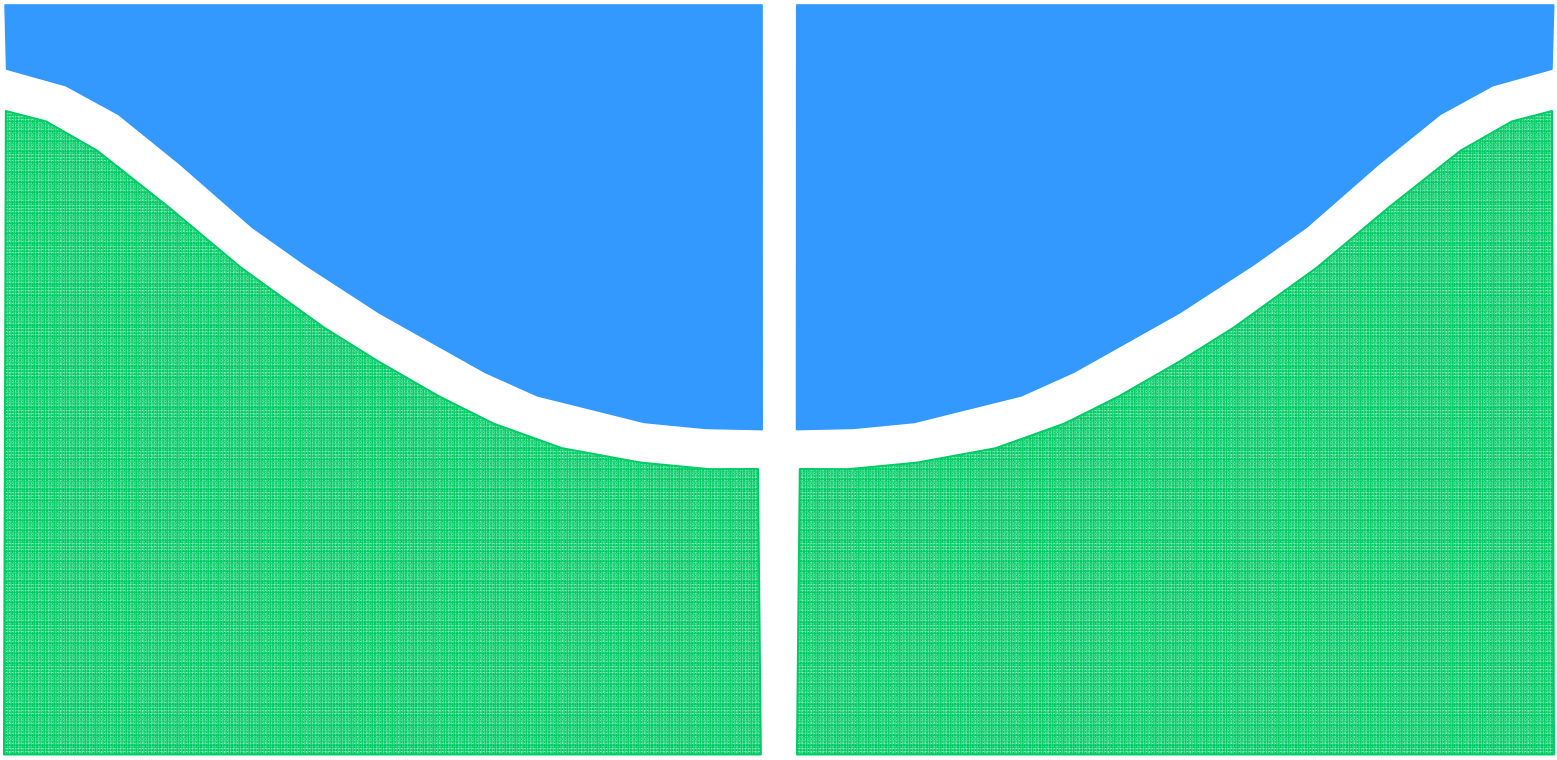

TRABALHO DE GRADUAÇÃO

SELEÇÃO DE PARÂMETROS *ADI* PARA RESOLUÇÃO DA EQUAÇÃO DE LYAPUNOV

Tiago Costa Borges

Brasília, 20 de dezembro de 2007

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SELEÇÃO DE PARÂMETROS *ADI* PARA RESOLUÇÃO DA EQUAÇÃO DE LYAPUNOV

TIAGO COSTA BORGES

ORIENTADOR: FRANCISCO DAMASCENO FREITAS

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

BRASÍLIA/DF, 20 DE DEZEMBRO DE 2007

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SELEÇÃO DE PARÂMETROS *ADI* PARA RESOLUÇÃO DA EQUAÇÃO DE LYAPUNOV

TIAGO COSTA BORGES

MONOGRAFIA SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRICISTA.

APROVADA POR:

_____

Prof. Francisco Damasceno Freitas, Doutor. (ENE-UnB)
(Orientador)

_____

Prof. Luís Filomeno de Jesus Fernandes, Doutor. (ENE-UnB)
(Examinador Interno)

_____

Prof. João Yoshiyuki Ishihara, Doutor (ENE-UnB)
(Examinador Interno)

BRASÍLIA, 20 DE DEZEMBRO DE 2007

FICHA CATALOGRÁFICA

BORGES, TIAGO COSTA Seleção de Parâmetros ADI para Resolução da Equação de Lyapunov [Distrito Federal] 2007. viii, 67p, (ENE/FT/UnB, Engenheiro Eletricista, 2007). Monografia de Graduação – Universidade de Brasília. Faculdade de Tecnologia. Departamento de Engenharia Elétrica. 1. Equação de Lyapunov 2. Iteração ADI 3. Redução de Modelo 4. Seleção de parâmetro ADI I. ENE/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

BORGES, T. C. (2007). Seleção de Parâmetros ADI para Resolução da Equação de Lyapunov. Monografia de Graduação, Publicação ENE 02/2007, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 67p.

CESSÃO DE DIREITOS

AUTORES: Tiago Costa Borges.

TÍTULO: Seleção de Parâmetros ADI para Resolução da Equação de Lyapunov

GRAU: Engenheiro Eletricista ANO: 2007

_____

Tiago Costa Borges

Colônia Agrícola Águas Claras, chácara 13, lote 01.
71090-000 Brasília – DF – Brasil.

DEDICATÓRIA

À minha família e amigos.

AGRADECIMENTOS

Agradeço a Deus, pelo dom da vida e por todas as graças que tem me dado, sem Ele não seria capaz.

Aos meus amigos e familiares, por terem me acompanhado nesta trajetória, com carinho e afeto.

Aos professores do Departamento de Engenharia Elétrica, pelo conhecimento transmitido e o empenho com que levam esta preciosa missão de ensinar, e especialmente ao professor Damasceno, por sua atenção e dedicação na orientação deste projeto.

RESUMO

SELEÇÃO DE PARÂMETROS ADI PARA RESOLUÇÃO DA EQUAÇÃO DE LYAPUNOV**.**
Autor: Tiago Costa Borges
Orientador: Francisco Damasceno Freitas
Palavras-chave: equação de Lyapunov, redução de modelo, iteração ADI.
Brasília, dezembro de 2007.

O presente trabalho apresenta estudo sobre a técnica ADI para solução da equação de Lyapunov. São abordadas de forma preliminar algumas técnicas de resolução. No entanto o trabalho é voltado para a técnica ADI, a qual para a sua aplicação depende da escolha de parâmetros específicos. Dois destes métodos estudados aqui são o método de parâmetros ótimos, proposto inicialmente por Wachspress, e um método heurístico modificado, proposto por Peter Benner, Hermann Mena e Jean Saak, que combina as vantagens do método de Wachspress e Thilo Penzl enquanto procura evitar suas desvantagens.

São apresentados testes a respeito das duas técnicas consideradas para avaliar os desempenhos de cada uma.

SUMÁRIO

LISTA DE FIGURAS

1 – INTRODUÇÃO

1.1 – CONSIDERAÇÕES INICIAIS

A equação de Lyapunov apresenta papel importante em várias áreas da engenharia. Aplicações desta equação são encontradas, por exemplo, na redução da ordem de sistemas lineares e na solução iterativa de equações de Riccati, a análise de estabilidade de sistemas não lineares e no controle ótimo, dentre outras. Estas aplicações são encontradas em especial na área de sistemas de controle, como na análise de estabilidade de sistemas lineares e em controle ótimo.

O aprimoramento dos sistemas de controle conduz a um aumento na complexidade da representação do sistema e conseqüentemente, no incremento do número de variáveis. Por isso, os cálculos numéricos tornaram-se cada vez mais dispendiosos, ultrapassando muitas vezes a capacidade de processamento dos computadores. Surgiu então a necessidade de diminuição do número de variáveis e de estudos com relação aos de cálculo.

Com o objetivo de tornar possível o cálculo de sistemas que ultrapassavam a capacidade computacional da época, Donald Peaceman e Henry Rachford [10] trabalharam no desenvolvimento do método ADI (*Alternating Direction Implicit*) no ano de 1955. A proposta inicial deste método visava a solução implícita do problema de fluxo de calor, por meio de uma equação diferencial parcial parabólica, em duas dimensões. Em seguida estendida para três dimensões.

Atualmente, o método é utilizado em várias áreas de estudo, devido ao seu baixo custo computacional, rapidez na convergência e a adequabilidade para uso em processamento paralelo. Como exemplos, pode-se citar problemas para simulação de reservatórios de petróleo, em aplicações na astrofísica e na bioengenharia.

1.2 – OBJETIVO DO PRESENTE TRABALHO

O objetivo do presente trabalho é avaliar o resultado da resolução da equação de Lyapunov através da técnica que considera os parâmetros ADI. Com o uso do software

Matlab® foram implementadas duas rotinas de cálculo dos parâmetros que são apresentados nos apêndices A e B, que foram testados posteriormente em um algoritmo que resolve a equação [12].

O algoritmo implementado gera os parâmetros por diferentes métodos, e estes são usados em outro algoritmo que resolve a equação de Lyapunov. A análise de desempenho aborda as duas etapas.

O projeto apresenta alguns métodos de solução da equação de Lyapunov, com ênfase no método de iteração ADI [13] e a escolha de seus parâmetros. Os algoritmos implementados para o cálculo dos parâmetros foram propostos por Rachford [13] [5] e Peter Banner [9], enquanto que a resolução de equação é feita por algoritmo que utiliza rotinas do *LYAPACK* desenvolvido por Thilo Penzl [17] utilizando o software Matlab®.

1.3 - ORGANIZAÇÃO DO TRABALHO

O Capítulo 2 apresenta uma revisão do estado da arte sobre a equação de Lyapunov, descrevendo suas principais aplicações e os métodos mais comuns utilizados na sua resolução.

O capítulo 3 descreve o método ADI, e dois outros métodos derivados: o LR-ADI (juntamente com o método LR-Smith, que serve como base) e CF-ADI, propostos por Thilo Penzl [4] e Rebecca Li [11], respectivamente, além da descrição dos principais algoritmos para a escolha de parâmetros utilizados na iteração.

A metodologia dos testes realizados é apresentada no capítulo 4 e os resultados das simulações realizadas neste trabalho junto com a sua análise são apresentados no capítulo 5, com a comparação dos dois métodos de escolha de parâmetros.

No capítulo 6 são apresentadas as conclusões do trabalho.

Dois apêndices ao final mostram os códigos em Matlab® desenvolvidos para análise de desempenho das técnicas estudadas.

2 – EQUAÇÃO DE LYAPUNOV

Considere o seguinte sistema linear invariante no tempo caracterizado pelas seguintes equações:

$$\begin{aligned} \dot{\mathrm{x}}(\mathrm{t}) &= Ax(t) + Bu(t) \\ \mathrm{y}(\mathrm{t}) &= Cx(t) \end{aligned} \tag{2.1}$$

O vetor $\mathrm{x}(\mathrm{t})$ é o estado no instante de tempo *t* e tem *n* componentes. A entrada $\mathrm{u}(\mathrm{t})$ e a saída $\mathrm{y}(\mathrm{t})$ possuem, respectivamente, *m* e *p* componentes. As matrizes $A \in R^{nxn}$, $B \in R^{nxm}$, e $C \in R^{pxn}$, são respectivamente, a matriz de estado do sistema, a matriz de entrada, e a matriz de saída.

Se todos os autovalores da matriz *A* possuem parte real negativa, isto é, estão no semi-plano esquerdo do plano complexo, existe uma única solução positiva-definida para o sistema (2.1). Em regime permanente tem-se $X(\infty)$, que é a solução da equação algébrica de Lyapunov (EAL)

$$A^T X + XA + R = 0 \tag{2.2}$$

A solução de (2.2) é dada pela seguinte expressão:

$$X = \int_0^\infty e^{A^T t} R e^{At} dt \tag{2.3}$$

Esta integral é denominada graminiana do sistema. Caso a matriz *R* esteja na forma $C^T C$, pode-se definir

$$X_C \triangleq \int_0^\infty e^{A^T t} C^T C e^{At} dt \tag{2.4}$$

como sendo a graminiana de observabilidade do sistema linear (2.1), assim pode-se escrever a equação (2.2) da seguinte forma

$$A^T X_C + X_C A = -C^T C \tag{2.5}$$

Com o mesmo raciocínio, pode-se definir escrever a graminiana de controlabilidade do sistema, neste caso com $R = BB^T$ da seguinte forma

$$X_B \triangleq \int_0^\infty e^{At} BB^T e^{A^T t} dt \tag{2.6}$$

O que resulta em uma equação da forma

$$AX_B + X_B A^T = -BB^T \tag{2.7}$$

Estas duas formas, (2.5) e (2.7), da equação de Lyapunov são as mais comuns quando o sistema linear está representado em espaço de estados. Sendo as matrizes BB^T e C^TC simétricas e positivamente definidas, então as soluções dadas pelas graminianas de observabilidade e controlabilidade, X_C e X_B, também o serão.

A importância física dos autovetores dominantes das graminianas de observabilidade e de controlabilidade é que eles estão associados, respectivamente, a elementos mais sensíveis à saída e a elementos aos quais a entrada influencia mais.

2.1 – APLICAÇÕES DA EQUAÇÃO DE LYAPUNOV

2.1.1 – Redução de sistemas lineares

Em muitos casos a equação de Lyapunov é utilizada para reduzir sistemas dinâmicos representados na forma de espaço de estados, como o sistema (2.1). O propósito da redução do sistema é substituí-lo (as matrizes *A*, *B* e *C*) por um sistema de ordem menor, ou seja $\hat{A}, \hat{B}$ e $\hat{C}$, com $\hat{A} \in \mathbb{R}^{kxk}$, $\hat{B} \in \mathbb{R}^{nxk}$ e $\hat{C} \in \mathbb{R}^{qxk}$ e $k << n$. Porém, esta aproximação não pode apresentar comportamento muito diferente do sistema original, ou seja, o sistema aproximado deve comportar-se de maneira análoga ao sistema original, mas deve possuir ordem menor.

Para exemplificar a redução de um sistema linear, considerar-se-á aqui uma técnica de balanceamento truncado proposta por Safonov e Chiang [1], que faz uso das graminianas de controlabilidade, X_B, e de observabilidade, X_C. O truncamento balanceado faz parte de uma classe de métodos que procura identificar uma realização para o sistema original, que

identifique os estados menos importantes, permitindo depois a operação de truncamento de estados. Considere as matrizes F, U e Σ e as seguintes relações:

$$F^T F = X_B \tag{2.8}$$

$$U\Sigma U^T = FX_C F^T \tag{2.9}$$

onde U e Σ definem uma possível decomposição de $FX_C F^T$ em valores singulares. Considere que a partição de U e Σ sejam dadas por:

$$U \triangleq \begin{bmatrix} U_1 U_2 \end{bmatrix} \tag{2.10}$$

$$\Sigma \triangleq \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \tag{2.11}$$

Sendo $U_1 \in \mathbb{R}^{n x q}, \Sigma_1 \in \mathbb{R}^{q x q}$ com $q < n$. O modelo reduzido é dado por:

$$\begin{aligned} \hat{A} &= R_1^T A T_1 \\ \hat{B} &= R_1^T B \\ \hat{C} &= C T_1 \end{aligned} \tag{2.12}$$

em que R_1 e T_1 são obtidas a partir de F, U_1 e Σ_1 da seguinte forma:

$$T_1 \triangleq F^T U_1 \Sigma_1^{-1/4} \tag{2.13}$$

$$R_1 \triangleq F^{-1} U_1 \Sigma_1^{1/4} \tag{2.14}$$

A idéia central do truncamento balanceado é que estados pouco observáveis ou pouco controláveis têm pequena influência no sinal de saída. Entretanto pode ser difícil a identificação destes estados pois é freqüente que eles sejam simultaneamente pouco observáveis e muito controláveis ou muito observáveis e pouco controláveis. A solução para este problema de redução é encontrar uma realização em que as graminianas X_B e X_C do problema original sejam matrizes diagonais idênticas, denominada realização internamente balanceada que existem para todo sistema linear semelhante ao (2.1). Como a controlabilidade e observabilidade de um sistema estão intimamente relacionadas aos valores singulares em seus graminianos, estes valores são utilizados para determinar os

modos de menor influência. O modelo reduzido é obtido a partir do truncamento dos modos menos controláveis e menos observáveis de X_B e X_C.

2.1.2 – Controle ótimo

Considere um sistema linear como em (2.1). Dele pode-se derivar o seguinte problema linear-quadrático de controle ótimo [15]

$$\Upsilon(x_0, u) = \frac{1}{2}\int_0^\infty y^T(\tau)Qy(\tau) + u^T(\tau)Ru(\tau)d\tau \tag{2.15}$$

Com

$$Q = Q^T \geq 0 \; e \; R = R^T > 0 \tag{2.16}$$

Ou seja, a matriz Q positiva semi-definida e R positiva definida. A solução do problema de controle ótimo é representado pela realimentação linear

$$u(\tau) = -R^{-1}B^T P\, x(\tau) =: -K^T x(\tau), \tag{2.17}$$

Onde P é a solução da equação algébrica de Riccati

$$C^T QC + A^T P + PA - PBR^{-1}B^T P = 0, \tag{2.18}$$

A solução desta equação é denominada estabilizadora se cada autovalor da matriz $A - BR^{-1}B^T P$ tem a parte real negativa. Assumindo-se que existe somente uma solução estabilizadora da equação (2.18). O método de Newton é o procedimento padrão para encontrar a solução desta equação. Usando-se este método surge uma equação de Lyapunov, e a solução desta equação é uma aproximação em casos onde a solução explícita é muito volumosa para ser armazenada na memória. Outra vantagem é que em várias situações no âmbito do controle ótimo, deseja-se a matriz de realimentação (K) ao invés da solução do problema (P).

