LIII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2021

Métodos de Pontos Interiores para Problemas com Desigualdades

Aurelio Ribeiro Leite de Oliveira
UNICAMP
Rua Sérgio Buarque de Holanda, 651, Campinas, SP, Brasil
aurelio@ime.unicamp.br

Yara Verdum Santiago
UNICAMP
Rua Sérgio Buarque de Holanda, 651, Campinas, SP, Brasil
yaraverdum@hotmail.com

RESUMO

Um Problema de Programação Linear costuma ser resolvido em sua forma padrão. No entanto, na grande maioria das vezes, não se apresenta inicialmente nesta forma. Para deixá-lo nesta configuração, algumas transformações são realizadas, como a inserção de variáveis de folga e excesso, por exemplo. Porém, estas ações podem ocasionar o aumento significativo da dimensão do problema em questão. Neste trabalho, desenvolveu-se um código apto a resolver Problemas de Programação Linear que não estão na forma padrão sem transformá-los para esta forma e, consequentemente, sem ocorrer o aumento de suas dimensões. Testes numéricos mostraram que a implementação desenvolvida resolve um número maior de problemas se comparado a uma implementação que soluciona os problemas exclusivamente em sua forma padrão, indicando deste modo ser um código mais robusto. Além disso, o código criado também demonstrou ser mais estável.

PALAVRAS CHAVE. Programação Linear, Métodos de Pontos Interiores, Restrições de Desigualdade.

Programação Matemática, Outras Aplicações em PO

ABSTRACT

A Linear Programming Problem is usually solved in its standard form. However, in the vast majority of cases, it does not initially appear in this form. To leave it in this configuration, some transformations are carried out, such as the insertion of slack and surplus variables, for example. Nevertheless, these actions can cause a significant increase in the dimension of the problem. In this work, a code was developed to solve Linear Programming Problems that are not in the standard form without transforming them into this form and, consequently, without increasing its dimensions. Numerical tests have shown that the developed implementation solves a greater number of problems compared to an implementation that solves the problems exclusively in its standard form, thus indicating that it is a more robust code. In addition, the code created also proved to be more efficient.

KEYWORDS. Linear Programming. Interior Point Methods. Inequality Constraints.

Mathematical Programming, Other Applications in OR

1. Introdução

Um Problema de Programação Linear é normalmente estudado em sua forma padrão, onde o conjunto de restrições é formado por igualdades e as variáveis são todas não negativas. No entanto, na grande maioria das vezes, não se apresenta inicialmente nesta forma, pois frequentemente contém restrições que são desigualdades. Assim, para deixá-lo na forma padrão, algumas transformações são realizadas, como a inserção de variáveis de folga e/ou excesso, fazendo com que as desigualdades tornem-se igualdades de modo que o problema fique, então, na forma padrão. Porém, estas ações podem acarretar o aumento significativo da dimensão do problema em questão, pois há o acréscimo no número de variáveis do problema e, consequentemente, um custo computacional de resolução possivelmente elevado, principalmente se o problema for de grande porte.

O trabalho aqui apresentado expõe o desenvolvimento de um código apto a resolver Problemas de Programação Linear cujas restrições são compostas por igualdades e/ou desigualdades e que possuem variáveis canalizadas, sem a necessidade de transformá-los para a forma padrão para iniciar sua resolução e, portanto, sem que ocorra o aumento de suas dimensões.

Para tanto, foi feito o uso do *Método Preditor-Corretor*, um dos *Métodos de Pontos Interiores*, onde algumas transformações realizadas proporcionaram condições favoráveis para que o objetivo do trabalho fosse atingido. De maneira sucinta, pode-se dizer que o código foi criado baseando-se na aplicação do *Método Primal Dual* às condições de otimalidade provenientes do problema primal e da sua versão dual após a inserção de variáveis de folga e excesso, isto é, após ambos os problemas (primal e dual) serem transformados para a forma padrão, como mostrado na Seção 3.

