 CMES, vol.80, no.4, pp.275-298, 2011

An Iterative Method Using an Optimal Descent Vector, for Solving an Ill-Conditioned System Bx = b, Better and Faster than the Conjugate Gradient Method

Chein-Shan Liu[1,2] **and Satya N. Atluri**[1]

Abstract: To solve an ill-conditioned system of linear algebraic equations (LAEs): $\mathbf{Bx}-\mathbf{b}=\mathbf{0}$, we define an invariant-manifold in terms of $\mathbf{r}:=\mathbf{Bx}-\mathbf{b}$, and a monotonically increasing function $Q(t)$ of a time-like variable t. Using this, we derive an evolution equation for $d\mathbf{x}/dt$, which is a system of Nonlinear Ordinary Differential Equations (NODEs) for $\mathbf{x}$ in terms of t. Using the concept of discrete dynamics evolving on the invariant manifold, we arrive at a purely iterative algorithm for solving $\mathbf{x}$, which we label as an *Optimal Iterative Algorithm* (OIA) involving an *Optimal Descent Vector* (ODV). The presently used ODV is a modification of the Descent Vector used in the well-known and widely used Conjugate Gradient Method (CGM). The presently proposed OIA/ODV is shown, through several examples, to converge faster, with better accuracy, than the CGM. The proposed method has the potential for a wide-applicability in solving the LAEs arising out of the spatial-discretization (using FEM, BEM, Trefftz, Meshless, and other methods) of Partial Differential Equations.

Keywords: Linear algebraic equations, Ill-conditioned linear system, Conjugate Gradient Method (CGM), Optimal Iterative Algorithm with an Optimal Descent Vector (OIA/ODV), Invariant-manifold, Linear PDE

1 Introduction

In this paper we propose a simple, and easy-to-implement iterative algorithm, to solve the following system of Linear Algebraic Equations (LAEs):

$$\mathbf{Bx}=\mathbf{b}, \tag{1}$$

where $\mathbf{x}\in\mathbb{R}^n$ is an unknown vector, to be determined from a given coefficient matrix $\mathbf{B}\in\mathbb{R}^{n\times n}$ (which might be unsymmetric), and the input $\mathbf{b}\in\mathbb{R}^n$. We first convert

[1] Center for Aerospace Research & Education, University of California, Irvine

[2] Department of Civil Engineering, National Taiwan University, Taipei, Taiwan. E-mail: liucs@ntu.edu.tw

Eq. (1) to a system of nonlinear ODEs for $\mathbf{x}(t)$ where t is a time-like parameter, and then derive an *optimal iterative algorithm*, for solving these nonlinear ODEs, for $\mathbf{x}(t)$. This purely iterative algorithm for solving for $\mathbf{x}$ is shown to converge much faster, and with better accuracy, than the widely-used Conjugate Gradient Method (CGM). It may be pointed that Eq. (1) may result from the spatial-discretization (using FEM, BEM, Trefftz, Meshless, or other methods) of linear partial differential equations (PDEs).

A measure of the ill-posedness of Eq. (1) is the condition number of $\mathbf{B}$:

$$\text{cond}(\mathbf{B}) = \|\mathbf{B}\|\|\mathbf{B}^{-1}\|, \tag{2}$$

where $\|\mathbf{B}\|$ is the Frobenius norm of $\mathbf{B}$:

$$\|\mathbf{B}\| = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} B_{ij}^2}, \tag{3}$$

with B_{ij} denoting the ij-th component of $\mathbf{B}$. For arbitrary $\varepsilon > 0$, there exists a matrix norm $\|\mathbf{B}\|$ such that $\rho(\mathbf{B}) \le \|\mathbf{B}\| \le \rho(\mathbf{B}) + \varepsilon$, where $\rho(\mathbf{B})$ is a radius of the spectrum of $\mathbf{B}$. Therefore, the condition number of $\mathbf{B}$ can be estimated by

$$\text{cond}(\mathbf{B}) = \frac{\max_{\sigma(\mathbf{B})} |\lambda|}{\min_{\sigma(\mathbf{B})} |\lambda|}, \tag{4}$$

where $\sigma(\mathbf{B})$ is the set of all the eigenvalues of $\mathbf{B}$.

Instead of Eq. (1), we can solve a normal linear system:

$$\mathbf{C}\mathbf{x} = \mathbf{b}_1, \tag{5}$$

where

$$\begin{aligned} &\mathbf{b}_1 = \mathbf{B}^{\mathrm{T}}\mathbf{b}, \\ &\mathbf{C} = \mathbf{B}^{\mathrm{T}}\mathbf{B} > \mathbf{0}. \end{aligned} \tag{6}$$

We consider an iterative method for solving Eq. (5) and define, for any vector $\mathbf{x}_k$, the descent vector

$$\mathbf{R}_k := \mathbf{C}\mathbf{x}_k - \mathbf{b}_1. \tag{7}$$

Ascher, van den Doel, Hunag and Svaiter (2009), and also Liu and Chang (2009) have viewed the gradient descent method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{R}_k, \tag{8}$$

as a forward Euler scheme of the linear system of ODEs:

$$\dot{\mathbf{x}} = \mathbf{b}_1 - \mathbf{C}\mathbf{x}. \tag{9}$$

The absolute stability bound

$$\alpha_k \leq \frac{2}{\max_{\sigma(\mathbf{C})} \lambda} \tag{10}$$

must be obeyed if a uniform stepsize is employed.

Specifically, Eq. (8) presents a steepest descent method, if

$$\alpha_k = \frac{\|\mathbf{R}_k\|^2}{\mathbf{R}_k^{\mathrm{T}}\mathbf{C}\mathbf{R}_k}. \tag{11}$$

$\mathbf{R}_k$ is now a stepwise steepest descent direction. There are methods that converge significantly faster than the steepest descent method (SDM); unlike the conjugate gradient method (CGM), they insist their search directions to be the gradient vector at each iteration [Barzilai and Borwein (1988); Friedlander, Martinez, Molina and Raydan (1999); Raydan and Svaiter (2002); Dai and Yuan (2003); Dai, Hager, Schittkowsky and Zhang (2006); Liu (2011)]. The SDM performs poorly, yielding iteration counts that grow linearly with cond$(\mathbf{C})$ [Akaike (1959); Forsythe (1968); Nocedal, Sartenar and Zhu (2002)]. The well-known slowness of SDM has to do with the choice of the gradient descent direction $\mathbf{R}_k$ as well as the stepsize α_k given above. Recently, Liu (2011) has derived a relaxed steepest descent method, to accelerate the convergence. However, up to now *no one expects any of the gradient descent methods to ever perform* **better** *than the conjugate gradient method (CGM)* for the numerical solution of Eq. (5). In this paper we explore a variant of the SDM by developing the concept of an Optimal Iterative Algorithm driven by an "Optimal Descent Vector", to solve a system of ODEs for $\mathbf{x}$ in a time-like parameter t, on the invariant-manifold. Here we will modify the direction $\mathbf{R}_k$ as well as the stepsize α_k from a theoretical foundation. This novel method performs **better** than CGM, and of course **better** than all the previous variants of the gradient descent methods.

The remaining parts of this paper are arranged as follows. In Section 2 we start from an invariant-manifold to derive a system of nonlinear ODEs for the numerical solution of Eq. (1). Then, a genuine dynamics on the invariant-manifold is constructed in Section 3, resulting in an optimal vector control algorithm in terms of a weighting factor which is optimized explicitly. Section 4 is devoted to reformulating the matrix type linear equations of PDE into a vector-form linear equations. The numerical examples are given in Section 5 to display some advantages of the newly developed Optimal Iterative Algorithm (OIA) [involving an Optimal Descent

Vector (ODV)], which is compared with the CGM. Finally, some conclusions are drawn in Section 6.

