SANDIA REPORT

SAND2007-xxxx
Unlimited Release
Printed October 2007

Orthogonalization Study and Design Document for the Anasazi and Belos Iterative Solver Packages

Christopher G. Baker, Heidi K. Thornquist

Sandia National Laboratories

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.osti.gov/bridge

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2007-xxxx
Unlimited Release
Printed October 2007

Orthogonalization Study and Design Document for the Anasazi and Belos Iterative Solver Packages

Christopher G. Baker
Sandia National Laboratories
M.S. 1320, P.O. Box 5800
Albuquerque, NM 87185-1320
cgbaker@sandia.gov

Heidi K. Thornquist
Sandia National Laboratories
M.S. 0316, P.O. Box 5800
Albuquerque, NM 87185-0316
hkthorn@sandia.gov

Abstract

The current framework for orthogonalization in the Anasazi eigensolver package and the Belos linear solver package is reviewed. Current orthogonalization needs are described, illustrating shortcomings with the current framework, and modifications are proposed.

Contents

1 Introduction

This report contains a study of the orthogonalization needs in the Anasazi and Belos iterative solver packages of the Trilinos project. We first review the necessary notations, terminology and concepts.

Let $\mathbb{F}$ be the field over which multivectors and operators are defined, usually $\mathbb{R}$ or $\mathbb{C}$. Mathematically, both multivectors and the linear operators discussed in this document can be considered as matrices over the field $\mathbb{F}$; they will be denoted by upper case Roman characters. S^H denotes the conjugate transpose of the matrix S. The range subspace of the basis V is denoted $\mathcal{R}(V)$. Orthonormal multivectors will be represented by the letters Q and V; general multivectors will be represented by S, T, X and Y. Letters C and B will denote small matrices containing coefficients. A lower Roman character represents a vector over the field $\mathbb{F}$. A subscripted vector s_i typically implies that s_i is the ith column of the matrix/multivector S; this should always be clear from context. The vectors e_i are the elementary basis vectors and the columns of the multiplicative identity matrix I. Subscripts on a matrix S_i typically implies that S_i is one in a series of associated multivectors. Lower case Greek letters will represent scalar elements of $\mathbb{F}$. Dual subscripts typically indicate that the scalar comes from an associated matrix: $\gamma_{ij} = e_i^H C e_j$. The associated between scalars and matrices should be clear from context.

The concepts of orthogonality and orthonormality are defined with respect to an inner product:

$$\langle \cdot, \cdot \rangle : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F} \ .$$

Our notation will admit the application of the inner product to multivectors, as follows:

$$\langle \cdot, \cdot \rangle : \mathbb{F}^{n\times m} \times \mathbb{F}^{n\times p} \to \mathbb{F}^{m\times p} : \langle S, T \rangle_{ij} = \langle s_i, t_j \rangle \ .$$

Two multivectors S, T are *orthogonal* with respect to the inner product if and only if $\langle S, T \rangle = 0$. A basis V is *orthonormal* if and only if $\langle V, V \rangle = I$.

Orthogonalization plays an important role in iterative solvers for linear systems and eigenvalue problems, due to the frequent occurrence of projectors, as well as the need to expand bases in Krylov subspace methods. The following examples illustrate some of the orthogonalization needs that we seek to address. Note that while these examples employ block solvers, these orthogonalization requirements exists even the block size is one.

