Background

C. T. Kelley
NC State University
tim.kelley@ncsu.edu
- vectors, matrices, norms
  $\ell^1$: max col sum . . .
- spectral radius
- scaled integral norms
A **Stationary Iterative Method** converts $Au = f$ to $u = Ru + c$ and the iteration is

$$u_{n+1} = Ru_n + c$$

$M$ is called the *iteration matrix*. This iteration is also called **Richardson Iteration**. The method is called stationary because the formula does not change as a function of $u_n$. 
Let $R$ be $N \times N$. Assume that

$$\|R\| < 1$$

for some induced matrix norm. Then

- $(I - R)$ is nonsingular
- $(I - R)^{-1} = \sum_{l=0}^{\infty} R^l$
- $\|(I - R)^{-1}\| \leq (1 - \|R\|)^{-1}$
We will show that the series
\[ \sum_{l=0}^{\infty} R^l = (I - R)^{-1}. \]

The partial sums

\[ S_k = \sum_{l=0}^{k} R^l \]

form a Cauchy sequence in $R^{N \times N}$. To see this note that for all $m > k$

\[ \|S_k - S_m\| \leq \sum_{l=k+1}^{m} \|R^l\|. \]

And ...
\[ \| R^l \| \leq \| R \|^l \text{ because } \| \cdot \| \text{ is a matrix norm that is induced by a vector norm. Hence} \]

\[ \| S_k - S_m \| \leq \sum_{l=k+1}^{m} \| R \|^l = \| R \|^{k+1} \left( \frac{1 - \| R \|^{m-k}}{1 - \| R \|} \right) \to 0 \]

as \( m, k \to \infty \). So the series converges. Let

\[ S = \sum_{l=0}^{\infty} R^l \]
Proof of Banach Lemma: II

Clearly

\[ RS = \sum_{l=0}^{\infty} R^{l+1} = \sum_{l=1}^{\infty} R^l = S - I \] 

and so

\[ (I - R)S = I \] 

and \[ S = (I - R)^{-1}. \]

Finally

\[ \|(I - R)^{-1}\| \leq \sum_{l=0}^{\infty} \|R\|^l = (1 - \|R\|)^{-1}. \]
If $\|R\| < 1$ for any induced matrix norm then the stationary iteration

$$u_{n+1} = R u_n + c$$

converges for all $c$ and $u_0$ to $u^* = (I - R)^{-1} c$

Proof: Clearly

$$u_{n+1} = \sum_{l=0}^{n} R^l c + R^n u_0 \rightarrow (I - R)^{-1} c = u^*.$$
Convergence Speed

Let $\|R\| = \alpha < 1$ and $u^* = (I - R)^{-1}c$. Then

$$\|u^* - u_n\| \leq \alpha^n \|u^* - u_0\|.$$

Proof:

$$\|u^* - u_n\| = \| \sum_{l=n}^{\infty} R^l c - R^n u_0 \|$$

$$= \| R(\sum_{l=n-1}^{\infty} R^l c - R^{n-1} u_0) \| = \| R(u^* - u_{n-1}) \|$$

$$\leq \alpha \| u^* - u_{n-1} \| \leq \cdots \leq \alpha^n \| u^* - u_0 \|.$$
The spectrum of $R \sigma(R)$, is the set of eigenvalues of $R$. The spectral radius of $R$ is

$$\rho(R) = \max_{\lambda \in \sigma(R)} |\lambda|$$

Theorem $\rho(R) < 1$ if and only if $\|R\| < 1$ for some induced matrix norm.

A stationary iterative method will $u_{n+1} = Ru_n + c$ converges for all initial iterates and right sides if and only if $\rho(R) < 1$.

