The Block Conjugate Gradient for Multiple Right Hand Sides in a Direct Current Resistivity Inversion

Rowan Cockett
Department of Earth and Ocean Science
University of British Columbia
rcockett@eos.ubc.ca

Abstract

In geophysical applications direct current (DC) resistivity surveys are collected by injecting current and recording voltage potentials over a field site of interest. These data sets can provide valuable non-invasive images of the sub-surface. The optimization problem that arrises in producing these images requires multiple solves of a linear system with numerous right hand sides (RHSs). Due to the three dimensional nature of the DC resistivity problem, the system to solve is typically too large for direct methods, and iterative methods must be used; in this case conjugate gradient (CG) is ideal. However, standard iterative solvers have no way to share information between the different solves. Information sharing techniques between the RHSs is necessary for speedy convergence and are why direct methods are so attractive. Standard iterative solvers can be modified to block methods that share information by solving multiple RHSs at once. The extension from CG to the corresponding block method, block conjugate gradient (BLCG), is clearly laid out and practical implementation issues are discussed. BLCG can have have numerical instabilities if linear dependencies exist between the different RHSs. This is dealt with through initial deflation and orthogonalization of the RHSs and an unequal convergence scheme. Numerical experiments were ran and suggest that using BLCG can have significant gains over using standard CG. These gains are most dramatic if initial deflation of the RHSs is possible and enforced.

1 Introduction

A set of problems in geophysical applications involves the conversion of observational data into a predictive model that feasibly creates those observations. These problems can be divided into two distinct elements: the forward problem and the inverse problem. The forward problem simulates an observational data set given model parameters; this is described mathematically by a partial differential equation. The inverse problem solves for feasible model parameters given a sparse set of observational data. The inverse problem can be formed as an optimization problem and solved iteratively. One specific context for this problem is in terms of the direct-current (DC) resistivity problem. In this problem source electrodes inject current, and receiver electrodes measure the responding voltages. Many of these measurements are taken at a field site of interest and the information is combined in a geophysical inversion to create valuable images of the subsurface. The DC resistivity problem can be described mathematically using a discretized version of Poisson's equation with appropriate boundary conditions [1]. A set of linear systems is generated, $\mathbf{A}$, which can be very large due to the three dimensional nature of this problem. This system of equations must be solved multiple times in an iterative optimization procedure. Additionally, the system of equations has multiple right hand sides (RHSs) that relate to the source configurations of the survey. Generally, the more RHSs gathered the better the results of the inversion; when direct methods are used to solve the system of equations the factorization can easily be applied to these new RHSs with a forward and back solve. However, both the large and sparse nature of $\mathbf{A}$ commonly leads to the use of iterative solvers. The information sharing techniques that are attractive in direct solvers are no longer available and standard iterative solvers require each RHS to be solved individually.

To share information between the RHSs the standard iterative methods must be extended. There are two general ways in which this can be done: (a) through a seed system where search spaces are recycled in the next solve, and (b) through a block system solve. Since all RHSs in this application are known at the beginning of the solve it is typical to use block methods [10]. In this paper these block methods are investigated in detail in the context of the conjugate gradient algorithm.

2 Motivation - The DC Resistivity Problem

The motivation of this research is the direct current (DC) resistivity problem in geophysical applications. In this experiment, electrodes are laid out over a field site of interest; source pairs (input at a known current) are sequentially turned on and off and receiver pairs observe the responding voltage potentials. After many voltage measurements have been made, they can be combined in a geophysical inversion to gain information about the conductivity of the subsurface. The conductivity of the ground can be linked to physical parameters of interest. For example, pore-water conductivity is greatly influenced by electrolytic contaminants, these contaminants can thus be imaged through their electrical properties in a DC resistivity survey and appropriate actions can be taken [4]. The DC resistivity problem is governed by Poisson's equation with appropriate boundary conditions applied

$$\nabla \cdot (-\sigma \nabla \phi) = \mathbf{I}(\delta(\mathbf{r} - \mathbf{r_{s^+}}) - \delta(\mathbf{r} - \mathbf{r_{s^-}})) \tag{1}$$

where σ is the conductivity structure of a medium; ϕ is the electrical potential field induced by a dipole; and $\mathbf{I}$ is the electrical current from a dipole. The dipole is represented by two dirac delta functions centered on the positive and negative source locations ($\mathbf{r_{s^+}}$ and $\mathbf{r_{s^-}}$ respectively) [1]; where $\mathbf{r}$ is a position vector. Neumann boundary conditions are often applied at infinity or sufficiently far from the area of interest in a discretized model [5]. These boundary conditions are representative of a field experiment where the site is unbounded.

2.1 Forward Problem

The generation of data given a conductivity model is called the forward problem; this can be completed by discretizing the DC resistivity equation. In this paper a cell-centered, finite-volume mesh is implemented. In matrix notation Equation 1 can be written:

$$\mathbf{D}\text{diag}(\mathbf{A_v}\mathbf{e^m})\mathbf{G}\mathbf{x} = \mathbf{b} \tag{2}$$

where $\mathbf{D}$ and $\mathbf{G}$ are matrix representations of the divergence and gradient operators; $\mathbf{x}$ is a vector containing the potential difference field; and $\mathbf{b}$ contains the positive and negative source locations. $\mathbf{A_v}$ averages the conductivity values from cell-centers to cell-faces, and is included on the diagonal. The conductivity model of the medium, $\mathbf{m}$, is in log-conductivity, and is included via the exponential $\mathbf{e^m}$. Choosing to work in log conductivity (a) enforces a positivity constraint, and (b) allows for interpretations of the results in either conductivity or the inverse: resistivity. The entire forward operator is dependent on the conductivity model and can be written

$$\mathbf{A}(\mathbf{m})\mathbf{x}_i = \mathbf{b}_i \tag{3}$$

Where the subscript denotes the i^{th} source configuration in the survey; this equation must be solved for every RHS of interest. Neumann boundary conditions were implemented in the forward operator; however, these boundary conditions lead to a constant null space in the forward operator. The null space is removed by modifying cell $\mathbf{A}(1,1)$ by ± 1 to conform to the sign of that element [1]. It is noted that the data collected in any DC resistivity experiment is a subset of the entire potential field, thus a projection matrix, $\mathbf{P}$, is used to pick out these measured potentials

$$\mathbf{d}(\mathbf{m}) = \mathbf{P}\mathbf{x}_i = \mathbf{P}\mathbf{A}(\mathbf{m})^{-1}\mathbf{b}_i \tag{4}$$

These data, $\mathbf{d}(\mathbf{m})$, can now be directly compared to the observed data, $\mathbf{d_{obs}}$, generated for that same source configuration. In a DC-resistivity survey there are many RHSs to Equation 3, and each must be evaluated to compare with the full observational dataset.

