Some comments on the DIIS method

Ron Shepard & Michael Minkoff

Molecular Physics, Vol. 105, Nos. 19–22, 10 October–20 November 2007, 2839–2848

Some comments on the DIIS method

RON SHEPARD* and MICHAEL MINKOFF
Chemistry Division, Argonne National Laboratory, Argonne, IL 60439, USA

(*Received 18 July 2007; in final form 18 September 2007*)

Over twenty ways to solve the DIIS equation are introduced and discussed. Some of these solution methods are inherently inaccurate and are never recommended, others have some efficiency or accuracy advantages that may be exploited in different situations. Two of these ways, linear least squares with substitution and elimination, and linear least squares with equality constraint, have the potential to produce solutions that are significantly more accurate than the traditional normal equation solutions. The choice of the most accurate solution approach for a given problem depends on various vector and matrix norm relations which are discussed. The advantage of one of these methods is demonstrated on a model problem, which has a closed-form solution. A simple geometrical interpretation is given for the DIIS method that involves minimization of a function value within a particular multidimensional plane.

Keywords: DIIS; Least squares; Optimization

1. Introduction

The direct inversion in the iterative subspace (DIIS) method of Pulay [1, 2] is used commonly as a convergence acceleration technique. It consists of two steps, a least-squares condition imposed on a set of error vectors, followed by an interpolation of the associated input vectors. We discuss some aspects of the method regarding the numerical accuracy of the computed solution to the least squares equation.

Suppose a solution $\mathbf{x}_*$ of some general nonlinear vector equation is desired

$$\mathbf{F}(\mathbf{x}_*) = 0 \tag{1}$$

with $\mathbf{F}(\mathbf{x}_0) \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^q$. We assume that the components of $\mathbf{F}(\mathbf{x})$ are smooth and continuous with respect to the variables $\mathbf{x}$ in some neighborhood of the solution. Let $\mathbf{x}_0$ be some reference point within this neighbourhood. The function can then be expanded about this reference point as

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x}_0) + \nabla\mathbf{F}(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0) + O\left(\|\mathbf{x}-\mathbf{x}_0\|^2\right) \tag{2}$$

where the matrix $\nabla\mathbf{F}(\mathbf{x}_0) \in \mathbb{R}^{m\times q}$ is the Jacobian at the reference point $\mathbf{x}_0$. (Unless designated otherwise, all vector norms hereafter are Euclidian 2-norms.) Let

$$\mathbf{F}^{[1]}(\mathbf{x}) = \mathbf{F}(\mathbf{x}_0) + \nabla\mathbf{F}(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0) \tag{3}$$

be the first-order approximation of the exact function. In a standard Newton or quasi-Newton scheme, $\mathbf{x}$ would be determined from the solution of $\mathbf{F}^{[1]}(\mathbf{x}) = \mathbf{0}$. This computed $\mathbf{x}$, which approximates the desired $\mathbf{x}_*$, would be used to replace the expansion point $\mathbf{x}_0$ (either directly or, for example, combined with line-searches or trust-radius constraints), and the process would be repeated until convergence is achieved. This would require computing, or approximating, the Jacobian $\nabla\mathbf{F}(\mathbf{x}_0)$ at each step in the iterative procedure, and this is often impractical or inefficient. In the DIIS method, the Jacobian is neither computed nor approximated explicitly. Instead, the vector $\mathbf{x}$ is taken as a linear combination of basis vectors

$$\mathbf{x} = \sum_{k=1}^{n} \mathbf{x}_k c_k = \mathbf{X}\mathbf{c} \tag{4}$$

with $n > 1$. This expansion holds for arbitrary $\mathbf{c}$, but we are particularly interested in expansion vectors $\mathbf{c}$ for which

$$\mathbf{u}^T\mathbf{c} = 1 \tag{5}$$

*Corresponding author. Email: Shepard@tcg.anl.gov

Molecular Physics
ISSN 0026–8976 print/ISSN 1362–3028 online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00268970701691611

with the vector $u_k = 1$ for all k (which we hereafter call the *uno* vector). This relation has a simple geometrical interpretation. Consider the values c_k to be coordinates in $\mathbb{R}^n$; the vector $\mathbf{u}$ is seen to have a specific angle, $\theta = \cos^{-1}(1/\sqrt{n})$ relative to each of the coordinate axes in this space. There are an infinite number of parallel planes within this n-dimensional space that are perpendicular to the vector $\mathbf{u}$, and the relation $\mathbf{u}^T\mathbf{c} = 1$ selects one of these planes; the values c_k are the full set of coordinates in $\mathbb{R}^n$ that lie within this particular plane. This is the particular plane that contains the n points $(1, 0, \ldots, 0)^T$, $(0, 1, \ldots, 0)^T, \ldots, (0, 0, \ldots, 1)^T$ in $\mathbb{R}^n$. In this case, the following sequence of identities holds.

$$\begin{aligned}\mathbf{F}^{[1]}(\mathbf{x}) &= \mathbf{F}(\mathbf{x}_0)\mathbf{u}^T\mathbf{c} + \nabla\mathbf{F}(\mathbf{x}_0)\big(\mathbf{X}\mathbf{c} - \mathbf{x}_0\mathbf{u}^T\mathbf{c}\big)\\ &= \big(\mathbf{F}(\mathbf{x}_0)\mathbf{u}^T + \nabla\mathbf{F}(\mathbf{x}_0)\big(\mathbf{X} - \mathbf{x}_0\mathbf{u}^{\mathrm{T}}\big)\big)\mathbf{c}\\ &= \sum_{k=1}^{n}(\mathbf{F}(\mathbf{x}_0) + \nabla\mathbf{F}(\mathbf{x}_0)(\mathbf{x}_k - \mathbf{x}_0))c_k\\ &= \sum_{k=1}^{n}\mathbf{F}^{[1]}(\mathbf{x}_k)c_k. \qquad (6)\end{aligned}$$

The approximate function $\mathbf{F}^{[1]}(\mathbf{x})$ is replaced by the exact function $\mathbf{F}(\mathbf{x})$ on both sides of this expression to give

$$\mathbf{F}(\mathbf{x}) + \mathrm{O}\big(\|\mathbf{x} - \mathbf{x}_0\|^2\big) = \sum_{k=1}^{n}\mathbf{F}(\mathbf{x}_k)c_k = \sum_{k=1}^{n}\mathbf{e}_k c_k = \mathbf{E}\mathbf{c} \qquad (7)$$

$\mathbf{e}_k = \mathbf{F}(\mathbf{x}_k)$ is regarded as the error associated with the basis vector $\mathbf{x}_k$. In the plane defined by the constraint, equation (5), the function value $\mathbf{F}(\mathbf{x})$ is approximated through first order as the simple linear combination of the error values $\mathbf{e}_k$. This approximation requires only the function values $\mathbf{F}(\mathbf{x}_k)$, it does not require the computation of the Jacobian. In general, there does not exist a set of coordinates values $\mathbf{c}$ in the plane equation (5) for which $\mathbf{E}\mathbf{c} = -\mathbf{r} = \mathbf{0}$ is exactly satisfied. Pulay proposed to determine $\mathbf{c}$ according to the least-squares minimization of $\mathbf{r}^2$, which may be written

$$\min_{\mathbf{c}} \quad \|\mathbf{E}\mathbf{c}\| \quad \text{with } \mathbf{u}^T\mathbf{c} = 1. \qquad (8)$$

Once $\mathbf{c}$ is determined in this manner, an interpolated value of $\mathbf{x}_{\mathrm{DIIS}} = \mathbf{X}\mathbf{c}$ is given by equation (4) which approximates the desired solution $\mathbf{x}_*$. This new interpolated input $\mathbf{x}_{\mathrm{DIIS}}$ is then added to the subspace basis, or it replaces some existing member of the subspace, or it is combined with a line-search or a Newton step, and the iterative process is continued.

