Bibliografia

[1] MORAWETZ, C. S.. **The eigenvalues of some stability problems involving viscosity**. J. Rational Mech. Analysis., 1:579–603, 1952.

[2] ROUSE, P. E.. **A theory of the linear viscoelastic properties of dilute solutions of coiling polymers.** J. Chem. Phys., 21:1272–1280, 1953.

[3] GORODTSOV, V. A.; LEONOV, A. I.. **On a linear instability of plane couette flow of viscoelastic fluid.** Journal Appl. Math. Mech., 31:310–319, 1967.

[4] ROMANOV, V. A.. **Stability of plane-parallel couette flow**. Funct. Anal. Applics., 7:137–146, 1973.

[5] DRAZIN, P. G.; H., R. W.. **Hidrodynamic stability.** Cambridge University Press, 1981.

[6] GIESEKUS, H.. **Simple constitutive equation for polymer fluids based on the consept of deformation dependent tensorial mobility.** Journal of Non-Newtonian Fluid Mechanics, 11:69–109, 1982.

[7] SUN, J.; SMITH, M. D.; ARMSTRONG, R. C. ; BROWN, R. A.. **Finite element method for viscoelastic flows based on the discrete adaptative viscoelastic stress splitting and the discontinuous galerkin method.** Journal of Non-Newtonian Fluid Mechanics, 11:69–109, 1982.

[8] RUSCHAK, K. J.. **A three-dimentional linear stability analysis for two-dimentional free boudary flows by the finite element method**. Computers and Fluids., 11(04):391–401, 1983.

[9] RENARDY, M.; RENARDY, Y.. **Linear stability of plane couette flow of an upper convected maxwell fluid**. Journal of non-Newtonian Fluid Mechanics., 22:23–33, 1986.

[10] BIRD, R. B.; ARMSTRONG, R. C. ; O., H.. **Dynamics of Polymeric Liquids.** Wiley-Intercience Publication, 1987.

[11] CHRISTODOULOU, K. N.; SCRIVEN, L. E.. **Finding leading modes of a viscous free surface flow: an asymmetric generalized eigenproblem.** Journal of Scientific Computation, 3:355–406, 1988.

[12] COYLE, D. J.; MACOSKO, C. W. ; SCRIVEN, L. E.. **Stability of symmetric film-splitting between counter-rotating cylinders**. Journal of Fluid Mechanics., 216:437–458, 1990.

[13] TILLMARK, N.; ALFREDSSON, P. H.. **Experiments on trasition and turbulence in plane couette flow**. Journal of Fluid Mechanics., 235:89–102, 1992.

[14] RAMANAN, N.; HOMSY, G. M.. **Linear stability of lid-driven cavity flow**. Physics and Fluids., 06(08):2690–2701, 1994.

[15] SURESHKUMAR, R.; BERIS, A. N.. **Linear stability analysis of viscoelastic poiseuille flow using an arnoldi-based orthogonalization algorithm.** Journal of Non-Newtonian Fluid Mechanics, 56:151–182, 1995.

[16] SZADY, M. J.; SALOMON, T. R.; LIU, A. W.; BORNSIDE, D. E.; ARMSTRONG, R. C. ; BROWN, R. A.. **A new mixed finite element method for viscoelastic flows governed by diferential contitutive equations.** Journal of non-Newtonian Fluid Mechanics., 59:215–243, 1995.

[17] DONGARRA, J. J.; STRAUGHAM, B. ; WALKER, D. W.. **Chebyshev tau-qz methods for calculating spectra of hydrodynamic stability problems**. App. Numer. Math., 22:399–434, 1996.

[18] SHAQFEH, S. G.. **Purely elastic instabilities in viscometric flows.** Annu. Rev.J. Fluid. Mech., 28:129–185, 1996.

[19] SAAD, Y.. **Iterative methods for saprse linear systems**. PWS publishing N.Y., 1996.

[20] GOLUB, G. H.; F., V. L. C.. **Matrix Computations.** The Johns Hopkins University Press, 1996.

[21] PANTON, R. L.. **Incompressible flow.** Wiley-Intercience Publication., second edition, 1996.

