Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

Residual algorithm for large-scale positive definite generalized eigenvalue problems

Lenys Bello, William La Cruz y Marcos Raydan

RT 2008-08

Centro de Cálculo Científico y Tecnológico de la UCV
CCCT-UCV
Caracas, Noviembre, 2008.

Residual algorithm for large-scale positive definite generalized eigenvalue problems

Lenys Bello* William La Cruz† Marcos Raydan‡

November 20, 2008

Abstract

In the positive definite case, the extreme generalized eigenvalues can be obtained by solving a suitable nonlinear system of equations. In this work, we adapt and study the use of recently developed low-cost derivative-free residual schemes for nonlinear systems, to solve large-scale generalized eigenvalue problems. We demonstrate the effectiveness of our approach on some standard test problems, and also on a problem associated with the vibration analysis of large structures. In our numerical results we use preconditioning strategies based on incomplete factorizations, and we compare with and without preconditioning with a well-known available package.

Key words: Generalized eigenvalues; Generalized Rayleigh quotient; Spectral gradient method; DF-SANE residual method.

1 Introduction

We are interested in the generalized eigenvalue problem

$$Ax = \mu Bx, \tag{1}$$

where A and B are $n \times n$ Symmetric and Positive Definite (SPD) matrices, $x \in \mathbb{R}^n$, n is large, and $\mu \in \mathbb{R}$. The values of μ that satisfy (1) are the generalized eigenvalues and the corresponding vectors x are the generalized eigenvectors. In most applications, it is only required to compute a few smallest eigenvalues, and their corresponding eigenvectors, see e.g. [1, 8, 11, 15, 20].

For solving problem (1) several schemes have been proposed, including factorization techniques for small size problems (see [10] and references therein) and iterative schemes, for large-scale problems, based on Krylov subspace methods (see [11] and references therein).

Recently, for large-scale problems, an optimization algorithm that combines the Spectral Projected Gradient (SPG) method [4, 5] with preconditioning strategies [2] was introduced and analyzed

*Centro Multidisciplinario de Visualización y Cómputo Científico, FACYT, Universidad de Carabobo, Valencia, Venezuela (lcbello@uc.edu.ve).

†Dpto. de Electrónica, Computación y Control, Escuela de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad Central de Venezuela, Caracas 1051-DF, Venezuela (william.lacruz@ucv.ve). This author was supported by CDCH project PI-08-14-5463-2006.

‡Departamento de Cómputo Científico y Estadística, Universidad Simón Bolívar (USB), Ap. 89000, Caracas 1080-A, Venezuela (marcos.raydan@ciens.ucv.ve). This author was supported by USB and the Scientific Computing Center at UCV.

in [3] for minimizing the generalized Rayleigh quotient. However, as we will discuss in our next section, this novel approach requires the projection onto the ellipsoids, defined by the matrix B, which involves inner iterations at each outer SPG iteration. In this work, motivated by the approach developed in [3], we present a suitable residual scheme that avoids the inner iterations, and for which we propose to use a combined and adapted version of the SANE [17] and DF-SANE [16] algorithms that have proved to be convenient for large-scale nonlinear systems of equations.

2 New Algorithm and Convergence Analysis

Let us recall the generalized Rayleigh quotient, associated with A and B, for a given $x \neq 0$

$$r(x) = \frac{x^T Ax}{x^T Bx}. \tag{2}$$

We observe that the generalized Rayleigh quotient is a continuously differentiable map $r : \mathbb{R}^n \mapsto \mathbb{R}$ for all $x \neq 0$, whose gradient is given by

$$\nabla r(x) = \frac{2}{x^T Bx}(Ax - r(x)Bx). \tag{3}$$

It is clear that any eigenvector x and its associated eigenvalue μ satisfy that $r(x) = \mu$, and hence in that case x is a stationary point of r, i.e., $\nabla r(x) = 0$. Therefore, (1) can be solved using optimization techniques. For a review of the optimization approach for solving (1) see e.g. [21] and references therein. In particular, a vast literature can be found on gradient related methods for the SPD eigenvalue problem (see, e.g., [6, 7, 13, 18, 19]).

