Introduction and Stationary Iterative Methods

C. T. Kelley
NC State University
tim_kelley@ncsu.edu

Research Supported by NSF, DOE, ARO, USACE

DTU ITMAN, 2011

Outline

General References

- C. T. Kelley, Iterative Methods for Linear and Nonlinear Equations, no. 16 in Frontiers in Applied Mathematics, SIAM, Philadelphia, 1995.
- J. W. Demmel, Applied Numerical Linear Algebra, SIAM, Philadelphia, 1997.
- G. H. Golub and C. G. VanLoan, Matrix Computations, Johns Hopkins studies in the mathematical sciences, Johns Hopkins University Press, Baltimore, 3 ed., 1996.
- E. Isaacson and H. B. Keller, Analysis of numerical methods, Wiley, New York, 1966.
- G. W. Stewart, Introduction to matrix computations, Academic Press, New York, 1973.

Background

I assume you have had courses in

- Numerical methods (Gaussian elimination, SVD, QR, ...)
- Linear Algebra (Vector spaces, norms, inner products, ...)
- Calculus and differential equations

Some functional analysis would help.
If at any time I use something you are not familiar with, stop me and I will review.

Other things you should know

- I have never done this before.
- I may have too much or too little material.
- Some of the codes I will give you are new.
  So they may have a few bugs.
- I've set too many exercises. You'll have to be selective or stay up late.

What's in Monday's Directory

- A copy of today's lectures in pdf.
- A matlab code `gauss.m`, which you'll need for one of the exercises.
- A copy of a paper that may help.

Vectors

All vectors are column vectors of dimension N.

$$x = \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_N \end{pmatrix}, y = \begin{pmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_N \end{pmatrix}, \in R^N$$

Components of vectors use Greek letters.
Scalar product

$$x^T y = \sum_{i=1}^{N} \xi_i \eta_i,$$

where x^T is the row vector

$$x^T = (\xi_1, \dots, \xi_N).$$

Matrices

Matrices are in upper case, columns in lower case.

$$A = \begin{pmatrix} a_{11} & \dots & a_{1M} \\ a_{21} & \dots & a_{2M} \\ \vdots & \ddots & \vdots \\ a_{N1} & \dots & a_{NM} \end{pmatrix} = (a_1, a_2, \dots, a_M)$$

is an $M \times N$ matrix.

Transpose

$$A^T = \begin{pmatrix} a_{11} & \dots & a_{1N} \\ a_{21} & \dots & a_{2N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \dots & a_{MN} \end{pmatrix}$$

is an $N \times M$ matrix. This is consistent with x^T being a row vector. We mostly do real arithmetic. In complex arithmetic we use $A^\#$, the complex conjugate transpose, in place of A^T.

Linear Equations

$$Ax = b$$

explicitly

$$\begin{aligned}
a_{11}\xi_1 + \cdots + a_{1j}\xi_j + \cdots + a_{1N}\xi_N &= b_1 \\
\vdots & \\
a_{i1}\xi_i + \cdots + a_{ij}\xi_j + \cdots + a_{iN}\xi_N &= b_i \\
\vdots & \\
a_{N1}\xi_1 + \cdots + a_{Nj}\xi_j + \cdots + a_{NN}\xi_N &= b_N
\end{aligned}$$

Unless we explicitly say otherwise, A is nonsingular.

Vector

Our vector norms will be the l^p norms

$$\|x\|_p = \left(\sum_{j=1}^{N} |\xi_j|^p \right)^{1/p} \quad (1 \le p < \infty) \text{ and } \|x\|_\infty = \max_{1 \le j \le N} |\xi_j|$$

The l^2 norm is connected with the scalar product

$$x^T x = \|x\|_2^2.$$

Matrix Norms

Let $\| \cdot \|$ be a norm on R^N. The Induced Matrix Norm of an $N \times N$ matrix A is defined by

$$\|A\| = \max_{\|x\|=1} \|Ax\|.$$

We use induced norms. They have the important property:

$$\|Ax\| \le \|A\| \|x\|,$$

which implies that if $Ax = b$ then

$$\|A^{-1}\|^{-1} \|x\| \le \|b\| \le \|A\| \|x\|.$$

Types of linear equations

- Dense: A has very few non-zero entries.
- Sparse: A has many zeros.
- Structured: A has structure which algorithms can use
  For example: sparsity, symmetry ($A = A^T$), connection to differential or integral equations.
- Unstructured: One must use general methods.