O código foi formulado no software *MATLAB* [MATLAB, 7.10 R2010] e seu desenvolvimento e concepção é algo inédito.

Ademais, optou-se em trabalhar com restrições em forma de desigualdade do tipo maior ou igual, porém, sem perda de generalidade, a escolha poderia ter sido trabalhar com restrições em forma de desigualdade do tipo menor ou igual, de modo a obter resultados análogos [Oliveira, 1997].

Por conseguinte, testes numéricos foram realizados a fim de por em prova a funcionalidade do código criado, bem como a comparação entre Problemas de Programação Linear resolvidos sem e com o aumento de suas dimensões através da transformação inicial para a forma padrão.

Assim sendo, a Seção 2 apresenta uma síntese do *Método Primal Dual*, empregado no desenvolvimento deste trabalho. A Seção 3 explana o desenvolvimento teórico do problema objeto de estudo, resolver um Problema de Programação Linear com restrições de igualdades e/ou desigualdades e com variáveis canalizadas sem transformá-lo para a forma padrão, que serviu de esteio para a formulação do código criado. Posteriormente, a Seção 4 traz os dados obtidos com os testes numéricos, bem como suas análises. Por fim, as conclusões são trazidas à tona.

2. Método Primal Dual

Considere o Problema de Programação Linear na forma padrão como um problema primal e, a partir dele, define-se o seu dual na forma padrão, como exibido a seguir [Luenberger e Ye, 2008]:

(Primal)

$$\begin{array}{lll} \text{Minimizar} & c^T x & \\ \text{Sujeito a} & Ax = b\ , & \\ & x \geq 0 & \end{array} \tag{1}$$

(Dual com Restrições de Igualdade)

$$\begin{array}{ll} \text{Maximizar} & b^T y \\ \text{Sujeito a} & A^T y + z = c\ , \\ & y \text{ livre},\ z \geq 0 \end{array} \tag{2}$$

onde x é o vetor das variáveis do problema primal, z é o vetor das variáveis de folga do problema dual e y é o vetor das variáveis livres do problema dual, também chamado *vetor multiplicador*.

O funcionamento do *Método Primal Dual* [Monteiro et al., 1990] consiste, basicamente, em aplicar o Método de Newton [Ruggiero e da Rocha Lopes, 1996] às condições de otimalidade do Problema de Programação Linear que está sendo resolvido, para obter as direções que possibilitam o encontro das soluções primais-duais e manter os pontos interiores.

Assim, considerando as condições de otimalidade dos problemas (1) e (2), tem-se o sistema de equações não lineares (3) a seguir:

$$\left\{ \begin{array}{lcll} Ax & = & b & x \geq 0 \\ A^T y + z & = & c & z \geq 0 \\ XZe & = & 0 & \end{array} \right. . \tag{3}$$

Fazendo uso do Método de Newton para resolver o sistema (3) precedente [Vanderbei, 1996; Wright, 1996], tem-se:

$$F(x,y,z) = \begin{pmatrix} f_p \\ f_d \\ f_a \end{pmatrix} = \begin{pmatrix} Ax - b \\ A^T y + z - c \\ XZe \end{pmatrix} = - \begin{pmatrix} r_p \\ r_d \\ r_a \end{pmatrix} = 0,$$

onde r_p é o resíduo primal dado por $r_p = b - Ax$, r_d é o resíduo dual dado por $r_d = c - A^T y - z$ e r_a é o resíduo afim dado por $r_a = -XZe$.