More recently, for large-scale problems, the preconditioned SPG method [2] was applied in [3] for minimizing the quadratic form $x^T Ax$ subject to the convex set

$$\Omega = \{x \in \mathbb{R} : x^T Bx \leq 1\},$$

that yields the eigenvector associated with the smallest eigenvalue of (1). For computing the projection onto the ellipsoid Ω, at every SPG iteration, the iterative schemes recently developed by Dai [9] were used in [3].

A simple and key observation at this point is that, since B is SPD, then $\nabla r(x) = 0$ if and only if x is a solution of the following nonlinear system of equations

$$F(x) \equiv Ax - r(x)Bx = 0. \tag{4}$$

Motivated by this observation, our approach for solving (1) consist in solving the nonlinear system of equations (4). For that, we propose a variant of the low-cost methods SANE [17] and the derivative free DF-SANE [16] for solving (4). This combined and adapted variant for solving (1) avoids the projection onto Ω, and will be denoted as `saeig`.

The methods SANE and DF-SANE use in a systematic way the residual $\pm F(x_k)$ as a search direction combined with a nonmonotone line search globalization strategy. Since A and B are SPD, then the `saeig` algorithm only uses the residual $-F(x_k) = -(Ax_k - r(x_k)Bx_k)$ as a descent search direction. Initially, `saeig` takes $x_0 \in \mathbb{R}^n$ and generates the iterates

$$x_{k+1} = x_k + \lambda d_k,$$

in which the steplength $\lambda \in (0, 1]$ and the direction $d_k = -\alpha_k F(x_k)$ satisfy the following inequality:

$$r(x_{k+1}) \leq r(x_k) + \eta_k - \gamma \lambda^2 \|d_k\|^2, \tag{5}$$

where $\alpha_k > 0$ is the spectral coefficient to be described later, $\gamma \in (0,1)$, $\{\eta_k\}$ is a positive sequence such that

$$\sum_{k=0}^{\infty} \eta_k \leq \eta < \infty, \tag{6}$$

and $\|\cdot\|$ denotes the Euclidian norm. Throughout this work $\mathbb{N}$ is the set of natural numbers. Algorithm 1 below is a formal description of the `saeig` method.

Algorithm 1 (`saeig` method)**.**

Step 0. Choose $x_0 \in \mathbb{R}^n$, $x_0 \neq 0$, $0 < \alpha_{min} < \alpha_{max} < \infty$, $0 < \sigma_{min} < \sigma_{max} < 1$, $0 < \gamma < 1$, and a positive sequence $\{\eta_k\}$ that satisfies (6). Set $k := 0$.

Step 1. If $F(x_k) = 0$ stop the process.

Step 2. Choose α_k such that $\alpha_k \in [\alpha_{min}, \alpha_{max}]$.

Step 3. Set $d := -\alpha_k F(x_k)$.

Step 4. Set $\lambda := 1$.

Step 5. If $r(x_k + \lambda d) \leq r(x_k) + \eta_k - \gamma\lambda^2\|d\|^2$, set $d_k = d$, and go to Step 7.

Step 6. Choose $\sigma \in [\sigma_{min}, \sigma_{max}]$, set $\lambda := \sigma\lambda$, and go to Step 5. (*Backtracking process*)

Step 7. Set $\lambda_k = \lambda$, $x_{k+1} = x_k + \lambda_k d_k$, $k := k + 1$, and go to Step 1.

Remark 2.1.

(i) Algorithm 1 is well defined. Indeed, by the continuity of r and since $\eta_k > 0$, the condition (5) is satisfied after a finite number of reductions of λ (backtrackings).

(ii) $d_k = -(\alpha_k/2)(x_k^T B x_k)\nabla r(x_k)$, for all $k \geq 0$.

(iii) Since A and B are SPD matrices, and $\alpha_k > 0$, we obtain

$$\nabla r(x_k)^T d_k = -(\alpha_k/2)(x_k^T B x_k)\|\nabla r(x_k)\|^2 < 0.$$

In other words, for all k, $d_k = -\alpha_k F(x_k)$ is a descent direction for r at x_k.

