

DRAFT

Technical Report CEPEL/DSE – 167/2004

OVERVIEW OF NUMERICAL ALGORITHMS FOR SMALL SIGNAL STABILITY ANALYSIS AND CONTROL DESIGN

Nelson Martins

CEPEL, Caixa Postal 68007, 21944-970, Rio de Janeiro, RJ, Brazil
Fax: +55-21-260-1245 e-mail: nelson@cepel.br

I. INTRODUCTION

This report describes some of the algorithms currently used in the small signal stability analysis and controller design of power system models. Focus are on numerically robust algorithms which are able to deal with large scale systems and exploit sparsity. The described algorithms allow the computation of eigenvalues and eigenvectors, transfer function zeros and residues, eigenvalues sensitivities, time and frequency response plots. Several partial eigensolution algorithms are described, including those for computing only the dominant pole spectrum of SISO and MIMO transfer functions. Reduced order models can therefore be readily computed. This appendix is restricted to the description of the numerical algorithms. Use of double-precision arithmetic is assumed essential to these computations. The practical use of these algorithms in the stability analysis and control design methods is described along the body of the technical report. Important modal informations like mode shapes and participation factors are assumed to be known and not described, for being simple applications of right and left eigenvectors.

In addition to numerical linear algebra [18,9], linear system [1] and feedback control [2,3] are the key to power system small-signal stability analysis and control [4].

A modern package for the small-signal stability analysis of power systems should have a good part of the described algorithms [5,6,7]. A production grade software must have ease of data input, flexible user defined controller models capable of sensing any combination of local and remote system variables and good program output. Program efficiency may be slightly sacrificed in favor of the above facilities.

Most of these methods have been implemented in PacDyn, CEPEL's comprehensive software for small-signal stability analysis and control. You may obtain more information on these algorithms and their use in power system studies by visiting PacDyn's homepage: www.pacdyn.cepel.br.

II. SYSTEM MODELING

The power system electromechanical stability problem can be represented by a set of differential equations together with a set of algebraic equations, to be solved simultaneously with each other [7]:

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{r}) \\ \mathbf{0} &= g(\mathbf{x}, \mathbf{r}) \end{aligned} \quad (1)$$

where \mathbf{x} is the state vector and \mathbf{r} is a vector of algebraic variables.

Small-signal stability analysis involves the linearization of (1) around a system operating point $(\mathbf{x}_0, \mathbf{r}_0)$

$$\begin{pmatrix} \Delta \dot{\mathbf{x}} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{J}_1 & \mathbf{J}_2 \\ \mathbf{J}_3 & \mathbf{J}_4 \end{pmatrix} \begin{pmatrix} \Delta \mathbf{x} \\ \Delta \mathbf{r} \end{pmatrix} \quad (2)$$

The power system state matrix can be obtained by eliminating the vector of algebraic variables $\Delta \mathbf{r}$ in equation (2):

$$\Delta \dot{\mathbf{x}} = \left(\mathbf{J}_1 - \mathbf{J}_2 \mathbf{J}_4^{-1} \mathbf{J}_3 \right) \Delta \mathbf{x} = \mathbf{A} \Delta \mathbf{x} \quad (3)$$

The symbol \mathbf{A} is used to represent the system state matrix, whose eigenvalues provide information on the singular point stability of the non-linear system.

The symbol Δ signifies an incremental change from a steady-state value and will often be omitted in the remaining part of this appendix.

II.1 Conventional Algorithms

For many years, programs have been developed to form explicitly the state space equations:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A} \mathbf{x} + \mathbf{b} u \\ y &= \mathbf{c}^t \mathbf{x} \end{aligned} \quad (4)$$

where u and y are a given pair of input and output variables.

Full eigensolution of non-sparse matrix \mathbf{A} (QR method) is normally restricted to systems of moderate size (about 800 states) due to the large memory and computation time requirements. Round-off errors in power system matrices of dimension over 1,000 also become significant, and a subset of eigenvalues in the QR eigensolution may be inaccurate. The state matrices in this application are fortunately not very ill-conditioned regarding eigensolutions. Figure 1 shows the state matrix topology map for the well known New England stability test system, showing it to be a dense matrix. The symbol “nz” in the caption of the figure stands for number of non-zero elements.

The transfer function $F(s)$ relating the input u and output y variables is obtained from the Laplace transformation of equation (4):

$$F(s) = \mathbf{c}^t (s\mathbf{I} - \mathbf{A})^{-1} \mathbf{b} \quad (5)$$

The frequency response analysis of dynamic systems can be performed by replacing the Laplace variable “ s ” by “ $j\omega$ ” in equation (5) and numerically calculating $F(j\omega)$ for discrete values of “ $j\omega$ ” within the frequency range of interest. The use of equation (5) becomes prohibitive for large order systems due to excessive computational time and memory requirements.

Transfer function residue calculation [14,19], eigenvalue sensitivity coefficients [20], time response to step disturbance and many other needed functions are all prohibitively expensive for large scale systems using the state space formulation.

II.2 Algorithms for Large Scale Systems

The concept that allows the methods of the previous section to be applied to large scale systems is the use of the *augmented system equations* [12,15], which is now described.

The basic equation relating state matrix, eigenvalues and eigenvectors is:

$$\mathbf{A} \mathbf{u} = \lambda \mathbf{u} \quad (6)$$

where λ is a system eigenvalue and \mathbf{u} its associated eigenvector.

This basic equation when expressed in terms of the Jacobian matrix shown in (2), becomes a generalized eigenvalue problem:

$$\begin{pmatrix} \mathbf{J}_1 & \mathbf{J}_2 \\ \mathbf{J}_3 & \mathbf{J}_4 \end{pmatrix} \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{r} \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{r} \end{pmatrix} \quad (7)$$

where $(\boldsymbol{\omega}^t | \mathbf{r}^t)^t$ is the *augmented* right eigenvector of λ and is denoted by $\underline{\boldsymbol{\omega}}^a$. Similarly, the *augmented* left eigenvector can be defined as $(\boldsymbol{\omega}^t | \mathbf{r}^t)^t$ and is denoted by $\underline{\mathbf{y}}^a$.

The state space equations of (4) can in a similar way be expressed as:

$$\begin{pmatrix} \dot{\mathbf{x}} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{J}_1 & \mathbf{J}_2 \\ \mathbf{J}_3 & \mathbf{J}_4 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{r} \end{pmatrix} + \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} u \quad (8)$$

$$y = (\mathbf{c}_1^t | \mathbf{c}_2^t) \begin{pmatrix} \mathbf{x} \\ \mathbf{r} \end{pmatrix} = (\mathbf{c}^a)^t \mathbf{x}^a$$

where

$(\mathbf{b}^a)^t = (\mathbf{b}_1^t | \mathbf{b}_2^t)$ = augmented input vector

\mathbf{c}^a = augmented output vector

\mathbf{x}^a = augmented state vector

These equations are referred to, in the modern control literature, as the Descriptor System Equations [37].

The large advantage of equations (7) and (8) is that the system Jacobian matrix is highly sparse and allows the use of efficient sparsity-based algorithms. Unfortunately, use is generally not made of partial pivoting during the sparse LU factorization. Pivoting improves the numerical stability of the factorization process at the expense of loss of sparsity in the LU factors and large rise in computational cost.

A good part of the algorithms for the solution of small-signal stability problems which make use of the augmented system equations concept are briefly described along the remaining part of this appendix.

