Non-Linear Least Squares and Sparse Matrix Techniques: Fundamentals

Richard Szeliski
Microsoft Research

UW-MSR Course on
Vision Algorithms

CSE/EE 577, 590CV, Spring 2004

Readings

- Press *et al.*, Numerical Recipes, Chapter 15 (Modeling of Data)
- Nocedal and Wright, Numerical Optimization, Chapter 10 (Nonlinear Least-Squares Problems, pp. 250-273)
- Shewchuk, J. R. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain.
- Bathe and Wilson, Numerical Methods in Finite Element Analysis, pp.695-717 (sec. 8.1-8.2) and pp.979-987 (sec. 12.2)
- Golub and VanLoan, Matrix Computations. Chapters 4, 5, 10.
- Nocedal and Wright, Numerical Optimization. Chapters 4 and 5.
- Triggs *et al.*, Bundle Adjustment – A modern synthesis. Workshop on Vision Algorithms, 1999.

Outline

Nonlinear Least Squares
- simple application (motivation)
- linear (approx.) solution and least squares
- normal equations and pseudo-inverse
- LDL^T, QR, and SVD decompositions
- correct linearization and Jacobians
- iterative solution, Levenberg-Marquardt
- robust measurements

Outline

Sparse matrix techniques
- simple application (structure from motion)
- sparse matrix storage (skyline)
- direct solution: LDL^T with minimal fill-in
- larger application (surface/image fitting)
- iterative solution: gradient descent
- conjugate gradient
- preconditioning

Non-linear Least Squares

Triangulation – a simple example

Problem: Given some image points $\{(u_j,v_j)\}$ in *correspondence* across two or more images (taken from calibrated cameras $\mathbf{c}_j$), compute the 3D location $\mathbf{X}$

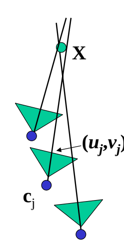
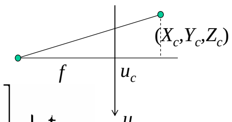


Image formation equations



$$\begin{bmatrix} X_c\\ Y_c\\ Z_c\end{bmatrix} = \left[\,\mathbf{R}\,\right]_{3\times 3}\begin{bmatrix} X\\ Y\\ Z\end{bmatrix} + \mathrm{t}$$

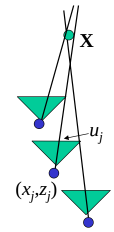
$$\begin{bmatrix} u\\ v\\ 1\end{bmatrix} \sim \begin{bmatrix} U\\ V\\ W\end{bmatrix} = \begin{bmatrix} f & 0 & u_c\\ 0 & f & v_c\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix} X_c\\ Y_c\\ Z_c\end{bmatrix}$$

Simplified model

Let $\boldsymbol{R}=\boldsymbol{I}$ (known rotation), $f=1$, $Y = v_j = 0$ (flatland)

$$u_j = \frac{X - x_j}{Z - z_j}$$

How do we solve this set of equations (constraints) to find the best (X,Z)?



"Linearized" model

Bring the denominator over to the LHS

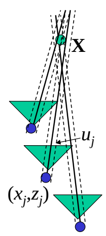
$$u_j(Z - z_j) = X - x_j$$

or

$$X - u_j Z = x_j - u_j z_j$$

(Measures horizontal distance to each line equation.)

How do we solve *this* set of equations (constraints)?



Linear regression

Overconstrained set of linear equations

$$X - u_j Z = x_j - u_j z_j$$

or

$\mathbf{Jx} = \mathbf{r}$

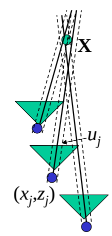
where

$J_{j0}=1,\ J_{j1} = -u_j$

is the *Jacobian* and

$r_j = x_j - u_j z_j$

is the *residual*



Normal Equations

How do we solve $\boldsymbol{Jx} = \boldsymbol{r}$?

