Numerical Methods in Economics

MIT Press, 1998

Notes for Chapter 3: Linear Equations and Iterative Methods

September 30, 2008

Linear Equations

- Linear equation

$$Ax = b$$

  where $b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$

- Importance of linear solution methods

  - Some important problems are linear problems
  - Nonlinear solution methods are generally sequences of linear problems
  - Solution methods illustrate general ideas and concepts for solving equations

Triangular Systems

- A is *lower triangular* if all nonzero elements lie on or below the diagonal:

$$A = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

  – *Upper triangular*: all nonzero entries on or above the diagonal.
  – A is a *triangular matrix* if it is either upper or lower triangular.
  – A *diagonal* matrix has nonzero elements only on the diagonal.
  – A triangular matrix is nonsingular iff all diagonal elements are nonzero
  – Lower (upper, diagonal) triangular matrices are closed under multiplication and inversion.

- Solve triangular systems by *back-substitution*.
  - Assume: A is lower triangular, nonsingular.
  - Back-substitution is

$$x_1 = \frac{b_1}{a_{11}} \tag{3.1.1}$$

$$x_k = \frac{b_k - \Sigma_{j=1}^{k-1} a_{kj}\, x_j}{a_{kk}}, k = 2, 3, ..., n \tag{3.1.2}$$

    is well-defined for nonsingular, lower triangular matrices.
  - Similar for upper triangular except we begin with $x_n = b_n/a_{nn}$ and proceed to x_k, $k = n - 1, n - 2, ...2, 1$.

Gaussian Elimination, LU Decomposition

- Suppose A is nonsingular
- Factor $A = LU$ where L is lower triangular, U is upper triangular
  - Computed by Gaussian elmination; see details in any numerical analysis book.
  - There are many operations like (3.1.1, 3.1.2) executed to find L and U.
  - Rows and columns often must be reordered to avoid division by zero – pivoting
  - Given LU decomposition, find x by
    * Solve $Lz = b$ by back substitution
    * Solve $Ux = z$ by back substitution

QR factorization

- Definition: A is *orthogonal* iff $A^\top A$ is a diagonal matrix
- Factor $A = QR$ where Q is orthogonal and R is upper triangular
  - See details in books on linear numerical analysis.
  - Given QR decomposition, find x by
    - Solve $Qz = b$ by $z = \left(Q^\top Q\right)^{-1} Q^\top b$ which requires only inversion of a diagonal matrix and matrix multiplication
    - Solve $Rx = z$ by back substitution

Cholesky Factorization

- Suppose A is symmetric positive definite
- Factor $A = LL^\top$ where L is lower triangular
  - L is a Cholesky factor, or "squareroot" of A.
  - See details in book.
  - A special case of LU decomposition: $L^\top$ is upper triangular and is U in LU decomposition procedure.

Cramer's Rule

- Cramer's rule solves for x in $Ax = b$ by applying a direct formula to the elements of A and b.
- Is only method for symbolic expressions
- Very slow, with operation count of $\mathcal{O}(n!)$.

Error Bounds

We want to approximate errors in solving $Ax = b$.

- True system: $Ax = b$
  - Errors in b (due to roundoff, etc.) cause computer to solve $A\tilde{x} = b + r$
  - Error in solution is $e \equiv \tilde{x} - x$
  - Hence, $e = A^{-1} r$.
- Sensitivity of e to r is

$$\frac{\| e \|}{\| x \|} \div \frac{\| r \|}{\| b \|},$$

  - Equals percentage error in x relative to the percentage error in b – an elasticity
  - Minimum sensitivity is 1, achieved when $A = aI$, $x = b/a$.
  - Sensitivity can be computed for *any* numerical problem
  - Sensitivity≡Elasticity

- Matrix analysis
  - If $\| \cdot \|$ is a norm on $\mathbb{R}^n$, define norm of A

$$\| A \| \equiv \max_{x \neq 0} \frac{\| Ax \|}{\| x \|} = \max_{\| x \| = 1} \| Ax \|$$