2.1.3 – Aplicações em sistemas de potência

Um problema de particular interesse no comportamento dinâmico de sistemas de potência é avaliar a estabilidade de um sistema [14]. Em uma aproximação deste problema, um passo importante é identificar grupos de geradores coerentes, ou seja, máquinas que

tendem a girar sincronamente quando o sistema sofre algum distúrbio. Trata-se de um problema complicado quando se estuda um sistema de grande porte constituído por muitas máquinas.

Um método usado para resolver este problema é usar alguns programas de análise transitória, gerar gráficos para a resposta do rotor e compará-los, um a um, para se obter as relações de máquinas coerentes. Como se pode imaginar, este método consome muito tempo.

Por outro lado, utilizando-se modelos simples e linearizados das máquinas, pode-se montar sistemas lineares que descrevem matematicamente o conjunto de máquinas e fazer assim a comparação para se identificar os grupos coerentes. A comparação de grupos é realizada via equação de Lyapunov

Outra aplicação da equação de Lyapunov em sistemas de potência é no estudo das oscilações fracamente amortecidas de potência [2], um dos maiores problemas nas operações em sistemas de potência. Consegue-se o amortecimento utilizando-se estabilizadores de sistemas de potência (*Power System Stabiliser - PSS*) ou pelo uso de dispositivos de sistemas de transmissão CA flexíveis (*Flexible AC Transmission Systems - FACTS*). Os estabilizadores *PSS* aumentam o torque de amortecimento de um gerador afetando seu controle de excitação, enquanto dispositivos *FACTS* fazem o amortecimento modulando o ângulo de potência equivalente do sistema.

FACTS são dispositivos concebidos a partir de eletrônica de potência e outros equipamentos estáticos que fazem o controle de um ou mais parâmetros em sistemas de transmissão em CA para melhoria da controlabilidade e aumento da capacidade da transferência de potência do sistema.

O principal problema é como controlar equipamentos *FACTS* específicos, em particular da família *UPFC (Unified Power Flow Controller)*. Em modelos de sistemas linearizados, como o controle do sistema é válido somente em torno de um ponto de operação, não se sabe como ele irá reagir quando o sistema mudar ou quando uma linha de transmissão ou gerador saírem, por exemplo. Sabendo-se que sistemas de potência bastante carregados usualmente exibem comportamento não linear, então a aproximação linear não

será válida. No entanto é possível utilizar uma aproximação linear e avaliar a estabilidade do ponto de operação utilizando-se a equação de Lyapunov.

2.2 – MÉTODOS DE SOLUÇÃO DA EQUAÇÃO DE LYAPUNOV

2.2.1 – Método da Quadratura numérica ou *Bartels-Stwart*

Considerando que a matriz *B* possui somente uma coluna, ou seja, é um vetor coluna, a solução da equação de Lyapunov é dada pela sua graminiana de controlabilidade [16]

$$X_B = \int_0^{\infty} e^{\tau A} B B^T e^{\tau A^T} d\tau \tag{2.19}$$

Pode-se resolver a integral por uma aproximação no intervalo de integração [0,s], onde *s* é escolhido de maneira a admitir-se um erro de valor tolerável.

$$X_B(s) = \int_0^{s} e^{\tau A} B B^T e^{\tau A^T} d\tau \tag{2.20}$$

Nota-se que a integração pode ser vista como o produto de dois termos $w(t)w^T(t)$, onde $w(t)=e^{tA}B$. Aproxima-se esta função por $w_m(t) = q_m(At)B$, onde q_m é um polinômio de grau $(m-1)$. Então pode-se escrever a aproximação da seguinte maneira

$$X_m(s) = \int_0^{s} w_m(\tau) w_m^T(\tau) d\tau \tag{2.21}$$

A precisão da aproximação depende do valor de *s* e o grau do polinômio q_m. Considerando $V_m = [b, Ab, ..., A^{m-1}b]$ e $q_m(t) = \alpha_0 + \alpha_1 t + ... + \alpha_{m-1} t^{m-1}$. Então pode-se escrever $w_m(t) = V_m z_m(t)$, onde $z_m(t) = [\alpha_0, \alpha_1 t, ..., \alpha_{m-1} t^{m-1}]^T$ e daí calcular

$$X_m(s) = V_m \left(\int_0^{s} z_m(t) z_m^T(t) dt \right) V_m^T = V_m G_m V_m^T \tag{2.22}$$

onde G_m é uma matriz *mxm* formada da seguinte forma

$$g_{ij}^{(m)} = \frac{\alpha_{i-1}\alpha_{j-1}s^{i+j-1}}{i+j-1} \tag{2.23}$$

As aproximações de X_m terão a forma

$$X_m = V_m G_m V_m^T \tag{2.24}$$

Onde V_m é um conjunto de vetores $[v_1, v_2, ..., v_m]$ fixos e G_m uma matriz *mxm* arbitrária. O conjunto dessas matrizes é um subespaço do espaço de matrizes *NxN*. O alcance de qualquer matriz neste subespaço está incluso no subespaço $K = span\{V_m\}$, onde o subespaço das matrizes de (2.24) estão unicamente definidos pelo alcance *K* destas matrizes. Então as matrizes que tenham este formato serão denotadas por $Z_m(K_m)$, onde V_m é a base do subespaço *K*.

Ou seja, a aproximação $Z_m(K_m)$ onde K_m é o subespaço de Krylov $K = span\{b, Ab, ..., A^{m-1}b\}$, irá convergir para *X* com *m* tendendo ao infinito. Se encontrarmos uma boa aproximação para *w(t)*, em um intervalo [0,*s*], com um polinômio de baixo grau, Z_m será uma boa aproximação para X_m, contudo este cálculo não é muito utilizado por apresentar instabilidade.

Para resolver a integral (2.20), primeiramente vamos assumir que o valor do intervalo de integração *s* já foi computado e pode-se resolver *w(t)* em *m* pontos igualmente espaçados

$$t_i = (i-1)\frac{s}{m-1}, i = 1,2,...,m \tag{2.25}$$

Chega-se a seguinte fórmula para se resolver a integral (2.21)

$$X_m = \sum_{i=1}^{m} \delta_i w(t_i) w^T(t_i) \tag{2.26}$$

Onde os δ_i's são os coeficientes de quadratura. De outra forma

$$X_m = W_m \Delta W_m^T \tag{2.27}$$

Sendo $W_m = [w(t_1), w(t_2), ..., w(t_m)]$ uma matriz *Nxm* e $\Delta = diag(\delta_1, \delta_2, ..., \delta_m)$. Note que $w(t_1)=b$, de forma que para se resolver (2.26), precisa-se computar e armazenar $w(t_i)$ com i = 2,3,...,*m*.

A matriz X_m não deve ser calculada explicitamente, desde que ela geralmente é uma matriz densa de ordem *NxN*, só é necessário gravar W_m e Δ. De fato é importante saber que quando o objetivo é aproximar *Xv*, onde *v* é algum vetor, não é necessário se armazenar nem mesmo W_m. Neste caso a aproximação $x = X_m v$ pode ser acrescida de $x := x + (\delta_i w^T(t_i)v)w(t_i)$ toda vez que for gerado um novo $w(t_i)$, descartando-se o último gerado.

2.2.2 – Método de Hammarling

Hammarling encontrou um método de se computar os fatores de Cholesky da solução *X*, dada por (2.20) ,diretamente [3], bastando a mesma ser ao menos semi-definida. Este método é baseado na propriedade de que estruturas triangulares permitem substituições em elementos anteriores ou posteriores, ou seja, existe um algoritmo de estrutura recursiva para sistemas triangulares.

Considerando a equação de Lyapunov da seguinte forma, com matrizes complexas

$$AX + XA^H + D = 0 \tag{2.28}$$

Onde o expoente *H* denota o conjugado transposto e $D = D^H$, seja

$$A = QRQ^H \tag{2.29}$$

Suponha a decomposição de Schur da matriz *A*, $\tilde{X} = Q^H X Q$, $\tilde{D} = Q^H D Q$. Então a equação (2.28) se torna

$$R\tilde{X} + \tilde{X}R^H + \tilde{D} = 0 \tag{2.30}$$

Particiona-se *R*, $\tilde{X}$ e $\tilde{D}$ da seguinte forma

$$R = \begin{bmatrix} R_1 & r \\ 0 & \lambda_n \end{bmatrix}, \quad \tilde{X} = \begin{bmatrix} X_1 & x \\ x^H & x_{nn} \end{bmatrix}, \quad \tilde{D} = \begin{bmatrix} D_1 & d \\ d^H & d_{nn} \end{bmatrix} \tag{2.31}$$

Onde $R_1, X_1, D_1 \in \mathbb{C}^{(n-1)x(n-1)}$; $r, x, d \in \mathbb{C}^{n-1}$. Então, a partir da equação (2.30) tira-se as três equações seguintes

$$(\lambda_n + \bar{\lambda}_n)x_{nn} + d_{nn} = 0 \tag{2.32}$$

$$(R_1 + \bar{\lambda}_n I)x + d + x_{nn}r = 0 \tag{2.33}$$

$$R_1 X_1 + X_1 R_1^H + D_1 + rx^H + xr^H = 0 \tag{2.34}$$

Da equação (2.32), tem-se

$$x_{nn} = -d_{nn} / (\lambda_n + \bar{\lambda}_n) \tag{2.35}$$

Substituindo x_{nn} na equação (2.33) calcula-se x, que, depois de determinado pode ser substituído em (2.34), que se torna uma equação de Lyapunov com a mesma estrutura, só que de ordem *(n-1)*. Pode se seguir o processo até se chegar a uma matriz R_1 de ordem 1.

Hammarling se baseou nesta propriedade de redução recursiva e percebeu que quando $D = BB^T$, é possível se computar o fator de Cholesky de X diretamente sem a necessidade de se formar a matriz D. A matriz solução X é formada a partir da decomposição UU^H e o objetivo é computar U diretamente.

Assumindo-se controlabilidade, particiona-se U de forma adequada e a partir de procedimento que segue o mesmo padrão que a anterior, chega-se a um sistema linear com a mesma propriedade recursiva:

$$(\lambda + \bar{\lambda})\tau^2 + b^H b = 0 \tag{2.36}$$

$$(R_1 + \bar{\lambda} I)u + \tau r + \frac{1}{\tau}\hat{B}_1 b = 0 \tag{2.37}$$

$$R_1(U_1 U_1^H) + (U_1 U_1^H)R_1^H + \hat{B}_1 \hat{B}_1^H = 0 \tag{2.38}$$

Onde $U_1 \in \mathbb{C}^{(n-1)(n-1)}, u \in \mathbb{C}^{(n-1)}$ e $\tau \in \mathbb{C}$. Se R é estável, tem-se da equação (2.36) que $\tau = \frac{\| b \|_2}{\sqrt{-2real(\lambda)}}$. Como por construção tem-se $\hat{B}_1 = B_1 - \frac{1}{\tau} u b^H$, pode-se reescrever a seguinte relação

$$(R_1 + \bar{\lambda} I)u = -\pi r - \frac{1}{\tau} B_1 b \tag{2.39}$$

Desta equação tira-se o valor de *u*. Substituindo-se *u* e τ em $\hat{B}_1$, percebe-se que (2.38) é uma equação triangular de Lyapunov de ordem *(n-1)*. Assim pode-se resolver esta equação para U_1, usando a mesma técnica recursiva até se chegar a R_1 de ordem 1, quando todos os elementos de *U* serão conhecidos. A recursividade também existe para sistemas com matrizes reais.

2.2.3 – Método de Smith

Assume-se a equação da seguinte forma [4]

$$XA + A^T X = -BB^T \tag{2.40}$$

Com a matriz *A* estável, autovalores no semi-plano complexo negativo, a solução *X* é única, simétrica e positivamente semi-definida. Sabe-se que os métodos mencionados não consideram a esparsialidade das matrizes. Eles são considerados métodos padrões para a solução de equações de Lyapunov de matrizes densas e pequenas, mas são bastante limitados quando a equação envolve sistemas de grande porte e esparsos. Quando o problema é desta natureza, usam-se métodos numéricos de iteração que não destroem a esparsidade do sistema linear.

O método de Smith deriva da equação de Stein

$$X - S^T X S = T \tag{2.41}$$

Onde

$$\begin{aligned} S &= (A - pI)(A + pI)^{-1} \\ T &= -2p(A + pI)^{-T} BB^T (A + pI)^{-1} \end{aligned} \tag{2.42}$$

Que é equivalente a equação (2.40) no caso de *p*'s reais e negativos. Assim, com a seqüência $\{X_i^S\}_{i=0}^{\infty}$ gerada a partir de (2.41) converge para a solução de *X*, podendo-se escrever

$$X_0^S = 0, \quad X_{i+1}^S = T + S^T X_{i+1}^S S \tag{2.43}$$

$$X_i^S = \sum_{j=1}^{i} (S^{j-1})^T T S^{j-1} \tag{2.44}$$

Pode-se demonstrar que o método de Smith é equivalente ao método ADI quando os parâmetros não se alteram, ou seja $p = p_1 = p_2 = ...$, mas geralmente sua convergência é mais lenta do que a iteração ADI com parâmetros não constantes. O método se tornou popular devido a uma versão acelerada, denominado método Smith quadrado, pois possui taxa de convergência quadrática. Mas deve-se ter cuidado ao usar este método em equações com matrizes grandes e esparsas, pois em seus cálculos formam-se matrizes densas, mesmo que a matriz *A* seja esparsa, e em cada iteração esta matriz densa é elevada ao quadrado, tornando o processamento muito oneroso do ponto de vista computacional.

Se a estrutura da matriz *A* permite a solução eficiente de sistemas lineares do tipo $(A^T + pI)x = y$, ambos métodos ADI e Smith podem ser adotados como técnicas para solução de matrizes esparsas. Porém, se a ordem da matriz se torna muito grande, impossibilitando o seu armazenamento em memória, estes métodos tornam-se inviáveis para a solução da equação de Lyapunov para sistemas grandes e esparsos. Neste caso existem apenas poucos métodos que produzem uma aproximação de baixo *rank* da solução *X*. As iterações são gravadas implicitamente de forma fatorizada, diminuindo consideravelmente a necessidade de memória. Porém a desvantagem é que a precisão da aproximação não é muito boa.

3 – MÉTODO ADI

O método ADI (*Alternate Direction Implicit*) é um método iterativo para a solução de grandes sistemas de equações lineares, dentre eles a equação algébrica de Lyapunov. Foi apresentado por Peaceman e Rachford [10] em 1955. A primeira aplicação deste método foi na solução de problemas discretos de contorno em equações parciais diferenciais elípticas. Pode-se citar como exemplo o seguinte sistema linear:

$$(H+V)u = b \tag{3.1}$$

Onde H e V são matrizes esparsas. Em particular, dependendo da ordem dos valores desconhecidos do vetor u, H é uma matriz bloco-diagonal com blocos tridiagonais e V é uma matriz tridiagonal com blocos diagonais, ou vise-versa. A iteração ADI para a resolução deste sistema é dada por

$$(H + p_j I)u^{(j+0,5)} = (p_j I - V)u^{(j)} + b \tag{3.2}$$

$$(p_j I - V)u^{(j+1)} = (H + p_j I)u^{(j+0,5)} - b \tag{3.3}$$

Para a solução da equação de Lyapunov, tem-se $V=A$, $H=A^T$, $u=X$ e $b=BB^T$, e o seguinte algoritmo [13]:

Algoritmo 1 – *método ADI*

Entrada: A, B

1. *diagonalize as matrizes A e B se elas não forem triangulares*
2. *senão faça* $\bar{A} := A, \bar{B} := B.$
3. *escolha os parâmetros ADI, com a parte real negativa, ou seja,* $\{p_1,...,p_j\}$, $Re\{p_j\} < 0.$
4. *o valor inicial de* X_j *deve ser nulo.*
5. *para j = 1,2,...J, faça*

$$(\bar{A} + p_j I)\,\bar{X}_{j-0,5} = -BB^T - \bar{X}_{j-1}(\bar{A}^T - p_j I) \quad (3.a)$$

$$(\bar{A} + p_j I)\,\bar{X}_{j} = -BB^T - \bar{X}_{j-0,5}(\bar{A}^T - p_j I) \quad (3.b)$$

6. *se A foi diagonalizada, a solução é:*
7. $X_J^{adi} := S^{-1}\,\bar{X}_J\,S^{-T}$,
8. *senão faça* $X_J^{adi} = \bar{X}_J.$

Saída: $X_J^{adi} \in \mathbb{R}^{nxn}$, $X_J^{adi} \approx X$.

A iteração resulta na aproximação X_J^{adi} da solução X. A diagonalização é feita da seguinte forma:

Seja S a matriz de autovetores da matriz A

$$\bar{A} := SAS^{-1} \quad (3.4)$$

$$\bar{B} := SB$$

Os parâmetros p_j, $Re\{p_j\} < 0$, são conhecidos como parâmetros ADI. Para se manter a aproximação ADI X_J^{adi} um número real, assume-se que os parâmetros p_j são reais ou vêm

como parte de um par complexo conjugado. Por $\bar{A}$ ser estável, desde que *A* também seja estável, e *Re{p*$_j$*}<0* para todo *j*, $(\bar{A}+ p_j I)$ é não singular e existe solução para as equações (3.a) e (3.b) para todo *j*. A matriz intermediária $\bar{X}_{j-0,5}$ pode não ser simétrica , mas $\bar{X}_{j-1}$ e $\bar{X}_j$ são simétricas.