Assim, a direção de Newton será:

$$d = \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = \begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{pmatrix}^{-1} \begin{pmatrix} r_p \\ r_d \\ r_a \end{pmatrix}.$$

Portanto, obtém-se o sistema

$$\left\{ \begin{array}{lcl} Adx & = & r_p \\ A^T dy + dz & = & r_d \\ Zdx + Xdz & = & r_a \end{array} \right. ,$$

que ao ser resolvido, as direções dx, dy e dz são encontradas. Desta maneira, tem-se:

$$dz = X^{-1}(r_a - Zdx).$$

Definindo $D = Z^{-1}X$, chega-se nas equações:

$$dx = D(A^T dy - r_d + X^{-1} r_a),$$

$$dy = (ADA^T)^{-1}(r_p + AD(r_d - X^{-1}r_a)).$$

Como

$$S = (ADA^T) \tag{4}$$

é uma matriz definida positiva, este sistema pode ser resolvido fazendo uso da fatoração de Cholesky [Golub e Loan, 1996].

3. Métodos de Pontos Interiores para Problemas com Restrições de Igualdades e/ou Desigualdades e com Variáveis Canalizadas

Considere o seguinte Problema de Programação Linear primal, onde parte das restrições são igualdades e parte são desigualdades do tipo maior ou igual e algumas variáveis são canalizadas. Neste caso, as variáveis que são canalizadas são tratadas como restrições.

(Primal)

$$\begin{array}{lrcl} \text{Minimizar} & c^T x & & \\ \text{Sujeito a} & A_1 x & = & b_1 \\ & A_2 x & \geq & b_2 \\ & Fx & \leq & u \\ & x \geq 0 & & \end{array},$$

onde A_1 é a matriz dos coeficientes e b_1 é o vetor dos termos independentes relacionados às restrições de igualdade, A_2 é a matriz dos coeficientes e b_2 é o vetor dos termos independentes relacionados às restrições de desigualdade, F é uma matriz de dimensão $q \times n$, em que q é a quantidade de elementos do vetor x que são canalizados, n é a dimensão do vetor x e F é formada por linhas da matriz Identidade de ordem n. Além disso, u é o limite superior para os elementos de x que são canalizados.

Sua forma padrão é dada por:

(Primal - Forma Padrão)

$$\begin{array}{lrcrcrcl} \text{Minimizar} & c^T x & & & & & & \\ \text{Sujeito a} & A_1 x & & & & & = & b_1 \\ & A_2 x & - & w & & & = & b_2 \\ & Fx & + & v & & & = & u \\ & x \geq 0, & & w \geq 0, & & v \geq 0 & & \end{array}, \tag{5}$$

onde w é a variável de excesso e v é a variável de folga.

Apesar do problema com restrições de igualdade e desigualdade e com variáveis canalizadas ser transformado para a forma padrão (como indicado em (5)) e, de certa forma, assemelhar-se com o problema (1), neste as variáveis de folga e excesso não são tratadas como diferentes das demais variáveis, ao contrário do que ocorre no problema (5).

O Problema de Programação Linear dual associado ao Problema de Programação Linear primal na forma padrão é escrito como:

(Dual)

$$\begin{array}{lccccccc} \text{Maximizar} & b_1^T y_1 & + & b_2^T y_2 & + & u^T \tilde{y_3} & & \\ \text{Sujeito a} & A_1^T y_1 & + & A_2^T y_2 & + & F^T \tilde{y_3} & \leq & c \\ & & & -y_2 & & & \leq & 0 \\ & & & & & \tilde{y_3} & \leq & 0 \end{array} .$$

A partir disto, considerando $y_3 = -\tilde{y_3}$, sua forma padrão é:

(Dual - Forma Padrão)

$$\begin{array}{lccccccccc} \text{Maximizar} & b_1^T y_1 & + & b_2^T y_2 & - & u^T y_3 & & & & \\ \text{Sujeito a} & A_1^T y_1 & + & A_2^T y_2 & - & F^T y_3 & + & z & = & c \ , \\ & y_1 \text{ livre}, \ y_2 \geq 0, \ y_3 \geq 0, \ z \geq 0 & & & & & & & & \end{array} \tag{6}$$

onde z é a variável de folga.

