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. 895-717 (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.

Outline

Nonlinear Least Squares
- simple application (motivation)
- linear (approx.) solution and least squares
- normal equations and pseudo-inverse
- LDLᵀ, 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: LDLᵀ with minimal fill-in
- larger application (surface/image fitting)
- iterative solution: gradient descent
- conjugate gradient
- preconditioning

Triangulation – a simple example

Problem: Given some image points \((u_j, v_j)\) in correspondence across two or more images (taken from calibrated cameras \(c_j\)), compute the 3D location \(X\).
Image formation equations

\[
\begin{bmatrix}
X_c \\
Y_c \\
Z_c
\end{bmatrix}
= 
\begin{bmatrix}
R \\
1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
+ t
\]

Simplified model

Let \( R - I \) (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

\[
X - u_jZ = x_j - u_jz_j
\]

(Measures horizontal distance to each line equation.)

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

Linear regression

Overconstrained set of linear equations

\[
X - u_jZ = x_j - u_jz_j
\]

or

\[
Jx = r
\]

where

\[
J_j^0 = 1, \quad J_j^1 = -u_j
\]

is the Jacobian and

\[
r_j = x - u_jz_j
\]

is the residual

Normal Equations

How do we solve \( Jx = r \)?

Least squares:

\[
\arg \min_x \| Jx - r \|^2
\]

\[
E = \| Jx - r \|^2 = (Jx - r)^T(Jx - r) = x^TJ^TJx - 2x^TJr + r^Tr
\]

\[
\frac{\partial E}{\partial x} = 2(J^TJ)x - 2J^Tr = 0
\]

\[
(J^TJ)x = J^Tr \quad \text{normal equations}
\]

\[
A x = b
\]

\[
x = [J^TJ]^{-1}J^T r \quad \text{pseudoinverse}
\]

LDLT 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

\[
LDLTx = b, \quad DL^Tx = L^{-1}b, \quad L^Tx = D^{-1}L^{-1}b, \quad x = L^{-T}D^{-1}L^{-1}b
\]
8.2.1 Introduction to Gauss Elimination

We propose to introduce the Gauss solution procedure by solving the system of equations \( K \mathbf{x} = \mathbf{b} \) derived in Example 3.27 with the parameters \( L = 3, E = 1, \ldots, \) and

\[
\begin{bmatrix}
  3 & -1 & 0 & | & 5 \\
  -1 & 3 & -1 & | & 4 \\
  0 & -1 & 3 & | & 2 \\
  0 & 0 & 0 & | & 1
\end{bmatrix}
\]

In this case the stiffness matrix \( K \) 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.)

4/30/2004 NLS and Sparse Matrix Techniques 15

LDLT factorization – details

4/30/2004 NLS and Sparse Matrix Techniques 16

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 \( U \) can be calculated by back substitution. We shall want to formulate 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.

Following the operations performed in the Gauss elimination solution presented in the preceding section, the solution of the stiffness matrix \( K \) by upper triangular form can be written

\[
L^T \ldots L^T U^T K = 8
\]

4/30/2004 NLS and Sparse Matrix Techniques 17

LDLT factorization – details

4/30/2004 NLS and Sparse Matrix Techniques 18

LDLT 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 equation to eliminate elements in the first column of \( K \). This means that \(-1\) times the first row is subtracted from the second row and \(-1\) times the first row is subtracted from the third row. The resulting equations are

\[
\begin{align*}
  x_1 & - 1 \cdot x_1 = 5 \\
  x_3 & - 1 \cdot x_1 = 4 \\
  x_4 & - 1 \cdot x_1 = 2 \\
  x_4 & - 1 \cdot x_1 = 1
\end{align*}
\]

Step 2: Considering next the equations in (8.3), subtract \(-1\) times the second equation from the third equation and \(-1\) times the second equation from the fourth equation. The resulting equations are

\[
\begin{align*}
  x_1 & - 1 \cdot x_1 = 5 \\
  x_3 & - 1 \cdot x_1 = 4 \\
  x_4 & - 1 \cdot x_1 = 2 \\
  x_4 & - 1 \cdot x_1 = 1
\end{align*}
\]

Using (8.5), we can now simply solve for the unknowns \( u_1, v_1, e_1, \) and \( E_1 \):

\[
\begin{align*}
  u_1 & = 5 - x_1 \\
  v_1 & = 4 - x_1 \\
  e_1 & = 2 - x_1 \\
  E_1 & = 1 - x_1
\end{align*}
\]

The procedure is then to subtract in step number \( (m - i + 1) \) the \( (i - 1) \)th equation from equation \( m \) (from equations \( m = 1, 2, \ldots, n \), where \( m = \frac{1}{2} (n + 1) \)). In this way the coefficient matrix \( K \) 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 top equation (1), then proceeding with the second equation (2) and continuing in this manner, the \( u \), \( v \), \( e \), and \( E \) can be solved for in the order \( u_1, v_1, e_1, E_1 \).

