

Regiões de confiança para minimização irrestrita em problemas de cálculo variacional*

Larissa O. Xavier[†]

Sandra A. Santos

Departamento de Matemática Aplicada, IMECC, UNICAMP,
13083-970, Campinas, SP

E-mail: larissa@ime.unicamp.br, sandra@ime.unicamp.br,

1 Resumo

Neste trabalho apresentamos um estudo dos métodos de região de confiança (RC) para problemas de otimização irrestrita. O modelo quadrático adotado corresponde à aproximação de Taylor de segunda ordem para a função objetivo do problema original a cada iteração. Também são discutidos a escolha da região de confiança e o critério para a aceitação do ponto durante o processo iterativo. O raio da região de confiança inicial é calculado de duas maneiras, uma baseada na norma do gradiente e outra proposta por Sartenaer. Para a solução aproximada dos subproblemas é utilizado o algoritmo de Steihaug-Toint em suas formas original e preconditionada, com o preconditionador diagonal baseado no escalamento dinâmico proposto por Roma. O desempenho das duas abordagens é analisado na solução de problemas-teste do cálculo variacional, propostos por Gill e Murray, em uma implementação que conta apenas com produtos matriz-vetor, feita em `Matlab`.

2 Métodos RC para minimização irrestrita

O problema de minimização a ser resolvido durante este trabalho é dado por:

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

Utilizamos um processo iterativo para obter a solução aproximada através de um ponto inicial x_0 e gerando a sequência $x_{k+1} = x_k + s_k$, onde s_k é o passo dado na k -ésima iteração.

A cada iteração adotamos um modelo quadrático (m_k) correspondente à aproximação de Taylor de segunda ordem para a função objetivo (f) do problema original:

$$m_k(s) = f_k + g_k^T s + \frac{1}{2} s^T H_k s,$$

* Apoios FAPESP 02/13486-4 e CNPq 302412/2004-2.

[†] Bolsista de Iniciação Científica FAPESP no período de abril de 2003 a dezembro de 2004.

onde $f_k = f(x_k)$, $g_k = \nabla f(x_k)$ e $H_k = \nabla^2 f(x_k)$.

Resolvemos então o problema de minimização do modelo quadrático em uma região de confiança:

$$\begin{aligned} \min \quad & m_k \\ \text{s.a.} \quad & \|s\| \leq \Delta_k, \end{aligned} \quad (1)$$

onde Δ_k é o raio da região de confiança na k -ésima iteração.

Com este processo, a cada iteração obtemos o ponto $\bar{x} = x_k + \bar{s}$, onde \bar{s} é o minimizador do subproblema (1). Comparamos então o decréscimo da função objetivo com o decréscimo obtido pelo modelo quadrático, através de:

$$\rho_k = \frac{f(x_k) - f(\bar{x})}{m_k(0) - m_k(\bar{s})}.$$

Para a aceitação do passo verificamos se $\rho_k \geq \bar{\rho}$, com $\bar{\rho} \in (0, 0.5)$, evitando assim a aceitação de passos para os quais $\rho_k \approx 0$, o que indicaria uma má representação da função pelo modelo. No caso $\rho_k < \bar{\rho}$ diminuímos o raio da região de confiança e tomamos $x_{k+1} = x_k$.

Por outro lado, se $\rho_k \geq \bar{\rho}$ mas ρ_k não está próximo de 1 ($\rho_k \leq 1 - \epsilon$, com $\epsilon \in (0, 0.5)$), aceitamos o passo \bar{s} fazendo $x_{k+1} = x_k + \bar{s}$, mas não aumentamos o raio da região de confiança.

Se a função teve um decréscimo satisfatório com $\rho_k \approx 1$ ou até mesmo $\rho_k \gg 1$, casos em que é verificado $\rho_k > 1 - \epsilon$, aumentamos o raio da região de confiança e aceitamos o passo \bar{s} ($x_{k+1} = x_k + \bar{s}$).

Dessa forma teremos a atualização do raio fazendo $\Delta_{k+1} = \beta_k \Delta_k$ com

$$\beta_k = \begin{cases} \gamma_1^k & \text{se } \rho_k < \bar{\rho} \\ \gamma_2^k & \text{se } \rho_k > 1 - \epsilon \\ 1 & \text{caso contrário} \end{cases} \quad (2)$$

onde $0 < \bar{\rho} < 1 - \epsilon < 1$, $\gamma_1^k \in [\gamma_1, 1)$, $\gamma_2^k \in [1, \gamma_2]$ e $0 < \gamma_1 < 1 \leq \gamma_2$.