2.2 Inverse Problem

In a field experiment, data is collected in the interest of obtaining knowledge about the conductivity structure of the subsurface; this is classified as the inverse problem. The DC resistivity inversion results in an unconstrained non-linear optimization problem that is often underdetermined; that is, observational data is much fewer than the number of model parameters [6]. Due to the inverse problem being underdetermined, a weighted regularization parameter is generally added to the optimization problem and the two-part function is minimized

$$\Phi(\mathbf{m}) = \sum_{i=1}^{m} \rho(\mathbf{d^{(i)}}(\mathbf{m}) - \mathbf{d^{(i)}_{obs}}) + \quad \frac{\beta}{2}\|\mathbf{G_w}(\mathbf{m} - \mathbf{m_{ref}})\|_2^2 \tag{5}$$

where the first term minimizes data-misfit between the generated data, $\mathbf{d}(\mathbf{m})$, and the observed data ($\mathbf{d_{obs}}$); $\rho(\cdot)$ is the data objective function (often taken as $\rho(\cdot) = (\cdot)^2$ to yield the sum of squares). Data is generated for each source configuration, and the individual data-misfit terms are summed over the data corresponding to the m different right-hand sides. The second term is a Tikhonov style term that controls model regularization with regard to a reference model ($\mathbf{m_{ref}}$) [7]. $\mathbf{G_w}$ is a combination of the gradient operator in Equation 2, which is sensitive to model flatness; and the identity matrix, which is sensitive to model smallness. The optimization problem is solved using an iterative method such as steepest descent or a quasi-Newton method [6]. Any method chosen, however, requires the computation of the gradient of the objective function; using a linearization about the current model, the gradient for one source configuration has the form:

$$\nabla\Phi(\mathbf{m}) = \mathbf{J^T}\nabla\rho + \beta\mathbf{G_w^T}\mathbf{G_w}(\mathbf{m} - \mathbf{m_{ref}}) \tag{6}$$

Here $\mathbf{J}$ is the Jacobian of the objective function ($^{\partial \mathbf{d}}/_{\partial \mathbf{m}}$) and describes the sensitivity of data to changes in model parameters. Note that this gradient must be computed once for each RHS. The Jacobian is a large and dense matrix, and it is preferable to avoid explicitly forming the matrix to reduce memory costs. Haber [3] shows the Jacobian has the form:

$$\mathbf{J} = \mathbf{PA}(\mathbf{m})^{-\mathbf{1}}(\mathbf{D}\ \mathrm{diag}(\mathbf{Gu})\mathbf{A_v}\ \mathrm{diag}(\mathbf{e^m})) \tag{7}$$

Using this expression for the Jacobian, it is seen that the full matrix $\mathbf{J}$ need not be formed explicitly, and only the effect of this matrix on a vector is needed. Two functions are needed to calculate a matrix-vector product, one that calculates $\mathbf{J}(\mathbf{v})$ and one that calculates $\mathbf{J^T}(\mathbf{w})$ for some vectors $\mathbf{v}$ and $\mathbf{w}$. It is noted that this formulation of $\mathbf{J}$ requires a linear system involving $\mathbf{A}(\mathbf{m})$ to be solved for each RHS at every iteration of the optimization algorithm.

2.3 Multiple Right-Hand Sides

The DC-resistivity inversion requires solving a linear system with $\mathbf{A}(\mathbf{m})$ in the evaluation of the forward problem as well as in the evaluation of the gradient. These linear systems must be completed with multiple RHSs and be computed at every iteration of an optimization algorithm. Additionally, the objective function may require more than one solve per iteration if line-search iterations are needed [8, 9]. To complicate matters, the system $\mathbf{A}(\mathbf{m})$ may be too large to handle with direct methods such as $\mathbf{LU}$ decompositions or Cholesky factorizations. It is clear that this linear system must be solved efficiently through the use of iterative methods.

3 Iterative Methods

Iterative methods allow for the computation of a linear system through a series of matrix vector products; unlike direct methods, there is no need to compute a factorization of the matrix $\mathbf{A}$. When computing the factorization $\mathbf{A}$ is prohibitively expensive, as is the case when $\mathbf{A}$ is very large and sparse, iterative methods can be used to find the solution of $\mathbf{Ax} = \mathbf{b}$. The appealing aspect of direct methods is their ability to share information between different RHSs; a factorization of $\mathbf{A}$ is determined, and to evaluate a new RHS only one forward and back solve must be completed. Information sharing between RHSs is not possible in standard iterative solvers and each RHS must be evaluated individually. The extension of these standard methods to share information can be completed in two ways: (a) seed systems and (b) block systems [10]. The seed system solves each RHS independently but stores information from the solve to use in subsequent solves. As information about the system is accumulated the additional RHSs converge to their solutions in fewer iterations [10]. In the block system all RHSs are solved at once, and information about the different systems is shared between all RHSs as the system converges [11]. The block system requires that all RHSs be known at the outset of the problem, while the seed system is generally used when all RHSs are not known at the outset. In the DC resistivity problem the RHSs are the source configurations of the survey; thus, they are all known at the outset and block methods are appropriate.

When $\mathbf{A}$ is sparse and symmetric positive definite, as is the case in this application, conjugate gradient is the iterative method of choice due to its low memory requirements and exact convergence in a finite number of iterations [8]. Conjugate gradient is presented in Section 4 and then extended to the corresponding block method in Section 5.