[22] BAAIJENS, F. P. T.. **Mix finite element methods for viscoelastic flow analysis: a review**. Journal of non-Newtonian Fluid Mechanics., 79:361–385, 1998.

[23] SURESHKUMAR, R.; SMITH, M. D.; ARMSTRONG, R. C. ; BROWN, R. A.. **Linear stability and dynamic of viscoelastic flows using time-dependent numerical simulations.** Journal of Non-Newtonian Fluid Mechanics, 82:57–104, 1999.

[24] WILSON, H. J.; RENARDY, M. ; RENARDY, Y.. **Structure of the spectrum in zero reynolds number shear flow of the ucm and odroyd-b liquids**. Journal of non-Newtonian Fluid Mechanics., 80:251–268, 1999.

[25] SOUZA, M. O.. **Introdução à estabilidade hidrodinâmica.** SBMAC - XXII Congresso Nacional de Matemática Aplicada e Computacional, CNMAC, 1999.

[26] CARVALHO, M. S.; SCRIVEN, L. E.. **Three-dimensional stability analisis of free surface flows: Application to forward deformable roll coating.** J. Comput. Phys., 151:534–562, 1999.

[27] WATKINS, D. S.. **Infinite eigenvalues and qz algorithm**. preprint SFB 393, 99(23), 1999.

[28] SURESHKUMAR, R.; ARORA, K.. **Efficient computation of the eigenspectrum of viscoelastic flows using submatrix-based transformation of the linerized equations**. Journal of Non-Newtonian Fluid Mechanics, 104:75–85, 2002.

[29] RENARDY, M.. **Effect of upstream boundary conditions on stability of fiber spinning in the highly elastic limit**. Journal of Rheology., 46(4):1023–1028, 2002.

[30] PASQUALI, M.; SCRIVEN, L. E.. **Free surface flows of polymer solutions with mod- els based on the conformation tensor**. Journal of non-Newtonian Fluid Mechanics., 108:363–400, 2002.

[31] BOTTARO, A.; CORBETT, P. ; LUCHINI, P.. **The effect of base flow variation on flow stability.** Journal of Fluid Mechanics, 476:293–302, 2003.

[32] SURESHKUMAR, R.. **Stability analysis using compressible viscoelastic formulations**. Journal of Non-Newtonian Fluid Mechanics, 116:471–477, 2004.

[33] KUPFERMAN, R.. **On the linear stability o plane couette flow for an oldroid-b fluid and its numerical approximation.** Journal of Non-Newtonian Fluid Mechanics., 127:169–190, 2005.

A
Formulação com líquido Newtoniano

A.1
Programa do método proposto para a solução do problema de autovalor generalizado vindo da análise de estabilidade de um escoamento de Couette com líquido Newtoniano.