2 An invariant-manifold, depending on r and a monotonically increasing function $Q(t)$

2.1 A motivation

Because we aim to solve a system of LAEs in Eq. (1), the following uncoupled algebraic system is a good example to demonstrate the present approach:

$$x-1=0,\ \ y-1=0. \tag{12}$$

The above two equations can be combined into a single one:

$$\frac{1}{2}\|\mathbf{r}\|^2=\frac{1}{2}[(x-1)^2+(y-1)^2]=0, \tag{13}$$

where $\mathbf{r}$ is a residual vector. We can see that it is a circle in the spatial-plane (x,y) with a center $(1,1)$ but with a zero radius.

Now, we introduce a parameter t, which is a time-like variable, and let x and y be functions of t. We consider the following equation, defined in the space-time domain (x,y,t):

$$\frac{1}{2}e^{2\alpha t}\|\mathbf{r}\|^2=\frac{1}{2}e^{2\alpha t}[(x-1)^2+(y-1)^2]=C, \tag{14}$$

where $\alpha>0$ and C is a constant determined by the initial values of $x(0)=x_0$ and $y(0)=y_0$,

$$C=\frac{1}{2}[(x_0-1)^2+(y_0-1)^2].$$

Eq. (14) can be written as

$$(x-1)^2+(y-1)^2=2Ce^{-2\alpha t}, \tag{15}$$

which is a manifold in the space-time domain (x,y,t), each cross-section at a fixed t being a circle with the center $(1,1)$ and with a radius $\sqrt{2Ce^{-2\alpha t}}$. We can construct the following system of ODEs:

$$\dot{x}=-\alpha(x-1),\ \ \dot{y}=-\alpha(y-1), \tag{16}$$

such that the path of $(x(t),y(t))$ generated from the above ODEs is located on the manifold (15).

Solving Eq. (16) we have

$$x(t) = [x_0 - 1]e^{-\alpha t} + 1, \;\; y(t) = [y_0 - 1]e^{-\alpha t} + 1. \tag{17}$$

Inserting them into Eq. (15) we indeed can prove that $(x(t), y(t))$ is located on the manifold, and $(x(t), y(t))$ can fast tend to the solution $(x, y) = (1, 1)$ when t increases.

The idea behind this approach is that we can construct a suitable invariant-manifold (15), and derive a system of ODEs on this manifold. Then the path will approach to the solution, when we solve the system of ODEs. Usually, we do not have a closed-form solution of the system of ODEs for the general system of linear equations; however, a numerical integration method can help us to obtain an approximate solution.

2.2 A space-time manifold

For the system (1) of LAEs, which is expressed in terms of the residual vector:

$$\mathbf{r} = \mathbf{B}\mathbf{x} - \mathbf{b} \tag{18}$$

with $\mathbf{r} = \mathbf{0}$ in the final solution, we can, as inspired by Eq. (14), formulate a scalar homotopy function:

$$h(\mathbf{x}, t) = \frac{1}{2} Q(t) \|\mathbf{r}(\mathbf{x})\|^2 - \frac{1}{2} \|\mathbf{r}(\mathbf{x}_0)\|^2 = 0, \tag{19}$$

where we let $\mathbf{x}$ to be a function of a time-like variable t, with initial value $\mathbf{x}(0) = \mathbf{x}_0$. We expect $h(\mathbf{x}, t) = 0$ to be an invariant-manifold in the space-time domain $(\mathbf{x}, t)$ for a dynamical system $h(\mathbf{x}(t), t) = 0$ to be specified further. When $Q > 0$, the manifold defined by Eq. (19) is continuous and differentiable, and thus the following differential operation carried out on the manifold makes sense. As a consequence of the "consistency condition", we have

$$\frac{1}{2} \dot{Q}(t) \|\mathbf{r}(\mathbf{x})\|^2 + Q(t) \mathbf{R} \cdot \dot{\mathbf{x}} = 0, \tag{20}$$

which is obtained by taking the differential of Eq. (19) with respect to t and considering $\mathbf{x} = \mathbf{x}(t)$, where corresponding to $\mathbf{r}$ in Eq. (18),

$$\mathbf{R} := \mathbf{B}^{\mathrm{T}} \mathbf{r} \tag{21}$$

is a descent vector.

We suppose that the evolution of $\mathbf{x}$ is driven by a vector $\mathbf{u}$:[1]

$$\dot{\mathbf{x}} = \lambda \mathbf{u}, \tag{22}$$

where

$$\mathbf{u} = \mathbf{R} + \alpha \mathbf{r} = (\mathbf{B}^{\mathrm{T}} + \alpha \mathbf{I}_n)\mathbf{r} \tag{23}$$

[where $\mathbf{I}_n$ is the $n \times n$ diagonal-unit matrix] is a suitable combination of the descent vector $\mathbf{R}$ and the weighted residual vector $\alpha\mathbf{r}$. *As mentioned in Section 1, here we have modified the steepest descent direction* $\mathbf{R}$ *as used in SDM to* $\mathbf{u}$ *as our search direction for the solution of* $\mathbf{x}$.

Inserting Eq. (22) into Eq. (20) we can derive

$$\dot{\mathbf{x}} = -q(t)\frac{\|\mathbf{r}\|^2}{\mathbf{r}^{\mathrm{T}}\mathbf{v}}\mathbf{u}, \tag{24}$$

where

$$\mathbf{A} := \mathbf{B}\mathbf{B}^{\mathrm{T}}, \tag{25}$$

$$\mathbf{v} := \mathbf{B}\mathbf{u} = \mathbf{v}_1 + \alpha \mathbf{v}_2 = \mathbf{A}\mathbf{r} + \alpha \mathbf{B}\mathbf{r}, \tag{26}$$

$$q(t) := \frac{\dot{Q}(t)}{2Q(t)}. \tag{27}$$

Hence, in our algorithm, if $Q(t)$ can be guaranteed to be a monotonically increasing function of t, we have an absolutely convergent property in solving the linear equations system (1):

$$\|\mathbf{r}(\mathbf{x})\|^2 = \frac{C}{Q(t)}, \tag{28}$$

where

$$C = \|\mathbf{r}(\mathbf{x}_0)\|^2 \tag{29}$$

is determined by the initial value $\mathbf{x}_0$. We do not need to specify the function $Q(t)$ a priori, but $\sqrt{C/Q(t)}$ merely acts as a measure of the residual error of $\mathbf{r}$ in time. Hence, we impose in our algorithm that $Q(t) > 0$ is a monotonically increasing function of t. When t is increased to a large value, the above equation will enforce the residual error $\|\mathbf{r}(\mathbf{x})\|$ to tend to zero, and meanwhile the solution of Eq. (1) is obtained approximately. However, it is still a challenge to develop a suitable

[1] Note that $\mathbf{u}$ is not a vector along the *normal* to the hyper-surface $h(\mathbf{x},t) = 0$

numerical integrator for Eq. (24), such that the orbit of $\mathbf{x}$ can really remain on the invariant-manifold (28).

Notice that the dynamical system in Eq. (24) is time-dependent and nonlinear, which is quite different from that of the so-called Dynamical Systems Method (DSM), which was previously developed by Ramm (2007, 2009), Hoang and Ramm (2008, 2010), and Sweilam, Nagy and Alnasr (2009). The system of ODEs which appears in the DSM for solving linear problems is time-independent and linear. Here, we may stress the importance of the concept of *invariant-manifold*; without which we cannot derive a nonlinear ODEs system to govern the solution of $\mathbf{x}$. The effectiveness of the *invariant-manifold* will be further explored in the following section.

