MA3227 Numerical Analysis II

Lecture 11: Multigrid Method

Simon Etter

2019/2020

Multigrid Method

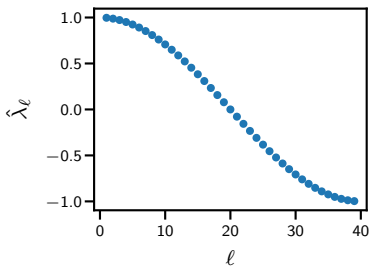**Recap: Jacobi iteration**

Jacobi iteration is given by $\quad x_k = D^{-1}\Big(b - (A - D)\,x_{k-1}\Big).$

The Jacobi iterates x_k satisfy the error recursion

$$x_k - x = R^k\,(x_0 - x) \qquad \text{where} \qquad R = -D^{-1}\,(A - D).$$

Throughout this lecture, we will assume $A = -\Delta_n^{(1)}$.
The eigenvalues of R are then given by $\hat{\lambda}_\ell = \cos\big(\pi\frac{\ell}{n+1}\big)$.

Observation:

Only error components associated with $\ell \approx 1$ and $\ell \approx n$ converge slowly.
Error components associated with $\ell \approx \frac{n}{2}$ converge very fast.

Multigrid Method

Relaxed Jacobi iteration

We can tweak which error components converge rapidly by switching to the relaxed Jacobi iteration

$$x_k = (1 - \theta) \, x_{k-1} + \theta \, D^{-1} \left(b - (A - D) \, x_{k-1} \right).$$

This method is called "relaxed" (or sometimes also "damped") because we only take a factor θ of the step $x_{k-1} \to x_k$ proposed by Jacobi.

The error recursion formula for relaxed Jacobi is

$$x_k - x = R_\theta^k \, (x_0 - x) \qquad \text{with} \qquad R_\theta = (1 - \theta) \, I - \theta \, D^{-1} \, (A - D),$$

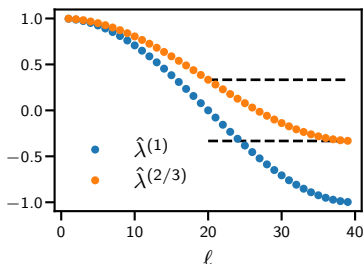and the eigenvalues of R_θ for $A = -\Delta_n^{(1)}$ are

$$\hat{\lambda}_\ell^{(\theta)} = (1 - \theta) + \theta \, \hat{\lambda}_\ell^{(1)} = (1 - \theta) + \theta \cos \left(\pi \tfrac{\ell}{n+1} \right).$$

Multigrid Method

Relaxed Jacobi iteration (continued)

Observation:

▶ Error components with large ℓ are damped rapidly for an appropriately chosen θ. (Precisely, we have $|\hat{\lambda}_\ell^{(2/3)}| \leq \frac{1}{3}$ for all $\ell \geq \frac{n}{2}$.)

▶ Error components with small ℓ have a poor damping factor $\hat{\lambda}^{(\theta)}$ regardless of how we choose θ.

Multigrid Method

Relaxed Jacobi iteration (continued)

Conclusions:

- ▶ Relaxed Jacobi can easily eliminate the large-ℓ error components.
- ▶ To make relaxed Jacobi fast, we need to combine it with another method which eliminates the small-ℓ error components.

The ideal low-ℓ eliminator

Warning: This operation is called "coarse-grid correction" in the literature.

Let $A = V \Lambda V^T$ be the eigendecomposition of $A = -\Delta_n^{(1)}$.

Let us introduce $P = V[:, 1 : n/2]$ (arbitrary rounding for odd n) and set

$$x_1 = x_0 + P \, (P^T A P)^{-1} \, P^T (b - A x_0)$$

where x_0 is an arbitrary initial guess.

Claim: The update $x_0 \rightarrow x_1$ eliminates the low-ℓ error, i.e. we have

$$P^T (x_1 - x) = 0.$$

Proof. See next slide.