Although only conjugate gradient is presented in this paper, it should be noted that block methods exist for all popular iterative methods including minimum-residual (MINRES) for symmetric indefinite systems and generalized-minimum-residual (GMRES) for non-symmetric systems [12, 13, 10, 14].

4 Conjugate Gradient

Since the matrix $\mathbf{A}$ is very large, sparse, and symmetric positive definite it is a candidate for the iterative method of conjugate gradient (CG). CG relies on the positive definiteness of the system as it minimizes the objective function

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\mathbf{b}, \tag{8}$$

which is a convex quadratic of the matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, where $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ are the solution vector and the RHS vector respectively. The objective function (Equation 8) will only have a global minimum if the function is convex (i.e. $\mathbf{A}$ is positive definite). The gradient of the objective function

$$\nabla\phi(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{r}(\mathbf{x}) \tag{9}$$

can be seen to be equivalent to the residual ($\mathbf{r} \in \mathbb{R}^n$) of the original Equation 3. Additionally, it is clear that the minimum of $\phi(\cdot)$ will have a gradient of zero; thus solving $\mathbf{A}\mathbf{x} = \mathbf{b}$. An equivalent interpretation of CG is that it minimizes the $\mathbf{A}$-norm of the error

$$\|\mathbf{e}_k\|_{\mathbf{A}}^2 = \mathbf{e}_k^{\mathrm{T}}\mathbf{A}\mathbf{e}_k = (\mathbf{x} - \mathbf{x}_k)^{\mathrm{T}}\mathbf{A}(\mathbf{x} - \mathbf{x}_k) \tag{10}$$

where each new solution iterate $\mathbf{x}_k$ is contained in an expanding Krylov subspace

$$\mathbf{x}_k \in \mathbf{x}_0 + \text{span}\left\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\right\} \tag{11}$$

The CG iteration is very similar to a gradient descent algorithm (see [8]) with the exception of the search direction $\mathbf{p}_k \in \mathbb{R}^n$.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k \tag{12}$$

where the new approximate solution $\mathbf{x}_{k+1}$ is a linear combination of the previous iterate and a search direction $\mathbf{p}_k$. The step-length α_k is chosen such that is is the minimizer along the search direction $\mathbf{p}_k$; this is done by taking the directional derivative of the objective function and equating it to zero:

$$\mathbf{p}_k^{\mathrm{T}}\nabla\phi(\mathbf{x}_k + \alpha_k\mathbf{p}_k) = \mathbf{p}_k^{\mathrm{T}}(\mathbf{A}(\mathbf{x}_k + \alpha_k\mathbf{p}_k) - \mathbf{b}) = 0 \tag{13a}$$

$$\alpha_k = -\frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{p}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k} \tag{13b}$$

The important difference between conjugate gradient and gradient descent is the choice of $\mathbf{p}_k$, which is chosen such that the search direction comes from a set of non-zero $\mathbf{A}$-conjugate vectors $\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_k\}$ where

$$\mathbf{p}_i^{\mathrm{T}}\mathbf{A}\mathbf{p}_j = 0, \forall i \neq j. \tag{14}$$

This choice of conjugacy has the outcome that the conjugate set is linearly independent and most importantly $\phi(\cdot)$ can be minimized exactly in n steps given exact arithmetic. The proof of this is laid out in several texts (e.g. [8, 12]) and is generated through the observation that $\tilde{\mathbf{x}}$ will be the minimizer of $\phi(\cdot)$ over the conjugate set if and only if $\mathbf{r}(\tilde{\mathbf{x}})^{\mathrm{T}}\mathbf{p}_i = 0, \forall i = 0, 1, \ldots, k-1$. It can be shown that this is the case:

$$\mathbf{r}_k^{\mathrm{T}}\mathbf{p}_i = 0, \forall i = 0, 1, \ldots, k-1 \tag{15}$$

and that $\mathbf{x}_k$ is the exact minimizer of $\phi(\cdot)$ over the set

$$\{\mathbf{x}|\mathbf{x}_0 + \text{span}\{\mathbf{p}_0, \mathbf{p}_1, \ldots \mathbf{p}_{k-1}\}\}. \tag{16}$$

As the conjugate set expands to the full space $\mathbb{R}^n$ it is seen that the iterate will be the global minimizer of the objective function [8].

The construction of the conjugate set can be iteratively constructed by using the steepest descent direction, $-\nabla\phi(\mathbf{x}_k)$, (equivalently the current residual, $-\mathbf{r}_k$) and the previous search direction $\mathbf{p}_{k-1}$.

$$\mathbf{p}_k = -\mathbf{r}_k + \beta_k\mathbf{p}_{k-1} \tag{17}$$

where β_k is chosen such that the new search direction is $\mathbf{A}$-conjugate. This is completed by premultiplying Equation 17 by $\mathbf{p}_{k-1}^{\mathrm{T}}\mathbf{A}$ and imposing the conjugacy constraint in Equation 14:

$$\beta_k = \frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_{k-1}}{\mathbf{p}_{k-1}^{\mathrm{T}}\mathbf{A}\mathbf{p}_{k-1}} \tag{18}$$

From the previous equations, the standard Conjugate Gradient algorithm can be formed and is seen in Algorithm 1. It is noted that the equations for updating $\alpha_k, \beta_k,$ and $\mathbf{r}_{k+1}$ appear to be different than stated; the new forms can be obtained using Equations 15, 17. These rearrangements to the standard form of CG ensure only one matrix vector product must be computed every iteration; additionally, this form allows for reuse of some dot products. Conjugate Gradient allows for exact convergence to x^* in exactly n iterations, however, an approximate solution below a given tolerance is usually obtained in far fewer iterations. The exact convergence rates will depend on the eigenvalues of $\mathbf{A}$, where clustered eigenvalues hasten convergence [8]. Clustering of eigenvalues can be achieved through preconditioning of $\mathbf{A}$ and modifying the CG algorithm to incorporate the preconditioning.