O Problema de Programação Linear primal na forma padrão (5) pode ser escrito como apresentado abaixo:

(Primal - Forma Padrão)

$$\begin{array}{lccccc} \text{Minimizar} & c^T x & & & & \\ \text{Sujeito a} & Ax & - & G^T w & = & b \\ & Fx & + & v & = & u \\ & x \geq 0, \ w \geq 0, \ v \geq 0 & & & & \end{array} \ , \tag{7}$$

onde $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$, $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, G é uma matriz de dimensão $p \times m$, em que p é a quantidade restrições de desigualdades que o problema possui, m é o número de linhas da matriz A e G é formada por linhas da matriz Identidade de ordem m.

De modo análogo, o Problema de Programação Linear dual na forma padrão (6) pode ser escrito como apresentado a seguir:

(Dual - Forma Padrão)

$$\begin{array}{lccccccc} \text{Maximizar} & b^T y & - & u^T y_3 & & & & \\ \text{Sujeito a} & A^T y & - & F^T y_3 & + & z & = & c \ , \\ & y \geq 0, \ y_3 \geq 0, \ z \geq 0 & & & & & & \end{array} \tag{8}$$

onde $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ e z é a variável de folga.

Deste modo, os problemas primal e dual na forma padrão trabalhados são os apresentados em (7) e (8), respectivamente.

As condições de complementaridade para os problemas primal (7) e dual (8) analisados são $XZe = 0$, $WYe = 0$ e $VY_3e = 0$, onde X, Z, W, V e Y_3 são matrizes diagonais cujas diagonais principais são formadas pelo vetores x, z, w, v e y_3, respectivamente e, Y é a matriz diagonal cuja diagonal principal é formada pelo vetor Gy.

Assim, as condições de otimalidade para os problemas primal (7) e dual (8) apresentados são dadas por:

$$\begin{cases} Ax - G^T w - b & = & 0 & (x, w) \geq 0 \\ Fx + v - u & = & 0 & (x, v) \geq 0 \\ A^T y - F^T y_3 + z - c & = & 0 & (y, y_3, z) \geq 0 \\ XZe & = & 0 & \\ WYe & = & 0 & \\ VY_3e & = & 0 & \end{cases}.$$

Aplicando o Método Primal Dual para este problema, conforme descrito na Seção 2, o seguinte sistema linear é encontrado:

$$\begin{cases} Adx - G^T dw & = & r_p \\ Fdx + dv & = & r_u \\ A^T dy - F^T dy_3 + dz & = & r_d \\ Zdx + Xdz & = & r_c \\ Wdy + Ydw & = & r_i \\ Vdy_3 + Y_3 dv & = & r_b \end{cases},$$

onde $r_c = \mu e - XZe$, $r_i = \mu e - WYe$ e $r_b = \mu e - VY_3e$. Resolvendo este sistema, obtêm-se as direções dx, dy, dy_3, dz, dw e dv. Desta maneira, tem-se:

$$\begin{aligned} dv &= r_u - Fdx \\ dz &= X^{-1}(r_c - Zdx) \\ dw &= Y^{-1}(r_i - Wdy) \\ dy_3 &= V^{-1}(r_b - Y_3 dv). \end{aligned}$$

Definindo $D = X^{-1}Z + F^T V^{-1} Y_3 F$ e $E = G^T Y^{-1} W$, chega-se nas equações:

$$dx = D^{-1}(A^T dy - r_d - F^T V^{-1} r_b + X^{-1} r_c + F^T V^{-1} Y_3 r_u),$$

$$dy = (AD^{-1}A^T + E)^{-1}(AD^{-1}(r_d + F^T V^{-1} r_b - X^{-1} r_c - F^T V^{-1} Y_3 r_u) + r_p + G^T Y^{-1} r_i).$$

Como

$$\bar{S} = (AD^{-1}A^T + E) \tag{9}$$

é uma matriz definida positiva, este sistema pode ser resolvido fazendo uso da fatoração de Cholesky.