Least squares: $\arg\min_x \|Jx\text{-}r\|^2$

$E = \|Jx-r\|^2 = (Jx-r)^T(Jx-r)$

$\quad = x^TJ^TJx - 2x^TJ^Tr - r^Tr$

$\partial E/\partial x = 2(J^TJ)x - 2J^Tr = 0$

$(J^TJ)x = J^Tr$ *normal equations*

$A\, x = b$ " (A is *Hessian*)

$x = [(J^TJ)^{-1}J^T]r$ *pseudoinverse*

LDL^T factorization

Factor $A = LDL^T$, where L is *lower triangular* with 1s on diagonal, D is *diagonal*

How?

L is formed from columns of Gaussian elimination

Perform (similar) forward and backward elimination/substitution

$LDL^Tx = b$, $DL^Tx = L^{-1}b$, $L^Tx = D^{-1}L^{-1}b$,

$x = L^{-T}D^{-1}L^{-1}b$

LDL^T factorization – details

8.2.1 Introduction to Gauss Elimination

We propose to introduce the Gauss solution procedure by studying the solution of the equations $\mathbf{KU} = \mathbf{R}$ derived in Example 3.27 with the parameters $L = 5$, $EI = 1$; i.e.,

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{8.2}$$

In this case the stiffness matrix $\mathbf{K}$ corresponds to a simply supported beam with four translational degrees of freedom, as shown in Fig. 8.1. (We should recall that the equilibrium equations have been derived by finite differences; but, in this case, they have the same properties as in finite element analysis.)

LDL^T factorization – details

Let us first consider the basic mathematical operations of Gauss elimination. We proceed in the following systematic steps:

Step 1: Subtract a multiple of the first equation in (8.2) from the second and third equations to obtain zero elements in the first column of $\mathbf{K}$. This means that $-\frac{4}{5}$ times the first row is subtracted from the second row, and $\frac{1}{5}$ times the first row is subtracted from the third row. The resulting equations are

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ 0 & \frac{14}{5} & -\frac{16}{5} & 1 \\ 0 & -\frac{16}{5} & \frac{29}{5} & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{8.3}$$

Step 2: Considering next the equations in (8.3), subtract $-\frac{16}{14}$ times the second equation from the third equation and $\frac{5}{14}$ times the second equation from the fourth equation. The resulting equations are

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ 0 & \frac{14}{5} & -\frac{16}{5} & 1 \\ 0 & 0 & \frac{15}{7} & -\frac{20}{7} \\ 0 & 0 & -\frac{20}{7} & \frac{65}{14} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ \frac{8}{7} \\ -\frac{5}{14} \end{bmatrix} \tag{8.4}$$

LDL^T factorization – details

Step 3: Subtract $-\frac{20}{15}$ times the third equation from the fourth equation in (8.4). This gives

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ 0 & \frac{14}{5} & -\frac{16}{5} & 1 \\ 0 & 0 & \frac{15}{7} & -\frac{20}{7} \\ 0 & 0 & 0 & \frac{5}{6} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ \frac{8}{7} \\ \frac{7}{6} \end{bmatrix} \tag{8.5}$$

LDL^T factorization – details

Using (8.5), we can now simply solve for the unknowns U_4, U_3, U_2, and U_1:

$$U_4 = \frac{\frac{7}{6}}{\frac{5}{6}} = \frac{7}{5}; \qquad U_3 = \frac{\frac{8}{7} - (-\frac{20}{7})U_4}{\frac{15}{7}} = \frac{12}{5}$$
$$U_2 = \frac{1 - (-\frac{16}{5})U_3 - (1)U_4}{\frac{14}{5}} = \frac{13}{5} \tag{8.6}$$
$$U_1 = \frac{0 - (-4)\frac{13}{5} - (1)\frac{12}{5} - (0)\frac{7}{5}}{5} = \frac{8}{5}$$

The procedure in the solution is therefore to subtract in step number i in succession multiples of equation i from equations $i + 1, i + 2, \ldots, n$, where $i = 1, 2, \ldots, n - 1$. In this way the coefficient matrix $\mathbf{K}$ of the equations is reduced to upper triangular form, i.e., a form in which all elements below the diagonal elements are zero. Starting with the last equation, it is then possible to solve for all unknowns in the order $U_n, U_{n-1}, \ldots, U_1$.