Vamos mostrar o programa feito em matlab sobre o problema de autovalor generalizado discutido no capítulo 4 que elimina os autovalores no infinito. O programa compara o tempo e o espectro calculado pelo método QZ das matrizes originais com o tempo de realizar as transformações e calcular o espectro nas matrizes reduzidas. As matrizes são formadas pela discretização do sistema que descreve a estabilidade do escoamento de Couette com líquido Newtoniano com o método de Galerkin/elementos finitos.

```
clear all;  % para limpar a área de trabalho do matlab.
% Os parâmetros:
NELE = 200   % número de elementos,
Re = 500;   % número de Reynolds,
α = 1.5;   % número de onda.
tic    % o comando 'tic toc' fornece o tempo gasto para a
% realização da rotina.
% monta as matrizes M e J vindas dos elementos finitos. De
forma esparsa!
[M] = formM12DNs(NELE, Re, α);
[J] = formJ12DNs(NELE, Re, α);
% número total de graus de liberdade:
n = NDoF = 2 * (NELE + 1) + 2 * NELE;
t1 = toc  % t1 eh o tempo em segundos para a formação das
% matrizes.
```

```
% nomenclatura para as dimensões envolvidas no problema:
nu = 2 * NELE + 1; % número de graus de liberdade de u
nv = 2 * NELE + 14; % número de graus de liberdade de v
np = 2 * NELE; %numero de graus de liberdade de p
ntv = 2 * (2 * NELE + 1); % número de graus de
% liberdade de u + v
ntp = 2 * (2 * NELE + 1) + 2 * NELE; % número
% de graus de liberdade de u + v + p
Jnew = J; % guardando as matrizes originais.
Mnew = M;
tic
% retirando as linhas e colunas relativas as condições
% de contorno; neste caso, elas estão em 1, 2NELE,
% 2NELE + 2 e 4NELE + 1.
%tirando as linhas:
Jnew(1,:) = [ ];
Jnew(2 * NELE - 1,:) = [ ];
Jnew(2 * NELE,:) = [ ];
Jnew(4 * NELE - 2,:) = [ ];
Mnew(1,:) = [ ];
Mnew(2 * NELE - 1,:) = [ ];
Mnew(2 * NELE,:) = [ ];
Mnew(4 * NELE - 2,:) = [ ];
%colunas:
Jnew(:,1) = [ ];
Jnew(:,2 * NELE - 1) = [ ];
Jnew(:,2 * NELE) = [ ];
Jnew(:,4 * NELE - 2) = [ ];
Mnew(:,1) = [ ];
Mnew(:,2 * NELE - 1) = [ ];
Mnew(:,2 * NELE) = [ ];
Mnew(:,4 * NELE - 2) = [ ];
% montando os blocos da matriz A para a utilização do método
% proposto.
J11 = Jnew(1 : np, 1 : np);
M11 = Mnew(1 : np, 1 : np);
J12 = Jnew(1 : np, np + 1 : 2 * nu - 4);
M12 = Mnew(1 : np, np + 1 : 2 * nu - 4);
J21 = Jnew(np + 1 : 2 * nu - 4, 1 : np);
```

$M21 = Mnew(np+1:2*nu-4, 1:np);$
$J22 = Jnew(np+1:2*nu-4, np+1:2*nu-4);$
$M22 = Mnew(np+1:2*nu-4, np+1:2*nu-4);$
% Blocos sem σ $\Rightarrow$ **M** é zero!!!
$A13 = Jnew(1:np, (2*nu)-3:ntp-4);$
$A31 = Jnew((2*nu)-3:ntp-4, 1:np);$
$A23 = Jnew(np+1:2*nu-4, (2*nu)-3:ntp-4);$
$A32 = Jnew((2*nu)-3:ntp-4, np+1:2*nu-4);$
% calculando a inversa dos blocos A_{13} e A_{31}, pelo método LU:
$[L13, U13] = lu(A13);$ $[L31, U31] = lu(A31);$
$I13 = speye(length(A13));$
$A13inv = U13\backslash(L13\backslash I13);$
$A31inv = U31\backslash(L31\backslash I13);$
% montando os blocos referentes as transformações:
$Tlb = -A23 * A13inv;$
$Trb = -A31inv * A32;$
% calculando somente a sub-matriz:
$SubJ22 = (Tlb * J11 + J21) * (Trb) + (Tlb * J12 + J22);$
$SubM22 = (Tlb * M11 + M21) * (Trb) + (Tlb * M12 + M22);$
$SubM22inv = inv(SubM22);$
$SubMJ22 = -SubM22inv * SubJ22;$
$SubMJ22 = full(SubMJ22);$
$[Vs, Ds] = eig(SubMJ22);$ % calcula o problema cássico de
% autovalor.
t_{EVP} = toc % tempo em que o problema de autovalor generalizado
% leva para tornar-se clássico e o espectro ser obtido.
$for \quad is = 1 : length(SubMJ22)$
$\quad subg22(is, 1) = Ds(is, is);$
end
% calculando o autovalor generalizado do problema nas matrizes
% originais para comparar:
$Jnew = full(J);$
$Mnew = full(M);$
tic
$[Vf, Df] = eig(Jnew, -Mnew, 'qz');$ % calcula o problema
% generalizado de autovalor como método QZ.
t_{GEVP} = toc % tempo para calcular os autovalores das matrizes
% originais:
$for \ ifull = 1 : length(Jnew)$

$gnew(ifull, 1) = Df(ifull, ifull);$
end
$rt = t_{GEVP}/t_{EVP}$ `% razão entre os tempos.`

Os autovalores estão armazenados nos vetores **subg22** e **gnew** relativos ao problema reduzido e ao original, respectivamente. Pronto para serem salvos, usados em um gáfico, etc.