Structure

- Sparsity
- Symmetry $A^T = A$
- A is symmetric positive definite (spd) if $A = A^T$ and $x^T Ax > 0$ for all $x \neq 0$.
  In this case $\|x\|_A = (x^T Ax)^{1/2}$ is a vector norm.
- Normality $A^T A = AA^T$
- Diagonalizability $A = V \Lambda V^{-1}$ where Λ is diagonal.
  A is diagonalizable if and only if A has N linearly independent eigenvectors and then $V = (v_1, \dots v_n)$.
  If A is symmetric then V is orthogonal, $VV^T = V^T V = I$.
  If A is normal then V is unitary, $VV^\# = V^\# V = I$.

Direct and Iterative Methods

Direct Methods solve $Ax = b$ in finite time in exact arithmetic. Examples:

- Gaussian Elimination and other matrix factorizations
- FFT for some problems (Toeplitz, Hankel, PDEs)

Iterative Methods produce a sequence $\{x_n\}$ which (you hope) converges to $x^* = A^{-1}x$. Examples:

- Stationary iterative methods (L1 and L3)
- Krylov methods (L2, 3, 4)
- Multigrid methods (L3)

Condition Numbers

The Condition Number of A relative to the norm $\| \cdot \|$ is

$$\kappa(A) = \|A\| \|A^{-1}\|,$$

where $\kappa(A)$ is understood to be infinite if A is singular.
$\kappa_p(A)$ means relative to the l^p norm.
If A is poorly conditioned (say $\kappa > 10^8$) we may not be able to obtain an accurate solution with any choice of algorithm.
Poor conditioning is a property of A. Algorithms cannot help.

Termination of Iterations

Most iterative methods terminate when the Residual

$$r = b - Ax$$

is sufficiently small. One termination criterion is

$$\frac{\|r_k\|}{\|r_0\|} < \tau$$

where $r_0 = b - Az_0$ and z_0 is a reference value.
So, what does a small relative residual tell us about the error

$$e = x^* - x?$$

Residuals, Errors, Conditioning

Theorem: *Let $b, x, x_0 \in R^N$. Let A be nonsingular and let $x^* = A^{-1}b$.*

$$\kappa(A)^{-1}\frac{\|r\|}{\|r_0\|} \le \frac{\|e\|}{\|e_0\|} \le \kappa(A)\frac{\|r\|}{\|r_0\|}.$$

We prove this. Note that

$$r = b - Ax = Ae$$

so

$$\|e\| = \|A^{-1}Ae\| \le \|A^{-1}\|\|Ae\| = \|A^{-1}\|\|r\|$$

and

$$\|r\| = \|Ae\| \le \|A\|\|e\|.$$

So,

$$\frac{\|e\|}{\|e_0\|} \le \frac{\|A^{-1}\|\|r\|}{\|A\|^{-1}\|r_0\|} = \kappa(A)\frac{\|r\|}{\|r_0\|}$$

and

$$\frac{\|e\|}{\|e_0\|} \ge \frac{\|A\|^{-1}\|r\|}{\|A^{-1}\|\|r_0\|} = \kappa(A)^{-1}\frac{\|r\|}{\|r_0\|}$$

as asserted.

Application of the Theorem

Most of the methods we use will set the reference vector to zero. Hence

$$r_0 = b \text{ and } e_0 = x^* = A^{-1}b$$

and the theorem connects the Relative Residual

$$\frac{\|b - Ax\|}{\|b\|}$$

to the Relative Error

$$\frac{\|x^* - x\|}{\|x^*\|}$$

If A is poorly conditioned, then termination on small relative residuals may be unreliable.