Existem dois produtos e duas soluções entre matrizes em cada iteração ADI. A matriz $\bar{X}_{j-1}$ e BB^T são simétricas e geralmente cheias. Tem-se o produto $(\bar{A}+ p_j I)\bar{X}_{j-0,5}$, e o resultado é transposto. A primeira solução matricial é $(\bar{A}+ p_j I)\bar{X}_{j-0,5} = -BB^T - \bar{X}_{j-1}(\bar{A^T} - p_j I)$, cuja solução $\bar{X}_{j-0,5}$ também é uma matriz. Esta solução matricial pode ser feita resolvendo-se n sistemas lineares com a matriz $(A+ p_j I)$ e as colunas de $-BB^T - \bar{X}_{j-1}(\bar{A^T} - p_j I)$ por n vezes. Então, em cada passo da iteração, $(\bar{A}+ p_j I)$ é multiplicada por duas matrizes completas, e ***2n*** sistemas lineares são resolvidos com a matriz $(\bar{A}+ p_j I)$.

Por isto, a matriz *A* deve ser primeiramente convertida à forma tridiagonal antes de se prosseguir com a iteração, para se evitar esses produtos entre matrizes cheias e a solução de tantos sistemas lineares, também cheios. Entretanto, sabe-se que a tridiagonalização de uma matriz não simétrica pode ser instável.

Para isso será necessário um custo computacional O(n^3) por iteração. Se $v \mapsto \bar{A}v, v \in \mathbb{R}^n$ necessita de um custo O(n), então as duas soluções matriciais podem ser feitas com um custo O(n^2). Se a matriz original *A* é esparsa, o método ADI é vantajoso, pois não se faz necessária a redução de *A* para qualquer forma especial, e os seus requisitos de trabalho se tornam O(Jn^2), e frequentemente *J<<n*. A complexidade do algoritmo ADI, no caso da matriz *A* não ser esparsa, é o trabalho O(n^3) + O(Jn^2), onde J é o número total de iterações. O(n^3) advêm da tridiagonalização de uma matriz *A* e a transformação $X_J^{adi} := S^{-1}\bar{X}_J S^{-T}$ para se obter a aproximação ADI final. Se a matriz *A* é esparsa ou já

está bem estruturada, não é necessário reduzi-la para a forma tridiagonal. O trabalho computacional $O(Jn^2)$ é determinado pelas J iterações do algoritmo. Em termos de complexidade, o método ADI, neste caso de matriz A cheia, compete com o método *Bartels-Stewart* e *Hammarling*, que também são métodos $O(n^3)$. Entretanto, a necessidade de se tridiagonalizar a matriz A pode representar um grave problema.

Se A é diagonalizável, então a aproximação X_J^{adi} possui o seguinte erro (*error bound*):

$$\left\| X_J^{adi} - X \right\|_F \leq \left(\|T\|_2^2 \right) \left(\left\| T^{-1} \right\|_2^2 \right) k(p)^2 \left(\left\| X_J^{adi} - X \right\|_F \right) \tag{3.4}$$

e

$$k(p) = \max_{x \in spec(A)} \left| \prod_{j=1}^{J} \frac{(p_j - x)}{(p_j + x)} \right| \tag{3.5}$$

Onde T é a matriz dos autovetores de A e $p=\{p_1, p_2,..., p_j\}$, são os parâmetros ADI.

3.1 – Métodos Low Rank-ADI e Low Rank-Smith

A idéia central destes métodos é a substituição das iterações *ADI* e *Smith* por produtos matriciais [4]:

$$X_i^A = Z_i^A \left(Z_i^A \right)^T \text{ e } X_i^A = Z_{il} Z_{il}^T \tag{3.6}$$

O método *LR-ADI* é derivado das equações (3.2) e (3.3). Usando (3.6) nesta fórmula, pode-se reescrevê-la em termos de Z_i^A

$$Z_i^A = \left[(A^T - p_i I)(A^T - p_i I)^{-1} Z_{i-1}^A \quad \sqrt{-2p_i}\,(A^T - p_i I)^{-1} \right] \tag{3.7}$$

Com o valor inicial:

$$Z_1^A = \sqrt{-2p_1} \left(A^T - p_1 I \right)^{-1} B \tag{3.8}$$

São acrescentadas colunas em Z_1^A a cada iteração, e o *rank* da aproximação de X é menor que $(m \cdot i)$, onde m é o número de colunas da matriz Z a cada iteração, e i é o

número de iterações. Embora o espaço na memória e o custo operacional cresça linearmente, o método *LR-ADI* representa uma boa solução para a equação de Lyapunov pois o número de iterações *ADI* é bem menor que o tamanho do sistema. Em particular, este método é interessante se existe uma série de parâmetros *ADI*, $\{p_i\}_{i=1}^{\infty}$, de diferentes valores.

Quando se tem um número limitado de parâmetros, o método *LR-Smith(l)* (método cíclico de Smith de baixo *rank*) é mais aconselhável por ser mais eficiente. Este método é constituído por duas partes. A primeira resolve a *l-ésima* iteração Z_l^A com os parâmetros *ADI* $\{p_1, p_2, ..., p_l\}$ pelo método *LR-ADI*. Inicia-se a iteração pela matriz

$$\begin{aligned} Z^{(l)} &= Z_l^A \\ Z_l &= Z^{(l)} \end{aligned} \tag{3.9}$$

Depois, inicia-se a iteração *LR-Smith(l)*

$$\begin{aligned} Z^{(i+1)l} &= S^T Z^{(i \cdot l)} \\ Z_{(i+1)l} &= \begin{bmatrix} Z_{il} & Z^{(i+1)l} \end{bmatrix} \end{aligned} \tag{3.10}$$

Onde $S = \prod_{j=1}^{l} (A - p_j I)(A - p_j I)^{-1}$, o custo operacional por iteração é constante. Os métodos *LR-ADI* e *LR-Smith(l)* são matematicamente equivalentes sob o ponto de vista de $X_{il}^A = Z_{il} Z_{il}^T$ se forem usados os parâmetros $\{p_1, p_2, ..., p_l\}$ de modo cíclico na iteração *ADI*. Mesmo assim, é mais fácil computar X_{il} pelo método *LR-Smith(l)*, quando *n* é grande e *m* é pequeno.

Geralmente não se sabe qual o número de iterações *(l)* necessárias para se conseguir a resposta com a precisão determinada. Para isto, seria necessário calcular a norma de Frobenius da matriz residual em cada iteração. Como isto é impraticável para matrizes grandes, faz-se este cálculo da seguinte forma

$$\left\| A^T Z_{il} Z_{il}^T + Z_{il} Z_{il}^T A + B B^T \right\|_F = \left\| [A^T Z_{il} \; Z_{il} \; B][Z_{il} \; A^T Z_{il} \; B]^T \right\|_F = \left\| R_1 R_2^T \right\|_F \tag{3.11}$$

Onde R_1 e R_2 são matrizes resultantes da decomposição *QR*, com $Q_1 R_1 = [A^T Z_{il} \; Z_{il} \; B]$ e $Q_2 R_2 = [Z_{il} \; A^T Z_{il} \; B]$.

3.2 – Método Cholesky Factor-ADI

O método CF-ADI [11] é uma derivação do método ADI. Este método produz as mesmas aproximações que este. Mas é bem mais eficiente pois calcula seu resultado a partir dos fatores de Cholesky da iteração ADI, o que diminui o custo computacional quando comparado com o método ADI clássico. No algoritmo 1, tem-se dois produtos entre matrizes e duas soluções, estas operações matriciais tornam a operação muito mais dispendiosa do ponto de vista computacional. Operação entre matriz e vetor tornaria o cálculo muito mais simples. O primeiro passo para se derivar o método CF-ADI do método ADI é combinar as equações (3.a) e (3.b), para se obter

$$\begin{aligned} X_j = &-2p_j(A+p_jI)^{-1}BB^T((A+p_jI)^T)^{-1} + \\ &+(A+p_jI)^{-1}(A+p_jI)X_{j-1}(A+p_jI)^T((A+p_jI)^T)^{-1} \end{aligned} \tag{3.12}$$

Desde que a matriz inicial $X_0 = 0$ e assumindo-se que as matrizes são de baixo *rank*, pode-se representar X_j pelo produto

$$X_j = Z_j Z_j^T \tag{3.13}$$

Sendo r_b o número de colunas da matriz B, Z_j tem jr_b colunas e é um fator de Cholesky de X_j. Substituindo (3.13) em (3.12) tem-se:

$$Z_0 = 0 \tag{3.14}$$

$$\begin{aligned} Z_j Z_j^T = &-2p_j\left[\left(A+p_jI\right)^{-1}B\right]\left[\left(A+p_jI\right)^{-1}B\right]^T + \\ &+\left[\left(A+p_jI\right)^{-1}\left(A-p_jI\right)Z_{j-1}\right]\left[\left(A+p_jI\right)^{-1}\left(A-p_jI\right)Z_{j-1}\right]^T \end{aligned} \tag{3.15}$$

Como a o lado esquerdo da equação (3.15) é um produto externo e o lado direito é a soma de dois produtos externos, pode-se obter o fator Z_j combinando-se os dois fatores do produto externo do lado direito da seguinte forma:

$$Z_j = \left[\sqrt{-2p_j}\left\{\left(A+p_jI\right)^{-1}B\right\}, \left\{\left(A+p_jI\right)^{-1}\left(A-p_jI\right)Z_{j-1}\right\}\right] \tag{3.16}$$

Desta maneira pode-se reformular o algoritmo ADI, calculando-se em cada iteração os fatores Z_j, que como se percebe são produtos *matriz x vetor*, ao invés de X_j. Este será

calculado somente no fim da iteração, quando já estiver formada a matriz Z_j. Assim escreve-se a iteração com os fatores de Cholesky da seguinte forma:

$$Z_1 = \sqrt{-2p_1}\left(A + p_1 I\right)^{-1} B \tag{3.17}$$

$$Z_j = \left[\sqrt{-2p_j}\left(A + p_j I\right)^{-1} B, \left(A + p_j I\right)^{-1}\left(A - p_j I\right) Z_{j-1}\right] \tag{3.18}$$

O problema de escrever a iteração desta maneira é que a cada nova iteração, o fator Z_{j-1} é modificado pela multiplicação na esquerda por $\left(A + p_j I\right)^{-1}\left(A - p_j I\right)$, deste modo, o número de colunas que devem ser modificadas cresce em r_b a cada iteração. Assim, alguns passos adicionais são tomados afim de evitar estas multiplicações.

Escreve-se as J colunas do fator Cholesky Z_J da seguinte forma

$$Z_J = \left[S_J\sqrt{-2p_J}B, S_J\left(T_J S_{J-1}\right)\sqrt{-2p_{J-1}}B, \ldots, S_J T_J \cdots S_2\left(T_2 S_1\right)\sqrt{-2p_1}B\right] \tag{3.19}$$

Onde

$$S_i = \left(A + p_i I\right)^{-1}, \quad T_i = \left(A - p_i I\right) \tag{3.20}$$

Então a matriz Z_J se torna:

$$Z_J = \left[z_J, \ P_{J-1}\left(z_J\right), \ P_{J-2}\left(P_{J-1} z_J\right), \ldots, \ P_1\left(P_2 \cdots P_{J-1} z_J\right)\right] \tag{3.21}$$

Pois os fatores S_i e T_i comutam, ou seja

$$S_i S_j = S_j S_i, \quad T_i T_j = T_j T_i, \quad S_i T_j = T_j S_i \quad \forall i \in J \tag{3.22}$$

Tem-se que

$$z_J := \left(\sqrt{-2p_J}\right) S_J B = \sqrt{-2p_J}\left(A + p_J I\right)^{-1} B \tag{3.23}$$

$$\begin{aligned} P_l &:= \left(\frac{\sqrt{-2p_l}}{\sqrt{-2p_{l+1}}}\right) S_l T_{l+1} = \frac{\sqrt{-2p_l}}{\sqrt{-2p_{l+1}}}\left(A + p_l I\right)^{-1}\left(A - p_{l+1} I\right) = \\ &= \left(\frac{\sqrt{-2p_l}}{\sqrt{-2p_{l+1}}}\right)\left[I - \left(p_{l+1} + p_l\right)\left(A + p_l I\right)^{-1}\right] \end{aligned} \tag{3.24}$$

A aproximação da matriz X fica

$$X_J^{cf\ adi} = Z_J^{cf\ adi}(Z_J^{cf\ adi})^T \tag{3.25}$$

Que é a mesma aproximação X_J^{adi} resultante do método ADI clássico. Transcreve-se abaixo o algoritmo CF-ADI

Algoritmo 2 *– método CF-ADI*

ENTRADAS: A,B.

SAIDAS: $Z_J^{cfadi} \in C^{nxJm}, X \approx X_J^{cfadi} := Z_J^{cfadi}\left(Z_J^{cfadi}\right)^T \in R^{nxn}$

1. *Escolha dos parâmetros ADI* $\{p_1, \ldots p_{J\max}\}$, $\mathrm{Re}\{p_i\} < 0$ *(real ou pares complexos conjugados).*
2. *Defina:* $P_i = \left(\frac{\sqrt{-2p_{i+1}}}{\sqrt{-2p_i}}\right)\left[I - (p_{i+1} + p_i)(A + p_i I)^{-1}\right]$

   a. $z_1 = \left(\sqrt{-2p_1}\right)(A + p_1 I)^{-1} B,$

   b. $Z_1^{cfadi} = [z_1].$
3. *para* $j = 2, 3, \ldots, J_{\max}$

   a. $z_j = P_{j-1} z_{j-1},$

   b. *se* $\left(\|z_j\|_2 > tol_1 \quad ou \quad \frac{\|z_j\|_2}{\|z_{j-1}\|_2} > tol_2\right)$ *e* $(j \le J\max)$

   $Z_j^{cfadi} = [z_j \quad Z_{j-1}^{cfadi}].$

   senão $J = j - 1$, *pare.*

Fim

3.3 – SELEÇÃO DE PARÂMETROS

A seleção parâmetros ADI adequados é muito importante para o sucesso da iteração ADI, ou seja, para a sua rápida convergência. A maioria dos métodos cobre o espectro da matriz do sistema, *A*, pelo domínio $\Omega \subset \mathbb{C}_-$, e resolvem o problema de *min_max* com relação a Ω, ao invés de $\sigma(A)$, ou seja, consideram um domínio arbitrário no semi-plano

direito ao invés de considerar apenas os autovalores da matriz de estado. Nesta seção, descreve-se alguns métodos utilizado no cálculo destes parâmetros.

3.3.1 – Parâmetros ótimos

Parâmetros ótimos são uma função de J e resolvem o seguinte problema racional discreto de *min_max*: [13] [9]

$$\min_{p_1,p_2...,p_j} \max_{x\in\Re} \left| \prod_{j=1}^{J} \frac{(p_j - \lambda)}{(p_j + \lambda)} \right| \tag{3.26}$$

Entretanto, como o espectro da matriz *A* pode não estar facilmente disponível, o seguinte problema contínuo pode ser utilizado:

$$\min_{p_1,p_2...,p_j} \max_{x\in\Re} \left| \prod_{j=1}^{J} \frac{(p_j - x)}{(p_j + x)} \right| \tag{3.27}$$

Onde $\Re$ é uma região no lado esquerdo do plano complexo, e $\lambda_1(A),...,\lambda_n(A) \in \Re \subset \mathbb{C}^{-1}$. Se os autovalores de *A* forem estritamente reais e estiverem contidos no intervalo $-b \le \lambda_1(A),...,\lambda_n(A) \le -a \le 0$, então a solução do problema de *min_max* é conhecida, e é calculada como

$$\min_{p_1,p_2...,p_j} \max_{x\in[-b,-a]} \left| \prod_{j=1}^{J} \frac{(p_j - x)}{(p_j + x)} \right| \tag{3.28}$$

Cuja solução é descrita abaixo:

- definir os limites espectrais *a*,*b* e α

$$a = \min_i(\mathrm{Re}\{\lambda_i\}), \tag{3.29}$$

$$b = \max_i(\mathrm{Re}\{\lambda_i\}), \tag{3.30}$$

$$\alpha = \tan^{-1} \max_i\left(\frac{\mathrm{Im}\{\lambda_i\}}{\mathrm{Re}\{\lambda_i\}}\right) \tag{3.31}$$

Onde $\lambda_1, \lambda_2, ..., \lambda_n$ são os autovalores de (–*A*). Assume-se que o espectro de (–*A*) está inteiramente dentro do domínio das funções elípticas, determinado por *a*,*b* e α. Um algoritmo de seleção mais geral consiste no seguinte:

- definir

$$\cos^2 \beta = \frac{2}{1 + \frac{1}{2}\left(\frac{a}{b} + \frac{b}{a}\right)}, \tag{3.32}$$

$$m = \frac{2\cos^2 \alpha}{\cos^2 \beta} - 1 \tag{3.33}$$

Se m < 1, os parâmetros são complexos, ou seja, a matriz do sistema possui autovalores complexos. Em caso contrário, os parâmetros são reais. No caso de parâmetros reais, tem-se:

- define-se *k'* e *k*

$$k' = \frac{1}{m + \sqrt{m^2 - 1}}, \tag{3.34}$$

$$k = \sqrt{1 - k'^2} \tag{3.35}$$

- definir as integrais elípticas *K* e *v* como

$$F[\psi, k] = \int_0^{\psi} \frac{dx}{\sqrt{1 - k^2 \sin^2 x}} \tag{3.36}$$

$$K = K(k) = F\left[\frac{\pi}{2}, k\right] \tag{3.37}$$

$$v = F\left[\sin^{-1}\sqrt{\frac{a}{bk'}}, k'\right] \tag{3.38}$$

- definir o número de iterações

O número de iterações para se obter $k(p)^2 \leq \in_1$ (sendo $\in_1$ uma tolerância) é

$$J = \left[\frac{K}{2v\pi} \log \frac{4}{\in_1}\right] \tag{3.39}$$

- por fim, definir os parâmetros ADI, dados pela seguinte expressão

$$p_j = -\sqrt{\frac{ab}{k'}}\, dn\left[\frac{(2j-1)K}{2J}, k\right], \qquad j = 1, 2, \ldots, J \tag{3.40}$$

Onde *dn(u,k)* é uma função elíptica dada pela seguinte equação:

$$dn(x,k) = \sqrt{1 - k\sin^2(x)} \tag{3.41}$$

Caso existam valores complexos, define-se um espectro elíptico dual:

$$a' = \tan\left(\frac{\pi}{4} - \frac{\alpha}{2}\right), \tag{3.42}$$

$$b' = \frac{1}{\alpha'}, \tag{3.43}$$

$$\alpha' = \beta \tag{3.44}$$

Substituindo (3.42) em (3.32), encontra-se

$$\beta' = \alpha, \quad m' = \frac{2\cos^2\beta}{\cos^2\alpha} - 1 \tag{3.45}$$

Pela construção feita, *m'* deve ser maior que 1. Então, devemos calcular os parâmetros reais, $p_j^{'}$ para o problema dual. Os parâmetros complexos do espectro correspondente são dados por:

$$\cos\alpha_j = \frac{2}{p_j^{'} + \frac{1}{p_j^{'}}} \tag{3.46}$$

Para $j=1,2,\ldots,\left[\frac{1+J}{2}\right]$

$$p_{2j-1} = \sqrt{ab}\, e^{i\alpha_j}, \quad p_{2j} = \sqrt{ab}\, e^{-i\alpha_j} \tag{3.47}$$

3.2.2 – Parâmetros sub-ótimos

Este método foi apresentado por Thilo Penzl [4] e implementado no *LYAPACK* [17] com código em Matlab®.

