Projeto 1 - Métodos para minimizar problemas irrestritos
Projeto 2 - Lagrangeano Aumentado para minimizar problemas restritos

UFPR - Programação Não Linear - Bacharelado em Matemática (4º Semestre)
Prof. Luiz Carlos Matioli - Departamento de Matemática

Dyckson Ternoski - GRR20185648
Marina Sayuri Vieira - GRR20185652
dycksonternoski@hotmail.com
marinasayuri16@gmail.com

1. Introdução do Projeto 1

Uma das questões mais importantes para a Matemática Aplicada é resolver problemas do tipo

$$(P) \quad \min_{x \in \Omega} f(x),$$

em que $f : \mathbb{R}^n \to \mathbb{R}$ é chamada de **função objetivo** do problema (P).

Para o caso em que $\Omega = \mathbb{R}^n$, temos um problema irrestrito. Este é o tipo de problema com que trabalhamos no Projeto 1.

Durante a primeira parte da disciplina, estudamos os seguintes métodos que podem ser usados para minimizar problemas irrestritos: Gradiente (Cauchy), Newton (Puro e Modificado), Quase Newton e Gradientes Conjugados.

Note que todos esses métodos consistem em algoritmos de descida, e por isso precisamos fazer buscas, exatas ou inexatas, para encontrar parâmetros de descida. Desse modo, também implementamos juntamente aos métodos, a busca de Armijo (inexata).

Neste projeto desenvolvemos os métodos citados acima em linguagem computacional Julia, cujos códigos podem ser acessados pelo link do repositório no Github: `https://github.com/dycksont/Metodos-Minimizar-Problemas`. Cada linha de nosso código está comentada de forma que o leitor possa compreender o que está sendo realizado a cada passo, mesmo sem o domínio da linguagem utilizada. Os códigos do Projeto 2 também estão inclusos neste diretório.

Com os códigos finalizados, aplicamos nossos algoritmos aos problemas da biblioteca CUTEst.

Este relatório tem como foco:

1. Fundamentação teórica dos métodos
    - Resumo de cada método
    - Algoritmos
    - Convergência dos métodos
2. Implementação e resultados
    - Resultados dos algoritmos para os problemas do CUTEst
    - Comparação dos resultados
3. Conclusão
    - Comparação: Teoria x Prática
    - Utilização do projeto

2. Métodos para Minimizar Problemas Irrestritos

Todos os métodos que serão apresentados são algoritmos de descida, cuja ideia se baseia na condição de otimalidade de 1ª ordem e na definição de direção de descida.

Condição de Otimalidade de 1ª ordem: Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função diferenciável e $\bar{x} \in \mathbb{R}^n$. Se $\bar{x}$ é um ponto estacionário de f, então $\nabla f(\bar{x}) = \bar{0}$

Definição: Seja $d \in \mathbb{R}^n \setminus \{\bar{0}\}$. Dizemos que d é de descida para f a partir de $\bar{x}$ se existir $\delta > 0$ tal que

$$f(\bar{x} + td) < f(\bar{x}), \qquad \forall t \in (0, \delta)$$

Além disso, se $\nabla f(\bar{x})^T d < 0$, então d é uma direção de descida a partir de $\bar{x}$.

Entendendo esses dois conceitos, podemos, a partir de um ponto inicial x^0, obter uma direção de descida d^0, e encontrar um ponto $x^1 = x^0 + t_0 d^0$ tal que $f(x^1) < f(x^0)$. Agora repetimos o processo para x^1, encontrando uma direção de descida d^1 e um ponto x^2.

Repetimos este processo até que a condição de otimalidade de primeira ordem seja satisfeita. Quando finalmente encontrarmos um x^k tal que $\nabla f(x^k) = \bar{0}$, saberemos que x^k é o minimizador de f, já que estamos caminhando em direções de descida.

Computacionalmente, não é possível conferir se $\nabla f(x^k) = \bar{0}$. Ao invés disso, dada uma **tolerância** $\epsilon > 0$ próxima de 0, veremos se $\left\|\nabla f(x^k)\right\| < \epsilon$. Em caso afirmativo, assumiremos que a norma está muito perto de zero, e portanto, o vetor gradiente calculado em x^k é aproximadamente o vetor nulo.

Não apenas isso, também temos outros parâmetros para analisar. Devemos definir um número k_{max} suficientemente grande para representar a **iteração máxima**, ou seja: se o algoritmo realizar mais do que k_{max} iterações, significa que ele está convergindo muito lentamente. Logo, não é vantajoso utilizar o método em questão para minimizar tal função objetivo. Nesse caso, devemos buscar um método com melhor convergência. O mesmo vale para o parâmetro $time_{max}$, que representa o **tempo máximo de convergência** do algoritmo.

Portanto, podemos finalmente definir um protótipo para algoritmos de descida:

Algoritmo 1: Protótipo de Algoritmo de Descida

Entrada: $x_0 \in \mathbb{R}^n, \epsilon > 0, k_{max} = 10000, time_{max} = 30(s)$
Saída: $\bar{x} \in \mathbb{R}^n$

início
  $k = 0$
  repita
    $d^k : \nabla f(x^k)^T d^k < 0$
    $t_k : f(x^k + t_k d^k) < f(x^k)$
    $x^{k+1} = x^k + t_k \cdot d^k$
    $k = k + 1$
  até $\left\|\nabla f(x^k)\right\| < \epsilon$ *ou* $k > k_{max}$ *ou* $time > time_{max}$;
  retorna $\bar{x}$
fim

Note, entretanto, que temos algumas questões a serem solucionadas.

1. Como determinar as direções de descida?

   Resposta: Isso varia para cada método utilizado. Em geral, a convergência do método depende das direções de descida escolhidas.

2. Como determinar o t_k?

Resposta: Podemos determinar t_k por busca linear. Isso pode ser feito de duas maneiras:

- Busca Linear Exata: Determina o t_k que minimiza a função $f(x^k + td^k)$. Em nosso projeto, utilizamos fórmulas para a busca exata quando necessário (casos quadráticos).
- Busca Linear Inexata: Determina um t^k que satisfaça

$$f(x^k + td^k) < f(x^k) \tag{1}$$

sem a necessidade que este parâmetro seja o minimizador. Implementamos o Método de Armijo para busca inexata. Vejamos detalhadamente como funciona a busca pela condição de parada de Armijo:

2.1. Busca Liner Inexata: Armijo

A primeira pergunta a ser feita é: por que devemos utilizar a busca inexata ao invés da exata? Bem, nem sempre é fácil determinar o t_k exato. Na prática, fazemos isso apenas no caso quadrático. Além disso, pode ser custoso computacionalmente (em memória e alocações) encontrar a solução exata para cada iteração do algoritmo. O método de Armijo é utilizado justamente para solucionar este problema.

Para introduzir o método, devemos primeiro definir a função

$$\varphi : \mathbb{R}_+ \to \mathbb{R}$$

$$\varphi(t) = f(x^k + td^k)$$

Utilizando aproximações do Polinômio de Taylor de 1^a ordem de φ em $t = 0$, encontramos que t_k satisfaz (1) se, e somente se, satisfaz

$$\varphi(t) < \varphi(0) + mt_k\varphi'(0) \tag{2}$$

para algum $m \in (0, 1)$.

Usando a definição da função φ, podemos reescrever (2) como:

$$f(x^k + t_k d^k) < f(x^k) + mt_k \nabla f(x^k)^T d^k \tag{2}$$

Logo, basta que esta seja a condição de parada de nosso algoritmo. Para encontrar tal t_k, utilizaremos a estratégia de "Backtracking": começaremos com $t_0 = 1$ e o reduziremos a cada iteração até encontrá-lo.