Algorithm 1 Standard Conjugate Gradient

```
Given $\mathbf{x}_0$
Set $\mathbf{r}_0 \leftarrow \mathbf{A}\mathbf{x}_0 - \mathbf{b}, \mathbf{p}_0 \leftarrow -\mathbf{r}_0, k \leftarrow 0$
while $\mathbf{r}_k \neq 0$ do
    $\alpha_k \leftarrow \mathbf{r}_k^{\mathrm{T}}\mathbf{r}_k / \mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k$
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k\mathbf{p}_k$
    $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k + \alpha_k\mathbf{A}\mathbf{p}_k$
    $\beta_{k+1} \leftarrow \mathbf{r}_{k+1}^{\mathrm{T}}\mathbf{r}_{k+1} / \mathbf{r}_k^{\mathrm{T}}\mathbf{r}_k$
    $\mathbf{p}_{k+1} \leftarrow -\mathbf{r}_{k+1} + \beta_{k+1}\mathbf{p}_k$
    $k = k + 1$
end while
```

5 Block Conjugate Gradient

The extension of CG from one RHS to many, where each RHS is a column of $\mathbf{X}$, can be achieved through block conjugate gradient (BLCG). The new equation to be solved can be written

$$\mathbf{A}\mathbf{X} = \mathbf{B} \tag{19a}$$

$$\mathbf{A}\begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_m \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_m \\ | & | & & | \end{bmatrix} \tag{19b}$$

where there are m RHSs to be solved. It is possible to again write this equation as a minimization problem of an objective function

$$\mathbf{\Phi}(\mathbf{X}) = \sum_{i=1}^{m} \phi_i(\mathbf{x}_i) = \sum_{i=1}^{m} \frac{1}{2}\mathbf{x}_i^{\mathrm{T}}\mathbf{A}\mathbf{x}_i - \mathbf{x}_i^{\mathrm{T}}\mathbf{b} \tag{20}$$

Here, $\mathbf{\Phi}(\mathbf{X})$ contains all the objective functions from the standard formulation of CG (Equation 8). It is simple to see that the minimum of this function corresponds to the solution of Equation 19a if $\mathbf{A}$ is symmetric positive definite. The gradient of this objective function has a similar form as before

$$\nabla\mathbf{\Phi}(\mathbf{X}) = \mathbf{A}\mathbf{X} - \mathbf{B} = [\mathbf{A}\mathbf{x}_1 - \mathbf{b}_1, \mathbf{A}\mathbf{x}_2 - \mathbf{b}_2, \dots, \mathbf{A}\mathbf{x}_m - \mathbf{b}_m] := \mathbf{R}(\mathbf{X}) \tag{21}$$

where $\mathbf{R}(\mathbf{X})$ is defined as the block residual and $\mathbf{R}(\mathbf{X}_k)$ is denoted $\mathbf{R}_k \in \mathbb{R}^{n\times m}$. This is again equivalent to the minimization of the $\mathbf{A}$-norm of the error; however, here the Frobenius-norm is used (denoted $||\cdot||_F$).

$$\left\|(\mathbf{X}-\mathbf{X}_k)^{\mathrm{T}}\mathbf{A}(\mathbf{X}-\mathbf{X}_k)\right\|_F^2 = \text{trace}[(\mathbf{X}-\mathbf{X}_k)^{\mathrm{T}}\mathbf{A}(\mathbf{X}-\mathbf{X}_k)] \tag{22}$$

where each new solution iterate $\mathbf{X}_k$ is contained in an expanding block-Krylov subspace

$$\mathbf{X}_k \in \mathbf{X}_0 + \text{span}\left\{\mathbf{R}_0, \mathbf{A}\mathbf{R}_0, \dots, \mathbf{A}^{k-1}\mathbf{R}_0\right\} \tag{23}$$

The block-Krylov space is defined such that $\mathbf{x}_k^{(i)}$ (the i^{th} column of $\mathbf{X}_k$) can be a linear combination of **all** the Krylov spaces created thus far

$$\mathbf{x}_k^{(i)} \in \mathbf{x}_0^{(i)} + \bigcup_{i=1}^{m} \text{span}\left\{\mathbf{r}_0^{(i)}, \mathbf{A}\mathbf{r}_0^{(i)}, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0^{(i)}\right\} \tag{24}$$

where $\mathbf{r}_0^{(i)}$ denotes the i^{th} column of $\mathbf{R}_0$. This information sharing technique is the desirable aspect of BLCG. The new approximate solution is updated through

$$\mathbf{X}_{k+1} = \mathbf{X}_k + \mathbf{P}_k \mathbf{\Lambda}_k \tag{25}$$

where $\mathbf{P}_k \in \mathbb{R}^{n\times m}$ spans an m-dimensional search subspace; and $\mathbf{\Lambda}_k \in \mathbb{R}^{m\times m}$ is now a step-length matrix that minimizes $\mathbf{\Phi}(\mathbf{X})$ over the entire search space. The minimum of $\mathbf{\Phi}(\mathbf{X})$ is found similarly to CG

$$\frac{\partial \mathbf{\Phi}(\mathbf{X}_k + \mathbf{P}_k\mathbf{\Lambda}_k)}{\partial \mathbf{\Lambda}_k} = \mathbf{P}_k^{\mathrm{T}}(\mathbf{A}(\mathbf{X}_k + \mathbf{P}_k\mathbf{\Lambda}_k) - \mathbf{B}) = 0 \tag{26a}$$

$$\mathbf{\Lambda}_k = (\mathbf{P}_k^{\mathrm{T}}\mathbf{A}\mathbf{P}_k)^{-1}\mathbf{P}_k^{\mathrm{T}}\mathbf{R}_k \tag{26b}$$

It is noted that the step-length matrix requires the solve of a linear system, implying that the columns of $\mathbf{P}_k^{\mathrm{T}}\mathbf{A}\mathbf{P}_k$ must be linearly independent; enforcing this linear independence will be discussed further in Section 6.