It is important to note that at the end of step i the lower right submatrix of order $n - i$ [indicated by dashed lines in (8.3) to (8.5)] is symmetric. Therefore, the elements above and including the diagonal can give all elements of the coefficient matrix at all times of the solution. We will see in Section (8.2.3) that in the computer implementation we work with only the upper triangular part of the matrix.

LDL^T factorization – details

8.2.2 The LDL^T Solution

We have seen in the preceding section that the basic procedure of the Gauss elimination solution is to reduce the equations to correspond to an upper triangular coefficient matrix from which the unknown displacements $\mathbf{U}$ can be calculated by a back-substitution. We now want to formalize the solution procedure using appropriate matrix operations. An additional important purpose of the discussion is to introduce a notation that can be used throughout the following presentations. The actual computer implementation is given in the next section.

Considering the operations performed in the Gauss elimination solution presented in the preceding section, the reduction of the stiffness matrix $\mathbf{K}$ to upper triangular form can be written

$$\mathbf{L}_{n-1}^{-1} \ldots \mathbf{L}_2^{-1}\mathbf{L}_1^{-1}\mathbf{K} = \mathbf{S} \tag{8.10}$$

LDL^T factorization – details

where $\mathbf{S}$ is the final upper triangular matrix and

$$\mathbf{L}_i^{-1} = \begin{bmatrix} 1 & & & & & \text{elements not shown are zeros} \\ & \ddots & & & & \\ & & 1 & & & \\ & & -l_{i+1,i} & \ddots & & \\ & & -l_{i+2,i} & & \ddots & \\ & & \vdots & & & \\ & & -l_{ni} & & & 1 \end{bmatrix}; \qquad l_{i+j,i} = \frac{k_{i+j,i}^{(i)}}{k_{ii}^{(i)}} \tag{8.11}$$

The elements $l_{i+j,i}$ are the Gauss multiplying factors, and the right superscript (i) indicates that an element of the matrix $\mathbf{L}_{i-1}^{-1} \ldots \mathbf{L}_2^{-1}\mathbf{L}_1^{-1}\mathbf{K}$ is used.

LDL^T factorization – details

We now note that $\mathbf{L}_i$ is obtained by simply reversing the signs of the off-diagonal elements in $\mathbf{L}_i^{-1}$. Therefore, we obtain

$$\mathbf{K} = \mathbf{L}_1\mathbf{L}_2 \ldots \mathbf{L}_{n-1}\mathbf{S} \qquad (8.12)$$

where

$$\mathbf{L}_i = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & l_{i+1,i} & \ddots & & & \\ & & l_{i+2,i} & & \ddots & & \\ & & \vdots & & & \ddots & \\ & & l_{ni} & & & & 1 \end{bmatrix} \qquad (8.13)$$

LDL^T factorization – details

Hence, we can write

$$\mathbf{K} = \mathbf{LS} \qquad (8.14)$$

where $\mathbf{L} = \mathbf{L}_1\mathbf{L}_2 \ldots \mathbf{L}_{n-1}$; i.e., $\mathbf{L}$ is a lower unit triangular matrix,

$$\mathbf{L} = \begin{bmatrix} 1 & & & & & \\ l_{21} & 1 & & & & \\ l_{31} & l_{32} & 1 & & & \\ l_{41} & l_{42} & & 1 & & \\ \cdot & \cdot & & & \ddots & \\ \cdot & \cdot & & & & 1 \\ l_{n1} & \cdot & \cdot & \cdot & l_{n,n-1} & 1 \end{bmatrix} \qquad (8.15)$$