Uma vez finalizadas as ideias para encontrar t_k, temos nosso algoritmo para a busca de Armijo:

Algoritmo 2: Busca Inexata: Armijo

Entrada: $x_0 \in \mathbb{R}^n$, d_0: direção de descida, $m \in (0,1), \gamma \in (0,1)$
Saída: $t_k \in \mathbb{R}$

```
1 início
2 |  k = 0
3 |  t_0 = 1
4 |  repita
5 |  |  t_k = γ t_k
6 |  |  k = k + 1
7 |  até f(x^k + t_k d^k) < f(x^k) + m t_k ∇f(x^k)^T d^k;
8 |  retorna t_k
9 fim
```

Observação: Quando encontramos um t_k satisfazendo (2), dizemos que, para este t_k, $x^{k+1} \doteq x^k + t_k d^k$ entrou na **região de decréscimo de f**.

2.2. Método de Cauchy/Gradiente

O Método de Cauchy (ou como também chamado, Método do Gradiente) é um dos algoritmos de descida mais simples que podemos obter. Proposto pelo matemático Augustin-Louis Cauchy em 1847, o método se fundamenta na dedução de que dada uma função $f : \mathbb{R}^n \to \mathbb{R}$, a direção $d = -\nabla f(x^k) \neq \bar{0}$ é de **máxima descida** a partir de $x^k \in \mathbb{R}^n$.

De fato, não é muito difícil provar que $d = -\nabla f(x^k)$ é de descida. Sabemos que d é de descida a partir de x^k se $\nabla f(x^k)^T d < 0$. Logo, para $d = -\nabla f(x^k)$:

$$f(x^k)d = f(x^k)(-\nabla f(x^k)) = -\left\|\nabla f(x^k)\right\|^2 < 0$$

Portanto, sendo d uma direção de descida, segue o algoritmo de descida do Método de Cauchy, descrito abaixo.

Algoritmo 3: Método de Cauchy/Gradiente

Entrada: $x_0 \in \mathbb{R}^n, \epsilon > 0, k_{max} = 10000, time_{max} = 30(s)$
Saída: $\bar{x} \in \mathbb{R}^n$

```
1  início
2  |  k = 0
3  |  repita
4  |  |  d^k = −∇f(x^k)
5  |  |  t_k : f(x^k + t_k d^k) < f(x^k) (Busca Linear por Armijo)
6  |  |  x^{k+1} = x^k + t_{k+1} · d^k
7  |  |  k = k + 1
8  |  até ‖∇f(x^k)‖ < ε ou k > k_max ou time > time_max;
9  |  retorna x^k
10 fim
```

2.2.1. Cauchy para o Caso Quadrático

Para o caso em que a função objetivo é da forma quadrática, o t_{k+1} calculado pela busca na **Linha 5** do algoritmo tem solução exata, e é dada por

$$t_{k+1} = \frac{\nabla f(x^k)^T \nabla f(x^k)}{(d^k)^T A d^k} = \frac{\left\|\nabla f(x^k)\right\|^2}{(d^k)^T A d^k}$$

2.2.2. Convergência do Método de Cauchy

Temos como garantia, na teoria, que o método de Cauchy converge para um mínimo local da função objetivo. Além disso, se a função objetivo for convexa, temos que este mínimo local também é global. Além disso, independente do ponto inicial escolhido, o algoritmo convergirá. Logo, a convergência é global. Veremos mais adiante que isso não vale para todos os métodos.

Entretanto, na prática, a convergência do método do Gradiente não é garantida. Se escolhermos para minimizar um problema que tenha função objetivo da forma quadrática com matriz Hessiana mal condicionada, o algoritmo pode levar milhares de interações até convergir, ou não convergir. Para ambos os casos, temos um grande gasto de memória, alocações e tempo. Logo, nessas condições, não é viável utilizar Cauchy.

Observação: Uma matriz é mal condicionada quando seus autovalores apresentam uma diferença em módulo muito grande. Neste caso, dizemos que o problema tem condições ruins para ser tratado numericamente. Formalmente, defina:

- $\lambda_1 \doteq$ Menor autovalor da matriz Hessiana de f
- $\lambda_n \doteq$ Maior autovalor da matriz Hessiana de f

Se $\lambda_1 \ll \lambda_n$, então a matriz Hessiana é mal condicionada. Para o caso de funções quadráticas, as curvas de nível da função objetivo serão elipses alongadas, o que dificulta a convergência do método de Cauchy.

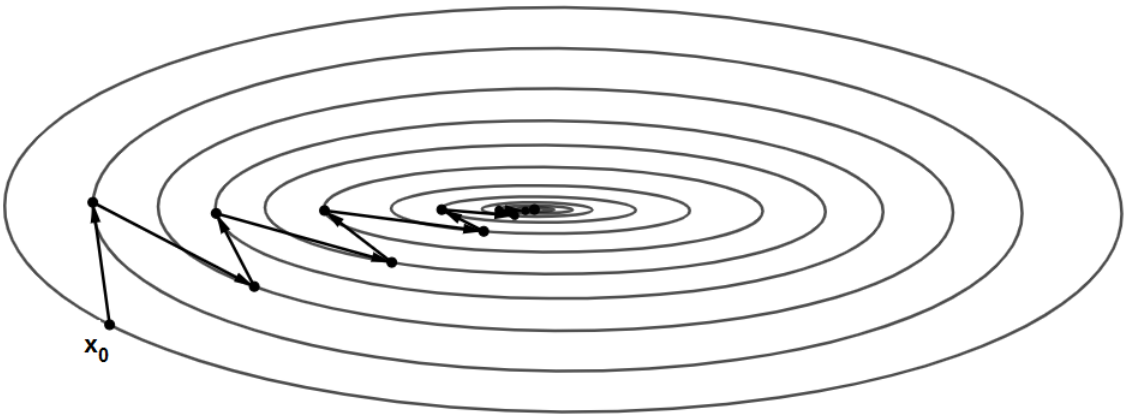

Figura 1: *Ao ter uma matriz Hessiana mal condicionada na função objetivo quadrática, o método de Cauchy leva várias iterações para convergir a partir de um ponto inicial* x_0.

Contudo, também existem casos em que Cauchy converge bem. Dependendo do ponto inicial escolhido ou das características da função objetivo, o método pode convergir em uma iteração, devido ao fato do vetor gradiente em um ponto ser perpendicular à reta tangente a este ponto nas curvas de nível.

Um exemplo disso é quando temos $\lambda_1 = \lambda_n$. Assim, no caso quadrático, as curvas de nível são circunferências.

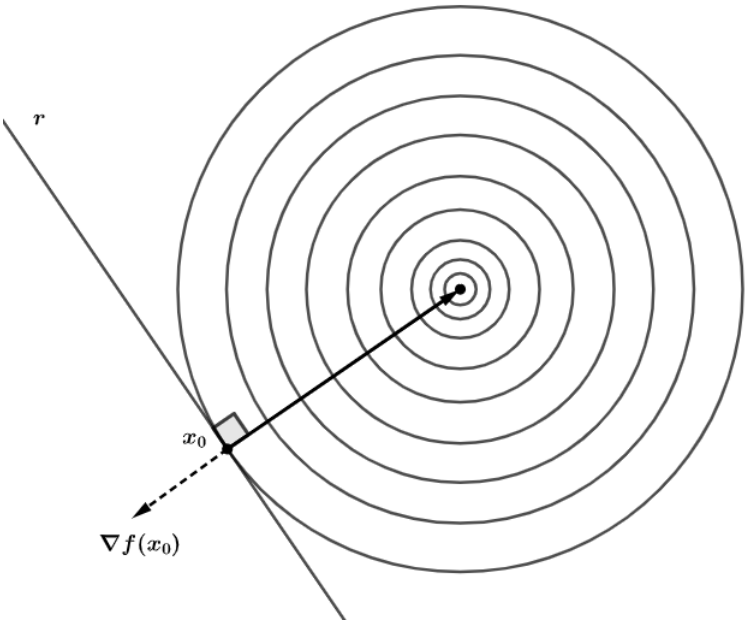

Figura 2: *Quando os autovalores da matriz Hessiana são iguais, as curvas de nível obtidas são circunferências. Neste caso, o método converge em apenas uma iteração.*

Em termos técnicos, a convergência do Método do Gradiente é classificada como sublinear.