Para os testes computacionais $\epsilon = \bar{\rho} = 0.1$. As escolhas para γ_1^k e γ_2^k foram feitas baseadas no estudo de Conn, Gould e Toint [2].

2.1 Raio de região de confiança inicial

Para a escolha do raio da região de confiança inicial é utilizado dez por cento da norma do gradiente no ponto inicial, com o qual é feita uma comparação com o desempenho da proposta de Sartenaer [7]. A idéia principal na dedução do raio inicial de Sartenaer é determinar o raio máximo da região de confiança na qual a função objetivo esteja bem representada pelo modelo ao longo da direção de máxima descida ($-g_0$). Para a determinação do raio máximo é utilizada uma busca iterativa ao longo desta direção.

2.2 Solução dos subproblemas

Com o raio da região de confiança e o modelo para a função objetivo bem determinados, resolvemos o subproblema (1) aproximadamente com a implementação do algoritmo de Steihaug-Toint [8, 9] em suas formas original e preconditionada. Este algoritmo é baseado no método dos gradientes conjugados no contexto dos algoritmos de região de confiança, resolvendo os sistemas lineares que aparecem ao ser aplicado o método de Newton aos subproblemas (1). A principal diferença entre o método dos gradientes conjugados e o proposto por Steihaug e Toint é a colocação de dois critérios de parada extras no segundo método. O primeiro critério de parada extra é ativado se em uma determinada iteração do método dos gradientes conjugados obtivermos s fora da região de confiança ($\|s\| > \Delta_k$). Outro critério de parada é imposto quando uma direção de curvatura negativa é encontrada, ou seja, $s^T H_k s \leq 0$. Quando ocorre qualquer destes casos, encontramos o minimizador de (1) na fronteira da região de confiança.

Para a forma preconditionada do algoritmo de Steihaug-Toint é adotado o preconditionador diagonal baseado no escalamento dinâmico proposto por Roma [6]. A escolha deste preconditionador corresponde a um escalamento das colunas da matriz H_k , pretendendo obter um sistema em que a matriz seja equilibrada por colunas, isto é, seus vetores coluna tenham norma de magnitudes semelhantes.

3 O problema do cálculo variacional

Os problemas-teste do cálculo variacional consistem em encontrar a função $x^*(t)$ que minimiza o funcional

$$J(x(t)) = \int_0^1 f(t, x(t), \dot{x}(t)) dt \quad (3)$$

com condições de contorno $x(0) = x_a$ e $x(1) = x_b$.

Para resolver este problema, aproximamos $x(t)$ por uma combinação finita de funções base $w_i(t)$, $i = 1, \dots, N$:

$$x(t) = \sum_{i=1}^N c_i w_i(t) \quad \text{e} \quad \dot{x}(t) = \sum_{i=1}^N c_i \dot{w}_i(t).$$

Se as N funções base $\{w_i(t)\}_{i=1}^N$ geram o subespaço de dimensão finita W_N , então a forma discretizada do problema é dada por:

$$\min_{x(t) \in W_N} \int_0^1 f(t, x(t), \dot{x}(t)) dt = \min_{c \in \mathbb{R}^N} F(c),$$

onde

$$F(c) = \int_0^1 f \left(t, \sum_{i=1}^N c_i w_i(t), \sum_{i=1}^N c_i \dot{w}_i(t) \right) dt.$$

Para a escolha das funções base utilizamos funções polinomiais por partes $w(t)$ definidas em $[0, 1]$ tais que $w(t) \in C^1[0, 1]$ e, em cada subintervalo $[t_i, t_{i+1}]$ definido em $\Pi : 0 = t_0 < t_1 < \dots < t_{n+1} = 1$, $w(t)$ é um polinômio de grau três, como sugerido em [4]:

$$w_{2i+1}(t_j) = \delta_{ij} \delta_{00} = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j, \end{cases}$$

$$\frac{dw_{2i+1}(t_j)}{dt} = \delta_{ij} \delta_{01} = 0$$

e

$$w_{2i+2}(t_j) = \delta_{ij} \delta_{10} = 0,$$

$$\frac{dw_{2i+2}(t_j)}{dt} = \delta_{ij} \delta_{11} = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j, \end{cases}$$

ver Figura 1.