The DIIS method has been applied in many situations in chemistry [1–13], including orbital optimization (RHF, UHF, GVB, and Kohn-Sham DFT orbitals), orbital localization, and molecular geometry optimization. In the orbital optimization cases, the input variables have been chosen to be either the orbital coefficients, the essential variables that define those coefficients, or the density that results from those orbitals, and the error is taken usually to be the gradient of the energy with respect to an orbital variation (i.e. a Fock matrix element). The expensive Jacobian, the constructions of which the DIIS method attempts to avoid or minimize, is the Hessian of the energy. For geometry optimization, the input variables are the coordinates $\mathbf{q}_k$ of each molecular conformation and the corresponding error vectors are taken as $\mathbf{e}_k = \mathbf{g}_k$, $\mathbf{e}_k = \mathbf{H}^{-1}\mathbf{g}_k$, or $\mathbf{e}_k = \mathbf{H}^{-1/2}\mathbf{g}_k$ where $\mathbf{g}_k$ is a gradient of the potential energy surface at $\mathbf{q}_k$ and $\mathbf{H}$ is either an exact or an approximate Hessian matrix (which may be regarded as a preconditioner to accelerate convergence). These applications demonstrate the wide range of applicability of the DIIS method and the general nature of the input variables and of the error measure. In this work, we discuss several approaches to determine the solution of the constrained least-squares problem, equation (8). We do not discuss the details of how the vectors $\mathbf{x}_k$ are chosen, how the subspace dimension n is increased or decreased during the overall procedure, or how the interpolated $\mathbf{x}_{\mathrm{DIIS}}$ is used in the overall convergence procedure.

DIIS is associated with the broad class of iterative subspace methods for optimization and linear equation solutions such as conjugate-gradients (in the symmetric positive-definite case) and GMRES (in the general case) [14]. It is clear from equation (4) that $\mathbf{x}_{\mathrm{DIIS}} \in \mathrm{Span}(\mathbf{X})$ and that the DIIS procedure alone has no ability to expand the rank of this search subspace. This implies that the DIIS procedure must generally be combined with some other computational procedure, or set of procedures, that does have the ability to generate appropriate trial vectors. The purpose of the DIIS step is to extract the best approximation to $\mathbf{x}_*$ from the current subspace with the minimal effort (e.g. without additional Jacobian computations). We point out at this time that there has been imposed no additional conditions on the input vectors $\{\mathbf{x}_k\colon k = 1, \ldots, n\}$. They need not be members of any specific sequence, convergent or otherwise, they need not be all determined by the same numerical procedure, and they need not satisfy any kind of orthogonality conditions. Given any set of basis vectors within the neighbourhood associated with equation (2), and their corresponding errors, the above DIIS interpolation may be applied. We also note that the reference vector $\mathbf{x}_0$ serves no specific role in the computational procedure; its only purpose is to define the formal expansion upon which the computational procedure is based.

2. Discussion

There are several approaches to the solution of equation (2). Pulay [1] employed a Lagrange multiplier expression of the form

$$L(\mathbf{c},\lambda) = \mathbf{c}^T\mathbf{B}\mathbf{c} - 2\lambda\left(\mathbf{u}^T\mathbf{c} - 1\right) \tag{9}$$

with $\mathbf{B} \equiv \mathbf{E}^T\mathbf{E}$. Minimization of $L(\mathbf{c}, \lambda)$ results in the coupled equations

$$\mathbf{B}\mathbf{c} = \lambda\mathbf{u} \tag{10}$$

$$\mathbf{u}^T\mathbf{c} = 1. \tag{11}$$

Pulay proposed to solve these coupled equations as a linear equation solution of the augmented system

$$\begin{pmatrix} \mathbf{B} & -\mathbf{u} \\ -\mathbf{u}^T & 0 \end{pmatrix}\begin{pmatrix} \mathbf{c} \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}. \tag{12}$$

The signs and scale factors in equations (9)–(12) are somewhat arbitrary, and the above choice is largely based on precedence. When written in this form, the linear equation system in equation (10) is symmetric and nonnegative definite, and the system in equation (12) is symmetric indefinite (there is one negative eigenvalue). The solution of equation (12) is unique provided $\mathbf{B}$ is nonsingular (symmetric positive definite), which in turn occurs when the columns of $\mathbf{E}$ are linearly independent. Once the solution has been found, then $\lambda = \mathbf{c}^T\mathbf{B}\mathbf{c} = \|\mathbf{r}\|^2$ which is the quantity being minimized. In exact arithmetic, the solution of equation (12) is exactly the solution of equation (8). It is well known, however, in the solution of least squares equations that use the matrix $\mathbf{B}$ can result in inaccurate solutions. The solutions equations (10)–(12) are called the *normal equation solution*, and there are two main problems with this approach.

The first problem is that useful information in $\mathbf{E}$ is lost when forming $\mathbf{B}$ in finite precision arithmetic. A simple example will illustrate this point. Suppose $\mathbf{E} = \left(\begin{smallmatrix} 1 & 1 \\ 0 & \delta \end{smallmatrix}\right)$, then $\mathbf{B} = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 1+\delta^2 \end{smallmatrix}\right)$ in exact arithmetic. Suppose that $\delta < \sqrt{\varepsilon}$ where ε (the *machine epsilon*) is the relative precision of the floating point representation. If $fl(x)$ is the floating point representation of the exact number x, then $fl(1+\delta^2) = 1$, $\mathbf{B} = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right)$ in finite precision, and all information about δ has been lost in equations (9)–(12) (or with any other formulation based on the matrix $\mathbf{B}$); furthermore, because the computed $\mathbf{B}$ is singular in this example, a unique solution to equation (12) does not exist even though the columns of $\mathbf{E}$ are linearly independent. δ is not necessarily a small number. For example, with 32-bit IEEE floating point arithmetic, $\varepsilon \approx 1.2 \times 10^{-7}$ and if $\delta = 10^{-4}$ then the finite precision $\mathbf{B}$ is singular; with 64-bit IEEE floating point arithmetic, $\varepsilon \approx 2.2 \times 10^{-16}$ and if $\delta = 10^{-8}$ then the finite precision $\mathbf{B}$ is singular. In general, even if the matrix $\mathbf{B}$ is not singular and the optimization equations could be solved exactly, the computed solution would still reflect the errors associated with the neglect of quantities of magnitude δ^2 in the finite precision representation of $\mathbf{B}$ which result from contributions of magnitude δ in the matrix $\mathbf{E}$.