Similar to CG, the new search subspace is created from the current block residual and the previous search subspace

$$\mathbf{P}_k = -\mathbf{R}_k + \mathbf{P}_{k-1}\mathbf{\Psi}_k \tag{27}$$

where $\mathbf{\Psi}_k \in \mathbb{R}^{m\times m}$ is chosen such that the next search space is $\mathbf{A}$-conjugate to the current search space

$$\mathbf{P}_i^{\mathrm{T}}\mathbf{A}\mathbf{P}_j = 0, \forall i \neq j \tag{28}$$

By premultiplying Equation 27 by $\mathbf{P}_{k-1}^{\mathrm{T}}\mathbf{A}$ and enforcing $\mathbf{A}$-conjugacy of the next search space, $\mathbf{\Psi}_k$ can be found to be

$$\mathbf{\Psi}_k = (\mathbf{P}_{k-1}^{\mathrm{T}}\mathbf{A}\mathbf{P}_{k-1})^{-1}\mathbf{P}_{k-1}^{\mathrm{T}}\mathbf{A}\mathbf{R}_k \tag{29}$$

The coefficient matrix $\mathbf{\Psi}_k$ again requires an $m \times m$ linear system solve; and requires linear independence. By combining the equations for BLCG an algorithm can be formed, and is laid out in Algorithm 2. Again, there are some slight changes in the coefficient matrices to simplify such that only one product $\mathbf{A}\mathbf{P}_k$ is required; these rearrangements are the same as in CG but generalized to matrix form.

Algorithm 2 Block Conjugate Gradient - Basic

```
Given $\mathbf{X}_0$
Set $\mathbf{R}_0 \leftarrow \mathbf{A}\mathbf{X}_0 - \mathbf{B}, \mathbf{P}_0 \leftarrow -\mathbf{R}_0, k \leftarrow 0$
while $\mathbf{R}_k \neq 0$ do
    $\mathbf{\Lambda}_k \leftarrow (\mathbf{P}_k^{\mathrm{T}}\mathbf{A}\mathbf{P}_k)^{-1}\mathbf{R}_k^{\mathrm{T}}\mathbf{R}_k$
    $\mathbf{X}_{k+1} \leftarrow \mathbf{X}_k + \mathbf{P}_k\mathbf{\Lambda}_k$
    $\mathbf{R}_{k+1} \leftarrow \mathbf{R}_k + \mathbf{A}\mathbf{P}_k\mathbf{\Lambda}_k$
    $\mathbf{\Psi}_{k+1} \leftarrow (\mathbf{R}_k^{\mathrm{T}}\mathbf{R}_k)^{-1}\mathbf{R}_{k+1}^{\mathrm{T}}\mathbf{R}_{k+1}$
    $\mathbf{P}_{k+1} \leftarrow -\mathbf{R}_{k+1} + \mathbf{P}_k\mathbf{\Psi}_{k+1}$
    $k = k + 1$
end while
```

6 Deflation and Linear Independence

The linear independence of $\mathbf{P}_k$ and $\mathbf{R}_k$ must be maintained for the two linear solves in BLCG to be possible [15]. It can be shown that, for each value of k, the ranks of $\mathbf{P}_k$ and $\mathbf{R}_k$ are equal [15].

$$\text{span}\{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_k\} = \text{span}\{\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_k\} \tag{30}$$

This is due the construction of $\mathbf{R}_k \in \text{span}\{\mathbf{R}_{k-1}, \mathbf{A}\mathbf{P}_{k-1}\}$ and is shown in [15]. Monitoring of linear dependent columns and subsequent removal of dependent columns of either $\mathbf{P}_k$ or $\mathbf{R}_k$ can be used to ensure that the two inverses are always defined. The process of reducing the number of RHSs to maintain linear independence is known as deflation; this process can occur before the algorithm starts or at a later step.

6.1 Initial Deflation

Initial deflation occurs on $\mathbf{R}_0$ before the BLCG algorithm begins. Typically this is done through a rank-revealing QR factorization involving column pivoting [10]. For example, if rank$(\mathbf{R}_0) < m$ then the number of RHSs to be solved can be reduced via initial deflation to rank$(\mathbf{R}_0)$ RHSs.

$$\mathbf{R}_0 = \mathbf{AX}_0 - \mathbf{B} = \mathbf{QR} = \begin{bmatrix} | & | & | & | \\ \mathbf{q_1} & \mathbf{q_2} & \mathbf{q_3} & \mathbf{q_4} \\ | & | & | & | \end{bmatrix} \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & & & & & \\ & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\ & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & & & \bullet & \bullet & & \bullet & \bullet & \bullet & \bullet \end{bmatrix} \tag{31}$$

In this small example, the number of RHSs to be solved has been reduced from 10 to 4 using an economy-QR decomposition. Initial deflation is incorporated into BLCG (Algorithm 3) through an economy QR-decomposition and the entire algorithm is competed on $\mathcal{Q}$ rather than on the full number of RHSs. This can be thought of as doing a QR-decomposition of $\mathbf{B}$

$$\mathbf{AX} = \mathbf{B} = \mathcal{QR}, \tag{32}$$

and then solving for $\mathbf{X}$ using $\mathcal{Q}$ only, and then post multiplying by $\mathcal{R}$

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{B} = (\mathbf{A}^{-1}\mathcal{Q})\mathcal{R} \tag{33}$$

This is equivalent to the initial deflation described above (Equation 31) with an $\mathbf{X}_0$ of 0. Another special case that is worth considering is when $\mathbf{R}_0$ has m identical columns; in this case the economy-QR decomposition reduces from m RHSs to one. If this special case was handled by a standard CG algorithm, m RHSs would still have to be solved. It is clear that using block methods can be extremely effective when deflation is possible.

6.2 Internal Deflation

Often when a linear system converges it is necessary to perform deflation inside the algorithm. The rank of $\mathbf{P}_k$ and $\mathbf{R}_k$ can be monitored through a rank-revealing orthogonalization procedure; when linear dependencies are detected, these dependent columns are removed from $\mathbf{X}_k$, $\mathbf{R}_k$, and $\mathbf{P}_k$ [16]. The dependent systems are treated separately from the BLCG algorithm, and both are able to converge [12]. A special case that will result in the convergence of linear systems is when the initial residual has the the form of a Krylov subspace.

$$\mathbf{R}_0 = \begin{bmatrix} | & | & | & | \\ \mathbf{r}_1 & \mathbf{Ar}_1 & \dots & \mathbf{A}^{m-1}\mathbf{r}_1 \\ | & | & | & | \end{bmatrix} \tag{34}$$

Although $\mathbf{R}_0$ may be full rank, and thus not be changed by initial deflation, after one iteration the first Krylov space created, $[\mathbf{R}_0, \mathbf{AR}_0]$, will only have a rank of $m + 1$. Column deflation must be completed on this new space for the BLCG algorithm to continue. The chance of several systems converging at once in practical applications, however, is small and does not always warrant the work required for the orthogonalization procedure.