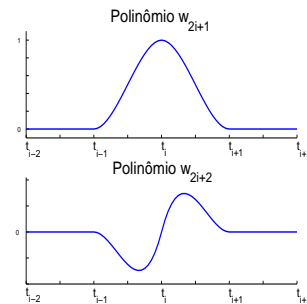


Figura 1: Representação dos polinômios w_{2i+1} e w_{2i+2} no intervalo $[t_{i-2}, t_{i+2}]$

Denotando cada componente do gradiente de $F(c)$ por $g_j(c) = \frac{\partial F(c)}{\partial c_j}$ temos:

$$g_j(c) = \int_0^1 [f_x(t, x(t), \dot{x}(t)) w_j(t) + f_{\dot{x}}(t, x(t), \dot{x}(t)) \dot{w}_j(t)] dt,$$

$j = 1, \dots, N$, onde $N = 2(n + 2)$.

A matriz Hessiana de $F(c)$ não é calculada explicitamente. Optamos por implementar o cálculo do produto da matriz Hessiana por um vetor $w = \nabla^2 F(c)v$ via diferenças finitas através da expressão:

$$w \approx \frac{\nabla F(c + hv) - \nabla F(c)}{h}$$

onde h é dado por $h = (1 + \|c\|)\sqrt{\text{eps}}$ e eps é a precisão do `Matlab` utilizado, que é aproximadamente 2.2204e-016.

Para o cálculo das integrais que aparecem na avaliação da função objetivo e do gradiente, utilizamos integração numérica via quadratura gaussiana de quatro pontos (cf. [3]).

Os elementos necessários para a solução dos problemas foram implementados modularmente por meio de um conjunto de funções programadas no `Matlab`, conforme esquema da Figura 2.

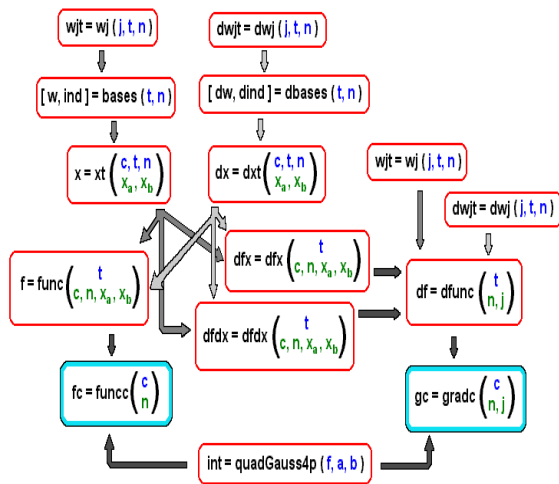


Figura 2: Esquema da dependência das funções implementadas em `Matlab`

4 Testes computacionais

Para os testes computacionais implementamos o método com região de confiança em `Matlab` para a resolução do problema de minimização da função $F(c)$, com gradiente $g(c)$, programados em `funcc` e `gradc`, respectivamente (Figura 2).

Para a implementação do algoritmo, para as iterações externas (método com região de confiança) e para as internas (algoritmo de Steihaug-Toint), utilizamos como critério de parada, respectivamente, que o processo se repita até que $\|g_k\| < \text{ep}\|g_0\|$ e $\|r_i\| < \text{epit}\|r_0\|$, onde $r_i = H_k s_i + g_k$ é o resíduo na i -ésima iteração do método de Steihaug-Toint, $r_0 = g_k$ e adotamos $\text{ep} = \text{epit} = 10^{-5}$. Além deste critério de parada,

as iterações externas são realizadas até que o número máximo de iterações 50 seja atingido. Para o caso das iterações internas do algoritmo de Steihaug-Toint, o número máximo de iterações adotado foi 1.2dim , onde dim é a dimensão do problema, que é igual ao número de componentes do vetor c ($\text{dim} = 2n + 2$). Este número máximo foi escolhido devido ao fato do método dos gradientes conjugados, em que Steihaug-Toint se baseia, ter a convergência atingida em no máximo dim iterações para o caso de resolução de problemas quadráticos como o subproblema (1). Escolhemos então o número máximo de iterações igual a 1.2dim , realizando, no pior dos casos, algumas iterações a mais do que dim , levando em consideração os erros causados pela precisão finita do `Matlab`, podendo ocorrer a convergência do algoritmo de Steihaug-Toint com mais de dim iterações.