  - Spectral radius: $\rho(A) = \max\{\| \lambda \| \mid \lambda \text{ an eigenvalue of } A\}$
  - For any norm $\| \cdot \|$, $\rho(A) \leq \| A \|$.
- The *condition number of A relative to* $\| \cdot \|$ is

$$\text{cond}(A) \equiv \| A \| \| A^{-1} \|,$$

  - Upper and lower bounds on error

$$\frac{\| e \|}{\| x \|} \leq \frac{\| r \|}{\| b \|} \text{cond}(A) \tag{3.5.1}$$

  - Depends on norm $\| \cdot \|$
  - Numerical analysis typically uses $\| \cdot \|_\infty$

- Spectral condition number
  - Define:
$$\text{cond}_*(A) \equiv \frac{\max_{\lambda\in\sigma(A)}|\lambda|}{\min_{\lambda\in\sigma(A)}|\lambda|} = \frac{\rho(A)}{\rho(A^{-1})}$$
  - Theorem: For any norm,
$$\text{cond}(A) \geq \text{cond}_*(A)$$
  - Practical fact: For standard norms, such as max or Euclidean norm,
$$\text{cond}(A) \approx \text{cond}_*(A)$$
  - We arrive at an *approximate* error bound
$$\frac{\| e \|}{\| x \|} \lesssim \frac{\| r \|}{\| b \|}\,\text{cond}_*(A)$$
    which is more practical since $\text{cond}_*(A)$ is relatively easy to estimate and we are only interested in orders of magnitude.

- Hilbert matrix example:
  - Definition

$$H_n \equiv \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \cdots & \cdots & \frac{1}{2n-1} \end{pmatrix}$$

  - Condition numbers (table in book has some errors)

n:	4	5	6	8	11
$\mathrm{cond}_*(H_n)$:	1.6(4)	4.8(5)	1.5(7)	1.5(10)	5.2(14)
$\mathrm{cond}_\infty(H_n)$:	2.8(4)	9.4(5)	2.9(7)	3.4(10)	1.2(15)

- Notes on condition numbers
  - The error bound is an *approximate* upper bound; errors could possibly be greater, but are more likely to be substantially less.
  - Condition numbers are sensitive to scaling
    * Consider the problem $x = a,\ My = b$; trivial to solve
    * This matrix has spectral condition number M :

$$\begin{pmatrix} 1 & 0 \\ 0 & M \end{pmatrix}$$

    * Define $z = My$; problem becomes one with condition number 1.

$$x = a,\ z = b$$

    * Lesson: change in units (a.k.a., rescaling), or a linear transformation ("pre-conditioning") may improve conditioning
    * Recommendation: formulate problem so answer is $O(1)$.
    * See McCullough and Vinod, *AER* (2003), and followup comments.

Iterative Methods

- Direct methods (LU, QR, Cholesky)
  - High accuracy
  - Time cost is order n^3; too large for large matrices.
- Iterative methods
  - Can handle large problems
  - Less accuracy
  - Less time
  - User has time-accuracy tradeoffs under his control
  - Ideas are used in nonlinear as well as linear problems.

- Fixed-Point Iteration.
  - $G(x) \equiv Ax - b + x$
  - Compute sequence
    $$x^{k+1} = G(x^k) = (A + I)x^k - b \tag{3.6.1}$$
  - Clearly x is a fixed point of $G(x)$ if and only if x solves $Ax = b$.
  - (3.6.1) will converge iff $|\lambda| < 1$ for all $\lambda \in \sigma\,(A + I)$; i.e., G is a contraction

- Gauss-Jacobi
  - Idea: Replace *system* of multivariate linear equations with *sequence* of single variable linear problems
  - The equation from the first row of $Ax = b$:

$$\begin{aligned} b_1 &= a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ &\Longrightarrow x_1 = a_{11}^{-1}(b_1 - a_{12}x_2 - \cdots - a_{1n}x_n). \end{aligned}$$

