CS 6210: Matrix Computations

CG and GMRES

David Bindel

2025-11-19

References

There is a lot of ground to cover when it comes to Krylov subspace methods, and we scarcely have time to do justice to the two most popular Krylov subspace methods (CG for the SPD case and GMRES elsewhere). Apart from the material in Golub and Van Loan and other standard texts, I highly recommend two books for a survey of other methods and some practical details:

1. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* by Barrett *et al.* This is freely available, and includes what you need to know to get started with various methods.
2. *Iterative methods for sparse linear systems* by Y. Saad. This is now in a second edition (available from SIAM), but you can also get the first edition at Saad's web page.
3. *Iterative methods for solving linear systems* by Anne Greenbaum (published by SIAM) is a fairly up-to-date treatment of the major iterative solvers for linear systems, including the whole family of Krylov subspace solvers as well as classical stationary iterations and multigrid methods.
4. *Iterative methods for linear and nonlinear equations* by C. T. Kelley is another SIAM book — are you seeing a theme? It covers CG and GMRES, though not the other Krylov iterations; however, it also covers nonlinear iterations. It is short and tutorial in nature.

For the conjugate gradients method, another popular introduction is Shewchuk's "Introduction to the Conjugate Gradient Method Without the Agonizing Pain".

Conjugate gradients

We now turn to the method of conjugate gradients (CG), perhaps the best known of the Krylov subspace solvers. The CG iteration can be characterized as the iteration for a symmetric positive definite A that minimizes the energy

$$\phi(x) = \frac{1}{2}x^T Ax - x^T b$$

over a Krylov subspace; as we have already seen,

$$\phi(x) + \frac{1}{2}b^T A^{-1} b = \frac{1}{2}\|x - A^{-1}b\|_A^2 = \frac{1}{2}\|Ax - b\|_{A^{-1}}^2,$$

so this minimization corresponds to minimizing the error in the A-norm or the residual in the A^{-1} norm. We also have seen the shape of the standard error analysis, which involves looking at a Chebyshev polynomial on an interval containing the spectrum. The iteration turns out to be forward unstable, so the behavior in floating point arithmetic is not the same as the behavior in theory; but this does not prevent the iteration from being highly effective, partly because we can write the iteration in a form that involves an explicit residual, and looking at a freshly-computed residual gives the method a self-correcting property.

Our goal for today is to look at the mechanics of the method.

CG via Lanczos

Last time, we discussed the Lanczos iteration, which produces the Lanczos decomposition

$$AQ_k = Q_{k+1}\bar{T}_k$$

via the iteration

$$\beta_k q_{k+1} = Aq_k - \alpha_k q_k - \beta_{k-1} q_{k-1}$$

where $\alpha_k = q_k^T A q_k$. One of the simplest derivations for the *conjugate gradient* (CG) method is in terms of the Lanczos decomposition.

In terms of the energy

$$\phi(x) = \frac{1}{2}x^T Ax - x^T b,$$

the problem of finding the "best" (minimum energy) approximate solution in the space becomes

$$\text{minimize } \phi(Q_k y_k) = \frac{1}{2}y_k^T T_k y_k - y_k^T e_1 \|b\|,$$

which is solved by

$$T_k y_k = e_1 \|b\|.$$

Now let us suppress the indices for a moment and write $T = LU$ (which can be computed stably without pivoting, as T is SPD). Then we can write the approximate solution $\hat{x}$ as

$$\hat{x} = QU^{-1}L^{-1}e_1\|b\|,$$

which we will group as

$$\hat{x} = V\hat{y}, \quad VU = Q, \quad Ly = e_1\|b\|.$$

Solving the system for y by forward substitution yields

$$\begin{aligned} y_1 &= \|b\| \\ y_k &= -l_{k,k-1}y_{k-1}. \end{aligned}$$

Similarly, we can compute the columns of V by forward substitution:

$$\begin{aligned} v_1 &= q_1/u_{11} \\ v_k &= \frac{1}{u_{kk}}\left(q_k - v_{k-1}u_{k-1,k}\right). \end{aligned}$$

The advantage of this formulation is that if we extend the Krylov subspace, we simply extend the tridiagonal (and associated factorization), add another component to y, and bring in a new vector v — all without disturbing the computations done before. Hence, we have a sequence of coupled recurrences for the columns of Q and of V that allow us to incrementally update the solution at the cost of a matrix-vector multiply and a constant amount of vector arithmetic per step.