The spectral radius does not depend on any norm.
Suppose you know that $\|\| \leq \alpha < 1$. Then

$$e_{n+1} = u^* - u_{n+1} = (Ru^* + c) - (Ru_n + c) = Re_n$$

Hence $\|e_n\| \leq \alpha^n \|e_0\|$ and

$$\|e_n\| \leq \tau \|e_0\| \text{ if } \alpha^n < \tau$$

or $n > \log(\tau) / \log(\alpha)$. 
If $\|I - A\| < 1$ then one can apply Richardson iteration directly to $Au = f$

$$u_{n+1} = (I - A)u_n + f$$

Sometimes one can find a approximate inverse $B$ for which

$$\|I - BA\| < 1$$

and precondition with $B$ to obtain

$$BAu = Bf$$ and the iteration is $u_{n+1} = (I - BA)u_n + Bf$
Approximate Inverse Preconditioning: I

**Theorem:** If $A$ and $B$ are $N \times N$ matrices and $B$ is an approximate inverse of $A$, then $A$ and $B$ are both nonsingular and

$$
\|A^{-1}\| \leq \frac{\|B\|}{1 - \|I - BA\|}, \quad \|B^{-1}\| \leq \frac{\|A\|}{1 - \|I - BA\|},
$$

and

$$
\|A^{-1} - B\| \leq \frac{\|B\|\|I - BA\|}{1 - \|I - BA\|}, \quad \|A - B^{-1}\| \leq \frac{\|A\|\|I - BA\|}{1 - \|I - BA\|}.
$$
Approximate Inverse Preconditioning: II

Proof: Let $R = I - BA$. The Banach Lemma implies that

$$I - R = I - (I - BA) = BA$$

is nonsingular. Hence both $A$ and $B$ are nonsingular. Moreover

$$\|A^{-1}B^{-1}\| = \|(I - R)^{-1}\| \leq \frac{1}{1 - \|R\|} = \frac{1}{1 - \|I - BA\|}.$$
Approximate Inverse Preconditioning: III

Use $A^{-1} = (I - R)^{-1}B$ to get the first part

$$\|A^{-1}\| \leq \|B\|\|(I - R)^{-1}\| \leq \frac{\|B\|}{1 - \|I - BA\|}.$$  

The second pair of inequalities follows from

$$A^{-1} - B = (I - BA)A^{-1}, \quad A - B^{-1} = B^{-1}(I - BA)$$

and the first.
Matrix Splittings

One way to convert $Au = f$ to $Ru = c$ is to split $A$

$$A = A_1 + A_2$$

where

- $A_1$ is nonsingular
- $A_1y = q$ is easy to solve for all $q$

and then solve

$$u = A_1^{-1}(f - A_2u) \equiv Ru + c.$$ 

Here $R = -A_1^{-1}A_2$ and $c = A_1^{-1}f$. Remember $A^{-1}z$ means solve $A_1y = z$, not compute $A_1^{-1}$. 

Jacobi Iteration: I

Write \( Au = f \) explicitly

\[
a_{11}\xi_1 + \ldots + a_{1N}\xi_N = \beta_1 \\
\vdots \\
a_{N1}\xi_1 + \ldots + a_{NN}\xi_N = \beta_N
\]

and solve the \( i \)th equation for \( \xi_i \), pretending the other components are know. You get

\[
\xi_i = \frac{1}{a_{ii}} \left( \beta_i - \sum_{j \neq i} a_{ij}\xi_j \right)
\]

which is a linear fixed point problem equivalent to \( Au = f \).
The iteration is

\[ \xi_i^{\text{New}} = \frac{1}{a_{ii}} \left( \beta_i - \sum_{j \neq i} a_{ij} \xi_j^{\text{Old}} \right) \]

So what are \( R \) and \( c \)?

- Split \( A = A_1 + A_2 \), where \( A_1 = D, A_2 = -L - U \),
- \( D \) is the diagonal of \( A \), and
- \( -L \) and \( -U \) are the (strict) lower and upper triangular parts.