  - In general, if $a_{ii} \neq 0$, the ith row of A implies

$$x_i = \frac{1}{a_{ii}} \left\{ b_i - \sum_{j \neq i} a_{ij}\, x_j \right\}.$$

  - Turn this into an iterative process as in

$$x_i^{k+1} = \frac{1}{a_{ii}} \left\{ b_i - \sum_{j \neq i} a_{ij}\, x_j^k \right\}, \quad i = 1, \ldots, n \tag{3.6.2}$$

  - Note: no x_i^{k+1} is used until each x_i^{k+1} has been computed.
  - We *hope* that (3.6.2) converges to the true solution
  - Results are sensitive to which equation goes with which equation

- Gauss-Seidel
  - Idea: Replace multivariate system with sequence of univariate problems *and* use new information *immediately*
  - Given x^k, compute guess for x_1 from row 1

$$x_1^{k+1} = a_{11}^{-1}(b_1 - a_{12}x_2^k - \cdots - a_{1n}x_n^k),$$

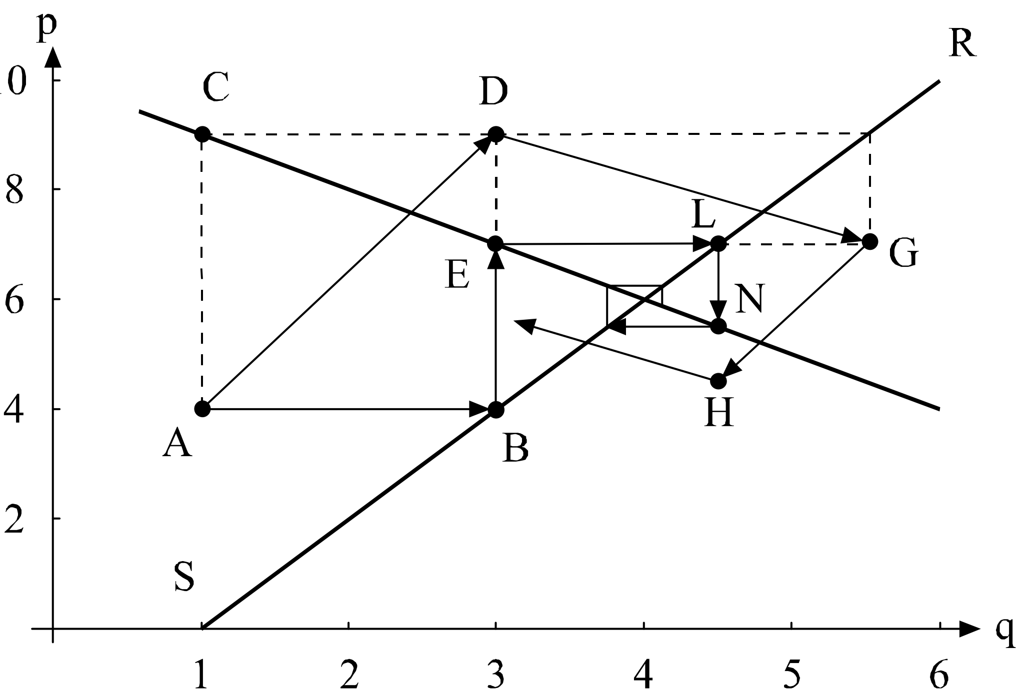
  - Use x_1^{k+1} *immediately* to compute x_2^{k+1}:

$$x_2^{k+1} = a_{22}^{-1}(b_2 - a_{21}x_1^{k+1} - a_{23}x_3^k - \cdots - a_{2n}x_n^k).$$

  - In general, define the sequence $\{x^k\}_{k=1}^{\infty}$

$$x_i^{k+1} = \frac{1}{a_{ii}} \left\{ b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}\, x_j^k \right\} , i = 1, \cdots, n \tag{3.6.3}$$

  - Each component of x^{k+1} is used immediately after computed
  - Gauss-Seidel sensitive to (i) matching between variables and equations, and (ii) ordering of equations.



