QUADRADOS MÍNIMOS NÃO–LINEARES E APLICAÇÕES EM DEFORMAÇÃO DE OBJETOS 2D *

Matheus Souza[†] Maria A. Diniz-Ehrhardt[‡]

Resumo

Problemas de quadrados mínimos não–lineares surgem em diversas aplicações, principalmente em ajustes de curvas, mas podem ocorrer no caso mais geral em um sistema de equações não–lineares com mais equações do que incógnitas (sistemas sobredeterminados). Para resolver problemas de quadrados mínimos não–lineares, usaremos o Método de Gauss–Newton, que será analisado tanto em suas propriedades teóricas, quanto na parte computacional. Aplicações do método foram feitas ao estudo de deformações de objetos 2D.

Abstract

Nonlinear least squares problems arise in many applications, especially in curve fitting, but they may occur in a more general case as a overdetermined system of nonlinear equations. To solve these problems we use the Gauss–Newton Method, analysing not only its theoretical properties but also its computational performance. We apply these concepts to the study of 2D-Shape Deformation problems.

*Este projeto contou com o apoio do CNPq e da FAPESP.

[†]FEEC-Unicamp. e-mail: ra071855@fee.unicamp.br

[‡]DMA-IMECC-Unicamp. e-mail: cheti@ime.unicamp.br

1 Introdução

Problemas de ajuste de curvas surgem em diversas áreas do conhecimento humano, desde os estudos populacionais na demografia, até experimentos físicos e análise de dados na engenharia. Portanto, desenvolver métodos computacionais robustos e eficientes para resolver esse tipo de problema é um dos principais objetivos da matemática aplicada e da computação científica, tendo em vista que os dados resultantes devem ser confiáveis. Problemas como esses podem ser modelados por quadrados mínimos não–lineares, doravante denotados por QMNL.

Uma das primeiras táticas a serem utilizadas para resolver problemas de QMNL é buscar a linearização, ou seja, buscar uma mudança de variáveis e uma manipulação algébrica para que o problema seja possível de ser resolvido com quadrados mínimos lineares, cuja resolução é normalmente mais simples. Porém, quando essa abordagem do problema é usada, surgem complicações, pois o resíduo obtido pode não ser de fato o ótimo, além de a aplicabilidade desse método ser muito restrita, já que nem todos os problemas podem ser linearizados.

Assim, uma forma mais desenvolvida para lidar com estruturas desse tipo é tratá-las como problemas de minimização, onde a função a ser minimizada é a função resíduo. Portanto, para resolver esse problema de otimização, o Método de Newton poderia ser utilizado. Entretanto, com uma pequena modificação, obtemos um método iterativo mais simples e mais específico, que é o Método de Gauss–Newton, atualmente o método mais difundido para a resolução de problemas de QMNL.

Este relatório está organizado como se segue. Na Seção 2 descrevemos o problema de quadrados mínimos lineares, tanto na parte computacional quanto na teoria. Já na Seção 3, analisamos o Método de Newton e sua aplicação para resolver problemas de otimização. Na seção 4, o problema de quadrados mínimos não–lineares é apresentado e é apresentada sua solução pelo Método de Newton, que não é satisfatória. Assim, na Seção 5, buscamos uma modificação do Método de Newton que aproveite a estrutura do problema de QMNL. Essa modificação é o Método de Gauss–Newton, que é mais eficiente que o algoritmo original e apresenta resultados de convergência semelhantes. Finalmente, nas Seções 6 e 7 discutimos os experimentos computacionais com problemas de QMNL: ajustes de curvas e deformação de objetos 2D.

Em todo o trabalho utilizaremos conceitos e resultados básicos de otimização irrestrita, como soluções de problemas de otimização e elementos de análise convexa. O leitor pode buscar esses resultados nas referências [2] e [4].

2 Quadrados Mínimos Lineares

Sistemas lineares são estruturas muito comuns e surgem, por exemplo, em problemas de ajuste de curvas, de análise de circuitos elétricos, de resolução numérica de

equações diferenciais ordinárias, de interpolação polinomial, entre outros. Exemplos com essas e outras aplicações podem ser obtidos em [3] e [6]. O mais importante, quando nos deparamos com um sistema linear, é verificar se ele possui solução e, se ele possuir, saber se é única. Sistemas sobredeterminados são sistemas que possuem mais equações que incógnitas, de onde podemos perceber que é muito difícil que tenham solução, o que de fato não ocorre na maioria dos casos. Problemas que envolvem sistemas lineares desse tipo são chamados de *problemas de quadrados mínimos lineares*.

Nesta seção, faremos uma análise dos principais tópicos desse problema, que nos será muito útil para entendermos o caso mais geral, que trata de sistemas não–lineares.

Considere o sistema sobredeterminado

$$Ax = b, \tag{1}$$

onde $A \in I\!R^{m\times n}$, $b \in I\!R^m$, $m > n$. Seja $r(x) = Ax - b$ a *função resíduo* do problema (1). Nosso objetivo é encontrar x^* tal que a norma-2 da função resíduo seja mínima, ou seja, queremos resolver o seguinte problema:

$$Minimizar\ ||r(x)||_2 = ||Ax - b||_2, \tag{2}$$

onde $x \in I\!R^n$. Note que (2) é um problema de otimização e queremos buscar x^* que é minimizador local de $||r(x)||_2$. Primeiramente, notemos que minimizar $||r(x)||_2$ é o mesmo que minimizar seu quadrado, ou seja, podemos minimizar a função

$$f(x) = r(x)^t r(x) = ||r(x)||_2^2.$$

Portanto, primeiramente analisemos as *Condições Necessárias de Primeira Ordem* (`CNI`): se x^* é um minimizador, então $\nabla f(x^*) = 0$. Assim, para termos candidatos a minimizador de f, devemos resolver o sistema $\nabla f(x) = 0$. Agora,

$$f(x) = r(x)^t r(x) = (Ax - b)^t(Ax - b) = x^t A^t Ax - x^t A^t b - b^t Ax + b^t b,$$

de onde podemos escrever que:

$$f(x) = x^t(A^t A)x - 2x^t A^t b + b^t b \rightarrow \nabla f(x) = 2A^t Ax - 2A^t b.$$

Assim, para satisfazer as `CNI`, devemos ter $2A^t Ax - 2A^t b = 0$, donde

$$A^t Ax = A^t b, \tag{3}$$

que é chamado de **sistema normal**. Qualquer minimizador de f deve satisfazer (3). Agora, devemos analisar a Hessiana de f, para as condições de segunda ordem.

Facilmente percebemos que $\nabla^2 f(x) = A^t A$, que é semidefinida positiva, para qualquer matriz A. De fato, seja $x \in R^n$, então $x^t(A^tA)x = (Ax)^t(Ax) = ||Ax||_2^2 \geq 0, \forall x$. Então f é uma função convexa e o ponto x^* que é solução do sistema normal é minimizador global de f e, portanto, do resíduo. O ponto x^* é chamado **solução de quadrados mínimos** para o problema (1).

Geometricamente, estamos buscando o ponto pertencente ao espaço-imagem de A que mais se aproxima do vetor b. Para tanto, encontramos a projeção ortogonal de b em $\mathcal{I}(A)$, e encontramos $x^* \in I\!R^n$ tal que Ax^* seja essa projeção.

