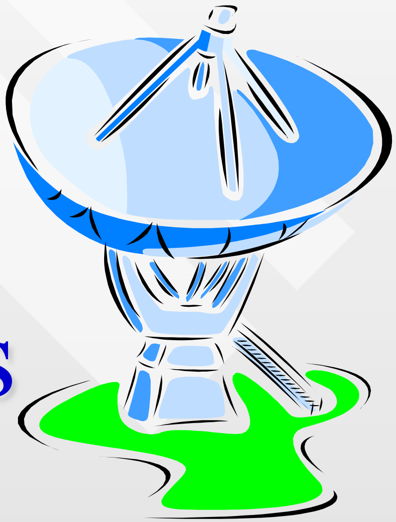
Inverse problems

Concluding Remarks
Lecture 8



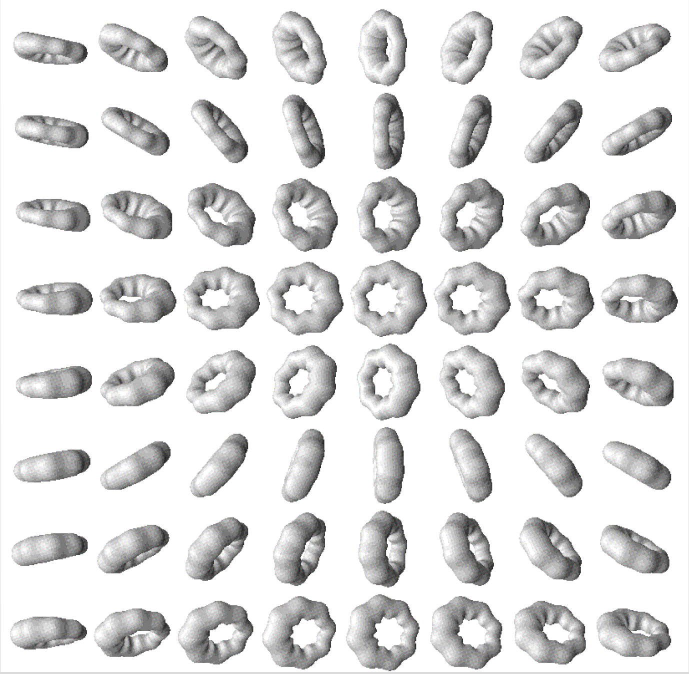
Nikolai Piskunov
2014

We will talk about

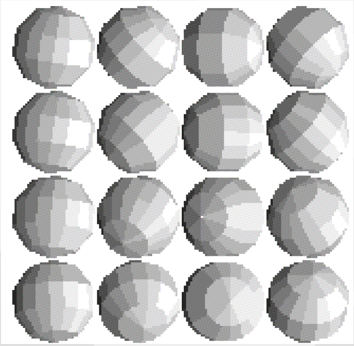
- More examples of inverse problem applications
- Some comments about writing and debugging programs
- Applications to your research areas

3D Objects

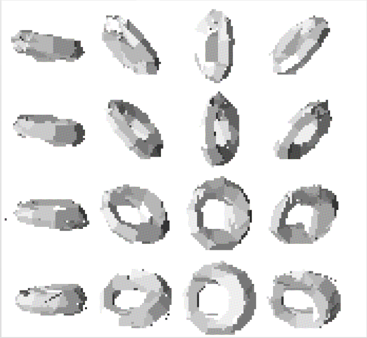
Reconstruction of a 3D shape using sequence of images from different angles or/and from different orientations