(iv) The preconditioned version of `saeig` method consists in building $d = -\alpha_k M^{-1} F(x_k)$ in Step 3, where M is a suitable given SPD matrix of order n.

The following proposition shows that the sequence $\{x_k\}$ generated by Algorithm 1 is contained in a certain closed and bounded set.

Proposition 2.1. *The sequence $\{x_k\}$ generated by Algorithm 1 is contained in the set*

$$\Omega_0 = \{x \in \mathbb{R}^n : r(x) \leq r(x_0) + \eta\}.$$

Proof. By (5) and (6) we can write

$$\begin{aligned} r(x_{k+1}) &\leq r(x_k) + \eta_k \\ &\leq r(x_{k-1}) + \eta_{k-1} + \eta_k \\ &\leq \cdots \leq r(x_0) + \sum_{j=0}^{k} \eta_j \leq r(x_0) + \eta. \end{aligned}$$

Therefore, the sequence $\{x_k\}$ is contained in Ω_0. □

Proposition 2.2. *Let $\{x_k\}$ be the sequence generated by Algorithm 1. Then*

$$\lim_{k\to\infty} \lambda_k \|d_k\| = 0. \tag{7}$$

Proof. By (5) we have that

$$\lambda_k^2 \|d_k\|^2 \leq \frac{\eta_k}{\gamma} + \frac{r(x_k) - r(x_{k+1})}{\gamma}, \text{ for all } k \geq 0. \tag{8}$$

Since η_k satisfies (6), adding all terms in both sides of (8) it follows that

$$\sum_{k=0}^{\infty} \lambda_k^2 \|d_k\|^2 \leq \frac{\eta + r(x_0)}{\gamma} < \infty,$$

and the result follows. □

The theorem below shows that all limit points of the sequence $\{x_k\}$ generated by Algorithm 1 are stationary points of r.

Theorem 2.1. *Let $\{x_k\}$ be the sequence generated by Algorithm 1. Then*

$$\lim_{k\to\infty} \nabla r(x_k) = 0. \tag{9}$$

Proof. Let x_* be a limit point of $\{x_k\}$. Without loss of generality we can assume that the sequence $\{x_k\}$ converges to x_*. The equation (7) holds if

$$\lim_{k\to\infty} \|d_k\| = 0. \tag{10}$$

or if

$$\liminf_{k\to\infty} \lambda_k = 0. \tag{11}$$

Since $d_k = -(\alpha_k/2)(x_k^T B x_k)\nabla r(x_k)$, $\alpha_k > 0$, and B is an SPD matrix, then by (10) the result holds. On the other hand, if (11) holds there exists an infinite set of indices $K \subset \mathbb{N}$ such that

$$\lim_{k\to\infty, k\in K} \lambda_k = 0.$$

By the way λ_k was chosen in Step 6 of Algorithm 1, there exists an index $\overline{k}$ sufficiently large such that for all $k \geq \overline{k}$, $k \in K$, there exists σ_k ($0 < \sigma_{min} \leq \sigma_k \leq \sigma_{max}$) for which $\lambda = \lambda_k/\sigma_k$ fails to satisfy condition (5), i.e.,

$$\begin{aligned} r\left(x_k + (\lambda_k/\sigma_k)d_k\right) &> r(x_k) + \eta_k - \gamma(\lambda_k/\sigma_k)^2\|d_k\|^2 \\ &> r(x_k) - \gamma(\lambda_k/\sigma_k)^2\|d_k\|^2. \end{aligned}$$

Thus,

$$\frac{r\left(x_k + (\lambda_k/\sigma_k)d_k\right) - r(x_k)}{\lambda/\sigma_k} > -\gamma(\lambda_k/\sigma_k)\|d_k\|^2 > -\gamma(\lambda_k/\sigma_{min})\|d_k\|^2.$$

By the Mean Value Theorem we obtain

$$\nabla r(x_k + t_k d_k)^T d_k > -(\lambda_k/\sigma_{min})\|d_k\|^2, \text{ for all } k \geq \overline{k}, k \in K, \tag{12}$$

where $t_k \in [0, \lambda_k/\sigma_k]$ and $\lim_{k\to\infty, k\in K} t_k = 0$.