Then \( x^{\text{New}} = D^{-1}(f + (L + U))x^{\text{Old}} \).
So the iteration is

$$u_{n+1} = D^{-1}(L + U)u_n + D^{-1}f$$

and the iteration matrix is $R_{JAC} = D^{-1}(L + U)$. Is there any reason for $\rho(R_{JAC}) < 1$?
Theorem: Let $A$ be an $N \times N$ matrix and assume that $A$ is strictly diagonally dominant. That is for all $1 \leq i \leq N$

$$0 < \sum_{j \neq i} |a_{ij}| < |a_{ii}|.$$  

Then $A$ is nonsingular and the Jacobi iteration converges to $u^* = A^{-1}f$ for all $f$. 

C. T. Kelley
Our assumptions imply that $a_{ii} \neq 0$, so the iteration is defined. We can prove everything else showing that

$$\|R_{JAC}\|_\infty < 1.$$  

Remember that $\|R_{JAC}\|_\infty < 1$ is the maximum absolute row sum. By assumptions, the $i$th row sum of $R = R_{JAC}$ satisfies

$$\sum_{j=1}^{N} |m_{ij}| = \frac{\sum_{j \neq i} |a_{ij}|}{|a_{ii}|} < 1.$$  

That’s it.
Observations

- Convergence of Jacobi implies $A$ is nonsingular.
- Showing $\|R_{JAC}\| < 1$ for any norm would do. The $l^\infty$ norm fits the assumptions the best.
- We have said nothing about the speed of convergence.
- Jacobi iteration does not depend on the ordering of the variables.
- Each $\xi_i^{\text{New}}$ can be processed independently of all the others. So Jacobi is easy to parallelize.
Gauss-Seidel Iteration

Gauss-Seidel changes Jacobi by updating each entry as soon as the computation is done. So

\[
\xi_i^{New} = \frac{1}{a_{ii}} \left( \beta_i - \sum_{j<i} a_{ij} \xi_j^{New} - \sum_{j>i} a_{ij} \xi_j^{Old} \right)
\]

You might think this is better, because the most up-to-date information is in the formula.
One advantage of Gauss-Seidel is that you need only store one copy of $x$. This loop does the job with only one vector.

\[
\text{for } i = 1:N \text{ do}
\]
\[
\text{sum} = 0;
\]
\[
\text{for } j \neq i \text{ do}
\]
\[
\text{sum} = \text{sum} + a(i,j)\cdot u(j)
\]
\[
\text{end for}
\]
\[
\text{u}(i) = (f(i) + \text{sum})/a(i,i)
\]
\[
\text{end for}
\]
Gauss-Seidel Iteration Matrix

From the formula, running for $i = 1, \ldots N$.

$$\xi_i^{New} = \frac{1}{a_{ii}} \left( \beta_i - \sum_{j<i} a_{ij} \xi_j^{New} + \sum_{j>i} a_{ij} \xi_j^{Old} \right)$$

you can see that

$$(D + U)u_{n+1} = b - Lu_n$$

so

$$R_{GS} = (D - U)^{-1}L \text{ and } c = (D - U)^{-1}b.$$
Backwards Gauss-Seidel

Gauss-Seidel depends on the ordering. Backwards Gauss-Seidel is

\[
\xi_i^{New} = \frac{1}{a_{ii}} \left( \beta_i - \sum_{j>i} a_{ij} \xi_j^{New} - \sum_{j<i} a_{ij} \xi_j^{Old} \right)
\]

running from \( i = N, \ldots 1 \). So \( R_{BGS} = (D - L)^{-1}U \).
Symmetric Gauss-Seidel

A symmetric Gauss-Seidel iteration is a forward Gauss-Seidel iteration followed by a backward Gauss-Seidel iteration. This leads to the iteration matrix

\[
R_{SGS} = R_{BGS} R_{GS} = (D - U)^{-1} L (D - L)^{-1} U.
\]

If \( A \) is symmetric then \( U = L^T \). In that event

\[
R_{SGS} = (D - U)^{-1} L (D - L)^{-1} U = (D - L^T)^{-1} L (D - L)^{-1} L^T.
\]
Add a relaxation parameter $\omega$ to Gauss-Seidel.