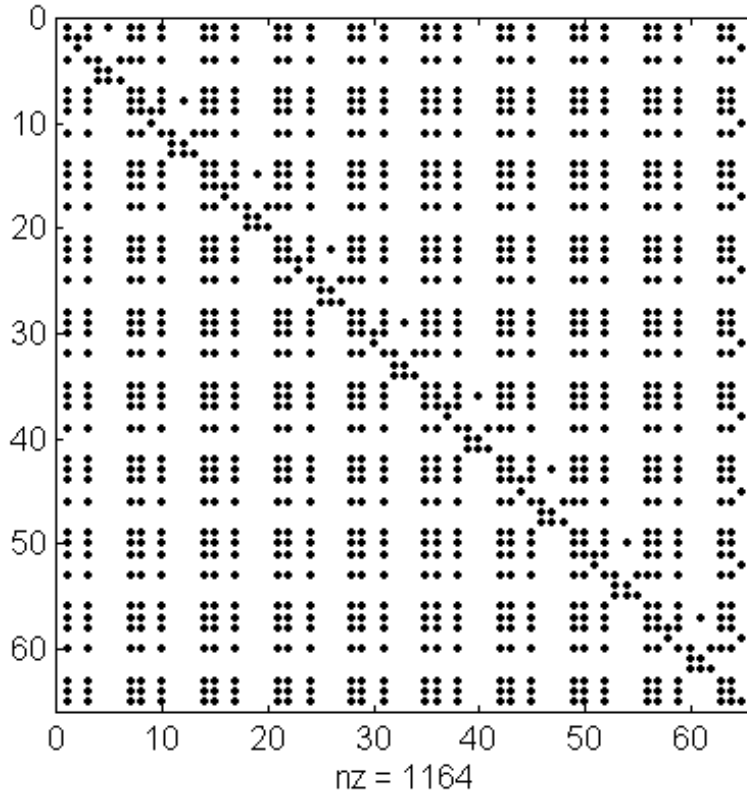


Figure 1 - State Matrix for New England System

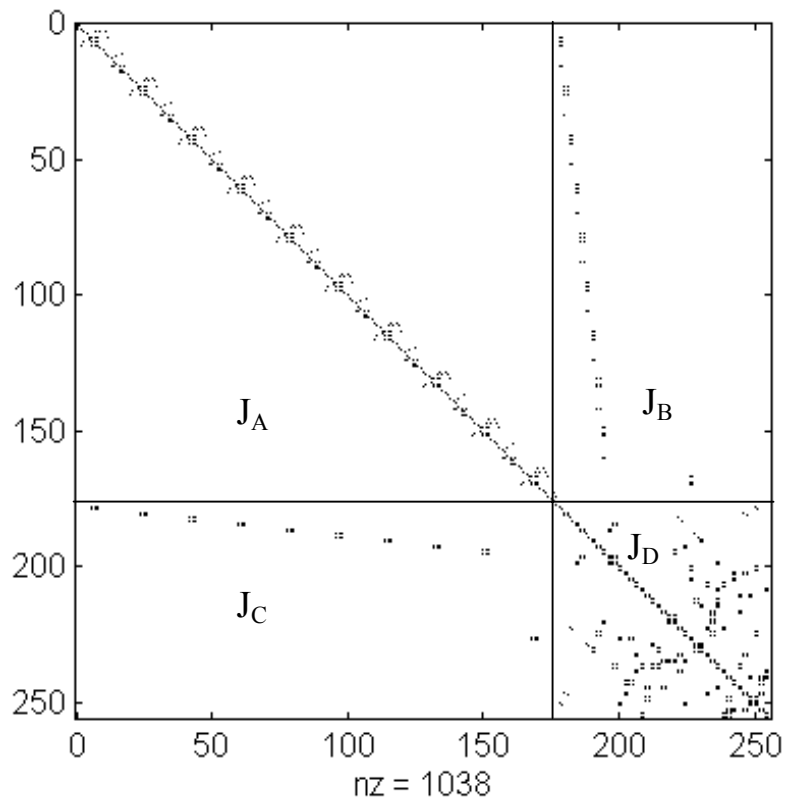


Figure 2 - Jacobian Matrix for New England System

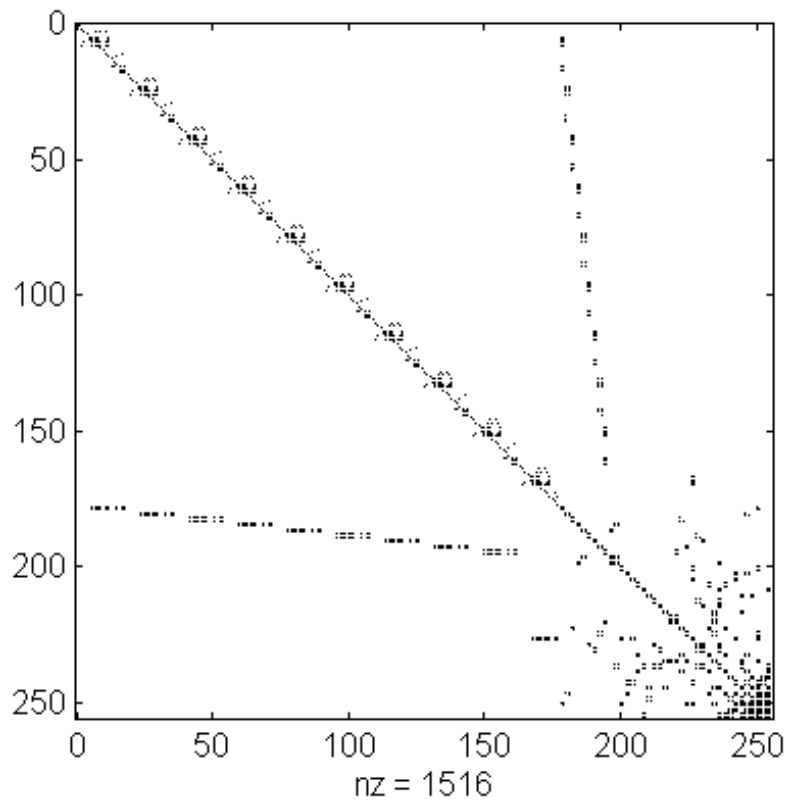


Figure 3 - LU Factors Topology Map for New England System Jacobian Matrix

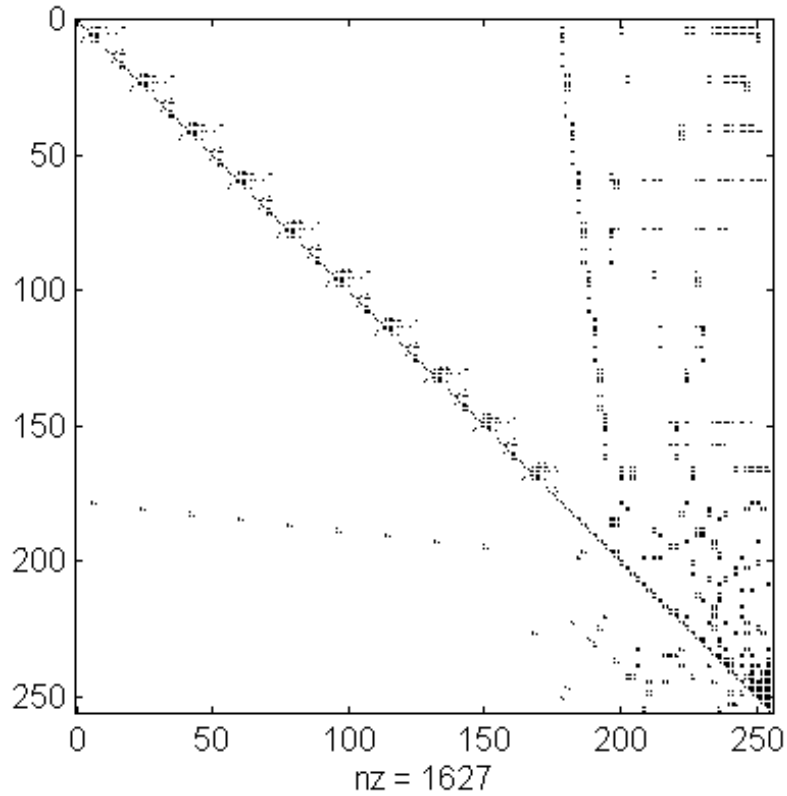


Figure 4 - LU Factors Topology Map for New England System Jacobian Matrix When Using Partial Pivoting

Figure 2 shows the Jacobian matrix topology for the New England system where the system variables are ordered in a different way to that shown in equation (2). In Figure 2 the equations for the dynamic components, either differential or algebraic, are assembled in J_A . The matrix block J_D contains the complex nodal admittance matrix equations, expanded into its real and imaginary components. The nodal admittance matrix equations are ordered by the Tinney-2 scheme, for minimum fill-in during factorization. The reader should refer to [12] for further details.

The Jacobian matrix shown in Figure 2 is real, unsymmetric and very sparse. Figure 3 pictures the triangular factors topology map for the Jacobian matrix of this system, which are also highly sparse. The LU factors were produced without the use of partial pivoting. Figure 4 shows the topology map for the LU factors obtained when using partial pivoting, which have a larger number of fill-ins. The reader should note that the New England system is of a too modest size to show the dramatic computational benefits gained with the use of sparsity techniques and the augmented system equations.

III. INVERSE ITERATION

The basic inverse iteration algorithm [8] with a complex shift ' q ' can be described by:

(i) Solve for \mathbf{w}_{k+1} :

$$(\mathbf{A} - q\mathbf{I}) \mathbf{w}_{k+1} = \mathbf{z}_k \quad (9)$$

(ii) Compute the vector \mathbf{z}_{k+1} for the next iteration:

$$\mathbf{z}_{k+1} = \frac{\mathbf{w}_{k+1}}{\max(\mathbf{w}_{k+1})} \quad (10)$$

then increment the iteration counter ($k = k+1$) and return to (i). Convergence occurs when the change in \mathbf{z} at any iteration is less than some specified tolerance. In this algorithm the subscript 'k' is the iteration number, ' \mathbf{I} ' is the identity matrix, ' q ' the approximation of the desired eigenvalue ' λ_i ' and $\max(\mathbf{w}_{k+1})$ is the element of largest magnitude in this vector. The vector \mathbf{z}_k has arbitrary initial value, and corresponds to the desired right eigenvector at convergence. After convergence, the factor $1/(\lambda_i - q)$ will be dominant in the element $\max(\mathbf{w}_{k+1})$ and the correct eigenvalue λ_i is given by:

$$\lambda_i = q + \frac{1}{\max(\mathbf{w}_{k+1})} \quad (11)$$

The rate of convergence of the inverse iteration algorithm depends on the ratio of the factor $(\lambda_i - q)$ to the next larger factor $(\lambda_j - q)$ [18]. Provided a good eigenvalue estimate ' q ' is given this method converges in 2 or 3 iterations. A variation of this method, which is more suitable when the given eigenvalue estimate is not good, involves refactorizing the matrix $(\mathbf{A} - q\mathbf{I})$ at every iteration, using as the new estimate ' q ' a corrected value obtained from the Rayleigh quotient (refer to the next section).

The application of the inverse iteration method in the form described in equations (9) and (10) is impractical for large power systems since the state matrix \mathbf{A} is not sparse. Equation (9) may however be written in implicit form [19] by defining an extra vector \mathbf{r}_{k+1} :

$$\begin{pmatrix} \mathbf{J}_1 - q\mathbf{I} & \mathbf{J}_2 \\ \mathbf{J}_3 & \mathbf{J}_4 \end{pmatrix} \begin{pmatrix} \mathbf{w}_{k+1} \\ \mathbf{r}_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{z}_k \\ \mathbf{0} \end{pmatrix} \quad (12)$$

where \mathbf{r}_{k+1} is not of direct interest in the solution and $\mathbf{0}$ is a null vector.

IV. RAYLEIGH QUOTIENT ITERATION

The Symmetric Matrix Case [9,10]

Given a symmetric matrix \mathbf{A} and a vector \mathbf{x} , the Rayleigh quotient of \mathbf{x} is:

$$r(\mathbf{x}) = \frac{\mathbf{x}' \mathbf{A} \mathbf{x}}{\mathbf{x}' \mathbf{x}} \quad (13)$$

Given an initial vector \mathbf{z}_k , the Rayleigh quotient iteration algorithm is as follows:

- (i) Compute the Rayleigh Quotient $r(\mathbf{z}_k)$
- (ii) Solve $(\mathbf{A} - r(\mathbf{z}_k)\mathbf{I})\mathbf{w}_{k+1} = \mathbf{z}_k$ for \mathbf{w}_{k+1}
- (iii) Compute \mathbf{z}_{k+1} for the next iteration $\mathbf{z}_{k+1} = \mathbf{w}_{k+1} / \|\mathbf{w}_{k+1}\|_2$
- (iv) Increment the iteration counter $k = k+1$ and return to (i)

The convergence process is ultimately cubic.

The Unsymmetric Matrix Case [11]

The Rayleigh quotient iteration algorithm for unsymmetric matrices involve the solution of approximate left eigenvectors and right eigenvectors at each iteration.

Given initial vectors \mathbf{z}_k and \mathbf{p}_k , the unsymmetric Rayleigh quotient iteration is as follows:

- (i) Compute the Rayleigh Quotient $r(\mathbf{z}_k, \mathbf{p}_k) = \frac{\mathbf{p}_k' \mathbf{A} \mathbf{z}_k}{\mathbf{p}_k' \mathbf{z}_k}$
- (ii) Solve $(\mathbf{A} - r(\mathbf{z}_k, \mathbf{p}_k)\mathbf{I})\mathbf{w}_{k+1} = \mathbf{z}_k$ for \mathbf{w}_{k+1}
- (iii) Solve $(\mathbf{A} - r(\mathbf{z}_k, \mathbf{p}_k)\mathbf{I})' \mathbf{q}_{k+1} = \mathbf{p}_k$ for \mathbf{q}_{k+1}
- (iv) Compute \mathbf{z}_{k+1} and \mathbf{p}_{k+1} for the next iteration $\mathbf{z}_{k+1} = \mathbf{w}_{k+1} / \|\mathbf{w}_{k+1}\|_2$ and $\mathbf{p}_{k+1} = \mathbf{q}_{k+1} / \|\mathbf{q}_{k+1}\|_2$

(v) Increment the iteration counter $k = k+1$ and return to (i)

The convergence process is again ultimately cubic. The Rayleigh quotient, at convergence, corresponds to a matrix eigenvalue while \mathbf{z}_k and \mathbf{p}_k correspond to the associated right and left eigenvectors.

V. FREQUENCY RESPONSE CALCULATIONS

The transfer function $F(s)$, expressed in equation (5) in terms of the state-space formulation, is given below in terms of the Descriptor System Equations [8]:

$$F(s) = \begin{bmatrix} \mathbf{c}'_1 & \mathbf{c}'_2 \end{bmatrix} \begin{bmatrix} s\mathbf{I} - \mathbf{J}_1 & -\mathbf{J}_2 \\ -\mathbf{J}_3 & -\mathbf{J}_4 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \quad (14)$$

The values of $F(s)$ are not computed through the explicit inversion of the above matrix. Use is made of sparse factorization and solution as indicated below. For discrete values of $s = j\omega$, within the frequency range of interest, solve for:

$$\begin{pmatrix} j\omega\mathbf{I} - \mathbf{J}_1 & | & -\mathbf{J}_2 \\ -\mathbf{J}_3 & | & -\mathbf{J}_4 \end{pmatrix} \begin{pmatrix} \mathbf{X}(j\omega) \\ \mathbf{R}(j\omega) \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \mathbf{U}(j\omega) \quad (15)$$

Normally, generator terminal voltage magnitude and active power are among the output variables of interest. The linearized expressions for these variables in the frequency domain are similar to those in the time domain, and are given by:

$$\Delta V_t(j\omega) = \frac{V_{r0}}{V_{t0}} \Delta V_r(j\omega) + \frac{V_{m0}}{V_{t0}} \Delta V_m(j\omega) \quad (16)$$

$$P_t(j\omega) = V_{r0} \Delta I_r(j\omega) + V_{m0} \Delta I_m(j\omega) + I_{r0} \Delta V_r(j\omega) + I_{m0} \Delta V_m(j\omega) \quad (17)$$

where the subscript $_o$ denotes a steady-state value for the variable, while r and m stand for real and imaginary components.

Expressions of the type shown in (16) and (17) define the augmented output vector $[\mathbf{c}'_1 \mid \mathbf{c}'_2]^t$ which is highly sparse and obviate the need to calculate the system output vector \mathbf{c} , associated with the state space description, which may be non-sparse.

VI. TRANSFER FUNCTION RESIDUES

The transfer function $F(s)$ shown in (5) can be expressed as [12,13]:

$$F(s) = \mathbf{c}'^t (s\mathbf{I} - \mathbf{A})^{-1} \mathbf{b} = \sum_{i=1}^n \frac{R_i}{s - \lambda_i} \quad (18)$$

where n is the dimension of \mathbf{A} and R_i is the residue of $F(s)$ associated with the eigenvalue λ_i . The residue R_i can be expressed as the product of a mode observability factor (c_i) by a mode controllability factor (b_i). These two factors can be easily calculated from the knowledge of both right and left augmented eigenvectors and the augmented input and output vectors as described in [13] and repeated below:

$$R_i = b_i \cdot c_i$$

$$\text{where } b_i = (\mathbf{w}^a)^t \cdot \mathbf{b}_a \\ c_i = (\mathbf{u}^a)^t \cdot \mathbf{c}_a$$

Left ($\underline{\omega}^a$) and right (\underline{u}^a) augmented eigenvectors are obtained through use of Rayleigh Quotient Iteration or any other partial eigensolution method.

Once the desired right and left eigenvectors are calculated, the residues for thousands of different transfer functions can be efficiently obtained by computing their scalar products with the highly sparse augmented input/output vectors.