Problemas de quadrados mínimos lineares podem ser resolvidos de diversas formas. Se A tem posto completo, então a matriz A^tA é definida positiva e, portanto, pode ser resolvido eficientemente através da fatoração de Cholesky. Algumas vezes essa solução não nos é viável, quer seja o caso em que não gostaríamos de calcular o produto A^tA, quer seja o caso em que A não tenha posto completo. Nesses casos, podemos resolver esse problema através de outras fatorações matriciais como a fatoração ortogonal ou a decomposição em valores singulares. Uma análise desses casos é feita em [6], por exemplo.

3 Método de Newton

Antes de prosseguirmos aos problemas de QMNL, descrevemos a idéia básica dos Métodos de Newton para Sistemas Não–Lineares e para Otimização, que serão muito importantes para a nossa análise posterior.

Primeiramente, temos por objetivo resolver o seguinte problema:

$$\textit{Encontrar } x^* \in I\!R^n \textit{ tal que } F(x^*) = 0, \tag{4}$$

onde $F : I\!R^n \to I\!R^m$ e F é continuamente diferenciável. A solução proposta pelo Método de Newton é construir um modelo afim nas proximidades de um ponto x^0 e calcular a raiz desse modelo, que será usado como o novo ponto central para outro modelo afim e assim sucessivamente. Com efeito, seja M_c o modelo afim construído para a função, centrado em um ponto $x^0 \in I\!R^n$, que é dado por:

$$M_c(x^0 + p) = F(x^0) + J(x^0) * p,$$

onde $J(x)$ é a **matriz Jacobiana** de F e $p \in I\!R^n$ é uma pequena perturbação. Queremos obter s tal que $M_c(x^0 + s) = 0$. Assim, temos:

$$M_c(x^0 + s) = 0 \iff F(x^0) + J(x^0) * s = 0$$

$$J(x^0) * s = -F(x^0)$$

$$s = -[J(x^0)]^{-1} F(x^0),$$

que é o passo do *Método de Newton para Sistemas Não–Lineares*. Então, podemos construir um método iterativo como o descrito em [3].

Agora, consideremos outro problema. Queremos encontrar soluções para problemas de otimização, usando um raciocínio similar ao feito anteriormente. Vamos primeiramente pensar no caso mais simples:

$$Minimizar\ f(x), \tag{5}$$

onde $f : I\!R^n \to I\!R$. Podemos tentar resolver o seguinte problema, supondo $f \in C^1$, usando a condição necessária de primeira ordem:

$$Encontrar\ x^*\ tal\ que\ \nabla f(x^*) = 0. \tag{6}$$

O principal problema de se resolver (6) no lugar de (5) é que apenas as condições necessárias não garantem que o ponto estacionário que obtermos será de fato um minimizador local da função, já que podemos obter pontos de máximo e de sela também. Assim, precisamos de um método mais poderoso que o explicitado acima para resolver esse problema.

A partir desse problema surge a idéia de Newton, que é muito simples: construir um modelo que aproxima a função localmente e calcular o ponto em que o modelo é mínimo, realizando esse mesmo procedimento até encontrar uma boa aproximação. Esse modelo foi construído a partir de uma função em $I\!R^n$ que é fácil de minimizar: uma **quadrática**. Chamando esse modelo de ϕ, temos:

$$\phi(x) = f(x^k) + \nabla f(x^k)^t(x - x^k) + \frac{1}{2}(x - x^k)^t\nabla^2 f(x^k)(x - x^k). \tag{7}$$

O modelo (7) é obtido pela aproximação da função por um *polinômio de Taylor* de segundo grau, centrado em x^k. Assim, como queremos encontrar x^{k+1} que minimiza a função ϕ, temos da `CNI`:

$$\nabla\phi(x^{k+1}) = 0.$$

Como $\nabla\phi(x^{k+1}) = \nabla^2 f(x^k)(x^{k+1} - x^k) + \nabla f(x^k)$, temos:

$$\nabla^2 f(x^k)(x^{k+1} - x^k) = -\nabla f(x^k). \tag{8}$$

Assim, chamando de $d^k = x^{k+1} - x^k$, podemos determinar d^k através da resolução do sistema linear:

$$\nabla^2 f(x^k)d^k = -\nabla f(x^k). \tag{9}$$

Se a Hessiana é definida positiva, podemos escrever:

$$d^k = -[\nabla^2 f(x^k)]^{-1}\nabla f(x^k). \tag{10}$$

A direção d^k dada em (10) é a direção usada pelo método de Newton para resolver o problema (6), e ela é de descida se a Hessiana for definida positiva. Se a Hessiana não for definida positiva, o sistema linear dado em (9) pode não ter solução. Portanto a hipótese da positividade da Hessiana é importante para a garantia da convergência do método. O teorema a seguir garante a **convergência local** do método, ou seja, a convergência em uma vizinhança do ponto ótimo.

Teorema 1 *(Convergência Local do Método de Newton)*
Seja $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^3$. Seja x^* um minimizador local de f em $\mathbb{R}^n$, tal que $\nabla^2 f(x^*)$ é definida positiva. Então, existe $\varepsilon > 0$ tal que se $x^0 \in \mathbb{R}^n$ é tal que $||x^0 - x^*|| < \varepsilon$, a seqüência x^k gerada pelo método de Newton verifica:
(i) $\nabla^2 f(x^k)$ é definida positiva, para todo $k \in \mathbb{N}$;
(ii) $\lim_{k\to\infty} x^k = x^*$;
(iii) Existe $c > 0$ tal que $||x^{k+1} - x^*|| \leq c||x^k - x^*||^2$ para todo $k \in \mathbb{N}$.

Para a demonstração, ver [1]. A condição *(iii)* do teorema acima garante que a **ordem de convergência** do método de Newton é *quadrática*. Em geral, tomando-se $e_k = ||x^k - x^*||$ como o erro da k-ésima iteração, se tivermos $e_{k+1} \leq ce_k^p$, com $c > 0$, dizemos que a ordem de convergência de x^k é de pelo menos p. Quando $p = 1$ e $c \in (0, 1)$ dizemos que a convergência é linear, e quando $p = 2$, dizemos que a ordem de convergência é quadrática.

Analisando criteriosamente o teorema acima, percebemos que o método, quando suficientemente próximo do minimizador local, converge rapidamente. Agora, se o ponto inicial estiver suficientemente distante, não podemos garantir essa convergência. A partir disso, surge o seguinte questionamento: se só podemos garantir que o método converge para o minimizador em uma certa vizinhança e se nós queremos determinar tal ponto, como podemos escolher um ponto x^0 para iniciar as iterações? Esse problema pode ser tratado de diversas formas, através do controle do passo e de busca linear, como é discutido em [1].

Quando as modificações referidas acima são utilizadas, garantimos a **convergência global** do método de Newton modificado, ou seja, se tormarmos qualquer ponto $x^0 \in \mathbb{R}^n$, então todo ponto limite da sequência gerada pelo método é um ponto estacionário $x^* \in \mathbb{R}^n$, isto é, um ponto que anula $\nabla f(x)$.