Multigrid Method

Proof (continued). Inserting the definition of x_1 yields

$$P^T(x_1 - x) = P^T(x_0 - x) + \underbrace{P^T P}_{I}(P^T AP)^{-1} P^T \underbrace{(b - Ax_0)}_{A(x - x_0)}$$

$$= P^T(x_0 - x) - (P^T AP)^{-1} P^T A (x_0 - x).$$

Since P is a matrix of eigenvectors, we have $P^T A = P^T APP^T$ and thus

$$P^T(x_1 - x) = P^T(x_0 - x) - (P^T AP)^{-1} (P^T AP) P^T (x_0 - x)$$

$$= P^T(x_0 - x) - P^T (x_0 - x)$$

$$= 0.$$

Multigrid Method

Discussion

The ideal low-ℓ eliminator is unpractical for two reasons.

▶ Assembling P requires us to compute eigenvectors which is expensive.

▶ The update $x_0 \to x_1$ requires us to solve $(P^T A P)^{-1} v$. This is almost always a dense linear system which is expensive to solve.

These issues can be avoided by observing that the low-ℓ eliminator only exploits that $\text{range}(P) = \{u_\ell \mid \ell \leq \frac{n}{2}\}$: if we have another matrix $\tilde{P}$ such that $P = \tilde{P} T$ for some invertible T, we can equivalently write

$$x_1 = x_0 + \tilde{P}(\tilde{P}^T A \tilde{P})^{-1} \tilde{P}^T (b - A x_0)$$

since replacing P with $\tilde{P} T$ only introduces two copies of $T T^{-1} = I$.

We conclude that ideally we would like to have a $P \in \mathbb{R}^{n \times (n/2)}$ such that
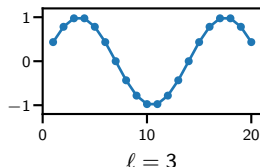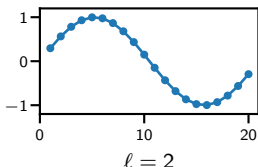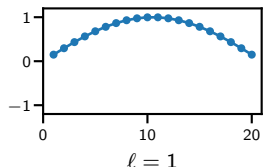
$$\text{range}(P) \approx \text{span}\{u_\ell \mid \ell \leq \frac{n}{2}\} \qquad \text{and} \qquad P^T A P \text{ is sparse.}$$

Note that we relax the subspace condition somewhat to give us more flexibility to achieve the sparsity condition.

Multigrid Method

Subspace condition for P

Let us take a look at the eigenvectors $u_\ell[i] = \sin\left(\pi \frac{\ell i}{n+1}\right)$ of $\Delta_n^{(1)}$ to get an idea for how to choose a P such that $\text{range}(P) \approx \text{span}\{u_\ell \mid \ell \leq \frac{n}{2}\}$.

Observations:

- ℓ indicates the *frequency* (number of oscillations) of u_ℓ.
- u_ℓ with small ℓ are *smooth*: $u_\ell[i] \approx \frac{u_\ell[i+1] + u_\ell[i-1]}{2}$.
  Smooth is only a qualitative term. Do not interpret this formula as a definition.

We conclude that columns of P should be smooth to satisfy the subspace condition.
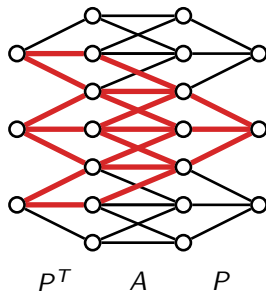
Multigrid Method

Sparsity condition for P

$A = -\Delta_n^{(1)}$ is a tridiagonal matrix. If we choose $P \in \mathbb{R}^{n \times (n/2)}$ such that

$$P[i,j] \neq 0 \iff |i - 2j| \leq 1,$$

then $P^T A P$ is tridiagonal as well.
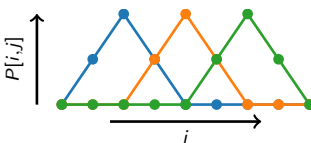
Proof.

$$P^T \qquad A \qquad P$$

The three columns of edges represents the sparsity pattern of $B \in \{P^T, A, P\}$.
The left vertices represent row indices i, the right vertices represent column indices j,
and the edges represent nonzero entries $B[i,j]$.