Stationary Iterative Methods

A Stationary Iterative Method converts $Ax = b$ to $x = Mx + c$ and the iteration is

$$x_{n+1} = Mx_n + c$$

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 x_n.

Banach Lemma

Let M be $N \times N$. Assume that

$$\|M\| < 1$$

for some induced matrix norm. Then

- $(I - M)$ is nonsingular
- $(I - M)^{-1} = \sum_{l=0}^{\infty} M^l$
- $\|(I - M)^{-1}\| \leq (1 - \|M\|)^{-1}$

Proof of Banach Lemma: I

We will show that the series

$$\sum_{l=0}^{\infty} M^l = (I - M)^{-1}.$$

The partial sums

$$S_k = \sum_{l=0}^{k} M^l$$

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

$$\|S_k - S_m\| \le \sum_{l=k+1}^{m} \|M^l\|.$$

And ...

Proof of Banach Lemma: II

$\|M^l\| \le \|M\|^l$ because $\|\cdot\|$ is a matrix norm that is induced by a vector norm. Hence

$$\|S_k - S_m\| \le \sum_{l=k+1}^{m} \|M\|^l = \|M\|^{k+1} \left(\frac{1 - \|M\|^{m-k}}{1 - \|M\|} \right) \to 0$$

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

$$S = \sum_{l=0}^{\infty} M^l$$

Proof of Banach Lemma: II

Clearly

$$MS = \sum_{l=0}^{\infty} M^{l+1} = \sum_{l=1}^{\infty} M^{l} = S - I \text{ and so}$$

$$(I - M)S = I \text{ and } S = (I - M)^{-1}.$$

Finally

$$\|(I - M)^{-1}\| \leq \sum_{l=0}^{\infty} \|M\|^{l} = (1 - \|M\|)^{-1}.$$

Convergence for Stationary Iterative Methods

If $\|M\| < 1$ for any induced matrix norm then the stationary iteration

$$x_{n+1} = Mx_n + c$$

converges for all c and x_0 to $x^* = (I - M)^{-1}c$

Proof: Clearly

$$x_{n+1} = \sum_{l=0}^{n} M^l c + M^n x_0 \rightarrow (I - M)^{-1}c = x^*.$$

Convergence Speed

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

$$\|x^* - x_n\| \le \alpha^n \|x^* - x_0\|.$$

Proof:

$$\begin{aligned} \|x^* - x_n\| &= \| \textstyle\sum_{l=n}^{\infty} M^l c - M^n x_0 \| \\ &= \|M(\textstyle\sum_{l=n-1}^{\infty} M^l c - M^{n-1} x_0)\| = \|M(x^* - x_{n-1})\| \\ &\le \alpha \|x^* - x_{n-1}\| \le \cdots \le \alpha^n \|x^* - x_0\|. \end{aligned}$$

Spectral Radius

The spectrum of M $\sigma(M)$, is the set of eigenvalues of M. The spectral radius of M is

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

Theorem $\rho(M) < 1$ if and only if $\|M\| < 1$ for some induced matrix norm.
A stationary iterative method will $x_{n+1} = Mx_n + c$ converges for all initial iterates and right sides if and only if $\rho(M) < 1$.
The spectral radius does not depend on any norm.

Predicting Convergence

Suppose you know that $\|M\| \leq \alpha < 1$. Then

$$e_{n+1} = x^* - x_{n+1} = (Mx^* + c) - (Mx_n + c) = Me_n$$

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

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

or $n > \log(\tau)/\log(\alpha)$.