VII. EIGENVALUE SENSITIVITY COEFFICIENTS

The eigenvalue sensitivity coefficients can be calculated, when using matrix \mathbf{A} , through the formula [14]:

$$\frac{\partial \lambda}{\partial \alpha} = \frac{\mathbf{v}^t \frac{\partial \mathbf{A}}{\partial \alpha} \mathbf{u}}{\mathbf{v}^t \mathbf{u}} \quad (19)$$

where $\underline{\mathbf{u}}$ and $\underline{\mathbf{v}}$ are the right and left eigenvectors associated with λ and α is a system parameter.

A general formula for eigenvalue sensitivity coefficients applied to the generalized eigenvalue problem was described in [15]. For the *augmented system equations* shown in (8) some simplification applies, yielding:

$$\frac{\partial \lambda}{\partial \alpha} = \frac{\mathbf{v}_a^t \frac{\partial \mathbf{J}}{\partial \alpha} \mathbf{u}_a}{\mathbf{v}^t \mathbf{u}} \quad (20)$$

where \mathbf{J} is the Jacobian matrix shown in (7).

Useful information can be obtained through eigenvalue sensitivity analysis when α is chosen to be a network impedance, a dynamic component or controller parameter, etc. Sensitivity with respect to system voltages and generation or load changes is more intricate, but can also be done.

Eigenvalue sensitivity to controller parameter changes can be efficiently computed through the product of system transfer function residues (refer to the previous section) by simple partial derivatives of the controller transfer function. A detailed description and use of this method can be found in [25,26].

VIII. TIME RESPONSE OF THE LINEARIZED SYSTEM

Consider the linearized system state space equations shown in equation (4). A step disturbance is to be applied to the input variable u and the dynamic response of the system monitored through the variables in $\underline{\mathbf{x}}$ and y . Adopting the implicit trapezoidal algorithm for the numerical integration of the system equations one obtains [7]:

$$\left(\frac{2}{h} \mathbf{I} - \mathbf{A} \right) \mathbf{x}_{i+1} = \left(\frac{2}{h} \mathbf{I} + \mathbf{A} \right) \mathbf{x}_i + 2\mathbf{b} \quad (21)$$

where h is the time step of integration, \mathbf{I} the identity matrix and $\underline{\mathbf{x}}_i$ and $\underline{\mathbf{x}}_{i+1}$ are values at time steps t_i and $t_{i+1} = t_i + h$. By referring to equations (4) and (8) it can be seen that equation (21) is equivalent to:

$$\left(\begin{array}{c|c} \frac{2}{h} \mathbf{I} - \mathbf{J}_1 & -\mathbf{J}_2 \\ \hline -\mathbf{J}_3 & -\mathbf{J}_4 \end{array} \right) \begin{pmatrix} \mathbf{x}_{i+1} \\ \mathbf{x}_{i+1} \end{pmatrix} = \left(\begin{array}{c|c} \frac{2}{h} \mathbf{I} + \mathbf{J}_1 & \mathbf{J}_2 \\ \hline \mathbf{J}_3 & \mathbf{J}_4 \end{array} \right) \begin{pmatrix} \mathbf{x}_i \\ \mathbf{r}_i \end{pmatrix} + 2 \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \quad (22)$$

The algorithm in (22) is efficient when implemented with sparsity coding. Vector $\underline{\mathbf{c}}^t$ of equation (4) is not needed since the output variable y can be obtained from simple expressions in terms of the variables contained in the solution vectors $\underline{\mathbf{x}}_{i+1}$ and $\underline{\mathbf{r}}_{i+1}$.

IX. TRANSFER FUNCTION ZEROS

The zeros of the transfer function of equation (5) can be obtained by solving the generalized eigenvalue problem [16]:

$$\left(\begin{array}{c|c} A & b \\ \hline c^t & 0 \end{array} \right) \begin{pmatrix} \mathbf{x} \\ \mathbf{u} \end{pmatrix} = \lambda \left(\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline \mathbf{0}^t & 0 \end{array} \right) \begin{pmatrix} \mathbf{x} \\ \mathbf{u} \end{pmatrix} \quad (23)$$

A standard library QZ eigensolution routine [17,18] can be used to obtain the complete set of transfer function zeros, as long as the system is of moderate size.

Subspace Iteration methods can be used to efficiently calculate several zeros at a time of large power system dynamic models. These methods must be applied to the augmented generalized eigenvalue problem, which is sparse:

$$\left(\begin{array}{c|c|c} \mathbf{J}_{1-q\mathbf{I}} & \mathbf{J}_2 & \mathbf{b}_1 \\ \hline \mathbf{J}_3 & \mathbf{J}_4 & \mathbf{b}_2 \\ \hline \mathbf{c}_1^t & \mathbf{c}_2^t & 0 \end{array} \right) \begin{pmatrix} \mathbf{x} \\ \mathbf{r} \\ u \end{pmatrix} = \lambda \left(\begin{array}{c|c|c} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0}^t & \mathbf{0}^t & 0 \end{array} \right) \begin{pmatrix} \mathbf{x} \\ \mathbf{r} \\ u \end{pmatrix} \quad (24)$$

X. EFFICIENT STATE MATRIX FORMATION

The state matrix \mathbf{A} of equation (4) can be directly obtained by performing sparse Gaussian elimination to the Jacobian matrix of (2) [8]. This method is however not efficient for large scale systems due to the inevitable excessive fill-in, since \mathbf{A} is non-sparse in this application (please refer to Figure 1).

An efficient method for obtaining \mathbf{A} directly from the augmented system equations [7] is now described. The large and non-sparse matrix \mathbf{A} is here formed by calculating one of its columns at a time. Let \mathbf{e}_i be a singleton, i.e., a vector with a real unity value at the i -th position and zeros elsewhere. The product $\mathbf{A} \mathbf{e}_i$ yields the i -th column of the state matrix \mathbf{A} . This elementary matrix/vector product constitutes the basis of this algorithm that fully exploits the sparsity of the Jacobian matrix.

Two steps need be performed to obtain \mathbf{a}_i , the i -th column of the state matrix:

1) Solve for vector \mathbf{r}_i

$$\left(\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline \mathbf{J}_3 & \mathbf{J}_4 \end{array} \right) \begin{pmatrix} \mathbf{c}_i \\ \mathbf{r}_i \end{pmatrix} = \begin{pmatrix} \mathbf{c}_i \\ 0 \end{pmatrix} \quad (25)$$

2) Perform the product below to obtain \mathbf{a}_i

$$\begin{pmatrix} \mathbf{a}_i \\ \mathbf{0} \end{pmatrix} = \left(\begin{array}{c|c} \mathbf{J}_1 & \mathbf{J}_3 \\ \hline \mathbf{J}_3 & \mathbf{J}_4 \end{array} \right) \begin{pmatrix} \mathbf{c}_i \\ \mathbf{r}_i \end{pmatrix} \quad (26)$$

Note that only one real matrix factorization is needed to obtain all columns of the state matrix \mathbf{A} .

The practical implementations of the algorithms described in the remaining part of this appendix operate directly on the sparse non-reduced Jacobian, also known as augmented system equations or Descriptor System Equations. However, for the sake of clarity and brevity, they will be all described as operating directly on the state matrix \mathbf{A} .

XI. AESOPS

The AESOPS algorithm [5,19] is a quasi-Newton one-at-a-time eigenvalue method designed to compute the electromechanical modes of oscillation for large power systems. The AESOPS algorithm is derived from the linearized equation of motion of a chosen generator, to which a complex frequency disturbance in the mechanical torque is applied. At every iteration, a corrected value for this complex frequency disturbance is applied until the system becomes resonant. This iterative process is almost always convergent and the converged complex frequency value corresponds to an electromechanical eigenvalue which is dominant at the disturbed generator.

In this section it is assumed that the AESOPS formulation and its notation [19,20] are familiar to the reader. The algorithm steps are:

(i) Solve for $\mathbf{X}(z^k)$, given an initial estimate of z^k :

$$(z^k \mathbf{I} - \mathbf{A}) \mathbf{X}(z^k) = \mathbf{b} \cdot T_x(z^k) \quad (27)$$

(ii) Compute the eigenvalue estimate $(\alpha + j\beta)^k$ at iteration k :

$$(\alpha + j\beta)^k = z^k - \frac{1}{2}(U + jV)^k (\beta_1 + j\beta_0) \quad (28)$$

(iii) Define the complex frequency z^{k+1} of the mechanical torque input T_x for the next iteration:

$$z^{k+1} = (\alpha + j\beta)^k \quad (29)$$

and return to (i). Convergence occurs when the difference between two successive eigenvalue estimates is below a specified tolerance. At convergence, the vector $\mathbf{X}(z^k)$ is equal to the eigenvector associated with the system eigenvalue z^k .

The following definitions are necessary regarding the above algorithm:

$T_x(z^k)$ = external torque phasor applied at a specified generator, at iteration k , which yields a rotor speed phasor $\omega(z^k) = (1 + j0.0)$ at this generator.

$\sum_{i=1}^n 2H_i |\omega_i(z^k)|$ = momentum function involving all generator inertia constants and the absolute values of the rotor speed phasors.

$(U + jV)^k$ = $T_x(z^k) / (\sum_{i=1}^n 2H_i |\omega_i(z^k)|)$

$(\beta_1 + j\beta_0)$ = acceleration factor

The rotor speed phasors $\omega_i(z^k)$ for all generators are a subset of $\mathbf{X}(z^k)$. The calculation of these rotor speed phasors is the only large computational task required by the AESOPS algorithm.

XII. THE DOMINANT POLE ALGORITHM

This algorithm efficiently computes the dominant poles of any specified high order transfer function [20]. It is completely general and, therefore, not restricted to power system applications. The method is closely related to unsymmetric Rayleigh Quotient Iterations (see Section IV) and has the numerical properties of global and ultimately quadratic convergence.

A scalar transfer function $G(s)$ is equal to its transpose:

$$G(s) = \frac{y(s)}{u(s)} = \mathbf{c}^t (s\mathbf{I} - \mathbf{A})^{-1} \mathbf{b} = \mathbf{b}^t (s\mathbf{I} - \mathbf{A})^{-t} \mathbf{c} \quad (30)$$

where \mathbf{A} is a state matrix of order n , s is the Laplace variable, \mathbf{b} is the input column vector, \mathbf{c}^t is the output row vector, u is the input variable, y is the output variable and the superscript $-t$ denotes the inverse transpose of a matrix.

Equation (30) can be expressed in matrix form:

$$\begin{bmatrix} (s\mathbf{I}-\mathbf{A}) & -\mathbf{b} \\ \mathbf{c}^t & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}(s) \\ u(s) \end{bmatrix} = \begin{bmatrix} (s\mathbf{I}-\mathbf{A})^t & -\mathbf{c} \\ \mathbf{b}^t & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}(s) \\ u(s) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ y(s) \end{bmatrix} \quad (31)$$

where $\mathbf{x}(s)$ and $\mathbf{v}(s)$ are the Laplace transforms of the state vectors for \mathbf{A} and \mathbf{A}^t .

A pole of a transfer function may be defined as $\lambda \in C$ such that $G(\lambda) \rightarrow \infty$. As s approaches λ , the input $u(s)$ tends to zero for any finite value of $y(s)$ and the vectors $\mathbf{x}(s)$ and $\mathbf{v}(s)$ in equation (31) converge to the right and left eigenvectors \mathbf{q} and \mathbf{p} . This reasoning led to the development of the algorithm described below:

Dominant Pole Algorithm

1. Given a single-input-single-output transfer function $G(s) = \mathbf{c}^t (s\mathbf{I} - \mathbf{A})^{-1} \mathbf{b}$, provide an initial eigenvalue estimate s_k and specify the value for the output at all iterations as unity ($y(s_k) = 1$), k being the iteration counter.
2. Solve $\begin{bmatrix} s_k \mathbf{I} - \mathbf{A} & -\mathbf{b} \\ \mathbf{c}^t & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}(s_k) \\ u(s_k) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$
3. Solve $\begin{bmatrix} s_k \mathbf{I} - \mathbf{A}^t & -\mathbf{c} \\ \mathbf{b}^t & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}(s_k) \\ u(s_k) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$ using the transpose of the LDU factors calculated at step 2.
4. Compute the new eigenvalue estimate: $s_{k+1} = s_k + \frac{u(s_k)}{\mathbf{v}^t(s_k) \cdot \mathbf{x}(s_k)}$

This expression shows that s_{k+1} converges to the eigenvalue λ as the input $u(s_k)$ approaches zero.

