

end