The second problem is that the condition number of the matrix $\mathbf{B}$ is the square of the condition number of the matrix $\mathbf{E}$; $\kappa(\mathbf{B}) = \kappa(\mathbf{E})^2$. The condition number [15, 16, 19] determines the relative precision that numerical solutions can be computed in finite precision. For our purposes, the condition number of a matrix with respect to linear equation solutions can be defined as the ratio of the largest singular value to the smallest singular value, $\kappa = \sigma_{\max}/\sigma_{\min}$. For the symmetric positive-definite matrix $\mathbf{B}$, this definition is equivalent to the ratio $\kappa = \omega_n/\omega_1$ of the largest and smallest eigenvalues. In practice, the singular value decomposition (SVD) of a matrix is a relatively expensive operation, and standard library routines often approximate the condition number with methods that require less effort; furthermore, for matrices with large condition numbers, the small singular values are typically computed with large relative error. Because of these and other numerical approximations, the bounds relations using these condition numbers are also approximations. The condition number is a measure of the sensitivity of the solution vector to small changes in the input, or alternatively, a measure of how much the relative error in the input grows in the error of the computed solution. Because $\kappa \geq 1$ from its definition, the relative error from the input to the output always tends to grow in a linear equation solution, it never decreases. In general, the maximum relative error of a computed linear equation result is at least $\kappa\varepsilon$ (depending on the rounding mode, the existence of guard bits, and other details of the floating point arithmetic) because of the finite representation of the input and of the intermediate quantities computed during the solution. If there are other errors in the input of magnitude $\sim\delta$ (due to errors in the raw data, numerical approximations, or propagated from previous steps in other ways), then the relative error in the output would be $\sim\kappa\delta$. If $\kappa \approx 1$, then the computed solution to a linear equation can be found that is accurate to the same precision as the input. This occurs for orthogonal matrices, and it is the reason that many numerical methods are based on sequences of orthogonal transformations. If $\kappa \approx 10^4$, then the computed solution will have about four fewer correct significant digits than the input. If $\kappa \approx 1/\varepsilon$, then the computed solution would be expected to have no correct digits regardless of the accuracy of the input. This error growth is a separate issue from the loss of precision in the construction of **B** because, in principle, even if **B** happened to be represented exactly, the subsequent computed solution in finite precision floating point arithmetic would show the condition number factor because of the truncation errors of the intermediate quantities in the computation.

The minimization in equation (8) is called a linear least squares problem with equality constraint (LSE). It is one of the standard minimization problems, it is well studied, and it may be solved with high-level drivers in LAPACK [16, 17] (i.e. SGGLSE and DGGLSE with the F77 interface, or LA_GGLSE with the F95 interface). These standard methods typically solve the somewhat more general optimization problem

$$\overset{\min}{\mathbf{c}} \|\mathbf{Ec} - \mathbf{b}\| \quad \text{with } \mathbf{Dc} = \mathbf{d} \tag{13}$$

with $\mathbf{D} \in \mathbb{R}^{p \times n}$ and $\mathbf{d} \in \mathbb{R}^p$ embodying p simultaneous linear constraints. The DIIS problem corresponds to $\mathbf{b} = \mathbf{0}$, $p = 1$, $\mathbf{D} = \mathbf{u}^T$, and $\mathbf{d} = 1$. Thus it is possible to solve the DIIS equations with a single call to a standard library routine. The solution of the least-squares equation with equality constraints for the DIIS problem has an error bound of the general form

$$\frac{\|\mathbf{c} - \mathbf{c}_{\text{exact}}\|}{\|\mathbf{c}_{\text{exact}}\|} \le \varepsilon \left\{ \alpha\kappa(\mathbf{E}) + \beta\kappa^2(\mathbf{E}) \frac{\|\mathbf{r}\|}{\|\mathbf{E}\| \cdot \|\mathbf{c}_{\text{exact}}\|} \right\}. \tag{14}$$

The scalars α and β depend on matrix and vector norms of expressions involving the arrays **E** and **u** [15, 16]. This shows that for small $\|\mathbf{r}\|/\|\mathbf{E}\|$, the relative error depends on the condition number $\kappa(\mathbf{E})$, but for large $\|\mathbf{r}\|/\|\mathbf{E}\|$, the error depends on $\kappa(\mathbf{E})^2 = \kappa(\mathbf{B})$. Thus, this method does not guarantee an accurate solution in all cases, but under optimal conditions the accuracy can exceed that of the normal equation solution.

Equation (12) is equivalent to the linear equation

$$\begin{pmatrix} \mathbf{1} & \mathbf{E} & \mathbf{0} \\ \mathbf{E}^T & \mathbf{0} & \mathbf{u} \\ \mathbf{0} & \mathbf{u}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{c} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ 1 \end{pmatrix}. \tag{15}$$

The main advantage of this formulation is that the left-hand side no longer requires the construction of **B**, and therefore no information is lost in the setup of the linear equation solution. This formulation is generally most useful when iterative methods are applied to the linear equation because the sparseness can be exploited in a straightforward manner. Otherwise, the dimension of this equation $(m+n+1)$ can to too large to be practical for a straightforward direct solution. However, the condition number of this expanded matrix is no better than the normal matrix formulation [15], so **c** still suffers the same loss of precision for large values of $\kappa(\mathbf{B})$ even though **B** is not constructed explicitly.

Another straightforward solution approach is to define $\mathbf{z} = (1/\lambda)\mathbf{c}$ in equation (10). The vector **z** can then be determined from the linear equation

$$\mathbf{B}\,\mathbf{z} = \mathbf{u} \tag{16}$$

λ can be computed as

$$\lambda = \frac{1}{\mathbf{u}^T\mathbf{z}} = \frac{1}{\sum_k^n z_k} \tag{17}$$

and **c** can be computed from scaling as

$$\mathbf{c} = \lambda\,\mathbf{z}. \tag{18}$$

Because of the simple form of the uno vector **u**, this procedure may also be written formally in terms of the elements of $\mathbf{B}^{-1}$ as [18]

$$\lambda = \frac{1}{\mathbf{u}^T\mathbf{B}^{-1}\mathbf{u}} = \frac{1}{\sum_{j,k}^n \mathbf{B}_{kj}^{-1}}$$
$$c_k = \frac{(\mathbf{B}^{-1}\mathbf{u})}{\mathbf{u}^T\mathbf{B}^{-1}\mathbf{u}} = \lambda \sum_j^n \mathbf{B}_{kj}^{-1}. \tag{19}$$

Equation (16) shows clearly how the condition number of **B** associated with linear equations solutions affects the accuracy of the closely related least squares problem. In exact arithmetic, this scaling approach results in the exact solution. However, in floating point arithmetic, this method suffers from the same disadvantages as the augmented normal equation solution. Namely, **B** itself can suffer loss of information, and the condition number of **B** is the square of the condition number of **E**. The first of these problems can be eliminated by replacing equation (16) with

$$\begin{pmatrix} \mathbf{1} & \mathbf{E} \\ \mathbf{E}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -\mathbf{u} \end{pmatrix} \tag{20}$$

to compute the vector **z**, but the condition number of this expanded matrix is no better than the condition number of **B**, so this method does not result in a **c** that is more accurate than the other normal equation solutions.