A.2
Variação do programa do método proposto para a solução do problema de autovalor generalizado vindo da análise de estabilidade de um escoamento de Couette com líquido Newtoniano.

No método proposto, transformações são identificadas com o intuito de eliminar os autovalores no infinito. Nessas transformações, inversas de dois blocos da matriz original são calculados, como mostrado na seção 3.3.1. No caso do escoamento de Couette com líquido Newtoniano e usando a formulação unidimensional, a discretização pelo método de Galerkin elementos finitos forma matrizes cujos blocos a serem invertidos podem sofrer permutações que os transformam em matrizes diagonais de inversas triviais. No código anterior, depois de tirar as condições de contorno e antes de montar os blocos, as seguintes permutações serão feitas:

`% Criando a PERMUTAÇÃO:`
$pc = zeros(1, length(Jnew));$
$ic = 1;$
$for\ \ ip = 2 : 2 : (2 * NELE - 2)$
$pc(ic) = ip;$
$pc(ic + 1) = ip + 2 * NELE - 1;$
$ic = ic + 2;$
end
$pc(2 * NELE - 1) = pc(2 * NELE - 3) + 1;$
$pc(2 * NELE) = pc(2 * NELE - 2) + 1;$
$icomp = 0;$
$icp = 0;$
$while(icomp < length(Jnew))$
$if(pc(1 : 2 * NELE + icp) \ \ = icomp + 1)$

$pc(2 * NELE + icp + 1) = icomp + 1;$
$icp = icp + 1;$
end
$icomp = icomp + 1;$
end
$Jperm = Jnew(pc, pc);$
$Mperm = Mnew(pc, pc);$

A divisão é a mesma de antes, porém alguns blocos que não eram utilizados são necessários agora.

% esses blocos trazem σ por isso tem **J** e **M**.
% E são usados se as transformações não forem usadas.
$J11 = Jperm(1 : np, 1 : np);$
$M11 = Mperm(1 : np, 1 : np);$
$J12 = Jperm(1 : np, np + 1 : 2 * nu - 4);$
$M12 = Mperm(1 : np, np + 1 : 2 * nu - 4);$
$J21 = Jperm(np + 1 : 2 * nu - 4, 1 : np);$
$M21 = Mperm(np + 1 : 2 * nu - 4, 1 : np);$
$J22 = Jperm(np + 1 : 2 * nu - 4, np + 1 : 2 * nu - 4);$
$M22 = Mperm(np + 1 : 2 * nu - 4, np + 1 : 2 * nu - 4);$
% Os que não tem σ:
$Ap13 = Jperm(1 : np, (2 * nu) - 3 : ntp - 4);$
$Ap31 = Jperm((2 * nu) - 3 : ntp - 4, 1 : np);$
$Ap23 = Jperm(np + 1 : 2 * nu - 4, (2 * nu) - 3 : ntp - 4);$
$Ap32 = Jperm((2 * nu) - 3 : ntp - 4, np + 1 : 2 * nu - 4);$
% inversa trivial:
for $iin = 1 : np$
$A13pinv(iin, iin) = 1/Ap13(iin, iin);$
$A31pinv(iin, iin) = 1/Ap31(iin, iin);$
end

As operações seguintes são as mesmas, apenas usando **A13pinv** e **A31pinv** no lugar de **A13inv** e **A31inv** na montagem dos blocos referentes as transformações.

A.3
Recuperação dos autovetores originais a partir do problema reduzido vindo da formulação unidimensional com líquido Newtoniano.

Nesse algoritmo, os blocos $\mathbf{A}_{13}$ e $\mathbf{A}_{31}$ foram invertidos e não permutados para se tornarem diagonais. Ele tem que estar no mesmo programa que o código para redução da matriz, pois usa as variáveis definidas nele.