Example 1.1
Consider a hard locking scheme for an iterative eigensolver. Such a scheme operators by removing converged eigenspaces from the iteration, and explicitly orthogonalizing the iterates against the locked (i.e., converged) eigenspaces. Given locked eigenvectors $L \in \mathbb{F}^{n\times l}$, it is necessary to apply a projector P_L to make the active eigenvectors X orthogonal to the locked eigenvectors:

$$\langle L, P_L X \rangle = 0 \ .$$

Example 1.2
The block Davidson iterative eigensolver requires that the current orthonormal basis $\begin{bmatrix} V_1 & \dots & V_i \end{bmatrix}$ be expanded by the preconditioned residual Z: find V_{i+1} such that

$$\begin{aligned} &\langle V_{i+1}, V_j \rangle = 0, \quad 1 \le j \le i \\ &\langle V_{i+1}, V_{i+1} \rangle = I \\ &Z \in \mathcal{R}(\begin{bmatrix} V_1 & \dots & V_{i+1} \end{bmatrix}) \ . \end{aligned}$$

This is usually accomplished by removing the components of previous V_j from Z and finding an orthonormal basis for the remaining multivector.

Example 1.3
A block Arnoldi method has requirements similar to the previous example. It is slightly more demanding in requiring the coefficients of orthogonalization. Given a linear operator A generating the Krylov subspace and the current orthonormal basis $\begin{bmatrix} V_1 & \dots & V_i \end{bmatrix}$, expand the basis with the columns V_{i+1}:

$$\begin{aligned} \langle V_{i+1}, V_j \rangle &= 0, \quad 1 \le j \le i \\ \langle V_{i+1}, V_{i+1} \rangle &= I \\ AV_i &= V_1 H_{1,i} + \cdots + V_i H_{i,i} + V_{i+1} H_{i+1,i} \ . \end{aligned}$$

The matrices $H_{i,j}$ blocks from the blocks Hessenberg matrix H.

These needs can be summarized into the following methods:

- **projection**: remove components pertaining to a basis Q from a given multivector: $S \overset{Q}{\mapsto} \hat{S}$ such that $\hat{S}$ and Q are orthogonal.
- **normalization**: compute an orthonormal basis for the subspace spanned by a given multivector: $S \mapsto V$ such that $S \in \mathcal{R}(V)$ and V is an orthonormal basis.
- **decomposition**: produce an orthonormal factorization of a given multivector: given S and orthonormal basis Q, find an orthonormal basis V such that $S = QC + VB$ and $\langle V, Q \rangle = 0$.

Anasazi and Belos were designed to use an abstract interface for these computations. This allows the implementation of linear and eigenvalue solvers to be separated from the implementation of these orthogonalization routines. This enables the user to decide which definition of orthogonality (via the choice of inner product) is appropriate for a particular type of problem, in addition to deciding which orthogonalization method (e.g., classical Gram-Schmidt) should be used in a particular situation. This follows the overarching theme in these solvers: to allow ample flexibility in subjacent computations such that a variety of different problems and approaches are possible.

In both Anasazi and Belos, matters related to orthogonalization are encapsulated in an abstract base class ORTHOMANAGER. Choices of inner product, as well as computational implementations, are provided by concrete derived classes. As a result, a solver can request orthogonalization routines via this abstract interface, without requiring knowledge of the underlying implementation. The abstract interface describes a framework for orthogonalization, whose description dictates exactly what is permitted and possible.

We desire the following properties of an orthogonalization framework:

P1. that it not restrict the efficiency of the orthogonalization routines, either in terms of computational effort or memory requirements;

P2. that requirements on the input and output of the orthogonalization routines be specified so as to allow their usage without knowing anything about the underlying implementation; and

P3. that it be broad enough to enable the implementation of a variety of orthogonalization procedures and techniques.

The first point requires that, whatever the benefits of using this framework, they are not met with a performance penalty, aside from constant time overhead associated with the object-oriented programming paradigm employed by Trilinos/Anasazi/Belos. Otherwise, a performance-oriented user will simply ignore the provided orthogonalization framework and develop efficient code. In order to effect code reuse, the provided code must be worth reusing. The second point requires that the operations of the orthogonalization manager be well-defined and internally consistent, so that the mathematical outcome of those operations are useful, even though the specific implementations of the operations are unspecified. The third point addresses the descriptive capability of the framework. This point was the motivation of this study, as the current orthogonalization framework is unable to describe some desired approaches. Next, we describe the current framework and provide an example of its shortcomings.