Observations



Initial guess



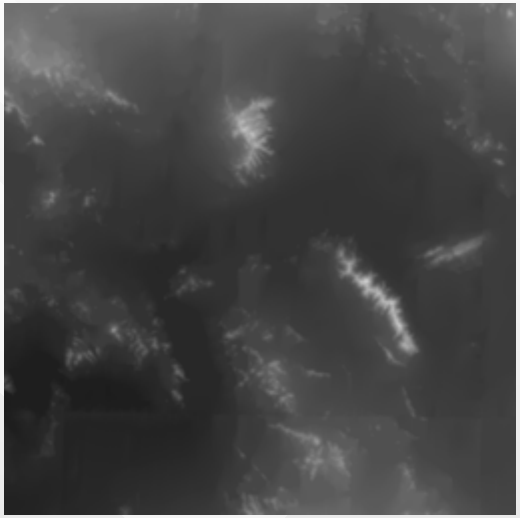
Solution

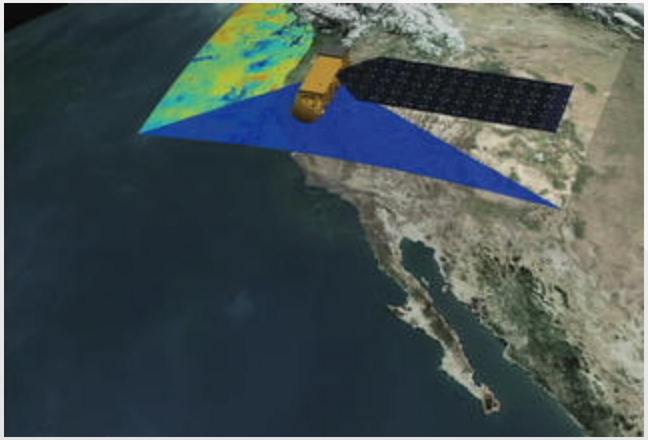
How does one do that?

- Assumption 1: we deal with a surface
- Assumption 2: albedo is the function of angle between the line of sight and the local normal (the same for all surface elements).
- Solution: we setup a dummy surface (topological cylinder), compute its 2D projections and adjust node positions.
- Regularization controls the geometry: e.g. it will allow top and bottom nodes to merge but prevent other nodes from doing so.

Satellite Imaging

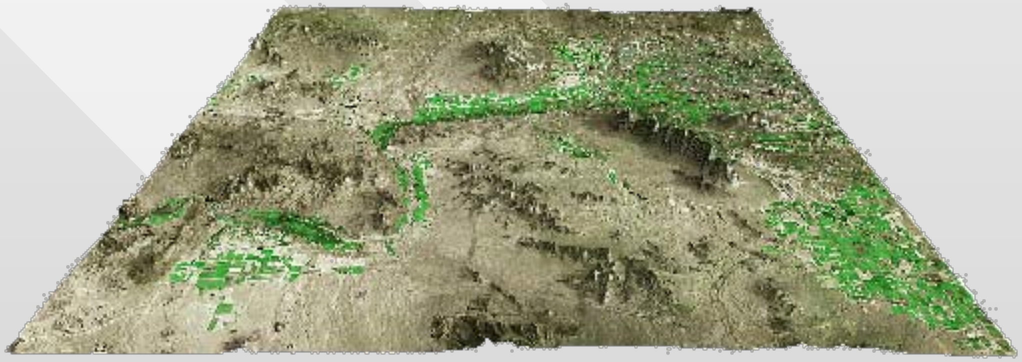
- Raw satellite images
  (here is one from LandSat 7)





- Analysis of image sequences:
  3 color filters × 16 viewing angles

- Processed 3D image



Medical Imaging

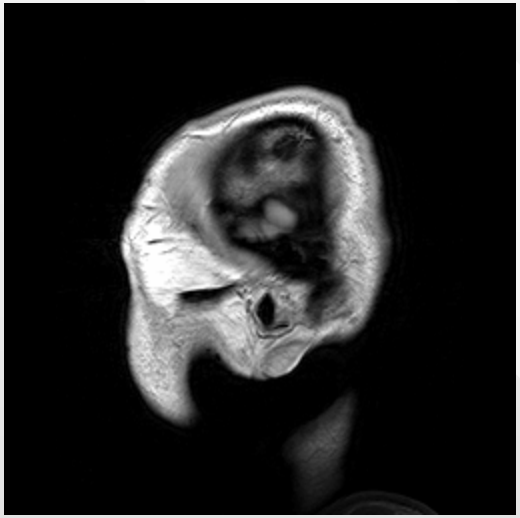
Computer Aided Tomography (CAT scanner): ray scattering:

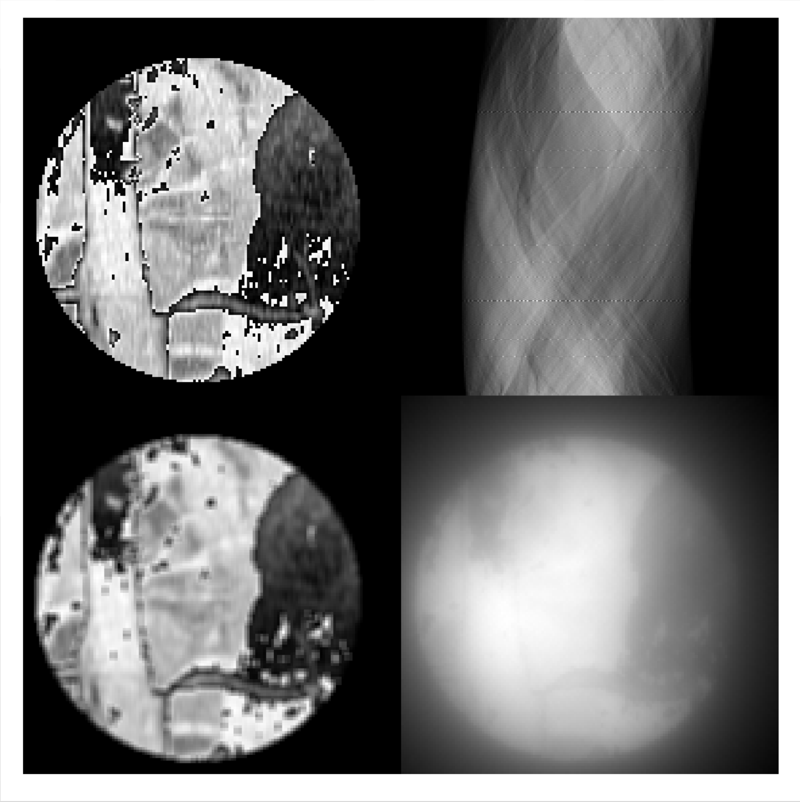
$$I_{\text{det}} = I_0 \cdot e^{-\Delta\tau} + \int_{\text{src}}^{\text{det}} \alpha(t) J(t) e^{-(\tau_{\text{det}} - t)} dt$$

$$\tau(\tau) = \int_{\tau_{\text{src}}}^{\tau} \alpha(x) dx;\ J(\tau) = \sum_{\text{rays}} \int_{\text{src}}^{\text{det}} I_0 e^{-t} e^{-\|t-\tau\|} dt$$