5. If change in eigenvalue estimate ($\Delta s = s_{k+1} - s_k$) is greater than convergence tolerance increase the iteration counter ($k = k+1$) and return to step 2. Otherwise the algorithm has converged to an eigenvalue which is a dominant pole in $G(s)$, and to its left and right eigenvectors.

The estimate s_{k+1} , described in step 4, stems from the Rayleigh quotient shift for non-symmetric matrices [11]

$$s_{k+1} = \frac{\mathbf{v}^t(s_k) \cdot \mathbf{A} \cdot \mathbf{x}(s_k)}{\mathbf{v}^t(s_k) \cdot \mathbf{x}(s_k)} \quad (32)$$

combined with the two expressions below which are taken from the matrix equations in steps 2 and 3.

$$\mathbf{A} \cdot \mathbf{x}(s_k) = \mathbf{b} \cdot u(s_k) + s_k \cdot \mathbf{x}(s_k) \quad (33)$$

$$\mathbf{v}^t(s_k) \cdot \mathbf{b} = \mathbf{b}^t \cdot \mathbf{v}(s_k) = 1 \quad (34)$$

The proposed algorithm has global and ultimately quadratic convergence of s_k to any of the transfer function dominant poles. It never converges to a system eigenvalue which is not observable nor controllable for the transfer function under analysis, even when the initial complex shift is made exactly equal to this eigenvalue.

Note, from step 2, that $-\mathbf{c}^t (s_k \mathbf{I} - \mathbf{A})^{-1} \mathbf{b} \cdot u(s_k) = G(s_k) \cdot u(s_k) = 1$. Therefore, if s_k is made equal to transfer function zero, $G(s_k)$ will be zero and the input $u(s_k)$ will assume an infinite value. The expression in step 4, shows that s_{k+1} will be catapulted away to infinity when s_k is made exactly equal to a transfer function zero.

A practical algorithm implementation requires defining a maximum allowed change in the eigenvalue estimate between iterations (step length control), to ensure convergence to the dominant poles which are closest to the given initial estimate (s_0).

XIII. DOMINANT TRANSFER FUNCTION ZEROS (to be published)

XIV. DOMINANT POLES FOR MIMO TRANSFER FUNCTIONS (to be published)

XV. SIMULTANEOUS ITERATION ALGORITHM

Lop-sided simultaneous iteration [21,22,23,24,25], is a method to determine an invariant subspace $\mathbf{U} \in \mathbb{C}^{n \times m}$ of the unsymmetric $n \times n$ matrix \mathbf{A} from an initial set of m trial vectors, $m < n$. This method can efficiently calculate the dominant eigenvalues/eigenvectors of \mathbf{A} . In the small-signal stability area, the least damped or unstable eigenvalues are of interest and can be made dominant (with large moduli) through the spectral transformation $(\mathbf{A}-q\mathbf{I})^{-1}$, where q is a complex shift and \mathbf{I} the identity matrix. This method is well described in [21,22,23] and briefly presented here.

The Lop-sided SI algorithm of this paper is shown in Figure 5, utilizing the nomenclature established in references [21,22,23]. According to [24,25], one may refer to this technique as Implicit Inverse Lop-sided Simultaneous Iteration (IILSI). The IILSI algorithm can be viewed as a combination of the simultaneous iteration method and the implicit inverse iteration method. This technique will compute the m eigenvalues of \mathbf{A} which are closest to the complex shift q and the associated right eigenvectors. Matrix $(\mathbf{A}-q\mathbf{I})^{-1}$ described in task 3 of Figure 1 is never explicitly formed. Matrix $\mathbf{V}^{(k)}$, of task 3, is formed by columns generated from repeat solutions on the LU factors obtained in task 2.

The various steps of the IILSI algorithm are numbered 1 to 10 in Figure 5. A description of these steps is given below:

- 1) Initialize m linearly independent vectors \mathbf{u}_i^k ($k = 1, i = 1, m$), which form the columns of matrix \mathbf{U} ;
- 2) Obtain the sparse LU factors of $(\mathbf{A}-q\mathbf{I})$;
- 3) Solve for each column \mathbf{v}_i^k of \mathbf{V}^k , given the sparse LU factors and the columns \mathbf{v}_i^k of \mathbf{V}^k ;
- 4) Compute \mathbf{G}^k , given matrix \mathbf{U}^k ;
- 5) Compute \mathbf{H}^k , given the \mathbf{U}^k and \mathbf{V}^k matrices;
- 6) Solve $\mathbf{G}^k \mathbf{B}^k = \mathbf{H}^k$ for \mathbf{B}^k ;
- 7) Perform complete eigensolution on \mathbf{B}^k , by the QR method. Denote the eigenvalues by $\{\lambda_1^k, \dots, \lambda_m^k\}$ and corresponding right eigenvectors by $\{p_1^k, \dots, p_m^k\}$;
- 8) Compute \mathbf{W}^k , given the \mathbf{V}^k and \mathbf{P}^k matrices;
- 9) Obtain matrix \mathbf{U}^{k+1} , for use at the next iteration, by normalizing every column of \mathbf{W}^{k+1} ;
- 10) Convergence Test. If not converged, increment iteration counter and return to step 3.

Practical SI algorithms incorporate several techniques to improve their efficiency and convergence: use of guard vectors, fast iteration cycles (omission of the reorientation process) and locking device (locking vectors) [25,26,27,28]. These techniques are all included in practical algorithms but are omitted from the flow diagram in Figure 1 for the sake of clarity.

The convergence test adopted here checks eigenvector tolerances between successive iterations.

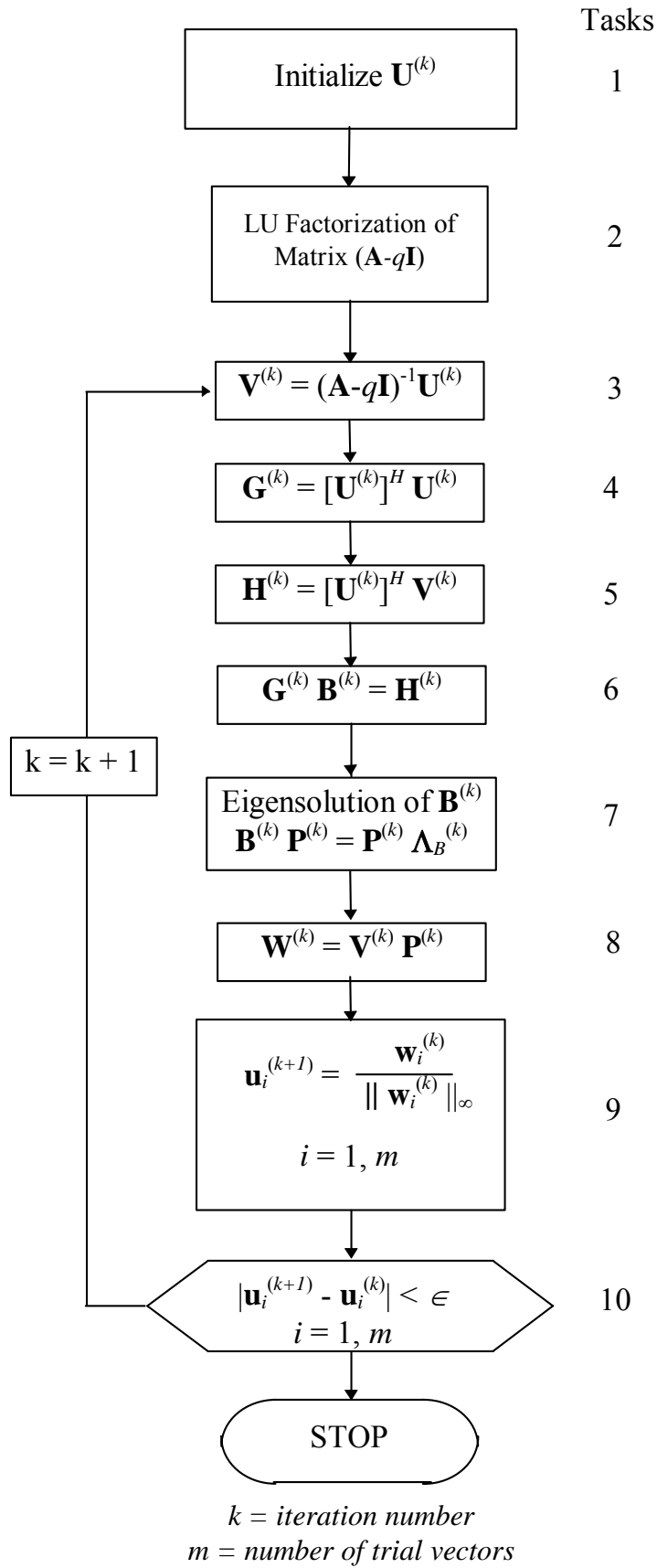


Figure 5 - Lop-sided SI Algorithm

XVI. MODIFIED ARNOLDI [25] (to be written)

XVII. REFACTORED BI-ITERATION (RBI) ALGORITHM

Refactored Bi-Iteration is a new numerical method for the partial eigensolution of large unsymmetric matrices. The method employs multiple moving shifts and converges to as many eigenvalues as the number of trial vectors utilized, in a reduced number of iterations. The method can be seen as a multi-vector (subspace) generalization of the single-vector Rayleigh Quotient iteration for unsymmetric matrices.

The RBI algorithm [29] is presented in the flow chart of Figure 6. The Lop Sided Simultaneous Iteration (LSSI) algorithm was described in a previous section with a similar flow chart. The symbols adopted to describe the RBI algorithm are the same as those utilized in [22,23,24,25]. A reader who is familiarized with the LSSI algorithm, commonly used in power system eigenanalysis [22,23,24,25,26,27,28], is expected to readily understand the concepts of the RBI algorithm.

The following comments apply regarding the flow chart of Figure 6:

- \mathbf{A} is the large, sparse ($n \times n$) matrix whose eigenvalues need be obtained;
- Symbols k and m denote the iteration counter and the number of trial vectors utilized, respectively;
- Matrices $\mathbf{U}^{(k)}$, $\mathbf{V}^{(k)}$, $\overline{\mathbf{U}}^{(k)}$ and $\overline{\mathbf{V}}^{(k)}$ have dimensions ' $n \times m$ ', being composed of vectors $\mathbf{u}_i^{(k)}$, $\mathbf{v}_i^{(k)}$, $\overline{\mathbf{u}}_i^{(k)}$ and $\overline{\mathbf{v}}_i^{(k)}$, $i = 1, \dots, m$;
- Matrices $\mathbf{G}^{\{k\}}$, $\mathbf{H}^{\{k\}}$ and $\mathbf{B}^{\{k\}}$ have order ' m ', $m \ll n$;
- The elements in $\mathbf{U}^{(o)}$ and $\overline{\mathbf{U}}^{(o)}$ are computed by a random number generator and they need not be bi-orthogonal;
- The initial shifts $\lambda_A^{(o)}$, $i = 1, \dots, m$ are provided by the user, according to the phenomenon under analysis;
- Although not explicitly indicated in the flow chart, use is made of Fast Iteration Cycles [22,23,24,25]. Three to five cycles per iteration are the recommended values;
- The symbols $\lambda_B^{(k)}$, $i = 1, \dots, m$ denote the eigenvalues of the interaction matrix $\mathbf{B}^{\{k\}}$. This matrix ultimately becomes completely diagonal with all elements having magnitudes larger than ϵ^{-1} , where ϵ is the convergence tolerance. The tolerance used to obtain the results of this paper was $\epsilon = 10^{-10}$;
- The formula utilized for updating the shifts $\lambda_A^{(k)}$, is only approximate when away from the solution but exact in its neighborhood. Each one of these shifts ultimately converges to a different eigenvalue of matrix \mathbf{A} ;
- The convergence test consists in verifying whether the Euclidean norm of the mismatch vector ($\mathbf{A} \mathbf{x}_i - \lambda_i \mathbf{x}_i$) is below some small tolerance.

Note: The practical implementation of this algorithm to the power system stability model operates on the sparse non-reduced Jacobian [22,23,24,25,26,27,28], but for the sake of clarity it is here described as operating directly on the state matrix \mathbf{A} .