Now, since

$$\nabla r(x_k)^T d_k = -(\alpha_k/2)(x_k^T B x_k)\|\nabla r(x_k)\|^2 < 0, \quad \text{for } k \geq 0,$$

taking limits in (12) as $k \to \infty$, $k \in K$, we obtain that $\nabla r(x_*) = 0$. This completes the proof. □

3 Implementation and Numerical Results

3.1 Implementation details

We implemented Algorithm `saeig` with the following parameters: $\alpha_{min} = 10^{-10}$, $\alpha_{max} = 10^{10}$, $\alpha_0 = 1$, $\sigma_{min} = 0.1$, $\sigma_{max} = 0.5$, $\gamma = 10^{-4}$, $\eta_k = \theta(1 - 10^{-6})^k$, where

$$\theta = \begin{cases} \|F(x_0)\|^2, & \text{if } \|F(x_0)\|^2 \leq 10^8; \\ 10^8 & \text{if } \|F(x_0)\|^2 > 10^8. \end{cases}$$

The spectral steplength was computed by the formula

$$\alpha_k = \frac{s_k^T s_k}{s_k^T y_k},$$

where $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$ (see [17] for details). However, if $\alpha_k \notin [\alpha_{min}, \alpha_{max}]$, we replace the spectral coefficient by

$$\alpha_k = \begin{cases} 1, & \text{if } \|F(x_k)\| > 1; \\ \|F(x_k)\|^{-1}, & \text{if } 10^{-5} \leq \|F(x_k)\| \leq 1; \\ 10^5, & \text{if } \|F(x_k)\| < 10^{-5}. \end{cases}$$

For choosing $\sigma \in [\sigma_{min}, \sigma_{max}]$ at Step 6 of `saeig`, we proceed as follows. Given $\lambda > 0$, we set

$$\sigma = \begin{cases} \sigma_{min}, & \text{if } \lambda_c < \sigma_{min}\lambda; \\ \sigma_{max}, & \text{if } \lambda_c > \sigma_{max}\lambda; \\ \lambda_c/\lambda, & \text{otherwise,} \end{cases}$$

where

$$\lambda_c = \frac{-\lambda^2 \|d_k\|^2}{2(r(x_k + \lambda d_k) - r(x_k) - \lambda^2 \|d_k\|^2)}.$$

We stopped the process when

$$\frac{\|F(x_k)\|}{\|x_k\|} \leq tol, \tag{13}$$

where $tol \in (0, 1)$.

3.2 Numerical results

We compare the numerical behavior of `saeig`, with and without preconditioning, and the function `eigs` from MATLAB, over a set of test problems. For all experiments we use the incomplete LU factorization of the matrix A as a preconditioner for `saeig`, using the drop tolerance value 10^{-6} (i.e., we use the MATLAB command `luinc(A,1.0e-6)`). All the runs were carried out using MATLAB version 6.0 on an Intel Centrino Duo computer at 1.8 GHz with 1GB of RAM.

We begin by solving a generalized eigenvalue problem associated with the vibration analysis of large structures [11]. For these problems, the smallest eigenvalues correspond to the natural frequencies of low mode of vibration, and so they are important to study the behavior of the structure [14]. We consider the stiffness and mass matrices A and B respectively, associated with a spring system with n masses that is shown in Figure 1.

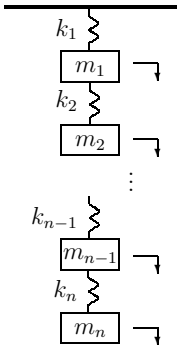

Figure 1: A spring system with n masses.