3 Dynamics on the invariant-manifold $h(\mathbf{x},t)=0$

3.1 Discretizing in time, yet keeping x *on the manifold*

Now we discretize the foregoing continuous time dynamics (24) into a discrete time dynamics by applying the forward Euler scheme:

$$\mathbf{x}(t+\Delta t)=\mathbf{x}(t)-\beta\frac{\|\mathbf{r}\|^2}{\mathbf{r}^{\mathrm{T}}\mathbf{v}}\mathbf{u}, \tag{30}$$

where

$$\beta=q(t)\Delta t \tag{31}$$

is the steplength. Correspondingly, $\mathbf{u}$ is a search direction endowed with a stepsize $\beta\|\mathbf{r}\|^2/(\mathbf{r}^{\mathrm{T}}\mathbf{v})$.

In order to keep $\mathbf{x}$ on the manifold (28) we can consider the evolution of $\mathbf{r}$ along the path $\mathbf{x}(t)$ by

$$\dot{\mathbf{r}}=\mathbf{B}\dot{\mathbf{x}}=-q(t)\frac{\|\mathbf{r}\|^2}{\mathbf{r}^{\mathrm{T}}\mathbf{v}}\mathbf{v}. \tag{32}$$

Similarly we use the forward Euler scheme to integrate Eq. (32), obtaining

$$\mathbf{r}(t+\Delta t)=\mathbf{r}(t)-\beta\frac{\|\mathbf{r}\|^2}{\mathbf{r}^{\mathrm{T}}\mathbf{v}}\mathbf{v}, \tag{33}$$

which by taking the square-norms of both sides and using Eq. (28) we can obtain

$$\frac{C}{Q(t+\Delta t)}=\frac{C}{Q(t)}-2\beta\frac{C}{Q(t)}+\beta^2\frac{C}{Q(t)}\frac{\|\mathbf{r}\|^2}{(\mathbf{r}^{\mathrm{T}}\mathbf{v})^2}\|\mathbf{v}\|^2. \tag{34}$$

Thus, by dividing both the sides by $C/Q(t)$ the following scalar equation is obtained:

$$a_0\beta^2 - 2\beta + 1 - \frac{Q(t)}{Q(t+\Delta t)} = 0, \tag{35}$$

where

$$a_0 := \frac{\|\mathbf{r}\|^2\|\mathbf{v}\|^2}{(\mathbf{r}^{\mathrm{T}}\mathbf{v})^2}. \tag{36}$$

As a result $h(\mathbf{x},t) = 0$, $t \in \{0,1,2,\ldots\}$ remains to be an invariant-manifold in the space-time of $(\mathbf{x},t)$ for the discrete time dynamical system $h(\mathbf{x}(t),t) = 0$, which will be explored further in the next few sections.

3.2 A trial discrete dynamics

Now we specify the discrete time dynamics $h(\mathbf{x}(t),t) = 0$, $t \in \{0,1,2,\ldots\}$, through specifying the discrete time dynamics of $Q(t)$, $t \in \{0,1,2,\ldots\}$. *Note that the discrete time dynamics is an iterative dynamics, which in turn amounts to an iterative algorithm.*

We first try the forward Euler scheme:

$$Q(t+\Delta t) = Q(t) + \dot{Q}(t)\Delta t. \tag{37}$$

Then from Eq. (27) we have

$$\beta = q(t)\Delta t = \frac{1}{2}[R(t) - 1], \tag{38}$$

where the ratio $R(t)$ is defined by

$$R(t) = \frac{Q(t+\Delta t)}{Q(t)}. \tag{39}$$

As a requirement of $\dot{Q}(t) > 0$, we need $R(t) > 1$.

Thus, through some manipulations, Eq. (35) becomes

$$a_0R^3(t) - (2a_0+4)R^2(t) + (a_0+8)R(t) - 4 = 0, \tag{40}$$

which can be further written as

$$[R(t)-1]^2[a_0R(t) - 4] = 0. \tag{41}$$

Because $R = 1$ is a double root and does not satisfy $R > 1$, we take

$$R(t) = \frac{4}{a_0} = \frac{4(\mathbf{r}^{\mathrm{T}}\mathbf{v})^2}{\|\mathbf{r}\|^2\|\mathbf{v}\|^2}. \tag{42}$$

By using Eq. (38), Eq. (30) can now be written as

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) - \frac{1}{2}[R(t)-1]\frac{\|\mathbf{r}\|^2}{\mathbf{r}^{\mathrm{T}}\mathbf{v}}\mathbf{u}. \tag{43}$$

Notice, however, that this algorithm has an unfortunate fate in that when the iterated a_0 starts to approach to 4 before it grows to a large value, the algorithm stagnates at a point which is not necessarily a solution. We should avoid to follow this kind of dynamics by developing a better dynamics as below. This indicates that a vector-driven algorithm will enter this trap to lose its dynamical force, if one insists the iterative orbit being located on the manifold (28) with specifying $Q(t)$.

3.3 A better discrete dynamics

Let

$$s = \frac{Q(t)}{Q(t+\Delta t)} = \frac{\|\mathbf{r}(\mathbf{x}(t+\Delta t))\|^2}{\|\mathbf{r}(\mathbf{x}(t))\|^2}, \tag{44}$$

which is an important quantity in assessing the convergence property of the numerical algorithm for solving the system (1) of LAEs.

From Eqs. (35) and (44) it follows that

$$a_0\beta^2 - 2\beta + 1 - s = 0, \tag{45}$$

where

$$a_0 := \frac{\|\mathbf{r}\|^2\|\mathbf{v}\|^2}{(\mathbf{r}^{\mathrm{T}}\mathbf{v})^2} \geq 1, \tag{46}$$

by using the Cauchy-Schwarz inequality:

$$\mathbf{r}^{\mathrm{T}}\mathbf{v} \leq \|\mathbf{r}\|\|\mathbf{v}\|.$$

From Eq. (45), we can take the solution of β to be

$$\beta = \frac{1-\sqrt{1-(1-s)a_0}}{a_0}, \ \text{ if } 1-(1-s)a_0 \geq 0. \tag{47}$$

Let

$$1-(1-s)a_0=\gamma^2\geq 0, \tag{48}$$

$$s=1-\frac{1-\gamma^2}{a_0}. \tag{49}$$

Thus, from Eq. (47) it follows that

$$\beta=\frac{1-\gamma}{a_0}, \tag{50}$$

and from Eqs. (30) and (36) we can obtain the following algorithm:

$$\mathbf{x}(t+\Delta t)=\mathbf{x}(t)-(1-\gamma)\frac{\mathbf{r}^{\mathrm{T}}\mathbf{v}}{\|\mathbf{v}\|^2}\mathbf{u}, \tag{51}$$

where γ is a parameter chosen to be

$$0\leq\gamma<1. \tag{52}$$

In this discrete dynamics, we do not specify $Q(t)$ a priori; instead, $Q(t)$ is an induced quantity, automatically derived from the discrete dynamics by Eqs. (44) and (49), beginning with $Q(0)=1$:

$$Q(t+\Delta t)=\frac{Q(t)}{s}=\frac{a_0Q(t)}{a_0-1+\gamma^2}. \tag{53}$$

Under the above conditions (46) and (52), from Eqs. (44) and (49) we can prove that the new algorithm satisfies

$$\frac{\|\mathbf{r}(t+\Delta t)\|}{\|\mathbf{r}(t)\|}=\sqrt{s}<1, \tag{54}$$

which means that the residual error is absolutely decreased. Alternatively, the convergence rate of present iterative algorithm reads as

$$\text{Convergence Rate}:=\frac{\|\mathbf{r}(t)\|}{\|\mathbf{r}(t+\Delta t)\|}=\frac{1}{\sqrt{s}}>1. \tag{55}$$

The property in Eq. (55) is very important, since it guarantees the new algorithm to be absolutely convergent to the true solution. *Smaller* (s) *implies Faster convergence*.