$$R_{SOR} = (D - \omega L)^{-1}((1 - \omega)D + \omega U).$$

Much better performance with good choice of $\omega$. 
Do example in one and two dimensions
Stress parallelism in Jacobi and RB ordering
Observations

- Gauss-Seidel and SOR depend on order of variables.
- So they are harder to parallelize.
- While they may perform better than simple Jacobi, it’s not a lot better.
- These methods are not competitive with Krylov methods.
- They require the least amount of storage, and are still used for that reason.
Splitting methods can be seen as preconditioned Richardson iteration. You want to find the preconditioner $B$ so that the iteration matrix from the splitting

$$R = -A_1^{-1}A_2 = I - BA.$$

So $I - R = BA$. 
Jacobi preconditioning

For the Jacobi splitting $A_1 = D$, $A_2 = -L - U$, we get

- $D^{-1}(L + U) = I - BA$ so
- $BA = I - D^{-1}(L + U) = D^{-1}A$

Jacobi preconditioning is multiplication by $D^{-1}$.

This can be a surprisingly good preconditioner for Krylov methods.
Poisson’s Equation in 1D

One space dimension

\[-u''(x) = f(x) \text{ for } x \in (0, 1)\]

Homogeneous Dirichlet boundary conditions

\[u(0) = u(1) = 0\]

Eigenvalue Problem

\[-u''(x) = \lambda u(x) \text{ for } x \in (0, 1), u(0) = u(1) = 0\]
Solution of Poisson’s Equation

We can diagonalize the operator using the solutions of the eigenvalue problem

\[ u_n(x) = \sin(n\pi x)/\sqrt{2}, \quad \lambda = n^2\pi^2 \]

\( \{u_n\} \) is an orthonormal basis for

\[ L_0^2 = cl_{L^2}\{u \in C([0, 1]) \mid u(0) = u(1) = 0\} \]

and the boundary value problem’s solution is

\[ u(x) = \sum_{n=1}^{\infty} u_n(x) \frac{1}{n^2\pi^2} \int_1^1 u_n(z)f(z)\,dz. \]
Properties of the Operator

The operator

\[- \frac{d^2}{dx^2} : C^2_0([0, 1]) \rightarrow C([0, 1])\]

is

- injective
- symmetric with respect to the $L^2$ scalar product
- has an $L^2_0$ orthonormal basis of eigenfunctions
- has positive eigenvalues
The solution of Poisson’s Equation on \([0, 1]\) with homogeneous Dirichlet boundary conditions is

\[
u(x) = \int_0^1 g(x, z)f(z)\,dz
\]

where

\[
g(x, z) = \begin{cases} 
x(1 - z) & 0 < x < z \\
z(1 - x) & z < x < 1 \end{cases}
\]
Add

\[ u(x + h) = u(x) + u'(x)h + u''(x)h^2/2 + u'''(h)h^3/6 + O(h^4) \]

to

\[ u(x - h) = u(x) - u'(x)h + u''(x)h^2/2 - u'''(h)h^3/6 + O(h^4) \]

and get

\[-u''(x) = (-u(x + h) + 2u(x) - u(x - h))/h^2 + O(h^2)\]
Equally spaced grid \( x_i = ih, \, 0 \leq i \leq N + 1, \, h = 1/(N + 1) \).
Approximate \( u(x_i) \) by \( \xi_i \). Let \( u = (\xi_1, \ldots, \xi_N)^T \).
Boundary conditions imply that \( \xi_0 = \xi_{N+1} = 0 \).
Finite difference equations at interior grid points are

\[
-\frac{\xi_{i-1}}{h^2} + 2\frac{\xi_i}{h^2} - \frac{\xi_{i+1}}{h^2} = b_i \equiv f(x_i)
\]

for \( 1 \leq i \leq N \).
Do inhomogeneous BC
Matrix Representation

\[ Au = b \]

where \( A \) is tridiagonal and symmetric

\[
A = \frac{1}{h^2} \begin{pmatrix}
2 & -1 & 0 & \ldots & 0, & 0 \\
-1 & 2 & -1 & \ldots & 0 \\
0 & -1 & 2 & -1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
0 & \ldots & 0 & -1 & 2 & -1 \\
0 & \ldots & \ldots & \ldots & 0 & -1 & 2
\end{pmatrix}
\]
Jacobi and Gauss-Seidel