6.3 Unequal Convergence

Another, algorithm presented by Feng *et al.* [17] allows for unequal convergence of the linear systems. In addition to incorporating initial deflation, unequal convergence of the RHSs is addressed by individually calculating the l_2-norm of each column of $\mathbf{R}_0$ (Algorithm 3). If this norm is below a convergence tolerance, ϵ, then the entire column is removed from $\mathbf{R}_k$, $\mathbf{X}_k$, and $\mathbf{P}_k$. Computationally this removal is done through indexing, once a column has converged the index is deleted and no more work is done to that column; at the end of the algorithm all columns can then be used. This adaptive solution allows for the early convergence of some RHSs without introducing numerical instabilities. Although this is not as robust as performing rank-revealing QR factorizations at every iteration, it performs well under mild conditions, is faster, and works well in many applications [17]. Algorithm 3 will fail if linear dependencies arise before convergence and more rigorous steps must be taken as outlined in Section 6.2. However, in practical applications these rank-revealing orthogonalization steps are seldom necessary and are a waste of computational resources [18, 17].

7 Comparisons & Other Methods

Standard CG will converge exactly in n iterations given exact arithmetic; however, the iteration count will be much smaller in practice to get an approximate solution [8]. The algorithm for BLCG will converge exactly in n/m iterations

Algorithm 3 Block Conjugate Gradient - Unequal Convergence

```
Given $\mathcal{X}_0$
Orthogonalize $\mathbf{A}\mathcal{X}_0 - \mathbf{B} = \mathcal{Q}\mathcal{R}$
Set $\mathbf{R}_0 \leftarrow \mathcal{Q}, \mathbf{P}_0 \leftarrow -\mathbf{R}_0, \mathbf{X}_0 \leftarrow 0, k \leftarrow 0$
for $k = 0, 1, 2, \ldots$ until convergence do
    $\mathbf{\Lambda}_k \leftarrow (\mathbf{P}_k^{\mathrm{T}}\mathbf{A}\mathbf{P}_k)^{-1}\mathbf{R}_k^{\mathrm{T}}\mathbf{R}_k$
    $\mathbf{X}_{k+1} \leftarrow \mathbf{X}_k + \mathbf{P}_k\mathbf{\Lambda}_k$
    $\mathbf{R}_{k+1} \leftarrow \mathbf{R}_k + \mathbf{A}\mathbf{P}_k\mathbf{\Lambda}_k$
    if $||\mathbf{R}_{k+1}^{(i)}|| < \epsilon$ then
        exit vector $i$
    end if
    if all vectors have exited then
        break
    end if
    $\mathbf{\Psi}_{k+1} \leftarrow (\mathbf{R}_k^{\mathrm{T}}\mathbf{R}_k)^{-1}\mathbf{R}_{k+1}^{\mathrm{T}}\mathbf{R}_{k+1}$
    $\mathbf{P}_{k+1} \leftarrow -\mathbf{R}_{k+1} + \mathbf{P}_k\mathbf{\Psi}_{k+1}$
    $k = k + 1$
end for
$\mathcal{X}^* = \mathcal{X}_0 + \mathbf{X}_{k+1}\mathcal{R}$
```

given exact arithmetic and linear independence of $\mathbf{P}_k$ and $\mathbf{R}_k$ [15] [17]. This reduction of the iterations required comes from the definition of the block-Krylov subspace that the iterates are generated from (Eq. 24), which is inherently much larger than in CG. Although the number of iterations is significantly reduced in BLCG, the work per iteration is increased due to the two linear solves. Additionally, the amount of memory is increased in BLCG from four length n vectors in CG to four $n \times m$ matrices in BLCG. In BLCG it is also possible to modify the matrix-vector products for all the RHSs ($\mathbf{A}\mathbf{P}_k$) to run in parallel. Parallelization of the BLCG algorithm can lead to additional gains [15, 17]. If the number of RHSs is so large that the two $m \times m$ linear solves every iteration are prohibitive, the RHSs could be broken into smaller sections and run with multiple BLCG algorithms (potentially in parallel). The biggest gains in using BLCG come when deflation is possible and enforced [10].

The implementations of other block methods such as MINRES and GMRES have similar intricacies involving deflation of the linear systems. In these methods, however, there is significantly more storage required. In MINRES, for example, the three term recurrence in the standard version is extended to $2m + 1$, where m is the number of RHSs. In both GMRES and MINRES the single sub-diagonal in the least-squares problem is replaced by m sub-diagonals [19]. The m sub-diagonals in the least squares problem are effectively dealt with by Householder reflections rather than Given's rotations that are the standard in the single RHS methods [20]. The extension of MINRES and GMRES to their corresponding block methods is effective for systems that are symmetric indefinite or non-symmetric, respectively. Again, as with all block methods, significant gains are only seen in these algorithms when it is possible to decrease the size of the system significantly via deflation.