Because the DIIS constraint has a particularly simple form, it is straightforward to convert the system into a simpler optimization problem with elimination. For example, the constraint equation can be solved for the last coefficient c_n.

$$c_n = 1 - \sum_{k=1}^{n-1} c_k = 1 - \tilde{\mathbf{u}}^T \tilde{\mathbf{c}} \tag{21}$$

This expression can be substituted into the minimization equation, and c_n can be eliminated entirely from the optimization step.

$$\begin{aligned}\mathbf{r} &= \sum_{k=1}^{n-1} \mathbf{e}_k c_k + \mathbf{e}_n c_n \\ &= \sum_{k=1}^{n-1} \mathbf{e}_k c_k + \mathbf{e}_n (1 - \tilde{\mathbf{u}}^T \tilde{\mathbf{c}}) \\ &= \sum_{k=1}^{n-1} (\mathbf{e}_k - \mathbf{e}_n) c_k + \mathbf{e}_n \\ &= \tilde{\mathbf{E}} \tilde{\mathbf{c}} + \mathbf{e}_n.\end{aligned} \tag{22}$$

The vector $\tilde{\mathbf{c}} \equiv \mathbf{c}(1{:}n-1)$ may be determined from the (unconstrained) standard linear least squares problem

$$\min_{\tilde{\mathbf{c}}} \left\| \tilde{\mathbf{E}} \tilde{\mathbf{c}} + \mathbf{e}_n \right\|. \tag{23}$$

Once $\tilde{\mathbf{c}}$ has been found, c_n can be computed using equation (21). The relative error of this equation is [16]

$$\frac{\|\tilde{\mathbf{c}} - \tilde{\mathbf{c}}_{\text{exact}}\|}{\|\tilde{\mathbf{c}}_{\text{exact}}\|} \le \frac{p(n)\varepsilon\kappa(\tilde{\mathbf{E}})}{\cos(\theta)} \left\{ 2 + \sin(\theta)\kappa(\tilde{\mathbf{E}}) \right\} \tag{24}$$

where θ is the acute angle between $\tilde{\mathbf{E}}\tilde{\mathbf{c}}$ and $\mathbf{e}_n$, $p(n)$ is a slowly growing function of n, and $\tilde{\mathbf{c}}_{\text{exact}}$ and $\tilde{\mathbf{c}}$ are the exact and computed solutions respectively. Another error bound for this equation is [15]

$$\frac{\|\tilde{\mathbf{c}} - \tilde{\mathbf{c}}_{\text{exact}}\|}{\|\tilde{\mathbf{c}}_{\text{exact}}\|} \le \frac{\varepsilon\kappa(\tilde{\mathbf{E}})}{1 - \varepsilon\kappa(\tilde{\mathbf{E}})} \left\{ 2 + \left(\kappa(\tilde{\mathbf{E}}) + 1\right) \frac{\|\mathbf{r}\|}{\|\tilde{\mathbf{E}}\| \cdot \|\tilde{\mathbf{c}}_{\text{exact}}\|} \right\}. \tag{25}$$

Thus, if θ in equation (24) is small, or if $\|\mathbf{r}\|$ in equation (25) is small relative to $\|\tilde{\mathbf{E}}\|$, then the relative error in **c** is determined by $\kappa(\tilde{\mathbf{E}})$ rather than $\kappa(\tilde{\mathbf{E}})^2$, and an accuracy better than the normal equation solution may be achieved.

The matrix **E** may be factored as

$$\mathbf{E} = \mathbf{QR} \tag{26}$$

where $\mathbf{Q} \in \mathbb{R}^{m\times n}$ is orthogonal, $\mathbf{Q}^T\mathbf{Q} = \mathbf{1}$, and $\mathbf{R} \in \mathbb{R}^{n\times n}$ is square upper triangular, or using SVD as

$$\mathbf{E} = \mathbf{U}\sigma\mathbf{V}^T \tag{27}$$

where **U** and **V** are orthogonal and σ is the diagonal matrix of singular values. The noniterative QR factorization requires less effort than the iterative SVD, and it is employed by many of the standard library routines for least squares problems. The computed **Q** has error bounded by $mn\varepsilon/(1 - mn\varepsilon)$, and the computed **R** has error bounded by to $\kappa(\mathbf{E})\varepsilon$. If the QR factorization is computed with a sequence of orthogonal factors (e.g. with Householder reflectors), then the relative error in a column of the product **QR** is given by [15]

$$\frac{\left\| \mathbf{e}_j - (\mathbf{QR})_j \right\|}{\left\| \mathbf{e}_j \right\|} \le \sqrt{n} \frac{mn\varepsilon}{1 - mn\varepsilon}. \tag{28}$$

This unexpectedly small error is the result of cancellations in the errors in individual factors **Q** and **R**.

It is this QR factorization that is used in the standard least-squares problem, equation (23). If $\tilde{\mathbf{E}} = \mathbf{QR}$, then the DIIS solution from equation (23) is determined from the steps: (1) $\mathbf{b} = \mathbf{Q}^T\mathbf{e}_n$, (2) Solve $\mathbf{R}\tilde{\mathbf{c}} + \mathbf{b} = \mathbf{0}$, and (3) $c_n = 1 - \tilde{\mathbf{u}}^T\tilde{\mathbf{c}}$. Because **R** is triangular, the solution of the linear equation in step 2 consists of a simple sequence of back-substitution steps. The high-level LAPACK routines that solve the standard linear least squares equation using this approach are SGELSY and DGELSY in f77 and LA_GELSY in f95. A similar solution sequence holds for the SVD of $\tilde{\mathbf{E}}$, and the corresponding high-level LAPACK routines for this approach are SGELSS, DGELSS, and LA_GELSS.