Gauss-Jacobi (ADGH) versus Gauss-Seidel (ABELN..)

Tatonnement and Iterative Schemes.

- Equilibrium problem
  - Inverse demand equation $p = 10 - q$
  - Supply curve $q = p/2 + 1$
  - Equilibrium

$$p + q = 10 \tag{3.6.6a}$$

$$p - 2q = -2 \tag{3.6.6b}$$

- Gauss-Jacobi
  - Initial guess: $p = 4$ and $q = 1$, point A in figure 3.2.
  - New guess:
    * Solve demand eqn for p, holding q fixed; move to C on the demand eqn.
    * Move from A to the B on supply curve to solve for q holding p fixed.
    * Similar to a pair of auctioneers
    * General iteration is

$$\begin{aligned} q_{n+1} &= 1 + \tfrac{1}{2} p_n, \\ p_{n+1} &= 10 - q_n. \end{aligned} \tag{3.6.7}$$

    * Slow convergence, spiraling to $p = 6$ and $q = 4$.

- Gauss-Seidel
  - Start from A.
  - Use the supply curve to get a new q at B
  - Move from B up to E, get new p from the demand equation.
  - Similar to an auctioneer alternating between markets.
  - Also called hog cycle – firms expect p_0, produce q_1, which causes prices to rise to p_1, causing production to be q_2, and so on.
  - General iteration is

$$\begin{aligned} q_{n+1} &= 1 + \tfrac{1}{2} p_n, \\ p_{n+1} &= 10 - q_{n+1}. \end{aligned} \tag{3.6.8}$$

  - Gauss-Seidel converges more rapidly.

Operator Splitting Approach.

- General strategy: Transform problem into *another* problem with *same* solution where fixed-point iteration is cheap and works.
  - Problem: $Ax = b$.
  - Split A into two operators
    $$A = N - P, \tag{3.7.1}$$
  - Note: $Ax = b$ if and only if $Nx = b + Px$.
  - Define the iteration
    $$Nx^{m+1} = b + Px^m \tag{3.7.2}$$
  - Goal: find N so that
    * each step of (3.7.2) is easy to solve, and
    * (3.7.2) converges

- Gauss-Jacobi is a splitting with diagonal N

$$N = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}, \quad P = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{pmatrix}.$$

- Gauss-Seidel is a splitting with lower triangular N

$$N = \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}, \quad P = -\begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

- Many possible splittings; just keep N simple
- Note: A can be any operator, not just linear operator

Convergence of Iterative Schemes.

- Rate of convergence.
  - Suppose $A = N - P$, and $Ax^* = b$.
  - Consider $Nx^{m+1} = b + Px^m$
    - Error $e^m \equiv x^* - x^m$ obeys iteration $e^m = (N^{-1}P)^m e^0$.
    - $e^m \to 0$ iff $(N^{-1}P)^m e^0 \to 0$ iff $\rho(N^{-1}P) < 1$.
  - At best linearly convergent
- *Diagonal dominance.* A is diagonally dominant iff

$$\sum_{j \neq i} |a_{ij}| < |a_{ii}|, \quad i = 1, \cdots, n.$$

Theorem 1 *If A is diagonally dominant, both Gauss-Jacobi and Gauss-Seidel iteration schemes are convergent for all initial guesses.*

- Economic intuition:
  - If $(Ap)_i$ is excess demand for good i at price $p \in \mathbb{R}^n$, then diagonal dominance says excess demand for each good is more sensitive to its own price than to a similar change in all other prices.
  - Also known as *gross substitutability*.
- This tells us how to match variables with equations:
  - Match x_i with some equation where x_i has a large coefficient
  - In tatonnement, use the apple excess demand equation to compute the apple price, use cheese excess demand equation to compute cheese price, etc.