4. Testes Numéricos

Após a criação do código que resolve Problemas de Programação Linear com restrições de igualdades e/ou desigualdades e com variáveis canalizadas sem transformá-los para a forma padrão, testes numéricos foram realizados para testar sua funcionalidade. Para tais testes, foram utilizados problemas da biblioteca *Netlib* [Netlib, 2020]. Constatou-se que os problemas resolvidos fazendo uso do referido código tiveram os mesmos valores de solução ótima de quando resolvidos na forma padrão.

Além disso, foi efetuada uma comparação entre as resoluções de Problemas de Programação Linear usando o código criado, que foi denominado (*Código Geral*), e suas resoluções através de um código que resolve estes problemas apenas se forem transformados inicialmente para a forma padrão, denominado *Código Padrão*. Portanto, os problemas resolvidos pelo *Código Padrão* têm dimensão superior se comparado à sua resolução através do *Código Geral*.

Dos problemas da biblioteca *Netlib*, os que apresentavam restrições de igualdades e/ou desigualdades, além de variáveis não negativas e possivelmente canalizadas, alcançavam um total de 59 problemas a serem testados. O *Código Geral* solucionou corretamente 55 destes 59 problemas. Os quatro problemas que não foram resolvidos são: *scorpion*, *brandy*, *25fv47* e *wood1p*.

Entre os 59 problemas da biblioteca *Netlib* testados no *Código Geral*, 15 não foram resolvidos pelo *Código Padrão*. São eles: *kb2*, *scorpion*, *brandy*, *grow7*, *degen2*, *bnl1*, *grow15*, *grow22*, *25fv47*, *ship08l*, *fit1d*, *degen3*, *fit2p*, *wood1p* e *fit2d*.

Na Tabela 1, há a relação dos 55 Problemas de Programação Linear que foram resolvidos fazendo uso do *Código Geral* e, posteriormente, usando o *Código Padrão*. Além disso, apresenta-se a dimensão dos problemas em ambas as abordagens. A barra sobre A indica os dados pertinentes ao *Código Geral* e onde não a houver é relativo ao *Código Padrão*. Na última coluna, apresenta-se o aumento percentual do número de variáveis dos problemas resolvidos com o *Código Padrão* em relação ao número de variáveis dos problemas resolvidos com o *Código Geral*.

Problema		**Dimensão de $\bar{A}$**	**Dimensão de A**	**Aumento do Nº de Variáveis (%)**
1	afiro	27×32	27×51	$59,38\%$
2	sc50b	50×48	50×78	$62,50\%$
3	sc50a	50×48	50×78	$62,50\%$
4	sc105	105×103	105×163	$58,25\%$
5	kb2	43×41	43×68	$65,85\%$
6	adlittle	56×97	56×138	$42,27\%$
7	scagr7	129×140	129×185	$32,14\%$
8	stocfor1	117×111	117×165	$48,65\%$
9	blend	74×83	74×114	$37,35\%$
10	sc205	205×203	205×317	$56,16\%$
12	share2b	96×79	96×162	$105,06\%$
14	lotfi	153×308	153×366	$18,83\%$
15	share1b	117×225	117×253	$12,44\%$
19	scagr25	471×500	471×671	$34,20\%$
20	sctap1	300×480	300×660	$37,50\%$
23	israel	174×142	174×316	$122,54\%$
25	gfrdpnc	616×1092	616×1160	$6,23\%$
26	scsd1	77×760	77×760	0%
28	agg	488×163	488×615	$277,30\%$
29	bandm	305×472	305×472	0%
30	e226	223×282	223×472	$67,38\%$
31	scfxm1	330×457	330×600	$31,29\%$
32	grow7	140×301	140×301	0%
34	scrs8	490×1169	490×1275	$9,08\%$
35	beaconfd	173×262	173×295	$12,60\%$
40	degen2	444×534	444×757	$41,76\%$
41	agg2	516×302	516×758	$150,99\%$
42	agg3	516×302	516×758	$150,99\%$
43	scsd6	147×1350	147×1350	0%
44	ship04s	402×1458	402×1506	$3,29\%$