2 Current Framework

Currently, each orthogonalization manager provides an inner product, denoted here by $\langle \cdot, \cdot \rangle$, as well as routines for block orthonormalization and orthogonalization. The outcome of these routines (i.e., the concepts of orthogonality and orthonormality) is defined in term of the inner product provided by the orthogonalization manager. Table 1 lists the operations provided by ORTHOMANAGER.

Table 1. This table lists the methods currently provided by the class ORTHOMANAGER.

Inputs	Outputs	Description
$innerProd(S,T)$	C	Compute inner products into matrix $C = \langle S, T \rangle$
$norm(S)$	z	Compute induced norms, $z_i = \sqrt{\langle s_i, s_i \rangle}$
$normalize(S)$	V, B	Compute orthonormal V such that $S = V \langle V, S \rangle = VB$
$project(Q,S)$ $\langle Q, Q \rangle = I$	$\hat{S}, C$	Compute $\hat{S}$ such that $\langle Q, \hat{S} \rangle = 0$ and $S = \hat{S} + QC$
$projectAndNormalize(Q,S)$ $\langle Q, Q \rangle = I$	V, C, B	Compute orthonormal V such that $\langle V, Q \rangle = 0$ and $S = QC + VB$

In many cases, the operator defining the inner product is given in the form of a symmetric positive definite operator, M:

$$\langle S, T \rangle_M = S^H (MT) = (MS)^H T \ .$$

Computing the inner product (a necessary step in the listed orthogonalization routines) requires applying the operator M to one of the multivectors. In the case that the caller has already computed this intermediate result, the interface must be capable of exploiting this information, for the sake of efficiency. As a result, the current orthogonalization framework describes a subclass of ORTHOMANAGER called MATORTHOMANAGER (see Figure 1). This class augments the methods of ORTHOMANAGER by the provision of operator-multivector products in the case that the user might already have them. The methods of MATORTHOMANAGER are listed in Table 2. Two orthogonalization managers are currently provided by Anasazi. The class BASICORTHOMANAGER implements a classical Gram-Schmidt orthogonalization routine with optional reorthogonalization according to a DGKS strategy[2]. The block orthogonalization method described by Stathapoulos and Wu [7] is implemented in the subclass SVQBORTHOMANAGER.

A study of these tables reveals some assumptions made by the current framework, both regarding the form of the input as well as the specific action of the orthogonalization methods. Some of these assumptions are limiting in our ability to apply some projectors. Section 3 describes an example of these limitations.

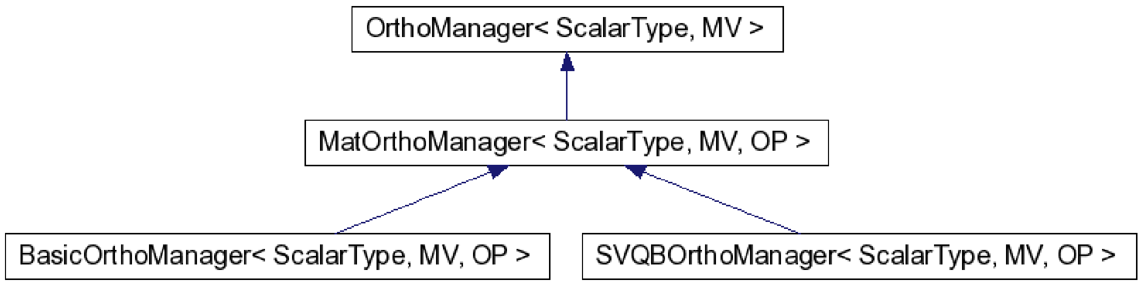
OrthoManager< ScalarType, MV >
MatOrthoManager< ScalarType, MV, OP >
BasicOrthoManager< ScalarType, MV, OP >
SVQBOrthoManager< ScalarType, MV, OP >

Figure 1. Class inheritance for current framework.