LDL^T factorization – details

elements of $\mathbf{S}$; i.e., $d_{ii} = s_{ii}$. Substituting for $\mathbf{S}$ into (8.14) and noting that $\mathbf{K}$ is symmetric and the decomposition is unique, we obtain $\tilde{\mathbf{S}} = \mathbf{L}^T$, and hence,

$$\mathbf{K} = \mathbf{LDL}^T \qquad (8.16)$$

It is this $\mathbf{LDL}^T$ decomposition of $\mathbf{K}$ that can be used effectively to obtain the solution of the equations in (8.1) in the following two steps:

$$\mathbf{LV} = \mathbf{R} \qquad (8.17)$$

$$\mathbf{DL}^T\mathbf{U} = \mathbf{V} \qquad (8.18)$$

where in (8.17) the load vector $\mathbf{R}$ is reduced to obtain $\mathbf{V}$,

$$\mathbf{V} = \mathbf{L}_{n-1}^{-1} \ldots \mathbf{L}_2^{-1}\mathbf{L}_1^{-1}\mathbf{R} \qquad (8.19)$$

and in (8.18) the solution $\mathbf{U}$ is obtained by a back-substitution,

$$\mathbf{L}^T\mathbf{U} = \mathbf{D}^{-1}\mathbf{V} \qquad (8.20)$$

In the implementation the vector $\mathbf{V}$ is frequently calculated at the same time as the matrices $\mathbf{L}_i^{-1}$ are established. This was done in the example solution of the simply supported beam in Section 8.2.1.

It should be noted that in practice the matrix multiplications to obtain $\mathbf{L}$ in (8.15) and $\mathbf{V}$ in (8.19) are not formally carried out, but that $\mathbf{L}$ and $\mathbf{V}$ are established by directly modifying $\mathbf{K}$ and $\mathbf{R}$. This is discussed further in the next section, in which the computer implementation of the solution procedure is presented. However, before proceeding, consider the example in Section 8.2.1 for the derivation of the matrices defined above.

LDL^T factorization – details

EXAMPLE 8.3: Establish the matrices $\mathbf{L}_i^{-1}$, $i = 1, 2, 3$, $\mathbf{S}$, $\mathbf{L}$, and $\mathbf{D}$ and the vector $\mathbf{V}$ corresponding to the stiffness matrix and the load vector of the simply supported beam treated in Section 8.2.1.

Using the information given in Section 8.2.1, we can directly write down the required matrices:

$$\mathbf{L}_1^{-1} = \begin{bmatrix} 1 & & & \\ \frac{4}{5} & 1 & & \\ -\frac{1}{5} & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix}; \qquad \mathbf{L}_2^{-1} = \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & \frac{8}{7} & 1 & \\ 0 & -\frac{5}{14} & 0 & 1 \end{bmatrix}$$

$$\mathbf{L}_3^{-1} = \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & \frac{4}{3} & 1 \end{bmatrix}; \qquad \mathbf{S} = \begin{bmatrix} 5 & -4 & 1 & 0 \\ & \frac{14}{5} & -\frac{16}{5} & 1 \\ & & \frac{15}{7} & -\frac{20}{7} \\ & & & \frac{5}{6} \end{bmatrix}$$

where the matrix $\mathbf{L}_i^{-1}$ stores in the ith column the multipliers that were used in the elimination of the ith equation and the matrix $\mathbf{S}$ is the upper triangular matrix obtained in (8.5). The matrix $\mathbf{D}$ is a diagonal matrix with the pivot elements on its diagonal. In this case,

$$\mathbf{D} = \begin{bmatrix} 5 & & & \\ & \frac{14}{5} & & \\ & & \frac{15}{7} & \\ & & & \frac{5}{6} \end{bmatrix}$$

LDL^T factorization – details

To obtain $\mathbf{L}$ we use (8.15); hence,

$$\mathbf{L} = \begin{bmatrix} 1 & & & \\ -\frac{4}{5} & 1 & & \\ \frac{1}{5} & -\frac{8}{7} & 1 & \\ 0 & \frac{5}{14} & -\frac{4}{3} & 1 \end{bmatrix}$$

and we can check that $\mathbf{S} = \mathbf{DL}^T$.