48	bnl1	643×1175	643×1586	$34,98\%$
50	scfxm2	660×914	660×1200	$31,29\%$
51	grow15	300×645	300×645	0%
53	fffff800	524×854	524×1028	$20,37\%$
54	ship04l	402×2118	402×2166	$2,27\%$
55	sctap2	1090×1880	1090×2500	$32,98\%$
57	ship08s	778×2387	778×2467	$3,35\%$
59	scfxm3	990×1371	990×1800	$31,29\%$
60	ship12s	1151×2763	1151×2869	$3,84\%$
61	grow22	440×946	440×946	0%
62	stocfor2	2157×2031	2157×3045	$49,93\%$
63	scsd8	397×2750	397×2750	0%
64	sctap3	1480×2480	1480×3340	$34,68\%$
67	fit1p	627×1677	627×1677	0%
70	ship08l	778×4283	778×4363	$1,87\%$
73	fit1d	24×1026	24×1049	$2,24\%$
74	bnl2	2324×3489	2324×4486	$28,58\%$
76	ship12l	1151×5427	1151×5533	$1,95\%$
79	degen3	1503×1818	1503×2604	$43,23\%$
80	truss	1000×8806	1000×8806	0%
83	d2q06c	2171×5167	2171×5831	$12,85\%$
84	woodw	1098×8405	1098×8418	$0,15\%$
86	fit2p	3000×13525	3000×13525	0%
87	stocfor3	16675×15695	16675×23541	$49,99\%$
90	fit2d	25×10500	25×10524	$0,23\%$

Tabela 1: Dimensão dos Problemas de Programação Linear analisados e seu aumento percentual.

Ao resolver os 55 Problemas de Programação Linear mencionados fazendo uso do *Código Geral* e do *Código Padrão*, percebeu-se que na grande maioria dos problemas, o número de iterações em que estes atingiram o valor ótimo da função objetivo foi o mesmo em ambas as abordagens.

Ademais, a fatoração de Cholesky da matriz S dada por (4), presente no *Código Padrão*, e da matriz $\bar{S}$ dada por (9), presente no *Código Geral*, permite que os sistemas lineares existentes nos respectivos códigos possam ser resolvidos.

Na Tabela 2, há os números de condição das matrizes $\bar{S}$ e S relativas aos 55 Problemas de Programação Linear resolvidos através do *Código Geral* e do *Código Padrão*, respectivamente. Os dados coletados foram o da última iteração em comum para ambas as abordagens. Pelos números apresentados, percebe-se que em 31 dos 55 problemas analisados, o número de condição de $\bar{S}$ foi menor que o número de condição de S.

Problema		**Número de Condição de $\bar{S}$**	**Número de Condição de S**
1	afiro	$5,88925651867 \times 10^{12}$	$5,88925651860 \times 10^{12}$
2	sc50b	$1,30806258725 \times 10^{30}$	$1,30806258733 \times 10^{30}$
3	sc50a	$1,90399731996 \times 10^{16}$	$1,90399731953 \times 10^{16}$
4	sc105	$7,07615606857 \times 10^{16}$	$7,07615606730 \times 10^{16}$
5	kb2	$8,30195363562 \times 10^{11}$	$1,76239889122 \times 10^{18}$
6	adlittle	$6,96821232418 \times 10^{19}$	$6,96821232745 \times 10^{19}$