Because **Q** is orthogonal, $\|\mathbf{Ec}\| = \|\mathbf{QRc}\| = \|\mathbf{Rc}\|$, and the original minimization problem equation (8) is equivalent to

$$\min_{\mathbf{c}} \|\mathbf{Rc}\| \quad \text{with } \mathbf{u}^T\mathbf{c} = 1 \tag{29}$$

The QR factorization allows the original minimization problem to be written as

$$\begin{pmatrix} \mathbf{R}^T\mathbf{R} & -\mathbf{u} \\ -\mathbf{u}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -1 \end{pmatrix} \tag{30}$$

which in turn is equivalent to

$$\begin{pmatrix} 1 & \mathbf{R} & \mathbf{0} \\ \mathbf{R}^T & \mathbf{0} & \mathbf{u} \\ \mathbf{0} & \mathbf{u}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{c} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ 1 \end{pmatrix}. \tag{31}$$

This latter linear system has dimension $2n+1$, in contrast to equation (15) which has the (usually much larger) dimension $m+n+1$. In this approach, all of the useful information in $\mathbf{E} \in \mathbb{R}^{m\times n}$ has been transferred to $\mathbf{R} \in \mathbb{R}^{n\times n}$. $\mathbf{Q}$ is not needed after the initial factorization has been computed. These equations provide an alternative formulation of the normal equations, and they suffer from the same limitations as the normal equations solution. Namely, the condition number of the linear equation in equation (31) is $\kappa(\mathbf{B})$, and this method does not result in a high-accuracy solution.

Equation (30) can also be solved by scaling using the sequence of steps (compare equations (16)–(18)).

$$\begin{aligned} &\text{Solve } \mathbf{R}^T\mathbf{y} = \mathbf{u} \\ &\text{Solve } \mathbf{R}\mathbf{z} = \mathbf{y} \\ &\text{Compute } \lambda = \frac{1}{\mathbf{u}^T\mathbf{z}} = \frac{1}{\sum_{k=1}^{n} z_k} \\ &\text{Compute } \mathbf{c} = \lambda\mathbf{z} \end{aligned} \tag{32}$$

Both of the linear equation solutions are simple back- or forward-substitution operations, and they each correspond to a local error related to $\kappa(\mathbf{R})$. No information is lost due to the computation of $\mathbf{B}$ with this two-step approach. However, the sequence of the two linear equation steps introduces an overall error related to $\kappa(\mathbf{B})$: the computed $\mathbf{y}$ has relative error $\kappa(\mathbf{R})\varepsilon$, and the computed $\mathbf{z}$ has relative error $\kappa(\mathbf{R})^2\varepsilon$ due to the error propagated from the first step. Thus, this approach does not result in a high-accuracy solution.

The $\mathbf{y}$ and $\mathbf{z}$ vectors from the sequence of two linear solutions in equations (32) is equivalent to the solution of the single linear equation

$$\begin{pmatrix} \mathbf{1} & \mathbf{R} \\ \mathbf{R}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -\mathbf{u} \end{pmatrix}. \tag{33}$$

This linear equation solution has error $\kappa(\mathbf{R})^2\varepsilon$, the same as for the above two-step sequence, so it does not result in a high-accuracy solution for $\mathbf{c}$.

Given the $\mathbf{E} = \mathbf{QR}$ factorization, the DIIS equations can also be solved by elimination. Using equation (29), we have (compare equations (21)–(23))

$$\begin{aligned} \mathbf{r} = \mathbf{R}\mathbf{c} &= \sum_{k}^{n-1} \mathbf{R}_k c_k + \mathbf{R}_n \mathbf{c}_n \\ &= \sum_{k}^{n-1} \mathbf{R}_k c_k + \mathbf{R}_n(1 - \tilde{\mathbf{u}}^T\tilde{\mathbf{c}}) \\ &= \sum_{k}^{n-1} (\mathbf{R}_k - \mathbf{R}_n) c_k + \mathbf{R}_n \\ &= \tilde{\mathbf{R}}\tilde{\mathbf{c}} + \mathbf{R}_n. \end{aligned} \tag{34}$$

The associated unconstrained optimization equation is

$$\overset{\min}{\tilde{\mathbf{c}}} \quad \left\| \tilde{\mathbf{R}}\tilde{\mathbf{c}} + \mathbf{R}_n \right\|. \tag{35}$$

Because the computed $\mathbf{R}$ has error proportional to $\kappa(\mathbf{E})\varepsilon$, and the subsequent least-squares solution amplifies this error by an additional factor of (at least) $\kappa(\mathbf{E})$, this approach also does not result in a solution with high accuracy.

If the constraint equation (5) is not required to be satisfied exactly, then this equation may be added to the least-squares equation.

$$\overset{\min}{\mathbf{c}} \quad \left\| \begin{pmatrix} \mathbf{E} \\ w\mathbf{u}^T \end{pmatrix} \mathbf{c} - \begin{pmatrix} \mathbf{0} \\ w \end{pmatrix} \right\|. \tag{36}$$

With exact arithmetic, the limit of this equation as the weight factor $w \to \infty$ is the exact solution of the DIIS equation. However, the condition number of the augmented matrix increases in this limit, thereby limiting the ultimate accuracy of the computed $\mathbf{c}$. In practice, there is a tradeoff between smaller values of w, for which the constraint is not satisfied exactly, and larger values of w for which the condition number limits the accuracy of the solution. If χ^2 is the residual square and if $\alpha = 1/\mathbf{u}^T\mathbf{c}$ from equation (36), then the following sequence of inequalities holds.

$$\|\mathbf{Ec}\|^2 \le \chi^2 \le \lambda_{\text{exact}} \le \alpha^2 \|\mathbf{Ec}\|^2 \tag{37}$$

where the vector $\alpha\mathbf{c}$ is the computed vector scaled to satisfy the constraint exactly. $\alpha > 1$ in exact arithmetic because the computed $\mathbf{c}$ satisfies $\mathbf{u}^T\mathbf{c} < 1$. These bounds may be used to choose appropriate values of weight factors w, and equations (24) and (25) bound the error due to the condition number.

All of the equation solutions discussed above are candidates for improved accuracy through iterative refinement. Consider, for example, the substitution of

$$\mathbf{c} = \mathbf{c}_0 + \mathbf{c}_1 \tag{38}$$

for some available reference vector $\mathbf{c}_0$ into equation (8), giving the equation

$$\overset{\min}{\mathbf{c}_1} \quad \|\mathbf{E}\mathbf{c}_1 + (\mathbf{E}\mathbf{c}_0)\| \quad \text{with } \mathbf{u}^T\mathbf{c}_1 = (1 - \mathbf{u}^T\mathbf{c}_0). \tag{39}$$

This is another standard linear least squares optimization equation with a single linear equality constraint (LSE). Once $\mathbf{c}_1$ has been computed, then the reference vector $\mathbf{c}_0$ may be updated to define a new optimization equation, and the process can be repeated until convergence is achieved. The QR factorization of $\mathbf{E}$ can be reused for each step of this process, and it is only the vector $\mathbf{E}\mathbf{c}_0$ and the updated constraint $(1-\mathbf{u}^T\mathbf{c}_0)$ that changes. In the typical application of iterative refinement, these quantities are computed in extended precision, but in some cases improved accuracy can be achieved with fixed-precision arithmetic [15, 19, 20]. Iterative refinement based on the substitution of equation (28) may be applied to all of the above DIIS solution methods including equations (9)–(12), (16)–(18), (23), (29), (30)–(33), and (36)