3.4 Optimization of α in the presently used descent vector $\mathbf{u} = (\mathbf{B}^T + \alpha \mathbf{I}_n)\mathbf{r}$, $\mathbf{r} = \mathbf{B}\mathbf{x} - \mathbf{b}$

The algorithm (51) does not specify how to choose the parameter α. We can determine a suitable α such that s defined in Eq. (49) is minimized with respect to α, because a smaller s will lead to a faster convergence as shown in Eq. (55).

Thus by inserting Eq. (36) for a_0 into Eq. (49) we can write s to be

$$s = 1 - \frac{(1-\gamma^2)(\mathbf{r}\cdot\mathbf{v})^2}{\|\mathbf{r}\|^2\|\mathbf{v}\|^2}, \tag{56}$$

where $\mathbf{v}$ as defined by Eq. (26) includes a parameter α. Let $\partial s/\partial\alpha = 0$, and through some algebraic operations we can solve α by

$$\alpha = \frac{(\mathbf{v}_1\cdot\mathbf{r})(\mathbf{v}_1\cdot\mathbf{v}_2) - (\mathbf{v}_2\cdot\mathbf{r})\|\mathbf{v}_1\|^2}{(\mathbf{v}_2\cdot\mathbf{r})(\mathbf{v}_1\cdot\mathbf{v}_2) - (\mathbf{v}_1\cdot\mathbf{r})\|\mathbf{v}_2\|^2}. \tag{57}$$

Remark: For the usual three-dimensional vectors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c} \in \mathbb{R}^3$, the following formula is famous:

$$\mathbf{a}\times(\mathbf{b}\times\mathbf{c}) = (\mathbf{a}\cdot\mathbf{c})\mathbf{b} - (\mathbf{a}\cdot\mathbf{b})\mathbf{c}. \tag{58}$$

Liu (2000) has developed a Jordan algebra by extending the above formula to vectors in n-dimension:

$$[\mathbf{a},\mathbf{b},\mathbf{c}] = (\mathbf{a}\cdot\mathbf{b})\mathbf{c} - (\mathbf{c}\cdot\mathbf{b})\mathbf{a}, \quad \mathbf{a},\mathbf{b},\mathbf{c}\in\mathbb{R}^n. \tag{59}$$

Thus α in Eq. (57) can be expressed as

$$\alpha = \frac{[\mathbf{v}_1,\mathbf{r},\mathbf{v}_2]\cdot\mathbf{v}_1}{[\mathbf{v}_2,\mathbf{r},\mathbf{v}_1]\cdot\mathbf{v}_2}. \tag{60}$$

The above parameter α is the optimal one, because it brings us a new strategy to select the best descent vector $\mathbf{u}$ to search the solution of linear equations system (1). Furthermore, we have an explicit form of a formula to implement it into the numerical solution program, and thus it is very simple and inexpensive to calculate α.

3.5 An optimal iterative algorithm to solve an ill-conditioned system $\mathbf{B}\mathbf{x} = \mathbf{b}$, using an optimal descent vector $\mathbf{u}$

Since the fictitious time variable is now discrete, $t \in \{0,1,2,\ldots\}$, we let $\mathbf{x}_k$ denote the numerical value of $\mathbf{x}$ at the k-th step. Thus, we arrive at a purely iterative

algorithm by Eq. (51):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (1-\gamma)\frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{v}_k}{\|\mathbf{v}_k\|^2}\mathbf{u}_k. \tag{61}$$

Then, the following *optimal iterative algorithm* (OIA), involving an *optimal descent vector* (ODV), for solving the ill-conditioned system $\mathbf{Bx} = \mathbf{b}$ is derived:

(i) Select $0 \le \gamma < 1$, and give an initial $\mathbf{x}_0$.

(ii) For $k = 0, 1, 2 \ldots$, we repeat the following computations:

$$\begin{aligned}
&\mathbf{r}_k = \mathbf{B}\mathbf{x}_k - \mathbf{b},\\
&\mathbf{v}_1^k = \mathbf{A}\mathbf{r}_k,\\
&\mathbf{v}_2^k = \mathbf{B}\mathbf{r}_k,\\
&\alpha_k = \frac{[\mathbf{v}_1^k, \mathbf{r}_k, \mathbf{v}_2^k]\cdot\mathbf{v}_1^k}{[\mathbf{v}_2^k, \mathbf{r}_k, \mathbf{v}_1^k]\cdot\mathbf{v}_2^k}: \quad \text{Optimum } \alpha \text{ in the Descent Vector},\\
&\mathbf{u}_k = \alpha_k\mathbf{r}_k + \mathbf{B}^{\mathrm{T}}\mathbf{r}_k = (\mathbf{B}^{\mathrm{T}} + \alpha_k\mathbf{I}_n)\mathbf{r}_k: \quad \text{Optimum Descent Vector},\\
&\mathbf{v}_k = \mathbf{v}_1^k + \alpha_k\mathbf{v}_2^k,\\
&\mathbf{x}_{k+1} = \mathbf{x}_k - (1-\gamma)\frac{\mathbf{r}_k\cdot\mathbf{v}_k}{\|\mathbf{v}_k\|^2}\mathbf{u}_k.
\end{aligned} \tag{62}$$

If $\mathbf{x}_{k+1}$ converges according to a given stopping criterion $\|\mathbf{r}_{k+1}\| < \varepsilon$, then stop; otherwise, go to step (ii).

In summary, we have derived a simple and novel algorithm for solving the ill-conditioned system (1) of LAEs. While the parameter γ is chosen by the user for the problem-dependence, the optimal parameter α [in the descent vector $\mathbf{u} = (\mathbf{B}^{\mathrm{T}} + \alpha\mathbf{I}_n)\mathbf{r}$, $(\mathbf{r} = \mathbf{Bx} - \mathbf{b})$] is exactly given by Eq. (60). Indeed these two parameters play the roles of *bifurcation parameter* and *optimization parameter*, respectively. See Liu and Atluri (2011) for the discussions of the influences of these two parameters for solving nonlinear algebraic equations.

3.6 The conjugate gradient method

For the later purposes of comparison, the conjugate gradient method (CGM) for solving Eq. (5) is summarized as follows.

(i) Give an initial $\mathbf{x}_0$ and then compute $\mathbf{R}_0 = \mathbf{C}\mathbf{x}_0 - \mathbf{b}_1$ and set $\mathbf{p}_1 = \mathbf{R}_0$.

(ii) For $k = 1, 2, \ldots$, we repeat the following computations:

$$\begin{aligned}
&\eta_k = \frac{\|\mathbf{R}_{k-1}\|^2}{\mathbf{p}_k^{\mathrm{T}}\mathbf{C}\mathbf{p}_k},\\
&\mathbf{x}_k = \mathbf{x}_{k-1} - \eta_k \mathbf{p}_k,\\
&\mathbf{R}_k = \mathbf{B}^{\mathrm{T}}\mathbf{r}_k = \mathbf{C}\mathbf{x}_k - \mathbf{b}_1: \quad \text{Descent Vector},\\
&\alpha_k = \frac{\|\mathbf{R}_k\|^2}{\|\mathbf{R}_{k-1}\|^2},\\
&\mathbf{p}_{k+1} = \alpha_k \mathbf{p}_k + \mathbf{R}_k.
\end{aligned} \tag{63}$$

If $\mathbf{x}_k$ converges according to a given stopping criterion $\|\mathbf{R}_k\| < \varepsilon$, then stop; otherwise, go to step (ii). Here, $\mathbf{R}_k = \mathbf{B}^{\mathrm{T}}\mathbf{r}_k$, and we call $\mathbf{r}_k$ the residual vector, while $\mathbf{R}_k$ is called the descent vector.