Multigrid Method

Choosing P

The observations on the last two slides combined suggest that

$$P = \begin{pmatrix} \frac{1}{2} & & \\ 1 & & \\ \frac{1}{2} & \frac{1}{2} & \\ & 1 & \\ & \frac{1}{2} & \frac{1}{2} \\ & & 1 \\ & & \frac{1}{2} \end{pmatrix} \qquad \longleftrightarrow$$

should be a good choice to obtain an approximate low-ℓ eliminator

$$x_1 = x_0 + P\,(P^T A P)^{-1}\,P^T(b - A x_0).$$

This P has the required sparsity pattern, and the columns are reasonably smooth given the sparsity constraint.

Multigrid Method

Coarse-grid correction

We can verify numerically for the above choice of P, we have

$$P^T \Delta_{2n+1}^{(1)} P \propto \Delta_n^{(1)}.$$

The linear system $(P^T A P)^{-1} v$ in the low-ℓ elimination step

$$x_1 = x_0 + P (P^T A P)^{-1} P^T (b - A x_0)$$

is therefore of the same type as the original linear system $A^{-1} v$, but on a grid with fewer grid points.

For this reason, "low-ℓ elimination" is called "coarse-grid correction" in the literature.

Multigrid Method

Aside: iterative methods and approximate inverses

There is a common idea underlying both restarted GMRES, Jacobi iteration and coarse-grid correction.

All three methods are based on an approximate inverse $B \approx A^{-1}$:

▶ Restarted GMRES: $B =$ prematurely terminated GMRES iteration.

▶ Jacobi iteration: $B = D^{-1}$ (this will make sense in hindsight).

▶ Coarse-grid correction: $B = P(P^T A P)^{-1} P^T$.

In all three methods, B is too inaccurate for $\tilde{x} = Bb$ to be an acceptable approximation to $x = A^{-1}b$.

Instead, we define the iteration

$$x_{k+1} = x_k + B(b - Ax_k)$$

which yields the error recursion

$$x_{k+1} - x = (I - BA)(x_k - x).$$

Multigrid Method

Remark: iterative methods and approximate inverses (continued)

The error recursion shows that if B was the exact inverse A^{-1}, we would get the exact solution $x_1 = x$ after a single step for any initial guess x_0. Moreover, we have $x_k \to x$ even if $B \neq A^{-1}$ as long as the largest eigenvalue of $I - BA$ has magnitude < 1.

Proof that Jacobi fits into this framework

We defined the Jacobi iteration as $x_{k+1} = D^{-1}\big(b - (A - D)\,x_k\big)$.

This formula is inspired by the idea that Jacobi solves each equation for the diagonal, i.e. Jacobi writes $Ax = b$ as $Dx + (A - D)\,x = b$ and then replaces $Dx \to D\,x_{k+1}$ and $(A - D)\,x \to (A - D)\,x_k$.

To fit Jacobi into the above framework, we rewrite

$$x_{k+1} = D^{-1}\big(b - (A - D)\,x_k\big) = x_k + D^{-1}(b - Ax_k).$$

This shows that Jacobi implicitly treats D^{-1} as an approximation to A^{-1}. For this reason, $P = D$ is called the *Jacobi preconditioner*.

Similarly, $P = D + U$ is called the *Gauss-Seidel preconditioner*.

Multigrid Method

Two-grid iteration

Combining the above tools leads us to the following algorithm.

1. Run a few steps of the relaxed Jacobi iteration

$$x_{k+1} = x_k + \tfrac{2}{3} D^{-1}(b - Ax_k)$$

   to eliminate the high-frequency (large ℓ) errors.

2. Perform a coarse-grid correction

$$x_1 = x_0 + P(P^T A P)^{-1} P^T (b - Ax_0)$$

   to eliminate the low-frequency (small ℓ) errors.

3. Repeat from step 1 until convergence.

   Repetition is needed because we do not use the ideal P in step 2.

This algorithm switches back and forth between a grid with $2n + 1$ points for the Jacobi iteration and another grid with n points for the coarse-grid correction. For this reason, it is called "two-grid iteration".

See twogrid_step().

Multigrid Method

Discussion

We observe numerically that relaxed Jacobi combined with coarse-grid correction leads to *grid-independent convergence*:
the convergence plot looks virtually the same for different grid sizes n.