Preconditioned Richardson Iteration

If $\|I - A\| < 1$ then one can apply Richardson iteration directly to $Ax = b$

$$x_{n+1} = (I - A)x_n + b$$

Sometimes one can find a approximate inverse B for which

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

and precondition with B to obtain

$$BAx = Bb \text{ and the iteration is } x_{n+1} = (I - BA)x_n + Bb$$

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

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

and

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

Approximate Inverse Preconditioning: II

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

$$I - M = I - (I - BA) = BA$$

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

$$\|A^{-1}B^{-1}\| = \|(I - M)^{-1}\| \le \frac{1}{1 - \|M\|} = \frac{1}{1 - \|I - BA\|}.$$

Approximate Inverse Preconditioning: III

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

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

The second pair of inequalities follows from

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

and the first.

Matrix Splittings

One way to convert $Ax = b$ to $Mx = c$ is to split A

$$A = A_1 + A_2$$

where

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

and then solve

$$x = A_1^{-1}(b - A_2 x) \equiv Mx + c.$$

Here $M = -A_1^{-1} A_2$ and $c = A_1^{-1} b$. Remember $A^{-1} z$ means solve $A_1 y = z$, not compute A_1^{-1}.

Jacobi Iteration: I

Write $Ax = b$ explicitly

$$\begin{aligned} a_{11}\xi_1 + \dots a_{1N}\xi_N &= \beta_1 \\ &\vdots \\ a_{N1}\xi_1 + \dots a_{NN}\xi_N &= \beta_N \end{aligned}$$

and solve the ith equation for ξ_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 $Ax = b$.

Jacobi Iteration: II

The iteration is

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

So what are M 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 $x^{New} = D^{-1}(b - (L + U))x^{Old}$.

Jacobi Iteration: III

So the iteration is

$$x_{n+1} = -D^{-1}(L+U)x_n + D^{-1}b$$

and the iteration matrix is $M_{JAC} = -D^{-1}(L+U)$.
Is there any reason for $\rho(M_{JAC}) < 1$?

Convergence for Strictly Diagonally Dominant A

Theorem: Let A be an $N \times N$ matrix and assume that A is strictly diagonally dominant. That is for all $1 \le i \le N$

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

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

Proof: Convergence for Strictly Diagonally Dominant A

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

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

Remember that $\|M_{JAC}\|_\infty < 1$ is the maximum absolute row sum. By assumptions, the ith row sum of $M = M_{JAC}$ 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 $\|M_{JAC}\| < 1$ for any norm would do. The l^∞ norm fit 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 ξ_i^{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.

Gauss-Seidel Iteration

One advantage of Gauss-Seidel is that you need only store one copy of x. This loop does the job with only one vector.

```
for i=1:N do
    sum=0;
    for $j \neq i$ do
        sum=sum+a(i,j)*x(j)
    end for
    x(i) = (b(i) + sum)/a(i,i)
end for
```

Gauss-Seidel Iteration Matrix

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

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

you can see that

$$(D + U) x_{n+1} = b - L x_n$$

so

$$M_{GS} = -(D + U)^{-1} L \text{ and } c = (D + U)^{-1} b.$$

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 $i = N, \dots 1$. So $M_{BGS} = -(D + L)^{-1} U$.

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

$$M_{SGS} = M_{BGS} M_{GS} = (D + U)^{-1} L (D + L)^{-1} U.$$

If A is symmetric then $U = L^T$. In that event

$$M_{SGS} = (D + U)^{-1} L (D + L)^{-1} U = (D + L^T)^{-1} L (D + L)^{-1} L^T.$$

SOR iteration

Add a relaxation parameter ω to Gauss-Seidel.

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

Much better performance with good choice of ω.

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 to Preconditioners

Splitting methods can be seen as preconditioned Richardson iteration.
You want to find the preconditioner B so that the iteration matrix from the splitting

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

So $I - M = BA$.

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.

References for Poisson's Equation

- P. Henrici, Discrete Variable Methods in Ordinary Differential Equations, Wiley, New York, 1962.
- R. J. LeVeque, Finite Difference Methods for Ordinary and Partial Differential Equations, SIAM, 2007.
- I. Stakgold, Green's Functions and Boundary Value Problems, Wiley-Interscience, New York, 1979.