Sabe-se que a escolha dos parâmetros ADI é de fundamental importância para o bom desempenho dos vários métodos de iteração ADI. Convencionalmente, a aproximação destes parâmetros é feita cobrindo-se o espectro da matriz A em um domínio $\Omega \subset C_-$ e então resolve-se o problema *minimax* do método ADI, aqui escrito de maneira mais generalizada pela seguinte fórmula [4]:

$$\{p_1,..., p_l\} = \arg\min_{\{p_1,..., p_l\} \subset C_-} \max_{t \in \sigma(A)} \frac{|r_l(t)|}{|r_l(-t)|} \tag{3.47}$$

Assim o problema é resolvido com relação a Ω ao invés de σ(A). Muitos métodos foram desenvolvidos com o uso de domínios mais generalizados [5], entretanto estas aproximações requerem o conhecimento de certos contornos do espectro da matriz. São raros os casos onde estes contornos podem ser encontrados analiticamente, assim, para a solução destes problemas recorre-se a métodos numéricos.

As aproximações são baseadas na teoria de aproximação, que é um ramo da matemática que estuda como funções podem ser melhor aproximadas por funções mais simples e os possíveis erros adquiridos devido a esta aproximação [6].

Se a matriz é simétrica, pode-se computar o espectro pelo método *QR*. Este método é utilizado para se achar os autovalores de uma matriz real, mas para uma matriz complexa tem-se muitas iterações e se torna computacionalmente cara. Para matrizes simétricas tri-diagonais, Hessenberg e de banda simétrica os resultados são computados de forma rápida.

No caso de matrizes não simétricas a aproximação mais simples vem da estimativa dos autovalores extremos para os valores da parte simétrica de *A* por iteração inversa, a qual resulta em uma fronteira para o espectro de *A* pelo teorema de Bendixon. Este procedimento não pode ser aplicado se a parte simétrica de *A* é indefinida, desde que o retângulo obtido por este procedimento não é um subsistema de C_-. Outra alternativa para a cálculo dos autovalores de *A* é o processo de Arnoldi, entretanto este método também pode falhar no caso indefinido, pois ele deve entregar estimativas com partes reais não negativas.

Neste capítulo determinar-se-á um vetor *P* de *l* parâmetros ADI sub-ótimos sem precisar calcular inicialmente uma região Ω do espectro. O algoritmo resultante está mais

ligado a heurística do que com a teoria da aproximação, e os resultados numéricos são bastante satisfatórios. Na verdade, através deste algoritmo, não é necessário conhecimento prévio do espectro da matriz *A*. Toda a informação sobre esta matriz é obtida através de pares de processo Arnoldi relacionados a *A* e à sua inversa. Escolhe-se o valor inicial do vetor *r* aleatoriamente. Os inteiros k_+ e k_- denotam o número de iterações no processo Arnoldi nos passos referentes às matrizes *A* e A^-, respectivamente. Fazendo-se $k = k_+$, as iterações do par **(*A*,*r*)** como uma equação são:

$$AV_k = A_{k+1}\tilde{H}_k \tag{3.48}$$

Com $V_k \in \Re^{n,k}$, $\tilde{H}_k \in \Re^{k+1,k}$, $V_1 \in spam\{r\}$, $V_{k+1}^T V_{k+1} = I_{k+1}$, $(V_{k+1})_{(1:n,1:k)} = V_k$. Além disso,

$$H_k := (\tilde{H}_k)_{(1:k,1:k)} = V_k^T A V_k \tag{3.49}$$

É uma matriz superior de Hessenberg e é chamada *matriz de Ritz* e os seus autovetores são os *valores de Ritz*. Sabe-se que $\Re_+ := \sigma(H_k)$ representa uma aproximação do espectro de *A* [7]. Repetindo-se o mesmo processo, mas agora com a inversa da matriz *A*, tem-se em $\Re_-$, os elementos que aproximam os autovalores de A^-**.** Consequentemente, o conjunto $\Re := \Re_+ \cup 1/\Re_-$ pode ser considerado uma estimativa do espectro de *A*.

Os valores de *Ritz* obtidos pelo processo de Arnoldi tendem a se localizar próximos aos autovalores mais distantes. Ou seja, formam a fronteira convexa do espectro. Os autovalores de grande magnitude são melhor aproximados que os autovalores mais próximos da origem, de pequena magnitude. Então, envolve-se o conjunto $1/\Re_-$ para se aproximar estes autovalores. Este procedimento pode resultar em um impacto significativo na velocidade de convergência da iteração.

A idéia central deste procedimento heurístico é substituir σ(**A**) por $\Re$ na equação (3.47), desde que $\Re \subset C_-$. Além disso, escolhemos os parâmetros ADI sub-ótimos $P:=\{p_1,...,p_l\}$ dentro dos elementos de $\Re$ por que a função

$$s_P(t) = \frac{|\, r_l(t)\,|}{|\, r_l(-t)\,|} = \frac{|\,(t-p_1)\cdot\ldots\cdot(t-p_l)\,|}{|\,(t+p_1)\cdot\ldots\cdot(t+p_l)\,|} \tag{3.50}$$

torna-se pequena sobre σ(**A**) se existe algum parâmetro p_i na proximidade de cada autovalor. Desde que a equação de Lyapunov a ser resolvida seja real, precisa-se de $P = \bar{P}$. Isto assegura que as aproximações da matriz *X*, $Z_{il}^{A} Z_{il}^{A^T}$ e $Z_{il} Z_{il}^{\ T}$, também são reais.

Baseado nisto, determina-se os elementos de *P* primeiramente detectando-se o elemento $\rho_i \in \Re$ que minimiza a função $s_{\{\rho_i\}}$ sobre $\Re$. O vetor *P* é inicializado tanto por $\{\rho_i\}$ ou $\{\rho_i, \bar{\rho}_i\}$. Depois aumenta-se o vetor *P* com os elementos ou pares de elementos de $\Re$. O máximo de s_P com respeito ao vetor *P* é substituído por um zero na função s_P refinada. Esta estratégia é apresentada no seguinte algoritmo. A notação *card(P)* representa o número de elementos do vetor *P*.

Algoritmo 3 *- (parâmetros ADI sub-ótimos)*

ENTRADAS: $\mathbf{A}$, l_0, k_+, k_-

SAÍDA: $\mathbf{P}$

1. escolha algum $r \in R^n$ *;*

2. faça k_+ *passos do processo de Arnoldi relativos a* ($\mathbf{A}$,*r) e compute o conjunto de valores Ritz* $\Re_+$ *;*

3. faça k_- *passos do processo de Arnoldi relativos a* ($\mathbf{A}^-$,*r) e compute o conjunto de valores Ritz* $\Re_-$ *;*

4. $\Re = \{\rho_1,...,\rho_{k_+ + k_-}\} \coloneqq \Re_+ \cup (1/\Re_-)$ *;*

5. se $\Re \not\subset C_-$ *, pare;*

6. Ache i *com* $\max_{t\in\Re} s_{\{\rho_i\}}(t) = \min_{\rho\in\Re} \max_{t\in\Re} s_{\{\rho\}}(t)$ *e faça*

$$P \coloneqq \begin{Bmatrix} \{\rho_i\} & : real \\ \{\rho_i, \bar{\rho}_i\} & : não\ real \end{Bmatrix};$$

enquanto $card(P) < l_0$

7. ache i com $s_\rho(\rho_i) = \max_{t\in\Re} s_{\{\rho\}}(t)$ *e estabeleça* $P \coloneqq \begin{Bmatrix} P \cup \{\rho_i\} & : real \\ P \cup \{\rho_i, \bar{\rho}_i\} & : não\ real \end{Bmatrix}$

Fim enquanto

O passo 5 pode ser omitido se $A+A^T$ é negativamente definida. Isso permite que se prove que $A^{-1}+A^{-T}$ assim como as partes simétricas das matrizes *Ritz* relacionadas a A e a A^{-1} são absolutamente negativas. De outra forma, este algoritmo pode falhar, apesar desta hipótese ainda não ter acontecido em nenhum teste numérico com o mesmo [4].

Se o algoritmo parar no passo 5, pode-se reiniciá-lo com um novo vetor aleatório *r* ou os valores de k_+ ou k_- podem ser aumentados. Este aumento é motivado pela observação

de que os valores de *Ritz* obtidos pelo processo de Arnoldi tendem a aproximar melhor o espectro da matriz se aumenta-se o número de iterações de Arnoldi. Aproximações mais sofisticadas envolverão técnicas de reinicialização implícita [8] para excluir os conjuntos $\Re_+$ e $\Re_-$ de elementos com parte real não negativa.

A importância de se usar valores *Ritz* relativos a A^{-1} está na convergência ao valor final da equação de Lyapunov. Utilizando-se os dois valores, relativos a A e A^{-1}**,** consegue-se uma rápida e linear convergência para a solução final, ao passo que utilizado-se apenas os valores relativos a A tem-se inicialmente rápida convergência e depois fica praticamente constante, como se pode ver na figura [4]:

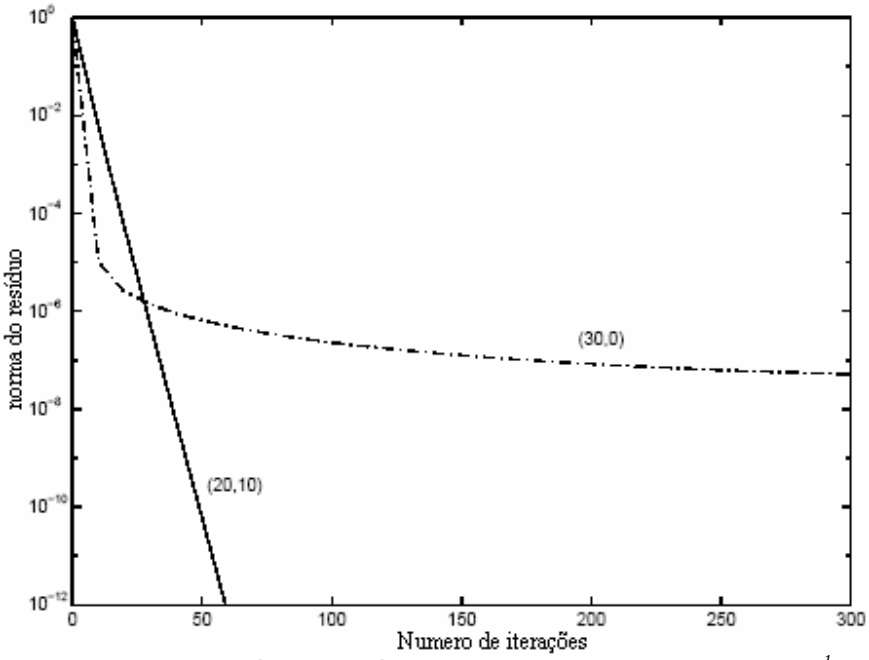

FIGURA 3.1- IMPORTÂNCIA DO CÁLCULO DOS VALORES RITZ RELATIVOS A A^{-1}

Aqui tem-se o resultado de dois processos de iteração para a solução de determinada equação de Lyapunov, onde fez-se $(k_+,k_-)=(20,10)$ e $(k_+,k_-)=(30,0)$. Nos dois casos, P consiste em $l=l_0=10$ elementos. Este fenômeno deve-se ao fato de que existem poucos autovalores que são mal aproximados por $\Re$. Como conseqüência , a função $s_P(t)$ dada pelo algoritmo 1 é praticamente 1 se t for igual a algum desses autovalores, mas é relativamente pequena se t pertence ao conjunto da maioria dos autovalores que são bem aproximados. Daí a componente do resíduo relacionada aos últimos autovalores computados decai rapidamente no primeiro estágio da iteração, entretanto a iteração é

atrasada no segundo estágio por pequeno número de autovalores, precariamente aproximados por $\Re_+$. Estes autovalores são tipicamente de pequena magnitude e frequentemente bem representados por elementos de $1/\Re_-$.

3.3.2.1 – Procedimento heurístico modificado

Este procedimento, dado por Benner, Mena e Saak em [9] e procura evitar os problemas dos métodos descritos e combinar suas vantagens. Tem-se como informação importante que precisa-se saber do método para parâmetros ótimos (Wachspress) [5] é a forma externa do espectro da matriz *A*, assim, este método irá aproximar esta região. Com esta aproximação, os dados de entrada do método de Wachspress *a*,*b* e α são calculados e os parâmetros ótimos do espectro aproximado são determinados. Assim estes parâmetros podem ser considerados sub-ótimos, mas a vantagem deste método está no custo operacional, que resulta parâmetros próximos aos valores ótimos com um custo operacional próximo ao do método heurístico.

Caso o espectro seja real, pode-se calcular simplesmente os valores limítrofes da reta real do espectro, ou seja, *a* e *b* por um processo de Arnoldi e iniciar o cálculo pelo método de Wachspress com estes valores, fazendo-se $\alpha = 0$. Desta forma, calcula-se apenas duas integrais elípticas completas, com baixo custo computacional, já que é um cálculo escalar com convergência quadrática.

Para o espectro complexo usa-se um passo a mais para se processar a iteração Arnoldi com maior eficiência. Já que os sistemas são estáveis computa-se os autovalores de maior e menor magnitude, e utilizas-se a média aritmética de sua parte real para se realizar um salto horizontal, de forma que o espectro fique centralizado ao redor da origem. Então o processo de Arnoldi é utilizado no espectro modificado para se escolher determinado número dos maiores autovalores. Isto ira incluir automaticamente os menores autovalores do sistema original antes de voltar o salto, evitando-se o uso da processo de Arnoldi para a inversa de *A* extensivamente, precisa-se apenas de uma aproximação grosseira dos menores autovalores para computar os valores de $\tilde{a}$ e $\tilde{b}$.

Algoritmo 4 *–(procedimento heurístico modificado)*

se $\sigma(A) \subset \mathbb{R}$ *então*

compute a fronteira espectral e faça $a = \min \sigma(-A)$ *e* $b = \max \sigma(-A)$

$k_1 = \frac{a}{b}, k = \sqrt{1 - k_1^2}$

$K = F\left(\frac{\pi}{2}, k\right), v = F\left(\frac{\pi}{2}, k_1\right)$, *sendo F definida por (3.36)*

Calcule o número de iterações J e os parâmetros de acordo com (3.39) e (3.40)

se não

compute $\tilde{a} = \min \sigma(\mathrm{Re}(-A))$, $\tilde{b} = \max \sigma(\mathrm{Re}(-A))$ *e* $c = \frac{\tilde{a} + \tilde{b}}{2}$

cacule os 'l' maiores autovalores $\hat{\lambda}_i$ *da matriz deslocada –A+cI por um processo de Arnoldi*

desloque estes autovalores de volta, ou seja, $\tilde{\lambda}_i = \hat{\lambda}_i + c$

calcule a,b e α a partir de $\tilde{\lambda}_i$ *como em (3.29), (3.30) e (3.31)*

se $m \geq 1$ *em (3.33)*

calcular os parâmetros através de (3.32) – (3.40)

se não (neste caso os parâmetros são complexos)

calule as variáveis duais

calcule os parâmetros para variáveis duais através de (3.32) – (3.40)

use (3.45) e (3.46) para calcular os pares complexos

fim se

fim se

4 - METODOLOGIA DO PROJETO

Após ter-se feito os esclarecimentos necessários sobre a teoria envolvida neste trabalho, descreve-se, a seguir, brevemente a metodologia do projeto. Elaborou-se duas rotinas que calculam os parâmetros ADI. A primeira é baseada no método de parâmetros ótimos (apêndice A), descrito na seção 3.3.1, e a segunda foi retirada do método desenvolvido por Peter Benner, Hermann Mena e Jens Saak [9] (apêndice B), descrito na seção 3.3.2.1.

Para a análise dos algoritmos que calculam os parâmetros ADI, primeiramente estudou-se um sistema de ordem reduzida, somente com autovalores reais. Um segundo sistema foi analisado agora com autovalores complexos. Finalmente realizaram-se testes em um sistema de grande porte, com uma matriz com 606 estados, gerada a partir de equivalente do sistema interligado brasileiro.

Após a aquisição dos parâmetros, estes foram inseridos em um algoritmo [12] que utiliza rotinas do *LYAPACK*.

5 – TESTES REALIZADOS

Simulação 1

Primeiro fez-se testes com a rotina *rli* (apêndice A), para a escolha de parâmetros ótimos, onde foi usado um sistema linear com uma matriz *A* de pequena ordem. Os dados de entrada são as matrizes *A, B, C* e *D* e a precisão desejada na escolha dos parâmetros.As matrizes são fictícias e têm ordem *n = 10*, e a matriz *A* possui apenas autovalores reais.