This is a useful approach, but it does not shed much insight into how the method could be extended to optimize more general objectives than quadratics. For that, we need the approach that gives the CG method its name.

Another approach to CG

An alternate approach to the conjugate gradient method does not directly invoke Lanczos, but instead relies on properties that must be satisfied at each step by the residual $r_m = b - Ax_m$ and the update $d_m = x_{m+1} - x_m$. We assume throughout that x_m is drawn from $\mathcal{K}_m(A, b)$, which implies that $r_m \in \mathcal{K}_{m+1}(A, b)$ and $d_m \in \mathcal{K}_{m+1}(A, b)$.

First, note that $r_m \perp \mathcal{K}_m(A, b)$ and $d_m \perp_A \mathcal{K}_m(A, b)$.[1] The former statement comes from the Galerkin criterion in the previous section. The latter statement comes from recognizing that $r_{m+1} = Ad_m + r_m \perp \mathcal{K}_m(A, b)$; with Galerkin condition $r_m \perp \mathcal{K}_m(A, b)$, this means $Ad_m \perp \mathcal{K}_m(A, b)$. Together, these statements give us r_m and d_m to within a scalar factor, since there is only one direction in $\mathcal{K}_{m+1}(A, b)$ that is orthogonal to all of $\mathcal{K}_m(A, b)$, and similarly there is only one direction that is A-orthogonal. This suggests the following idea to generate the sequence of approximate solutions x_k:

[1] $u \perp_A v$ means u and v are orthogonal in the A-induced inner product, i.e. $u^T Av = 0$.

1. Find a direction $p_{k-1} \in \mathcal{K}_k(A, b)$ that is A-orthogonal to $\mathcal{K}_{k-1}(A, b)$.
2. Compute $x_k = x_{k-1} + \alpha_k p_{k-1}$ so that
$$r_k = r_{k-1} - \alpha_k A p_{k-1} \perp r_{k-1},$$
i.e. set $\alpha_k = (r_{k-1}^T r_{k-1})/(p_{k-1}^T A p_{k-1})$. Orthogonality to the rest of $\mathcal{K}_k(A, b)$ follows automatically from the construction.
3. Take $r_k \in \mathcal{K}_{k+1}(A, b)$ and A-orthogonalize against everything in $\mathcal{K}_k(A, b)$ to generate the new direction p_k. As with the Lanczos procedure, the real magic in this idea is that we have to do very little work to generate p_k from r_k. Note that for any $j < k - 1$, we have $p_j^T A r_k = (Ap_j)^T r_k = 0$, because $Ap_j \in \mathcal{K}_{j+2}(A, b) \subset \mathcal{K}_k(A, b)$ is automatically orthogonal to r_k. Therefore, we really only need to choose
$$p_k = r_k + \beta p_{k-1},$$
such that $p_{k-1}^T A p_k$, i.e. $\beta_k = -(p_{k-1}^T A r_k)/(p_{k-1}^T A p_{k-1})$. Note, though, that $Ap_{k-1} = -(r_k - r_{k-1})/\alpha_k$; with a little algebra, we find
$$\beta_k = -\frac{r_k^T A p_k}{p_{k-1}^T A p_{k-1}} = \frac{(r_k^T r_k)/\alpha_k}{r_{k-1}^T r_{k-1}/\alpha_k} = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}.$$

Putting everything together, we have the following coupled recurrences for the solutions x_k, residuals r_k, and search directions p_k:
$$\begin{aligned}
\alpha_k &= (r_{k-1}^T r_{k-1})/(p_{k-1}^T A p_{k-1}) \\
x_k &= x_{k-1} + \alpha_k p_{k-1} \\
r_k &= r_{k-1} - \alpha_k A p_{k-1} \\
\beta_k &= (r_k^T r_k)/(r_{k-1}^T r_{k-1}) \\
p_k &= r_k + \beta_k p_{k-1}.
\end{aligned}$$
The sequences r_k and p_k respectively form orthogonal and A-orthogonal bases for the nested Krylov subspaces generated by A and b.

Preconditioning

What happens if we want to compute not on the space $\mathcal{K}_k(A, b)$, but the preconditioned space $\mathcal{K}_k(M^{-1}A, M^{-1}b)$ where M is some symmetric positive definite matrix? Unfortunately, we cannot apply CG directly to a system involving $M^{-1}A$, since even if M and A are SPD, the product will generally not be. On the other hand, we can certainly work with the related system
$$(M^{-1/2} A M^{-1/2})(M^{1/2} x) = M^{-1/2} b.$$

This is a symmetric positive definite system, and the eigenvalues of $M^{-1/2}AM^{-1/2}$ are the same as the generalized eigenvalues of the pencil (A, M). Moreover, we can work with this system *implicitly* without ever having to form the awkward square root.