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

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

$\{u_n\}$ is an orthonormal basis for

$$L^2_0 = cl_{L^2}\{u \in C([0,1]) \mid u(0) = u(1) = 0\}$$

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

$$\frac{-d^2}{dx^2} : C_0^2([0,1]) \rightarrow C([0,1]) \text{ is}$$

- injective
- symmetric with respect to the L^2 scalar product
- has an L_0^2 orthonormal basis of eigenfunctions
- has positive eigenvalues

Solution Operator

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

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

Central Difference Approximation

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

$$-u''(x) = (-u(x+h) + 2u(x) - u(x-h))/h^2 + O(h^2)$$

Finite Difference Equations

Equally spaced grid $x_i = ih$, $0 \le i \le N+1$, $h = 1/(N+1)$.
Approximate $u(x_i)$ by ξ_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

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

for $1 \le i \le N$.

Matrix Representation

$$Au = b$$

where A is tridiagonal and symmetric

$$A = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & 0 & \dots & 0, & 0 \\ -1 & 2 & -1 & ,0 & \dots & 0 \\ 0 & -1 & 2 & -1, & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots, & ,0, & -1 & 2 & -1 \\ 0 & \dots, & \dots,, & 0 & -1 & 2 \end{pmatrix}$$

Jacobi and Gauss-Seidel

Jacobi:

```
for i=1:n do
    $\xi_i^{New} \leftarrow (1/2)(h^2 b_i + \xi_{i-1}^{Old} + \xi_{i+1}^{Old})$
end for
```

Gauss-Seidel:

```
for i=1:n do
    $\xi_i \leftarrow (1/2)(h^2 b_i + \xi_{i-1} + \xi_{i+1})$
end for
```

Jacobi Iteration in 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?

Jacobi Example

Let's solve

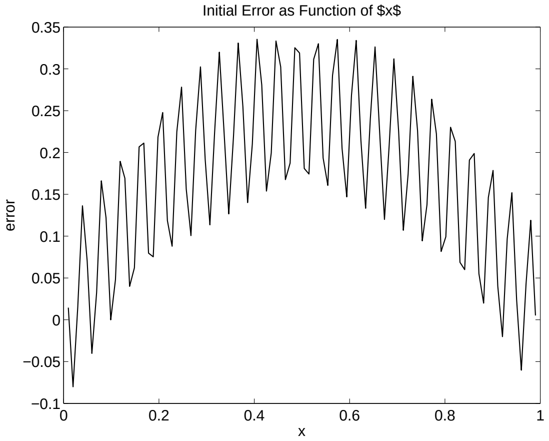
$$-u'' = 0, \; u(0) = u(1) = 0.$$

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

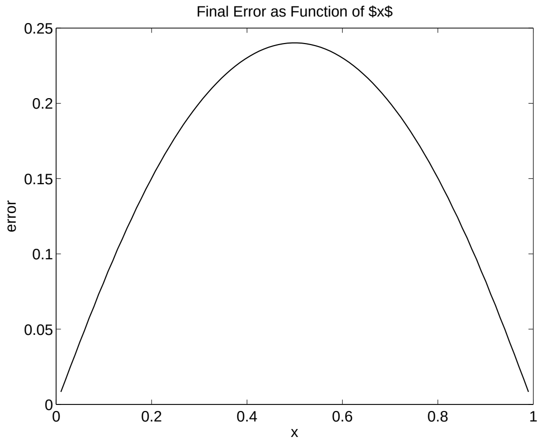
$$u_0 = x(1-x) + \frac{1}{10}\cos(49\pi x)$$

We will take 100 Jacobi iterations.