Com uma precisão igual a $\varepsilon_{par} = 10^{-10}$, obteve-se os seguintes parâmetros:

TABELA 5.1 – PARÂMETROS ADI SIMULAÇÃO 1

$\mathbf{p_1}$	-.9314
$\mathbf{p_2}$	-1.4121
$\mathbf{p_3}$	-0.5331
$\mathbf{p_4}$	-0.5798
$\mathbf{p_5}$	-1.4483
$\mathbf{p_6}$	-1.9443
$\mathbf{p_7}$	-1.9173

Sendo que os autovalores da matriz *A* são:

TABELA 5.2 – AUTOVALORES DA MATRIZ *A*

$\boldsymbol{\lambda_1}$	-1
$\boldsymbol{\lambda_2}$	-2
$\boldsymbol{\lambda_3}$	-1
$\boldsymbol{\lambda_4}$	-0,2
$\boldsymbol{\lambda_5}$	-1
$\boldsymbol{\lambda_6}$	-1,4
$\boldsymbol{\lambda_7}$	-0,6
$\boldsymbol{\lambda_8}$	-0,7
$\boldsymbol{\lambda_9}$	-1,2
$\boldsymbol{\lambda_{10}}$	-1

Pode-se perceber que os parâmetros estão dentro da faixa de abrangência dos autovalores. A partir daí, utiliza-se as rotinas do LYAPACK para se achar o sistema

reduzido. Os dados de entrada deste algoritmo são basicamente o número máximo de iterações e a precisão para o sistema reduzido, *max_it* e *tol*. A simulação foi feita com *max_it* = 5 e *tol* = 10^{-10}. A figura abaixo descreve o resíduo entre o modelo original e o modelo reduzido, com a redução sendo calculada pela graminiana de controlabilidade, ou seja, através da equação $AX + XA^T = -BB^T$. A mesma indica se o modelo reduzido está convergindo ou não para o modelo original.

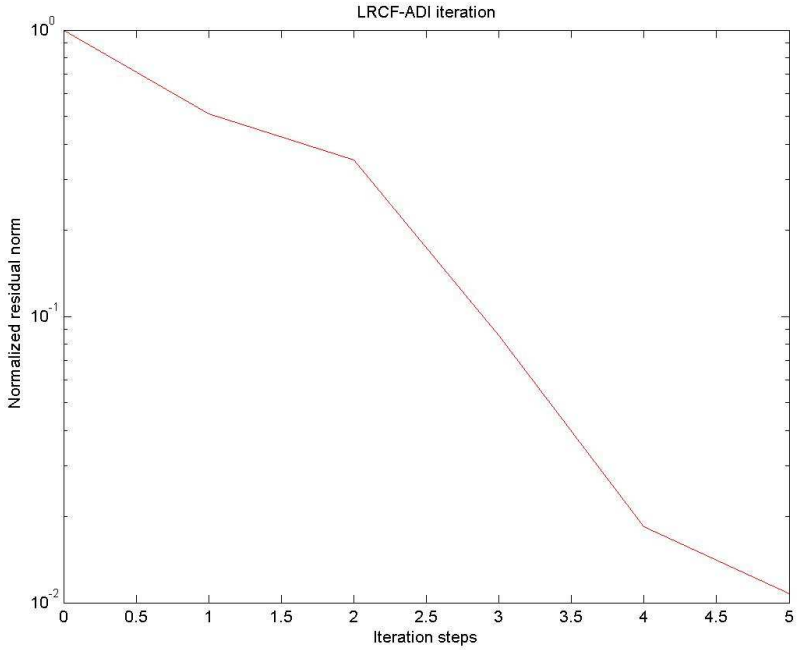

FIGURA 5.1- RESÍDUO BB^T DA SIMULAÇÃO 1

A solução da equação através dos parâmetros ADI calculados resultou na seguinte matriz X_B:

```
XB =

  Columns 1 through 7

    0.4255    0.1473    0.1518    0.5210         0         0         0
    0.1473    0.8183   -1.1391    2.7668         0         0         0
    0.1518   -1.1391    2.0426   -4.1014         0         0         0
    0.5210    2.7668   -4.1014    9.8952         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  Columns 8 through 10

         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
```

Já a solução exata feita pela função *lyap* do MATLAB ® resultou na seguinte matriz P_B:

```
PB =

  Columns 1 through 7

    0.4302    0.1396    0.1657    0.4987         0         0         0
    0.1396    0.8308   -1.1616    2.8030         0         0         0
    0.1657   -1.1616    2.0833   -4.1667         0         0         0
    0.4987    2.8030   -4.1667   10.0000         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  Columns 8 through 10

         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
```

Comparando os valores, vê-se que estão bem próximos, o que indica que os parâmetros ADI calculados estão representando bem o sistema original. A diferença entre as duas representações pode ser mensurada pela comparação entre as normas destas duas matrizes, feita pela expressão $\Delta X_B = \frac{\|P_B - X_B\|}{\|P_B\|}$. Neste caso, obteve-se $\Delta X_B = 0.0129$, indicando que a graminiana de observabilidade aproxima bem o sistema original.

A mesma comparação foi realizada, agora com a graminiana de observabilidade, isto significa resolvendo a equação $A^T X + XA = -C^T C$.

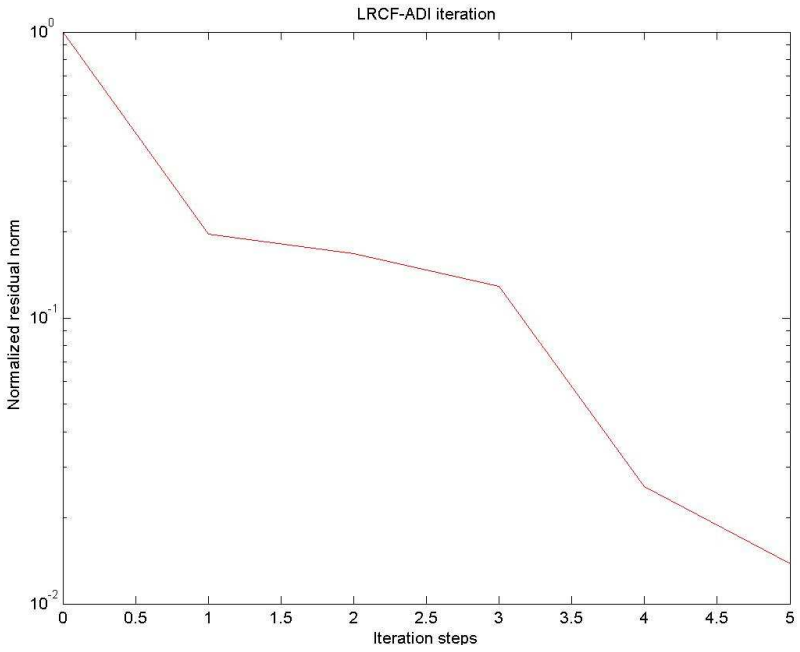

FIGURA 5.2 – RESÍDUO $C^T C$ DA SIMULAÇÃO 1

Aqui também geraram-se as matrizes X_C e P_C:

```
XC =

  Columns 1 through 7

    0.5000   -0.0833    0.5417   -0.2256    0.1015    0.0771   -0.0481
   -0.0833    0.0208   -0.0972    0.0734   -0.0389   -0.0283    0.0201
    0.5417   -0.0972    0.5972   -0.3098    0.1588    0.1144   -0.0840
   -0.2256    0.0734   -0.3098    0.7567   -0.6270   -0.3500    0.4794
    0.1015   -0.0389    0.1588   -0.6270    0.5597    0.2977   -0.4498
    0.0771   -0.0283    0.1144   -0.3500    0.2977    0.1644   -0.2301
   -0.0481    0.0201   -0.0840    0.4794   -0.4498   -0.2301    0.3752
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  Columns 8 through 10

         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0

PC =

  Columns 1 through 7

    0.5000   -0.0833    0.5417   -0.2257    0.1016    0.0771   -0.0482
   -0.0833    0.0208   -0.0972    0.0734   -0.0389   -0.0283    0.0202
    0.5417   -0.0972    0.5972   -0.3100    0.1590    0.1145   -0.0842
   -0.2257    0.0734   -0.3100    0.7750   -0.6475   -0.3583    0.5004
    0.1016   -0.0389    0.1590   -0.6475    0.5827    0.3070   -0.4734
    0.0771   -0.0283    0.1145   -0.3583    0.3070    0.1682   -0.2396
   -0.0482    0.0202   -0.0842    0.5004   -0.4734   -0.2396    0.3994
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  Columns 8 through 10

         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
```

Aqui também encontra-se valores da matriz solução aproximada bastante próximos da matriz solução original, o que é corroborado pela norma $\Delta X_C = 0.0336$. Assim,

podemos concluir também que neste caso a graminiana de observabilidade aproxima melhor o sistema original.

A ordem do sistema reduzido foi de $n_{red} = 4$, ou seja, o mesmo sistema pode ser representado agora com ordem 60% menor. A comparação dos sistemas reduzido e original pode ser avaliada pela figura 5.3.

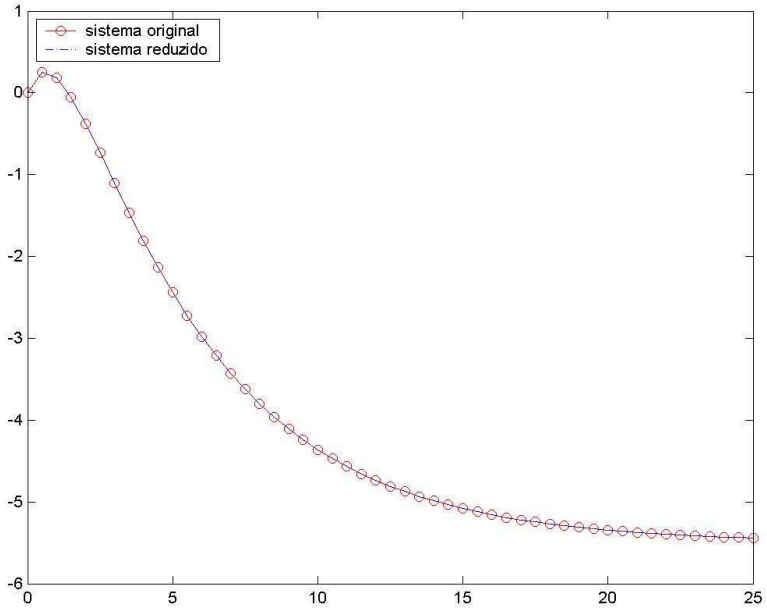

FIGURA 5.3 – COMPARAÇÃO ENTRE SISTEMAS

Neste caso, observa-se que o sistema reduzido acompanha perfeitamente o sistema original.

Simulação 2

Então utilizou-se a rotina *rli* (anexo A) par avaliar o desempenho de um sistema linear, também fictício, onde a matriz *A*, também de pequeno porte, possui autovalores complexos. Não foi modificado nenhum parâmetro da iteração anterior, assim tem-se ainda $tol = 10^{-10}$. Foram gerados os seguintes parâmetros:

TABELA 5.3 – PARÂMETROS ADI SIMULAÇÃO 2

$\mathbf{p_1}$	-0.2629 + 0.4592i
$\mathbf{p_2}$	-0.2629 - 0.4592i
$\mathbf{p_3}$	-0.3651 + 0.3830i
$\mathbf{p_4}$	-0.3651 - 0.3830i
$\mathbf{p_5}$	-0.4004 + 0.3459i
$\mathbf{p_6}$	-0.4004 - 0.3459i
$\mathbf{p_7}$	-0.4400 + 0.2939i
$\mathbf{p_8}$	-0.4400 - 0.2939i

Aqui, apesar do número de iterações *J = 7*, o número de parâmetros gerados foram 8, como o previsto, pois o algoritmo deve gerar pares complexos conjugados. Os autovalores desta matriz são:

TABELA 5.4 – AUTOVALORES DA MATRIZ *A*

$\boldsymbol{\lambda_1}$	-1+3i
$\boldsymbol{\lambda_2}$	-1-3i
$\boldsymbol{\lambda_3}$	-1
$\boldsymbol{\lambda_4}$	-0,2
$\boldsymbol{\lambda_5}$	-1
$\boldsymbol{\lambda_6}$	-1,4
$\boldsymbol{\lambda_7}$	-0,6
$\boldsymbol{\lambda_8}$	-0,7
$\boldsymbol{\lambda_9}$	-1,2
$\boldsymbol{\lambda_{10}}$	-1

Os parâmetros continuam dentro do intervalo de abrangência dos autovalores de *A*. Percebe-se que apesar de ter apenas 2 autovalores complexos, todos os parâmetros gerados foram complexos. Isto corrobora com o método, que procura aproximar a região formada pelos autovalores por uma região $\Omega \subset \sigma(A)$ ao invés de $\Omega \subset \mathbb{C}_-$.

Utilizando estes parâmetros para a redução do sistema através das sub-rotinas do LYAPACK, obteve-se os seguintes resultados:

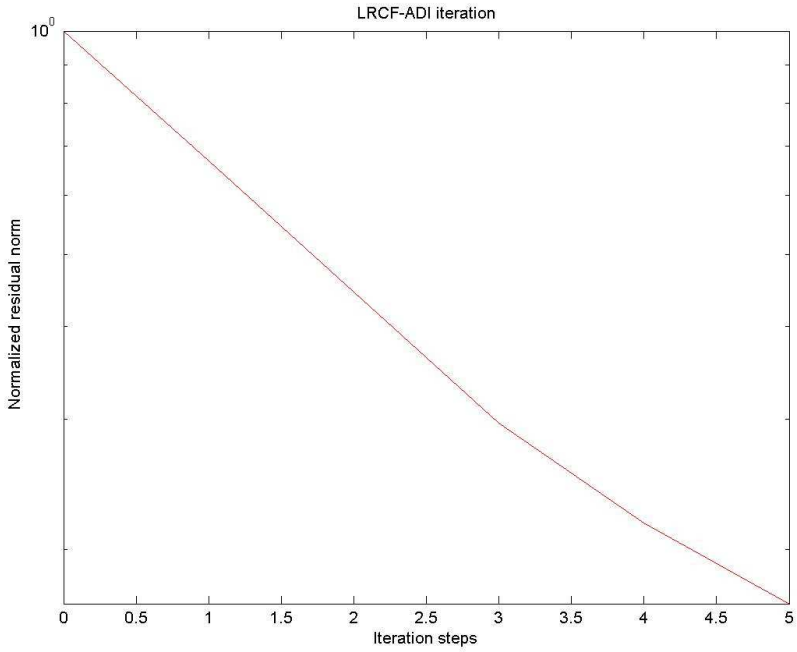

FIGURA 5.4 – - RESÍDUO BB^T DA SIMULAÇÃO 2

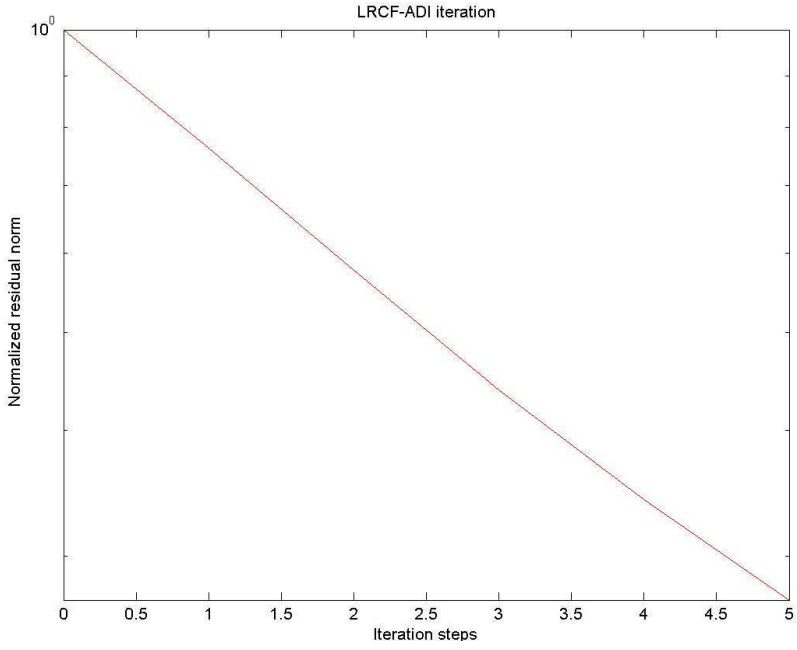

FIGURA 5.5 – - RESÍDUO C^TC DA SIMULAÇÃO 2

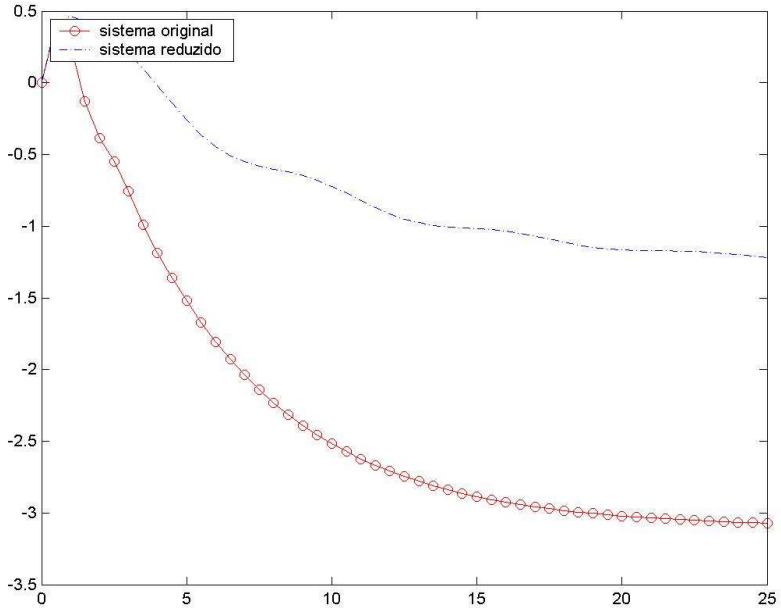

FIGURA 5.6 – COMPARAÇÃO ENTRE SISTEMAS

Na figura 5.6 vemos que para os parâmetros complexos, a aproximação se tornou pobre. Aumenta-se o valor de *max_it* para 6, e com isso o sistema reduzido diverge. Aumentou-se o número de parâmetros fazendo-se $\varepsilon_{par} = 10^{-15}$, obteve-se assim uma melhor aproximação nos primeiros instantes da resposta ao degrau e logo após o sistema ficou instável, como indicado na seguinte figura:

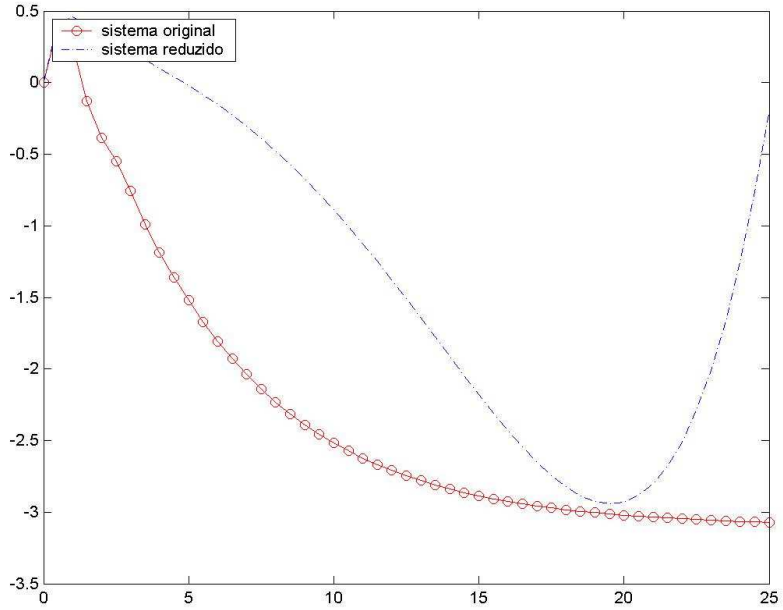

FIGURA 5.7 – COMPARAÇÃO ENTRE SISTEMAS, *TOL* = 10^{-15}

Caso o número de iterações for acrescido de 1, com *max_it = 7*, o resultado volta a divergir, enquanto a ordem do sistema reduzido permanece em 4. Para aumentar a ordem do sistema reduzido, faz-se $tol = 10^{-20}$, mas o resultado continua a divergir.