With a typical least-squares problem, for example fitting of data to some functional form, the matrix $\mathbf{E}$ is expected to be nonsingular. In fact, singularity usually implies some underlying problem with either or both the data or the fitting model. This is not necessarily true for the DIIS problem because the error vectors can be linearly dependent even when the input points $\mathbf{x}_k$ are distinct. Consider, for example, the interpolation of a function for which $\mathbf{F}(\mathbf{x}_0 - \mathbf{x}) = -\mathbf{F}(\mathbf{x}_0 + \mathbf{x})$ for some (unknown) symmetry point $\mathbf{x}_0$. Suppose two input points are unknowingly related by $\mathbf{x}_k = \mathbf{x}_0 - \mathbf{y}$ and $\mathbf{x}_j = \mathbf{x}_0 + \mathbf{y}$. For the odd function under consideration, this would result in $\mathbf{e}_k = -\mathbf{e}_j$, and the matrix $\mathbf{E}$ will be singular, the matrix $\mathbf{B}$ will be singular, and all of the various formulations of the normal equations will have an infinite number of solutions. We also see that $\kappa(\mathbf{E}) = \infty$, $\kappa(\mathbf{R}) = \infty$, and $\kappa(\mathbf{B}) = \infty$, so even if this exact singularity were not detected, or hidden by numerical noise, any computed result would be meaningless. The 'correct' answer to the DIIS equations in such a case is $c_k = c_j = 1/2$. This would result in $\|\mathbf{r}\|^2 = 0$, and in an interpolation $\mathbf{x}_{\text{DIIS}} = (1/2)\mathbf{x}_k + (1/2)\mathbf{x}_j = \mathbf{x}_0$, which is the correct solution for such a function. The linear combination, $c_k = -c_j$, corresponds to the null space for this problem. Other forms of linear dependence in $\mathbf{E}$ can also occur, and they may involve multiple columns which might not be obvious to the observer from casual inspection. Consequently, it is desirable to have a solution approach that recognizes these situations and handles them appropriately. For the two factorizations we discuss above, SVD and QR, linear dependence in $\mathbf{E}$ can be detected. In the case of SVD, a threshold tolerance τ can be specified such that singular values smaller than the product $\sigma_{\max}\,\tau$ can be ignored. In general τ should reflect the accuracy with which the columns of $\mathbf{E}$ can be computed, and this should always be larger than ε. The columns of the matrices $\mathbf{U}$ and $\mathbf{V}$ associated with these small singular values should be ignored, and the rest of the computation should proceed as usual within this numerically computed active subspace. A similar rank detection also applies to the computationally cheaper QR factorization approaches; the columns of $\mathbf{E}$ are permuted during the factorization process by selecting numerically large values as pivots. In the case that numerical linear dependence is detected, based on the tolerance parameter τ, a relatively cheap *full orthogonal factorization* is performed on $\mathbf{R}$. This results in an upper triangular $\mathbf{R}$ in which the last column(s) are zero [16], and the remaining calculation steps can ignore these columns and work just within the numerically computed active subspace. In both cases, SVD and QR, the computed results will be those that are orthogonal to the null space; these unique solutions are the minimum-norm solutions. In most applications of DIIS, it is exactly these minimum-norm solutions that are required. In other cases, the minimum-norm solution can be combined with the appropriate null space vector components in order to satisfy any additional constraints or boundary conditions.

3. Examples

In this section we discuss a few examples of the DIIS method. The first example is application to a simple one-dimensional function $F(x)$ with the exact solution $F(x_*) = 0$. With $n = 2$, we have $e_1 = F(x_1)$ and $e_2 = F(x_2)$. The DIIS equations may be solved in closed form with the substitution and replacement approach. Following equations (21)–(23), we have $\mathbf{E} = (e_1, e_2)$, $\tilde{E} = e_1 - e_2$, and $r = \tilde{E}c_1 - e_2$. r^2 is minimized by $c_1 = -e_2/\tilde{E} = e_2/(e_2 - e_1)$, and the constraint gives $c_2 = (1 - c_1) = -e_1/(e_2 - e_1)$. The interpolated input value is $x_{\text{DIIS}} = x_1c_1 + x_2c_2 = (e_2x_1 - e_1x_2)/(e_2 - e_1)$. Compare this to figure 1. A linear interpolation between the two points (x_1, e_1) and (x_2, e_2) gives

$$F^{[1]}(x) = \left(\frac{e_1x_2 - e_2x_1}{x_2 - x_1}\right) + \left(\frac{e_2 - e_1}{x_2 - x_1}\right)x. \tag{40}$$

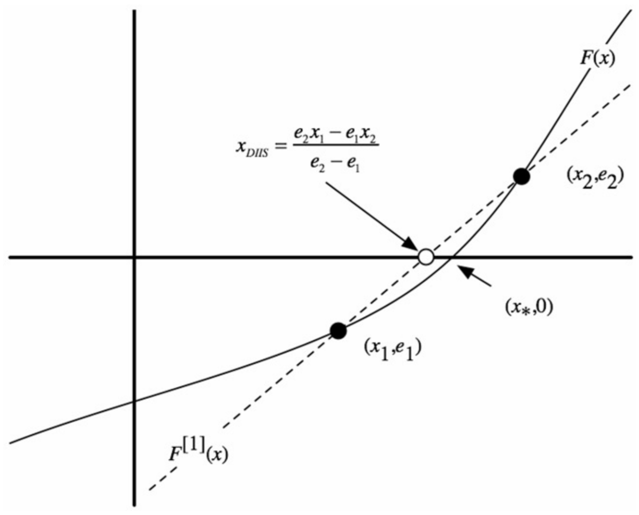

Figure 1. DIIS solution to a one-dimensional function $F(x)$. The DIIS solution $x_{\rm DIIS}$ approximates the exact solution $\mathbf{x}_*$ and is equivalent to the method of false position.

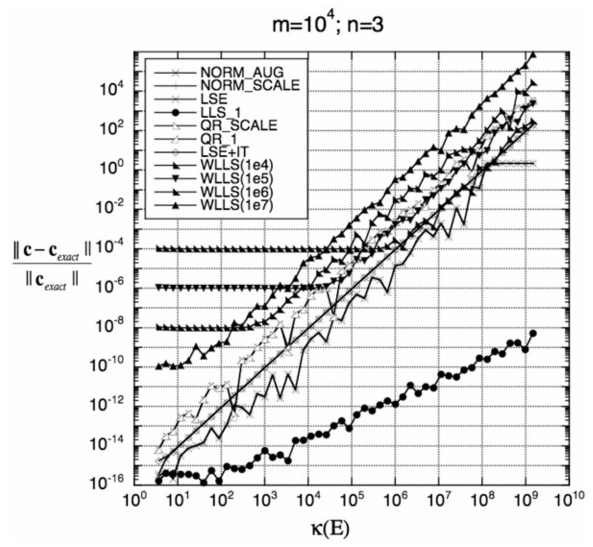

Figure 2. Relative error in $\mathbf{c}$ as a function of the condition number $\kappa(\mathbf{E})$ for various solution methods to the DIIS equation with dimensions $m = 10^4$ and $n = 3$. See the text for the details of the various solution methods.