Define $\bar{p}_k = M^{-1/2}p_k$ and $\bar{r}_k = M^{1/2}r_k$; then CG iteration on the related system can be rephrased as

$$\begin{aligned}
\alpha_k &= (\bar{r}_{k-1}^T M^{-1}\bar{r}_{k-1})/(\bar{p}_{k-1}^T A\bar{p}_{k-1}) \\
x_k &= x_{k-1} + \alpha_k \bar{p}_{k-1} \\
\bar{r}_k &= \bar{r}_{k-1} - \alpha_k A\bar{p}_{k-1} \\
\beta_k &= (\bar{r}_k^T M^{-1}\bar{r}_k)/(\bar{r}_{k-1}^T M^{-1}\bar{r}_{k-1}) \\
\bar{p}_k &= M^{-1}\bar{r}_k + \beta_k \bar{p}_{k-1}.
\end{aligned}$$

Because expressions involving M^{-1} and the residual appear throughout, we introduce a new variable $z_k = M^{-1}r_k$, leading to

$$\begin{aligned}
\alpha_k &= (\bar{r}_{k-1}^T z_{k-1})/(\bar{p}_{k-1}^T A\bar{p}_{k-1}) \\
x_k &= x_{k-1} + \alpha_k \bar{p}_{k-1} \\
\bar{r}_k &= \bar{r}_{k-1} - \alpha_k A\bar{p}_{k-1} \\
Mz_k &= r_k \\
\beta_k &= (\bar{r}_k^T z_k)/(\bar{r}_{k-1}^T z_{k-1}) \\
\bar{p}_k &= z_k + \beta_k \bar{p}_{k-1}.
\end{aligned}$$

Another way of thinking about the preconditioned CG iteration is that it is ordinary CG, whether thought of in terms of conjugate directions or in terms of Lanczos, but with a different inner product: the M^{-1} inner product on residuals, or the M inner product in the Lanczos procedure.

Nonlinear CG

One of the advantages of the interpretation of CG in terms of search directions and residuals is that it generalizes beyond the case of quadratic optimization or linear system solving to more general optimization problems. To derive nonlinear CG, we generalize the quantities in the ordinary CG iteration in the following way:

- In ordinary CG, we let ϕ be a quadratic energy function. In nonlinear CG, ϕ is a more general (though ideally convex) objective function.
- In ordinary CG, we have $r_k = -\nabla\phi(x_k) = b - Ax_k$. In nonlinear CG, we take $r_k = -\nabla\phi(x_k)$, though the gradient expression will generally be more complicated.

- In ordinary CG, we choose a search direction $p_k = r_k + \beta_k p_{k-1}$ where $\beta_k = r_k^T r_k / r_{k-1}^T r_{k-1}$. In nonlinear CG, we may use the same formula (the *Fletcher-Reeves* formula), or we may choose any number of other formulas that are equivalent in the quadratic case but not in the more general case.
- In ordinary CG, once we choose a search direction p_{k-1}, we compute a step $x_k = x_{k-1} + \alpha_k p_{k-1}$. The α_k has the property

  $$\alpha_k = \operatorname{argmin}_\alpha \phi(x_k + \alpha p_{k-1})$$

  In nonlinear CG, we instead use a line search to choose the step size.

Like ordinary CG, nonlinear CG iterations can be preconditioned.

The many approaches to CG

The description I have given in these notes highlights (I hope) how orthogonality of the residuals and A-orthogonality of search directions follows naturally from the Galerkin condition, and how the rest of the CG iteration can be teased out of these orthogonality relations. However, this is far from the only way to "derive" the method of conjugate gradients. The discussion given by Demmel and by Saad (in *Iterative Methods for Sparse Linear Systems*) highlights the Lanczos connection, and uses this connection to show the existence of A-orthogonal search directions. Golub and Van Loan show the Lanczos connection, but also show how conjugate gradients can be derived as a general-purpose minimization scheme applied to the quadratic function $\phi(x)$. Trefethen and Bau give the iteration without derivation first, and then gradually explain some of its properties. If you find these discussions confusing, or simply wish to read something amusing, I recommend Shewchuk's "Introduction to the Conjugate Gradient Method Without the Agonizing Pain".