8 Numerical Experiments

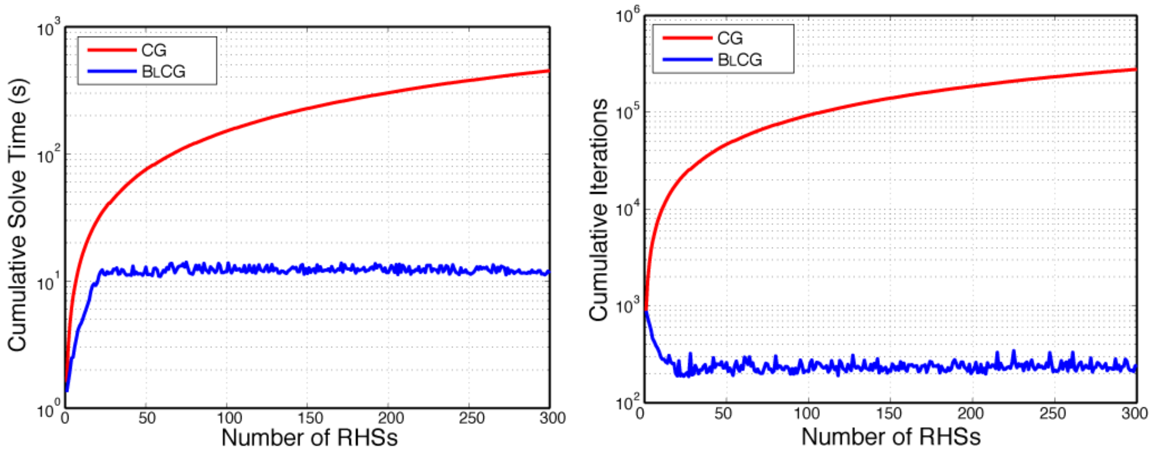
A forward modelling code was written that implemented the DC resistivity problem (Equations 2-4) over a unit cube with Neumann boundary conditions. The forward operators were tested for analytical potential fields with the appropriate boundary conditions. A series of electrode arrays (surveys) were written to produce and collect data from the forward model; the survey used in this paper considered all receiver permutations in a grid on the top surface of the model. It is noted that it is not possible experimentally to collect data at the same location as the source electrodes; and these permutations of source-receiver pairs were not included.

8.1 Experimental Setup

The models used for this experiment were 16x16x16 with a survey grid spacing of 3 cells centered on the upper surface of the model. There were a total of 25 electrodes, 300 source configurations, and 253 active measurements per source dipole. This gave rise to 75 900 total measurements half of which are symmetric and likely would not have been collected in a field experiment, but were collected in this numerical experiment. The forward operator $\mathbf{A}(\mathbf{m})$ has a size of 4096×4096, and is structurally identical to the standard Laplacian in three dimensions. The stopping criteria in both the CG and BLCG algorithms is based on the relative residual and is set to 10^{-5}. The relative residual must be used in BLCG as conditioning of the solver deteriorates if excessive precision is required; the two linear system solves inside the BLCG iteration tend to become ill-conditioned if precisions higher than approximately 10^{-7} are required. All experiments using BLCG are completed with the implementation laid out in Algorithm 3.

8.2 Comparison of CG and BLCG

The CG algorithm was run for all 300 source configurations independently and compared to the cumulative run time of theBLCG algorithm; the results for this experiment are seen in Figure 1. The CG algorithm took 450 seconds (2.8×10^5 iterations) to complete the calculation of all the RHSs versus just 12 seconds (2.2×10^2 iterations) to complete the calculation using BLCG. The matrix $\mathbf{B}$ for this experiment only had 24 independent columns even though there were 300 different source configurations; thus, the economy $\mathbf{QR}$ decomposition significantly reduced the size of the problem. Initial deflation produces 24 independent RHSs (down from 300), and explains the attractive plateau in both the cumulative solve time in Figure 1(a) and the cumulative iteration count in Figure 1(b). In CG both the iteration count and the solve time increase linearly with each new RHS to be solved, however, in BLCG there are significant gains in solving the systems jointly. The iteration counts in BLCG also decrease dramatically as information is added to the search space in the form of new RHSs; each CG solve took approximately 900 iterations while BLCG decreased to half that number in just 5 iterations.

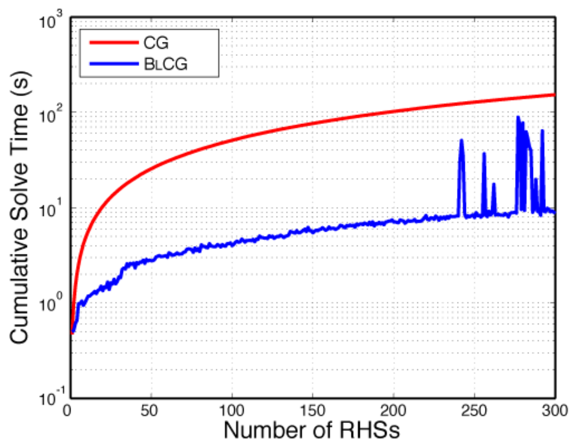


(a) Cumulative solve time in seconds per RHS

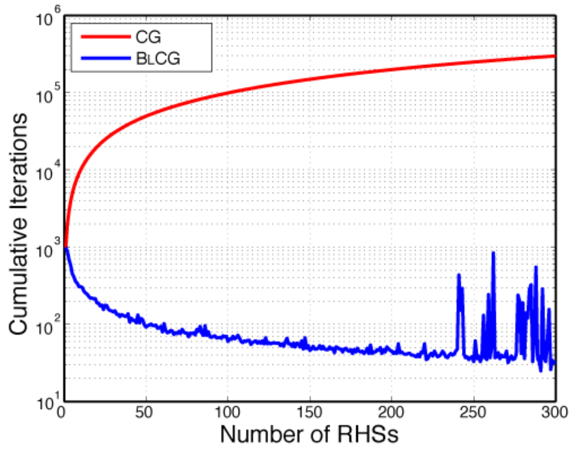
(b) Cumulative number of iterations per RHS

Figure 1: Comparison of cumulative work required between CG and BLCG for 300 source configurations.

In addition to running the CG and BLCG algorithms on the 300 source configurations, they were also run on a random RHS matrix (Figure 2). Using a random RHS matrix means that initial deflation is not possible, as all 300 vectors are linearly independent. The plateau effect seen before in Figure 1 is no longer apparent, and the addition of new RHSs means increased work for the BLCG solver. The CG algorithm solved the 300 random RHS vectors in 150 seconds (3.0×10^5 iterations) verses the BLCG algorithm that solved the same 300 RHSs in just 8 seconds (32 iterations). Although the gains made by BLCG are not as dramatic as when initial deflation can be implemented, the gains are still significant. As the number of RHSs increased above 230 in the BLCG solver, numerical instabilities were present in two linear system solves for $\mathbf{\Lambda}_k$ and $\mathbf{\Psi}_k$. The ill-conditioning present caused the BLCG algorithm to take much longer to converge, and in some cases it did not converge. This behaviour can be avoided by breaking up the number of RHSs into smaller sections that are easily solved by BLCG; for example, doing two BLCG solves on 150 RHSs instead of one with 300. This division could also be implemented in parallel, which would additionally increase performance.