Table 2. This table lists the methods currently provided by the class MatOrthoManager. Note these methods differ from those in Table 1 by their names and the ability to provide the intermediate products needed by the inner product.

Inputs	Outputs	Description
$innerProdMat(S,T,MT)$	C	Compute inner products $C = S^H MT$
$normMat(S,MS)$	z	Compute induced norms, $z_i = \sqrt{s_i^H M s_i}$
$normalizeMat(S,MS)$	V,B	Compute orthonormal V such that $S = V\langle V,S\rangle = VB$
$projectMat(Q,S,MS)$ $\langle Q,Q\rangle = I$	$\hat{S},C$	Compute $\hat{S}$ such that $\langle Q,\hat{S}\rangle = 0$ and $S = \hat{S}+QC$
$projectAndNormalizeMat(Q,S,MS)$ $\langle Q,Q\rangle = I$	V,C,B	Compute orthonormal V such that $\langle V,Q\rangle = 0$ and $S = QC+VB$

3 Motivating Examples

Assume that we wish to apply the projector

$$\Pi = I - MV(V^H MMV)^{-1} V^H M \ , \tag{3.1}$$

where M is a symmetric positive definite matrix. This projector removes the components of V with respect to the M-inner product, via a modification in the range of MV. This projector is used, for example, in a number of optimization-based eigensolvers; see [5, 4, 1].

Only one possibility exists for describing this projector in the current framework. By setting $\tilde{V} \doteq MVN$, where N is chosen so that $\tilde{V}^H \tilde{V} = N^H V^H MMVN = I$, we can use the *project*() method of an OrthoManager employing the Euclidean inner product:

$$\hat{S} = project(\tilde{V}, S) \ .$$

In this way, the resulting multivector satisfies $\langle \hat{S}, V \rangle_M = \langle \hat{S}, MV \rangle_I = \langle \hat{S}, \tilde{V} N^{-1} \rangle_I = 0$. There are two problems with this approach.

The first problem with this approach is that it required the construction of $\tilde{V}$, an orthonormal basis for $\mathcal{R}(MV)$. This constraint resulted from the assumption by *project*() that the input basis is orthonormal. The construction of this orthonormal basis can be performed using the orthogonalization manager's *normalize*() method. However, it requires additional storage for the multivector $\tilde{V}$, as well as additional effort on the part of the user.

The second problem with this approach is its reliance on the Euclidean inner product. In employing the Euclidean inner product for the *project*() method, the methods *normalize*() and *projectAndNormalize*() will return bases that are orthonormal with regard to this basis. It is possible for a user requiring M-orthonormal bases to employ the *normalize*() routine of a second orthogonalization manager, using an M-based inner product. We do not feel that this is a satisfactory solution; it should be possible to employ a single orthogonalization manager for all needs. Furthermore, the *projectAndNormalize*() method (if implemented according to specification) would be unavailable: orthogonality needs to be defined with respect to the I-based inner product, while normalization needs to be defined according to the M-based inner product.

For another example, consider also the application of the projector

$$\Omega = I - K^{-1} MV(V^H MK^{-1} MV)^{-1} V^H M \ , \tag{3.2}$$

where M is a symmetric positive definite matrix. This projector is used to provide the solution T to the linear system

$$\Pi K \Pi T = U \ ,$$

where Π is as in (3.1), $U = \Pi U$ and we desire $T = \Pi T$. The solution is given by

$$T = (I - K^{-1} MV(V^H MK^{-1} MV)^{-1} V^H M) K^{-1} U \ ,$$

a formula due to Olsen et al. [3]. This scenario occurs when preconditioning the inner iteration for a number of Newton-like eigensolvers; see [5, 6, 4, 1]. It is desirable to apply this product via an orthogonalization routine, as it is often more important that the resultant T is in the proper subspace than that the system is solved exactly.