GMRES

The *generalized minimal residual* (GMRES) method of solving linear systems works with general systems of linear equations. Next to CG, it is probably the second-most popular of the Krylov subspace iterations.

The GMRES method is so named because it chooses the solution from a linear subspace that minimizes the (Euclidean) norm of the residual over successive Krylov subspaces. In terms of the Arnoldi decompositions

$$AQ_k = Q_{k+1}\bar{H}_k,$$

we have that $x_k = Q_k y_k$ where

$$y_k = \operatorname{argmin}_y \|\bar{H}_k y - \|b\| e_1\|^2.$$

One can solve the Hessenberg least squares problem in $O(k^2)$ time, but this is generally a non-issue. The true cost of GMRES is in saving the basis (which can use memory very quickly) and in keeping the basis orthogonal.

Unlike the CG method, alas, the GMRES method does not boil down to a short recurrence through a sequence of clever tricks. Consequently, we generally cannot afford to run the iteration for many steps before *restart*. We usually denote the iteration with periodic restarting every m steps as GMRES(m). That is, at each step we

1. Start with an initial guess $\hat{x}$ from previous steps.
2. Form the residual $r = b - A\hat{x}$.
3. Run m steps of GMRES to approximately solve $Az = r$.
4. Update $\hat{x} := \hat{x} + z$.

The GMRES iteration is generally used with a preconditioner. The common default is to use preconditioning on the left, i.e. solve

$$M^{-1}Ax = M^{-1}b;$$

in this setting, GMRES minimizes not the original residual, but the *preconditioned* residual. To the extent that the preconditioner reduces the condition number of the problem overall, the norm of the preconditioned residual tends to be a better indicator for forward error than the norm of the un-preconditioned residual. Of course, one can also perform preconditioning on the right (i.e. changing the unknown), or perform two-sided preconditioning.

The standard GMRES iteration (along with CG and almost every other Krylov subspace iteration) assumes a single, fixed preconditioner. But what if we want to try several preconditioners at once, or perhaps to use Gauss-Southwell or a chaotic relaxation method for preconditioning? Or perhaps we want to use a variable number of steps of some other iteration to precondition something like GMRES? For this purpose, it is useful to consider the *flexible GMRES* variant (FGMRES). Though it no longer technically is restricted to a Krylov subspace generated by a fixed matrix, the FGMRES iteration looks very similar to the standard GMRES iteration; we build an Arnoldi-like decomposition with the form

$$AZ_m = V_{m+1}\bar{H}_m$$

and then compute updates as a linear combination of the columns of Z_m by solving a least squares problem with $\bar{H}_m$. But here, each column of Z_m looks like $z_j = M_j^{-1}v_j$ where each M_j may be different.

Bi-Lanczos

So far, our focus has been on Krylov subspace methods that we can explain via the Lanczos or Arnoldi decompositions. The Lanczos-based CG has many attractive properties, but it only works with symmetric and positive definite matrices. One can apply CG to a system of normal equations — the so-called CGNE method — but this comes at the cost of squaring the condition number. There are also methods such as the LSQR iteration that implicitly work with the normal equations, but use an incrementally-computed version of the Golub-Kahan bi-diagonalization. The Arnoldi-based GMRES iteration works for more general classes of problems, and indeed it is the method of choice; but it comes at a stiff penalty in terms of orthogonalization costs.

Are there alternative methods that use short recurrences (like CG) but are appropriate for nonsymmetric matrices? There are several, though all have some drawbacks; the QMR and BiCG iterations may be the most popular. The key to the behavior of these methods comes from their use of a different decomposition, the *bi-orthogonal Lanczos factorization*:

$$\begin{aligned} AQ_j &= Q_j T_j + \beta_{j+1} q_{j+1} e_j^* \\ A^* P_j &= P_j T_j^* + \bar{\gamma}_{j+1} p_{j+1} e_j^* \\ P_j^* Q_j &= I. \end{aligned}$$

Here, the bases Q_j and P_j span Krylov subspaces generated by A and A^*, respectively (which means these algorithms require not only a function to apply A to a vector, but also a function to apply A^*). The bases are not orthonormal, and indeed may become rather ill-conditioned. They *do* have a mutual orthogonality relationship, though, namely $P_j^* Q_j = I$.