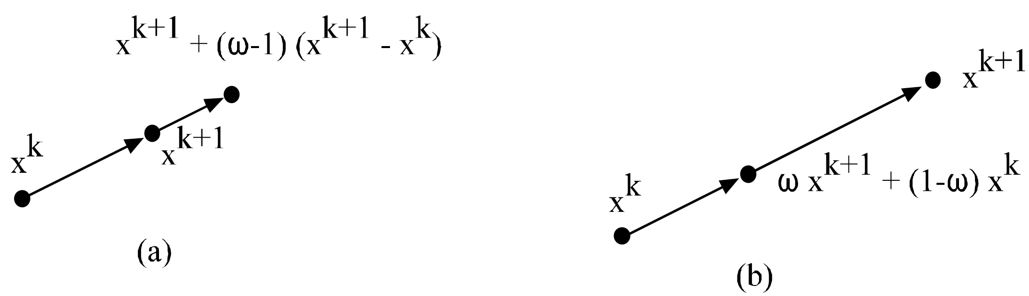
Acceleration and Stabilization Methods

- Convergence od Gaussian is linear; no way to change that.
- Sometimes we can increase the linear rate of convergence.
- Extrapolation and Dampening.
  - To solve $Ax = b$, define $G = I - A$.
  - Consider the iteration
  $$x^{k+1} = G\,x^k + b \tag{3.9.1}$$
    * (3.9.1) will converge iff $\rho(G) < 1$
    * If $\rho(G) < 1$ then G is a contraction mapping with contraction rate $\rho\,(G)$
    * If $\rho(G)$ is close to 1, convergence will be slow.

– For scalar ω, consider

$$\begin{aligned} x^{k+1} &= \omega G x^h + \omega b + (1-\omega)x^k \\ &\equiv G_{[\omega]} x^k + \omega b \end{aligned} \tag{3.9.2}$$

* When $\omega > 1$, (3.9.2) is called *extrapolation;* see Figure 3.3.b.
  · Convergence implies that $Gx^k + b$ is a good direction to move
  · Convergence may be accelerated by going further each iteration.
* When $\omega < 1$, (3.9.2) is called *dampening*; see Figure 3.3.b.
  · $Gx^k + b$ may be a good direction, but overshoots solution
  · If $\omega < 1$, (3.9.2) may avoid overshooting and converge



- Suppose all eigenvalues of G are real
  - Decompose $G = P^{-1}DP$. Then

$$\omega G + (1-\omega)I = P^{-1}(\omega D + (1-\omega)I)P$$

    and

$$\sigma(\omega G + (1-\omega)I) = \omega\sigma(G) + 1 - \omega.$$

  - From definition of $G_{[\omega]}$, the scalar ω
    * stretches or shrinks the spectrum of G,
    * then flips it around 0 if $\omega < 0$, and
    * finally shifts it by $1 - \omega$.

– Choose ω to minimize $\rho(G_{[\omega]})$. If $\sigma(G) \subset \mathbb{R}$, this is

$$\min_{\omega} \max_{\lambda \in \sigma(\omega G + (1-\omega) I)} |\lambda| \tag{3.9.3}$$

with solution

$$\omega^* = \frac{2}{2 - m - M} \tag{3.9.4}$$

and spectral radius

$$\rho(G_{[\omega^*]}) = \left| \frac{M - m}{2 - M - m} \right|. \tag{3.9.5}$$

– Properties of (3.9.5)

* If $-1 < m < M < 1$, ω^* accelerates convergence
* If $M < 1$, then $\rho(G_{[\omega^*]}) < 1$ for all m, ω^* stabilizes explosive roots
* If $M > 1$ and $m < -1$ (both kinds of unstable roots) then (3.9.5) fails

- Successive Overrelaxation
  - Combine Gauss-Seidel with extrapolation.
  - For scalar ω, define

$$x_i^{k+1} = \omega \left(\frac{1}{a_{ii}} \right) \left[b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{k} \right] + (1-\omega)\; x_i^k. \tag{3.9.6}$$

  - The i'th component of the $k+1$ iterate is a linear combination, parameterized by ω, of the Gauss-Seidel value and the kth iterate.
  - Decompose $A = D+L+U$ where D is diagonal, L is lower triangular, and U is upper triangular.