Um dos principais problemas do método de Newton é o cálculo da Hessiana, além de haver a necessidade de se resolver um sistema linear a cada iteração. Para contornar esses problemas, foram desenvolvidos métodos que aproximam a matriz Hessiana (ou sua inversa), a fim de que as operações se tornem mais simples e a convergência seja quase tão rápida quanto a de Newton. Esses métodos são chamados de *Métodos Quase-Newton*. Uma boa descrição desses métodos pode ser obtida em [1].

4 Quadrados Mínimos Não–Lineares

Já mencionamos que problemas de ajuste de curvas podem recair em problemas de quadrados mínimos lineares. Porém, isso apenas ocorre em um conjunto muito particular de problemas, onde os parâmetros a determinar são coeficientes de uma combinação linear de funções conhecidas. Agora, vamos considerar o caso mais geral, em que tais parâmetros estão dispostos não–linearmente na função. Portanto, teremos um problema do tipo

$$Minimizar\ f(x) = \frac{1}{2}R(x)^t R(x) = \frac{1}{2}\sum_{i=1}^{m} r_i(x)^2, \tag{11}$$

onde $x \in I\!R^n$, $m > n$ e $R : I\!R^n \to I\!R^m$ é a *função resíduo*, é não–linear em x e $r_i(x)$ representa a i–ésima componente de $R(x)$. A solução x^* do problema de **quadrados mínimos não–lineares** (11) é chamada de **solução de quadrados mínimos**. Note que em geral temos $m >> n$.

Podemos notar que o problema de QMNL é muito próximo de problemas já analisados aqui. Se $m = n$, temos um sistema não–linear. Se $m > n$, obtemos um problema de minimização irrestrita. Vamos analisar o gradiente e a Hessiana da função f:

$$\nabla f(x) = \sum_{i=1}^{m} r_i(x)\nabla r_i(x) = J(x)^t R(x). \tag{12}$$

$$\nabla^2 f(x) = \sum_{i=1}^{m} (\nabla r_i(x)\nabla r_i(x)^t + r_i(x)\nabla^2 r_i(x)) = J(x)^t J(x) + S(x), \tag{13}$$

onde $S(x) = \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x)$. Usando a direção de descida do método de Newton, teríamos:

$$x^{k+1} = x^k - [J(x^k)^t J(x^k) + S(x^k)]^{-1} J(x^k)^t R(x^k). \tag{14}$$

Note que essa iteração é muito difícil de se obter computacionalmente, já que temos que efetuar diversas operações matriciais, além de uma inversão matricial. A matriz $S(x)$ é incômoda, pois precisamos calcular a Hessiana de cada uma das componentes do resíduo. Dessa dificuldade, buscou-se obter um método eficiente e robusto que dispensasse essa computação, chegando-se ao **Método de Gauss–Newton**.

5 Método de Gauss–Newton

Da seção anterior, a resolução de um problema de QMNL pode exigir muito esforço computacional devido ao passo complexo que o Método de Newton propõe. Porém, o

Método de Gauss–Newton despreza a parte mais cara da iteração (14), chegando a uma iteração do tipo:

$$x^{k+1} = x^k - [J(x^k)^t J(x^k)]^{-1} J(x^k)^t R(x^k). \tag{15}$$

A iteração acima é realizada se a Jacobiana tem posto completo. Note que não realizamos a inversão de $J(x)^t J(x)$, mas sim resolvemos um problema de quadrados mínimos lineares a cada iteração. De fato, seja $s^k = x^{k+1} - x^k$ a direção de descida do método de Gauss–Newton, temos que:

$$J(x^k)s^k = -R(x^k),$$

ou seja, devemos encontrar s^k que resolve

$$Min\ ||\rho(s^k)||_2 = ||J(x^k)s^k + R(x^k)||, \tag{16}$$

que é um problema de quadrados mínimos lineares. Se $J(x^k)$ tem posto completo, podemos resolver o sistema normal através da fatoração de Cholesky. Se $J(x^k)$ não tem posto completo ou se queremos mais estabilidade numérica, buscaremos resolver (16) através de fatorações ortogonais ou decomposição em valores singulares.

Uma observação importante a se fazer: note que o problema de QMNL não é convexo. Assim, não podemos garantir que existe um minimizador global. Outro problema que decorre disso é a existência de funções com inúmeros minimizadores locais, que podem estar longe do minimizador global da função.

5.1 Convergência do Método de Gauss–Newton

Agora que temos a iteração do método de Gauss–Newton, vamos supor que $J(x^k)$ tem posto completo e usaremos a expressão dada em (15). Queremos estudar como se dá a convergência das seqüências geradas pelas iterações do método. Para isso, precisaremos da definição de continuidade de Lipschitz.

Definição 1

Sejam $m, n > 0$, $G : \mathbb{R}^n \to \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$. Sejam $||\cdot||$ uma norma em $\mathbb{R}^n$ e $|||\cdot|||$ uma norma em $\mathbb{R}^{m\times n}$. G é dita Lipschtiz–contínua em x se existe um conjunto aberto $D \subset \mathbb{R}^n$, $x \in D$, e uma constante real γ tal que para todo $v \in D$,

$$|||G(v) - G(x)||| \leq \gamma||v - x||. \tag{17}$$

A constante γ é chamada de constante de Lipschitz para G em x. Para cada conjunto D contendo x para o qual (17) é válida, G é dita Lipschitz–contínua em x na vizinhança D. Se (17) é válida para todo $x \in D$, então $G \in \mathtt{Lip}_\gamma(D)$.

Teorema 2 *(Convergência Local do Método de Gauss–Newton)*

Seja $R : \mathbb{R}^n \to \mathbb{R}^m$ *e seja* $f(x) = \frac{1}{2}R(x)^t R(x)$, $f \in C^2(D)$, *onde* $D \subset \mathbb{R}^n$ *é um conjunto aberto e convexo. Suponhamos que* $J(x) \in \mathtt{Lip}_\gamma(D)$, *com* $||J(x)||_2 \leq \alpha$ *para todo* $x \in D$, *e que existe* $x^* \in D$ *e* $\lambda, \sigma \in \mathbb{R}$, $\sigma \geq 0$, *tal que* $J(x^*)^t R(x^*) = 0$, λ *é o menor autovalor de* $J(x^*)^t J(x^*)$, *e*

$$||[J(x) - J(x^k)]^t R(x^*)||_2 \leq \sigma ||x - x^*||_2 \tag{18}$$

para todo $x \in D$. *Se* $\sigma < \lambda$, *então para todo* $c \in (1, \frac{\lambda}{\sigma}$, *existe* $\varepsilon > 0$ *tal que para todo* $x^0 \in \mathcal{N}(x^*, \varepsilon) \equiv \{x \in \mathbb{R}^n : ||x - x^*|| < \varepsilon\}$, *a seqüência gerada pelo método de Gauss–Newton*

$$x^{k+1} = x^k - [J(x^k)^t J(x^k)]^{-1} J(x^k)^t R(x^k) \tag{19}$$

é bem definida e converge para x^*, *obedecendo*

$$||x^{k+1} - x^*||_2 \leq \frac{c\sigma}{\lambda} ||x^k - x^*||_2 + \frac{c\alpha\gamma}{2\lambda} ||x^k - x^*||_2^2 \tag{20}$$

e também

$$||x^{k+1} - x^*||_2 \leq \frac{c\sigma + \lambda}{2\lambda} ||x^k - x^*||_2 < ||x^k - x^*||_2 \tag{21}$$

O teorema acima garante a convergência em uma vizinhança $\mathcal{N}(x^*, \varepsilon)$ da solução. A partir desse teorema, obtemos um importante resultado, enunciado abaixo.