Aumentou-se a precisão na escolha dos parâmetros, mas com o parâmetro de parada *max_it = 150*, para que o critério de parada fosse dado somente por ε_{red}. Também ajustou-se *max_ord = 100*, para não fosse suprimida a ordem da aproximação. Os resíduos BB^T e C^TC decaíram até valores muito baixos, mas a iteração atingiu seu valor máximo sem que o critério de parada fosse, além da ordem do sistema ter atingido o valor máximo, o que é um contra-senso já que o objetivo é reduzir a ordem do sistema. A comparação dos dois sistemas é feita pela figura abaixo:

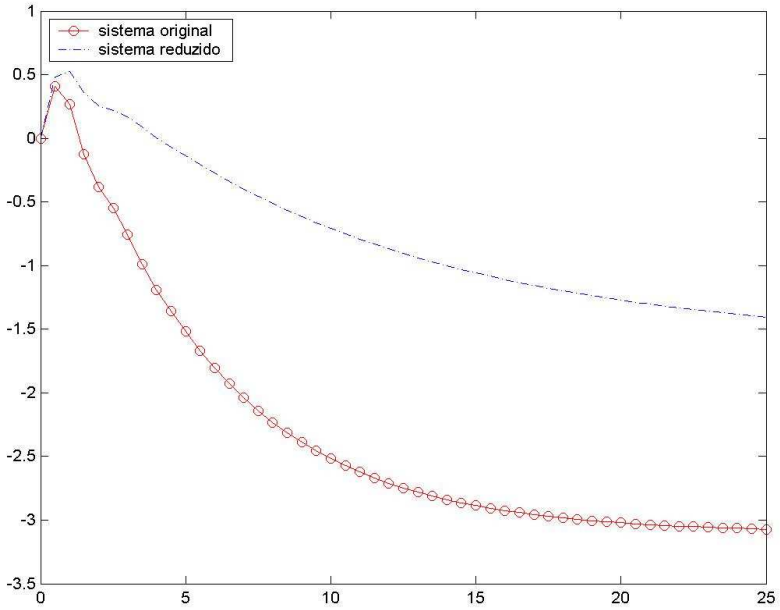

FIGURA 5.8 – COMPARAÇÃO ENTRE SISTEMAS

Simulação 3

Agora simula-se as mesmas matrizes de pequeno porte com outra rotina de cálculo de parâmetros, a rotina *sub_adi*, descrita no apêndice B e proposta por Benner, Mena e Saak em [9]. Com $\varepsilon_{par} = 10^{-10}$ e a sub-rotina de iteração de Arnoldi devidamente ajustada, obteve-se os seguintes parâmetros:

TABELA 5.5 – PARÂMETROS ADI SIMULAÇÃO 2

$\mathbf{p_1}$	-4,1407
$\mathbf{p_2}$	-3,6996
$\mathbf{p_3}$	-2,9921
$\mathbf{p_4}$	-2,4357
$\mathbf{p_5}$	-2,5559

Comparado com o método anterior, são dois parâmetros a menos e com valores bastante diferentes, concentrados em torno do maior autovalor. No algoritmo para redução do sistema, fez-se *max_it = 5* e *tol =* 10^{-10}, obteve-se os resultados que se seguem:

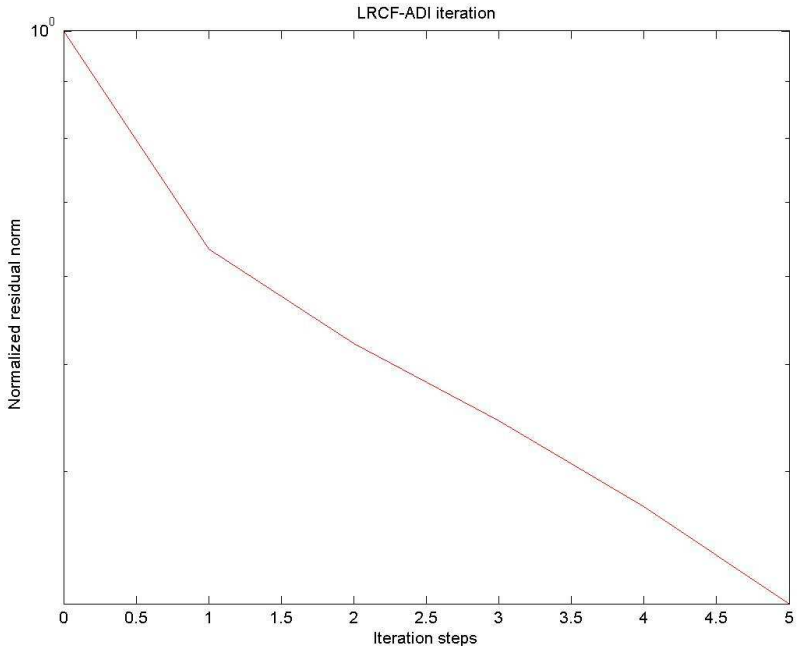

FIGURA 5.9 – - RESÍDUO BB^T DA SIMULAÇÃO 3

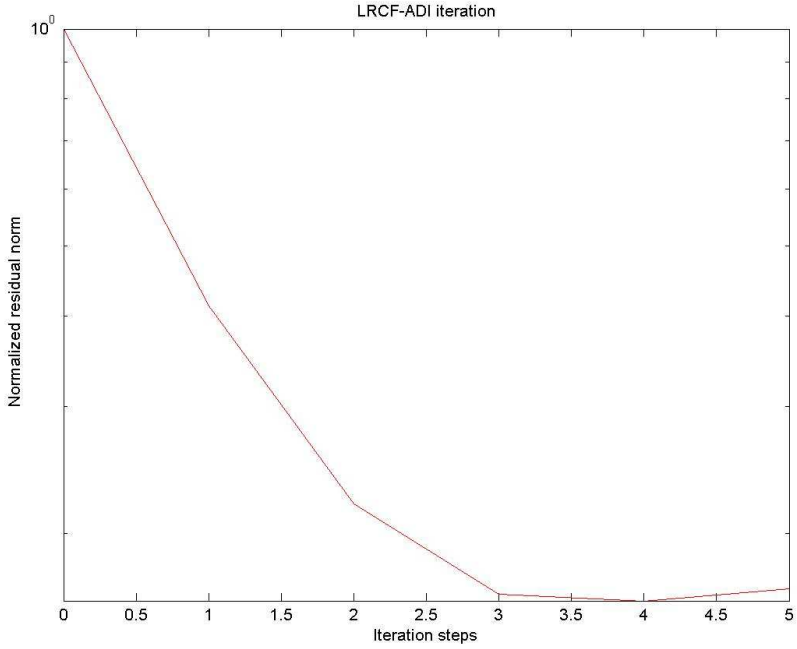

FIGURA 5.10 – - RESÍDUO C^TC DA SIMULAÇÃO 3

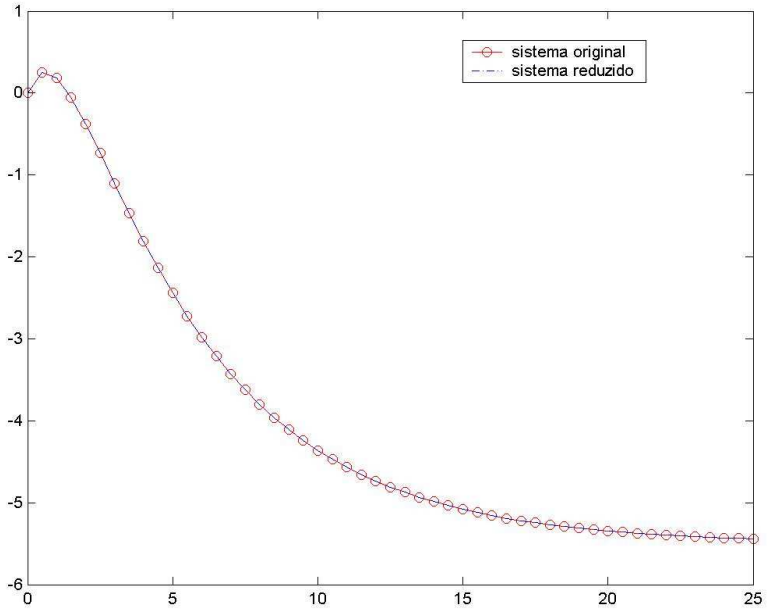

FIGURA 5.11 – COMPARAÇÃO ENTRE SISTEMAS, SIMULAÇÃO 3

Aqui temos a mesma aproximação idêntica ao sistema original e para um $tol = 10^{-10}$ a ordem do sistema reduzido ficou também em $n_{red} = 4$.

Simulação 4

Para a matriz complexa, mantendo os mesmo parâmetros, tem-se:

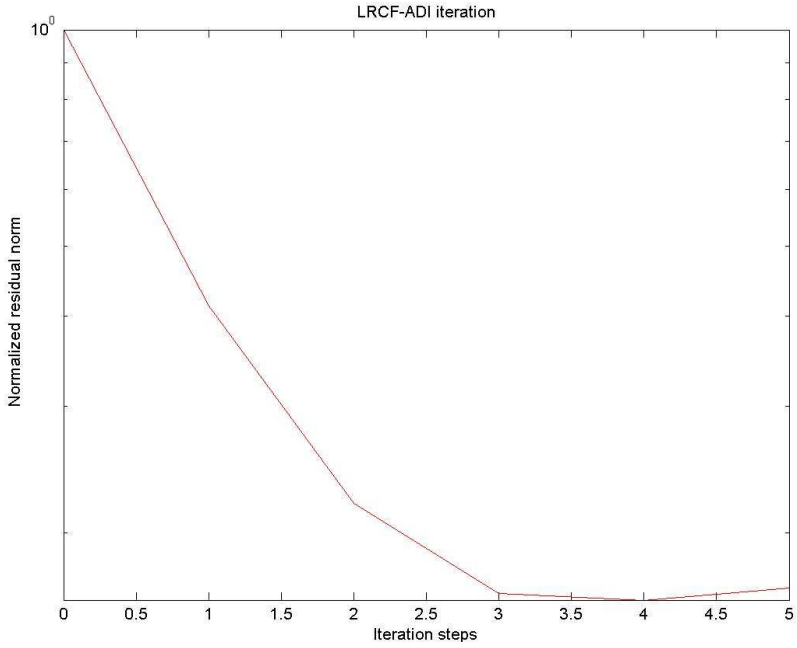

FIGURA 5.12 – RESÍDUO BB^T DA SIMULAÇÃO 4

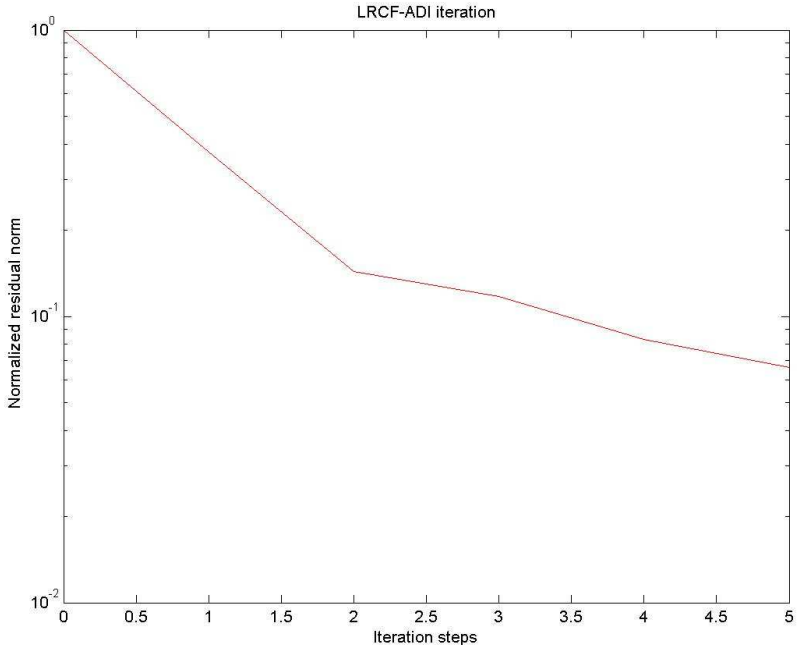

FIGURA 5.13 – RESÍDUO C^TC DA SIMULAÇÃO 4

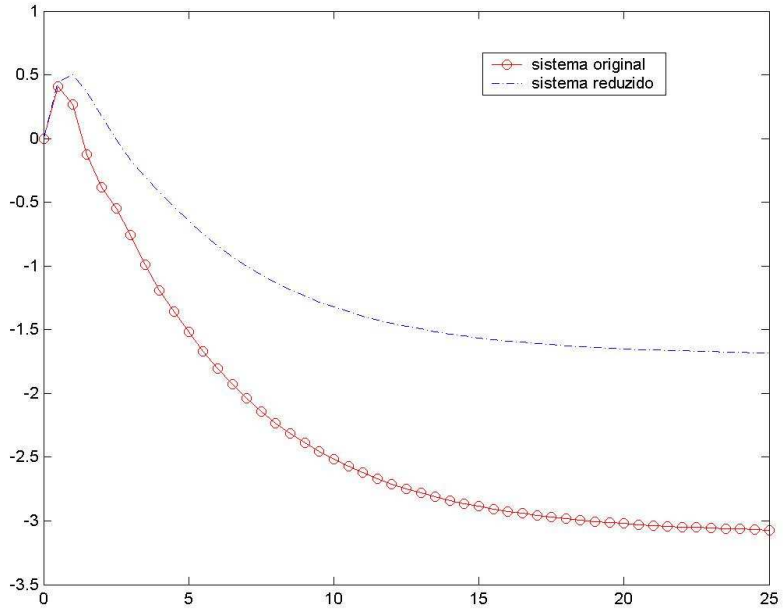

FIGURA 5.14 – COMPARAÇÃO ENTRE SISTEMAS, SIMULAÇÃO 4

Apesar de ainda ser uma aproximação pobre, percebe-se que a rotina *sub_adi* gerou parâmetros que melhor aproximou o sistema com autovalores complexos, com ordem do sistema reduzido permanecendo a mesma, $n_{red} = 4$. Fazendo-se *max_it = 6*, o resultado continuou a convergir. A fim de se tentar uma melhor aproximação fez-se $\varepsilon_{par} = 10^{-15}$, mas o sistema reduzido foi o mesmo. Assim, a rotina *sub_adi* gerou parâmetros que convergiram em todos os casos, apesar do sistema reduzido não acompanhar o sistema original com exatidão.

Simulação 5

Esta simulação foi feita com um sistema de grande porte, com *n = 606*, gerada a partir de dados do sistema interligado brasileiro. A rotina *rli* foi ajustada para uma precisão de $\varepsilon_{par} = 10^{-10}$ e na rotina de redução de sistema assumiu-se *max_it = 40* e $tol = 10^{-10}$. Com este valor de ε_{par} a rotina *rli* gerou 12 parâmetros ADI, mas o cálculo do sistema reduzido não convergiu. Alterou-se a precisão do parâmetro para $\varepsilon_{par} = 10^{-15}$ e o fez-se $tol = 10^{-15}$ e o resultado continuou a divergir.