2.3. Método de Newton

O algoritmo do Método de Newton localiza raízes de uma função $h : \mathbb{R}^n \to \mathbb{R}$. Isso é realizado ao tomar $x_0 \in \mathbb{R}^n$ um ponto inicial e encontrar a raíz do Polinômio de Taylor de ordem 1 da função h em x_0. Agora, tomamos essa raíz como o "novo x_0"e repetimos o processo. Com isso, encontraremos um $x^k \in \mathbb{R}^n$ com $h(x^k) = 0$, para algum $k \in \mathbb{N}$.

Como sabemos pela condição de otimalidade de primeira ordem, um ponto x^k definido numa função $f : \mathbb{R}^n \to \mathbb{R}$ é estacionário se $\nabla f(x^k) = \bar{0}$. Portanto, basta escolher $h = \nabla f$ e adaptar o Método de Newton. Como geraremos direções de descida, o ponto estacionário encontrado será com certeza um minimizador da função.

Algoritmo 4: Método de Newton

Entrada: $x_0 \in \mathbb{R}^n, \epsilon > 0, k_{max} = 10000, time_{max} = 30(s)$
Saída: $\bar{x} \in \mathbb{R}^n$

início
  $k = 0$
  repita
    $d^k = -(\nabla^2 f(x^k))^{-1} \nabla f(x^k)$
    $t_k : f(x^k + td^k) < f(x^k)$ (Busca Linear por Armijo)
    $x^{k+1} = x^k + t_k d^k$
    $k = k + 1$
  até $\left\|\nabla f(x^k)\right\| < \epsilon$ *ou* $k > k_{max}$ *ou* $time > time_{max}$;
  retorna x^k
fim

Note que na **Linha 5 do Algoritmo 2**, para determinar d^k o ideal é resolver o sistema $-\nabla^2 f(x^k)d^k = \nabla f(x^k)$, pois dessa forma temos um gasto computacional menor do que ao inverter a matriz $\nabla^2 f(x^k)$.

2.3.1. Newton Puro

No método acima, localizamos por meio de uma busca linear um t_k que satisfaz $f(x^k + t_k d^k) < f(x^k)$. Quando escolhemos $t_k = 1$ para todo $k \in \mathbb{N}$, sem realizar a busca, o método é chamado de Newton Puro.

2.3.2. Newton para o Caso Quadrático

O método de Newton apresenta um ótimo desempenho quando estamos trabalhando com uma função objetivo quadrática. Neste caso, o algoritmo converge em apenas uma iteração. Isso acontece por conta da **convergência quadrática** do Método de Newton.

2.3.3. Convergência do Método de Newton

Como citado na subseção anterior, o método de Newton apresenta convergência quadrática. Isso significa que se tivermos, por exemplo, $f : \mathbb{R} \to \mathbb{R}$, a cada iteração o erro absoluto diminuirá numa ordem de 10^2.

Sendo $\bar{x}$ a solução de (P) para a função f como acima, definimos o erro absoluto no ponto x^k como

$$\varepsilon_k \doteq | \bar{x} - x^k |$$

Suponha que $\varepsilon_1 \approx 0,002$.

Pela convergência quadrática, teremos que $\varepsilon_2 \approx 0,00002$ e $\varepsilon_3 \approx 0,000000002$. Perceba como o número de zeros está dobrando a cada iteração. Claro, já que $\varepsilon_3 = 10^{-2} * \varepsilon_2$ e $\varepsilon_2 = 10^{-2} * \varepsilon_1$, pela convergência quadrática.

Entretanto, o método de Newton não converge globalmente, assim como Cauchy. Dependendo do ponto inicial que escolhemos, pode ser que o método nunca encontre a solução. Abaixo vemos um exemplo em que o método de Newton fica em um *loop* infinito.

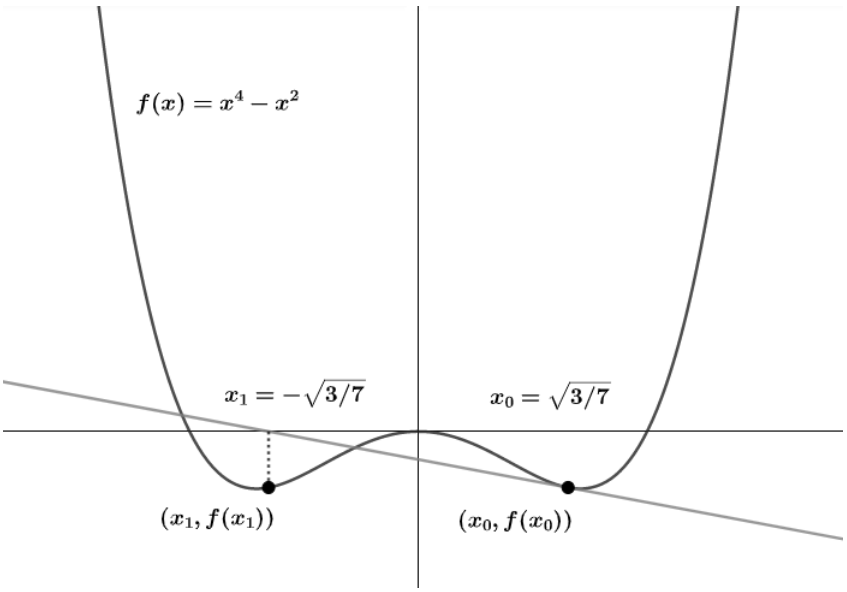

Figura 3: *Gostaríamos de encontrar um zero da função* $f : \mathbb{R} \to \mathbb{R}, f(x) = x^4 - x^2$. *Ao começar com um ponto* $x_0 = \sqrt{3/7}$, *obtemos, ao realizar uma iteração do Método de Newton Puro,* $x_1 = -\sqrt{3/7}$.

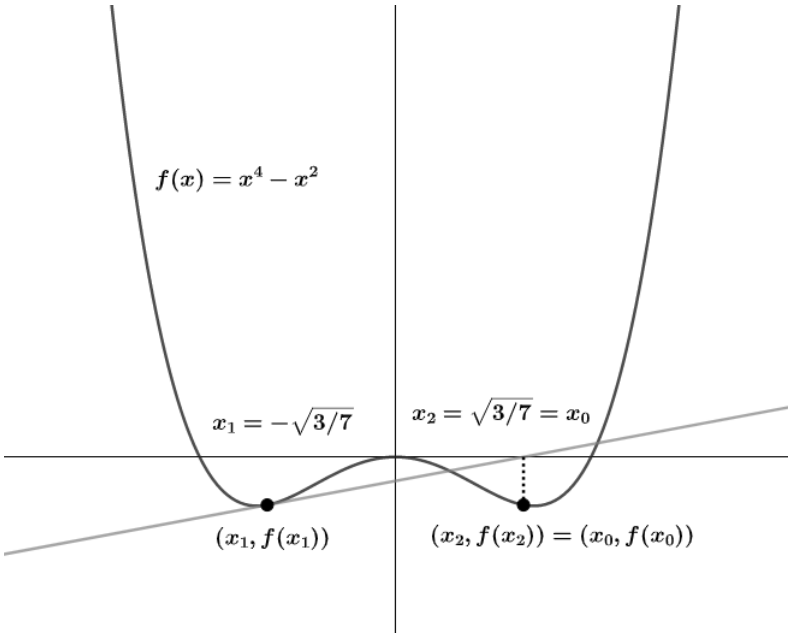

Figura 4: *Ao realizar a segunda iteração do método de Newton, agora partindo de* x_1*, encontramos* $x_2 = x_0$*. Logo, teremos que a k-ésima iteração resultará em* $x_k = x_0$*, se* k *for par, e em* $x_k = x_1$*, se* k *for ímpar. Como* $f(x_0) = f(x_1) \neq 0$*, concluimos que o método nunca convergirá.*

2.4. Método Quase Newton BFGS

Vimos que o Método de Newton apresenta convergência quadrática e inclusive converge em apenas uma iteração para minimizar problemas com função objetivo quadrática. Mas então, por que precisamos de um método que aproxima Newton? A resposta está no olhar computacional do método: calcular a matriz Hessiana é algo custoso computacionalmente. Perceba:

Se estamos lidando com um problema no $\mathbb{R}^n$, a matriz Hessiana da função objetivo terá dimensões nxn. Ou seja: existirão $n * n$ entradas. Logo, precisaremos calcular n^2 segundas derivadas. Isso nos custaria um número grande de alocações, e consequentemente, memória e tempo de CPU.