Corolário 1

Suponhamos as mesmas hipóteses do teorema anterior. Se $R(x^*) = 0$, *então existe* $\varepsilon > 0$ *tal que para todo* $x^0 \in \mathcal{N}(x^*, \varepsilon)$, *a seqüência* $\{x^k\}$ *gerada pelo método de Gauss–Newton está bem definida e converge quadraticamente para* x^*.

Para a demonstração do Teorema 2 e de seu Corolário, ver em [1]. O Corolário 1 é muito importante para que possamos prever o comportamento da convergência local. Podemos concluir que problemas de QMNL com resíduo grande tendem a ter convergência mais lenta, enquanto problemas com resíduo pequeno se aproximam do problema descrito no Corolário 1 e, então, sua convergência é mais rápida. Note também que, quanto maior se torna o resíduo, mais a nossa aproximação $\nabla^2 f(x^*) \approx$ $\approx J(x^*)^t J(x^*)$ se torna ruim. Uma análise dos casos em que isso se torna ruim está descrito na subseção seguinte.

5.2 Análise do Método de Gauss–Newton

O método de Gauss–Newton tem grandes vantagens em relação ao desempenho de outros métodos para a resolução de problemas de QMNL, como por exemplo:

- Não realiza o cálculo de matrizes Hessianas da função ou das componentes da função resíduo, o que diminui o esforço computacional necessário;

- Convergência local com ordem quadrática em problemas de resíduo–zero;
- Convergência local e rápida com ordem linear em problemas com resíduo pequeno e que não são muito não–lineares;
- Resolve problemas de quadrados mínimos lineares em uma iteração.

Agora, o método de Gauss–Newton também apresenta algumas dificuldades:

- Não podemos garantir sua convergência global;
- Não está bem definido quando a matriz Jacobiana não tem posto completo;
- Converge lentamente com ordem linear em problemas com resíduos grandes ou que são suficentemente não–lineares;
- Pode não convergir localmente quando o resíduo é muito grande ou em problemas muito não–lineares.

Assim, torna-se necessário aprimorar o método de Gauss–Newton de forma que possamos garantir a convergência global.

5.3 Modificações do Método de Gauss–Newton

Percebemos das afirmações feitas até agora que não temos um resultado que nos garanta a convergência global do método de Gauss–Newton. Vamos utilizar modificações próximas das já mencionadas para o método de Newton. Uma modificação comum é adicionar ao método de Gauss–Newton a busca linear em suas iterações. Assim, a iteração ficaria:

$$x^{k+1} = x^k - \lambda_k [J(x^k)^t J(x^k)]^{-1} J(x^k)^t R(x^k), \tag{22}$$

onde λ_k pode ser obtido através dos métodos mais comuns de busca linear, como o método da bissecção e o método da razão áurea. Para a descrição desses métodos, juntamente com condições (Armijo, Wolfe, entre outras) que controlam a busca linear, ver [1], [2] e [4]. Outras modificações envolvem regiões de confiança e não serão discutidas aqui.

A modificação descrita acima garante a convergência global, quando usada com as condições já mencionadas também. O problema que persiste está na velocidade de convergência, que é muito lenta quando o problema é muito não–linear ou tem resíduo muito grande. Outro problema que também permanece é a existência de funções com vários minimizadores locais, que atrapalham a convergência do método para o minimizador global.

6 Testes Numéricos: Ajustes de Curvas

Durante o desenvolvimento do projeto, foram realizados diversos testes numéricos utilizando o *software* `MatLab` como suporte computacional. Foram realizados testes das mais variadas estruturas. Nesta seção, vamos comentar os resultados obtidos com algumas delas.

6.1 Função `lsqnonlin` e Tratamento dos Dados

Para resolver problemas de QMNL, usaremos a função `lsqnonlin` do `MatLab`, que implementa o método de Gauss–Newton com algumas modificações que garantem sua convergência. Para maiores informações, consultar o `help` do `MatLab`.

A função `lsqnonlin` tem vários tipos de entrada e saída de dados possíveis. Estaremos interessados em uma estrutura em especial:

```
[X, resnorm] = lsqnonlin(@fun, X0,[],[] ,options, x, y),
```

onde teremos os seguintes parâmetros de entrada e de saída: `fun` é o nome do arquivo no formato `.m` onde a função resíduo está definida, `X0` é o valor de inicialização dos parâmetros que queremos determinar, `options` são as configurações feitas (serão discutidas posteriormente), `x` e `y` são as coordenadas dos pontos aos quais queremos ajustar a função, `X` é o valor dos parâmetros dado como saída e `resnorm` é a norma do resíduo após a finalização do procedimento.

Quanto às configurações feitas em `options`, modificaremos duas: `LargeScale` e `Jacobian`. A primeira será desativada, pois trataremos de problemas de pequeno e médio porte. A segunda será estudada com mais cuidado e será ativada em alguns procedimentos e desativada em outros. Se `Jacobian` for ativada, a rotina `lsqnonlin` buscará na função de entrada a Jacobiana fornecida pelo usuário avaliada em alguns pontos. Caso contrário, a matriz Jacobiana será aproximada por diferenças finitas.

Assim, toda função que daremos como entrada terá a seguinte estrutura:

```
function [F,J] = myfun(x)
    F = ...  % Função objetivo avaliada em x
    if nargout > 1
        J = ...  % Matriz Jacobiana avaliada em x
    end
```

Se o número de argumentos for maior que 1, a função também retorna a Jacobiana, o que nos permite utilizar as configurações já mencionadas.

6.2 Problemas analisados

Analisamos vários problemas em diversas classes. A seguir seguem as estruturas dos problemas analisados:

- Primeiramente, analisamos problemas com resíduo zero, com a Jacobiana calculada analiticamente. Nesses experimentos mudamos o número de pontos e os pontos iniciais fornecidos.
- Tendo realizado os experimentos acima, testamos como a Jacobiana aproximada por diferenças finitas influencia na solução obtida. Ainda trabalhamos com problemas de resíduo zero.
- Agora, já analisaremos casos em que os pontos estão próximos de funções conhecidas. Para gerá-los, calculamos os pontos na função e somamos uma perturbação. Nesses experimentos, mudamos o tamanho da perturbação, o número de pontos, o cálculo da Jacobiana e os pontos iniciais.

6.3 Resultados e Comentários

Nessa seção comentaremos alguns testes computacionais e seus resultados, que se destacaram no conjunto de experimentos realizados.

