Parallelization of the Iterative Procedure

For Jacobi, parallelization is utterly trivial:

1. Split up the unknowns onto processors.
2. Each processor updates all of its unknowns.
3. Each processor sends its unknowns to processors that need the updated information.
4. Continue iterating until done.

Common fallacies:

- When an element of the solution vector x_k has a small enough element-wise residual, stop updating the element. This leads to utterly wrong solutions since the residuals are affected by updates of neighbors after the element stops being updated.
- Keep computing and use the last known update from neighboring processors. This leads to chattering and no element-wise convergence.
- Asynchronous algorithms exist, but eliminate the chattering through extra calculations.

Parallel Gauss-Seidel and SOR are much, much harder. In fact, by and large, they do not exist. Googling efforts leads to an interesting set of papers that approximately parallelize Gauss-Seidel for a set of matrices with a very well known structures only. Even then, the algorithms are extremely complex.

Parallel Block-Jacobi is commonly used instead as an approximation. The matrix A is divided up into a number of blocks. Each block is assigned to a processor. Inside of each block, Jacobi is performed some number of iterations. Data is exchanged between processors and the iteration continues.

See the book (*absolutely shameless plug*),

> C. C. Douglas, G. Haase, and U. Langer, A Tutorial on Elliptic PDE Solvers and Their Parallelization, SIAM Books, Philadelphia, 2003.

for how to do parallelization of iterative methods for matrices that commonly occur when solving partial differential equations (*what else would you ever want to solve anyway???*).

3b (ii) Krylov Space Methods

Conjugate Gradients

Let A be symmetric, positive definite, i.e.,

$$A=A^T \text{ and } (Ax,x)\geq \rho\|x\|_2, \text{ where } \rho>0.$$

The *conjugate gradient* iteration method for the solution of $Ax+b=0$ is defined as follows with $r=r(x)=Ax+b$:

x_0 arbitrary	(approximate solution)
$r_0=Ax_0+b$	(approximate residual)
$w_0=r_0$	(search direction)

For $k=0,1,\cdots$

$$x_{k+1}=x_k+\alpha_k w_k, \qquad \alpha_k=-\frac{(r_k,w_k)}{(w_k,Aw_k)}$$

$$r_{k+1}=r_k+\alpha_k Aw_k$$

$$w_{k+1}=r_{k+1}+\beta_k w_k, \qquad \beta_k=-\frac{(r_{k+1},Aw_k)}{(w_k,Aw_k)}$$

Lemma CG1: If $Q(x(t))=\frac{1}{2}(x(t),Ax(t))+(b,x(t))$ and $x(t)=x_k+tw_k$, then α_k is chosen to minimize $Q(x(t))$ as a function of t.

Proof:

$$\begin{aligned} \text{Q(x(t))} &= \frac{1}{2}(x_k+tw_k,Ax_k+tAw_k)+(b,x_k+tw_k) \\ &= \frac{1}{2}\left\{(x_k,Ax_k)+2t(x_k,Aw_k)+t^2(w_k,Aw_k)\right\}+(b,x_k)+t(b,w_k) \end{aligned}$$

$$\frac{d}{dt}\mathrm{Q(x(t))} = (x_k, Aw_k) + t(w_k, Aw_k) + (b, w_k)$$

$$\begin{aligned}\frac{d}{dt}\mathrm{Q(x(\alpha_k))} &= (x_k, Aw_k) + \alpha_k(w_k, Aw_k) + (b, w_k) \\ &= (x_k, Aw_k) - (r_k, w_k) + (b, w_k) \\ &= (Ax_k + b - r_k, w_k) \\ &= 0\end{aligned}$$

since

$$\begin{aligned}Ax_k - r_k &= A(x_{k-1} + \alpha_{k-1} w_{k-1}) - (r_{k-1} + \alpha_{k-1} Aw_{k-1}) \\ &= Ax_{k-1} - r_{k-1} \\ &= \vdots \\ &= Ax_0 - r_0 \\ &= -b\end{aligned}$$

Lemma CG2: The parameter β_k is chosen so that $(w_{k+1}, Aw_k) = 0$.

Lemma CG3: For $0 \le q \le k$,

1. $(r_{k+1}, w_q) = 0$
2. $(r_{k+1}, r_q) = 0$
3. $(w_{k+1}, Aw_q) = 0$

Lemma CG4: $\alpha_k = \dfrac{(r_{k+1}, r_{k+1})}{(r_{k+1}, Ar_k)}$.

Lemma CG5: $\beta_k = \dfrac{(r_{k+1}, r_{k+1})}{(r_k, r_k)}$

Theorem 3.9 (CG): Let $A \in \mathbb{R}^{N \times N}$ be symmetric, positive definite. The the CG iteration converges to the exact solution of $Ax+b=0$ in not more than N iterations.

Preconditioning

We seek a matrix M (or a set of matrices) to use in solving $M^{-1}Ax = M^{-1}b$ such that

- $\kappa(M^{-1}A) \ll \kappa(A)$
- M is easy to use when solving $Mx=b$.
- M and A have similar properties (e.g., symmetry and definiteness)

Reducing the condition number reduces the number of iterations necessary to achieve an adequate convergence factor.

Thereom 3.9: In finite arithmetic, the preconditioned conjugate gradient method converges at the rate based on the largest and smallest eigenvalues of $M^{-1}A$,

$$\frac{\|x - x_k\|_2}{\|x - x_0\|_2} \le 2\sqrt{\kappa_2(M^{-1}A)}\left(\frac{\sqrt{\kappa_2(M^{-1}A)}-1}{\sqrt{\kappa_2(M^{-1}A)}+1}\right)^k, \text{ where } \kappa_2(M^{-1}A) = \frac{\lambda_{\max}}{\lambda_{\min}}.$$

What are some common preconditioners?

- Identity!!! ☺
- Main diagonal (the easiest to implement in parallel and very hard to beat)
- Jacobi
- Gauss-Seidel
- Tchebyshev
- Incomplete LU, known as ILU (or modified ILU)

Most of these do not work straight out of the box since symmetry may be required. How do we symmetrize Jacobi or a SOR-like iteration?

- Do two iterations: once in the order specified and once in the opposite order. So, if the order is *natural*, i.e., *1–N*, then the opposite is *N–1*.
- There are a few papers that show how to do two way iterations for less than the cost of two matrix-vector multiplies (which is the effective cost of the solves).

Preconditioned conjugate gradients

x_0 arbitrary (approximate solution)
$r_0 = Ax_0 + b$ (approximate residual)
$M\tilde{r}_0 = r_0$
$w_0 = \tilde{r}_0$ (search direction)

followed by for $k = 0,1,\cdots$ until $\left\|(\tilde{r}_{k+1}, r_{k+1})\right\| \le \varepsilon \left\|(\tilde{r}_0, r_0)\right\|$ and $\left\|(r_{k+1}, r_{k+1})\right\| \le \varepsilon \left\|(r_0, r_0)\right\|$

for a given ε:

$$x_{k+1} = x_k + \alpha_k w_k, \qquad \alpha_k = -\frac{(\tilde{r}_k, r_k)}{(w_k, Aw_k)}$$

$$r_{k+1} = r_k + \alpha_k A w_k$$

$$M\tilde{r}_{k+1} = r_{k+1}$$

$$w_{k+1} = \tilde{r}_{k+1} + \beta_k w_k, \qquad \beta_k = -\frac{(\tilde{r}_{k+1}, r_k)}{(\tilde{r}_k, r_k)}$$