The matrices A and B are of the form:

$$A = \begin{pmatrix} k_1 + k_2 & -k_2 & & \\ -k_2 & k_2 + k_3 & -k_3 & \\ & \ddots & \ddots & \\ & -k_{n-1} & k_{n-1} + k_n & -k_n \\ & & -k_n & k_n \end{pmatrix}, \quad B = \begin{pmatrix} m_1 & & & \\ & m_2 & & \\ & & \ddots & \\ & & & m_n \end{pmatrix}.$$

For our numerical experiments we use $k_i = 10000\,i$, and $m_i = 20000\,i$, for $i = 1, 2, \ldots, n$. We randomly generate x_0 (by `rand(n,1)` in MATLAB) and we set $tol = 5 \times 10^{-8}$ in (13).

Figure 2 shows the behavior of `saeig` with and without preconditioning, for a spring system with n masses. Table 1 presents the results for the spring system for different values of n, where we report the number of iterations (IT), the number of evaluations of $F(x)$ (EF), the CPU time in seconds (T), and the residual

$$\overline{e} = \|A\overline{x} - \overline{\mu}B\overline{x}\|_1, \tag{14}$$

where $\overline{x}$ is the eigenvector and $\overline{\mu}$ its associated eigenvalue obtained by the algorithm. Also, in this table, we report the smallest eigenvalue $\mu_\star$ (computed by `eigs`).

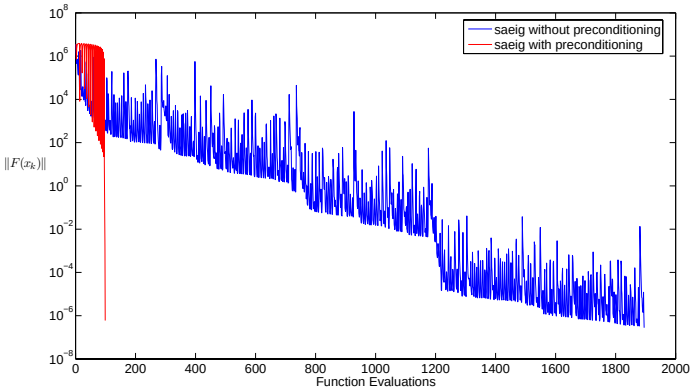

Figure 2: Behavior of `saeig` with and without preconditioning for a spring system with 100 masses.

For our second experiment, we consider the set of test matrices A and B taken from the Harwell-Boeing collection [12], as listed in Table 2. We choose the initial iterate x_0 as a randomly generated number, and we set $tol = 5 \times 10^{-9}$ in (13).

Table 1: Results for a spring system with n masses.

	`saeig` without preconditioning					`saeig` with preconditioning					
n	IT	EF	T	$\overline{e}$	$\overline{\mu}$	IT	EF	T	$\overline{e}$	$\overline{\mu}$	$\mu_\star$
100	1894	1894	0.20	2.4e-6	2.2e-5	21	98	0.00	4.2e-6	2.2e-5	2.2e-5
250	5764	5766	0.80	6.7e-6	3.0e-6	77	570	0.11	8.7e-6	3.0e-6	3.0e-6
500	21762	21773	4.41	2.1e-5	6.6e-7	217	1990	0.44	1.7e-4	6.6e-7	6.6e-7
1000	191584	191623	282.59	1.7e-4	1.5e-7	663	7239	3.16	8.4e-3	1.5e-7	1.5e-7

Table 2: Harwell-Boeing test matrices

No.	Matrix A	Matrix B	Size
1	bcsstk04	bcsstm04	132
2	bcsstk05	bcsstm05	153
3	bcsstk06	bcsstm06	420
4	bcsstk07	bcsstm07	420
5	bcsstk08	bcsstm08	1074
6	bcsstk09	bcsstm09	1083
7	bcsstk10	bcsstm10	1086
8	bcsstk11	bcsstm11	1473
9	bcsstk12	bcsstm12	1473
10	bcsstk13	bcsstm13	2003
11	bcsstk25	bcsstm25	15439
12	bcsstk14	I	1806
13	bcsstk15	I	3948
14	bcsstk16	I	4884
15	bcsstk17	I	10974
16	bcsstk18	I	11948

Table 3 presents the results, for problem (1) using the test matrices A and B from Table 2, where we compare the preconditioned version of `saeig` with `eigs(A,B,1,'SA')`, with `tol` $= 5 \times 10^{-9}$, for computing the algebraically smallest eigenvalue. We report the number of matrix-vector multiplications (MV), the CPU time, the residual $\overline{e}$ given by (14) and the eigenvalue $\overline{\mu}$ obtained by the algorithm. The CPU time was obtained with the on-screen outputs suppressed. The symbol "err" that appears in Problems 1, 7 and 10 indicates that `eigs` produced the following MATLAB error message: "Generalized matrix B must be the same size as A and either a symmetric positive (semi) definite matrix or its Cholesky factor".