(a) Cumulative solve time in seconds per RHS



(b) Cumulative number of iterations per RHS

Figure 2: Comparison of cumulative work required between CG and BLCG for 300 random RHSs.

9 Discussion and Conclusions

The DC-resistivity inverse problem requires multiple solves of the linear system $\mathbf{A}(\mathbf{m})$ with numerous RHSs. The linear system must be solved efficiently with a low memory footprint. Due to the very large and sparse nature of $\mathbf{A}$, as well as because it is symmetric positive-definite, conjugate gradient is an ideal iterative solver. However, in standard CG there is no way to share information between RHSs; thus, all RHSs must be solved individually. In this paper I have described and implemented a block conjugate gradient solver. The beneficial aspects to using this solver is that information is shared between RHSs and the system converges to the solution faster and in fewer iterations. In standard CG there is exact convergence in n iterations, BLCG has exact convergence in n/m iterations due to the increased search space. A major hurdle for implementing BLCG is maintaining linear independence of the individual systems. Initial deflation was used to combat dependencies between systems as well as to decrease the number of RHSs to solve. Additionally, an unequal convergence scheme was implemented that monitored the individual systems and removed them from BLCG when they converged. This unequal convergence scheme numerically stabilizes most practical systems at no additional cost.

Numerical experiments were run using 300 source configurations (RHSs) from a synthetic DC resistivity. Due to the source configurations sharing locations, the number of dependent vectors in **B** is much fewer than the number of source configurations. The economy QR decomposition of **B** has a form very similar to Equation 31, with only 24 distinct vectors in this case (much less than the initial 300 RHSs). This leads to significant deflation of the number of RHSs to solve in BLCG and results in significant gains over CG. The fact that significant initial deflation is possible is not an artifact of this particular experimental design; any DC resistivity survey shares physical electrode locations, thus initial deflation of **B** will always be possible. Numerical experiments were also completed using a random RHS matrix with 300 columns; here, due to the independence of the RHSs, initial deflation was not possible. However, BLCG still significantly outperformed CG in both cumulative solve time and iteration count. Some numerical instabilities exist when the number of RHSs is very large; in this case, initially breaking up the number of RHSs into two or more sub-blocks can be beneficial.

Block methods of standard iterative solvers can greatly reduce both the number of iterations and the solve times of large systems containing multiple RHSs. If initial deflation is possible and enforced the number of actual systems solved can be dramatically reduced. In this paper, I have presented a cheap and effective way to implement block conjugate gradient. It is clear that BLCG can be used to make significant gains when solving the DC resistivity problem.

Acknowledgments

Multiple conversations with Chen Grief were insightful and provided many avenues of research to explore. Eldad Haber provided invaluable guidance on the discretization of the partial differential equation.

References

[1] Adam Pidlisecky, Eldad Haber, and Rosemary Knight. RESINVM3D : A 3D resistivity inversion package. *Geophysics*, 72(2), 2007.

[2] Esben Auken and Anders Vest Christiansen. Layered and laterally constrained 2D inversion of resistivity data. *Geophysics*, 69(3):752–761, 2004.

[3] Eldad Haber, U.M. Ascher, and D.W. Oldenburg. On optimization techniques for solving nonlinear inverse problems. *Inverse Problems*, 16:1263–1280, 2000.

[4] William Daily, Abelardo Ramirez, and Andrew Binley. Electrical resistance tomography. *The Leading Edge*, 2:438–442, 2004.

[5] Eldad Haber. *Computational Methods for Simulation Based Optimization*. UBC Math, 2011.

[6] Richard C Aster, Brian Borchers, and Clifford Thurber. *Parameter Estimation and Inverse Problems*. Elsevier Inc., 2004.

[7] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-Posed Problems*. W.H. Winston and Sons., 1977.

[8] Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, New York, NY, 1999.

[9] Larry Armijo. Minimization Of Functions Having Lipschitz Continuous First Partial Derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

[10] Martin H Gutknecht. Block Krylov Space Methods For Linear Systems With Multiple Right-Hand Sides: An Introduction. In A.H. Siddiqi, I.S. Duff, and O. Christensen, editors, *Modern Mathematical Models, Methods and Algorithms for Real World Systems*, pages 420–447. Anamaya Publishers, New Delhi, India, 2007.

[11] A A Nikishin and A. Yu. Yeremin. Variable Block CG Algorithms for Solving Large Sparse Symmetric Positive Definite Linear Systems on Parallel Computers, I: General Iterative Scheme. *SIAM Journal on Matrix Analysis and Applications*, 16(4):1135–1153, 1995.

[12] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, January 1997.

[13] Axel Ruhe. Implementation Aspects of Band Lanczos Algorithms for Computation of Eigenvalues of Large Sparse Symmetric Matrices. *Mathematics of Computation*, 33(146):680–687, 1979.

[14] V. Simoncini and E. Gallopoulos. An Iterative Method for Nonsymmetric Systems with Multiple Right-Hand Sides. *SIAM J. Sci. Comput.*, 16(4):917–933, 1995.

[15] Dianne P. O'Leary. Parallel implementation of the block conjugate gradient algorithm. *Parallel Computing*, 5:127–139, 1987.

[16] Dianne P. O'Leary. The Block ConJugate Gradlent Algorlthm and Related Methods. *Linear Algebra and its Applications*, 29:293–322, 1980.

[17] Y T Feng, D R J Owen, and D Peric. A block conjugate gradient method applied to linear systems with multiple right-hand sides. *Computer Methods in Applied Mechanics and Engineering*, 127(1-4):203–215, 1995.

[18] J.F. McCarthy. Block-conjugate-gradient method. *The American Physical Society*, 40(6):2149–2152, 1989.

[19] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2000.

[20] Martin H. Gutknecht and Thomas Schmelzer. Updating the QR decomposition of block tridiagonal and block Hessenberg matrices. *Applied Numerical Mathematics*, 58(6):871–883, June 2008.