Somente quando se fez *tol* = 10^{-20}, para se obter uma ordem reduzida maior, o que resultou em n_{red} = *60*. Também aumentou-se a precisão na seleção dos parâmetros, fazendo-se $\varepsilon_{par} = 10^{-30}$, a fim de se gerar mais parâmetros, o que resultou em 34 parâmetros. Esta simulação não atingiu o valor da norma esperada e as iterações só pararam quando atingiram *max_it*. Os resultados são mostrados abaixo:

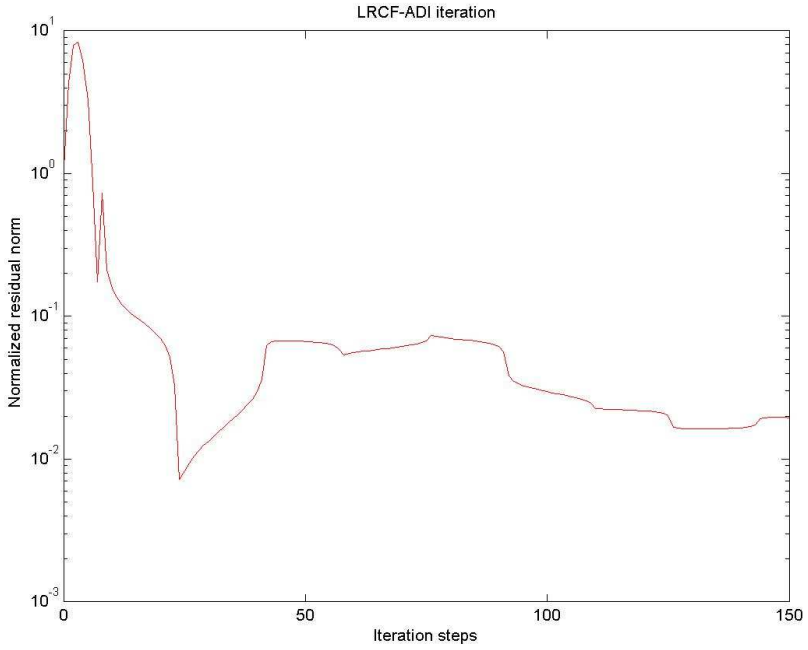

FIGURA 5.15 – RESÍDUO BB^T DA SIMULAÇÃO 5

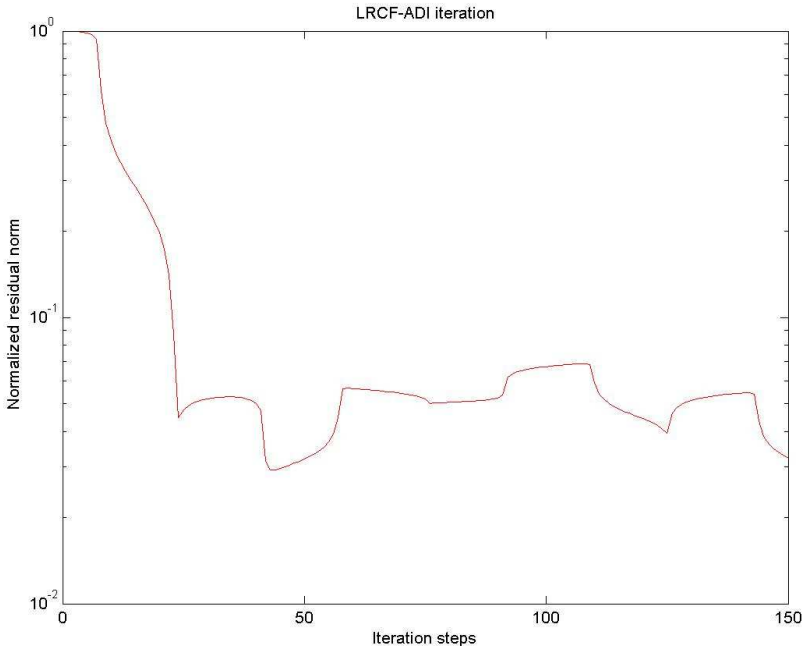

FIGURA 5.16 – RESÍDUO C^TC DA SIMULAÇÃO 5

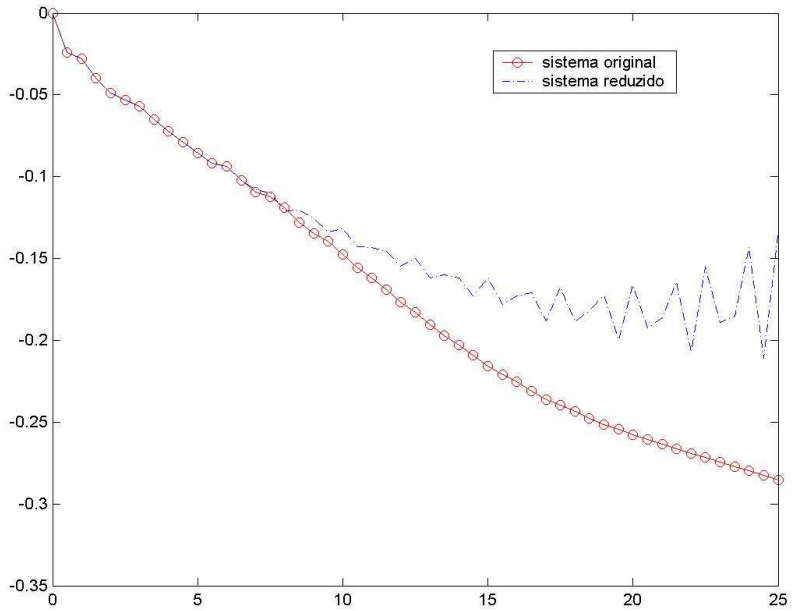

FIGURA 5.17 – COMPARAÇÃO ENTRE SISTEMAS, SIMULAÇÃO 5

Aumentando o tempo de análise do gráfico, obtem-se:

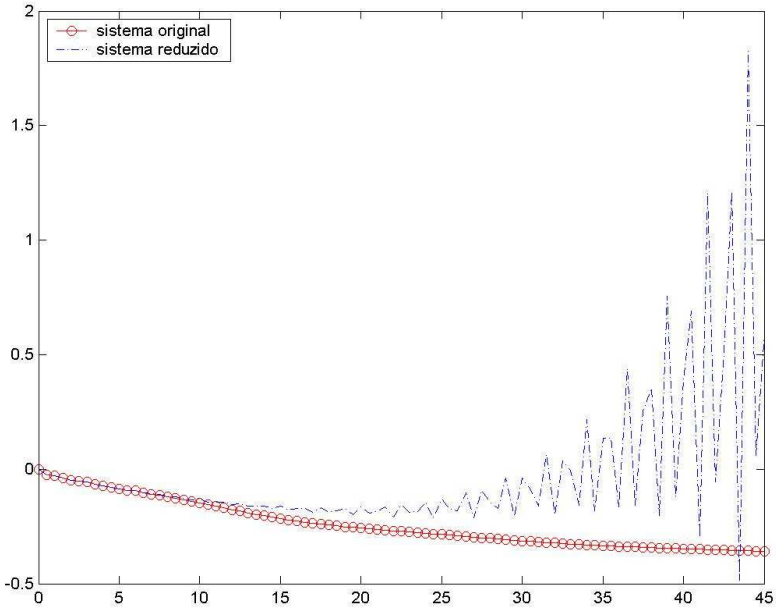

FIGURA 5.18 – COMPARAÇÃO ENTRE SISTEMAS, TEMPO EXTENDIDO SIMULAÇÃO 5

Percebe-se aqui que o sistema reduzido volta a divergir depois de certo tempo, e também que diverge com uma oscilação de freqüência basicamente constante. Isto deve-se ao fato de que quando a parte real dos autovalores da matriz do sistema *A* tem módulo consideravelmente maior que a parte imaginária, o método de parâmetros ótimos, implementado na rotina *rli*, tende a aproximar os parâmetros para o eixo real somente. Assim os autovalores imaginários são pobremente aproximados, o que gera o comportamento divergente e oscilatório de figura 5.14.

Simulação 6

Esta simulação foi feita com a rotina *sub_adi*, com $\varepsilon_{par} = 10^{-10}$, *max_it = 40* e *tol = 10*$^{-10}$, e a iteração de Arnoldi devidamente ajustada. A referida rotina gerou somente 5 parâmetros, que aproximaram pobremente o sistema original, como se pode perceber nas figuras a seguir:

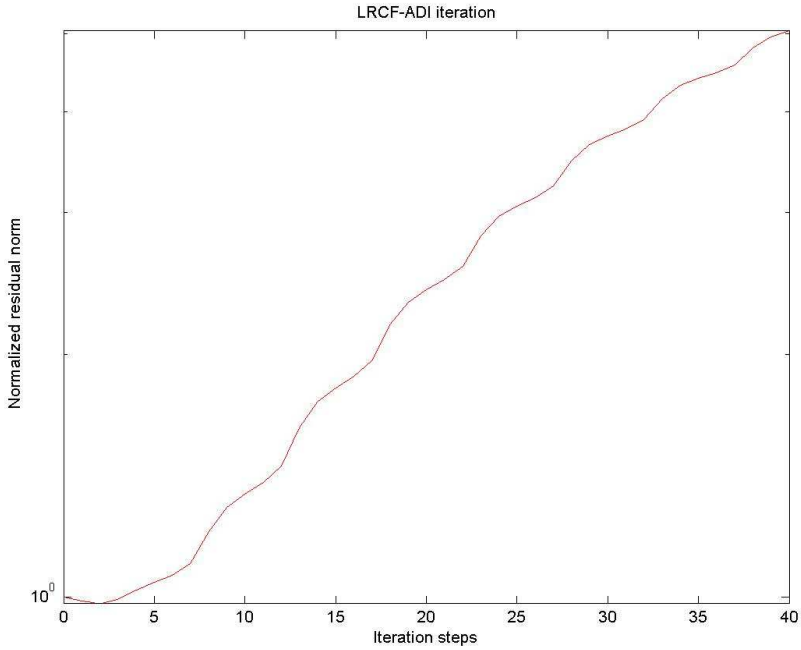

FIGURA 5.19 – RESÍDUO BB^T DA SIMULAÇÃO 6

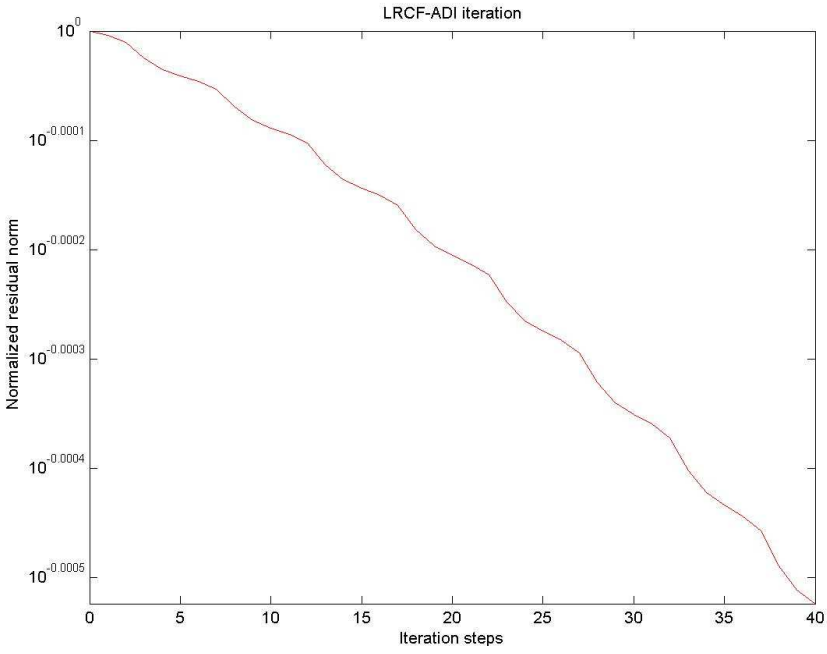

FIGURA 5.20 – RESÍDUO C^TC DA SIMULAÇÃO 6

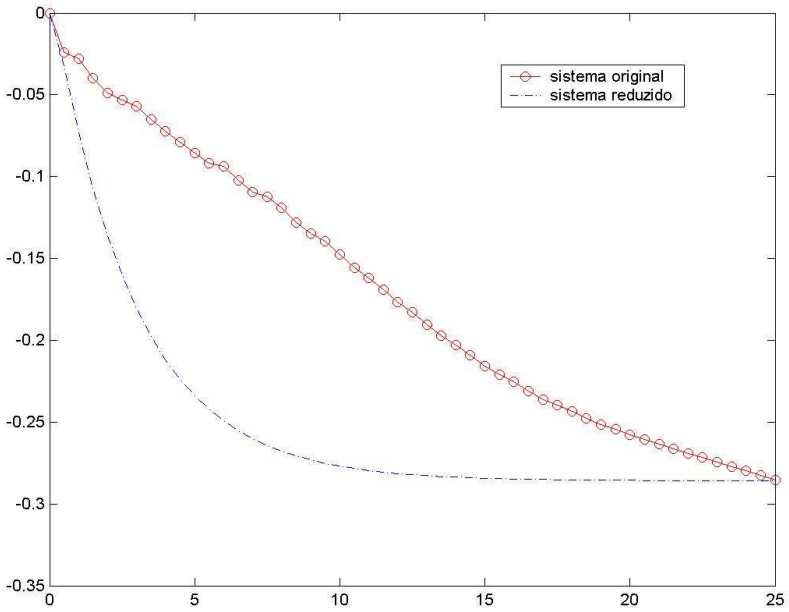

FIGURA 5.21 – COMPARAÇÃO ENTRE SISTEMAS, SIMULAÇÃO 6

O resíduo por $-BB^T$ não convergiu enquanto o de $-C^TC$ praticamente não se alterou. O sistema reduzido não divergiu, entretanto não acompanhou o sistema original. Pode-se justificar que isto aconteceu devido à pequena quantidade de parâmetros ADI calculadas. Para averiguação, faz-se a precisão na da rotina *sub_adi* igual a $\varepsilon_{par} = 10^{-30}$ e $tol = 10^{-20}$ com isto, consegue-se 15 parâmetros e um sistema reduzido de ordem $n_{red} = 40$, porém o resultado final praticamente não muda:

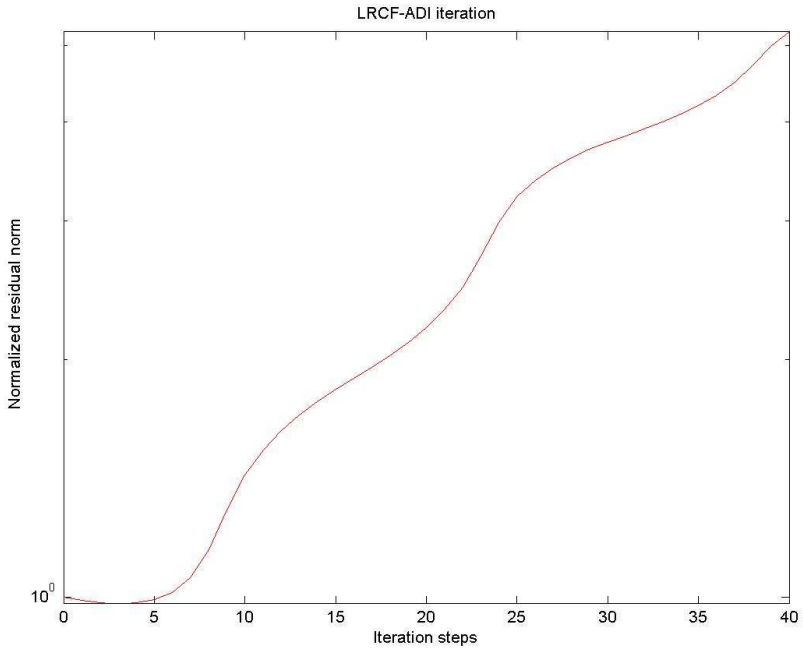

FIGURA 5.22 – – RESÍDUO BB^T DA SIMULAÇÃO 6

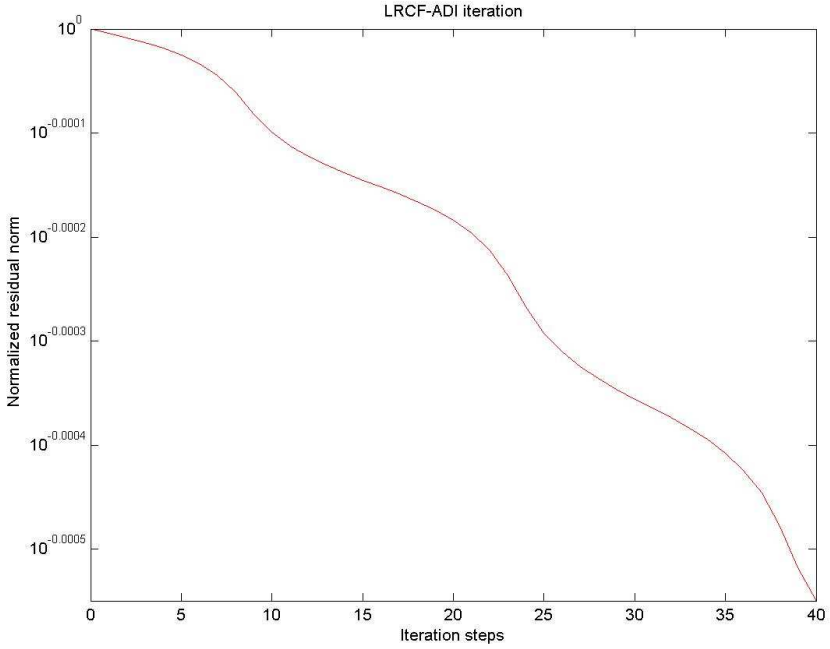

FIGURA 5.23 – RESÍDUO C^TC DA SIMULAÇÃO 6

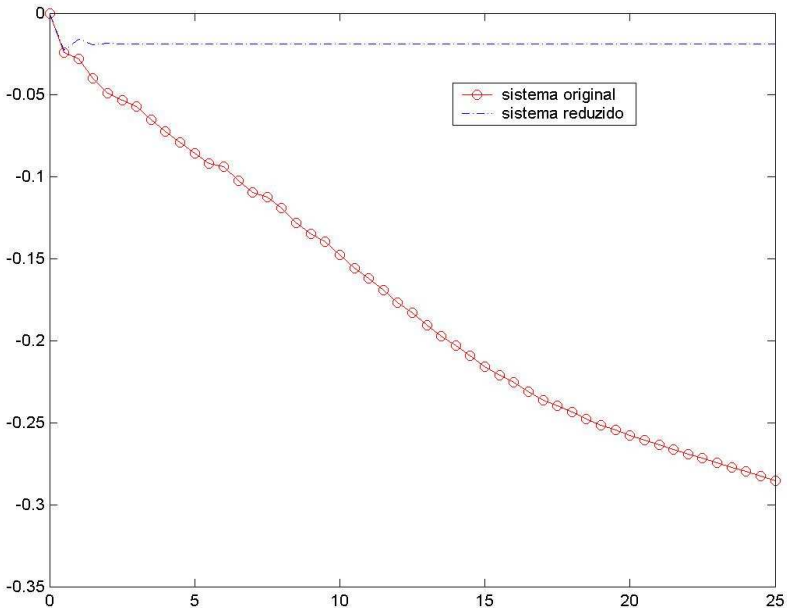

FIGURA 5.24 – COMPARAÇÃO ENTRE SISTEMAS, PARÂMETROS MODIFICADOS SIMULAÇÃO 6

Percebe-se que a parte inicial do gráfico é bem aproximada, e após isto o sistema reduzido se afasta do sistema original. Os instantes iniciais da resposta ao degrau estão relacionados aos pólos mais rápidos, ou seja, aqueles que estão mais afastados da origem. A título de comparação, os intervalos dos valores dos parâmetros ADI e autovalores são:

$$\begin{aligned} &\text{-}2.1363\cdot 10^{4} \le p_i \le \text{-}5.3749\cdot 10^{3} \\ &\text{-}1.0727\cdot 10^{4} \le \lambda_i \le 0.0029 \end{aligned} \tag{5.1}$$

Os parâmetros estão no extremo esquerdo do espectro. Então a referida rotina está aproximando somente os autovalores mais rápidos, por isto ela está acompanhando somente a parte inicial do sistema original. Aqui fica evidente a importância do cálculo do espectro de λ para os autovalores mais próximos à origem, como foi proposto em [4].