Details of the bi-orthogonal Lanczos factorization and related iterative algorithms can be found in the references. For the present, we satisfy ourseves with a few observations:

- The GMRES iteration shows monotonic reduction in the preconditioned residual, even with restarting. CG shows monotonic reduction in the error or residual when measured in appropriate norms. The methods based on bi-orthogonal Lanczos, however, can show rather erratic convergence; errors decay in general, but they may exhibit intermediate local increases. BiCG is generally more erratic than QMR.
- Even in exact arithmetic, the subspace bases formed by bi-Lanczos may become rather ill-conditioned.
- The bi-orthogonal iterations sometimes show *breakdown* behavior where the local approximation problem becomes singular. This can be overcome using *lookahead* techniques, though it complicates the algorithm.

The relative simplicity of GMRES — both in theory and in implementation — perhaps explains its relative popularity. Nonetheless, these other methods are worth knowing about.

Extrapolation and mixing

When we discussed CG, we also briefly discussed *nonlinear* CG methods (e.g. Fletcher-Reeves). One can similarly extend Krylov subspace ideas to accelerate nonlinear equation solving methods; that is, given a fixed point iteration

$$x^{(k+1)} = G(x^{(k)}),$$

we can accelerate the computation of the fixed point by taking an appropriate linear combination of the iterates $x^{(k)}$. This is a powerful idea; indeed, it is so powerful that it can be used to compute repulsive fixed points where the usual iteration would diverge! The techniques usually go under the heading of *extrapolation methods* (including Reduced Rank Extrapolation or RRE, Minimal Polynomial Extrapolation or MPE, and Vector Padé Extrapolation); and *acceleration* or *mixing* techniques, the most popular of which is the *Anderson acceleration* method. Applied to the iterates of a stationary linear system solver, these techniques are all formally equivalent to Krylov subspace solvers. In particular, RRE is equivalent to GMRES (in exact arithmetic).

The idea behind extrapolation methods is to exploit systematic patterns in the convergence of fixed point iteration. For example, suppose the error iteration gave us (approximately)

$$e^{(k)} = \sum_{j=1}^{m} v^{(j)} \alpha_j^k$$

where the vectors $v^{(j)}$ and the exponents α_j were unknown. The hope is that we can learn the parameters of the error iteration by fitting a model to the update sequence:

$$u^{(k)} = x^{(k+1)} - x^{(k)} = e^{(k+1)} - e^{(k)} = \sum_{j=1}^{m} (\alpha_j - 1) v^{(j)} \alpha_j^k.$$

If $p(z) = c_0 + c_1 z + ... + c_m z^m$ is a polynomial such that $p(\alpha_j) = 0$ for each α_j, then we should satisfy

$$\sum_{j=1}^{m} c_j u^{(k+j)} = 0.$$

If we look at enough update steps, we can determine both the coefficient vectors and the exponents.

With an appropriate model, extrapolation methods can produce rather astonishing results. Of course, extrapolation methods are subject to issues of overfitting, and (particularly when the convergence is irregular) may produce results that are wildly incorrect.

Communication-Avoiding (CA) Krylov

In highly parallel computing systems, the cost of computing with Krylov subspaces may be dominated not by the matrix-vector products, but by the cost of computing dot products for the purpose of orthogonalization. Repeatedly applying matrix-vector products may involves rather local communication patterns, but dot products involve a global communication. Of course, we could (in principle) form a power basis for the Krylov subspace; but this basis is typically too ill-conditioned for serious work. So what is one to do?

The *communication-avoiding* Krylov methods use the power of polynomials to thread between the Scylla of synchronization costs and the Charybdis of catastrophic ill-conditioning. In general, we write Krylov subspace bases as

$$\mathcal{K}_k(A, b) = \operatorname{span}\{p_j(A)b\}_{j=0}^{(k-1)}.$$

where $p_j(z)$ is a degree j polynomial. In the case of the power basis, $p_j(z) = z^j$; and in the case of the Lanczos or Arnoldi bases, $p_j(z)$ is chosen fully adaptively. The communication avoiding approach is to choose $p_j(z)$ in advance, but using information about the spectra to ensure that the vectors $p_j(A)b$ are not too nearly co-linear.

As with some of the other topics in this section, the big idea behind communication-avoiding Krylov methods is simple, but there are too many details to give a full treatment in the time we have allocated. For those interested in such details, I recommend the 2010 Ph.D. thesis of Mark Hoemmen.