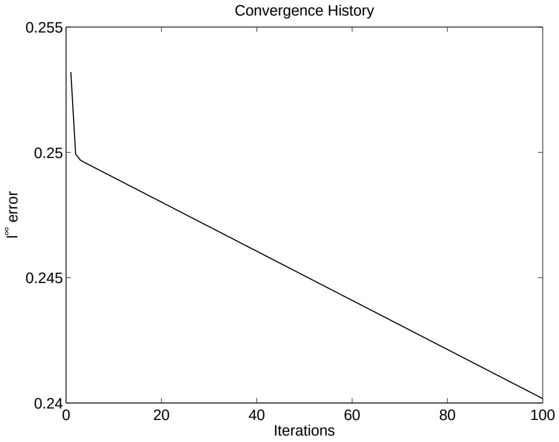
Initial Error as Function of x



Final Error as Function of x



Final Error as Function of x



Eigenvalues and Eigenvectors

Theorem: A is symmetric positive definite. The eigenvalues are

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

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

$$\xi_i^n = \sqrt{2/h} \sin(n i \pi h)$$

Comments and Proof

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

Proof: Note that $\xi_0^n = \xi_{N+1}^n = 0$

$$-\xi_{i-1}^n + 2\xi_i^n - \xi_{i+1}^n$$

$$= \sqrt{2/h}(-\sin(n(i-1)\pi h) + 2\sin(ni\pi h) - \sin(n(i-1)\pi h))$$

End of Proof

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

$$-\xi^n_{i-1} + 2\xi^n_i - \xi^n_{i+1} = -sin(x-y) + 2\sin(x) - \sin(x+y)$$

$$= 2\sin(x)(1-\cos(y)) = \lambda_n \xi^n_i$$

as asserted.

Jacobi does Poorly for Poisson

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

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

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

$$M = I - D^{-1}A = I - (h^2/2)A.$$

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

$$\rho(M) = 1 - O(h^2)$$

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.

$$u_{xx} \approx \frac{1}{h^2}\left(u(x-h,y) - 2u(x,y) + u(x+h,y)\right)$$

$$u_{yy} \approx \frac{1}{h^2}\left(u(x,y-h) - 2u(x,y) + u(x,y_h)\right)$$

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, ... are still easy. Here's GS

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

So how did I order the unknowns?

Ordering the Unknowns

$$\begin{array}{llll}
N^2-N+1 & N^2-N+2 & \dots & N^2 \\
\vdots & \vdots & \dots & \vdots \\
2N+1 & 2N+2 & \dots & 3N \\
N+1 & N+2 & \dots & 2N \\
1 & 2 & \dots & N
\end{array}$$

Creating a Matrix-Vector Representation

Define

$$u = (\xi_1, \dots, \xi_{N^2})^T \in R^{N^2}$$

by

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

The Matrix representation is

$$Au = b$$

where ...

Matrix Laplacian in 2D: I

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

where I is the $N \times N$ identity matrix and T is the $N \times N$ tridiagonal matrix

Discrete Laplacian in 2D: I

$$T = \begin{pmatrix} 4 & -1 & 0 & \dots & 0, & 0 \\ -1 & 4 & -1 & ,0 & \dots & 0 \\ 0 & -1 & 4 & -1, & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots, & ,0, & -1 & 4 & -1 \\ 0 & \dots, & \dots,, & 0 & -1 & 4 \end{pmatrix}$$

Mapping the 2D vector to/from a 1D vector

Use the MATLAB `reshape` command.
Example: $N = 3$

$$u_{2d} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

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

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

Richardson Iteration

Use the trapezoid rule to discretize the integral equation

$$u(x) - \frac{1}{2}\int_0^1 \sin(x-y)u(y)\,dy = \cos(x)$$

If you write your discrete equation as $u - Mu = b$, prove that $\rho(M) < 1/2$. Write a MATLAB code to demonstrate that the convergence is independent of the mesh width.

Poisson Equation

- Compute the eigenvalues and eigenvectors for the 2D discrete Poission equation.
- Encode the 2D Laplacian as a MATLAB sparse matrix and use the `eigs` command to find a few eigenvectors and eigenvalues to verify your work in the previous exercise.
- Solve the 1D and 2D Poission equations with Jacobi and Gauss-Seidel with zero boundary data and $f \equiv 1$ as the right side. Try more interesting right sides $f(x)$ and $f(x, y)$.