% **autovetsq** é a coluna k, referente ao autovalor σ_k, da matriz **Vsq** formada pelos autovetores calculados no problema reduzido:

$a(k) = -0.055279$

$b(k) = 0.95131$

$\sigma_k = a(k) + i * b(k)$

$for \;\; iaa = 1 : 2 * nv - np - 4$

$\quad if(norm(Dsq(iaa, iaa) - (a(k) + i * b(k))) < 10^{-5})$

$\quad\quad autovalsq = Dsq(iaa, iaa)$

$\quad\quad vetsq = iaa;$

$\quad end$

end

$autovetsq = (Vsq(:, vetsq))/norm(Vsq(:, vetsq));$

$Tr = speye(ntp - 4);$ % formando a matriz $\mathbf{T_r}$ necessária para % a recuperação do autovetor original.

$Tr(1 : np, np + 1 : 2 * nu - 4) = Trb;$

% as transformações não tinham sido feitas em $\mathbf{A}_{12}$.

$A12tilda = (J11 + autovalsq * M11) * Trb + (J12 + autovalsq * M12);$

$autovetT1 = zeros(np, 1);$

$autovetT2 = autovetsq;$

$autovetT3 = -A13inv * A12tilda * autovetT2;$

$autovetT = [autovetT1; autovetT2; autovetT3];$

$autovet = Tr * autovetT;$

$autovet = autovet/norm(autovet);$

%colocando zero nas condições de contorno do autovetor transformado.

$autovetTfim = zeros(NDoF, 1);$

$icv = 1;$

$for \;\; iv = 2 : 2 * NELE - 1$

$\quad autovetTfim(iv) = autovet(icv);$

$$
\begin{aligned}
&\quad icv = icv + 1;\\
&end\\
&autovetTfim(2 * NELE + 1) = autovet(icv);\\
&icv = icv + 1;\\
&foriv = 2 * NELE + 3 : 4 * NELE\\
&\quad autovetTfim(iv) = autovet(icv);\\
&\quad icv = icv + 1;\\
&end\\
&for\ \ iv = 4 * NELE + 2 : NDoF\\
&\quad autovetTfim(iv) = autovet(icv);\\
&\quad icv = icv + 1;\\
&end\\
&autovetTfim = autovetTfim/norm(autovetTfim)\,.
\end{aligned}
$$

O autovetor **autovetTfim** recupera o tirado da matriz de autovetores do problema original referente ao mesmo autovalor σ_k. Fizemos um pós-processamento separando os autovetores em campos de velocidade e pressão.

B
Formulação com líquido viscoelástico

B.1
Programa do método proposto para a solução do problema de autovalor generalizado vindo da análise de estabilidade de um escoamento de Couette com líquido viscoelástico.

Vamos mostrar o programa feito em matlab sobre o problema de autovalor generalizado discutido no capítulo 6 que elimina os autovalores no infinito. O programa compara o tempo e o espectro calculado pelo método QZ das matrizes originais com o tempo de realizar as transformações e calcular o espectro nas matrizes reduzidas. As matrizes são formadas pela discretização do sistema que descreve a estabilidade do escoamento de Couette com líquido descrito pelo modelo de Maxwell pela formulação $\mathbf{T}_{\ell \mathbf{d}}$ com o método de Galerkin/elementos finitos.

```
clear all;  % para limpar a área de trabalho do matlab.
% Os parâmetros:
NELE = 200   % número de elementos,
Re = 0;   % número de Reynolds,
We = 10; % número de Weissemberg
α = 1;   % número de onda.
tic   % o comando 'tic toc' fornece o tempo gasto para a
% realização da rotina.
% monta as matrizes M e J vindas dos elementos finitos. De
forma esparsa!
[M] = formMS(neta, NELE, We, Re, α);
[J] = formJS(neta, NELE, We, Re, α);
% número total de graus de liberdade:
n = NDoF = 20 * NELE + 2;
t1 = toc  % t1 eh o tempo em segundos para a formação das
```

```
% matrizes.
```
$Jve = J$; `% guardando as matrizes originais.`
$Mve = M$;
```
tic
% retirando as linhas e colunas relativas as condições
% de contorno:
```
$J(:, 8 * NELE + 1) = [\ ];$
$J(8 * NELE + 1, :) = [\ ];$
$J(:, 10 * NELE - 1) = [\ ];$
$J(10 * NELE - 1, :) = [\ ];$
$J(:, 10 * NELE) = [\ ];$
$J(10 * NELE, :) = [\ ];$
$J(:, 12 * NELE - 2) = [\ ];$
$J(12 * NELE - 2, :) = [\ ];$
$M(:, 8 * NELE + 1) = [\ ];$
$M(8 * NELE + 1, :) = [\ ];$
$M(:, 10 * NELE - 1) = [\ ];$
$M(10 * NELE - 1, :) = [\ ];$
$M(:, 10 * NELE) = [\ ];$
$M(10 * NELE, :) = [\ ];$
$M(:, 12 * NELE - 2) = [\ ];$
$M(12 * NELE - 2, :) = [\ ];$
```
% as dimensões:
```
$dimJ = length(J)$;
$dimM = length(M)$;
$dimT = 6 * NELE$;
$dimp = 2 * NELE$;
$dimu = 4 * NELE - 2$; `% sem condição de contorno.`
$dimG = 8 * NELE$;
$dimTpu = dimT + dimp + dimu.$
```
% Montando a primeira transformação que vai reduzir a dimensão
% de 20NELE − 2 (sem cond. cont.) para 12NELE − 2.
%Fazer a transformação somente na matriz jacobiana, pois a
% matriz massa permanece inalterada.
%Somente os blocos envolvidos são criados.
```
$J12 = J(1 : dimT + dimp, dimT + dimp + 1 : dimTpu)$;
$J13 = J(1 : dimT + dimp, dimTpu + 1 : dimJ)$;
$J32 = J(dimTpu + 1 : dimJ, dimT + dimp + 1 : dimTpu)$;
$J33 = J(dimTpu + 1 : dimJ, dimTpu + 1 : dimJ)$;