More Medical Imaging

Magnetic Resonance Imaging …

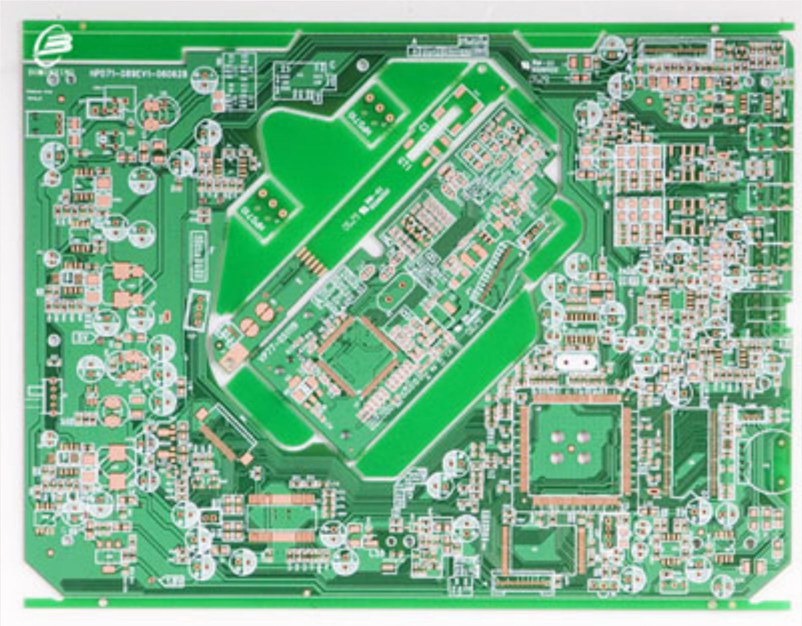




… and CAT again

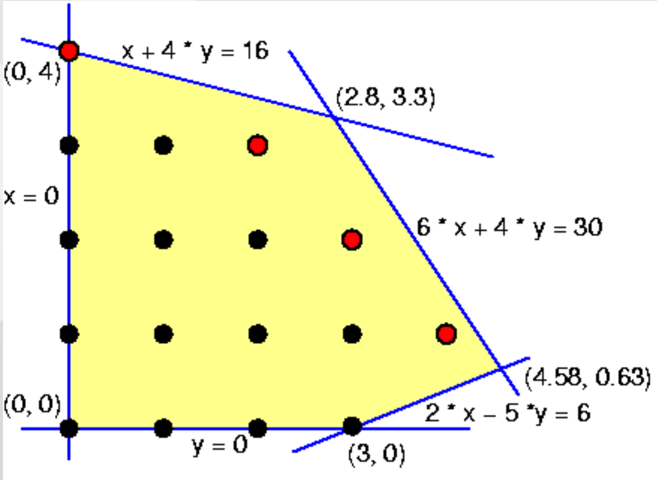
Other applications

- Printed Circuit Board design: from electrical scheme to the actual layout



Regularization: minimum distance between conductors carrying high-frequency

- Optimal control problems: *linear programming*



Programming style