4 A matrix type linear equations

Sometimes we may encounter the LAEs which are discretized from the linear PDE with a matrix type. In this situation it is not so straightforward to write its counterpart as being a vector-form linear equations. Let us consider

$$\Delta u(x,y) = F(x,y,u,u_x,u_y), \quad (x,y) \in \Omega, \tag{64}$$

$$u(x,y) = H(x,y), \quad (x,y) \in \Gamma, \tag{65}$$

where Δ is the Laplacian operator, Γ is the boundary of a problem domain $\Omega := [a_0, a_1] \times [b_0, b_1]$, and F and H are given functions. F is a linear function of u, u_x, u_y. By a standard five-point finite difference applied to Eq. (64), one has a system of linear equations of matrix type:

$$\begin{aligned}
F_{i,j} = {} & \frac{1}{(\Delta x)^2}[u_{i+1,j} - 2u_{i,j} + u_{i-1,j}] + \frac{1}{(\Delta y)^2}[u_{i,j+1} - 2u_{i,j} + u_{i,j-1}]\\
& - F\left(x_i, y_j, u_{i,j}, \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}, \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\right) = 0, \quad 1 \le i \le n_1, \quad 1 \le j \le n_2.
\end{aligned} \tag{66}$$

Here, we divide the rectangle Ω by a uniform grid with $\Delta x = (a_1 - a_0)/(n_1 + 1)$ and $\Delta y = (b_1 - b_0)/(n_2 - 1)$ being the uniform spatial grid lengths in the x- and y-direction, and $u_{i,j} := u(x_i, y_j)$ be a numerical value of u at the grid point $(x_i, y_j) \in \Omega$. Let $K = n_2(i-1) + j$ and with i running from 1 to n_1 and j running from 1 to n_2 we can, respectively, set up the vectorial variables x_K and the vectorial linear equations

$r_K = 0$ by

$$\begin{aligned}
&\text{Do } i = 1, n_1 \\
&\text{Do } j = 1, n_2 \\
&K = n_2(i-1) + j \\
&x_K = u_{i,j} \\
&r_K = F_{i,j}.
\end{aligned} \tag{67}$$

At the same time the components of the coefficient matrix **B** are constructed by

$$\begin{aligned}
&\text{Do } i = 1, n_1 \\
&\text{Do } j = 1, n_2 \\
&K = n_2(i-1) + j \\
&L_1 = n_2(i-2) + j, \;\; i > 1 \\
&L_2 = n_2(i-1) + j - 1, \;\; j > 1 \\
&L_3 = n_2(i-1) + j \\
&L_4 = n_2(i-1) + j + 1, \;\; j < n_2 \\
&L_5 = n_2 i + j, \;\; i < n_1 \\
&B_{K,L_1} = \frac{1}{(\Delta x)^2} + \frac{1}{2\Delta x} F_{u_x}\left(x_i, y_j, u_{i,j}, \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}, \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\right) \\
&B_{K,L_2} = \frac{1}{(\Delta y)^2} + \frac{1}{2\Delta y} F_{u_y}\left(x_i, y_j, u_{i,j}, \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}, \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\right) \\
&B_{K,L_3} = -\frac{2}{(\Delta x)^2} - \frac{2}{(\Delta y)^2} - F_u\left(x_i, y_j, u_{i,j}, \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}, \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\right) \\
&B_{K,L_4} = \frac{1}{(\Delta y)^2} - \frac{1}{2\Delta y} F_{u_y}\left(x_i, y_j, u_{i,j}, \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}, \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\right) \\
&B_{K,L_5} = \frac{1}{(\Delta x)^2} - \frac{1}{2\Delta x} F_{u_x}\left(x_i, y_j, u_{i,j}, \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}, \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\right).
\end{aligned} \tag{68}$$

In above, F_u, F_{u_x} and F_{u_y} denote, respectively, the partial differentials of the function $F(x, y, u, u_x, u_y)$ with respect to u, u_x and u_y, and the indices L_i are constrained by $1 \le L_i \le n$, where $n = n_1 \times n_2$. When the above quantities are available, we can apply the vector-form OIA/ODV to solve Eq. (66) with n equations for the linear PDE in Eqs. (64) and (65). For the purpose of comparison, we can also apply CGM to solve the above linear equations.

5 Numerical examples

5.1 Example 1

In this example we examine a two-dimensional but highly ill-conditioned linear system to display the superiority of Optimal Iterative Algorithm (OIA) with an Optimal Descent Vector (ODV):

$$\begin{bmatrix} 2 & 6 \\ 2 & 6.0001 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 8 \\ 8.0001 \end{bmatrix}. \tag{69}$$

The condition number of this system is $\text{cond}(\mathbf{C}) = 1.596 \times 10^{11}$, where $\mathbf{C} = \mathbf{B}^{\mathrm{T}}\mathbf{B}$ and $\mathbf{B}$ denotes the coefficient matrix . The exact solution is $(x,y) = (1,1)$.

Under a convergence criterion $\varepsilon = 10^{-13}$, and starting from an initial point $(x_0, y_0) = (10,10)$ we apply CGM and OIA/ODV with $\gamma = 0$ to solve the above equation. When CGM spent four iterations, OIA/ODV only spent two iterations, of which the iterative paths are compared in Fig. 1. When OIA/ODV goes on a straight line to the solution point, CGM traces a broken line to the solution point. The maximum error obtained by CGM is 1.94×10^{-5}, which is much larger than 1.61×10^{-9} obtained by OIA/ODV.

5.2 Example 2

In order to test the performance of the presently proposed algorithm, we apply OIA/ODV with the technique in Section 4 to the linear equations discretized from the following Laplace equation:

$$\begin{aligned} &u_{xx} + u_{yy} = 0, \;\; 0 < x < 1, \;\; 0 < y < 1, \\ &u(x,0) = \sin x, \;\; u(x,1) = \sin x \cosh 1, \\ &u(0,y) = 0, \;\; u(1,y) = \sin 1 \cosh y. \end{aligned} \tag{70}$$

The exact solution is $u(x,y) = \sin x \cosh y$. Under the discretization with $\Delta x = \Delta y = 1/16$, and with $\gamma = 0.4$ used in OIA/ODV, the numerical process spends 55 iterations by satisfying the convergence criterion with $\varepsilon = 10^{-5}$. The relative residual $\|\mathbf{r}_k\|/\|\mathbf{r}_0\|$ is shown in Fig. 2(a), and the numerical error is shown in Fig. 3(a), of which the maximum error is smaller than 1.31×10^{-5}. While very accurate numerical results are obtained, we find that OIA/ODV converges very fast with only 55 iterations to solve the total of 225 linear equations. The clue for this fast convergence can be seen from Fig. 2(b), where we have large convergence rate $1/\sqrt{s}$ and small a_0. Conversely, CGM spends 126 iterations to satisfy the above convergence criterion as shown in Fig. 2(a), and the maximum error as shown in Fig. 3(b) is