We observe that the CPU time and the number of matrix-vector multiplications of `saeig` are significantly smaller than those of `eigs`. However, the errors obtained by `eigs` are generally smaller than those obtained by `saeig`. Nevertheless, the errors of `saeig` can be reduced by reducing the tolerance tol in (13). For example in Problem 10 with $tol = 10^{-9}$, `saeig` converges to the eigenvalue $\overline{\mu} = 1475.34074596$ with residual $\overline{e} = 5 \times 10^{-1}$, using 424 matrix-vector multiplications in 8.56 seconds.

The MATLAB function `eigs` has not been designed to take advantage of preconditioning strategies. Therefore, in our third and final experiment we consider the medium size problems from Table 2 (No. 1, 2, 3, 4, and 5) and compare the behavior of `saeig` without preconditioning and the MATLAB command `eigs(A,B,1,'SA')`. In Table 4 we show the obtained results. We now observe that the CPU time and the number of matrix-vector multiplications of `eigs` are significantly smaller than those of `saeig` to achieve the same precision.

Based on these experiments, we conclude that the proposed residual method `saeig` is a robust option for solving large-scale generalized eigenvalue problems, and it is effective and competitive when a suitable preconditioning strategy is available, which usually happens in real applications.

Table 3: Results for the test matrices

	saeig				eigs			
No.	MV	T	$\overline{e}$	$\overline{\mu}$	MV	T	$\overline{e}$	$\overline{\mu}$
1	84	0.08	1.3e-4	43.26501367	err	err	err	err
2	308	0.11	2.8e-3	2508.36587811	2020	0.55	3.5e-5	2508.36587811
3	30	0.05	3.1e-3	186.23517292	24620	6.47	1.8e-4	186.23517291
4	30	0.05	1.2e-2	221.19315498	66540	37.55	9.9e-5	221.19315498
5	50	0.75	2.7e-5	6.90070261	183940	89.97	1.6e-5	6.90070260
6	28	0.03	2.5e-4	29068634.2093066	37100	20.31	3.4e-2	29068634.2026896
7	292	0.69	1.9e-4	0.07864764	err	err	err	err
8	586	6.03	5.0e-2	10.51148261	1010400	1105.13	2.9e-6	10.51148263
9	102	1.06	2.3e-2	3469.30544790	84800	141.58	1.6e-4	3469.30544794
10	420	6.91	1.1e+0	1475.34074596	err	err	err	err
11	980	302.83	1.3e-2	0.00096140	1262000	12127.03	1.1e-4	0.00096140
12	30	0.33	2.9e-3	1.00000000	1563200	2320.38	4.6e-4	1.00000000
13	28	1.50	1.2e-2	1.00000000	1642740	5005.22	8.4e-4	1.00000000
14	28	0.22	7.2e-3	1.00000000	16020	70.50	1.5e-4	0.99999795
15	34	2.63	9.2e-4	1.00000000	2430800	22408.94	9.4e-4	1.00000000
16	198	25.73	4.3e-2	0.12413874	10000020	156250.00	4.4e+3	0.95282392

Table 4: Results for the test matrices without preconditioning

	saeig				eigs			
No.	MV	T	$\overline{e}$	$\overline{\mu}$	MV	T	$\overline{e}$	$\overline{\mu}$
1	2168782	100805.0	2.5e-5	43.26501367	err	err	err	err
2	437664	322.8	1.1e-6	2508.36587811	2020	0.55	3.5e-5	2508.36587811
3	1543642	5143.0	1.5e-5	186.23517291	24620	6.47	1.8e-4	186.23517291
4	2002004	9204.4	2.3e-5	221.19315498	66540	37.55	9.9e-5	221.19315498
5	4365672	10345.3	3.4e-5	6.90070260	183940	89.97	1.6e-5	6.90070260

Aknowledgements. We would like to thank Dora Jiménez from Universidad de Carabobo, Valencia, Venezuela, for programming assistance in obtaining our numerical results.