De fato, visto o problema, seria ótimo se conseguíssemos aproximar a matriz Hessiana sem realizar muitas alocações. Esta é a motivação para os métodos Quase-Newton.

Neste projeto, implementamos o método BFGS, cujo mérito do desenvolvimento deve-se aos matemáticos Broyden, Fletcher, Goldfarb e Shanno. Nele, aproximamos a matriz Hessiana por uma fórmula, apresentada no algoritmo, atualizando-a a cada nova iteração.

De fato, há várias multiplicações na fórmula que aproxima a matriz Hessiana. Por exemplo, ao realizar a multiplicação de matrizes $(B_k * S^k)$, sendo $B_k \in \mathbb{R}^{nxn}$ e $S^k \in \mathbb{R}^{nx1}$, estamos realizando $n * n$ multiplicações de números reais e $n * (n-1)$ operações de soma.

Entretanto, realizar somas e multiplicações é computacionalmente menos custoso do que calcular n^2 segundas derivadas. Veremos mais adiante, através de exemplos, como essas suposições são confirmadas na prática.

Observação: No algoritmo abaixo, a notação $B_0 > 0$ significa que a matriz B_0 é definida positiva.

Algoritmo 5: Método de Quase Newton

Entrada: $x_0 \in \mathbb{R}^n, \epsilon > 0, B_0 > 0 \in \mathbb{R}^{n\times n}, k_{max} = 10000, time_{max} = 30(s)$
Saída: $\bar{x} \in \mathbb{R}^n$

início
  $k = 0$
  repita
    $d^k = -B_k \cdot \nabla f(x^k)$
    $t_k : f(x^k + t_k d^k) < f(x^k)$ (Busca Linear por Armijo)
    $x^{k+1} = x^k + t_k \cdot d^k$
    $S^k = x^{k+1} - x^k$
    $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$
    $B_{k+1} = B_k - \frac{(B_k S^k)(B_k S^k)^T}{(S^k)^T (B_k S^k)} + \frac{y_k y_k^T}{(y^k)^T S^k}$
    $k = k + 1$
  até $\left\|\nabla f(x^k)\right\| < \epsilon$ *ou* $k > k_{max}$ *ou* $time > time_{max}$;
  retorna x^k
fim

2.4.1. Convergência do Método de Quase Newton

Como a ideia principal deste método é aproximar a função Hessiana, utilizada no método de Newton, podemos prever que a velocidade de convergência deve ser menor que a do método de Newton. Além disso, temos que o método supera Cauchy em questões de convergência. Especificamente, a convergência do método BFGS é superlinear.

Perceba que não estamos concluindo qual método é melhor ou pior: de fato, o Quase Newton foi desenvolvido para evitar uma falha computacional do método de Newton. Aqui fazemos apenas uma comparação entre as convergências.

2.5. Método de Gradientes Conjungados - Caso Quadrático

Desenvolvido pelos matemáticos Hestenes e Stiefel em 1952, o método de Gradientes Conjugados vem como uma alternativa aos métodos de Newton e Cauchy.

Primeiramente, sejam $A \in \mathbb{R}^{n\times n}$ uma matriz simétrica e definida positiva e a função $f : \mathbb{R}^n \to \mathbb{R}$, com $f(x) = \frac{1}{2}x^T Ax + b^T x + c$. Resolveremos, claro, o problema (P):

$$(P) \quad \min_{x\in\mathbb{R}^n} f(x)$$

Note que f tem sua lei de formação no formato quadrático. Com isso, a matriz Hessiana de f será $\nabla^2 f(x) = A$

O método tem sua base na utilização de direções conjugadas em relação à matriz A. Mais especificamente, na definição e no resultado que seguem abaixo.

Definição: Seja $A \in \mathbb{R}^{n\times n}$ uma matriz simétrica e definida positiva. Duas direções $d^1, d^2 \in \mathbb{R}^n$ são A-conjugadas se vale que

$$(d^1)^T A d^2 = 0$$

Propriedade[1]**:** Dada a matriz $A \in \mathbb{R}^{n \times n}$ simétrica e definida positiva (Hessiana da função f a ser minimizada), é possível escolher n direções linearmente independentes e duas a duas A-conjugadas tais que o algoritmo de descida que as utiliza separadamente converge em n passos para resolver o problema (P).

Observação: Note que na propriedade acima, n é a dimensão da matriz A.

Com este resultado, garantimos a convergência do nosso método. Entretanto, como devemos escolher tais direções A-conjugadas? Uma das formas de fazer isso é calcular a direção d^{k+1} em função da d^k utilizando a fórmula inclusa no algoritmo abaixo.

Algoritmo 6: Método de Gradientes Conjugados (Caso Quadrático)

Entrada: $x_0 \in \mathbb{R}^n, \epsilon > 0, k_{max} = 10000, time_{max} = 30(s)$
Saída: $\bar{x} \in \mathbb{R}^n$

início
  $k = 0$
  $d^0 = -\nabla f(x^0)$
  $A = \nabla^2 f(x^0)$
  repita
    $t_k = -\frac{\nabla f(x^k)^T \cdot d^k}{(d^k)^T \cdot A \cdot d^k}$
    $x^{k+1} = x^k + t_k \cdot d^k$
    $\beta_k = \frac{(d^k)^T \cdot A \nabla f(x^{k+1})}{(d^k)^T \cdot A \cdot d^k}$
    $d^{k+1} = -\nabla f(x^{k+1}) + \beta_k d^k$
    $A = \nabla^2 f(x^{k+1})$
    $k = k + 1$
  até $\left\|\nabla f(x^k)\right\| < \epsilon$ *ou* $k > k_{max}$ *ou* $time > time_{max}$;
  retorna x^k
fim

Note que as direções geradas em cada iteração são de descida. De fato:

$$\nabla f(x^k)^T d^k = \nabla f(x^k)^T(-\nabla f(x^k) + \beta_{k-1} d^{k-1}) = -\|\nabla f(x^k)\|^2 < 0$$

Observação: Perceba que a última desigualdade é estrita pois $\nabla f(x^k) \neq \bar{0}$

2.5.1. Método de Gradientes Conjungados - Caso Não Quadrático

O procedimento do método para o caso não quadrático é similar ao utilizado para o caso quadrático. Entretanto, note que a propriedade que garante a convergência do método em n iterações não vale, já que nossa função objetivo não é mais da forma quadrática, e portanto, não valerá que $\nabla^2 f(x) = A$, para alguma matriz $A \in \mathbb{R}^{n \times n}$.

Contudo, o fator β_k de atualização das direções de descida depende da matriz A. Então como podemos provar a convergência do método para funções não quadráticas?