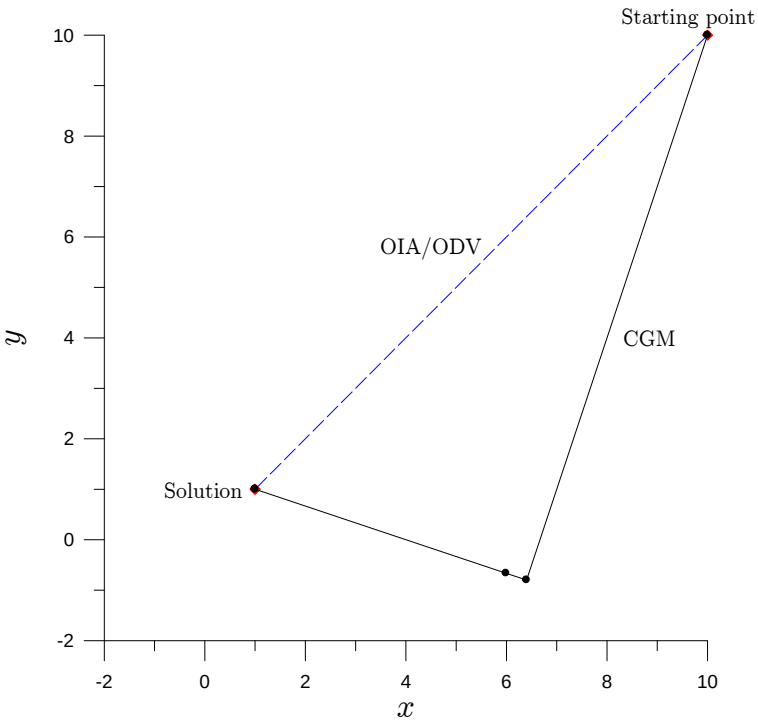

Figure 1: For example 1 comparing the iterative paths for CGM and OIA/ODV.

2.74×10^{-5}, which is larger than that of OIA/ODV. Theoretically, the CGM algorithm can find the solution with n iterations for a well-posed linear problem. However, when OIA/ODV has a monotonically decreasing convergent property, CGM does not converge monotonically. Thus, OIA/ODV is convergent faster and more accurately than CGM.

5.3 Example 3

In order to compare OIA/ODV with CGM, we solve the following Poisson equation:

$$\begin{aligned} &u_{xx}+u_{yy}=p(x,y),\ \ 0<x<1,\ \ 0<y<1,\\ &u(x,y)=x^2-y^2+e^{x+y},\ \ p(x,y)=2e^{x+y}. \end{aligned} \tag{71}$$

The boundary conditions can be derived from the exact solution. Under the discretization with $\Delta x=\Delta y=1/16$, and with $\gamma=0.04$, OIA/ODV spends 46 iterations

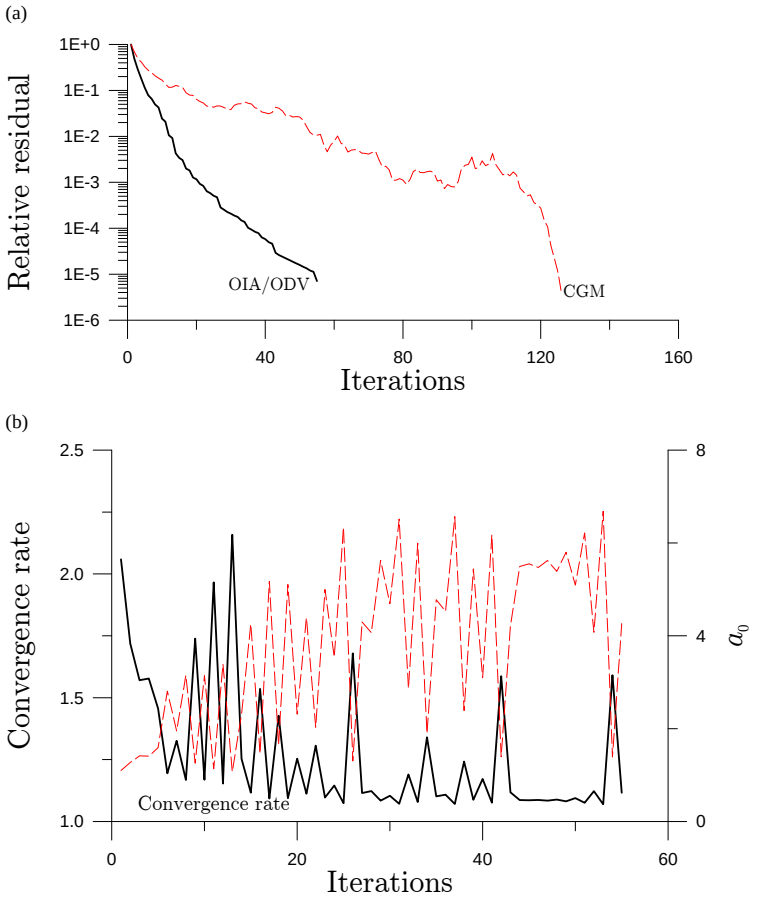

Figure 2: For example 2: (a) comparing the relative residuals of OIA/ODV and CGM, and (b) showing the convergence rate and a_0 for OIA/ODV.

by satisfying the convergence criterion with $\varepsilon = 10^{-5}$, and the maximum numerical error is 8.7×10^{-5}. The CGM spends 127 iterations under the same convergence criterion, and the maximum error is 1.42×10^{-4}. From the relative residuals as shown in Fig. 4(a), we can see that OIA/ODV converges faster than CGM. At the same time the accuracy is raised one order by OIA/ODV.

Next, we use this example to investigate the structural stability of OIA/ODV and CGM. It is known that CGM can lose efficiency rapidly if the matrix-vector multiplications are not accurate [Golub and Ye (2000); Haber and Ascher (2001)]. To simulate this incorrect matrix-vector multiplication we suppose an incorrect B_{ij}, which is constructed from Eq. (68) without the constraint $j < n_2$ for the index L_4.

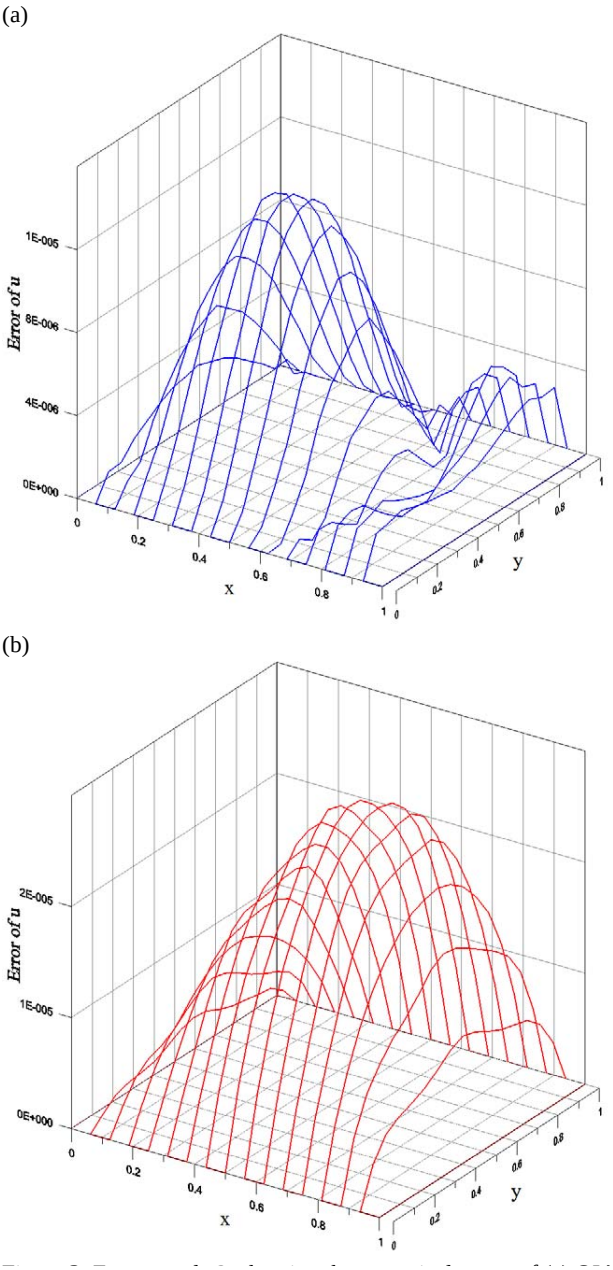

Figure 3: For example 2: showing the numerical errors of (a) OIA/ODV, and (b) CGM.