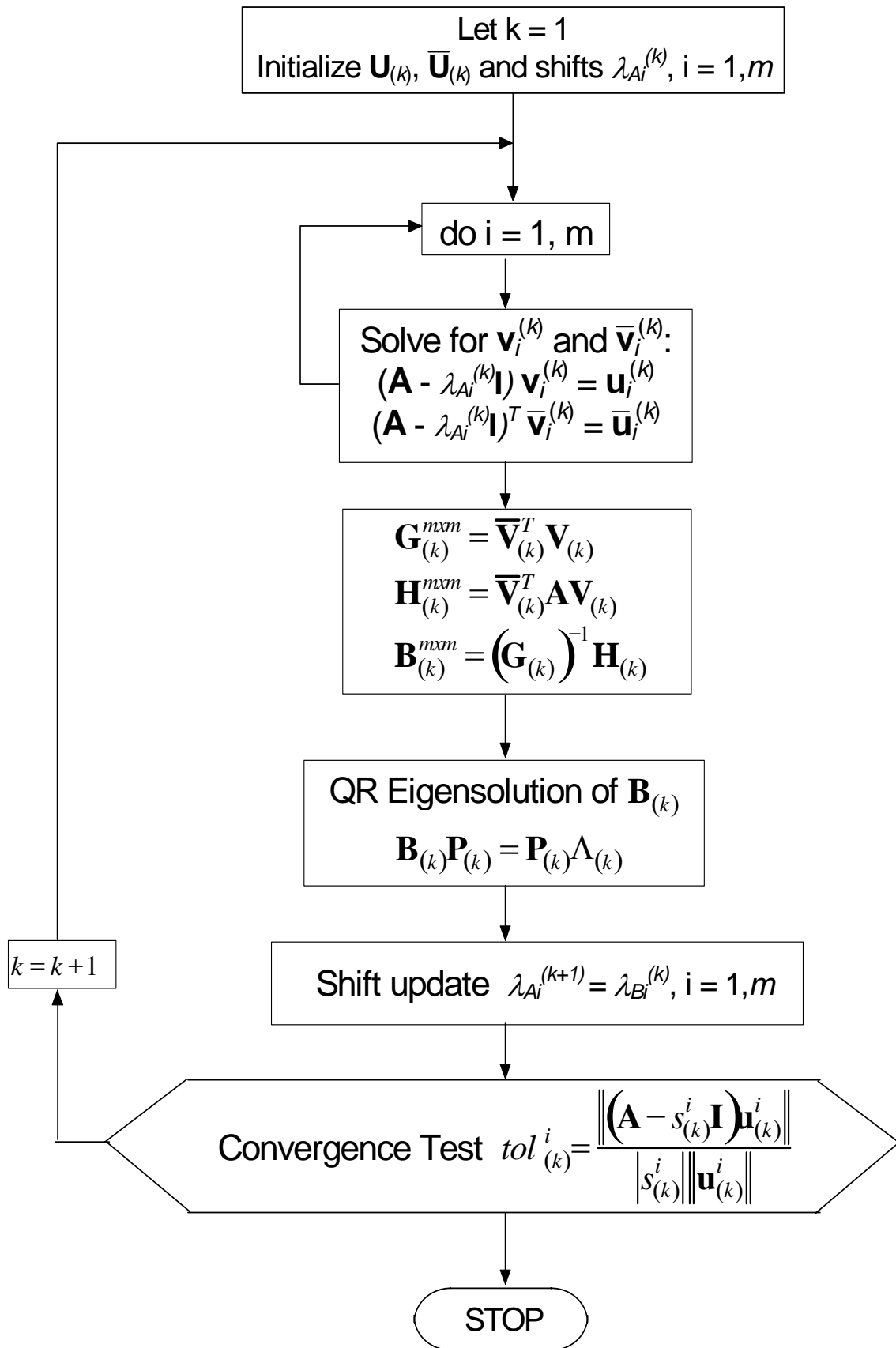


Figure 6 - Refactored Bi-Iteration Algorithm

XVIII. THE DOMINANT POLE SPECTRUM EIGENSOLVER

The DPSE [30] was developed from the combination of two recent algorithms:

1. The Refactored Bi-Iteration (RBI) [29], a subspace iteration method derived from the Bi-Iteration algorithm [21,22,23], but employing multiple moving-shifts;
2. The Dominant Pole Algorithm [20], a one-eigenvalue-at-a-time method which computes the dominant closed-loop poles in the transfer function $F(s)$.

The DPSE is presented in the flow chart of Figure 7. The RBI algorithm was described in a previous section with a similar flow chart. The DPSE is a subspace iteration method, which operates on both left and right subspaces to produce better estimates for the dominant poles of $F(s)$. The scalar transfer function $F(s) = \mathbf{c}^T \cdot (s\mathbf{I} - \mathbf{A})^{-1} \cdot \mathbf{b} + d$ need be expressed in matrix form as described in both Figure 7 and reference [20]. The term 'd', called the direct transmission element [1,2], is readily considered by the DPSE but it is here assumed to be zero for simplicity.

The following comments together with the flow chart in Figure 7 help explain the DPSE:

- \mathbf{A} is the ($n \times n$) state matrix whose selected set of eigenvalues need be obtained. Vectors \mathbf{b} and \mathbf{c} are the input and output vectors of transfer function $F(s)$. Symbols \mathbf{I} and $\mathbf{0}$ denote the identity matrix and the ($n \times 1$) null vector, respectively.
- Symbols k and m denote the iteration counter and the total number of moving-shifts $s_i^{(k)}$ utilized, respectively. A matrix or vector transposed has an upperscript T.
- A single **LDU** factorization is required to solve for both right and left eigenvector estimates $\mathbf{x}(s_i^{(k)})$ and $\mathbf{v}(s_i^{(k)})$.
- The matrix factorizations and vector solutions comprise the power method stage of the subspace iteration.
- Every vector solution is implicitly scaled [20] such that $\mathbf{c}^T \cdot \mathbf{x}(s_i^{(k)}) = -\mathbf{b}^T \cdot \mathbf{v}(s_i^{(k)}) = 1$.
- Matrices $\mathbf{V}^{(k)}$ and $\mathbf{X}^{(k)}$ have dimensions ' $n \times m$ ', being composed of the left and right eigenvector estimates $\mathbf{v}(s_i^{(k)})$ and $\mathbf{x}(s_i^{(k)})$, $i = 1, \dots, m$. Matrices $\mathbf{G}^{(k)}$, $\mathbf{H}^{(k)}$ and $\mathbf{B}^{(k)}$ have order ' m ', $m \ll n$. These matrices are needed in the reorthogonalization stage of the subspace iteration.
- As a result of a full reorthogonalization process, carried out at every iteration, the DPSE does not produce repeated eigensolutions.
- The initial shifts $s_i^{(1)}$, $i = 1, \dots, m$ are user-specified, and should preferably have values compatible with the physical phenomenon under analysis. The moving-shifts $s_i^{(k)}$, $i = 1, \dots, m$ ($k \geq 2$) are recomputed at every iteration and converge to those eigenvalues of \mathbf{A} belonging to the dominant pole spectrum of $F(s)$.
- The DPSE does not update the trial vectors, but only the moving-shifts. The method converges to as many dominant poles in $F(s)$ as the number of initial shifts provided.
- Mathematically speaking there can be no guarantee that all dominant poles in a high-order $F(s)$ will be found by DPSE when using a limited number of moving-shifts ($m \ll n$). Frequency and step response tests can be used to check whether all relevant poles have been found (see figures 10, 11, 12, 13 of [30]).
- The convergence test consists in verifying whether the Euclidean norm of the mismatch vector ($\mathbf{A} \mathbf{x}_i - \lambda_i \mathbf{x}_i$) is below some small tolerance. Convergence is usually faster for poles having large transfer function residues. Although not indicated in the flow chart of Figure 7, the integer ' m ' is decremented by one every time a dominant pole is accurately obtained. This means that the number of moving-shifts is reduced as more dominant poles become known.

The reader should refer to [30] for more details on this method.

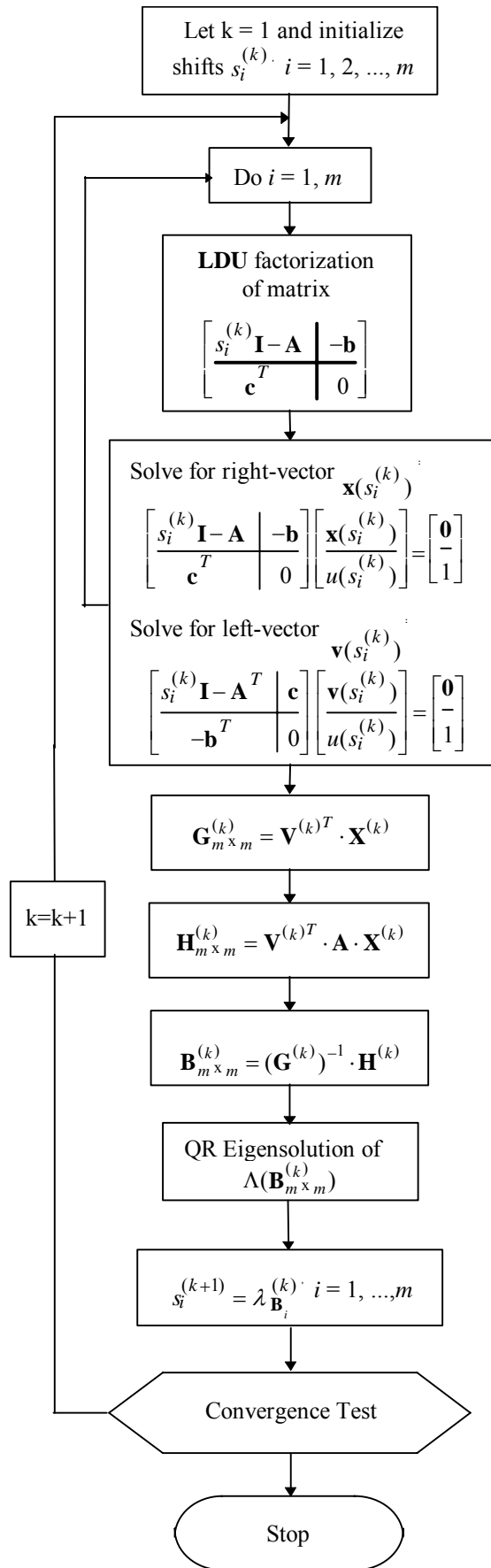


Figure 7 - The Dominant Pole Spectrum Eigensolver.

XIX The MIMO Transfer Function Dominant Pole (MDP) Algorithm and its use in model order reduction

This section describes a numerical linear algebra algorithm to compute the dominant poles of Multi-Input-Multi-Output (MIMO) high-order transfer functions. The results presented are related to the study of electromechanical oscillations in large electrical power systems, but the algorithm is completely general. The proposed algorithm allows obtaining reduced-order models for MIMO transfer functions of large linear systems.

The model-based, one-eigenvalue-at-a-time algorithm described in this section determines the dominant poles of an $m \times m$ transfer function matrix $\mathbf{G}(s)$, and will be referred to by the following name and acronym: the **MIMO Transfer Function Dominant Pole (MDP)** algorithm. Once the dominant poles of the high-order $\mathbf{G}(s)$ are known, the associated transfer function residue matrices can be readily computed. This paper also describes the use of this pole-residue information to build low-order approximations of MIMO transfer functions.

The results of this section show the effectiveness of the MDP algorithm in determining the dominant poles of a 8×8 transfer function matrix of a large power system model (1,676 state variables). Results on some of the reduced-order models obtained for this 8×8 transfer function are included.

All current eigensolution algorithms for large, practical power system problems operate on the sparse, unreduced Jacobian [8], [5], [35], [20], [30] (descriptor system formulation), but for the sake of brevity the MDP algorithm is here described as directly operating on the well-known state-space model. The symbols utilized are defined as used. The terms MIMO and multivariable are used interchangeably. The same applies to the terms poles and eigenvalues.

The eigensolution method described in this paper only converges to those eigenvalues of \mathbf{A} that are the dominant poles of the multivariable transfer function $\mathbf{G}(s)$. Reduced-order models of multivariable transfer functions of large system models may therefore be obtained without resorting to the computationally expensive singular value decomposition.

The model reduction method utilized in this paper only requires the knowledge of the dominant poles of $\mathbf{G}(s)$.

XIX.1 The MDP Algorithm

The $m \times m$ transfer function $\mathbf{G}(s)$ may be expressed in terms of the system state-space matrices:

$$\mathbf{G}(s) = \mathbf{C} \cdot (s\mathbf{I} - \mathbf{A})^{-1} \cdot \mathbf{B} + \mathbf{D} \quad (1)$$

\mathbf{A} ($n \times n$) is the system state matrix; \mathbf{B} ($n \times m$) and \mathbf{C} ($m \times n$) are the input and output matrices. Symbol \mathbf{I} denotes the $n \times n$ identity matrix. A vector or matrix transpose has a superscript T . Matrix \mathbf{D} is the direct transmission matrix [1], which can be readily considered by the MDP algorithm but it is here assumed to be zero for simplicity.