As with Example (3.1), it is possible to finagle the desired solution via the proper choice of orthogonal projector, inner product and post-application of K^{-1}:

$$\begin{aligned} U &= K^{-1}U - K^{-1}MV(V^HMK^{-1}MV)^{-1}V^HMK^{-1}U \\ &= K^{-1}\left(U - \tilde{V}\langle\tilde{V},U\rangle_{K^{-1}}\right) , \end{aligned}$$

where $\tilde{V}$ is a K^{-1}-orthonormal basis for MV. This may be an unfavorable solution, as it requires an orthogonalization manager using the K^{-1}-inner product along with the creation of the temporary basis $\tilde{V}$.

Although these examples employ less common projectors, they illustrates that fact that the current framework is too strict in dictating the behavior of the $project()$ and $projectAndNormalize()$ routines. A more general setting will allow us the flexibility to apply a broader class of projectors.

4 Proposed Framework

We first describe the form of a general projector, in order to design the proposed framework around this concept. Assume an inner product $\langle\cdot,\cdot\rangle$. Then given bases X and Y, define the general projector as

$$P_{X,Y}S = S - X\langle Y,X\rangle^{-1}\langle Y,S\rangle \ .$$

This projector works by modifying the input in the range of X so that the result is orthogonal to Y. In this way, the projector is specified according to three parameters:

- the inner product defining orthogonality,
- a basis Y for the subspace against which the orthogonality is effected, and
- a basis X for the subspace where the effect of the projection occurs.

The previous framework allowed only the first two of these three parameters to be specified. It was assumed there that $X = Y$.

This allows the example from (3.1) to be implemented using the M-inner product, by choosing $X = MV$ and $Y = V$:

$$P_{MV,V}S = S - MV\langle V,MV\rangle_M^{-1}\langle V,S\rangle_M = S - MV(V^HMMV)^{-1}(V^HMS) \ .$$

The example from (3.2) can be implementing using the M-inner product, by choosing $X = K^{-1}MV$ and $Y = V$, and applying the projector to $K^{-1}U$:

$$P_{K^{-1}MV,V}K^{-1}U = \left(I - K^{-1}MV\left(V^HMK^{-1}MV\right)^{-1}V^HM\right)K^{-1}U = T \ .$$

Therefore, we have the ability to natively manage two of the applications that are not sufficiently addressed by the current framework. Furthermore, the current framework is superseded. The projection operations described there are accessible via the proposed framework by supplying $X = Y$. Table 3 lists the proposed projection methods, with their abstract definitions.

One constraint is placed on the inputs X and Y. It will be assumed that $\langle Y,X\rangle$ is a symmetric positive definite matrix. This has benefit of allowing a Cholesky factorization to be used in the case that inverse of $\langle Y,X\rangle$ needs to be applied. Note that this requirement does not preclude the application of either example from Section 3.

Recall the desired properties of an orthogonalization framework:

P1. that it not restrict the efficiency of the orthogonalization routines, either in terms of computational effort or memory requirements;

Table 3. Projection methods under the proposed orthogonalization framework.

Inputs	Outputs	Description
$project(X,Y,S)$ $\langle Y,X\rangle$ is s.p.d.	$\hat{S},C$	Compute $\hat{S}$ such that $\langle Y,\hat{S}\rangle = 0$ and $S = \hat{S} + XC$
$projectAndNormalize(X,Y,S)$ $\langle Y,X\rangle$ is s.p.d.	V,C,B	Compute orthonormal V such that $\langle Y,V\rangle = 0$ and $S = XC + VB$

P2. that requirements on the input and output of the orthogonalization routines be specified so as to allow their usage without knowing anything about the underlying implementation; and

P3. that it be broad enough to enable the implementation of a variety of orthogonalization procedures and techniques.

Table 3 rigorously describes the orthogonalization methods provided by the proposed framework, satisfying P2. The generalized projection used in the proposed framework has more descriptive power than the current framework and enables the extensibility of the orthogonalization manager, which addresses P3.