1. O primeiro problema que iremos comentar aqui consiste em ajustar a uma curva do tipo $f(x) = Ae^{Bx}$ um conjunto de pontos dado. Para que o problema seja de resíduo zero, geramos os pontos da seguinte forma: tomamos as coordenadas x no intervalo $[0, 0.5]$, com um espaçamento de 0.01, e y, que foram geradas usando a função $y = 2e^{3x}$. Usando a rotina `lsqnonlin`, com a matriz Jacobiana calculada analiticamente, obtivemos os resultados:

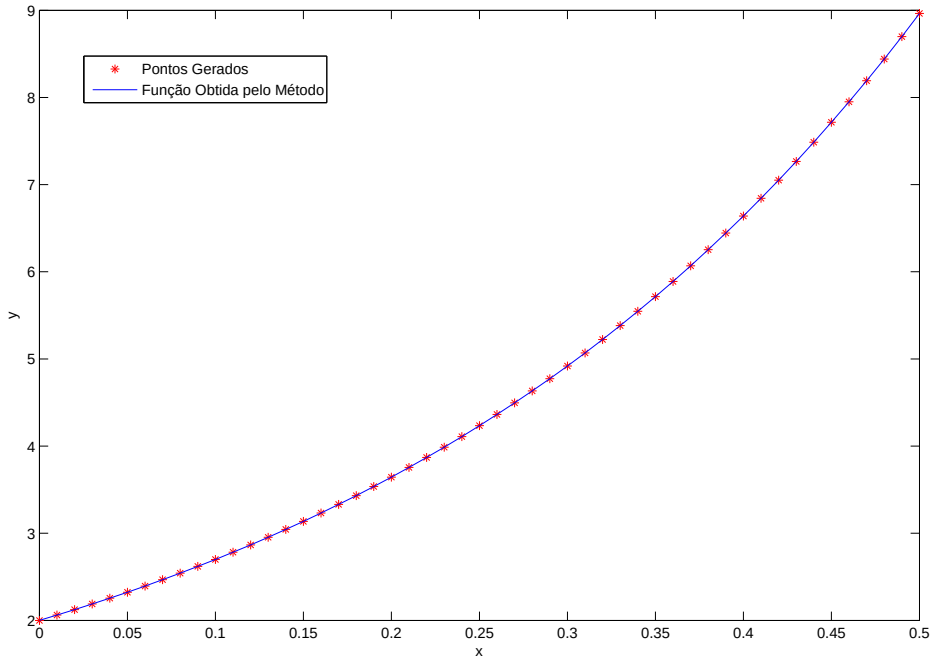
$$x = (2.0000, 3.0000)^t$$

$$||R(x)||_2 = 2.998e - 13$$

Assim, vemos que o método encontrou a solução esperada dentro da precisão de 4 casas decimais. Inicializando o método com diferentes pontos, obtemos resultados parecidos, já que a função é "bem comportada". Se o ponto inicial for o próprio ponto ótimo, $x^* = (2, 3)^t$, o procedimento pára e retorna $||R(x)||_2 = 0$. O gráfico da figura 1 representa os pontos iniciais e a função que melhor se ajusta a eles.

2. Um outro problema analisado é análogo ao anterior, mas a função se torna $f(x) = Ax^3 + B\sin(Cx)$. O conjunto de pontos é agora gerado da seguinte forma: toma-se x no intervalo de $[-0.5, 0.5]$, com um espaçamento de 0.01 e y é gerado pela função $y = x^3 - 0.5\sin(2x)$. A tabela 1 mostra os diferentes resultados para diferentes pontos iniciais.

Figura 1: Pontos iniciais (em vermelho) e a função encontrada pelo método (em azul)



Podemos perceber que existe mais de um ponto ótimo, mas o minimizador global ainda é o ponto $(1, -0.5, 2)^t$. Perceba que o terceiro ponto inicial é próximo da solução global, mas converge para um ponto totalmente diferente. Para melhor analisarmos esse comportamento estranho, fixamos o primeiro parâmetro como 1 e então plotamos o gráfico da função resíduo nas proximidades da origem. Verificamos que tal função não é convexa e, portanto, podemos ter diversos minimizadores locais, o que de fato ocorre. As figuras 2 e 3 mostram o comportamento da função resíduo.

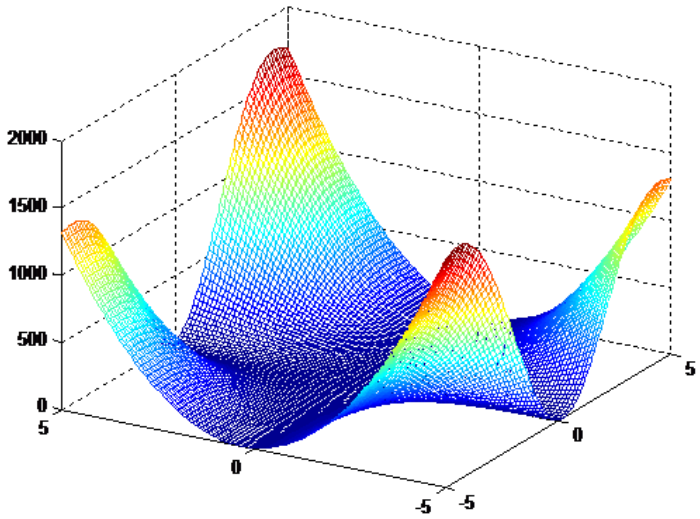
Observando os gráficos, percebemos que a função tem pontos de sela e minimizadores locais. Além disso, por simetria, temos a existência de vários minimizadores locais. O que percebemos também é que nas proximidades da origem, existem diversas direções de descida, que podem levar o método a convergir para vários pontos estacionários distintos.

3. Agora, tomemos a função $f(x) = 3 + \sin(Ax + B)$. De forma análoga, ge-ramos pontos pertencentes à curva $y = 3 + \sin(0.5x - 3)$. A função resíduo também possuía vários minimizadores locais, que foram atingidos a partir de diver-

Tabela 1: Algumas iterações do método

x^0	x^*	$\|\|R(x)\|\|_2$
$(1,1,1)^t$	$(1.4657,-0.9981,1.0000)^t$	$2.2862e-6$
$(1,-0.5,2)^t$	$(1.0000,-0.5000,2.0000)$	0
$(1.1,-0.7,2.2)^t$	$(0.8762,-0.4550,2.2000)^t$	$5.3664e-7$

Figura 2: Função resíduo



sos pontos iniciais. A figura 4 representa o gráfico da função resíduo.

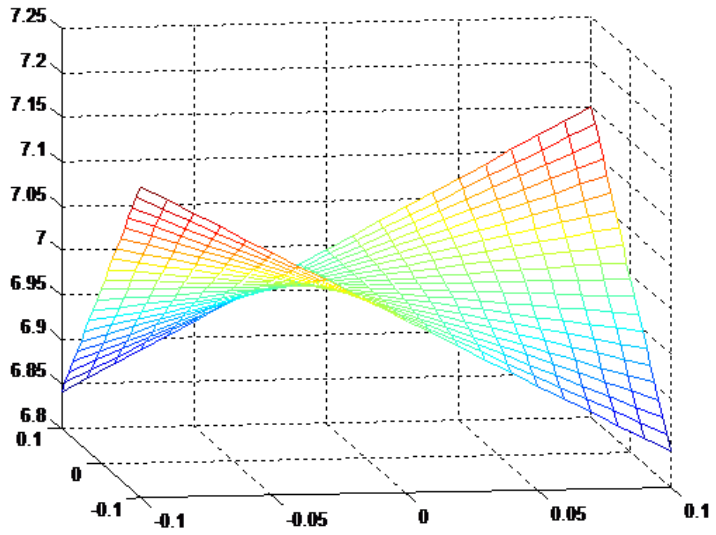
Os dois últimos exemplos foram importantes para percebermos como a estrutura da função pode prejudicar a descoberta das soluções. Quando a função resíduo tem muitos pontos estacionários, devemos gerar uma grande quantidade de pontos aleatórios, mas sempre tendo em mente que podemos nunca chegar ao minimizador global.