Para os testes computacionais do método com região de confiança utilizamos quatro problemas de cálculo variacional extraídos de Gill e Murray [4].

Estes problemas consistem em minimizar o funcional (3).

O primeiro teste foi para o problema 2 de [4] no qual

$$f(t, x(t), \dot{x}(t)) = (\dot{x}(t))^2 - (x(t))^2 - 2tx(t)$$

e $x(0) = x(1) = 0$.

Este problema pode ser resolvido analiticamente através da equação de Euler-Lagrange¹, nos fornecendo

$$x^*(t) = \frac{\text{sen}(t)}{\text{sen}(1)} - t,$$

que será comparada com a solução numérica.

O segundo teste realizado foi para o problema 3 de [4] com

$$f(t, x(t), \dot{x}(t)) = (\dot{x}(t))^2 + (x(t))^2 + 2x(t)e^{2t}$$

e condições de contorno $x(0) = \frac{1}{3}$ e $x(1) = \frac{e^2}{3}$. A solução exata para este problema é dada por

$$x^*(t) = \frac{e^{2t}}{3}.$$

O terceiro teste é o problema 4 de [4] onde

$$f(t, x(t), \dot{x}(t)) = e^{-2(x(t))^2} ((\dot{x}(t))^2 - 1)$$

com $x(0) = 0$ e $x(1) = 1$.

O último teste foi o problema 5 de [4], onde

$$f(t, x(t), \dot{x}(t)) = (x(t))^2 + \dot{x}(t) \arctan(\dot{x}(t)) - \ln \sqrt{1 + (\dot{x}(t))^2}$$

com $x(0) = 1$ e $x(1) = 2$.

¹Para o problema $\min \int_a^b f(t, x(t), \dot{x}(t)) dt$, a solução satisfaz a equação de Euler-Lagrange $f_x - \frac{d}{dt}(f_{\dot{x}}) = 0$ com condições de contorno $x(a) = x_a$ e $x(b) = x_b$, conforme [5].

Tanto para o problema 4 quanto para o problema 5, a equação de Euler-Lagrange tem a forma $\ddot{x}(t) = 2x(t)(1 + \dot{x}^2)$ (cf. [1, p.35] e não possui solução analítica com as condições de contorno dadas. Neste casos, compararemos a solução numérica obtida pelo método com região de confiança com a solução obtida pela minimização de $F(c)$ pelo comando `fminunc` do `Matlab`.

Desempenho do preconditionador

Para os primeiros testes fixamos a dimensão em $2n + 2 = 40$ e o ponto inicial adotado foi $c_0 \in \mathbb{R}^{40}$ cujas componentes foram geradas aleatoriamente entre -0.1 e 0.1 .

Nas Tabelas 1 e 2 estão apresentados os resultados para os 4 problemas com o raio inicial $\Delta_0 = 0.1\|g_0\|$ e Δ_0 de Sartenaer, respectivamente. Nas tabelas temos os resultados para o método de região de confiança com a resolução dos sub-problemas através do método de Steihaug-Toint em sua forma original (ST) e sua forma preconditionada com a matriz de condicionamento de Roma (STp). Em todas as tabelas, P indica o problema, Mét o método utilizado, k o número de iterações externas, $\frac{it}{k}$ indica a média do número de iterações internas por iteração externa, pHv indica o número de produtos da matriz Hessiana por um vetor realizados, $\|g_k\|$ indica a norma do vetor gradiente avaliado no ponto final e f_k indica o valor da função objetivo no ponto final.

P	Mét	k	$\frac{it}{k}$	pHv	$\ g_k\ $	f_k
2	ST	2	38.5	156	6.83e-6	-0.0246
	STp	3	48	294	2.47e-5	-0.0246
3	ST	3	35.33	214	8.47e-6	16.3772
	STp	3	39	239	1.10e-3	16.3772
4	ST	6	43.33	525	1.80e-4	-0.1472
	STp	6	46.33	567	1.47e-4	-0.1472
5	ST	50	21.32	2150	3.84e-3	2.1389
	STp	36	12.22	921	5.75e-5	2.1388

Tabela 1: Resultados dos testes com $\Delta_0 = 0.1\|g_0\|$ e $c_0 \in \mathbb{R}^{40}$ com componentes geradas aleatoriamente entre -0.1 e 0.1 .