The proposed MDP algorithm is a one-eigenvalue-at-a-time algorithm that computes the dominant poles of a square transfer function matrix $\mathbf{G}(s)$. It represents a significant, though natural, extension of the Dominant Pole Algorithm [20], a one-eigenvalue-at-a-time method that computes the dominant poles of scalar transfer functions. Every iteration of the MDP algorithm involves two eigensolutions by the power method, as described in the box below.

The MDP Algorithm

1. Let $k = 1$, k being the iteration counter for the Rayleigh type algorithm at the outer loop. Given an $m \times m$ transfer function $\mathbf{G}(s) = \mathbf{C} \cdot (s\mathbf{I} - \mathbf{A})^{-1} \cdot \mathbf{B}$, provide the initial eigenvalue estimate s_1 (also known as initial shift) and the randomly generated complex vectors $\mathbf{u}(s_0)$ and $\mathbf{y}(s_0)$.

2. Compute $\mathbf{G}(s_k)$. The symbols i and j denote the iteration counters for the two eigenvalue power methods (Inner Loops). Using the inner-loop eigenvectors $\mathbf{u}(s_{k-1})$ and $\mathbf{y}(s_{k-1})$ from the previous outer-loop iteration (or the random vectors $\mathbf{u}(s_0)$, $\mathbf{y}(s_0)$ associated with the initial shift s_1) as initial values, iterate the two eigenvalue power methods until convergence:

$$\mathbf{u}(1) = \mathbf{u}(s_{k-1})$$

for $i = 1, 2, \dots$

$$\mathbf{z}(i+1) = \mathbf{G}(s_k) \cdot \mathbf{u}(i)$$

$$\mathbf{u}(i+1) = \frac{\mathbf{z}(i+1)}{\|\mathbf{z}(i+1)\|}$$

end

$$\mathbf{y}(1) = \mathbf{y}(s_{k-1})$$

for $j = 1, 2, \dots$

$$\mathbf{w}(j+1) = \mathbf{G}^T(s_k) \cdot \mathbf{y}(j)$$

$$\mathbf{y}(j+1) = \frac{\mathbf{w}(j+1)}{\|\mathbf{w}(j+1)\|}$$

end

These two power methods converge in about the same number of iterations, when the changes in the vectors \mathbf{u} and \mathbf{y} become smaller than the inner-loop tolerance. The maximum number of power iterations should be set to about 40 at the first outer-loop iteration ($k = 1$), where the vectors \mathbf{u} and \mathbf{y} are given initial random values, but reduces to one or two iterations as the MDP algorithm approaches convergence. The two power methods will converge to the same eigenvalue of $\mathbf{G}(s_k)$ ($\lambda_{\max}(s_k)$, which has the largest modulus), but to different eigenvectors (at the inner-loop convergence, $\mathbf{u}(i+1)$ and $\mathbf{y}(j+1)$ become $\mathbf{u}(s_k)$ and $\mathbf{y}(s_k)$). The eigenvalue $\lambda_{\max}(s_k)$ is of no direct interest to the computation and tends to infinity as the MDP algorithm approaches convergence.

The inner-loop eigenvectors $\mathbf{u}(s_k)$ and $\mathbf{y}(s_k)$ provide the directions that yield large norms for $\mathbf{G}(s_k) \cdot \mathbf{u}(s_k)$ and $\mathbf{G}^T(s_k) \cdot \mathbf{y}(s_k)$. These norms become the largest possible norms in the neighborhood of a pole of $\mathbf{G}(s)$ (see Section II). This is so because these inner-loop eigenvectors tend to the right and left singular vectors associated with σ_{\max} of $\mathbf{G}(s_k)$, in the neighborhood of a pole [41].

3. Use the converged inner-loop eigenvectors $\mathbf{u}(s_k)$ and $\mathbf{y}(s_k)$ to generate the associated right and left eigenvector estimates of the state matrix \mathbf{A} . To this end, solve for $\mathbf{x}(s_k)$ and $\mathbf{v}(s_k)$ using the matrix equations:

$$\begin{bmatrix} s_k \mathbf{I} - \mathbf{A} & -\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x}(s_k) \\ \mathbf{u}(s_k) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{u}(s_k) \end{bmatrix}$$

$$\begin{bmatrix} s_k \mathbf{I} - \mathbf{A}^T & \mathbf{C}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v}(s_k) \\ \mathbf{y}(s_k) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{y}(s_k) \end{bmatrix}$$

4. Compute the Rayleigh quotient for unsymmetric matrices [11] and use it as the next eigenvalue estimate s_{k+1} :

$$s_{k+1} = \frac{\mathbf{v}^T(s_k) \cdot \mathbf{A} \cdot \mathbf{x}(s_k)}{\mathbf{v}^T(s_k) \cdot \mathbf{x}(s_k)}$$

An eigenvalue deflation algorithm may be effectively employed at this step, prior to the Rayleigh quotient update (see comment further down in this section).

5. If the Euclidean norm of the mismatch vector ($\mathbf{A} \cdot \mathbf{x}(s_k) - s_{k+1} \cdot \mathbf{x}(s_k)$) is greater than the outer-loop convergence tolerance, increase the iteration counter ($k = k + 1$) and return to step 2. Otherwise the algorithm has converged to an eigenvalue which is a dominant pole in $\mathbf{G}(s)$, and also to the associated left and right eigenvectors of the matrix \mathbf{A} . The estimates (s_{k+1} , $\mathbf{x}(s_k)$, $\mathbf{v}(s_k)$) become the eigentriplet (λ_i , \mathbf{x}_i , \mathbf{v}_i), for $1 \leq i \leq n$, at the convergence of the MDP algorithm.

The following comments help further explain the MDP algorithm:

- The initial shift s_k is user-specified. The full set of dominant poles of a generic $\mathbf{G}(s)$ may however not be found by the MDP unless enough good quality shifts are provided. The inspection of the Sigma plots of $\mathbf{G}(s)$ may help providing good initial shifts (see Section VI).
- An eigenvalue deflation scheme for non-symmetric matrices may be employed at Step 4. This should be carried out prior to the Rayleigh Quotient update, enabling the MDP algorithm to more easily converge to the full set of dominant poles of $\mathbf{G}(s)$. A description on a deflation algorithm for non-symmetric power system matrices can be found in [35].
- The MDP algorithm utilizes the concept of system duality [1] in stages 2 and 3 to find the estimates for both left and right eigenvectors of \mathbf{A} . Note that the transpose transfer function $\mathbf{G}^T(s) = \mathbf{B}^T \cdot (s\mathbf{I} - \mathbf{A}^T)^{-1} \cdot \mathbf{C}^T$ has the same poles and zeros as $\mathbf{G}(s)$. Matrix \mathbf{A}^T is the state matrix of the dual system, and \mathbf{B}^T and \mathbf{C}^T its output and input matrices.
- There are no gains in performing the inner loop power iterations until convergence. Adopting a fixed number of power iterations (such as 10) at the first two outer-loop iterations ($k = 1$ and $k = 2$) is enough to ensure convergence to the dominant poles that are nearest to the given initial shift. Only one power iteration should be carried out at the remaining outer-loop iterations. Though not shown in the paper, the MDP algorithm (with only one inner-loop power iteration) is actually a neat computer implementation of the Newton method applied to the matrix eigenvalue problem: $[\mathbf{G}(s)]^{-1} \cdot \mathbf{u}(s) = \mathbf{0}$. The inner-loop power iterations are needed to improve the orientation of vectors $\mathbf{u}(s_1)$ and $\mathbf{y}(s_1)$, once the initial shift s_1 is specified. These power iterations can therefore be seen as a good initialization process for the Newton method.
- The MDP algorithm automatically reduces to the Dominant Pole Algorithm [20] when applied to scalar transfer functions. The two eigenvalue power methods (inner loops) are not needed in this case, reducing the Step 2 of the MDP algorithm to the computation of the scalar $g(s_k)$. Note that the Dominant Pole Algorithm was shown in [42] to be a neat computer implementation of the Newton method applied to the scalar equation: $g(s)^{-1} = 0$.

XIX.2 Model Approximations To $\mathbf{G}(s)$

An $m \times m$ transfer function $\mathbf{G}(s)$ of a n -th order system, having a complete set of eigenvectors, can be expressed as:

$$\mathbf{G}(s) = \sum_{i=1}^n \frac{\mathbf{R}_i}{s - \lambda_i} \quad (2)$$

The above expression is referred to as the dyadic expansion of $\mathbf{G}(s)$ [43], involving a sum of $m \times m$ residue matrices over a first-order pole [1], [43]. The symbol \mathbf{R}_i denotes the residue matrix for $\mathbf{G}(s)$ at the pole λ_i ; it may be computed once the eigentriplet $(\lambda_i, \mathbf{x}_i, \mathbf{v}_i)$ has been obtained:

$$\mathbf{u}(\lambda_i) = \mathbf{C} \cdot \mathbf{x}_i \quad (3)$$

$$\mathbf{y}(\lambda_i) = \mathbf{B} \cdot \mathbf{v}_i \quad (4)$$

$$\mathbf{R}_i = \mathbf{u}(\lambda_i) \cdot \mathbf{y}(\lambda_i)^T \quad (5)$$

Note the inner-loop eigenvectors $\mathbf{u}(s_k)$ and $\mathbf{y}(s_k)$ described in step 2 of the MDP algorithm become $\mathbf{u}(\lambda_i)$ and $\mathbf{y}(\lambda_i)$ at convergence. $\mathbf{G}(s)$ may be approximated by the sum of only the p terms, $p \ll n$, that have residue matrix norms $\|\mathbf{R}_i\|$ above some given tolerance. This truncated sum of p terms contains only those poles that determine the effective transfer function behavior.

The Sigma plot of the reduced-order model

$$\mathbf{G}(s) \approx \sum_{i=1}^p \frac{\mathbf{R}_i}{s - \lambda_i} \quad (6)$$

Should be evaluated over a sufficiently large number of ' ω ' values within the dynamic range of $\mathbf{G}(s)$. Note that the order of the reduced model is $p = 2nc + nr$, where nc is the number of complex-conjugate eigenvalue pairs and nr the number of real eigenvalues retained in the model.

Comparing the Sigma plots obtained for the complete and the reduced-order transfer functions may assess the quality of the model approximation. A residue matrix, \mathbf{R}_i , being the product of a column vector and a row vector, is of unity rank. The sum of m residue matrices associated with different system poles yields a matrix of rank m [44]. Therefore, at least m poles must be retained in the reduced-order model of a $m \times m$ transfer function in order to produce a nonzero $\sigma_{\min}(\omega)$ plot.

XIX.3 Description Of Test System

The test system analysed was the North-South Brazilian interconnection, having about 60,000 MW of generating capacity [44], [45]. The system model has 2,370 buses, 3,401 lines, 2,519 voltage dependent loads (active and reactive loads separately accounted), 123 synchronous machines, 122 excitation systems, 46 power system stabilizers, 99 speed-governors, 4 static var compensators, 2 TCSCs and 1 HVDC link. These equipment were detailed modeled, yielding a system Jacobian matrix with 13,062 lines and 48,501 nonzero elements. The system has a total of 1,676 state variables and information on the damping control of its interarea modes may be found in [44], [45].

Fig. 1 pictures the eigenvalue spectrum for this 1,676-state system, obtained by the QR routine from the EISPACK library [17]. The full eigensolution was obtained for the sake of illustration, and is not needed in the study described in this paper. The multivariable transfer function $\mathbf{G}(s)$ analyzed in this paper has eight inputs and eight outputs, as shown in Fig. 2. The inputs are the mechanical power changes (ΔP_{ref}) in eight power plants located at the northeastern region of the system. The outputs are the rotor speed changes ($\Delta\Omega$) at the same power plants. The four-digit superscripts, such as 5015, are the generator bus identification numbers. There is a large cluster of electromechanical eigenvalues in the 6 to 10 rad/s range, with damping ratios between 5 and 20% (see Fig. 1). Existing subspace iteration and Krylov subspace algorithms [21] are not suitable for computing the dominant spectrum of $\mathbf{G}(s)$.

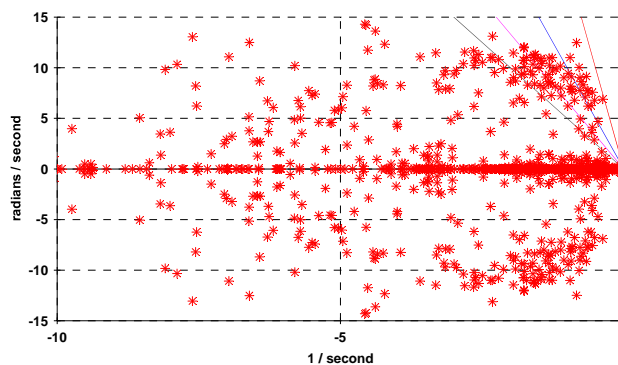


Fig. 1. Eigenvalue Spectrum of Brazilian Interconnected System.