The vector $\mathbf{V}$ was obtained in (8.5):

$$\mathbf{V} = \begin{bmatrix} 0 \\ 1 \\ \frac{8}{7} \\ \frac{7}{6} \end{bmatrix}$$

LDL^T and Cholesky

- Variant: *Cholesky*: $A = GG^T$, where $G = LD^{1/2}$ (involves scalar √)
- Advantages: more stable than Gaussian elimination
- Disadvantage: less stable than QR: (cond. #)^2
- Complexity: $(m+n/3)n^2$ flops

QR decomposition

Alternative solution for $Jx = r$

Find an *orthogonal* matrix Q s.t.

$J = Q R$, where R is *upper triangular*

$Q R x = r$

$R x = Q^T r$ solve for x using back subst.

Q is usu. computed using *Householder matrices*, $Q = Q_1 \ldots Q_m$, $Q_j = I - \beta v_j v_j^T$

Advantages: sensitivity $\propto$ condition number

Complexity: $2n^2(m-n/3)$ flops

SVD

Most stable way to solve system Jx = r.

$J = U^T \Sigma V$, where U and V are orthogonal

Σ is diagonal (singular values)

Advantage: most stable (very ill conditioned problems)

Disadvantage: slowest (iterative solution)

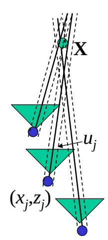
"Linearized" model – revisited

Does the "linearized" model

$$X - u_j Z = x_j - u_j z_j$$

which measures horizontal distance to each line give the *optimal* estimate?

No!



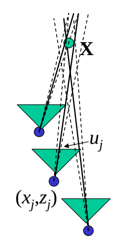
Properly weighted model

We want to minimize errors in the *measured* quantities

$$u_j = \frac{X - x_j}{Z - z_j}$$

Closer cameras (smaller denominators) have more weight / influence.

Weight each "linearized" equation by current denominator?



Optimal estimation

Feature measurement equations

$$u_j = f(X, Z; x_j, z_j) + n_j = \hat{u}_j + n_j, \quad n_j \sim N(0, \sigma_j^2)$$

Likelihood of (X,Z) given $\{u_i,x_j,z_j\}$

$$L = \prod_j p(u_j|\hat{u}_j) = \prod_j e^{-(u_j-\hat{u}_j)^2/\sigma_j^2}$$

Non-linear least squares

Log likelihood of (x,z) given $\{u_i,x_j,z_j\}$

$$E = -\log L = \sum_j (u_j - \hat{u}_j)^2 / \sigma_j^2$$

How do we minimize *E*?

Non-linear regression (least squares), because $\hat{u}_i$ are non-linear functions of $\{u_i,x_j,z_j\}$

Levenberg-Marquardt

Iterative non-linear least squares

- Linearize measurement equations

$$\hat{u}_j = f(X, Z; x_j, z_j) + \frac{\partial f_j}{\partial X}\Delta X + \frac{\partial f_j}{\partial Z}\Delta Z + \cdots$$

- Substitute into log-likelihood equation: quadratic cost function in (Δ*x*,Δ*z*)

$$\sum_j \sigma_j^{-2}(\hat{u}_j - u_j + \frac{\partial f_j}{\partial X}\Delta X + \frac{\partial f_j}{\partial Z}\Delta Z)^2$$

Levenberg-Marquardt

Linear regression (sub-)problem:

$$\mathbf{J}_j \cdot (\Delta X, \Delta Z) = r_j$$

with

$$\mathbf{J}_j = \sigma^{-2}(\frac{\partial f_j}{\partial X}, \frac{\partial f_j}{\partial Z}),$$

$$= \frac{\sigma^{-2}}{Z - z_j}(1, \frac{X - x_j}{Z - z_j}) \quad \hat{u}_i$$

$$r_j = \sigma^{-2}(u_j - \hat{u}_j)$$

Similar to *weighted* regression, but not quite.