4. Consideremos agora um experimento que estuda o comportamento de um circuito RC-série. O circuito é composto por uma fonte de tensão constante, um resistor, um capacitor e um amperímetro, todos ligados em série. Os valores da resistência e da capacitância são, respectivamente, $47k\Omega$ e $1mF$. O experimento pode ser visto em detalhes em [8].

Quando o capacitor está sendo carregado, a corrente elétrica $i(t)$ que passa pelo circuito é dada por

$$i(t) = i_0 e^{-\frac{t}{RC}}, \tag{23}$$

Figura 3: Função resíduo nas proximidades da origem



onde R é a resistência do circuito, C é a capacitância do capacitor e i_0 é a corrente em $t = 0$. A constante $\tau = RC$ é chamada de constante de tempo. Podemos calcular i_0 através de

$$i_0 = E/R, \tag{24}$$

onde E é a diferença de potencial imposta pela fonte. O objetivo principal do experimento é determinar experimentalmente o valor da constante de tempo e compará-lo com o valor esperado.

Medimos os dados, que estão dispostos na tabela 2.

Como os dados foram obtidos experimentalmente, esse problema não é de resíduo zero, por apresentar erros decorrentes de medidas e procedimentos experimentais. Os resultados obtidos pelo `MatLab` são:

$$i_0 = 287.4001$$

$$\tau = 59.1377$$

$$||R(x)||_2 = 139.5920$$

Se usarmos os resultados teóricos, obtemos os valores $i_0 = 319.14mA$ e $\tau = 47$. Houve uma pequena variação nos resultados encontrados, o que pode ser decorrente de, principalmente, erros de medida e dos erros dos próprios aparelhos. O gráfico 5 representa os resultados obtidos.

Figura 4: Função Resíduo

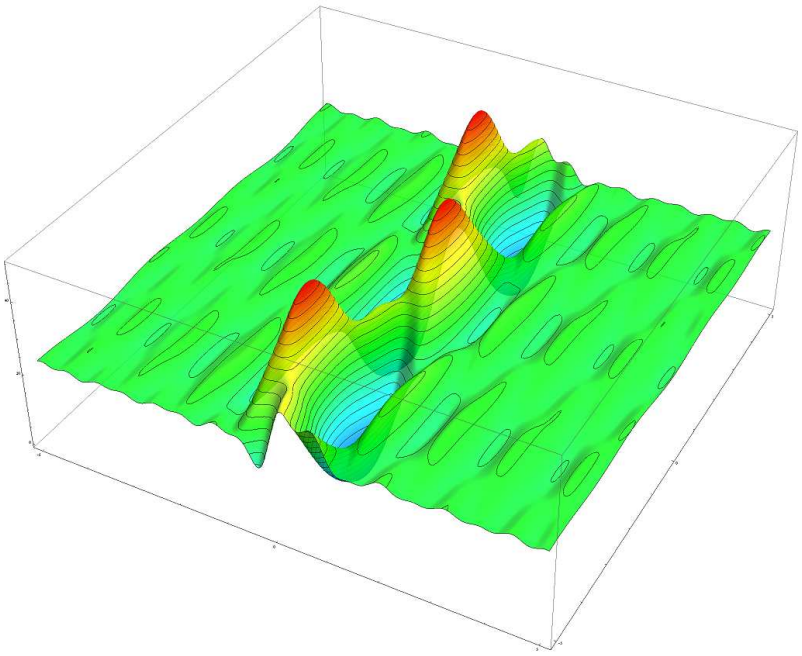


Figura 5: Experimento do Circuito RC-Série

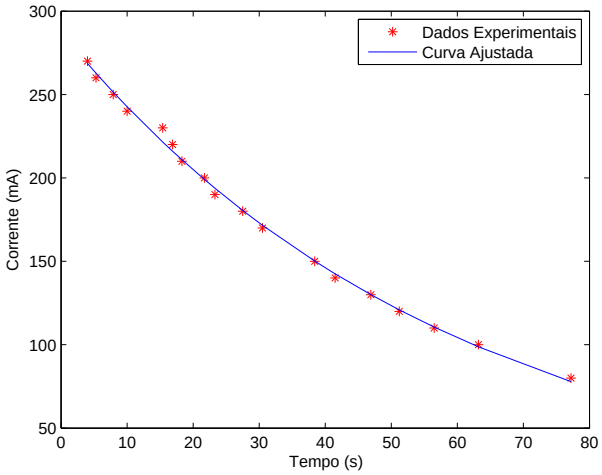


Tabela 2: Dados Experimentais do Circuito RC

Medida	Corrente (mA)	Tempo (s)
1	270	4.0
2	260	5.3
3	250	7.9
4	240	10.0
5	230	15.4
6	220	16.9
7	210	18.3
8	200	21.7
9	190	23.3
10	180	27.5
11	170	30.5
12	150	38.4
13	140	41.5
14	130	46.9
15	120	51.2
16	110	56.5
17	100	63.2
18	80	77.2

7 Aplicação em Deformação de Objetos 2D

Nesta seção, vamos discutir a aplicação de QMNL ao problema de deformação de objetos 2D, como proposto por Weng et al. em [7]. Para isso, apresentaremos alguns tópicos de teoria de grafos, que serão expostos a seguir.

7.1 Elementos de Teoria de Grafos

Para o problema de deformação de objetos 2D, foi necessário modelar os pontos marcados na figura como um grafo, de forma similar à utilizada em [7]. Sendo assim, destacamos alguns tópicos de Teoria de Grafos, que foram essenciais para o desenvolvimento do projeto.

Definição 2

Um grafo finito $G = (V, E)$ consiste de V, um conjunto finito e não-vazio de vértices, e E, um conjunto de arestas. Cada aresta é um par não-ordenado de vértices. Se uma aresta e_{ij} corresponde ao par $\{v_i, v_j\}$, dizemos que estes são suas extremidades e que estão conectados por e_{ij}.

Aplicaremos um grafo com estrutura especial em nosso problema, que chamamos de *grafo simples*.

Definição 3

Seja $G = (V, E)$ um grafo finito. Dizemos que G é um grafo simples se qualquer aresta $e \in E$ conecta dois vértices distintos e se, dados dois vértices distintos $v, u \in V$, não existe mais de uma aresta que os conecte.

Existem diversas possíveis representações para grafos. Convencionou-se que a representação gráfica usual é utilizar círculos para os vértices pertencentes a V e usar linhas para representar as arestas em E, realizando a ligação entre as extremidades.

Para uso computacional, precisaremos de uma estrutura melhor para representar um grafo. Existem várias maneiras de se fazer isso, mas nesse projeto estaremos especialmente interessados nas *matrizes de adjacências*.