% Como $\mathbf{J_{33}} = \mathbf{D}$ é uma matriz diagonal, sua inversa é trivial.
$for\ \ iin = 1 : dimG$
$\quad J33inv(iin, iin) = 1/J33(iin, iin);$
end
% O bloco $\mathbf{B}$ relevante depois da transformação $\mathbf{T_r}$:
$Trb = -J33inv * J32;$
$BTrb = J12 + J13 * Trb;$
$J11e21 = J(1 : dimTpu, 1 : dimT + dimp);$
$BJ = sparse(1 : dimTpu, 1 : dimTpu, 0);$
$BJ(1 : dimT + dimp + dimu, 1 : dimT + dimp) = J11e21;$
$BJ(1 : dimT + dimp, dimT + dimp + 1 : dimTpu) = BTrb.$
% Focando no bloco $\mathbf{B}$, já que os autovalores da matriz
% original são os mesmos:
$BM = M(1 : dimTpu, 1 : dimTpu);$
% Redividindo a matriz $(4N - 2, 4N + 2, 4N - 2)$
$dim13 = 4 * NELE - 2;$ % dimensões dos blocos extremos 1 e 3.
$dim2 = 4 * NELE + 2;$ % dimensão do bloco central 2, para onde
% migram os autovalores finitos.
$dimBJ = length(BJ);$
$BJ11 = BJ(1 : dim13, 1 : dim13);$
$BM11 = BM(1 : dim13, 1 : dim13);$
$BJ12 = BJ(1 : dim13, dim13 + 1 : dim13 + dim2);$
$BM12 = BM(1 : dim13, dim13 + 1 : dim13 + dim2);$
$BJ22 = BJ(dim13 + 1 : dim13 + dim2, dim13 + 1 : dim13 + dim2);$
$BM22 = BM(dim13 + 1 : dim13 + dim2, dim13 + 1 : dim13 + dim2);$
$BJ13 = BJ(1 : dim13, dim13 + dim2 + 1 : dimBJ);$
$BJ23 = BJ(dim13 + 1 : dim13 + dim2, dim13 + dim2 + 1 : dimBJ);$
$BJ31 = BJ(dim13 + dim2 + 1 : dimBJ, 1 : dim13);$
$BJ32 = BJ(dim13 + dim2 + 1 : dimBJ, dim13 + 1 : dim13 + dim2);$
% as inversas dos blocos:
$BJ13inv = inv(BJ13);$
$BJ31inv = inv(BJ31);$
$TTrb = -BJ31inv * BJ32;$
$TTlb = -BJ23 * BJ13inv;$
% **BMev** e **BJev** são as correspondentes das
% matrizes massa e jacobiana no bloco central $\mathbf{B_{22}}$, de
% dimensão $4N + 2$
$BJev = TTlb * (BJ11 * TTrb + BJ12) + BJ22;$
$BMev = TTlb * (BM11 * TTrb + BM12) + BM22;$