- Structure:
  - Loops
  - Using subroutines/functions
- Precision: range of omega variation is more than 10^6 ⇨ double precision is needed
- Vectorization:
  R=0. & for i=1,n-1 do R=R+(f[i]-f[i-1])^2
  is many times slower than
  R=total((f[1:n-1]-f[0:n-2])^2)
- Convergence conditions: for something that is positive (omega) relative change is better

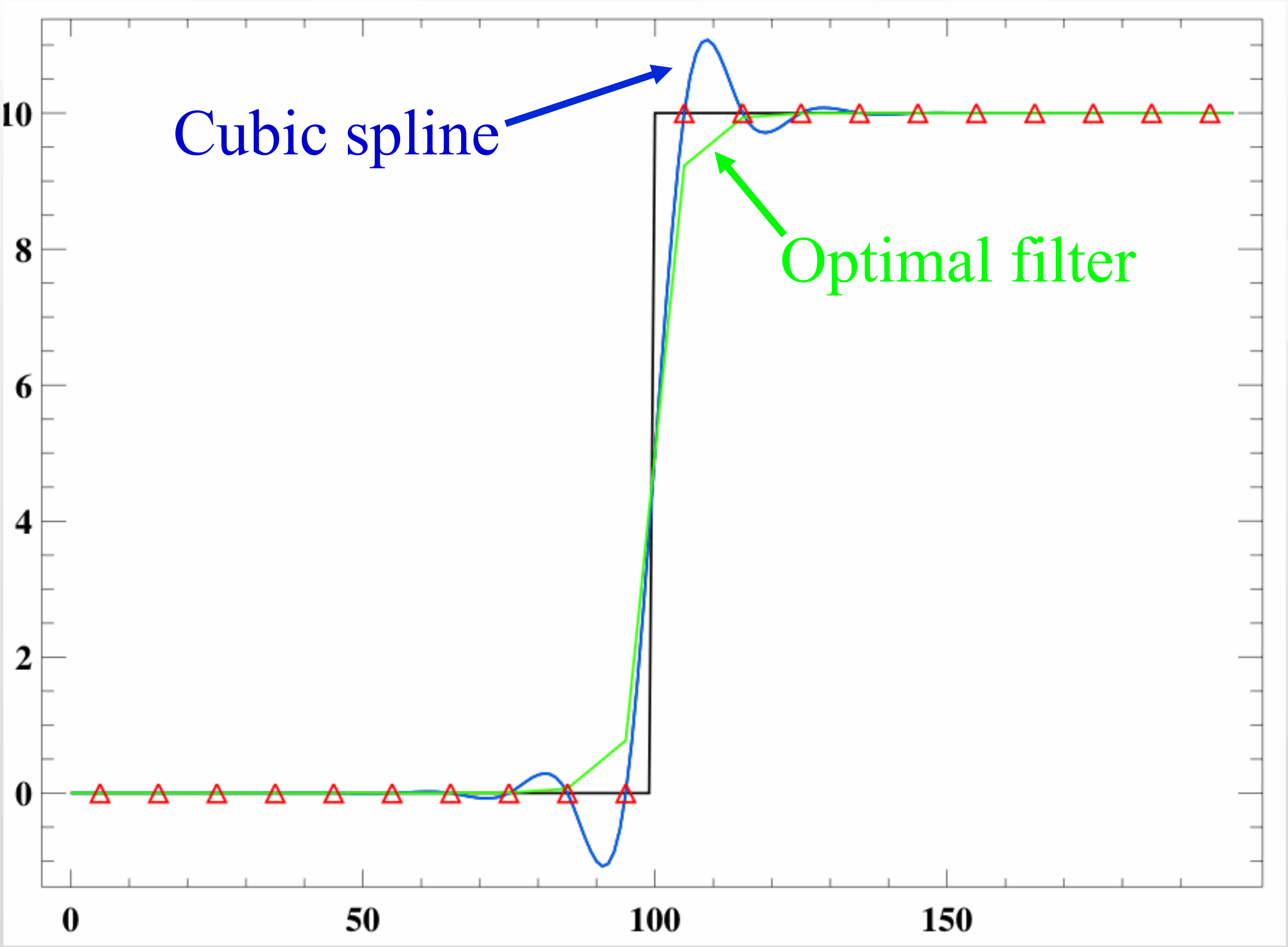
Debugging:

Think before acting!

- Debug building blocks before the whole code
- Never rely on the results that “appear” to be correct
- When debugging interfaces between building blocks think of proper tests and run them
- Example: test your derivative calculations using your function. It’s easy!

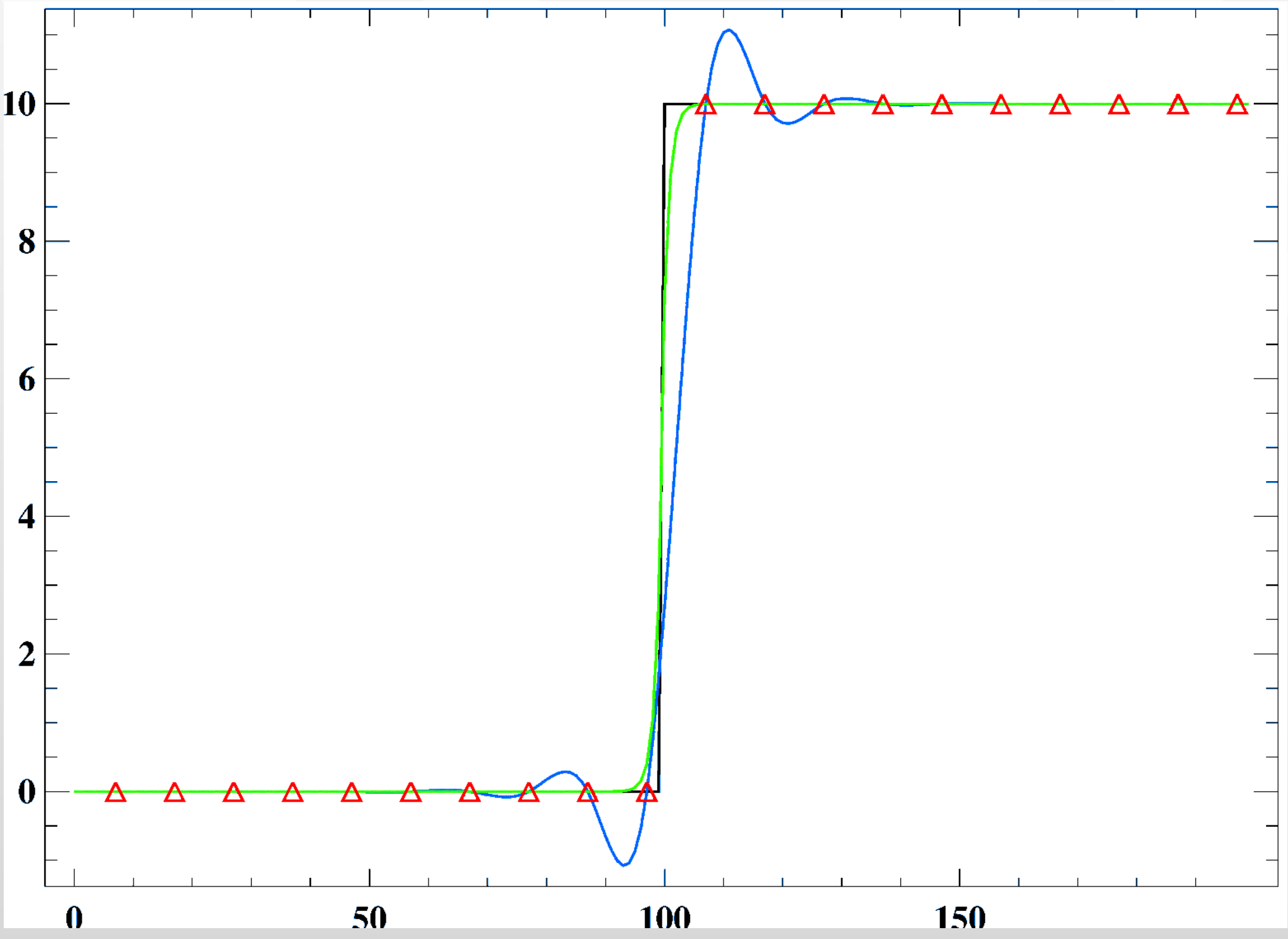
Efficient in-depth debugging saves time/lives!