Jacobi:

\[
\text{for } i=1:n \text{ do } \\
\xi_i^{\text{New}} \leftarrow \frac{1}{2}(h^2 b_i + \xi_{i-1}^{\text{Old}} + \xi_{i+1}^{\text{Old}}) \\
\text{end for}
\]

Gauss-Seidel:

\[
\text{for } i=1:n \text{ do } \\
\xi_i \leftarrow \frac{1}{2}(h^2 b_i + \xi_{i-1} + \xi_{i+1}) \\
\text{end for}
\]
Jacobi Iteration in MATLAB

```matlab
for ijac=1:N
    xnew(1) = .5*(h^2 * b(1) + xold(2));
    for i=2:N-1
        xnew(i) = .5*(h^2 * b(i) + xold(i-1) + xold(i+1));
    end
    xnew(N) = .5*(h^2 * b(N) + xold(N-1));
    xold=xnew;
end

How would you turn this into Gauss-Seidel with a text editor?
Let’s solve

\[-u'' = 0, \ u(0) = u(1) = 0.\]

with \( h = 1/101 \) and \( N = 100 \). The solution is \( u = 0 \). We will use as an intial iterate

\[ u_0 = x(1 - x) + \frac{1}{10} \cos(49\pi x) \]

We will take 100 Jacobi iterations.
Initial Error as Function of $x$
Final Error as Function of $x$
Final Error as Function of $x$
Theorem: A is symmetric positive definite. The eigenvalues are

$$\lambda_n = h^{-2} 2 (1 - \cos(\pi n h)) = \pi^2 n^2 + O(h^2).$$

The eigenvectors $$u_n = (\xi_1^n, \ldots, \xi_N^n)^T$$ are given by

$$\xi_i^n = \sqrt{2/h} \sin(ni \pi h)$$
Comments and Proof

- Eigenvalues agree with continuous problem to second order.
- $\kappa(A) = \lambda_N / \lambda_1 = O(N^2) = O(h^{-2})$.
- $\xi^n_i = u_n(x_i) \sqrt{2/h}$

Proof: Note that $\xi^n_0 = \xi^n_{N+1} = 0$

$$-\xi^n_{i-1} + 2\xi^n_i - \xi^n_{i+1}$$

$$= \sqrt{2/h}(-\sin(n(i - 1)\pi h) + 2 \sin(ni\pi h) - \sin(n(i - 1)\pi h))$$
Set \( x = ni\pi h \) and \( y = n\pi h \). Use the trig identities

\[
\sin(x \pm y) = \sin(x) \cos(y) \pm \sin(y) \cos(x)
\]

to get

\[
-\xi_{i-1}^n + 2\xi_i^n - \xi_{i+1}^n = -\sin(x - y) + 2\sin(x) - \sin(x + y)
\]

\[
= 2\sin(x)(1 - \cos(y)) = \lambda_n \xi_i^n
\]

as asserted.
Jacobi does Poorly for Poisson

If you apply Jacobi to Poisson’s equation, iteration matrix is

\[ R = -D^{-1}(L + U) = I - D^{-1}(D + L + U) = I - D^{-1}A \]

as we have seen. For Poisson, \( D = (2/h^2)I \) so

\[ R = I - D^{-1}A = I - (h^2/2)A. \]

The eigenvalues of \( R \) are \( \mu_n = 1 - (h^2/2)\lambda_n \). So

\[ \rho(R) = 1 - O(h^2) \]

which is very bad.

The performance gets worse as the mesh is refined!
Observations

- Jacobi (and GS, SOR, . . .) are not scalable.
  - The number of iterations needed to reduce the error by a given amount depends on the grid.
- Fixing this for PDE problems requires a different approach.
- You can solve the 1D problem in $O(N)$ time with a tridiagonal solver, but . . .
- Direct methods become harder to use for 2D and 3D problems on complex geometries with unstructured grids.