This is an important improvement compared to Jacobi iteration where the convergence plot becomes flatter as we increase n.

However, two-grid iteration is still not a $\mathcal{O}(N)$ algorithm because we rely on \ to solve the coarse-grid linear system $(P^T A P)^{-1} v$.

To solve this issue, recall that $P^T \Delta_n^{(1)} P \propto \Delta_n^{(1)}$, i.e. the linear system to solve in the coarse-grid correction is again Poisson's equation but on a grid with fewer grid points.

Thus, two-grid iteration essentially reduces the problem of solving Poisson's equation on a $2n + 1$ grid to that of solving Poisson's equation on an n-point grid.

Idea: use two-grid recursively until we end up with a problem which is small enough that we can afford to use \.

The resulting algorithm is called "multigrid iteration" and implemented in multigrid_step().

Multigrid Method

Discussion (continued)

Multigrid looks like it could be the desired $\mathcal{O}(N)$ algorithm.
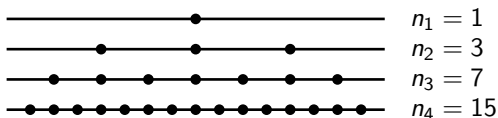
To be sure, we need to check:

- Each iteration requires only $\mathcal{O}(N)$ operations.
- We have $\|x_k - x\| \leq C \rho^k$ for some $\rho \neq \rho(n) < 1$.

Multigrid Method

Runtime and memory of a single multigrid iteration

Observations:

- ▶ Multigrid runs on a stack of L grids, and the ℓth grid in this stack has $n_\ell = 2^\ell - 1$ grid points.

$$n_1 = 1$$
$$n_2 = 3$$
$$n_3 = 7$$
$$n_4 = 15$$

- ▶ The Jacobi iterations and coarse-grid correction on the ℓth grid require $\mathcal{O}(n_\ell)$ runtime.

The overall runtime and memory of a multigrid iteration is therefore proportional to the total number of grid points in the stack of grids. This number is

$$n_{\text{total}} = \sum_{\ell=1}^{L}(2^\ell - 1) = \frac{2^{L+1} - 1}{2 - 1} - L = \mathcal{O}(2^L) = \mathcal{O}(n_L).$$

Multigrid Method

Runtime and memory of a single multigrid iteration (continued)

Conclusion: runtime and memory of a single multigrid iteration are proportional to the number of grid points n_L on the finest grid.

n_L is the number of unknowns N in the original linear system $Ax = b$.

Thus, each multigrid iteration indeed requires only $\mathcal{O}(N)$ runtime and memory.

Multigrid Method

Convergence analysis for two-grid

Recall: the relaxed Jacobi iteration

$$x_{k+1} = x_k + \tfrac{2}{3} \, , D^{-1} \left(b - A x_k \right)$$

leads to the error recursion

$$x_{k+1} - x = R \left(x_k - x \right) \qquad \text{where} \qquad R = I - \tfrac{2}{3} D^{-1} A.$$

Similarly, we obtain that the coarse-grid correction

$$x_1 = x_0 + P \left(P^T A P \right)^{-1} P^T (b - A x_0)$$

leads to the error recursion

$$x_1 - x = S \left(x_0 - x \right) \qquad \text{where} \qquad S = I - P \left(P^T A P \right)^{-1} P^T A.$$

Multigrid Method

Convergence analysis for two-grid (continued)

Combining the above results leads to an error recursion for the two-grid iteration:

$$x_{k+1} - x = SR(x_k - x).$$

Note that x_k denotes different vectors depending on whether we are talking about Jacobi, coarse-grid correction or two-grid iteration.

Convergence analysis for the two-grid iteration therefore boils down to determining the largest eigenvalue of SR.

We skip this step because the computations are fairly tedious. The upshot is that all eigenvectors of SR are indeed upper-bounded by some constant $\rho < 1$ which does not depend on k.

This shows that the two-grid iteration indeed converges exponentially with a fixed rate ρ regardless of the grid size n.

Multigrid Method

Convergence analysis for multigrid

Proving a grid-independent convergence result for two-grid is tedious but in principle straightforward.