The last goal is to ensure that efficient implementation is not hampered by the proposed framework. One efficiency requirement concerns the application of the general projector:

$$P_{X,Y}(S) = S - X\langle Y,X\rangle^{-1}\langle Y,S\rangle \ .$$

In many cases, the multivectors X and Y may be known to be bi-orthonormal, i.e., $\langle Y,X\rangle = I$. This is true in particular when $X = Y$ is orthonormal. In this scenario, we wish to avoid the computation and inversion of $\langle Y,X\rangle$, which may constitute a large portion of the cost of applying the projector, especially as X and Y grow large relative to S. This is not a concern in the current framework, as the projection operations assume $X = Y$ to be orthonormal. The framework allows for this consideration by accepting an argument to the $project()$ and $projectAndNormalize()$ methods which specifies the bi-orthogonality of X and Y, as shown in Table 4.

Note that the proposed framework modifies only the definition of projection; normalization is not modified. Therefore we need only to expand the methods $project()$ and $projectAndNormalize()$ to provide this additional capability. The modified interfaces will be implemented as a new abstract class, GenOrthoManager. The motivation for the class MatOrthoManager still stands: in the case of an M-based inner product, we wish to support the provision of MX, MY and MS. Therefore, GenOrthoManager will inherit from MatOrthoManager. The updated class hierarchy is listed in Figure 2.

Table 4. This table lists the method signatures for the class GENORTHOMANAGER. `tsdm` is a TEUCHOS::SERIALDENSEMATRIX, `mvec` is a multivector object, and `bool` is a boolean.

`projectGen(X,Y,isBiOrtho,S,C,MX,MY,MS)`		
`mvec X`	`in`	A basis for the subspace where S is modified
`mvec Y`	`in`	A basis for the subspace S is projected away from
`bool isBiOrtho`	`in`	`bool` indicating if $\langle X,Y\rangle = I$
`mvec S`	`in/out`	On output, $\langle Y,S_{out}\rangle = 0$ and $S_{in} = S_{out} + XC$
`tsdm C`	`out`	The coefficients $C = \langle X,Y\rangle^{-1}\langle Y,S_{in}\rangle$
`mvec MX`	`out`	*Optional:* Provides $M \cdot X$ for the inner product
`mvec MY`	`out`	*Optional:* Provides $M \cdot X$ for the inner product
`mvec MS`	`out`	*Optional:* On input, provides $M \cdot S_{in}$ for the inner product; on output, returns $M \cdot S_{out}$

`projectAndNormalizeGen(X,Y,isBiOrtho,S,C,B,MX,MY,MS)`		
`mvec X`	`in`	A basis for the subspace where S is modified
`mvec Y`	`in`	A basis for the subspace S is projected away from
`bool isBiOrtho`	`in`	`bool` indicating if $\langle X,Y\rangle = I$
`mvec S`	`in/out`	On output, the basis V satisfying $\langle V,V\rangle = I$ and $\langle Y,V\rangle = 0$ and $S_{in} = XC + VB$.
`tsdm C`	`out`	The coefficients $C = \langle X,Y\rangle^{-1}\langle Y,S_{in}\rangle$
`tsdm B`	`out`	The coefficients $B = \langle V,S_{in} - XC\rangle$
`mvec MX`	`out`	*Optional:* Provides $M \cdot X$ for the inner product
`mvec MY`	`out`	*Optional:* Provides $M \cdot X$ for the inner product
`mvec MS`	`out`	*Optional:* On input, provides $M \cdot S_{in}$ for the inner product; on output, returns $M \cdot S_{out}$
`return`		Returns integer specifying the rank of the computed basis