Optimal filter example

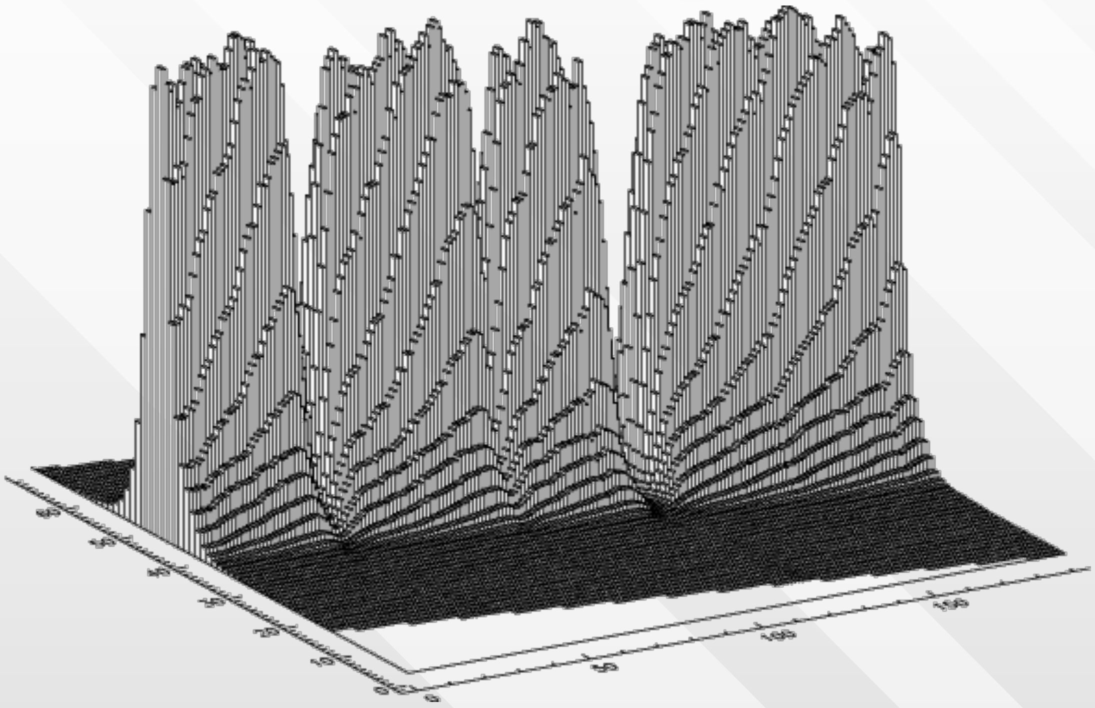


Actually, one can do better…

$$\Omega = \sum_i \left(f_{i\cdot m} - g_i \right)^2 + \Lambda_1 \cdot \sum_{l=1}^{N-1} \left(f_l - f_{l-1} \right)^2 + \Lambda_2 \cdot \sum_{l=1}^{N-2} \left(f_{l+1} - 2f_l + f_{l-1} \right)^2$$



Slit decomposition



Assumption: spectral order is a bunch of monochromatic 1D images of the spectrometer input slit

Slit decomposition math

- 2D fragment of a spectral order can be described as:

$$S_{x,y} = P_x \cdot \sum_i \omega_{x,y}^{y'} L(y' - y_{\mathrm{c}}(x))$$

- Weights show what parts of the slit function fall on pixel *x,y*
- The inverse problem is:

$$\Omega(P, L) \equiv \sum_{x,y} \| P_x \cdot \sum_i \omega_{x,y}^{y'} L(y' - y_{\mathrm{c}}(x)) - S_{x,y} \|^2 + \Lambda \mathbf{R}(L) = \min$$

Slit decomposition math

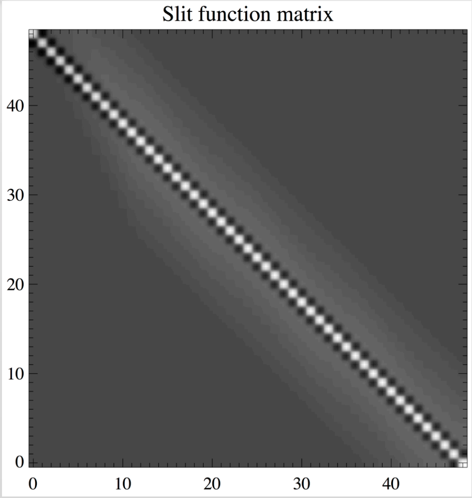
- Assuming to know ***L***, we can compute ***P***:

$$P_x \sum_y \left[\sum_i \omega_{x,y}^{y'} L(y' - y_{\mathrm{c}}(x)) \right]^2 = \sum_y S_{x,y} \sum_i \omega_{x,y}^{y'} L(y' - y_{\mathrm{c}}(x))$$

- Assuming to know ***P***, we can compute ***L***:

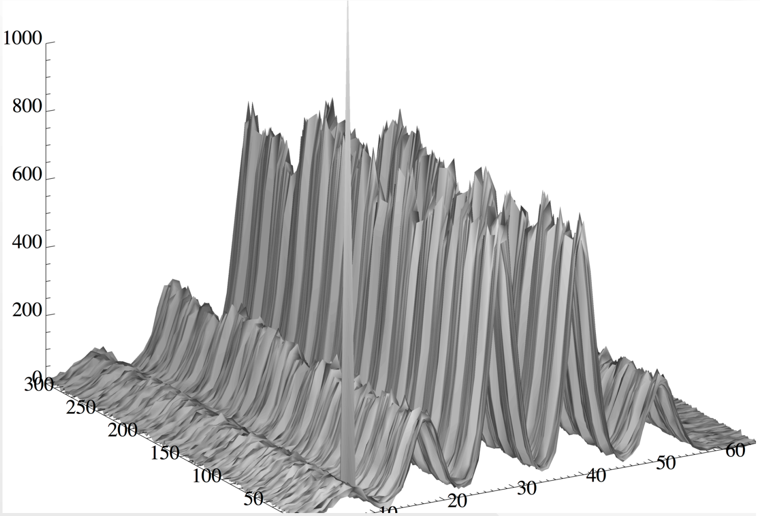
$$\mathbf{A} \cdot L = B$$

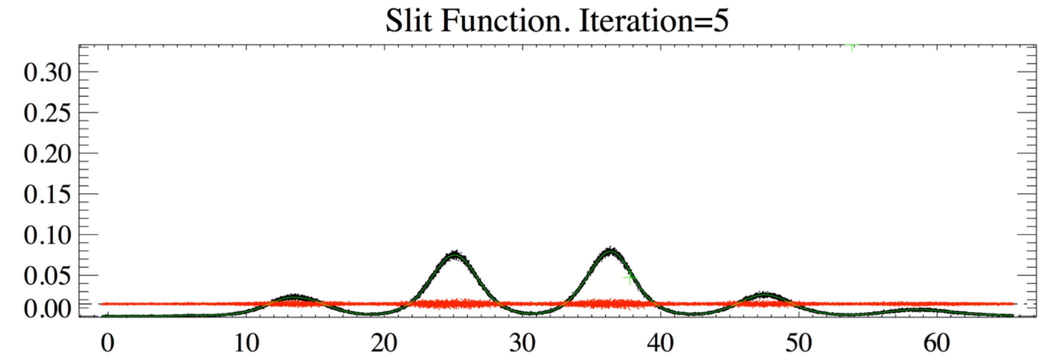
where the matrix looks like this:



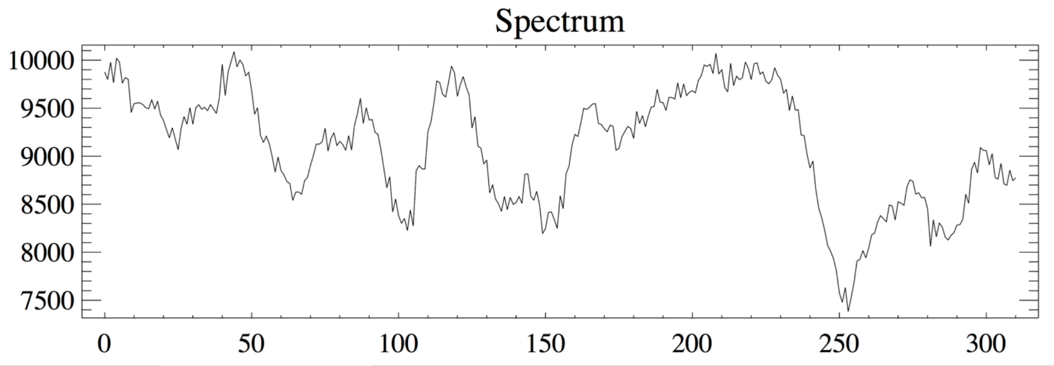
Slit decomposition solution

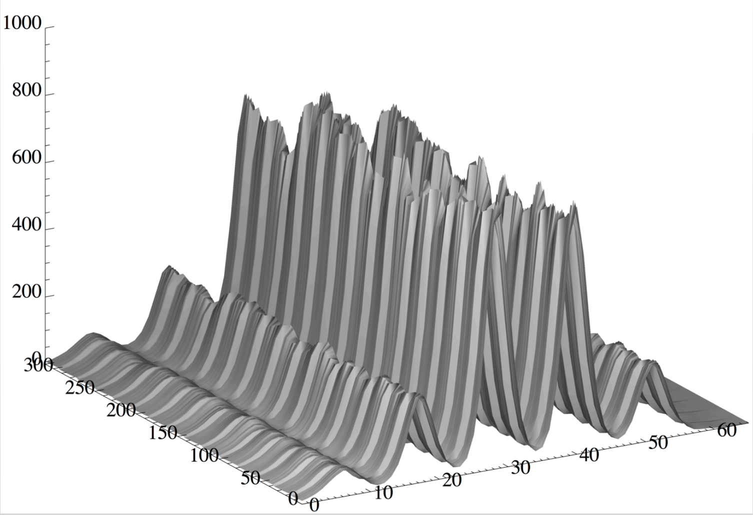
Input





Decomposition





Output

Applications to your research

- Please, tell us …