References

[1] J. Baglama, D. Calvetti, L. Reichel, and A. Ruttan. Computation of a few small eigenvalues of a large matrix with application to liquid crystal modeling. *J. Comput. Phys*, 146:203–226, 1998.

[2] L. Bello and M. Raydan. Preconditioned spectral projected gradient method on convex sets. *Journal of Computational Mathematics*, 22:4–49, 2005.

[3] Lenys Bello. *Gradiente Espectral Precondicionado sobre Convexos y Algunos Casos de Estudio.* Doctoral Thesis, Universidad Central de Venezuela, Caracas, Venezuela, 2007.

[4] E. G. Birgin, J. M. Martínez and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM J. Opt.*, 10:1196–1211, 2000.

[5] E. G. Birgin, J. M. Martínez and M. Raydan. Algorithm 813: SPG - software for convex-constrained optimization. *ACM Transactions on Mathematical Software*, 27:340–349, 2001.

[6] W. W. Bradbury and R. Fletcher. New iterative methods for solution of the eigenproblem. *Numer. Math.*, 9:259–267, 1966.

[7] J. H. Bramble, J. E. Pasciak and A. V. Knyazev. A subspace preconditioning algorithm for eigenvector/eigenvalue computation. *Adv. Comput. Math.*, 6:159–189, 1996.

[8] A. P. Costa, I. N. Figueiredo, J. Júdice, and J. A. C. Martins. The directional instability problem in systems with frictional contacts. *Computer Methods in Applied Mechanics and Engineering*, 193: 357–384, 2004.

[9] Y. H. Dai. Fast algorithms for projection on an ellipsoid. *SIAM J. Opt.*, 16:986–1006, 2006.

[10] B. N. Datta. *Numerical Linear Algebra and Applications*. Brooks/Cole Publishing Company, 1995.

[11] B. N. Datta. *Numerical Methods for Linear Control Systems Design and Analysis*. Elsevier Press, 2003.

[12] I. Duff, R. Grimes, and J. Lewis. Sparse matrix test problems. *ACM Trans. Math. Softw.*, 15:1–14, 1989.

[13] M. R. Hestenes and W. Karush. A method of gradients for the calculation of the characteristic roots and vectors of a real symmetric matrix. *J. Res. Nat. Bureau Standards*, 47:45–61, 1951.

[14] D. J. Inman. *Engineering Vibration*. Prentice-Hall, Englewood Cliffs, USA, 1994.

[15] J. J. Júdice, I. Ribeiro, and H. Sherali. The eigenvalue complementarity problem. *Computational Optimization and Applications*, 37:139–156, 2007.

[16] W. La Cruz, J. M. Martínez, and M. Raydan. Spectral residual method without gradient information for solving large-scale nonlinear systems. *Mathematics of Computation*, 75:1449–1466, 2006.

[17] W. La Cruz and M. Raydan. Nonmonotone spectral methods for large-scale nonlinear systems. *Optimization Methods and Software*, 18:583–599, 2003.

[18] S. F. McCormick. Some convergence results on the method of gradients for $Ax = \lambda Bx$. *J. Comp. Sys. Sci.*, 13:213–222, 1976.

[19] K. Neymeyr. A posteriori error estimation for elliptic eigenproblems. *Numer. Linear Alg. Appl.*, 9:263–279, 2002.

[20] M. G. Queiroz, J. J. Júdice, and C. Humes Jr. The symmetric eigenvalue complementarity problem. *Mathematics of Computation*, 73: 1849–1863, 2004.

[21] Y. Zhou. Eigenvalue computation from the optimization perpesctive: On Jacobi-Davidson, IIGD, RQI and Newton updates. *Numerical Linear Algebra with Appications*, 2:1–6, 2000.