Pensando em resolver este problema, os matemáticos Polak e Ribière desenvolveram uma fórmula para gerar, a cada iteração, uma nova direção de descida. Ela é dada por

$$\beta_k^{PR} = \frac{\nabla f(x^{k+1})^T [\nabla f(x^{k+1}) - \nabla f(x^k)]}{\nabla f(x^k)^T \nabla f(x^k)}$$

A partir disso, basta agora reiniciar este parâmetro β_k para garantir a convergência do método. Como faremos isso?

[1]Fonte: CUNHA, M. C. C. **Métodos Numéricos:** 2 Ed. Campinas: Unicamp, 2000. p.66

Estratégia: Sabemos que para funções quadráticas o método converge em n passos, sendo n a dimensão do vetor viável ao problema ($x \in \mathbb{R}^n$). Portanto, basta reiniciar o parâmetro sempre que o algoritmo alcançar um número de iterações múltiplo de n, ou seja, fazer $\beta_k = 0$.

Algoritmo 7: Método de Gradientes Conjugados (Caso Não Quadrático)

Entrada: $x_0 \in \mathbb{R}^n, \epsilon > 0, k_{max} = 10000, time_{max} = 30(s)$
Saída: $\bar{x} \in \mathbb{R}^n$

início
  $k = 0$
  $d^0 = -\nabla f(x^0)$
  repita
    $t_k : f(x^k + t_k d^k) < f(x^k)$ (Busca Linear por Armijo)
    $x^{k+1} = x^k + t_k d^k$
    se *(k+1) modn* $\neq 0$ **então**
      Determine β_k por β_k^{PR};
    senão
      $\beta_k = 0$;
    fim
    $d^{k+1} = -\nabla f(x^{k+1}) + \beta_k d^k$
    $k = k + 1$
  até $\left\|\nabla f(x^k)\right\| < \epsilon$ *ou* $k > k_{max}$ *ou* $time > time_{max}$;
  retorna x^k
fim

2.5.2. Convergência do Método de Gradientes Conjugados

Como vimos, se procuramos minimizar uma função quadrática, o algoritmo converge em n passos. Entretanto, na prática, isso depende do número de condicionamento da matriz Hessiana (A).

Caso a matriz A seja mal condicionada, teremos uma perda de precisão no parâmetro β_k e nas direções A-conjugadas, ou seja, podemos ter

$$d_i A d_j \neq 0, \quad \text{quando } i \neq j$$

Especificamente, nesses casos, o Método de Gradientes Conjugados tem uma convergência aproximadamente sublinear, ou seja, praticamente equivalente à convergência do Método de Cauchy.

3. Comparações de eficiência: Projeto 1

Com o objetivo de comparar as características teóricas e computacionais de nossos métodos, testamos os algoritmos implementados a dez problemas da biblioteca CUTEst. Segue abaixo as tabelas com os resultados; as conclusões obtidas estarão logo depois.

	Q-N	N-M	N-P	Cauchy	Q-N	N-M	N-P	CAUCHY
	Convergiu?				Critério de parada obtido			
HILBERTA	SIM	SIM	SIM	SIM	-	-	-	-
BOXBODLS	NÃO	NÃO	SIM	NÃO	max $_{\text{iter. a}}$	max $_{\text{iter. a}}$	-	max $_{\text{iter. m}}$
HIMMELBB	SIM	SIM	SIM	NÃO	-	-	-	max $_{\text{iter. m}}$
SISSER	SIM	SIM	SIM	NÃO	-	-	-	max $_{\text{iter. m}}$
SINEVAL	SIM	SIM	SIM	NÃO	-	-	-	max $_{\text{iter. m}}$
ROSENBR	SIM	SIM	SIM	NÃO	-	-	-	max $_{\text{iter. m}}$
HAIRY	NÃO	NÃO	NÃO	NÃO	max $_{\text{iter. a}}$	max $_{\text{iter. a}}$	max $_{\text{it. m}}$	max $_{\text{iter. a}}$
MARATOSB	NÃO	NÃO	SIM	NÃO	max $_{\text{iter. a}}$	max $_{\text{iter. a}}$	-	max $_{\text{iter. m}}$
POWELLBSLS	SIM	SIM	SIM	NÃO	-	-	-	max $_{\text{iter. m}}$
HIMMELBG	SIM	SIM	SIM	SIM	-	-	-	-

Tabela 1: *Mais à esquerda está o nome de cada problema do CUTEst, seguido então do status de sua convergência, juntamente com o respectivo motivo caso o algoritmo não tenha convergido.*

Legenda:

- *max iter. a: o algoritmo atingiu o máximo de iterações no algoritmo de Armijo, isto é, o* t_k *ficou pequeno demais e mesmo assim não entrou na região de decréscimo da função.*

- *max iter. m: o método utilizado atingiu seu número de iterações máximo* ($k > k_{max}$).

	Quase Newton	Newton Modificado	Newton Puro	Cauchy
	Solução Encontrada			
HILBERTA	[-5.82e-8, 9.79e-7]	[-3.57e-7, -3.57e-7]	[-1.3e-15, 2.6e-15]	[7.16e-6, -1.33e-5]
BOXBODLS	[213.8090, 5.47e-1]	[213.8090, 5.47e-1]	[172.50, 24.072]	[202.14, 6.35e-1]
HIMMELBB	[4.02e-10, 5.83786]	[-5.80e-3, 6.63e-3]	[-4.64e-3, 5.54e-3]	[-4.14e-1, 8.65e-1]
SISSER	[3.81e-3, 6.77e-4]	[4.21e-3, 4.21e-4]	[3.42e-3, 3.42e-4]	[1.50e-2, 1.22e-2]
SINEVAL	[4.96e-8, 4.99e-8]	[5.36e-8, 5.39e-8]	[7.11e-14, 7.10e-14]	[2.20864, 8.031e-1]
ROSENBR	[1.00000, 1.00000]	[1.0000, 1.0000]	[1.0000, 1.0000]	[7.85e-1, 6.16e-1]
HAIRY	[2.19e-9, -1.21e-9]	[-1.30e-9, 8.83e-11]	[4.73e-8, -92551.0]	[-2.04e-9, -8.54e-9]
MARATOSB	[-1.0000, 2.13e-8]	[-1.000, 4.38e-8]	[-1.0000, 1.05e-11]	[9.95e-1, 9.05e-2]
POWELLBSLS	[1.09e-5, 9.10615]	[1.09e-5, 9.10615]	[1.00e-4, -1.00e-4]	[1.00e-4, 1.00001]
HIMMELBG	[8.61e-8, -1.00e-7]	[-1.19e-7, 7.92e-8]	[1.2000, 8.00e-1]	[2.02e-7, -4.34e-11]

Tabela 2: *Nesta tabela vemos a solução encontrada por cada método para o problema tratado. Perceba como a solução para os métodos que não convergiram estão longe da ideal (apresentada pelos métodos que convergiram).*

	Q-N	N-M	N-P	Cauchy	Q-N	N-M	N-P	Cauchy
	$\|\nabla g_x\|$				Δt = variação do tempo			
HILBERTA	5.23e-7	6.13e-7	2.22e-16	9.98e-7	3.59e-5	5.60e-1	5.31e-1	3.50e-3
BOXBODLS	1.80e-4	4.69e-5	7.69e-7	44.2062	1.02e-3	1.62e-2	2.8e-2	1.04e-1
HIMMELBB	6.41e-7	6.89e-7	3.76e-7	28.0847	3.67e-4	5.58e-3	3.43e-3	6.67e-2
SISSER	6.72e-7	9.07e-7	4.84e-7	5.99e-5	3.88e-4	3.52e-3	2.08e-3	5.56e-2
SINEVAL	9.36e-7	9.31e-7	3.40e-13	9.48e-1	9.55e-4	1.08e-2	4.96e-4	6.48e-2
ROSENBR	6.26e-7	2.01e-6	8.28e-9	7.21e-1	7.86e-4	7.18e-3	1.08e-3	6.96e-2
HAIRY	1.25e-5	6.66e-6	110.6347	6.68e-6	2.00e-3	1.22e-2	1.0352	9.28e-3
MARATOSB	1.76e-5	5.57e-5	4.866e-7	9.05e-2	1.18e-2	1.12e-1	1.10e-2	5.12e-2
POWELLBSLS	5.47e-7	7.08e-7	2.13e-14	2.70e-1	2.09e-3	1.93e-2	2.44e-3	7.55e-2
HIMMELBG	6.92e-7	6.74e-7	1.02e-8	8.08e-7	2.39e-4	2.58e-3	7.91e-4	2.36e-4

