Notes

- Note that $r^2\log(r)$ is NaN at r=0:
  instead smoothly extend to be 0 at r=0
- Schedule a make-up lecture?

Review LU

- Write A=LU in block partitioned form
  - By convention, L has all ones on diagonal
- Equate blocks: pick order to compute them
  - "Up-looking": compute a row at a time
    (refer just to entries in A in rows 1 to i)
  - "Left-looking": compute a column at a time
    (refer just to entries in A in columns 1 to j)
  - "Bordering": row of L and column of U
  - "Right-looking": column of L and row of U
    (note: outer-product update of remaining A)
- Can do all of these "in-place" (overwrite A)

Pivoting

- LU and LDL^T can fail
  - Example: if $A_{11}=0$
- Go back to Gaussian Elimination ideas: reorder the equations (rows) to get a nonzero entry
- In fact, nearly zero entries still a problem
  - Perhaps cancellation error => few significant digits
  - Dividing through will taint rest of calculation
- Pivoting: reorder to get biggest entry on diagonal
  - Partial pivoting: just reorder rows (or columns)
  - Complete pivoting: reorder rows **and** columns (expensive)

Row Partial-Pivoting

- Row partial-pivoting: PA=LU
  - Compute a column of L, swap rows to get biggest entry on diagonal
  - Express as PA=LU where P is a permutation matrix
  - P is the identity with rows swapped (but store it as a permutation vector)
  - This is what LAPACK uses
- Guarantees entries of L bounded by 1 in magnitude
- No good guarantee on U – but usually fine
- If U doesn't grow too much, comes very close to optimal accuracy

Symmetric Pivoting

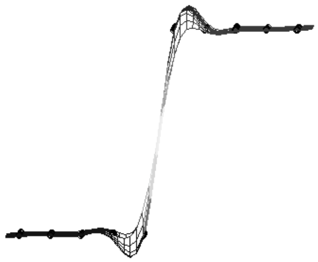
- Problem: partial (or complete) pivoting destroys symmetry
- How can we factor a symmetric indefinite matrix reliably but twice as fast as unsymmetric matrices?
- One idea: symmetric pivoting $PAP^T=LDL^T$
  - Swap the rows the same as the columns
- But let D have 2x2 as well as 1x1 blocks on the diagonal
  - Partial pivoting: Bunch-Kaufman (LAPACK)
  - Complete pivoting: Bunch-Parlett (safer)

Reconsidering RBF

- RBF interpolation has advantages:
  - Mesh-free
  - Optimal in some sense
  - Exponential convergence (each point extra data point improves fit everywhere)
  - Defined everywhere
- But some disadvantages:
  - It's a global calculation
    (even with compactly supported functions)
  - Big dense matrix to form and solve
    (though later we'll revisit that…

Gibbs

- Globally smooth calculation also makes for overshoot/ undershoot (Gibbs phenomena) around discontinuities
- Can't easily control effect



Noise

- If data contains noise (errors), RBF strictly interpolates them
- If the errors aren't spatially correlated, lots of discontinuities: RBF interpolant becomes wiggly

Linear Least Squares

- Idea: instead of interpolating data + noise, approximate
- Pick our approximation from a space of functions we expect (e.g. not wiggly -- maybe low degree polynomials) to filter out the noise
- Standard way of defining it:

$$f(x) = \sum_{i=1}^{k} \lambda_i \phi_i(x)$$

$$\lambda = \arg\min_{\lambda} \sum_{j=1}^{n} \left(f_j - f(x_j) \right)^2$$

Rewriting

- Write it in matrix-vector form:

$$\sum_{i=1}^{n} \left(f_i - \sum_{j=1}^{k} \lambda_j \phi_j(x_i) \right)^2 = \|b - Ax\|_2^2$$

$$b = \begin{pmatrix} f_1 & f_2 & \cdots & f_n \end{pmatrix}^T$$

$$x = \begin{pmatrix} \lambda_1 & \cdots & \lambda_k \end{pmatrix}^T$$

$$A_{ij} = \phi_j(x_i) \quad \text{(a rectangular n} \times \text{k matrix)}$$

Normal Equations

- First attempt at finding minimum: set the gradient equal to zero (called "the normal equations")

$$\frac{\partial}{\partial x} \|b - Ax\|_2^2 = 0$$

$$\frac{\partial}{\partial x} \left((b - Ax)^T (b - Ax) \right) = 0$$

$$\frac{\partial}{\partial x} \left(b^T b - 2x^T A^T b + x^T A^T A x \right) = 0$$

$$-2A^T b + 2A^T A x = 0$$

$$A^T A x = A^T b$$

Normal Equations: Good Stuff

- $A^T A$ is a square $k \times k$ matrix
  (k probably much smaller than n)

  Symmetric positive (semi-)definite

Normal Equations: Problem

- What if k=n?
  At least for 2-norm condition number,
  $\kappa(A^TA)=k(A)^2$
  - Accuracy could be a problem…
- In general, can we avoid squaring the errors?