Iterative Methods

With direct methods discussed:

- The answer is given after a finite number of operations but – Partial results are not available, i.e., you do all of the work and you get the answer. It is in general not possible to do part of the work and get an approximation to the solution.

- It may not be possible to reconcile the demands of exploiting structure to maintain low complexity and of maintaining accuracy.

An alternative to the direct methods seen so far which can often address the two issues above are iterative methods.
Iterative methods:

- They produce a sequence of approximations to the solution which under certain conditions converge
  \[ x_k \to x = A^{-1}b \]

- In general, a finite number of iterations and therefore operations cannot be guaranteed.

- Note however that some of the methods discussed in the following are, in fact, direct methods but with a form that allows them to have the advantages of an iterative method.

- The operations within an iteration exploit the structure to keep the work per iteration low.

- The basic problem is to balance the complexity of the iteration with the number of iterations until acceptable convergence is achieved (if at all).
• Typically applied to sparse, structured dense (e.g. Toeplitz), or nearly structured dense (discretized integral operators).

• They can be used to improve the accuracy of a direct method solution.

• They can be used to generate a solution when only low or moderate accuracy is required.

• Unlike direct methods, subsequent solutions with different right hand sides are not necessarily cheaper than the initial solution.
Symmetric Positive Definite Systems

Solving $Ax = b$ where $A$ is symmetric positive definite is equivalent to minimizing a quadratic functional:

$$
\phi(x) = (x - A^{-1}b)^T A(x - A^{-1}b)
$$

or sometimes in the literature

$$
\gamma(x) = \frac{1}{2} x^T A x - x^T b
$$

Both have a unique global minimum at $x = A^{-1}b$, the vector in which we are interested.
Conjugate Gradient

\[ x_0 \text{ arbitrary}; \ r_0 = b - Ax_0; \ p_0 = r_0 \]

\[
k = 0, 1, \cdots
\]

\[
v_k = Ap_k
\]

\[
\alpha_k = r_k^T r_k / p_k^T v_k
\]

\[
x_{k+1} = x_k + \alpha_k p_k
\]

\[
r_{k+1} = r_k - \alpha_k v_k
\]

\[
\beta_k = r_{k+1}^T r_{k+1} / r_k^T r_k
\]

\[
p_{k+1} = r_{k+1} + \beta_k p_k
\]

end

• solution in no more than \( n \) steps (it is a direct method)

• if \( A \) has \( m \) distinct eigenvalues then CG terminates in no more than \( m \) steps
\[ \|e_k\|_A \leq 2 \left( \frac{\sqrt{\lambda_{\text{max}}}-\sqrt{\lambda_{\text{min}}}}{\sqrt{\lambda_{\text{max}}}+\sqrt{\lambda_{\text{min}}}} \right)^k \|e_0\|_A \]

- attempts have been made to reduce number of dotproducts or rearrange them to reduce the number of barriers due to fan-in (Gannon and Van Rosendale 1983, Leland 1990, Eijkhout 1993 – 1995) – stability of the iteration tends to be the problem
Preconditioning

While finite termination and the distinct eigenvalue convergence theorems are nice they typically do not yield satisfactory convergence in practice. In most cases it is necessary to alter the system in order to improve the convergence rate. This is done by driving the coefficient matrix to have, effectively, fewer distinct eigenvalues. Ideally, the matrix can be transformed so that the coefficient matrix is the identity and therefore the solution is known trivially. Unfortunately, there is a tradeoff in the cost of transforming the system – or preconditioning the system – and the resulting improvement in performance.

Find a matrix $M = C^2$ which is symmetric positive definite, approximates $A$, and for which $Mu = y$ is easy to solve.

Apply CG to

$$C^{-1}AC^{-1}Cx = C^{-1}b$$

After some tedious manipulation CG applied to the system above can be written in terms of a simple modification of the earlier form.
Conjugate Gradient

$x_0$ arbitrary; $r_0 = b - Ax_0$;
solve $Mz_0 = r_0$; $p_0 = z_0$

$k = 0, 1, \ldots$

$v_k = Ap_k$

$\alpha_k = r_k^T z_k / p_k^T v_k$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k v_k$

solve $Mz_{k+1} = r_{k+1}$

$\beta_k = r_{k+1}^T z_{k+1} / r_k^T z_k$

$p_{k+1} = z_{k+1} + \beta_k p_k$

end

- each iteration requires the solution of the preconditioning equation

- this must be kept low in operations (or highly parallel) in order to maintain simplicity of each iteration which tends to contradict the requirement that $M$ approximates $A$ well.
Diagonal preconditioning

Choose $M$ to be the diagonal elements of $A$. By definition $M$ is positive definite. (In practice, the block diagonal portion of $A$ is often used.)