Comparando primeiramente o desempenho do algoritmo com Steihaug-Toint em sua forma preconditionada em relação ao da sua forma original, percebemos que na maioria destes testes o preconditionador de Roma não se mostrou eficiente, provocando um aumento tanto no número de iterações externas quanto internas, refletindo no maior número de avaliações de produto de matriz

P	Mét	k	$\frac{it}{k}$	pHv	$\ g_k\ $	f_k
2	ST	2	38.5	161	6.83e-6	-0.0246
	STp	3	48	299	2.47e-5	-0.0246
3	ST	1	48	102	9.13e-4	16.3772
	STp	3	48	299	3.66e-4	16.3772
4	ST	8	35.5	578	8.02e-5	-0.1472
	STp	7	37.71	545	1.37e-4	-0.1472
5	ST	50	22.04	2228	6.19e-4	2.1389
	STp	39	12.13	996	2.25e-5	2.1388

Tabela 2: Resultados dos testes com Δ_0 de Sartenaer e $c_0 \in \mathbb{R}^{40}$ com componentes geradas aleatoriamente entre -0.1 e 0.1 .

por vetor. A exceção foi o problema 5, para o qual o método com o algoritmo de Steihaug-Toint com o preconditionador de Roma teve um melhor desempenho do que o algoritmo em sua forma original. Para o raio inicial $\Delta_0 = 0.1\|g_0\|$ o método com Steihaug-Toint original realizou 2150 avaliações de produto da matriz Hessiana por um vetor, contra apenas 921 avaliações do método com Steihaug-Toint preconditionado. O mesmo ocorreu com Δ_0 de Sartenaer com $pHv = 2228$ para ST e $pHv = 996$ para STp. Com os dois raios iniciais o número máximo de iterações externas foi atingido com ST, enquanto que, com STp, este número abaixou consideravelmente.

Desempenho do raio inicial de Sartenaer

A segunda análise que pode ser feita a partir das tabelas é sobre o desempenho do método com a escolha do raio inicial de Sartenaer.

Comparando cada problema da Tabela 1 com o seu respectivo na Tabela 2, percebemos que a escolha do raio inicial de Sartenaer não é eficiente em todos os casos. Para o problema 2, obtivemos os mesmos resultados, com exceção do número de avaliações de produto da matriz Hessiana por vetor, o qual foi maior para Δ_0 de Sartenaer, devido às avaliações extras para o cálculo do raio inicial. Os resultados para este problema foram os mesmos pois o raio inicial Δ_0 de Sartenaer coincidiu com $0.1\|g_0\|$.

Para os problemas 5 com ST e STp, 4 com ST e 3 com STp, o raio inicial de Sartenaer provocou um aumento no número de avaliações de produto matriz por vetor, refletido pelo aumento em alguns casos do número de iterações externas ou internas, como foi o caso do problema 4 com ST, que passou de 6 iterações externas e $pHv = 525$ com $\Delta_0 = 0.1\|g_0\|$ para 8 iterações externas e $pHv = 578$ com Δ_0 de Sartenaer.

Já para os problemas 3 com ST e 4 com STp, tivemos o melhor desempenho para o método com Δ_0 de Sartenaer, com a diminuição do número de avaliações de produto da matriz Hessiana por vetor

de 214 e 567 com $\Delta_0 = 0.1\|g_0\|$ para 102 e 545 com Δ_0 de Sartenaer, respectivamente.

Dependência da dimensão e do ponto inicial

Com o intuito de analisar o comportamento do método com a alteração da dimensão e do ponto inicial, realizamos mais alguns testes com o método de Steihaug-Toint em sua forma original e o raio inicial $\Delta_0 = 0.1\|g_0\|$ para a resolução do problema 3. Optamos pela escolha do problema 3 pois em [4] a resolução deste problema não apresentou variações com a alteração da dimensão e ponto inicial.

Na Tabela 3 apresentamos os resultados com a variação da dimensão e na Tabela 4 a alteração do ponto inicial, onde c_0^1 é o vetor nulo de dimensão $Dim = 2n + 2 = 40$, $c_0^2 \in \mathbb{R}^{40}$ e $c_0^3 \in \mathbb{R}^{40}$ são vetores com componentes geradas aleatoriamente nos intervalos $(-0.1, 0.1)$ e $(-5, 5)$, respectivamente.