Tabela 3: *À esquerda vemos a comparação entre a norma do gradiente da função objetivo no suposto ponto ótimo, encontrado pelo algoritmo. Note que a tolerância de nossos algoritmos era de* 10^{-6}*. Logo, podemos relacionar os resultados em vermelho com os casos em que o método não convergiu.*

	Q-N	N-M	N-P	Cauchy	Q-N	N-M	N-P	Cauchy
	Memória Utilizada				iterações			
HILBERTA	91.70 KiB	188.883 KiB	8.148 KiB	854.609 KiB	30	23	1	682
BOXBODLS	253.5 KiB	898.539 KiB	1.408 MiB	12.217 MiB	74	104	196	10001
HIMMELBB	121.3 KiB	401.148 KiB	177.234 KiB	12.220 MiB	35	49	24	10001
SISSER	111.7 KiB	246.031 KiB	103.719 KiB	12.212 MiB	36	30	14	10001
SINEVAL	287.2 KiB	649.570 KiB	15.500 KiB	12.215 MiB	91	79	2	10001
ROSENBR	189.7 KiB	409.313 KiB	44.906 KiB	12.215 MiB	59	50	6	34
HAIRY	551.7 KiB	904.398 KiB	63.854 MiB	1.261 MiB	166	106	10001	1010
MARATOSB	3.799 MiB	8.135 MiB	522.758 KiB	12.222 MiB	1247	155	71	10001
POWELLBSLS	574.2 KiB	1.237 MiB	118.422 KiB	12.223 MiB	184	21	16	10001
HIMMELBG	73.20 KiB	172.148 KiB	30.203 KiB	59.359 KiB	23	1016	4	45

Tabela 4: *Agora, comparamos a eficiência computacional dos métodos. Perceba que, quando o método de Cauchy não converge, sua memória em CPU se torna muito alta quando comparado aos outros métodos para a mesma função. Além disso, note como as informações dessa tabela estão relacionadas com a variação de tempo, da tabela 3.*
Nota: 1 MiB = 1024 KiB.

4. Conclusões do Projeto 1

Após a base teórica apresentada e a visualização dos dados em problemas reais (CUTEst), podemos tomar algumas conclusões sobre os métodos. Apesar de não estarmos lidando com problemas tão grandes (isso é perceptível devido ao número de iterações, por exemplo), os resultados são interessantes:

1. O Método de Cauchy é inviável para a maioria dos casos. Isso se dá por conta de sua convergência sublinear. Como vimos nas tabelas acima, são necessárias muitas iterações para o método convergir, e muitas vezes, nem isso acontece. Apesar de ser um método de fácil implementação, isso não é compensado devido ao alto gasto computacional.

2. Newton Puro convergiu em uma iteração ao tentar resolvar o problema "HILBERTA". Isso nos diz que possivelmente este problema tem sua função objetivo na forma quadrática.

3. Os métodos Quase Newton, Newton Modificado e Cauchy não convergiram no problema "MARATOSB". Isso porque este problema é direcionado a realizar o Efeito Maratos com os métodos irrestritos, de forma a impedi-los de alcançar sua convergência superlinear, e assim prejudicar a eficiência do método. Note que para Quase Newton e Newton Modificado, o critério de parada foi o máximo de iterações na busca linear de Armijo. Perceba que o método de Newton Puro convergiu após várias iterações, utilizando bastante memória, e não falhou justamente porque não há buscar linear neste método.

4. Ao comparar as tabelas de memória, confirmamos que o método irrestrito Quase Newton utiliza menos memória que o Newton Modificado. Isso já era esperado, pois como vimos, o Quase Newton foi desenvolvido justamente para compensar o alto custo computacional que existe ao calcular a Matriz Hessiana da função objetivo.

5. Note como a norma do gradiente da função objetivo é muito menor para o Newton Puro do que para os outros métodos. Essa peculiaridade pode ser explicada por conta da convergência quadrática do método. Além disso, como o Newton Modificado e o Quase Newton têm buscas lineares inexatas, seu método de convergência é o *backtracking*, então a norma do gradiente estará, geralmente, entre 10^{-6} e 10^{-7}, ou seja, entre a tolerância e $(10^{-1} * \mathrm{tol})$.

6. Enfim, com base nos resultados obtidos, o melhor método a ser utilizado é o Newton Puro. Além de garantir sua convergência em 90% dos casos, o gasto em memória, iterações e tempo é menor do que quando comparado aos outros métodos. Entretanto, não significa que este método será sempre o melhor, independente do problema escolhido para resolver. Note que no problema "HAIRY", por exemplo, o método não convergiu: atingiu seu máximo de iterações, e com isso, utilizou uma grande quantidade de memória.

5. Introdução do Projeto 2

Na segunda parte do projeto lidaremos com problemas restritos, no qual temos como objetivo resolver o problema:

$$(P) \quad \min_{x \in \Omega} f(x),$$

em que $f : \mathbb{R}^n \to \mathbb{R}$ é chamada de **função objetivo** do problema (P).

Neste caso, o problema possui restrição, ou seja, teremos $\Omega \neq \mathbb{R}^n$.

As três possíveis restrições são: igualdade, desigualdade e caixa. Porém neste trabalho estaremos interessados no estudo dos problemas com restrição de igualdade, da forma:

$$(P) \quad \begin{array}{l} \min f(x) \\ \text{s.a. } h(x) = 0, \end{array}$$

em que $h : \mathbb{R}^n \to \mathbb{R}^m$ é a restrição do problema (P).

O problema em questão consiste em encontrar a distância mínima de um ponto $p \in \mathbb{R}^n$ ao conjunto $C = \{x \in \mathbb{R}^n : a^T x = b\}$, cuja representação é uma reta ou um plano, quando em duas dimensões ou 3 dimensões, respectivamente. No caso geral, chamamos de hiperplano. Podemos descrevê-lo como:

$$(PC) \quad \begin{array}{l} \min \dfrac{1}{2} \|x - p\|^2 \\ \text{sujeito a } a^T x = b \end{array}$$

Por KKT, podemos encontrar uma solução fechada para o problema (PC), dada por:

$$(S) \quad \bar{x} = p + \frac{b - a^T p}{a^T a} a$$

Para tal, implementaremos o método de Lagrangeano Aumetado para resolvê-lo. Note durante a fundamentação teórica que a cada passo do algoritmo citado serão gerados subproblemas irrestritos, os quais resolveremos com os métodos implementados no Projeto 1.

6. Fundamentação Teórica

Primeiramente, para enunciar o algoritmo do método de Lagrangeano Aumentado, precisamos entender as condições de KKT para restrições de igualdade, já que elas serão utilizadas como critério de convergência do método.

6.1. Condições de KKT para restrições de igualdade

Desenvolvidas por William Karush, Harold W. Kuhn, e Albert W. Tucker, as condições de KKT são necessárias para que $\bar{x} \in \mathbb{R}^n$ seja solução de (P).