%Sendo $\mathbf{M_{22}} = \mathbf{BMev}$ não mais singular, calcula-se sua inversa
% pra transformar o problema de autovalor generalizado em
% cássico.
$Mi = inv(BMev);$
$SubMiJ = -Mi * BJev;$
$SubMiJ = full(SubMiJ);$
$subg = eig(SubMiJ);$

O vetor formado pelos autovalores do problema reduzido **subg** foi comparado na seção 6.7 com os da matriz original, mostrando uma excelente concordância.

B.2
Recuperação dos autovetores originais a partir do problema reduzido vindo da formulação unidimensional com líquido viscoelástico.

Vamos recuperar o autovetor original a partir do reduzido. Esse código tem que estar no mesmo programa que o código para redução da matriz, pois usa as variáveis nele definidas.

% o autovetor **autovetf** e **autovet22** são a coluna k, referente ao autovalor σ_k, das matrizes formadas pelos autovetores calculados no problema original e reduzido, respectivamente:
$a(k) = -0.055279;$
$b(k) = 0.95131$
$\sigma_k = a(k) + i * b(k)$
% o autovetor do problema original:
$for\ \ iaa = 1 : n$
    $if(norm(Dnew(iaa, iaa) - (a(k) + i * b(k))) < 10^{-5})$
        $autovalf = Dnew(iaa, iaa)$
        $vet1 = iaa;$
    end
end
$autovetf = (Vnew(:, vet1))/norm(Vnew(:, vet1));$
% o autovetor do problema reduzido:
$for\ \ iab = 1 : length(BJev)$

$if(norm(D22(iab,iab)-(a(1)+i*b(1)))<10^{-5})$
$autoval22 = D22(iab,iab)$
$vet22 = iab;$
end
end
$autovet22 = (V22(:,vet22))/norm(V22(:,vet22));$
$TTr = speye(dimBJ);$ % formando a matriz $\mathbf{TT_r} = \mathbf{M_r}$ necessária
% para a recuperação do autovetor original.
$TTr(1:dim13, dim13+1:dim13+dim2) = TTrb;$
% as transformações não tinham sido feitas em $\mathbf{B}_{12}$.
$B12tilda = (BJ11+autoval22*BM11)*TTrb+(BJ12+autoval22*BM12);$
$autovetT1 = zeros(dim13,1);$
$autovetT2 = autovet22;$
$autovetT3 = -BJ13inv*B12tilda*autovetT2;$
$autovetT = [autovetT1; autovetT2; autovetT3];$
$autovet = TTr*autovetT;$
$autovet = autovet/norm(autovet);$
$c3 = zeros(dimG,1);$
$vet = [autovet; c3];$
$Tr = speye(dimJ);$
$Tr(1:dimT+dimp, dimT+dimp+1:dimTpu) = Trb;$
$c = Tr*vet;$
% colocando zero nas posições referentes às condições de
% contorno do autovetor transformado.
$autovetTfim = zeros(n,1);$
$icv = 1;$
$for\ \ iv = 1:8*NELE$
$autovetTfim(iv) = c(icv);$
$icv = icv+1;$
end
$icv = icv+1;$
$for\ \ iv = 8*NELE+2:10*NELE-1$
$autovetTfim(iv) = c(icv);$
$icv = icv+1;$
end
$autovetTfim(10*NELE+1) = c(icv);$
$foriv = 10*NELE+3:12*NELE$
$autovetTfim(iv) = c(icv);$
$icv = icv+1;$

end
$for\ \ iv = 12 * NELE + 2 : n$
$\quad autovetTfim(iv) = c(icv);$
$\quad icv = icv + 1;$
end
$autovetTfim = autovetTfim/norm(autovetTfim);$

O autovetor **autovetTfim** recupera o tirado da matriz de autovetores do problema original referente ao mesmo autovalor σ_k. Fizemos um pós-processamento separando os autovetores em campos de velocidade e pressão. A seção 6.7.2 mostra a concordância obtida.