Levenberg-Marquardt

What if it doesn't converge?

- Multiply diagonal by (1 + λ), increase λ until it does
- Halve the step size (my favorite)
- Use line search
- Other *trust region* methods [Nocedal & Wright]

Levenberg-Marquardt

Other issues:

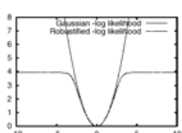
- Uncertainty analysis: covariance $\Sigma = A^{-1}$
- Is *maximum* likelihood the best idea?
- How to start in vicinity of global minimum?
- What about outliers?

Robust regression

Data often have *outliers* (bad measurements)

- Use robust penalty applied to each set of joint measurements

$$\sum_i \sigma_i^{-2}\rho(u_i - \hat{u}_i)$$



  [Black & Rangarajan, IJCV'96]
- For extremely bad data, use random sampling [RANSAC, Fischler & Bolles, CACM'81]

Sparse Matrix Techniques

Direct methods

Structure from motion

Given many points in *correspondence* across several images, $\{(u_{ij},v_{ij})\}$, *simultaneously* compute the 3D location $\mathbf{X}_i$ and camera (or *motion*) parameters ($\mathbf{K}$, $\mathbf{R}_j$, $\mathbf{t}_j$)

$$\hat{u}_{ij} = f(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$
$$\hat{v}_{ij} = g(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$

Two main variants: calibrated, and uncalibrated (sometimes associated with Euclidean and projective reconstructions)

Bundle Adjustment

Simultaneous adjustment of bundles of rays (photogrammetry)

$$\hat{u}_{ij} = f(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$
$$\hat{v}_{ij} = g(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$

What makes this non-linear minimization hard?

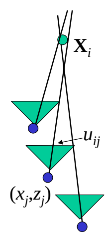
- many more parameters: potentially slow
- poorer conditioning (high correlation)
- potentially lots of outliers
- gauge (coordinate) freedom

Simplified model

Again, $\mathbf{R}=\mathbf{I}$ (known rotation), f=1, Z = v_j = 0 (flatland)

$$u_{ij} = \frac{X_i - x_j}{Z_i - z_j}$$

This time, we have to solve for *all* of the parameters $\{(X_i,Z_i), (x_j,z_j)\}$.

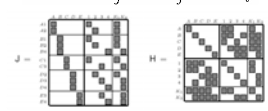


Lots of parameters: sparsity

$$\hat{u}_{ij} = f(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$
$$\hat{v}_{ij} = g(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$

Only a few entries in Jacobian are non-zero

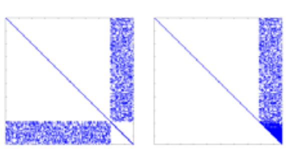
$$\frac{\partial \hat{u}_{ij}}{\partial \mathrm{K}}, \frac{\partial \hat{u}_{ij}}{\partial \mathrm{R}_j}, \frac{\partial \hat{u}_{ij}}{\partial \mathrm{t}_j}, \frac{\partial \hat{u}_{ij}}{\partial \mathrm{x}_i},$$



Sparse LDL^T / Cholesky

First used in finite element analysis [Bathe…]

Applied to SfM by [Szeliski & Kang 1994]

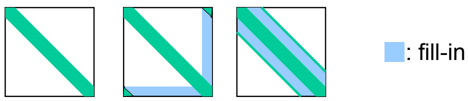


structure | motion fill-in

Skyline storage [Bathe & Wilson]

Sparse matrices–common shapes

Banded (tridiagonal), arrowhead, multi-banded



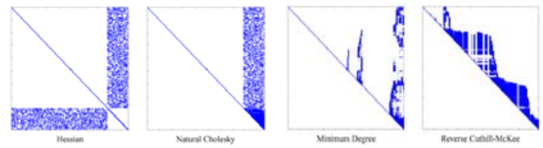
Computational complexity: $O(n\ b^2)$

Application to computer vision:

- snakes (tri-diagonal)
- surface interpolation (multi-banded)
- deformable models (sparse)

Sparse matrices – variable reordering

Triggs *et al.* – Bundle Adjustment



Sparse Matrix Techniques

Iterative methods

Two-dimensional problems

Surface interpolation and Poisson blending

Poisson blending

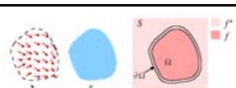


Figure 1: **Guided interpolation notations.** Unknown function f interpolates in domain Ω the destination function f^*, under guidance of vector field $\mathbf{v}$, which might be or not the gradient field of a source function g.

A *guidance field* is a vector field $\mathbf{v}$ used in an extended version of the minimization problem (1) above:

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}, \qquad (3)$$

whose solution is the unique solution of the following Poisson equation with Dirichlet boundary conditions:

$$\Delta f = \operatorname{div}\mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}, \qquad (4)$$

where $\operatorname{div}\mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$ is the divergence of $\mathbf{v} = (u, v)$. This is the fundamental machinery of Poisson editing of color images: three

Poisson blending

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega},$$

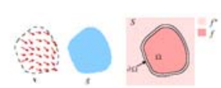


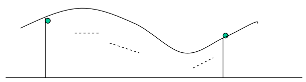
Figure 1: **Guided interpolation notations.** Unknown function f interpolates in domain Ω the destination function f^*, under guidance of vector field $\mathbf{v}$, which might be or not the gradient field of a source function g.

$$E(f) = \sum_i w_{ij}(f_{ij} - g_{ij})^2 + s_{ij}(f_{i+1,j} - f_{ij} - h_i)^2 + t_{ij}(f_{i,j+1} - f_{ij} - v_i)^2$$

→ multi-banded (sparse) system

One-dimensional example

Simplified 1-D height/slope interpolation

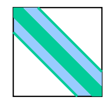


$$
\begin{aligned}
E(f) &= \sum_i w_i (f_i - g_i)^2 + v_i (f_{i+1} - f_i - h_i)^2 \\
A_{i,i} &= w_i + 2v_i, \quad A_{i,i\pm1} = -v_i \\
b_i &= w_i g_i + v_i (h_{i+1} - h_i) v_{i-1} (h_i - h_{i-1})
\end{aligned}
$$

tri-diagonal system (generalized snakes)

Direct solution of 2D problems

Multi-banded Hessian



: fill-in

Computational complexity: n x m image

$O(nm\,m^2)$

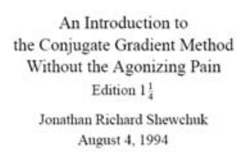
… too slow!

Iterative techniques

Gauss-Seidel and Jacobi
Gradient descent
Conjugate gradient
Non-linear conjugate gradient
Preconditioning

… see Shewchuck's TR

Conjugate gradient



An Introduction to
the Conjugate Gradient Method
Without the Agonizing Pain
Edition $1\frac{1}{4}$

Jonathan Richard Shewchuk
August 4, 1994

… see Shewchuck's TR for rest of notes …

Iterative vs. direct

Direct better for 1D problems and relatively sparse general structures

- SfM where #points >> #frames

Iterative better for 2D problems

- More amenable to parallel (GPU?) implementation
- Preconditioning helps a lot (next lecture)

Monday's lecture (Applications)

Preconditioning

- Hierarchical basis functions (wavelets)
- 2D applications:
  interpolation, shape-from-shading,
  HDR, Poisson blending,
  others (rotoscoping?)

Monday's lecture (Applications)

Structure from motion

- Alternative parameterizations (object-centered)
- Conditioning and linearization problems
- Ambiguities and uncertainties
- New research: *map correlation*