Solving this approximating function for $\mathrm{F}^{[1]}(x) = 0$ gives the approximate root $\bar{x} = (e_2x_1 - e_1x_2)/(e_2 - e_1)$, which is exactly the DIIS interpolation value. We conclude from this simple example that the DIIS interpolation step may be regarded as a multidimensional generalization of the well-known *method of false position*. We note that when $e_1$and e_2 have the same sign, then the two coefficients will have opposite signs, one or both coefficients will be larger than one in magnitude, and the interpolated root will be outside of the range $[x_1, x_2]$; that is, in these situations, the DIIS method will *extrapolate* to an approximate root rather than *interpolate* to a root. The same general considerations apply to the multidimensional DIIS method; large values of $\|\mathbf{c}\|$ imply extrapolation to a point far from the input $\mathbf{x}_k$ points, and such large-step extrapolations are typically less reliable than interpolations or smaller extrapolations. An important difference between the above 1-*D* example and the general case is that the 1-*D* application always interpolates (or extrapolates) to a point at which $r^2 = 0$, whereas the multidimensional case extrapolates typically to a point at which $\lambda = \mathbf{r}^2 > 0$. There are several situations in which the method of false position converges slowly even when each step is an interpolation. These situations involve typically cases where the linear approximation is a poor representation of the true function. However, there are also situations in which the linear interpolation results in accurate root estimates despite the fact that the higher derivatives are large within the interpolation range. It would be expected that the general multidimensional DIIS method would share these same strengths and weaknesses.

We examine next a particular multidimensional DIIS problem that has a closed-form solution, and the accuracy of several of the methods discussed in the previous section may be compared. This model problem is defined for a given m and n according to

$$E_{jk} = 1 + \delta_{jk}\Delta \quad \text{for } j = 1, \ldots, m, \quad \text{and} \quad k = 1, \ldots, n \tag{41}$$

in terms of the scalar parameter Δ. It may be verified that the exact solution is given by

$$\begin{aligned} c_k &= \frac{1}{n} \\ \lambda &= m + 2\Delta + \frac{\Delta^2}{n} \end{aligned} \tag{42}$$

and that $\kappa(\mathbf{E}) \to \infty$ as the parameter $\Delta \to 0$. All of the solution methods discussed in the previous section result in acceptably accurate solutions when the condition number is small $\kappa \approx 1$. We therefore focus on the more difficult situations with large condition numbers.

Figure 2 shows the relative error $\|\mathbf{c} - \mathbf{c}_{\rm exact}\|/\|\mathbf{c}_{\rm exact}\|$ as a function of $\kappa(\mathbf{E})$ for $m = 10^4$, $n = 3$. For this $\mathbf{E}$ matrix, values of Δ were chosen such that $\kappa(\mathbf{E})$ ranges from 10^0 to 10^{10}. Normally, $\kappa(\mathbf{E})$ would not be computed exactly, it would be estimated in various ways as discussed in the previous section, but for the purposes of this graph the exact condition number is computed from the (relatively expensive)

SVD factorization. When the relative error approaches unity, there are no correct significant digits in the computed result, and the result has no practical value; however, the errors are plotted beyond this point regardless to show the trends of the various methods.

The augmented normal equation solution (equation (12), denoted NORM_AUG) is almost indistinguishable from the scaled normal equation solution (equations (16)–(18), denoted NORM_SCALE), and both clearly scale as $\kappa(\mathbf{E})^2$ as expected from the discussion in the previous section.

The least squares solution with equality constraints (equation (13), denoted LSE) also scales as $\kappa(\mathbf{E})^2$; this is because this model problem corresponds to a large-norm $\|\mathbf{r}\|/\|\mathbf{E}\|$ situation (see equation (14)). The LSE solution has slightly larger errors than the NORM_AUG and NORM_SCALE solutions. This is because the symmetric positive-definite NORM_SCALE equation may be solved using the accurate Cholesky factorization method, and the symmetric indefinite NORM_AUG equation may be solved using the accurate Bunch-Kaufman factorization method [16]. If these same equations were solved with general linear equation methods that did not exploit the symmetry of the matrices (e.g. such as LU factorization), then the computed results would show slightly larger errors. Furthermore, larger errors would be observed if the augmented normal equations were written and solved in one of the mathematically equivalent nonsymmetric forms such as

$$\begin{pmatrix} \mathbf{B} & -\mathbf{u} \\ \mathbf{u}^T & 0 \end{pmatrix}\begin{pmatrix} \mathbf{c} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}. \tag{43}$$

Iterative refinement for the LSE solution (equation (39), denoted LSE + IT) is also shown in figure 2, and it seen to have minimal effect on the accuracy. The iterative refinement is performed in the same precision as the solution; extended precision might show improved accuracy [20].

For the substitution and elimination solution (equation (23), denoted LLS_1), the error is seen to scale with nearly unit slope as $\kappa(\mathbf{E})$ increases, and it is clearly more accurate throughout the range of $\kappa(\mathbf{E})$ than the other solutions that scale as $\kappa(\mathbf{E})^2$. Even for small condition numbers, the LLS_1 solution can be several orders of magnitude more accurate than the other methods. If the DIIS method were being used for very tight convergence, then clearly such a method would be preferred to the other less accurate solutions. The curve in figure 2 is the result of the substitution of $\mathbf{e}_1$ and elimination of the c_1 coefficient. Substitution and elimination of other error vectors results in indistinguishable curves, but that observation is possibly the result of the underlying symmetry in this model problem. In the general case with arbitrary sets of error vectors, it remains an open question if the computed error can be optimized with an appropriate transformation of $\mathbf{E}$ and $\mathbf{c}$ followed by substitution and elimination in this transformed representation.

Four different weighted solutions (equation (36), denoted WLLS) are shown corresponding to $w = 10^4$, 10^5, 10^6, and 10^7. For small $\kappa(\mathbf{E})$, the accuracy of the computed $\mathbf{c}$ depends on w, with larger w corresponding to smaller errors as expected. This error remains relatively constant as $\kappa(\mathbf{E})$ increases until a critical value is reached, at which time the relative error becomes proportional to $\kappa(\mathbf{E})^2$. At no point are any of the weighted solutions more accurate than the normal equation solution. At least for this model problem, the weighted LLS approach appears to offer no advantages.