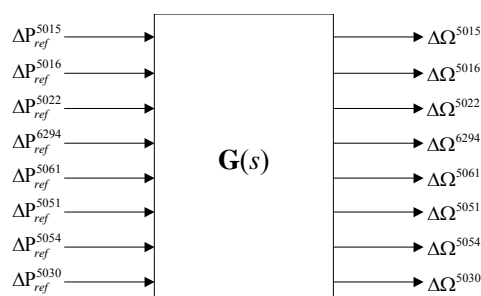


Fig. 2. Multivariable (8 x 8) Transfer Function.

XIX.4 Results

This section describes the results obtained with the MDP algorithm when solving for the dominant poles of the high-order (1,676 state variables) 8×8 transfer function matrix. The computation times provided in this paper refer to a personal computer with a 800 MHz, Pentium III processor with 128 MB of RAM. The convergence tolerance for the outer loop iterations was set at $1.0e-10$.

The torque-speed dynamics of practical power systems is confined to the 0.5 to 15 rad/s frequency range. Thirty equidistant initial shifts were therefore given within this frequency range and along the 15% damping ratio line. Their real and imaginary parts were $(-0.0759, -0.1517, \dots, -2.2757)$ and $(0.5j, 1j, \dots, 15j)$ respectively. The MDP algorithm produced 8 different eigenvalues (shown in Table 1) plus another 22 repeated eigensolutions (not shown).

This section depicts Sigma plots of $\mathbf{G}(s)$ for different damping ratios ($\xi = 0, \xi = 15\%$) and model approximations (16th-order, 20th-order and 41st-order). The plots for both the full model of $\mathbf{G}(s)$ and the reduced-order models are superimposed (there are, therefore, 4 curves per figure).

Fig. 3 compares the Sigma plots of the complete $\mathbf{G}(s)$ model (1,676 poles) and the 16th-order $\mathbf{G}(s)$ model having the poles shown in Table 1. Note that these plots do not show resonant peaks for frequencies close to most of the poles listed in Table 1, which have damping ratios of 15% and higher. These resonant peaks are however quite evident in the Sigma plots of Fig. 4, obtained by using complex frequencies with damping ratio of 15%. Note in Fig. 4 that the 16th-order model for $\mathbf{G}(s)$ yields a σ_{\max} plot that roughly matches the σ_{\max} plot ($\xi = 15\%$) for the full model. It is easily seen that two resonant peaks are missing in the $\sigma_{\max}(\omega)$ plot ($\xi = 15\%$) of the 16th-order model around the complex frequencies

$\omega_1 = -0.5462 + 3.60j$ and $\omega_2 = -1.4110 + 9.30j$ respectively. Table 2 shows the poles obtained by the MDP algorithm, when using ω_1 and ω_2 as initial shifts. Fig. 5 shows the improved matching of the Sigma plots obtained when incorporating these two additional complex pair of poles. The reduced model is now of 20th-order, and apart from capturing the two resonant peaks that were missing in Fig. 4 it yields a smaller discrepancy in the $\sigma_{\min}(\omega)$ plot for $\xi = 15\%$.

One may reason that the shape of the $\sigma_{\min}(\omega)$ plot is mainly determined by the location of the dominant zeros of $\mathbf{G}(s)$. The zeros of the 20th-order model are therefore not sufficiently close to the dominant zeros of $\mathbf{G}(s)$ for the full model. A higher-order reduced model (with additional poles) must then be used so as to reduce this discrepancy in the zeros. The authors do not yet have clear hints on how to choose the initial shifts for finding the other needed poles in applications involving large-scale systems.

Fig. 6 shows the Sigma plots ($\xi = 15\%$) for a 41st-order model of $\mathbf{G}(s)$, where another 21 poles (see Table 3) were added to the previous 20 in order to improve the matching of the $\sigma_{\min}(\omega)$ plot. Note in Fig. 6 that even with a relatively high number of retained poles, the 41st-order model does not show a good matching of the $\sigma_{\max}(\omega)$ plot, for frequencies above 10 rad/s.

The MDP algorithm has always converged in the numerous tests performed by the authors. Initial shifts of large moduli such as $10^5, 10^8, 10^5j$ and 10^8j yielded convergence to dominant poles of $\mathbf{G}(s)$ in 15, 13, 12 and 13 iterations respectively. The MDP algorithm has therefore global convergence characteristics. The term ‘globally convergent’ is used here to mean ‘convergent to a solution from any start point’, if a solution exists [21].

Table 1. Dominant Poles of $G(s)$ Obtained from Set of Equidistant Initial Shifts (repeated solutions not shown).

Results for the MDP Algorithm	
Initial Shift s	Converged Eigenvalues
-0.0759 +0.5j	-0.1158 +0.2445j (12)
-0.1517 +1.0j	-0.3179 +1.0437j (6)
-0.4551 +3.0j	-0.5199 +2.8814j (6)
-0.9103 +6.0j	-1.2098 +8.1765j (7)
-1.2137 +8.0j	-1.1129 +8.0075j (4)
-1.2896 +8.5j	-1.2902 +8.5407j (6)
-1.3654 +9.0j	-1.4778 +8.2550j (6)
-2.2757 +15.0j	-2.8323 +10.3949j (6)

Note: The number of iterations required for a convergence tolerance of $1.0e-10$ are given within parenthesis in tables 1 e 2.

Table 2. Dominant Poles of $G(s)$ Obtained from Shifts whose Location were Inferred from the Visual Inspection of Fig. 4.

Results for the MDP Algorithm	
Initial Shift s	Converged Eigenvalues
-0.5462 +3.60j	-0.5668 +3.6139j (6)
-1.4110 +9.30j	-1.6060 +9.3092j (5)

Table 3. Other Dominant Poles of $G(s)$ Used in the 41st-Order Model

Additional Eigenvalues Obtained by the MDP Algorithm	
-1.8959	-9.3974
-1.9784	-9.4719
-0.7168 +0.1238j	-10.492
-0.3464 +0.5796j	-11.197
-1.0537 +0.8877j	-11.208
-0.9113 +7.6649j	-2.0088 +0.0418j
-2.9960 +9.3897j	-0.9113 +7.6649j

The CPU time taken to converge to one dominant pole of $G(s)$ is about 2.7 seconds, assuming an average number of 7 iterations per pole. The Sigma plot of the 8×8 $G(s)$ transfer function, ($0.1 \leq \omega \leq 15$, $\Delta\omega = 0.1$ rad/s) is obtained in 20 seconds. The total CPU time required to obtain the 16th-order model (plus residues) and obtain the Sigma plots shown in Fig. 5 is therefore about 110 seconds. After a few minutes of analysis, focused on Fig. 4, and a few more seconds of computation the 20th-order model may be produced. Obtaining a minimum set of additional poles (such as those shown in Table 3), to refine the reduced order model, required however several hours of careful analysis and computer-aided work by the authors. Further work is therefore needed regarding reduced model refinement based on the method described in this paper.

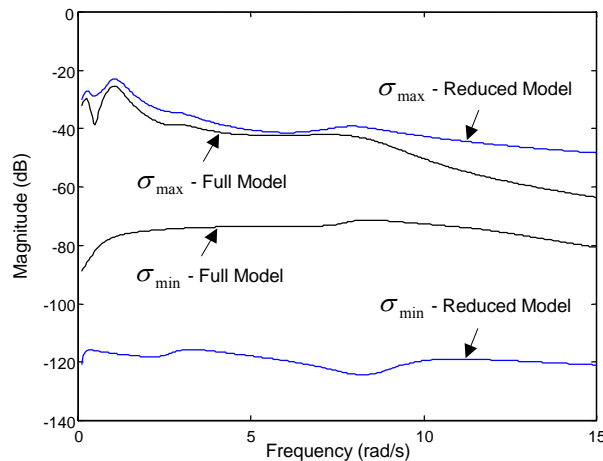


Fig. 3. Sigma-plot for 8×8 $G(s)$, $\xi = 0$ (reduced model has order 16).

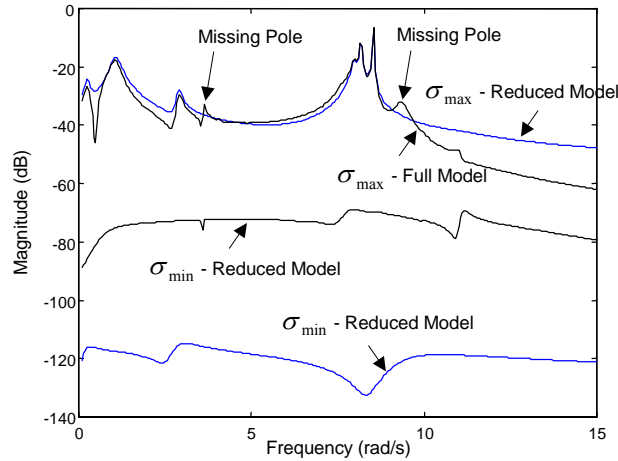


Fig. 4. Sigma-plot for $8 \times 8 \mathbf{G}(s)$, $\xi = 15\%$ (reduced model has order 16).

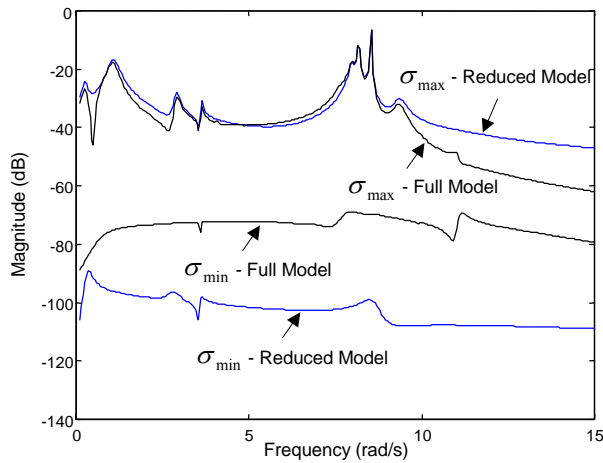


Fig. 5. Sigma-plot for $8 \times 8 \mathbf{G}(s)$, $\xi = 15\%$ (reduced model has order 20).

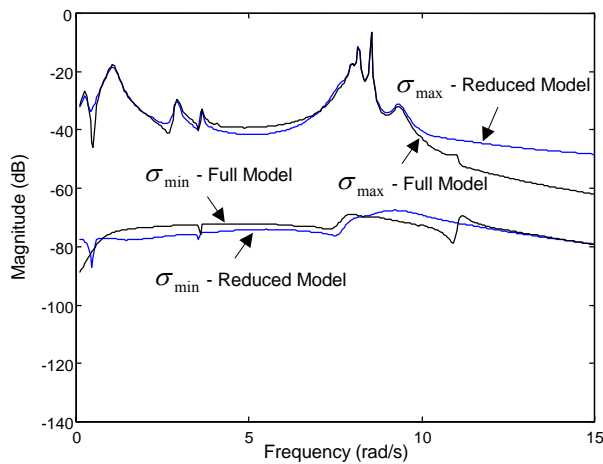


Fig. 6. Sigma-plot for $8 \times 8 \mathbf{G}(s)$, $\xi = 15\%$ (reduced model has order 41).

The reduced models for the 64 scalar transfer functions that constitute the 8×8 matrix $\mathbf{G}(s)$ must yield time and frequency responses that match those of the full model. The step response for a reduced-order, scalar transfer function model may be analytically computed, as described in [30]. A numerical integration algorithm may compute the step response for the scalar transfer function associated with the full model. The step responses for twelve of the 64 scalar transfer functions are depicted in Fig. 7. Note that the 41st-order model of the scalar transfer function $g_{11}(s)$ matches rather nicely the response of the 1,676th-order model. The same could not however be said about the $g_{33}(s)$ and a few other scalar transfer functions, where significant discrepancies exist

in the steady-state values. Note however, that the major system oscillations are all captured by the reduced-models of the twelve transfer functions.