Para que as condições de KKT sejam válidas, precisamos que nosso problema satisfaça alguma condição de qualificação. Geralmente, nossos exemplos satisfarão a Condição de Independência Linear de Qualificação (LICQ) em $\bar{x}$, como enunciado abaixo:

$$(LICQ) \qquad \nabla f(\bar{x}) \ // \ \nabla h(\bar{x}) \Leftrightarrow \exists \ \lambda \in \mathbb{R}^m \text{ tq } \nabla f(\bar{x}) = -\lambda^T \nabla h(\bar{x})$$

Isso acontece sempre que as componentes de h são linearmente independentes.

Também precisaremos definir a função Lagrangiana:

$$l : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$$

$$l(x, \lambda) = f(x) + \lambda^T h(x)$$

Agora que temos todas as ferramentas necessárias, podemos enunciar as condições de KKT: Se $\bar{x}$ é uma solução do problema (P) e $\bar{x}$ satisfaz $(LICQ)$, então existe $\bar{\lambda} \in \mathbb{R}^m$ tal que:

$$\begin{cases} \nabla_x l(\bar{x}, \bar{\lambda}) = \bar{0} \to \text{Condição de Otimalidade} \\ h(\bar{x}) = \bar{0} \to \text{Condição de Viabilidade} \end{cases}$$

A condição de otimalidade garante que $\bar{x}$ é um ponto estacionário. Já a condição de viabilidade garante que $\bar{x}$ está dentro do conjunto viável, ou seja, satisfaz a restrição do problema.

Perceba que as condições de KKT são necessárias para que $\bar{x}$ seja uma solução, mas não suficientes. Ao seguir as restrições de KKT para resolver um problema com um conjunto viável compacto, encontraremos ao menos duas soluções (sabemos disso pelo Teorema de Weierstrass). Entretanto, uma será o ponto de máximo, e a outra o de mínimo, que resolve nosso problema. Cabe a nós (ou ao algoritmo) identificar cada caso.

Compreendidos os conceitos de KKT, podemos seguir adiante.

6.2. Mais fundamentação teórica

Este método para resolver problemas restritos consiste em, além de usar KKT como condição de parada, utilizar a Penalização como fator corretor da viabilidade da solução. A ideia é construir subproblemas irrestritos (que, pelo Projeto 1, sabemos resolver) para encontrar a solução do problema (P). Mas como isso funciona?

Neste caso, utilizaremos a Penalização Externa, pois estamos tratando de um problema com restrição de igualdade. Nela, definimos uma função, que chamaremos de Penalidade. Começando com um $x_0 \in \mathbb{R}^n$ fora do conjunto viável, a função Penalidade aplicará, por meio da Barreira e do Parâmetro de penalidade, um deslocamento em x_0, de forma a deixá-lo mais perto do conjunto viável e mais próximo da solução ideal. Este ponto citado será o x_1, e o mesmo será feito com ele. Ao longo do processo, examinaremos a convergência da sequência $\{x_k\}_{n \in \mathbb{N}}$, e teremos que

$$\lim_{k \to \infty} x_k = \bar{x}$$

é a solução do problema. Na prática, em vez de analisar a sequência, definimos uma tolerância perto de zero e paramos quando o nosso ponto x_k satisfazer KKT para esta tolerância.

Para problemas do tipo (P), ou seja, com restrição de igualdade, utilizaremos a barreira externa quadrática. É comum utilizar a seguinte função penalidade:

$$P : \mathbb{R}^n \times \mathbb{R}_{++}$$

$$P_\rho(x) = f(x) + \frac{\rho}{2} \|h(x)\|^2$$

em que $\mathbb{R}_{++} = \{x \in \mathbb{R} : x > 0\}$ e $\rho \in \mathbb{R}$ é o parâmetro de penalidade, que é atualizado a cada iteração.

Agora que temos todos os conhecimentos necessários, podemos finalmente estudar o método:

6.3. O Método Lagrangeano Aumentado

Proposto por Hestenes e Powell na década de 1960, o algoritmo se baseia no resultado de que os seguintes problemas são equivalentes (ou seja, apresentam mesmo conjunto solução):

$$(PI) \qquad \min_{\text{s.a. } h(x)=\bar{0}} f(x) \qquad \Longleftrightarrow \qquad \min_{\text{s.a. } h(x)=\bar{0}} f(x)+\lambda^T h(x) \qquad (PIL)$$

Após isso, uniram a função Lagrangeano com a função Penalidade, definindo L, a função Lagrangeano Aumentado:

$$L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}_{++} \to \mathbb{R}$$

$$L(x,\lambda,\rho) = f(x) + \lambda^T h(x) + \frac{\rho}{2}\,\|h(x)\|^2$$

Entretanto, na função Lagrangeano, atualizamos o λ, ou seja, o multiplicador de Lagrange. Dessa vez, o parâmetro de penalidade está sendo atualizado. Pensando em um meio efetivo de atualizar tal variável, Hestenes e Powell olharam para o KKT dos problemas (PI) e (PIL), de modo a analisar o KKT de (PIL) para encontrar um λ que satisfizesse a condição de otimalidade de (PI).

Abaixo está o algoritmo Lagrangeano Aumentado, que contém todas as ideias apresentadas:

- Critério de parada: KKT
- Atualização do ρ (parâmetro de penalidade)
- Atualização do λ (multiplicador de Lagrange)

Algoritmo 8: Método Lagrangeano Aumentado

Entrada: $x_0 \in \mathbb{R}^n, \rho_0 > 0, \lambda^0 \in \mathbb{R}^m; \epsilon_1, \epsilon_2 > 0, k_{max} = 1000, time_{max} = 30(s)$
Saída: $\bar{x} \in \mathbb{R}^n$

1 **início**
2 $\quad k = 0$
3 $\quad$ **repita**
4 $\quad\quad x^k = \min\{L(x,\lambda_k,\rho_k) : x \in \mathbb{R}^n\}$
5 $\quad\quad \lambda^{k+1} = \lambda^k + \rho_k h(x^k)$
6 $\quad\quad$ **se** $\left\|h(x^k)\right\| \geqslant 0.1 \left\|h(x^{k-1})\right\|$ **então**
7 $\quad\quad\quad \rho_{k+1} > \rho_k$
8 $\quad\quad$ **fim**
9 $\quad\quad k = k+1$
10 $\quad$ **até** $\left(\left\|\nabla L(x^k)\right\| < \epsilon_1 \; e \; \|\nabla h(x^k) < \epsilon_2\right)$ *ou* $k > k_{max}$ *ou* $time > time_{max}$;
11 $\quad$ **retorna** x^k
12 **fim**

Na linha 4 temos nosso subproblema irrestrito a ser resolvido pelos métodos do Projeto 1. Note também, nas linhas 6 e 7, que o parâmetro de penalidade é atualizado somente se a norma da função de restrição, h, não tiver diminuído o suficiente com relação ao x da última iteração. Isso significa que é necessário aplicar uma penalidade maior, para que o próximo x fique mais próximo do conjunto viável.