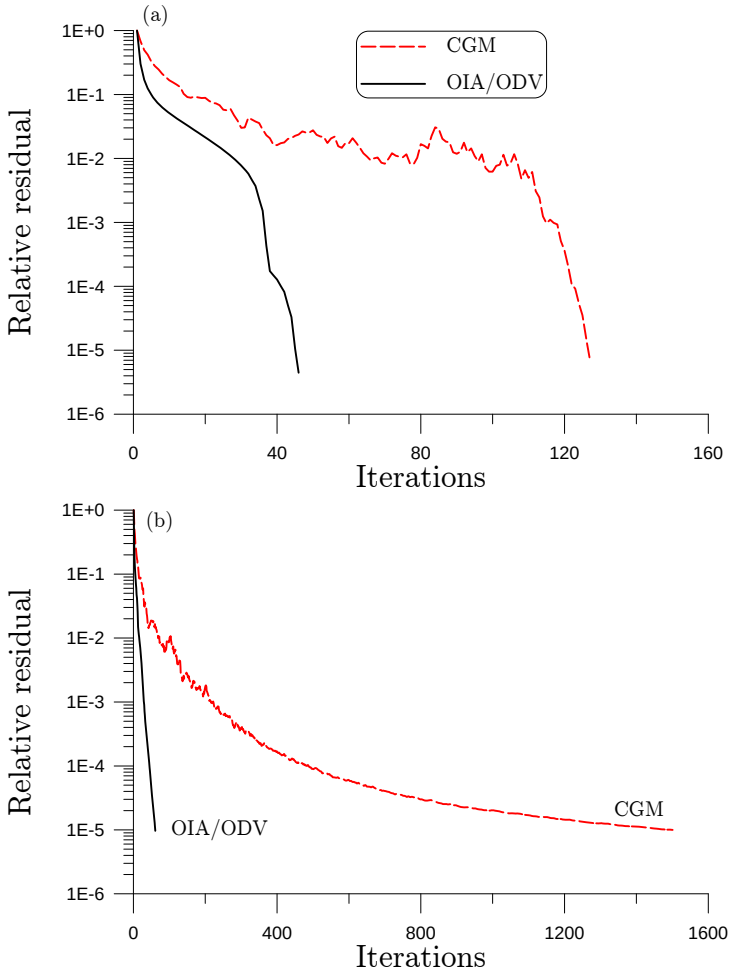

Figure 4: For example 3 of a linear Poisson PDE: comparing (a) the relative residuals for OIA/ODV and CGM, and (b) under a perturbation of the coefficient matrix.

However, we keep the linear equations $r_K = 0$ unperturbed. Under the above same conditions we find that OIA/ODV runs 61 iterations and is still rather accurate with the maximum error being 3.1×10^{-4}. But, CGM spends 1501 iterations as shown in Fig. 4(b), and the maximum error is increased to 1.32×10^{-2}. Obviously, the OIA/ODV is more structurally stable than CGM.

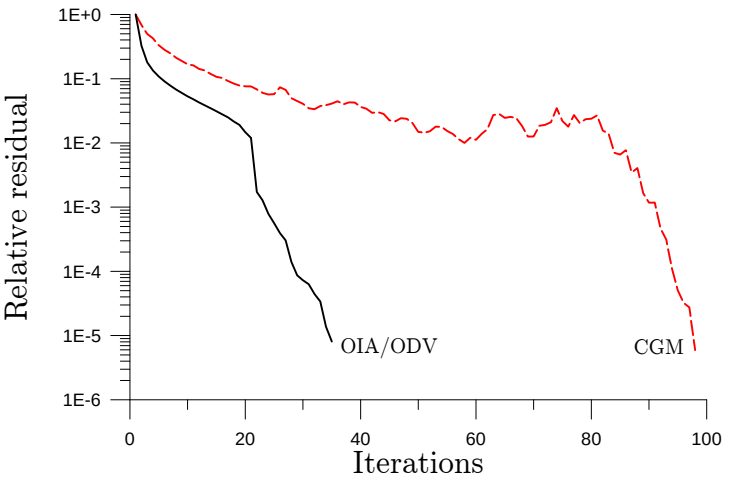

Figure 5: For example 4 of a Helmholtz equation: comparing the relative residuals for OIA/ODV and CGM.

5.4 Example 4

We consider the following Helmholtz equation:

$$\begin{aligned} &u_{xx}+u_{yy}+2u=0, \quad 0<x<1, \quad 0<y<1, \\ &u(x,y)=\sin(x+y). \end{aligned} \tag{72}$$

Under the discretization $\Delta x=\Delta y=1/14$, and with $\gamma=0.1$, the present OIA/ODV spends 35 iterations and the maximum error is 2.24×10^{-5}. The CGM spends 98 iterations and the maximum error is 5.7×10^{-5}. From the relative residuals as compared in Fig. 5, we can see that OIA/ODV converges faster and is more accurate than CGM.

5.5 Example 5

We solve the modified Helmholtz equation:

$$\begin{aligned} &u_{xx}+u_{yy}-3u+\frac{3y}{x^2+y^2}=0, \quad 1<x<2, \quad 1<y<2, \\ &u(x,y)=\sin(x)\cosh(2y)+\frac{y}{x^2+y^2}. \end{aligned} \tag{73}$$

Under $\Delta x=\Delta y=1/14$, and with $\gamma=0.1$, when the present OIA/ODV runs 34 iterations with the maximum numerical error being 4.1×10^{-3}, the CGM spends

93 iterations, and the maximum error is 5.2×10^{-3}. In Fig. 6, the relative residuals of OIA/ODV and CGM are compared, of which OIA/ODV converges faster than CGM.

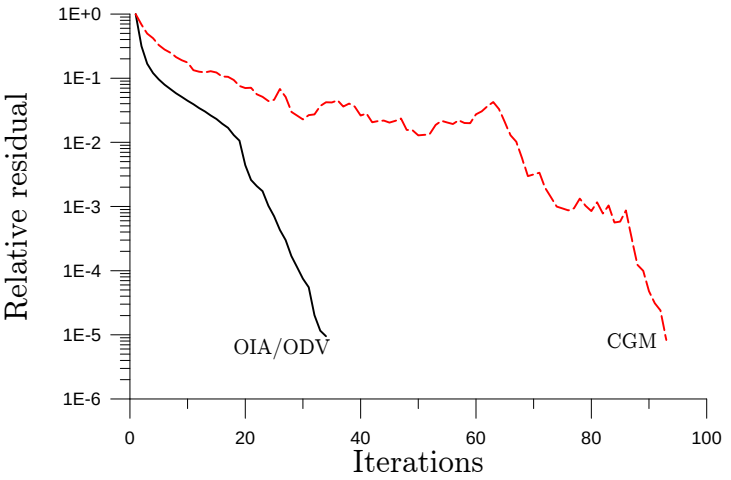

Figure 6: For example 5 of a modified Helmholtz equation: comparing the relative residuals for OIA/ODV and CGM.