Já os parâmetros advindos da rotina *rli* não estão nem em um extremo nem no outro, mas estão mais próximos da origem. Talvez por este motivo a redução calculada (figura 5.17) pelos seus parâmetros se aproximam bem do sistema original, até certo ponto, quando começa a divergir oscilando, já que o comportamento do sistema começa assim a se aproximar dos pólos mais lentos, e entre eles existe um autovalor instável, como pode-se perceber em (5.1). Como o autovalor que está no semi-plano direito é $\lambda_m = 0,0029$, fez-se

um salto de 0,003 no sistema linear, de modo a deixá-lo totalmente no semi-plano esquerdo. Obteve-se o seguinte resultado:

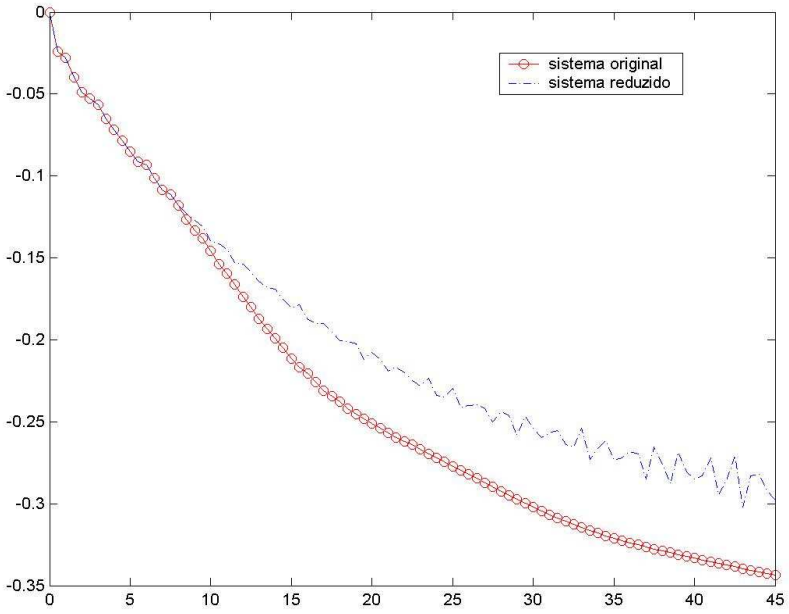

FIGURA 5.25 – COMPARAÇÃO ENTRE SISTEMAS, COM SALTO DE 0,003

Aqui tem-se que as oscilações estão mais contidas, como previsto nesta análise.

6 – CONCLUSÃO

Este trabalho apresentou técnicas para cálculo de parâmetros ADI os quais são utilizados na solução da equação de Lyapunov. No capítulo 2, fez-se uma apresentação da equação de Lyapunov, mostrando as suas principais aplicações atuais e alguns métodos já consagrados de solução da referida equação. Percebe-se que a equação de Lyapunov possui várias aplicações na solução de diversos problemas que extrapolam o âmbito da engenharia elétrica e matemática. Os métodos de solução vistos se relacionam e servem de subsídio para a análise do método de iteração ADI.

O objeto principal do trabalho foi enfocado no capítulo 3, onde se encontra um breve histórico do método, descrevendo o seu surgimento e posterior desenvolvimento, adaptando-se a diferentes casos e recebendo diferentes contribuições, como por exemplo o método LR-ADI e LR-Smith, desenvolvidos para resolver equações com sistemas lineares de grande porte e esparsos. O método CF-ADI descrito no capítulo procura otimizar o cálculo através da economia de custo computacional através dos fatores de Cholesky e sua propriedade de recursividade.

Descreveu-se três métodos de cálculo de parâmetros, o método dos parâmetros ótimos, sub-ótimos e um procedimento heurístico modificado. Este último é derivado dos dois primeiros. O primeiro e o último método foram implementados através do software Matlab® a fim de que fossem testados, utilizando-se uma rotina de redução de sistemas que utiliza sub-rotinas do LYAPACK.

A partir das simulações, foi possível se chegar a respostas da convergência dos modelos reduzidos de acordo com cada parâmetro e a fidelidade destes sistemas de ordem reduzida aos sistemas originais. A redução da ordem de sistemas lineares é uma área que sempre teve importância nos vários ramos das ciências exatas, porque sempre existirá preocupação com a eficiência dos cálculos numéricos computacionais, e a transposição de novos desafios.

Os testes realizados contribuíram para a tomada de decisão no cálculo de parâmetros ADI, pois evidenciou que a escolha dos parâmetros deve ser específica para cálculo do sistema em estudo.

Futuros trabalhos podem ser desenvolvidos no sentido a aprimorar os algoritmos de cálculo de parâmetros, pois como foi visto na seção anterior, quando se tem alguns autovalores complexos no espectro da matriz *A*, os parâmetros resultantes do cálculo são todos complexos, mas deveriam existir parâmetros reais. Além disso, pode-se buscar desenvolver outras técnicas que melhorem a solução do problema, uma vez que aqui foram testadas apenas duas técnicas.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] M.G. Safonov and R.Y. Chiang. *A schur method for balanced-truncation model reduction*. IEEE Trans. Automat .Control, 34:729-733, 1989.

[2] M. Januszewski, J. Machowski and J.W. Bialek. *Application of the direct Lyapunov method to improve damping of power swings by control of UPFC*. IEE Proceedings online no. 20040054, *http://ieeexplore.ieee.org*

[3] D. C. Sorensen, Y. Zhou. *Direct Methods for Matrix Sylvester and Lyapunov Equations*. Dept. of Computacional and Applied Mathematics, Rice University, 2003.

[4] Thilo Penzl., *A Cyclic Low Rank Smith Method for Large Sparse Lyapunov Equations with Applications in Model Reduction and Optimal Control*. Technische Universität Chemnitz, 1998.

[5] E.L. Wachspress. *The ADI minimax problem for complex spectra*. In D.Kincaidan L. Hayes, editors, *Iterative Methods for large Linear Systems*, p. 251-271. Academic Press, San Diego, 1990.

[6] K.-G. Steffens *The History of Approximation Theory: From Euler to Bernstein* Birkhauser, Boston 2006.

[7] W.E. Arnoldi. *The principle of minimized iterations in the solution of the matrix eigenvalue problem*. Quart. Appl. Math., 9:17-29, 1951.

[8] D.C. Sorensen. *Implicit application of polynomial filters in a k-step Arnoldi method*. SIAM J. Matrix Anal. Appl., 13:357-385, 1992.

[9] Banner, P., Mena, H. & Saak, J. *On the Parameter Selection Problem in the Newton-ADI Iteration for Large Scale Riccati Equations*. Chemnitz Scientific Computing Preprints, CSC/06-03, *http://archiv.tu-chemnitz.de/pub/2007/0193/index*

[10] D. Peaceman and H. Rachford, *The Numerical Solution of Elliptic and Parabolic Differential Equations*, Journal of SIAM., Vol. 3, pp. 28-41, 1955.

[11] Li, J.-Rebecca, White, J., *Low-Rank Solution of Lyapunov Equations*, SIAM Review, Vol 46, No 4, p 693-713, 2004.

[12] NETO, O. C. & MONTEIRO, P. A. G. (2006). *Técnicas de Redução de Sistemas Dinâmicos por meio da Solução de Equações de Lyapunov de Baixo Rank.*

Monografia de Graduação, Publicação ENE 01/2006, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 89p.

[13] Li, J.-Rebecca, *Model Reduction of Large Linear Systems via Low Rank Systems Gramians*, B. Sc., Mathematics University of Michigan, 1995.

[14] Athay, T. M., *Numerical Analisys of the Lyapunov Equation with Application to Interconnected Power Systems*, SB, Michigan Technological University, 1974.

[15] D.L. Kleinman. *On an iterative technique for Riccati equation computations*. IEEE Trans. Automat. Control, 13:114-115,1968.

[16] Saad, Y. *Numerical Solution of Large Lyapunov Equations.* 1990. *http://citeseer.ist.psu.edu/379497*

[17] Penzl, Thilo, *LYAPACK, a MATLAB Toolbox for large Lyapunov and Riccati Equation, Model Reduction Problems, and Linear-Quadratic Optimal Control Problems, Users' Guide*. 1999. http://www.tu-chemnitz.de/sfb393/lyapack/guide.pdf

APÊNDICE A (PARÂMETROS ÓTIMOS)

```
% Este programa trabalha no cálculo dos parâmetros ADI através do método parâmetros
      ótimos

%    Definiçao das fronteiras ----------------------------------------------------------------------
a=min(real(-eig(A)));
b=max(real(-eig(A)));
alfalinha=abs(imag(-eig(A))./real(-eig(A)));
alfalinha2=max(alfalinha);
alfa=atan(alfalinha2);
%-------------------------------------------------------------------------------------------------------
betaq=2/(1+.5*(a/b+b/a)); % valor de (cos(beta))^2          %cálculo de cos^2(β)
m=((2*(cos(alfa))^2)/(betaq)-1);       % define o valor m
epson=1e-10;            %define a precisao
if m >= 1        % os parametros sao reais
   disp('parametros reais')
   klinha=(m+sqrt(m^2-1))^-1;
   k=sqrt(1-klinha^2);
   K=ellipke(k^2);        %integral elíptica
   vlinha=asin((a/(b*klinha))^.5); %
   [SN,CN,DN]=ellipj(vlinha,klinha^2); %
   J=ceil(((K/(2*DN*pi))*log10(4/epson)));          % determina o numero de iteraçoes
   %iteraçao para achar os parametros reais
   for ka=1:J
      u=(2*ka-1)*K/(2*J);
      dn=sqrt(1-(k^2)*(sin(u))^2);
      pj(ka)=-sqrt(a*b/klinha)*dn;
   end
   pp=pj;
   disp(' numero de iteraçoes')
```

```
    J
    disp('  Parametros ADI')
    pj
    disp(' Tolerancia')
    Epson
else        % os parâmetros são complexos
    disp('parametros complexos')
    mlinha=(2*(betaq))/((cos(alfa))^2)-1;
    klinhac=(mlinha+sqrt(mlinha^2-1))^-1;
    kc=sqrt(1-klinhac^2);
    Kc=ellipke(kc^2);
    ac=tan(pi/4-alfa/2);
    bc=1/ac;
    vlinhac=asin((ac/(bc*klinhac))^.5);
    [SN,CN,DN]=ellipj(vlinhac,klinhac^2);              %integral elíptica jacobiana
    J=ceil(((Kc/(2*DN*pi))*log10(4/epson)));
    %iteraçao para achar os parametros complexos
    cte=sqrt(a*b);
    for kb=1:fix((1+J)/2)
       uc=(2*kb-1)*Kc/(2*J);
       dn=sqrt(1-(kc^2)*(sin(uc))^2);
       pj(kb)=-sqrt(ac*bc/klinhac)*dn;
       alfac(kb)=acos(2/(pj(kb)+1/pj(kb)));
       ni=2*(kb-1)+1;
       np=2*kb;
       pc(ni)=cte*exp(i*alfac(kb));
       pc(np)=conj(pc(ni));
    end
    pp=pc;
    disp(' numero de iteraçoes')
    J
```

```
    disp('  Parametros ADI')
    pc
    disp(' Tolerancia')
    epson
end
```

APÊNDICE B (PROCEDIMENTO HEURÍSTICO MODIFIACADO)

```
epson=1e-30;              define a precisao
if eig(A)=='R'             parametros reais
   disp('autovalores reais')
   %    Definiçao das fronteiras
   a=min(-eig(A));  valor de a
   b=max(-eig(A));  valor de b
   alfalinha=abs(imag(-eig(A))./real(-eig(A)));
   alfalinha2=max(alfalinha);
   alfa=atan(alfalinha2);
   k1=a/b;
   k=sqrt(1-k1^2);
   K=ellipke(k^2);
   v=ellipke(k1^2);
   J=ceil(((K/(2*DN*pi))*log10(4/epson))); %
   %iteraçao para achar os parametros reais
   for ka=1:J
     u=(2*ka-1)*K/(2*J);
     dn=sqrt(1-(k^2)*(sin(u))^2);
     pj(ka)=-sqrt(a*b/klinha)*dn;
   end
   pp=pj;
   disp(' numero de iteraçoes')
   J
   disp('  Parametros ADI')
   pj
   disp(' Tolerancia')
   epson
else     %    %iteraçao para achar os parametros complexos
```

```
disp('autovalores complexos')
ac=min(real(-eig(A)));
bc=max(real(-eig(A)));
c=(ac+bc)/2;
 % iteraçao Arnoldi
   disp('iteraçao Arnoldi, l =100 )e m=130
   F=-A+c*eye(n);
   m=130;
   kmax=100;
   %n=size(A,1);
   v1=rand(n,1);
   nv=norm(v1);
   vj=v1/nv;
   V=zeros(n,m+1);
   V(:,1)=vj;
   h=zeros(m+1,m);
   for j=1:m
      vj=V(:,j);
      w=F*vj;
      for k=1:j
         vk=V(:,k);
         h(k,j)=conj(vk')*w;
         w=w-h(k,j)*vk;
      end
      h(j+1,j)=norm(w);
      V(:,j+1)=w/h(j+1,j);
   end
   Hm=h(1:m,1:m);
   lamb=eig(Hm);
   lambda_kmax=lamb(1:kmax);
% fim da iteraçao Arnoldi
```

```
lambda_t=lambda_kmax+c;
%    Definiçao das fronteiras
at=min(real(lambda_t)); % retorna o valor de a modificado
bt=max(real(lambda_t));
alfalinhat=abs(imag(lambda_t)./real(lambda_t));
alfalinha2t=max(alfalinhat);
alfat=atan(alfalinha2t);
betaqt=2/(1+.5*(at/bt+bt/at));              % valor de cos^2(beta) modificado
mt=((2*(cos(alfat))^2)/(betaqt)-1)          ;% define o valor m modificado
if mt >= 1
   disp('m >= 1')
   klinhat=(mt+sqrt(mt^2-1))^-1;
   kt=sqrt(1-klinhat^2);
   Kt=ellipke(real(kt^2));
   vlinhat=asin((at/(bt*klinhat))^.5);
   [SN,CN,DN]=ellipj(real(vlinhat),real(klinhat^2));
   J=ceil(((Kt/(2*DN*pi))*log10(4/epson)));
   %iteraçao para achar os parametros reais
   for ka=1:J
     ut=(2*ka-1)*Kt/(2*J);
     dnt=sqrt(1-(kt^2)*(sin(ut))^2);
     pjt(ka)=-sqrt(at*bt/klinhat)*dnt;
   end
   pp=pjt;
   disp(' numero de iteraçoes')
   J
   disp('  Parametros ADI')
   pjt
   disp(' Tolerancia')
   epson
else          % m < 1
```

```
      disp('m < 1')
      mlinhat=(2*(betaqt)/(cos(alfat))^2)-1;
      klinhact=(mlinhat+sqrt(mlinhat^2-1))^-1;
      kct=sqrt(1-klinhact^2); %define o valor k
      Kct=ellipke(kct^2); %encontra o valor de K
      act=tan(pi/4-alfat/2);
      bct=1/act;
      vlinhact=asin((act/(bct*klinhact))^.5);
      [SN,CN,DN]=ellipj(vlinhact,klinhact^2);
      J=ceil(((Kct/(2*DN*pi))*log10(4/epson))); %
      %iteraçao para achar os parametros reais
      cte=sqrt(at*bt);
      for kb=1:fix((1+J)/2)
        uct=(2*kb-1)*Kct/(2*J);
        dnt=sqrt(1-(kct^2)*(sin(uct))^2);
        pjt(kb)=-sqrt(act*bct/klinhact)*dnt;
        alfact(kb)=acos(2/(pjt(kb)+1/pjt(kb)));
        ni=2*(kb-1)+1;
        np=2*kb;
        pct(ni)=cte*exp(i*alfact(kb));
        pct(np)=conj(pct(ni));
      end
      pp=pct;
      disp(' numero de iteraçoes')
      J
      disp('  Parametros ADI')
      pct
      disp(' Tolerancia')
      epson
    end
end
```