Dim	k	$\frac{it}{k}$	pHv	$\ g_k\ $	f_k
20	2	17	69	2.154e-4	16.37721
40	3	35.33	214	8.465e-6	16.37721
80	1	84	169	0.002286	16.37735

Tabela 3: Resultados dos testes para o problema 3 com $\Delta_0 = 0.1\|g_0\|$, $c_0 \in \mathbb{R}^{2n+2}$ com componentes geradas aleatoriamente entre -0.1 e 0.1 e com a resolução dos subproblemas pelo método de Steihaug-Toint original.

c_0	k	$\frac{it}{k}$	pHv	$\ g_k\ $	f_k
c_0^1	3	32.67	198	4.808e-6	16.37721
c_0^2	3	35.33	214	8.465e-6	16.37721
c_0^3	1	41	83	0.008634	16.37844

Tabela 4: Resultados dos testes para o problema 3 com $\Delta_0 = 0.1\|g_0\|$, $c_0 \in \mathbb{R}^{40}$ com a utilização do método de Steihaug-Toint original.

Analisando a Tabela 3 vemos que, com o aumento da dimensão de 20 para 40, houve um aumento do número de avaliações de produto matriz por vetor, embora a precisão atingida tenha sido melhor, com $\|g_k\| = 8.47 * 10^{-6}$. Já para a dimensão igual a 80, apesar do refinamento da malha ter sido maior, o que, teoricamente, nos forneceria uma solução mais precisa, obtivemos a norma do gradiente no ponto final maior que 10^{-3} e o valor da função objetivo no ponto final foi de aproximadamente 16.37735, enquanto que para a solução exata $x^*(t) = \frac{e^{2t}}{3}$, a função objetivo seria aproximadamente igual a 16.377212². Este fato de perda de precisão pode

$${}^2F(x^*) = \int_0^1 f(t, x^*(t), \dot{x}^*(t))dt = \int_0^1 \left\{ \left(\frac{2}{3}e^{2t}\right)^2 + \left(\frac{1}{3}e^{2t}\right)^2 + \frac{2}{3}e^{2t}e^{2t} \right\} dt = \int_0^1 \frac{11}{9}e^{4t} dt \approx 16.37721251013$$

ser atribuído a erros devidos à aritmética de ponto flutuante e à precisão do **Matlab**.

Agora, analisando a Tabela 4, percebemos que entre a escolha dos pontos iniciais c_0^1 e c_0^2 não há grandes diferenças, com uma pequena vantagem para o método com a escolha de c_0^1 com menos avaliações de produtos matriz por vetor. Já com a escolha de c_0^3 , temos menos iterações externas e produtos matriz por vetor, mas a precisão da solução foi menor, com $\|g_k\| > 10^{-3}$.

Qualidade das soluções

A próxima análise realizada foi para a verificação da qualidade das soluções obtidas. Para isso construímos os gráficos de $t \in [0, 1]$ por $x(t)$ obtido pelo método com região de confiança com Steihaug-Toint original, $\Delta_0 = 0.1\|g_0\|$ e $c_0 = c_0^2 \in \mathbb{R}^{40}$, comparado a $x(t)$ obtido pelo comando `fminunc` do **Matlab** e, no caso dos problemas 2 e 3, comparados também com a solução exata $x^*(t)$ obtida através da solução da equação de Euler-Lagrange. Os gráficos estão apresentados na Figura 3.

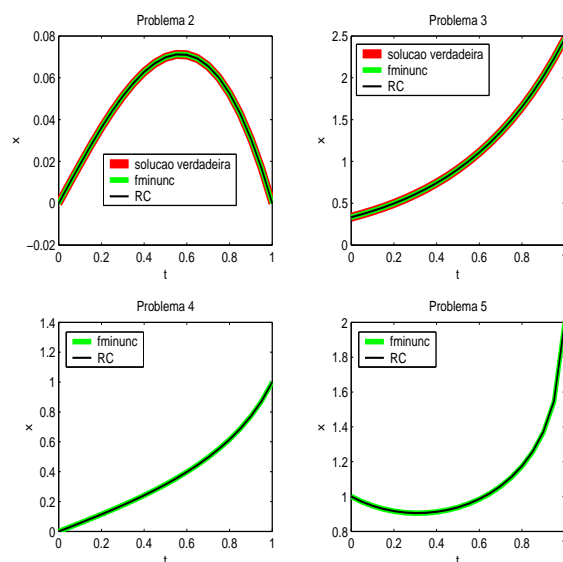


Figura 3: Comparação entre a solução verdadeira, a solução obtida pelo método com região de confiança e a solução obtida pelo comando `fminunc` do **Matlab**

Comparando graficamente as soluções percebemos que todas estão com uma qualidade satisfatória, sendo quase imperceptível as diferenças entre as soluções.