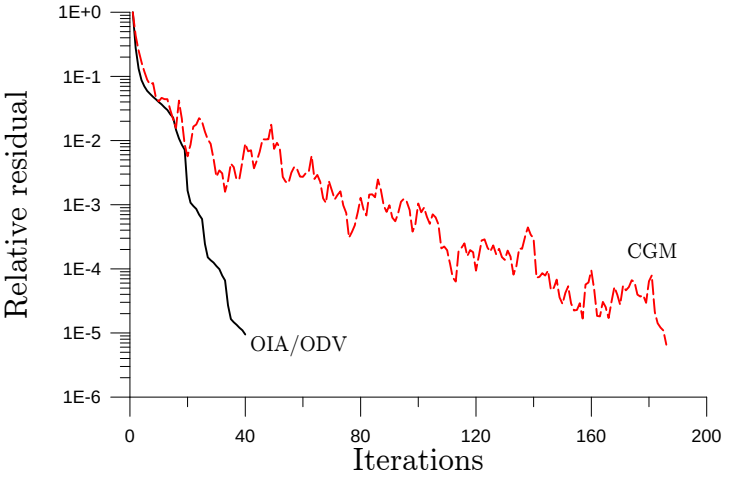

Figure 7: For example 6 of a heat conduction equation: comparing the relative residuals for OIA/ODV and CGM.

5.6 Example 6

We consider a linear heat conduction equation:

$$u_t = \alpha(x)u_{xx} + \alpha'(x)u_x + h(x,t), \quad 0 < x < 1, \quad 0 < t \le 1,$$
$$\alpha(x) = (x-3)^2, \quad h(x,t) = -7(x-3)^2 e^{-t}, \tag{74}$$

with a closed-form solution $u(x,t) = (x-3)^2 e^{-t}$. The boundary conditions and initial condition are computed from the exact solution.

Under the discretization with $\Delta x = 1/15$ and $\Delta t = 1/20$, and with $\gamma = 0.1$, the present OIA/ODV runs 40 iterations with the maximum error being 2.9×10^{-3}. The CGM spends 185 iterations, and the maximum error is 1.9×10^{-2}. In Fig. 7, the relative residuals of OIA/ODV and CGM are compared, revealing that OIA/ODV converges faster than CGM, and the accuracy is raised one order by OIA/ODV.

6 Conclusions

It is well-known that the Conjugate Gradient Method (CGM) for solving well-posed system of linear equations is very effective, and up to now, *no other variants of gradient descent algorithms can perform better than CGM*. In the present paper, we have derived a purely iterative algorithm involving a preset parameter γ $(0 \le \gamma < 1)$, and an optimal parameter α_k in the optimal descent vector $\mathbf{u}_k$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (1-\gamma)\frac{\mathbf{r}_k \cdot \mathbf{v}_k}{\|\mathbf{v}_k\|^2}\mathbf{u}_k, \tag{75}$$

where

$$\mathbf{u}_k = \alpha_k \mathbf{r}_k + \mathbf{B}^{\mathrm{T}}\mathbf{r}_k = (\alpha_k \mathbf{I}_n + \mathbf{B}^{\mathrm{T}})\mathbf{r}_k \tag{76}$$

is a stepwise optimal descent vector, with α_k being optimized by Eq. (60) and $\mathbf{v}_k = \mathbf{B}\mathbf{u}_k$. This algorithm is an *Optimal Iterative Algorithm (OIA) with an Optimal Descent Vector (ODV)*, which *has a better computational efficiency and accuracy than the CGM algorithm* in solving either an ill-conditioned or, of course, a well-conditioned system of LAEs. An example with a perturbed coefficient matrix was also used to verify that the present OIA/ODV is more structurally stable than the CGM.

Acknowledgement: The research at UCI is supported by a collaborative research agreement between the Army Research Labs and UCI. Taiwan's National Science Council project NSC-100-2221-E-002-165-MY3 granted to the first author is also highly appreciated.

References

Akaike, H. (1959): On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Ann. Inst. Stat. Math. Tokyo*, vol. 11, pp. 1-16.

Ascher, U.; van den Doel, K.; Hunag, H.; Svaiter, B. (2009): Gradient descent and fast artificial time integration. *M2AN*, vol. 43, pp. 689-708.

Barzilai, J.; Borwein, J. M. (1988): Two point step size gradient methods. *IMA J. Numer. Anal.*, vol. 8, pp. 141-148.

Dai, Y. H.; Yuan, Y. (2003): Alternate minimization gradient method. *IMA J. Numer. Anal.*, vol. 23, pp. 377-393.

Dai, Y. H.; Hager, W.; Schittkowsky, K.; Zhang, H. (2006): A cyclic Barzilai-Borwein method for unconstrained optimization. *IMA J. Numer. Anal.*, vol. 26, pp. 604-627.

Forsythe, G. E. (1968): On the asymptotic directions of the s-dimensional optimum gradient method. *Numer. Math.*, vol. 11, pp. 57-76.

Friedlander, A.; Martinez, J.; Molina, B.; Raydan, M. (1999): Gradient method with retard and generalizations. *SIAM J. Num. Anal.*, vol. 36, pp. 275-289.

Golub, G.; Ye, Q. (2000): Inexact preconditioned conjugate gradient method with inner-outer iteration. *SIAM J. Scient. Comput.*, vol. 21, pp. 1305-1320.

Haber, E.; Ascher, U. (2001): Preconditioned all-at-one methods for large, sparse parameter estimation problems. *Inverse Prob.*, vol. 17, pp. 1847-1864.

Hoang, N. S.; Ramm, A. G. (2008): Solving ill-conditioned linear algebraic systems by the dynamical systems method (DSM). *J. Inverse Prob. Sci. Eng.*, vol. 16, pp. 617-630.

Hoang, N. S.; Ramm, A. G. (2010): Dynamical systems gradient method for solving ill-conditioned linear algebraic systems. *Acta Appl. Math.*, vol. 111, pp. 189-204.

Liu, C.-S. (2000): A Jordan algebra and dynamic system with associator as vector field. *Int. J. Non-Linear Mech.*, vol. 35, pp. 421-429.

Liu, C.-S. (2011): A revision of relaxed steepest descent method from the dynamics on an invariant manifold. *CMES: Computer Modeling in Engineering & Sciences*, in press.

Liu, C.-S.; Chang, C. W. (2009): Novel methods for solving severely ill-posed linear equations system. *J. Marine Sci. Tech.*, vol. 9, pp. 216-227.

Liu, C.-S.; Atluri, S. N. (2011): An iterative algorithm for solving a system of nonlinear algebraic equations, $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, using the system of ODEs with an opti-

mum α in $\dot{\mathbf{x}} = \lambda[\alpha\mathbf{F} + (1-\alpha)\mathbf{B}^{\mathrm{T}}\mathbf{F}]$; $B_{ij} = \partial F_i/\partial x_j$. *CMES: Computer Modeling in Engineering & Sciences*, vol. 73, pp. 395-431.

Nocedal, J.; Sartenar, A.; Zhu, C. (2002): On the behavior of the gradient norm in the steepest descent method. *Comp. Optim. Appl.*, vol. 22, pp. 5-35.

Ramm, A. G. (2007): Dynamical System Methods for Solving Operator Equations. Elsevier, Amsterdam, Netherlands.

Ramm, A. G. (2009): Dynamical systems method for solving linear ill-posed problems. *Ann. Pol. Math.*, vol. 95, pp. 253-272.

Raydan, M.; Svaiter, B. F. (2002): Relaxed steepest descent and Cauchy-Barzilai-Borwein method. *Comp. Optim. Appl.*, vol. 21, pp. 155-167.

Sweilam, N. H.; Nagy, A. M.; Alnasr, M. H. (2009): An efficient dynamical systems method for solving singularly perturbed integral equations with noise. *Comp. Math. Appl.*, vol. 58, pp. 1418-1424.