7. Comparações de Eficiência: Projeto 2

Finalizada a base teórica do método, resta agora apresentar os resultados obtidos ao testar o algoritmo com os problemas selecionados. Aplicamos nosso código a três problemas do tipo (PC), apresentado na introdução do capítulo, com mudanças apenas nos valores das variáveis. São eles:

$$(P1) \qquad x \in \mathbb{R}^2, b = 8, a = (2,1)^T \text{ e } p = (10,7)^T$$

$$(P2) \qquad x \in \mathbb{R}^2, b = 8, a = (-3/5,2)^T \text{ e } p = (10,7)^T$$

$$(P3) \qquad x \in \mathbb{R}^{10}, b = 10, a = (5,5,5,5,5,5,5,5,5,5)^T \text{ e } p = (1,2,3,4,5,6,7,8,9,10)^T$$

PROBLEMA 1						
	Status	Solução	$\|\nabla L(\bar{x})\|$	$\|\nabla h(\bar{x})\|$	Iterações	Tempo
Cauchy	convergiu	[2.4, 3.2]	2.06e-7	2.23e-8	9	1.85e-1
Newton Puro	convergiu	[2.4, 3.2]	5.35e-7	1.19e-7	8	2.5850
Newton Modificado	convergiu	[2.4, 3.2]	5.35e-7	1.19e-7	8	2.80e-1
Quase Newton	convergiu	[2.4, 3.2]	5.35e-7	1.19e-7	8	2.89e-1

PROBLEMA 2						
	Status	Solução	$\|\nabla L(\bar{x})\|$	$\|\nabla h(\bar{x})\|$	Iterações	Tempo
Cauchy	convergiu	[10.0, 7.0]	9.20e-7	1.12e-8	7	9.99e-4
Newton Puro	convergiu	[10.0, 7.0]	3.27e-7	3.92e-8	7	2.2200
Newton Modificado	convergiu	[10.0, 7.0]	3.27e-7	3.92e-8	7	9.99e-4
Quase Newton	convergiu	[10.0, 7.0]	8.93e-7	1.15e-8	10	9.99e-4

PROBLEMA 3						
	Status	Solução	$\|\nabla L(\bar{x})\|$	$\|\nabla h(\bar{x})\|$	Iterações	Tempo
Cauchy	não convergiu	[-1.2, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	459.48	29.0	0	2.60e-2
Newton Puro	convergiu	[-4.3, -3.3, -2.3, -1.3, -0.3, 0.7, 1.7, 2.7, 3.7, 4.7]	5.97e-8	3.77e-9	4	9.4939
Newton Modificado	convergiu	[-4.3, -3.3, -2.3, -1.3, -0.3, 0.7, 1.7, 2.7, 3.7, 4.7]	5.97e-8	3.77e-9	4	9.99e-4
Quase Newton	convergiu	[-4.3, -3.3, -2.3, -1.3, -0.3, 0.7, 1.7, 2.7, 3.7, 4.7]	6.44e-7	2.22e-8	4	9.99e-4

Note que as iterações da tabela dizem respeito às iterações do método Lagrangeano Aumentado, não do subproblema. Além disso, perceba que as soluções (nos casos em que o algoritmo convergiu) correspondem com a solução (S), apresentada na introdução e que pode ser comprovada por KKT.

Observação: Em nosso algoritmo, utilizamos $\epsilon_1 = \epsilon_2 = 10^{-6}$, ou seja, a tolerância para as normas é de 10^{-6}.

Observação 2: Nesses casos, é desnecessário calcular o erro da solução encontrada, já que todas (com exceto de uma) convergiram para a solução exata, e portanto, o erro será zero. Nesta única exceção o erro é dado por:

$$\varepsilon = \| (-1.2, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0) - (-4.3, -3.3, -2.3, -1.3, -0.3, 0.7, 1.7, 2.7, 3.7, 4.7) \|$$

Dessa forma, $\varepsilon = 8.39166$.

8. Conclusões

Apesar dos três problemas selecionados serem pequenos ($x \in \mathbb{R}^2$ ou $\mathbb{R}^{10}$), algumas observações e conclusões importantes podem ser obtidas:

1. O método de Cauchy não convergiu no (P3). Note que como há 0 iterações, o algoritmo do Lagrangeano Aumentado não foi capaz de resolver o subproblema pelo método de Cauchy. Isso já era esperado, pois sabemos como a convergência de Cauchy é precária.

2. A solução do (P2) coincide com o próprio ponto p. Isso porque o ponto p está dentro do conjunto viável. Logo, o x que minimiza $\|x - p\|^2$ é o próprio p.

   De fato, um ponto $c \in \mathbb{R}^2$ está dentro do conjunto viável de (P2) se e somente se, $a^T c = b$, para $a = (-3/5, 2)^T$ e $b = 8$. Sendo $c = p$, temos:

   $$a^T c = \langle (-3/5, 2), (10, 7) \rangle = -6 + 14 = 8 = b$$

3. O tempo de CPU (em segundos) para chegar à solução de cada problema é maior quando o algoritmo do Lagrangeano Aumentado resolve o subproblema por Newton Puro. Este é um resultado que não esperávamos, já que nos resultados do Projeto 1 o Newton Puro foi o método mais efetivo para solucionar os problemas irrestritos. Entretanto, esses são apenas problemas em $\mathbb{R}^2$ e $\mathbb{R}^{10}$. Se mesmo com dimensões pequenas o tempo já esta alto, conjectura-se que esse tempo aumenta quando lidamos com problemas maiores. Ou seja: Newton Puro não parece ser uma boa opção para aplicar algoritmos de Penalidades Externas.

4. Apesar de não aparecer nas tabelas, calculamos a memória utilizada em CPU para cada um dos problemas e respectivos métodos do Projeto 1. Não colocamos essa informação na tabela pois como se trata de problemas pequenos, o gasto de memória estava bem próximo para todos os métodos. Mesmo com o tempo de CPU alto para Newton Puro, seu gasto em memória foi equivalente aos outros métodos apresentados. O mesmo vale para Cauchy: apesar de não convergir no (P3), não houve um salto gigantesco com relação aos dados do restante dos métodos.

5. Enfim, **qual método irrestrito é melhor?** Bem, juntando as conclusões dos projetos:

   Apesar de converger muito bem para os problemas irrestritos do CUTEst, o Newton Puro não apresentou uma boa performance quando aplicado a problemas restritos.

   Além disso, sabemos que Cauchy não é uma boa opção. Os motivos são óbvios e foram explicitados durante as conclusões.

   Também, na conclusão do Projeto 1, vimos que a frequência de convergência do Quase Newton e do Newton Modificado é a mesma. Entretanto, o método Quase Newton apresenta um gasto menor de memória, e isso pode ser valioso ao tratar de problemas grandes. A matemática aplicada está sempre tentando diminuir gastos computacionais: o método Quase Newton foi desenvolvido para isso e parece funcionar bem. Ainda que mediano quando comparado ao Newton Puro nos problemas do CUTEst, tal método apresentou boa convergência quando aplicado aos problemas restritos.

Apesar de apresentar mais iterações quando comparado aos outros métodos, isso é óbvio: o método se baseia na aproximação da matriz Hessiana, e por isso, levará mais iterações para convergir. É interessante perceber, entretanto, que apesar de mais iterações, o método realiza menos **alocações**. Claro, pois como vimos, o método utiliza menos memória. Comentamos exatamente isso durante a fundamentação teórica: calcular produtos entre matrizes é menos custoso computacionalmente do que calcular segundas derivadas (relacionadas à matriz Hessiana do método de Newton).

9. Considerações finais

De forma a finalizar o relatório, deixamos aqui nossas impressões sobre a realização dos projetos:

Ambos os projetos foram muito importantes para o bom aproveitamento da disciplina: podemos aprender, ao implementar os códigos, assuntos que não foram abordados nas avaliações. Além disso, ao se deparar com erros durante a implementação, aprendemos como as hipóteses (matrizes simétricas e definidas positivas, por exemplo) são importantes para o bom funcionamento do algoritmo, isto é, o sucesso da convergência.

Até mesmo durante a realização do relatório: foi necessário explicar cada método, de modo que precisamos procurar mais sobre cada assunto e fazer testes com nossos algoritmos. Dessa forma, desenvolvemos também a didática.

Não obstante, projetos como esses sempre instigam o saber matemático: ao obter os resultados práticos, ora confirmamos nossas hipóteses, ora obtemos conclusões que iam contra nossas expectativas, e claro, para cada peculiaridade dos métodos há um motivo.

10. Referências

[1] CUNHA, M. C. C. **Métodos Numéricos:** 2 Ed. Campinas: Unicamp, 2000.

[2] RIBEIRO A. A.; KARAS; E. W. **Otimização Contínua:** Aspectos teóricos e computacionais. São Paulo: Cengage Learning, 2013.