Figure 2 also shows the results of the QR solution with scaling (equation (32), denoted QR_SCALE) and of the QR solution with substitution and elimination (equation (35), denoted QR_1). In the latter case the vector $\mathbf{R}_1$ was substituted and the variable c_1 was eliminated. Other vectors were also examined, but the computed results were essentially identical. This is possibly a consequence of the symmetry in this model problem, and it is an open question if a transformation may be imposed on $\mathbf{E}$ and $\mathbf{c}$ in order to optimize the substitution vector and to minimize the computed error. However, from the discussion in the previous section, it seems unlikely that such a method would show $\kappa(\mathbf{E})$ error rather than the observed $\kappa(\mathbf{E})^2$ error for either the QR_SCALE or the QR_1 approach. At no point are either of the QR_SCALE or the QR_1 solutions more accurate than the normal equation solution. At least for this model problem, these approaches appear to offer no advantages.

Figure 3 shows the same set of error curves for a larger model problem with $m = 10^6$ and $n = 10$. For this $\mathbf{E}$ matrix, values of Δ were chosen such that $\kappa(\mathbf{E})$ ranges from 10^0 to 10^{10}. The error curves are qualitatively the same as for the previous example, and these results are presented primarily to demonstrate that similarity. In particular, the LLS_1 curve scales as $\kappa(\mathbf{E})$ and is significantly better than any other method. Among the curves that scale as $\kappa(\mathbf{E})^2$, the NORM_AUG and the NORM_SCALE curves are slightly better than the other solution methods.

4. Conclusions

The DIIS method is a commonly used convergence acceleration technique. A simple geometrical interpretation has been given for this method that involves

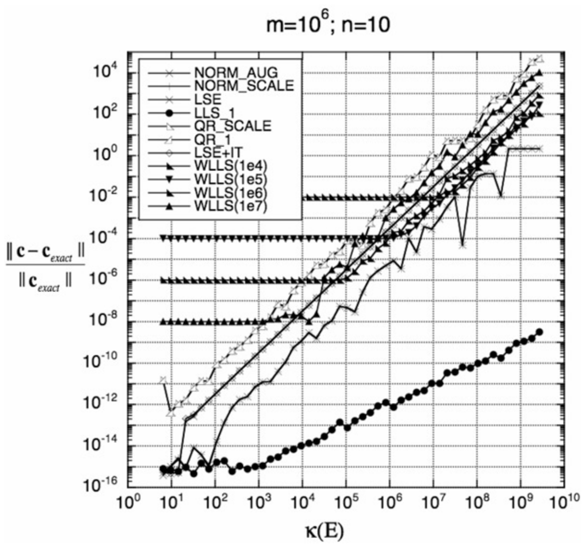

Figure 3. Relative error in **c** as a function of the condition number $\kappa(\mathbf{E})$ for various solution methods to the DIIS equation with dimensions $m = 10^6$ and $n = 10$. See the text for the details of the various solution methods.

minimization of a function value within a particular multidimensional plane. Traditionally, the solution of the DIIS equations has been formulated almost always in terms of the augmented normal equations. Over twenty other ways to solve the DIIS equations have been introduced and discussed in this manuscript. Some of these solution methods are inherently inaccurate and are never recommended, others have some efficiency or accuracy advantages that may be exploited in different situations. Two of these ways, linear least squares with substitution and elimination, and linear least squares with equality constraint, have the potential to produce solutions that are significantly more accurate than the traditional normal equation solutions. The choice of the most accurate solution approach for a given problem depends on various vector and matrix norm relations which have been presented. The advantage of one of these methods has been demonstrated on a model problem which has a simple closed-form solution.

Acknowledgments

This work was supported by the Office of Basic Energy Sciences, Division of Chemical Sciences, Geosciences, and Biosciences, U.S. Department of Energy, and in part through the SciDAC project "Advanced Software for the Calculation of Thermochemical, Kinetics, and Dynamics," under contract number DE-AC02-06CH11357.

References

[1] P. Pulay, Chem. Phys. Lett. **73,** 393 (1980).
[2] P. Pulay, J. Comput. Chem. **3,** 556 (1982).
[3] R. P. Muller, J.-M. Langlois, M. N. Ringnalda, R. A. Friesner, and W. A. Goddard III, J. Chem. Phys. **100,** 1226 (1994).
[4] J.-M. Langlois, T. Yamasaki, R. P. Muller, and W. A. Goddard III, J. Phys. Chem. **98,** 13498 (1998).
[5] L. Thogersen, J. Olsen, A. Kohn, P. Jorgensen, P. Salek, and T. Halgaker, J. Chem. Phys. **123,** 074103 (2005).
[6] M. Kawata, C. M. Cortis, and R. A. Friesner, J. Chem. Phys. **108,** 4426 (1998).
[7] T. van Voorhis and M. Head-Gordon, Mol. Phys. **100,** 1713 (2002).
[8] B. D. Dunietz, T. van Voorhis, and M. Head-Gordon, J. Theo. Comput. Chem. **1,** 255 (2005).
[9] J. E. Subotnik, Y. Shao, W. Liang, and M. Head-Gordon, J. Chem. Phys. **121,** 9220 (2004).
[10] P. Csaszar and P. Pulay, J. Mol. Struct. **114,** 31 (1984).
[11] T. H. Fischer and J. Almlof, J. Phys. Chem. **96,** 9768 (1992).
[12] F. Eckert, P. Pulay, and H.-J. Werner, J. Comput. Chem. **18,** 1473 (1997).
[13] P. Pulay Direct inversion in iterative subspaces (DIIS), in *Molecular Quantum Mechanics: Analytic Gradients and Beyond*, edited by A. G. Csaszar, G. Fogarasi, H. F. Schaefer III, and P. G. Szalay (ELTE Institute of Chemistry, Budapest, 2007), pp. 71–73.
[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. (Society for Industrial and Applied Mathematics, Philadelphia, 2003).
[15] N. J. Higham, *Accuracy and Stability of Numerical Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia, 2002).
[16] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. (Society for Industrial and Applied Mathematics, Philadelphia, 1999).
[17] V. A. Barker, L. L. Blackford, J. Dongarra, J. Du Croz, S. Hammarling, M. Marinova, J. Wasniewski, and P. Yalamov, *LAPACK95 Users' Guide* (Society for Industrial and Applied Mathematics, Philadelphia, 2001).
[18] J. Olsen Direct inversion in iterative subspaces (DIIS), in *Molecular Quantum Mechanics: Analytic Gradients and Beyond*, edited by A. G. Csaszar, G. Fogarasi, H. F. Schaefer III, and P. G. Szalay (ELTE Institute of Chemistry, Budapest, 2007), pp. 85–89.
[19] Å. Björck, *Numerical Methods for Least Squares Problems* (Society for Industrial and Applied Mathematics, Philadelphia, 2002) (Society for Industrial and Applied Mathematics, Philadelphia, 1996).
[20] J. Demmel, Y. Hida, W. Kahan, X. Li, S. Mukherjee, and E. Riedy, ACM Transactions on Mathematical Software **32,** 325 (2006).