– SOR has the convenient matrix representation

$$(D + \omega L)\, x^{k+1} = ((1-\omega)\, D - \omega U)\ x^k + \omega b.$$

Define $M_\omega \equiv D + \omega L$, $N_\omega \equiv (1-\omega)\, D - \omega U$; then

$$x^{k+1} = M_\omega^{-1}\, N_\omega\, x^k + \omega M_\omega^{-1}\, b \tag{3.9.7}$$

– Best choice for ω is $\arg\min_\omega \rho(M_\omega^{-1}\, N_\omega)$.

  * Gains can be substantial
  * Optima ω is difficult to compute; see Hageman and Young (1981) for ways to estimate it.

Dampening to Stabilize an Unstable "Hog Cycle".

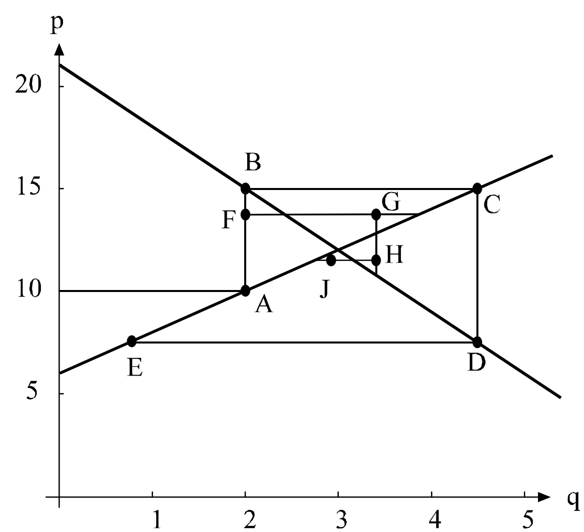
- Suppose inverse demand is $p = 21 - 3q$ and supply is $q = p/2 - 3$
- Linear system is not diagonally dominant:

$$\begin{pmatrix} 1 & 3 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} 21 \\ 6 \end{pmatrix} \tag{3.9.8}$$

- Gauss-Seidel is unstable:

$$p_{n+1} = 21 - 3q_n \tag{3.9.9a}$$

$$q_{n+1} = \frac{1}{2}p_{n+1} - 3 \tag{3.9.9b}$$



- Stabilize through damping: if $\omega = 0.75$, then we have stable system

$$p_{n+1} = 0.75(21 - 3q_n) + 0.25p_n \tag{3.9.10a}$$

$$q_{n+1} = 0.75(\frac{1}{2}p_{n+1} - 3) + 0.25q_n \tag{3.9.10b}$$

Exatrapolation to Accelerate Convergence in a Game

- Assume firm two's reaction curve is $p_2 = 2 + 0.80p_1 \equiv R_2(p_1)$, and firm one's reaction curve is $p_1 = 1 + 0.75p_2 \equiv R_1(p_2)$.
- Equilibrium system is diagonally dominant
- Gauss-Seidel is the iterative scheme

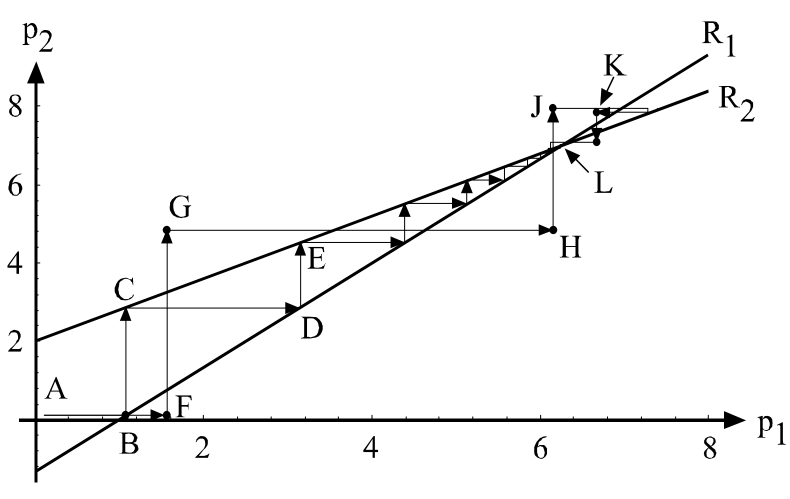
$$p_1^{n+1} = R_1\left(p_2^n\right) \tag{3.9.12a}$$