7	scagr7	$1,34798786462 \times 10^{10}$	$1,34798786241 \times 10^{10}$
8	stocfor1	$3,70706614502 \times 10^{16}$	$3,70705592772 \times 10^{16}$
9	blend	$4,57449338817 \times 10^{19}$	$4,57449339525 \times 10^{19}$
10	sc205	$1,14415966973 \times 10^{21}$	$1,14415966974 \times 10^{21}$
12	share2b	$1,02795577894 \times 10^{17}$	$1,02795575138 \times 10^{17}$
14	lotfi	$9,42423312340 \times 10^{25}$	$9,42435979592 \times 10^{25}$
15	share1b	$6,44636526497 \times 10^{14}$	$6,44639416333 \times 10^{14}$
19	scagr25	$4,91606240854 \times 10^{15}$	$4,91606230377 \times 10^{15}$
20	sctap1	$6,07869599138 \times 10^{16}$	$6,07929373460 \times 10^{16}$
23	israel	$3,75024051018 \times 10^{15}$	$3,75024112287 \times 10^{15}$
25	gfrdpnc	$1,80872233418 \times 10^{31}$	$5,49885209026 \times 10^{24}$
26	scsd1	$1,50950010276 \times 10^{11}$	$1,50950009778 \times 10^{11}$
28	agg	$3,21820520537 \times 10^{26}$	$3,21831375188 \times 10^{26}$
29	bandm	$1,12765634547 \times 10^{24}$	$1,12765635284 \times 10^{24}$
30	e226	$7,75890330757 \times 10^{25}$	$7,75890343666 \times 10^{25}$
31	scfxm1	$6,14100606166 \times 10^{21}$	$6,14100602956 \times 10^{21}$
32	grow7	$4,72430891769 \times 10^{13}$	$1,25890874948 \times 10^{30}$
34	scrs8	$2,55462285344 \times 10^{32}$	$2,55462299288 \times 10^{32}$
35	beaconfd	$2,03928598220 \times 10^{20}$	$2,03928598208 \times 10^{20}$
40	degen2	$4,04131163347 \times 10^{20}$	$1,18945657477 \times 10^{44}$
41	agg2	$1,50269649256 \times 10^{24}$	$1,50269649968 \times 10^{24}$
42	agg3	$5,94587063367 \times 10^{26}$	$5,94587073060 \times 10^{26}$
43	scsd6	$4,91723584722 \times 10^{12}$	$4,91722410656 \times 10^{12}$
44	ship04s	Inf	Inf
48	bnl1	Inf	Inf
50	scfxm2	$1,20832798431 \times 10^{19}$	$1,20833989846 \times 10^{19}$
51	grow15	$6,91447795750 \times 10^{14}$	$4,60707774484 \times 10^{23}$
53	fffff800	$1,00175346324 \times 10^{38}$	$1,00178360795 \times 10^{38}$
54	ship04l	Inf	Inf
55	sctap2	$7,11398025367 \times 10^{14}$	$7,11404626997 \times 10^{14}$
57	ship08s	Inf	Inf
59	scfxm3	$1,21241645929 \times 10^{23}$	$1,21241683416 \times 10^{23}$
60	ship12s	Inf	Inf
61	grow22	$1,52337026798 \times 10^{12}$	$5,06648435231 \times 10^{18}$
62	stocfor2	$4,33253976397 \times 10^{22}$	$4,19503773591 \times 10^{22}$
63	scsd8	$2,85294199265 \times 10^{16}$	$2,85251200254 \times 10^{16}$
64	sctap3	$3,18515404755 \times 10^{28}$	$3,94252361705 \times 10^{28}$
67	fit1p	$2,31456081823 \times 10^{09}$	$8,60608014330 \times 10^{10}$
70	ship08l	Inf	Inf
73	fit1d	$2,54687301985 \times 10^{12}$	$1,48258070119 \times 10^{11}$
74	bnl2	$3,15499861837 \times 10^{42}$	$3,15499867725 \times 10^{42}$
76	ship12l	Inf	Inf
79	degen3	$1,10309407123 \times 10^{19}$	$9,28346040186 \times 10^{23}$
80	truss	$4,19280806506 \times 10^{16}$	$4,19280825128 \times 10^{16}$
83	d2q06c	$2,29213078257 \times 10^{37}$	$2,29226302291 \times 10^{37}$
84	woodw	$8,39440019168 \times 10^{26}$	$1,93437759871 \times 10^{17}$
86	fit2p	$2,45218991317 \times 10^{7}$	$3,86258014602 \times 10^{6}$
87	stocfor3	$1,26125078053 \times 10^{21}$	$1,26179133019 \times 10^{21}$
90	fit2d	$4,12318114326 \times 10^{8}$	$1,51875057167 \times 10^{10}$

Tabela 2: Número de condição de $\bar{S}$ e de S dos Problemas de Programação Linear analisados.