Para complementar esta análise comparamos o número de iterações externas realizadas pelo método de região de confiança (RC) e pelo método com a utilização do comando `fminunc` do **Matlab** (fM). Os resultados estão apresentados na Tabela 5.

		Problema			
		2	3	4	5
fM	k	5	6	7	20
	$\ g_k\ $	1.8e-4	9.7e-4	3.2e-4	3.5e-5
	\dot{it}/k	10.4	11.83	14.57	12.2
RC	k	2	3	6	50
	$\ g_k\ $	6.8e-6	8.5e-6	1.8e-4	0.003843
	\dot{it}/k	38.5	35.33	43.33	21.32

Tabela 5: Comparação entre o número de iterações externas para o método com região de confiança e para o método com o comando `fminunc` do `Matlab`.

Percebemos pela Tabela 5 que o método com região de confiança para os problemas 2, 3 e 4 realizou menos iterações externas e atingiu uma melhor precisão para a norma do gradiente do que a resolução pelo comando `fminunc`. Já para o problema 5, o método com região de confiança não obteve a mesma precisão para a norma do gradiente, realizando o número máximo de iterações 50. Comparando também a média de iterações internas por iterações externas vemos que a resolução pelo comando `fminunc` foi mais eficiente, apesar de ter realizado mais iterações externas. Entre vários fatores, um dos motivos para os distintos resultados pode ser a diferença dos critérios de parada utilizados pelos métodos. Com a utilização do comando `fminunc`, um dos critérios de parada do algoritmo do `Matlab` é a diferença entre os valores da função objetivo avaliada no ponto x_k e no anterior x_{k-1} . A precisão exigida também para o critério de parada das iterações internas não é a mesma utilizada por nós, o que pode ser refletido na perda de precisão na norma do gradiente, como vemos na Tabela 5.

5 Considerações finais

Para a família de problemas do cálculo variacional testados, o preconditionador de Roma não produziu um desempenho mais favorável para o algoritmo aplicado aos três primeiros problemas. Para o quarto problema, no entanto, houve uma melhora significativa dos resultados com o uso deste preconditionador. Este comportamento ilustra a dificuldade de se propor um esquema geral de preconditionamento que acomode diferentes problemas, mesmo dentro de uma mesma classe.

Com relação ao raio inicial de Sartenaer, os resultados obtidos para os problemas considerados não superaram a escolha $\Delta_0 = 0.1\|g_0\|$. Desta forma, o custo-benefício da sofisticada estratégia não se mostrou interessante para os problemas do cálculo variacional.

Referências

- [1] N. I. Akhiezer, “The Calculus of Variations”, Harwood Academic Publishers, London, 1988.
- [2] A. R. Conn, N. I. M. Gould & Ph. L. Toint, *LANCELOT: a Fortran Package for Large-scale Nonlinear Optimization (Release A)*, Berlin, Heidelberg, New York: Springer-Verlag, 1992.
- [3] M. C. Cunha, “Métodos numéricos”, 2 ed., Editora da Unicamp, Campinas, 2003.
- [4] P. E. Gill & W. Murray, The Numerical Solution of a Problem in the Calculus of Variations, “Recent Mathematical Developments in Control” (D. J. Bell ed.) pp. 97–122, London: Academic Press, 1973.
- [5] M. L. Krasnov, G. I. Makarenko & A. I. Kiseliiov, “Cálculo Variacional”, Editora Mir, 1984.
- [6] M. Roma, Dynamic scaling based preconditioning for truncated Newton methods in large scale unconstrained optimization: the complete results. Istituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche, Italy. Report 579, December (2002), 33 p.
- [7] A. Sartenaer, Automatic Determination of an Initial Trust Region in Nonlinear Programming, *SIAM Journal on Scientific Computing*, 18 (1997) 1788-1803.
- [8] T. Steihaug, The conjugate gradient method and trust regions in large scale optimization, *SIAM Journal on Numerical Analysis*, 20 (1983) 626-637.
- [9] Ph. L. Toint, Towards an efficient sparsity exploiting Newton methods for minimization, “Sparse Matrices and Their Uses” (I. S. Duff ed.), (1981) 57-88, London: Academic Press.