$$p_2^{n+1} = R_2\left(p_1^{n+1}\right) \tag{3.9.12b}$$

- Accelerate (3.9.12). If $\omega = 1.5$, we arrive at faster scheme:

$$p_1^{n+1} = 1.5R_1\left(p_2^n\right) - 0.5p_1^n, \tag{3.9.13a}$$

$$p_2^{n+1} = 1.5R_2\left(p_1^{n+1}\right) - 0.5p_2^n. \tag{3.9.13b}$$



Sparse Matrices

- Classification
  - *Dense:* A is *dense* if $a_{ij} \neq 0$ for most i, j.
  - *Sparse:* A is *sparse* if $a_{ij} = 0$ for most i, j
    - “most” is not a precise definition
    - In practice, we are studying a class of problems of varying dimension and “most” means that the number of nonzero elements is Mn form some fixed M.

- Diagonal matrix:

$$D = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix}$$

$$Dx = b \Longrightarrow x_i = \frac{b_i}{d_i}$$

- Tridiagonal matrix has all nonzero elements on or next to the diagonal

$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & & \cdots & 0 \\ 0 & a_{32} & a_{33} & a_{34} & \cdots & 0 \\ 0 & 0 & a_{43} & a_{44} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

and $Ax = b$ is solved by

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 && \text{(Row 1)} \\ &\Longrightarrow x_2 = \frac{b_1 - a_{11}x_1}{a_{12}} \\ &= \alpha_2 - \beta_2 x_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = a_{21}x_1 + a_{22}(\alpha_2 - \beta_2 x_1) + a_{23}x_3 &= b_2 && \text{(Row 2)} \\ &\Longrightarrow x_3 = \alpha_3 - \beta_3 x_1 \\ &\vdots \\ x_n &= \alpha_{n-1} - \beta_{n-1}x_1 && \text{(Row n-1)} \\ a_{n,n-1}(\alpha_{n-2} - \beta_{n-2}x_1) + a_{nn}(\alpha_{n-1} - \beta_{n-1}x_1) &= b_n && \text{(Row n)} \\ &\Longrightarrow x_1 \text{ solution} \end{aligned}$$

- Taking advantage of sparseness
  - Storage:
    * Dense: n^2 numbers
    * Sparse: store only $m \sim O(n)$ nonzero elements along with their locations.
  - Operations: Matrix multiplication – Ax or yB
    * Dense uses $2n^2$ flops
    * Sparse approach uses $2m \sim O(n)$ flops
- Application: Ergodic distribution of a finite Markov chain
  - Markov transition matrices, Π, are often sparse
  - Ergodic distribution x solves $x\Pi = x$.
  - Solve by iteration: $x^{k+1} = x^k\Pi$; works well since $x^k\Pi$ is fast if Π is sparse.
- Software: Standard packages (Matlab, Mathematica, etc.) offer sparse storage and operation options.

Summary

- Linear equations are essential in numerical methods
  - Linear problems are common
  - Nonlinear problems are reduced to a sequence of linear problems
- Linear equation methods often inspire methods for nonlinear problems
  - The key concepts behind Gauss-Jacobi and Gauss-Seidel methods can also be applied to nonlinear problems
  - The key concepts behind relaxation methods can also be applied to nonlinear problems