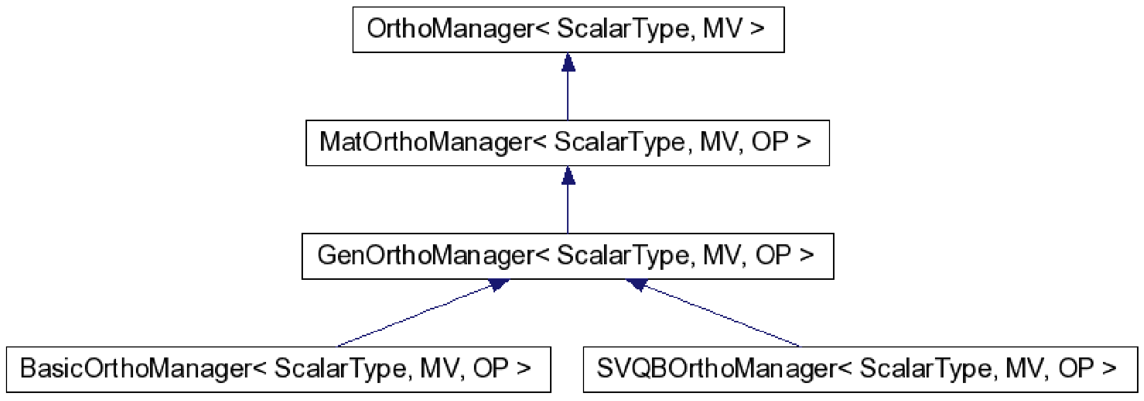
OrthoManager< ScalarType, MV >
MatOrthoManager< ScalarType, MV, OP >
GenOrthoManager< ScalarType, MV, OP >
BasicOrthoManager< ScalarType, MV, OP >
SVQBOrthoManager< ScalarType, MV, OP >

Figure 2. Class inheritance for proposed framework.

5 Transition and Testing

The proposed changes to the framework can be implemented so as to enable easy testing and avoid any conflict with existing code. Because GENORTHOMANAGER extends the current framework by inheriting from the lowest level of the orthogonalization manager class hierarchy, no changes are necessary for the existing classes. The new projection methods can be implemented alongside the current implementations. Existing code using the interfaces ORTHOMANAGER and MATORTHOMANAGER would be linked with the current and proven implementations. Furthermore, as the proposed methods $projectGen()$ and $projectAndNormalizeGen()$ generalize the methods $projectMat()$ and $projectAndNormalizeMat()$, a class which retains the previous implementations of these methods has a trivial means by which to validate a subset of the functionality of the new implementations.

For this reason, the transition from the current to the proposed framework would include moving the concrete orthogonalization managers BASICORTHMANAGER and SVQBORTHOMANAGER underneath GENORTHOMANAGER. The existing implementations supporting MATORTHOMANAGER and ORTHOMANAGER can remain as they are for an initial testing period, while new implementations can be created to support the projection methods of GENORTHOMANAGER.

References

[1] P.-A Absil, C. G. Baker, and K. A. Gallivan. A truncated-CG style method for symmetric generalized eigenvalue problems. *J. Comput. Appl. Math.*, 189(1–2):274–285, 2006.

[2] J. Daniel, W.B. Gragg, L. Kaufman, and G.W. Stewart. Reorthogonalization and stable algorithms for updating the gram-schmidt qr factorization. *Math. Comput.*, 30:772–795, 1976.

[3] J. Olsen, P. Jørgensen, and J. Simons. Passing the one-billion limit in full configuration-interaction (FCI) calculations. *Chemical Physics Letters*, 169:463–472, June 1990.

[4] A. Sameh and Z. Tong. The trace minimization method for the symmetric generalized eigenvalue problem. *J. Comput. Appl. Math.*, 123:155–175, 2000.

[5] A. H. Sameh and J. A. Wisniewski. A trace minimization algorithm for the generalized eigenvalue problem. *SIAM J. Numer. Anal.*, 19(6):1243–1259, 1982.

[6] G. L. G. Sleijpen and H. A. van der Vorst. A jacobi-davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17:401–425, 1996.

[7] A. Stathapoulos and K. Wu. A block orthogonalization procedure with constant synchronization requirements, 2000.

Sandia National Laboratories