Definição 4

Seja $G = (V, E)$ um grafo simples, com $V = \{v_1, v_2, \ldots, v_n\}$, ou seja, $|V| = n$. A matriz de adjacências do grafo G, denotada por $A_G = [a_{ij}]$, é uma matriz $n \times n$ com elementos binários, dados da seguinte forma:

$$a_{ij} = \begin{cases} 1 & \text{se } \{v_i, v_j\} \text{ é uma aresta de } G, \\ 0 & \text{caso contrário.} \end{cases}$$

Outra forma de representar grafos, que é usada por Weng et al. em [7], é a *Matriz Laplaciana* do grafo. Não faremos uso dessa matriz, pois utilizamos as funções como está descrito na próxima seção.

Agora, para finalizar, definiremos caminhos em grafos e grafos conexos, que serão muito importantes para a modelagem dos objetos 2D.

Definição 5

Sejam $G = (V, E)$ um grafo finito, n um inteiro não-negativo e $u, v \in V$. Um caminho de tamanho n de u para v em G é uma sequência de n arestas $e_1, e_2, \ldots, e_n$ de G tais que $e_k = \{x_{k-1}, x_k\}$, onde $k \in 1, 2, \ldots, n$, $x_0 = u$, $x_n = v$, $x_i \in V, \forall i \in 0, 1, \ldots, n$. Em grafos simples, denotaremos o caminho por sua sequência de vértices $x_0, x_1, \ldots, x_n$, já que a sequência de vértices determina unicamente o caminho. Um caminho é simples se ele não contém a mesma aresta mais de uma vez.

Definição 6

Seja $G = (V, E)$ um grafo finito. G é dito conexo se, e somente se, existe um caminho entre qualquer par de vértices distintos no grafo.

A conectividade em grafos será importante para a modelagem dos objetos 2D, pois sempre trataremos de figuras que poderão ser modeladas por grafos conexos. Ou seja, não trataremos o caso em que temos dois ou mais objetos 2D na mesma instância de entrada.

Para maiores detalhes a respeito dos tópicos de teoria de grafos discutidos acima, ver [5].

7.2 Modelagem das figuras

Para realizarmos a implementação do método para deformar as figuras 2D, primeiramente precisamos processar as imagens para marcar um conjunto de pontos na fronteira do objeto e em seu interior. Por enquanto, consideraremos objetos monocromáticos, de cor preta. Pontos que não fazem parte do objeto estarão na cor branca e serão considerados o " plano de fundo" da imagem.

Usando a rotina `bwtraceboundary` do `MatLab`, marcamos todos os pontos da fronteira da figura. A partir da fronteira demarcada, geramos uma lista com alguns pontos da fronteira, seguindo um espaçamento definido pelo usuário. Os demais pontos são desprezados. Outra lista foi gerada com pontos interiores ao objeto, que foram marcados usando-se uma grade de pontos igualmente espaçados. Esse espaçamento não é necessariamente igual ao anterior e é também um parâmetro de entrada do programa.

A partir desses pontos, modelamos a figura como um grafo $G = (V, E)$, onde V contém todos os n vértices do grafo, que correspondem aos pontos marcados. V é a união de dois conjuntos disjuntos de vértices: V_p, que contém m vértices do polígono determinados pelos pontos selecionados na fronteira, e V_g, que contém os $(m-n)$ pontos interiores ao objeto. Analogamente, dividimos o conjunto de arestas em dois: E_p, que contém as arestas na poligonal da fronteira, e E_g, que representa o restante das arestas no grafo.

A figura 6 representa uma entrada do programa que modela a imagem, retornando a figura 7, onde os pontos marcados em vermelho (na fronteira) representam V_p e os pontos em verde (no interior) representam V_g.

7.3 Preservação de Propriedades Locais

Antes de realizarmos a deformação, discutiremos cada propriedade que irá compor a função a ser minimizada. Essas funções são importantes para garantir que a figura não perca características que eram apresentadas antes da deformação.

7.3.1 Coordenadas Laplacianas da Curva

Dado um ponto $v_i \in V_p$, definimos sua Coordenada Laplaciana δ_i na curva poligonal da fronteira como a diferença entre v_i e a média dos vértices vizinhos a ele em V_p,

Figura 6: Imagem original fornecida pelo usuário.

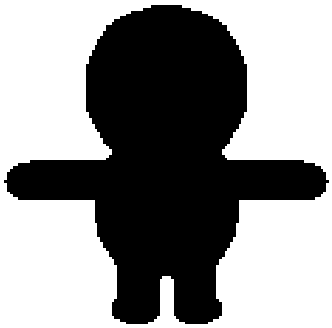
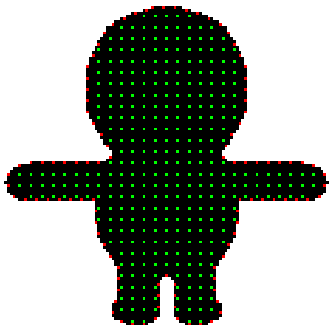


Figura 7: Imagem com os pontos do grafo representados.



ou seja,

$$\delta_i = \mathcal{L}_p(v_i) = v_i - \frac{v_{i-1} + v_{i+1}}{2}, \tag{25}$$

onde $1 \leq i \leq n$ e $v_0 = v_n, v_{n+1} = v_1$.

Temos a intenção de preservar as Coordenadas Laplacianas dos pontos em V_p, para que os pontos se ajustem às deformações que o objeto irá sofrer. Assim, iremos

minimizar a seguinte função:

$$\sum_{v_i \in V_p} ||\mathcal{L}_p(v_i^{(k+1)}) - \delta_i||^2, \tag{26}$$

onde $\delta_i = \mathcal{L}_p(v_i^{(0)})$, ou seja, minimizamos as mudanças nas Coordenadas de Laplace nos pontos em V_p, com o objetivo de preservar as propriedades do contorno da figura.

7.3.2 Coordenadas de Valor Médio

Além das coordenadas Laplacianas, para cada vértice em V_g, queremos manter sua posição relativa aos pontos vizinhos durante a deformação. Para isso, primeiramente calculamos as coordenadas de valor médio no polígono formado por seus vizinhos, da seguinte forma:

$$w_{i,j} = \frac{\tan(\alpha_j/2) + \tan(\alpha_{j+1}/2)}{||v_i - v_j||}, \tag{27}$$

onde α_j é o ângulo formado pelos vetores $v_j - v_i$ e $v_{j+1} - v_i$. Assim, para preservarmos as coordenadas de valor médio do objeto, minimizamos a seguinte função:

$$\sum_{v_i \in V_g} ||v_i^{(k+1)} - \sum_{(i,j) \in E} w_{i,j}^{(0)} * v_j^{(k)}||^2. \tag{28}$$

Note que enfatizamos aqui também que $w_{i,j}$ é calculado a partir dos vetores antes da deformação.

Essa função é muito importante, pois ela exerce o acoplamento entre as deformações nos pontos internos e externos.

7.3.3 Comprimento das Arestas

Como as coordenadas de valor médio são invariantes quanto ao escalamento dos elementos, preservá-las não é suficiente para manter a área local do objeto. Portanto, tentaremos preservar o comprimento das arestas do grafo durante a deformação. Nós minimizaremos a função abaixo:

$$\sum_{(i,j) \in E_g} ||(v_i^{(k+1)} - v_j^{(k+1)}) - e(v_i^{(0)}, v_j^{(0)})||^2, \tag{29}$$

onde $e(v_i, v_j) = \frac{l_{i,j}^*}{l_{i,j}}(v_i - v_j)$, $l_{i,j}$ é o comprimento atual da aresta (i, j) e $l_{i,j}^*$ é o comprimento antes da deformação. Assim, essa função é importante para preservar a estrutura interna da figura.