5. Conclusões

Os testes numéricos realizados mostraram que o código criado atingiu o objetivo de solucionar Problemas de Programação Linear com restrições de igualdades e/ou desigualdades e com variáveis canalizadas sem transformá-los para a forma padrão e, consequentemente, sem o aumento de suas dimensões. Ao realizar a comparação com a resolução destes problemas transformados inicialmente para a forma padrão, através do uso do código denominado *Código Padrão*, percebeu-se que o código criado, intitulado *Código Geral*, solucionou maior quantidade de problemas, sendo então, um código mais robusto.

Através da Tabela 1 percebeu-se que, em geral, a dimensão dos Problemas de Programação Linear resolvidos através do *Código Padrão* é superior à dimensão destes mesmos problemas quando solucionados fazendo uso do *Código Geral*. O caso mais notável foi do problema *agg*, cuja dimensão quando resolvido pelo *Código Geral* era 488×163 e quando resolvido pelo *Código Padrão* foi 488×615, caracterizando um acréscimo de $277,30\%$ no número de variáveis do problema resolvido por uma abordagem em relação à outra, representando um aumento bastante significativo.

Além disso, de acordo com os dados relativos aos números de condição, percebeu-se que em 31 dos 55 problemas analisados, o número de condição da matriz $\bar{S}$, relativa ao *Código Geral*, foi menor que o número de condição da matriz S, relativa ao *Código Padrão*. Portanto, o código criado é mais estável se comparado ao código que resolve Problemas de Programação Linear transformando-os para a forma padrão. Isto se deve à inserção da matriz diagonal E^k na matriz S dada por (4), presente no *Código Padrão*, resultando na matriz $\bar{S}$ dada por (9), presente no *Código Geral*.

Desta maneira, constatou-se ser mais vantajoso resolver Problemas de Programação Linear com restrições de igualdades e/ou desigualdades e com variáveis canalizadas fazendo uso do código criado (*Código Geral*) ao invés do código que resolve os referidos problemas transformando-os inicialmente para a forma padrão (*Código Padrão*), pois revolve estes problemas sem que ocorra o aumento de suas dimensões, é um código mais robusto, visto que soluciona um número maior de problemas, além de ser um código mais estável.

Agradecimentos

O presente trabalho foi realizado com apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Referências

Golub, G. H. e Loan, C. F. V. (1996). *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, 3 edition.

Luenberger, D. G. e Ye, Y. (2008). *Linear and Nonlinear Programming*. Springer, Nova Iorque, 3 edition.

MATLAB (7.10 R2010).

Monteiro, R. D. C., Adler, I., e Resende, M. G. C. (1990). A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension. *Mathematics of Operations Research*, 15:191?214.

Netlib (2020). Netlib lp repository: Netlib collection lp test sets. URL `https://www.netlib.org/lp/data/`. Agosto, 2020.

Oliveira, A. R. L. (1997). A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. Technical report, PhD Thesis, TR97-11, Department of Computational and Applied Mathematics, Rice University, Houston TX.

Ruggiero, M. A. G. e da Rocha Lopes, V. L. (1996). *Cálculo Numérico: Aspectos Teóricos e Computacionais*. Pearson, São Paulo, 2 edition.

Vanderbei, R. J. (1996). *Linear Programming – Foundations and Extensions*. Kluwer Academics Publishers, Boston, USA.

Wright, S. J. (1996). *Primal–Dual Interior–Point Methods*. SIAM Publications, SIAM, Philadelphia, PA, USA.