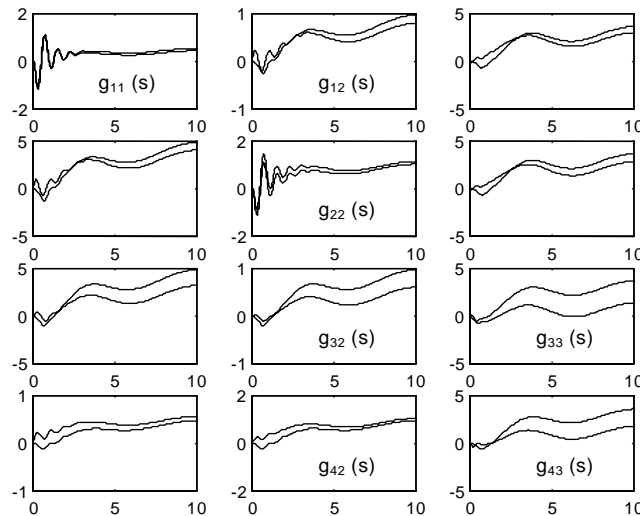


Fig. 7. Step responses for $g_{ij}(s)$ scalar transfer functions for the full model and the 41st-order model.
Note: The vertical axes in the 12 plots are given in radians/s and the horizontal axes in seconds.

XIX.5 Conclusions

This paper's description of the MDP algorithm should be adequate for computer implementation by an experienced programmer. The results obtained confirmed the algorithm effectiveness in dealing with multivariable transfer functions of large-scale power system models. Convergence is achieved in a reduced number of iterations (usually below 10). The MDP algorithm shows quadratic convergence in the neighborhood of the solution, is numerically stable, globally convergent and computationally efficient. The MDP is a full Newton algorithm with a very good initialization process and a simple computer implementation.

An eigenvalue deflation scheme was effectively employed to avoid repeated convergence to dominant poles that had previously been found [35], [46]. It was used to find the eigenvalues of Table 3 and will be reported in a later publication.

The MDP algorithm allows obtaining reduced-order models for multivariable transfer functions of large-scale systems. The reduced-order models of the 8×8 $\mathbf{G}(s)$ described in the paper were validated through time and frequency response tests. Utilizing Sigma plots to validate reduced-order models was found to be both neat and effective. Sigma plots alone do not however provide sufficient information for reduced-order model refinement.

The use of the concept of transfer function dominance has proved valuable to power system eigenanalysis [5], [20], [47], [30]. It is believed that several other areas of engineering will eventually be using the same concept and improving on the algorithms. Model reduction of large linear system models [30], selective eigenanalysis [48] and root locus plots [47] are some of the current applications of these algorithms.

XX. S-MATRIX, CAYLEY AND OTHER SPECTRAL TRANSFORMS [31,34,35,36] (to be written)

XXI. PARTIAL POLE LOCATION (to be written)

REFERENCES

- [1] T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1980;

- [2] **K. Ogata**, *Modern Control Engineering*, 2nd Edition, Prentice-Hall, Englewood Cliffs, NJ, 1990;
- [3] **J.M. Maciejowski**, *Multivariable Feedback Design*, New York, Addison Wesley, 1989;
- [4] **P. Kundur**, *Power System Stability and Control*, McGraw-Hill, 1994;
- [5] **P. Kundur, G.J. Rogers, D. Y. Wong, L. Wang & M. G. Lauby**, "A Comprehensive Computer Program Package for Small Signal Stability Analysis of Power Systems", *IEEE Trans. On Power Systems*, **PWRS-5**, pp. 1076-1083, 1990;
- [6] **L. Rouco, I.J. Pérez-Arriaga, R. Criado and J. Soto**, "A Computer Package for Analysis of Small Signal Stability in Large Electric Power Systems", *PSSC, 11th Power Systems Computation Conference, Proceedings*, Avignon, France, August 1993, pp. 1141-1148;
- [7] **N. Martins, L.T.G. Lima, H.J.C.P. Pinto and N.J.P. Macedo**, "The Brazilian Utilities Package for the Analysis and Control of Small-Signal Stability of Large Scale AC/DC Power Systems", in *Proceedings of III Symposium of Specialists in Electrical Operational and Expansion Planning*, Belo Horizonte, Brazil, May 1992;
- [8] **N. Martins**, "Efficient Eigenvalue and Frequency Response Methods Applied to Power System Small-Signal Stability Studies", *IEEE Trans. On Power Systems*, Vol. **PWRS-1**, pp. 217-226, February 1986;
- [9] **G.H. Golub, C.F. VanLoan**, *Matrix Computations*, Second Edition, The John Hopkins University Press, USA, 1989;
- [10] **B. Parlet**, *The Symmetric Eigenvalue Problems*, Prentice-Hall, New Jersey, 1973;
- [11] **J.H. Wilkinson**, "The Algebraic Eigenvalue Problem", Clarendon Press, Oxford, 1965;
- [12] **V. Arcidiacono, E. Ferrari, R. Marconato, J. Dos Ghali e D. Grandez**, "Evaluation and Improvement of Electromechanical Oscillation Damping by Means of Eigenvalue-Eigenvector Analysis. Practical Results in the Central Peru Power System", *IEEE Trans. on Power Apparatus and Systems*, Vol. **PAS-99**, pp. 769-778, March/April 1980;
- [13] **N. Martins & L.T.G. Lima**, "Determination of Suitable Locations for Power System Stabilizers and Static VAR Compensators for Damping Electromechanical Oscillations in Large Scale Power Systems", *IEEE Trans. on Power Systems*, Vol. **PWRS-5**, pp. 1455-1469, November 1990;
- [14] **J.E. Van Ness, J.M. Boyle and F.P. Imad**, "Sensitivities of Large Multiple-Loop Systems", *IEEE Trans. on Automatic Control*, **AC-10**, pp.308-318, 1965;
- [15] **T. Smed**, "Feasible Eigenvalue Sensitivity for Large Power Systems", presented at *1992 IEEE/PES Winter Meeting*, paper 92 WM 171-9 **PWRS**, New York, January 1992;
- [16] **N. Martins, H.J.C.P. Pinto and L.T.G. Lima**, "Efficient Methods for Finding Transfer Function Zeros of Power Systems", *Proceedings of 1991 IEEE Power Industry Computer Application Conference*, pp. 320-328, May 1991;
- [17] **B.T. Smith, J.M. Boyle, J. Dongarra, B. Garbow, Y. Ikebe, V.C. Klene & C.B. Moler**, *Matrix Eigensystem Routines: EISPACK Guide*, 2nd Edition, Springer-Verlag, New York, 1976;
- [18] **R.V. Patel, A.J. Laub, P.M. VanDooren**, *Numerical Linear Algebra Techniques for Systems and Control*, IEEE Press, 1994;
- [19] **R.T. Byerly, R.J. Bennon and D.E. Sherman**, "Eigenvalue Analysis of Synchronizing Power Flow Oscillations in Large Electric Power Systems", *IEEE PICA Conf.*, pp. 134-142, 1981;
- [20] **N. Martins, L.T.G. Lima, H.J.C.P. Pinto**, "Computing Dominant Poles of Very High Order Transfer Functions", *IEEE Trans. on Power Systems*, Vol. 11, No. 1, pp. 162-170, February 1996;

- [21] **A. Jennings and J.J. McKeown**, *Matrix Computation*, 2nd Edition, John Willey & Sons, UK, 1993;
- [22] **A. Jennings and W.J. Stewart**, “Simultaneous Iteration for Partial Eigensolution of Real Matrices”, *J. Inst. Maths. Applics.*, Vol. 15:351-361, 1975;
- [23] **W.J. Stewart and A. Jennings**, “A Simultaneous Iteration Algorithm for Real Matrices”, *ACM Trans. on Mathematical Software*, Vol. 7, No.2:184-198, June 1981;
- [24] **L. Wang and A. Semlyen**, “Application of Sparse Eigenvalues Techniques to the Small-Signal Stability Analysis of Large Power Systems”, *IEEE Trans. on Power Systems*, Vol. PWR5-6, No. 6, pp. 635-642, May 1990;
- [25] **L. Wang**, *Eigenvalue Analysis of Large Power Systems*, Ph.D. Thesis, University of Toronto, 1991;
- [26] **B. Gao, G.K. Morison and P. Kundur**, “Voltage Stability Evaluation Using Modal Analysis”, *IEEE Trans. on Power Systems*, Vol. 7, No. 4:1529-1542, November 1992;
- [27] **G. Angelidis and A. Semlyen**, “Efficient Calculation of Critical Eigenvalue Clusters in the Small-Signal Stability Analysis of Large Power Systems”, *Paper 94 SM 059 PWR5*, IEEE/PES Winter Meeting, San Francisco, CA, July 1994.
- [28] **G.J. Rogers**, “Methods for Small Signal Analysis of Very Large Power Systems”, *Proceedings of the 26th Conference on Decision and Control*, Los Angeles, pp.393-398, December 1987;
- [29] **J. M. Campagnolo, N. Martins, D. M. Falcão**, “Refactored Bi-Iteration: A High Performance Eigensolution Method for Large Power System Matrices” , *IEEE Trans. on Power Systems*, Vol. 11, No.3, pp. 1228-1235, August 1996;
- [30] **N. Martins**, “The Dominant Pole Spectrum Eigensolver” , *IEEE Trans. on Power Systems*, Vol. 12, No.1, pp. 245-254, February 1997;
- [31] **N. Uchida & T. Nagao**, “A New Eigen-Analysis Method of Steady-State Stability Studies for Large Power Systems: S Matrix Method”. *IEEE Trans. On Power Systems*, Vol. PWR5-3, pp. 706-714, May 1988;
- [32] **L. Rouco and F.L. Pagola**, “An Eigenvalue Sensitivity Approach to Location and Controller Design of Controllable Series Capacitors for Damping Power System Oscillations”, *IEEE Trans. on Power Systems*, Vol. 12, No. 4, pp.1660-1666, November 1997;
- [33] **F.L. Pagola, I.J. Pérez-Arriga and G.C. Verghese**, “On Sensitivities, Residues and Participations”, *IEEE Trans. on Power Systems*, Vol. PWR5-5, No. 1, pp.278-285, February 1989;
- [34] **L.T.G. Lima, L.H. Bezerra, C. Tomei and N. Martins**, “New Methods for Fast Small-Signal Stability Assessment of Large Scale Power Systems”, *IEEE Trans. on Power Systems*, Vol. 10, No.4, pp. 1979-1985, November 1995;
- [35] **G. Angelidis and A. Semlyen**, “Improved Methodologies for the Calculation of Critical Eigenvalues in Small-Signal Stability Analysis”, *Paper 95 SM 505-8 PWR5*, presented at the *IEEE/PES Summer Meeting*, Portland, OR, July 1995;
- [36] **K. Meerbergen, A. Spence and D. Roose**, “Shift-Invert and Cayley Transforms for the Detection of Right-Most Eigenvalues on Nonsymmetric Matrices”, *Revised Report TW 200*, Department of Computing Science, Katholieke Universiteit Leuven, Belgium, February 1994;
- [37] **David G. Luenberger**, “Dynamic Equations in Descriptor Form”, *IEEE Trans. on Automatic Control*, Vol. 22, No. 3, June 1977, pp. 312-321.

- [38] **N. Martins and P. E. M. Quintão**. “Computing Dominant Poles of Power System Multivariable Transfer Functions”. IEEE Transactions on Power Systems, USA, v. 18, n. 01, p. 152-159, 2003.
- [39] G. Peters, J.H. Wilkinson, “Inverse Iteration, Ill-Conditioned Equations and Newton’s Method”, SIAM Review, Vol.21, No. 3, July 1979.
- [40] L.H. Bezerra, Written Discussion to Reference [20].
- [41] D.H. Owens, “Dyadic Expansion for the Analysis of Linear Multivariable Systems”, Proc. IEE, Vol. 121, No. 7, 1974, pp. 713-716.
- [42] R.V Patel, N. Munro, “Multivariable System Theory and Design”, Pergamon Press, 1982.
- [43] N. Martins, A.A. Barbosa, J.C.R. Ferraz, M.G. dos Santos, A.L.B. Bergamo, C.S. Yung, V.R. Oliveira, N.J.P. Macedo, “Retuning Stabilizers for the North-South Brazilian Interconnection”, Panel Session on System Reliability as Affected by Power System Stabilizers, IEEE/PES Summer Meeting, Edmonton, Canada, July 1999.
- [44] Y. Saad, “Projection and Deflation Methods for Partial Pole Assignment in Linear State Feedback”, IEEE Trans. on Automatic Control, Vol. 33, No. 3, March 1998, pp. 290-297.
- [45] **CIGRE TF 38.02.16 Report**, Impact of the Interactions among Power System Controls, Report 166, CIGRE Paris, July 2000.
- [46] F.L. Pagola, L. Rouco, I.J. Pérez-Arriaga, “Analysis and Control of Small-Signal Stability in Electric Power System by Selective Modal Analysis”. Eigenvalue and Frequency Domain Methods for System Dynamic Performance, pp. 77-96, IEEE Publication No. 90TH0292-3-PWR, New York, 1990.