7.3.4 Restrições do Usuário

Além dessas funções para preservar as propriedades locais da figura, temos uma última função a ser minimizada, que é dada pelas restrições impostas pelo usuário, que determinam as deformações que o objeto sofrerá. O usuário seleciona alguns pontos $v_i \in V_p$ e determina a posição em que eles devem ficar após a deformação, que chamaremos de v_i^C, dados por um conjunto U. Assim, minimizaremos a função

$$\sum_{v_i^C \in U} ||v_i^{(k+1)} - v_i^C||^2. \tag{30}$$

7.4 Deformação de Objetos usando QMNL

7.4.1 Função Energia

Na seção anterior, discutimos cada propriedade local que queremos preservar durante a deformação do objeto. Assim, tendo as restrições do usuário, podemos escrever a função energia que iremos minimizar, que é a soma de cada uma das funções objetivo apresentadas anteriormente:

$$\begin{aligned}\mathcal{E}(V) = &\sum_{v_i \in V_p} ||\mathcal{L}_p(v_i) - \delta_i||^2 + \sum_{v_i \in V_g} ||v_i - \sum_{(i,j)\in E} w_{i,j} - v_j||^2 + \\ &+ \sum_{(i,j)\in E_g} ||(v_i - v_j) - e(v_i, v_j)||^2 + \sum_{v_C \in U} ||v_i - v_i^C||^2.\end{aligned} \tag{31}$$

Assim, o problema a ser resolvido para a deformação de objetos 2D será

$$Minimizar\ \mathcal{E}(V). \tag{32}$$

Para resolver (32), usaremos o método de Gauss–Newton, já descrito anteriormente. Encontrando a solução, a figura deve ser reconstruída a partir dos pontos do grafo.

7.4.2 Implementação

A implementação envolve diversas subrotinas. Cada uma das parcelas de (31) é computada separadamente a cada iteração e então uma rotina chamada `Energia` recebe a soma de todas essas subrotinas.

Por envolverem várias noções sobre representação computacional de grafos, as subrotinas foram um tanto complicadas de serem implementadas. Além disso, as questões geométricas do problema também dificultaram a elaboração dos algoritmos.

Ao iniciar a execução do programa o usuário fornece a imagem de entrada e o tamanho das grades de pontos a serem construídas. Algumas variáveis são pré-computadas e armazenadas, a fim de que o programa seja o mais eficiente possível.

(a) Original (b) Pontos marcados (c) Transladado (d) Deformação 1 (e) Deformação 2

Figura 8: Testes com um quadrado

Após isso, a rotina `lsqnonlin` gerencia o processo de minimização. Ao retornar a localização final dos pontos, suas coordenadas finais são convertidas para inteiros e são marcadas. Esse processo de arredondamento algumas vezes produz efeitos indesejados, mas no caso geral os resultados são satisfatórios. Após isso, as bordas da figura são reconstruídas e a figura é novamente preenchida de pontos pretos.

7.5 Resultados

Nesta subseção, explicitamos resultados obtidos através da análise de algumas figuras simples que foram deformadas no projeto. Figuras muito complexas, como o boneco exemplificado anteriormente, exigem algoritmos também complexos para reconstruir suas fronteiras a partir dos pontos marcados, o que não seria necessário para que os objetivos principais do projeto fossem atingidos.

Os primeiros testes que exemplificaremos envolvem um quadrado. Com esse quadrado, testamos se a translação funciona corretamente e experimentamos algumas deformações. A figura 8 ilustra alguns resultados obtidos. As deformações que estão ilustradas são: deslocar apenas um vértice a fim de formar um paralelogramo (figura 8d) e fixar um lado e mover um vértice do lado oposto, a fim de formar um retângulo (figura 8e).

Realizamos testes semelhantes com diversas outras imagens. Destacamos alguns deles nas figuras 9 e 10.

Percebemos que, em alguns casos, o fato de arredondarmos a posição final dos vértices pode provocar pequenos desvios de sua localização final esperada. Mesmo assim, notamos que em nenhum dos testes realizados a figura perde totalmente suas características. Por exemplo, realizamos várias deformações na estrela ilustrada na figura 10, mas suas características se preservaram bem, pois mesmo estando deformada podemos identificar que a figura se trata, de fato, de uma estrela.

(a) Original (b) Pontos marcados (c) Transladado (d) Deformação

Figura 9: Testes com um triângulo

(a) Original (b) Pontos marcados (c) Deformação 1 (d) Deformação 2

Figura 10: Testes com uma estrela

Conclusão e Comentários Finais

Os problemas de QMNL apresentam uma estrutura especial e, portanto, merecem um estudo detalhado e especializado. Técnicas de otimização geral não aproveitam a estrutura do problema e podem não ser eficazes para a sua solução. Contudo, a teoria geral de programação não–linear é importante para que tenhamos a perfeita compreensão do problema e possamos identificar situações como a da convergência para pontos de mínimo locais que não são boas soluções do problema.

Outro fato que torna os problemas de QMNL importantes é a grande limitação que os modelos lineares apresentam. Embora mais fáceis de resolver, problemas de QML ou de QMNL linearizados são muito restritos, e no segundo caso, as soluções podem ser ruins para o problema original.

Na aplicação à deformação de objetos 2D, notamos que o uso de teoria de grafos é importante para a modelagem do objeto, representando os pontos marcados na figura como vértices de um grafo, cujas arestas ligam alguns desses vértices e têm por peso a distância.

Finalmente, as aplicações de toda a teoria estudada ao problema proposto foram satisfatórias, pois obtivemos resultados coerentes e próximos aos que eram esperados. Embora a complexidade das figuras utilizadas não tenha sido muito alta, os experimentos computacionais foram suficientes para observar todas as funcionalidades previstas.

Referências

[1] J.E. Dennis & R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, PA, 1996.

[2] A. Friedlander, *Elementos de Programação Não Linear*. Editora da Unicamp, 1994.

[3] M.A. Gomes-Ruggiero & V.L.R. Lopes, *Cálculo Numérico – Aspectos Teóricos e Computacionais*. Segunda Edição, Makron Books, 1997.

[4] D.G. Luenberger & Y. Ye, *Linear and Nonlinear Programming*. Third Edition, Springer, New York, NY, 2008.

[5] K. H. Rosen, *Discrete Mathematics and Its Applications*. Sixth Edition, McGraw–Hill, New York, NY, 2006.

[6] D.S. Watkins, *Fundamentals of Matrix Computations*. Second Edition, Wiley–Interscience, New York, NY, 2002.

[7] Y. Weng, W.Xu, Y. Wu, K. Zhou & B.Guo, *2D Shape Deformation Using Nonlinear Least Squares Optimization. The Visual Computer*, pp. 653–660, 2006.

[8] *Estudo de um Circuito RC - Apostila de Física Experimental III (F329)*. IFGW, Unicamp, 2008.