In some cases, this can significantly improve convergence. It is highly parallel and low in cost in terms of operations.

\[
A = \begin{pmatrix}
1000 & -1 & 0 \\
-1 & 100 & -1 \\
0 & -1 & 10
\end{pmatrix}
\]

\[
C^{-1}AC^{-1} = \begin{pmatrix}
1 & -0.003 & 0 \\
-0.003 & 1 & -0.0032 \\
0 & -0.032 & 1
\end{pmatrix}
\]

Eigenvalues of $A$ are (approximately) $(1000, 100.01, 9.99)$, $\rho \approx 100.1$

Eigenvalues of $C^{-1}AC^{-1}$ are (approximately) $(100, 0.97, 1.03)$, $\rho \approx 1.06$

Unfortunately, diagonal (or even block diagonal) preconditioning is not good enough.
Approximate Factorization Preconditioning

- Incomplete Cholesky factorization
  - perform Cholesky but drop elements that are not in pre-specified positions in the (approximate) Cholesky factor.
  - drop elements that are at a level of fill beyond some specified level, e.g., fill-in created by operations involving all original elements is fill level 1, fill-in created involving elements of up to fill level $k - 1$ are fill level $k$.
  - drop elements whose magnitude relative to some reference magnitude (diagonal element, row sum, etc.)

- Existence and effective choice of parameters are problems here.

- Approximate LU is also possible

- Can be used in a hybrid with direct methods without dropping to guarantee a solution (see work of Gallivan, Zlatev, Sameh)
Iterative methods are also becoming increasingly important for structured dense systems such as those with coefficient matrices that are Toeplitz or have low displacement rank. The use of iterative methods typically implies the lack of a fast direct method that takes into account the structure of the matrix or the availability of a good preconditioner due to application attributes so that a small number of iterations can be expected.

Consider the following Toeplitz matrix

\[
\begin{pmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\
\alpha_4 & \alpha_1 & \alpha_2 & \alpha_3 \\
\alpha_3 & \alpha_4 & \alpha_1 & \alpha_2 \\
\alpha_2 & \alpha_3 & \alpha_4 & \alpha_1
\end{pmatrix}
\]

The matrix is also called *circulant* since the rows wrap around.

For any circulant matrix \( C \) we have

\[
C = F^* \Lambda F
\]

where \( F \) is a unitary matrix whose application is equivalent to the computation of the Fourier transform of a vector and \( \Lambda \) is a diagonal matrix. Therefore,

\[
z = Cv \\
x = C^{-1}b
\]

can both be computed using FFTs in \( O(n \log n) \) operations and have reasonable parallel versions.
Now consider the following Toeplitz matrix and an associated circulant matrix.

\[
T = \begin{pmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\
\alpha_5 & \alpha_1 & \alpha_2 & \alpha_3 \\
\alpha_6 & \alpha_5 & \alpha_1 & \alpha_2 \\
\alpha_7 & \alpha_6 & \alpha_5 & \alpha_1 \\
\end{pmatrix}
\]

\[
C = \begin{pmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_7 & \alpha_6 & \alpha_5 \\
\alpha_5 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_7 & \alpha_6 \\
\alpha_6 & \alpha_5 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_7 \\
\alpha_7 & \alpha_6 & \alpha_5 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\
\alpha_4 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_1 & \alpha_2 & \alpha_3 \\
\alpha_3 & \alpha_4 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_1 & \alpha_2 \\
\alpha_2 & \alpha_3 & \alpha_4 & \alpha_7 & \alpha_6 & \alpha_5 & \alpha_1 \\
\end{pmatrix}
\]

\[
Cv = \begin{pmatrix} T & A \\ B & E \end{pmatrix} \begin{pmatrix} x \\ 0 \end{pmatrix}
\]

\[
= \begin{pmatrix} Tx \\ Bx \end{pmatrix}
\]

So \(Tx\) can be found by using FFT's of approximately length \(2n\) via circulant matrix-vector multiplication.
Circulant matrices can also be used to generate a pre-conditioner for a Toeplitz matrix. Suppose $T$ is symmetric positive definite and Toeplitz.

Determine a circulant matrix $C$ such that

$$
\|C - T\|_F
$$

is minimized. Both matrices are determined by $n$ scalars. The optimization problem is trivially solved and the $n$ scalars defining $C$ can be computed in $O(n)$ operations (Tony Chan, 1988).

Or an approximation to the inverse of $T$ can be generated so that (Tyrtishnikov, 1988)

$$
\|I - C^{-1}T\|_F
$$

Given the Toeplitz matrix vector product and the fast solution or matrix vector product for a circulant matrix we can implement PCG with $O(n \log n)$ operations per iteration and have a natural parallel implementation.
The first preconditioner can be related to a preconditioner that we have already seen.

$\|x\|_F$ is unitarily invariant so we have

\[
\begin{align*}
\|C - T\|_F &= \|F(C - T)F^*\|_F \\
&= \|FCF^* - FTF^*\|_F \\
&= \|\Lambda - FTF^*\|_F \\
&= \|\Lambda - T\|_F
\end{align*}
\]

where $T$ is the transform domain version of $T$.

Clearly, the error is minimized by

\[\Lambda = diag(T).\]

So Tony Chan’s preconditioner is equivalent to diagonal preconditioning in the transform domain.

Some applications, e.g. adaptive filtering, work in the transform domain and create $T$ directly then use CG with diagonal preconditioning.