It is important to note that at the end of step \( m \) the upper right submatrix of \( K \) is truncated and replaced with \( \text{diag}(a_{m+1}, a_{m+2}, \ldots, a_{n}) \) in form (8.6). Therefore, the elements above and including the diagonal can give all elements of the coefficient matrix at all times of the solution. We will set in detail (8.7) that in the computer implementation we work with only the upper triangular part of the matrix.
**LDLᵀ factorization – details**

We now note that L, is obtained by simply reversing the signs of the off-diagonal elements in Lᵀ. Therefore, we obtain

\[
K = L A Lᵀ, \quad \text{for} \quad L = \begin{bmatrix}
L_{11} & 0 & 0 & \cdots \\
L_{21} & L_{22} & 0 & \cdots \\
L_{31} & L_{32} & L_{33} & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\]

where

\[
L_{ij} = \begin{cases}
1, & \text{if } i = j \\
-L_{ji}, & \text{if } i > j
\end{cases}
\]

Hence, we can write

\[
K = L Lᵀ
\]

where \( L = L₁, L₂, \ldots, Lₙ \), i.e., \( L \) is a lower unit triangular matrix.

\[
L = \begin{bmatrix}
1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{bmatrix}
\]

**LDLᵀ factorization – details**

Elements of \( \mathbf{K} \) are zero. Substituting for \( \mathbf{K} \) in (1.16) and noting that \( \mathbf{K} \) is symmetric and the decomposition of \( \mathbf{K} \), we obtain \( \mathbf{L} = \mathbf{L}ᵀ \) and hence,

\[
\mathbf{L} = \begin{bmatrix}
L₁ & 0 & \cdots & 0 \\
L₂ & L₂ & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
Lₙ & Lₙ & \cdots & Lₙ
\end{bmatrix}
\]

\[
\mathbf{L} ᴥ \mathbf{L}ᵀ = \mathbf{G} \mathbf{G}ᵀ
\]

\[
\mathbf{L} = \mathbf{G} \mathbf{L}ᵀ
\]

where \( \mathbf{L} \) is the LDLᵀ decomposition of \( \mathbf{K} \), and \( \mathbf{G} \) can be used effectively to obtain the solution of the equations in (0.1) in the following two steps:

\[
\mathbf{L} \mathbf{L}ᵀ \mathbf{V} = \mathbf{K} \quad \text{and } \quad \mathbf{L} \mathbf{V} = \mathbf{L}ᵀ \mathbf{V}
\]

\[
\mathbf{V} = \begin{bmatrix}
\mathbf{V}_₁ \\
\mathbf{V}_₂ \\
\vdots \\
\mathbf{V}_ₙ
\end{bmatrix}
\]

and \( \mathbf{V} \) is the solution of the equations (0.16). The vector \( \mathbf{V} \) is frequently calculated at the same time as the matrix \( \mathbf{L} \). This was done in the example solution of the simple supported beam in Section 9.2.1.

**LDLᵀ factorization – details**

\[
\mathbf{L} \mathbf{L}ᵀ \mathbf{V} = \mathbf{K}
\]

\[
\mathbf{L} \mathbf{V} = \mathbf{L}ᵀ \mathbf{V}
\]

In the implementation of the vector \( \mathbf{V} \), it is frequently calculated at the same time as the matrix \( \mathbf{L} \), as described in Section 9.2.1. The matrix \( \mathbf{L} \) should be stored in a packed form to save storage space.

**LDLᵀ and Cholesky**

**Variant:** Cholesky: \( \mathbf{A} = \mathbf{G} \mathbf{G}ᵀ \), where \( \mathbf{G} = \mathbf{L}^{1/2} \)

This involves scalar \( \sqrt{ } \)

Advantages: more stable than Gaussian elimination

Disadvantage: less stable than QR: \( \text{cond. #}^2 \)

Complexity: \( (m+n/3)n^2 \) flops
**QR decomposition**

Alternative solution for \( Jx = r \)
Find an orthogonal matrix \( Q \) s.t. 
\[ J = QR, \quad \text{where } R \text{ is upper triangular} \]
\[ QRx = r \]
\[ Rx = Q^Tr \]
solve for \( x \) using back subst.
\( Q \) is usu. computed using Householder matrices,
\[ Q = Q_1 \ldots Q_m, \quad Q_j = I - \beta_j v_j v_j^T \]
Advantages: sensitivity \( \propto \) condition number
Complexity: \( 2n^2(m-n/3) \) flops

**SVD**

Most stable way to solve system \( Jx = r \).
\[ J = U^T \Sigma V, \quad \text{where } U \text{ and } V \text{ are orthogonal} \]
\[ \Sigma \text{ is diagonal (singular values)} \]
Advantage: most stable (very ill conditioned problems)
Disadvantage: slowest (iterative solution)

**“Linearized” model – revisited**

Does the “linearized” model
\[ X - u_jZ = x_j - u_j z_j \]
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
\[ u_j = f(X, Z; x_j, z_j) + n_j = \tilde{u}_j + n_j, \quad n_j \sim N(0, \sigma_j^2) \]