The same does no longer hold true for multigrid: S has now a recursive structure and we would like a result which holds for any arbitrary recursion depth. It is therefore no longer obvious how to compute or bound the eigenvalues of SR.

Instead, we will content ourselves with the empirical observation that the convergence of multigrid is usually comparable to that of the associated two-grid iteration.

Conclusion

We have seen:

- Each multigrid iteration requires $\mathcal{O}(N)$ runtime and memory.
- We only need a constant number of iterations to achieve a fixed accuracy, regardless of n.

Combined, these two properties imply that multigrid is the $\mathcal{O}(N)$ solver for Poisson's equation that we have been chasing for the past four weeks.

Multigrid Method

Why three different algorithms for solving linear systems?

Short answer: it is a trade-off between generality and efficiency.

A bit more detail:

- ▶ LU can reliably solve any linear system, but it does not scale well to large systems.

- ▶ Krylov methods can in principle solve any linear system.
  However, we usually need an application-specific preconditioner to get acceptable performance.

- ▶ Multigrid is very fast but also requires a lot of additional information, e.g. how to choose the θ in relaxed Jacobi and how to construct P. Bad choices for these parameters may lead to divergence.

In practice, which method to choose depends on your requirements.

- ▶ Are you problems large enough that you need ultimate efficiency?

- ▶ How much time are you willing to invest to get your code to work?

- ▶ How important is reliability for your application?

Multigrid Method

Linear solvers and speed of propagation

The following presents a high-level argument why unpreconditioned Krylov methods and Jacobi iteration must necessarily be slow, and why multigrid can be fast.

Recall: $\Delta_n^{(d)}$ is a sparse matrix, but $\left(\Delta_n^{(d)}\right)^{-1}$ is dense.

This means that every entry $x[i]$ of the solution $x = -(\Delta_n^{(d)})^{-1}b$ depends on every entry $b[j]$ of the right-hand side.

Both Jacobi and unpreconditioned Krylov methods try to approximate x using a linear combination of $b, Ab, \ldots, A^{k-1}b$.

Jacobi also introduces some powers of D^{-1}, but this is irrelevant for our argument.

For any fixed k, $x_k[i]$ therefore depends only on entries $b[j]$ which are at most a distance k from vertex i in the graph of A.

This explains we cannot get convergence for $k \ll n$.

Also, it implies that P can only be a good preconditioner if P^{-1} is dense. However, denseness alone is not sufficient: the ILU preconditioner $P = \tilde{L}\tilde{U}$ has a dense inverse P^{-1}, but it still does not lead to convergence for $k \ll n$.
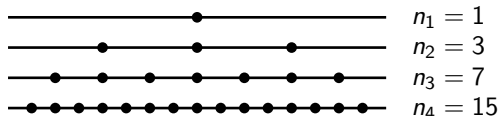
Multigrid Method

Linear solvers and speed of propagation (continued)

The above discussion indicates that a fast solver for sparse linear system must strike a balance between rapidly propagating the information in *b* and limiting the runtime for each propagation step.

In this sense, LU and Jacobi / Krylov methods represent two extremes:

- ▶ Sparse LU factorisation exchanges information globally.
  This leads to "convergence" in a single step, but the runtime per propagation step is larger than $\mathcal{O}(N)$ in general.

- ▶ Jacobi and Krylov methods exchange information locally.
  This leads to $\mathcal{O}(N)$ cost per iteration but requires at least $\mathcal{O}(n)$ iterations for convergence.

The power of multigrid is that it propagates different pieces of information at different speeds. As we move up in the stack of grids, the information content reduces but the speed of propagation increases.

$n_1 = 1$
$n_2 = 3$
$n_3 = 7$
$n_4 = 15$

Multigrid Method

Summary

▶ Multigrid interleaves relaxed Jacobi steps

$$x_{k+1} = x_k + \tfrac{2}{3} D^{-1} (b - Ax_k)$$

with coarse-grid corrections

$$x_1 = x_0 + P (P^T A P)^{-1} P^T (b - Ax_0).$$

Jacobi can also be replaced by other methods (e.g. Gauss-Seidel). Any such method is called a smoother because it eliminates the high-frequency components of the error.

▶ Each iteration requires $\mathcal{O}(N)$ operations, and the convergence rate is independent of n.