Poisson’s Equation in Two Dimensions

Equation: \(-u_{xx} - u_{yy} = f(x, y)\) for \(0 < x, y < 1\)

Boundary conditions: \(u(0, y) = u(x, 0) = u(1, y) = u(x, 1) = 0\)

- Similar properties to 1-D
- Physical Grid: \((x_i, x_j), x_i = i * h\).
- Begin with two-dimensional matrix of unknowns \(u_{ij} \approx u(x_i, x_j)\).
- Must order the unknowns (ie the grid points) to prepare for a packaged linear solver.
Background

Poisson's Equation

\[ u_{xx} \approx \frac{1}{h^2} (u(x - h, y) - 2u(x, y) + u(x + h, y)) \]

\[ u_{yy} \approx \frac{1}{h^2} (u(x, y - h) - 2u(x, y) + u(x, y + h)) \]

which leads to ...
Discrete 2D Poisson, Version 1

\[
\frac{1}{h^2} \left( -U_{i-1,j} - U_{i,j-1} + 4U_{ij} - U_{i+1,j} - U_{i,j+1} \right) = f_{ij} \equiv f(x_i, x_j)
\]

Jacobi, Gauss-Seidel, \ldots are still easy. Here’s GS

\begin{verbatim}
for i=1:N do
    for j=1:N do
        U_{ij} ← \frac{1}{4} \left( h^2 f_{ij} + U_{i-1,j} + U_{i,j-1} + U_{i+1,j} + U_{i,j+1} \right)
    end for
end for
\end{verbatim}

So how did I order the unknowns?
Ordering the Unknowns

\[
\begin{align*}
N^2 - N + 1 & \quad N^2 - N + 2 & \ldots & N^2 \\
: & : & \ldots & : \\
2N + 1 & 2N + 2 & \ldots & 3N \\
N + 1 & N + 2 & \ldots & 2N \\
1 & 2 & \ldots & N
\end{align*}
\]
Creating a Matrix-Vector Representation

Define

\[ u = (\xi_1, \ldots, \xi_{N^2})^T \in \mathbb{R}^{N^2} \]

by

\[ \xi_{N(i-1)+j} = U_{i,j} \quad \text{and} \quad \beta_{N(i-1)+j} = U_{i,j} \]

The Matrix representation is

\[ Au = b \]

where \ldots
Matrix Laplacian in 2D: 

\[ A = \frac{1}{h^2} \begin{pmatrix} 
T & -I & 0 & \ldots & 0, & 0 \\
-I & T & -I & \ldots & 0 & 0 \\
0 & -I & T & -I & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
0 & \ldots, & 0, & -I & T & -I \\
0 & \ldots, & \ldots, & 0 & -I & T 
\end{pmatrix} \]

where \( I \) is the \( N \times N \) identity matrix and \( T \) is the \( N \times N \) tridiagonal matrix.
Discrete Laplacian in 2D: I

\[ T = \begin{pmatrix}
  4 & -1 & 0 & \ldots & 0, & 0 \\
  -1 & 4 & -1 & \ldots & 0 & \ldots \\
  0 & -1 & 4 & -1 & \ldots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
  0 & \ldots, & 0, & -1 & 4 & -1 \\
  0 & \ldots, & \ldots, & 0 & -1 & 4 \\
\end{pmatrix} \]
Mapping the 2D vector to/from a 1D vector

Use the MATLAB `reshape` command.
Example: \( N = 3 \)

\[
\begin{pmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{pmatrix}
\]

\( u_{1d} = \text{reshape}(u_{2d}, N \times N, 1) = (1, 4, 7, 2, 5, 8, 3, 6, 9)^T \)

and \( u_{2d} = \text{reshape}(u_{1d}, N, N) \).
This means you can do things on the physical grid and still give solvers linear vectors when they need them.