Likelihood of \((X,Z)\) given \(\{u_j, x_j, z_j\}\)
\[ L = \prod_j p(u_j | \tilde{u}_j) \]
\[ = \prod_j e^{-(u_j - \tilde{u}_j)^2 / \sigma_j^2} \]

**Non-linear least squares**

Log likelihood of \((x,z)\) given \(\{u_j, x_j, z_j\}\)
\[ E = - \log L = \sum_j (u_j - \tilde{u}_j)^2 / \sigma_j^2 \]

How do we minimize \( E \)?
Non-linear regression (least squares), because \( \tilde{u}_j \) are non-linear functions of \( \{u_j, x_j, z_j\} \)
Levenberg-Marquardt

Iterative non-linear least squares

• Linearize measurement equations
  \[ \tilde{a}_j = f(X, Z; x_j, z_j) + \frac{\partial f_j}{\partial X} \Delta X + \frac{\partial f_j}{\partial Z} \Delta Z + \cdots \]

• Substitute into log-likelihood equation: quadratic cost function in \((\Delta x, \Delta z)\)
  \[ \sum_j \sigma_j^{-2} (\tilde{a}_j - u_j + \frac{\partial f_j}{\partial X} \Delta X + \frac{\partial f_j}{\partial Z} \Delta Z)^2 \]

Levenberg-Marquardt

Linear regression (sub-)problem:

\[ J_j \cdot (\Delta X, \Delta Z) = r_j \]

with

\[ J_j = \sigma^{-2} \begin{pmatrix} \frac{\partial f_j}{\partial X} & \frac{\partial f_j}{\partial Z} \end{pmatrix} \]

\[ r_j = \sigma^{-2} (u_j - \tilde{u}_j) \]

Similar to weighted regression, but not quite.

Levenberg-Marquardt

What if it doesn’t converge?

• Multiply diagonal by \((1 + \lambda)\), increase \(\lambda\) 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
  \[ \sum \sigma_j^{-2} h(x_j - \hat{x}_j) \]

[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 \( X_i \) and camera (or motion) parameters (\( K, R_j, t_j \))

\[
\begin{align*}
\tilde{u}_{ij} &= f(K, R_j, t_j, X_i) \\
\tilde{v}_{ij} &= g(K, R_j, t_j, X_i)
\end{align*}
\]

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

Bundle Adjustment
Simultaneous adjustment of bundles of rays (photogrammetry)

\[
\begin{align*}
\tilde{u}_{ij} &= f(K, R_j, t_j, X_i) \\
\tilde{v}_{ij} &= g(K, R_j, t_j, X_i)
\end{align*}
\]

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, \( R \sim I \) (known rotation), \( f=1, Z = v_j = 0 \) (flatland)

\[
\begin{align*}
\tilde{u}_{ij} &= X_i - x_j \\
\tilde{v}_{ij} &= Z_i - z_j
\end{align*}
\]

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

Lots of parameters: sparsity

\[
\begin{align*}
\tilde{u}_{ij} &= f(K, R_j, t_j, x_i) \\
\tilde{v}_{ij} &= g(K, R_j, t_j, x_i)
\end{align*}
\]

Only a few entries in Jacobian are non-zero

\[
\frac{\partial u_{ij}}{\partial k} \quad \frac{\partial u_{ij}}{\partial R_j} \quad \frac{\partial u_{ij}}{\partial t_j} \quad \frac{\partial u_{ij}}{\partial x_i}
\]

Sparse LDLᵀ / Cholesky
First used in finite element analysis [Bathe…] Applied to SfM by [Szeliski & Kang 1994]

Skyline storage [Bathe & Wilson]
Sparse matrices–common shapes

Banded (tridiagonal), arrowhead, multi-banded

\[ \text{: fill-in} \]

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

Two-dimensional problems

Surface interpolation and Poisson blending

Poisson blending

\[
\begin{align*}
\min \int_{\Omega} (\nabla f - v)^2 \mathrm{d}a + \int_{\partial R} f_{\partial R} = f'_{\partial R} \\
E(f) &= \sum_{i} w_i (f_j - g_j)^2 + \lambda_1 (f_{j+1} - f_j - h_j)^2 + \lambda_2 (f_{j+1} - f_j - v_j)^2 \\
&\rightarrow \text{multi-banded (sparse) system}
\end{align*}
\]
One-dimensional example

Simplified 1-D height/slope interpolation

\[ E(f) = \sum v_i (f_i - g_i)^2 + \frac{1}{\gamma} (f_{i+1} - f_i - h_i)^2 \]

[Diagram: A tri-diagonal system (generalized snakes)]

Direct solution of 2D problems

Multi-banded Hessian

[Diagram: Fill-in]

Computational complexity: \( n \times m \) image

\[ O(nm^2) \]

... 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
Jonathan Richard Shewchuck
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*