Opinionated Lessons in Statistics

by Bill Press

#45 Laplace Interpolation

Laplace Interpolation is a specialized interpolation method for restoring missing data on a grid. It's simple, but sometimes works astonishingly well.

Mean value theorem for solutions of Laplace's equation (harmonic functions):

If y satisfies $\nabla^2 y = 0$ in any number of dimensions, then for any sphere not intersecting a boundary condition,

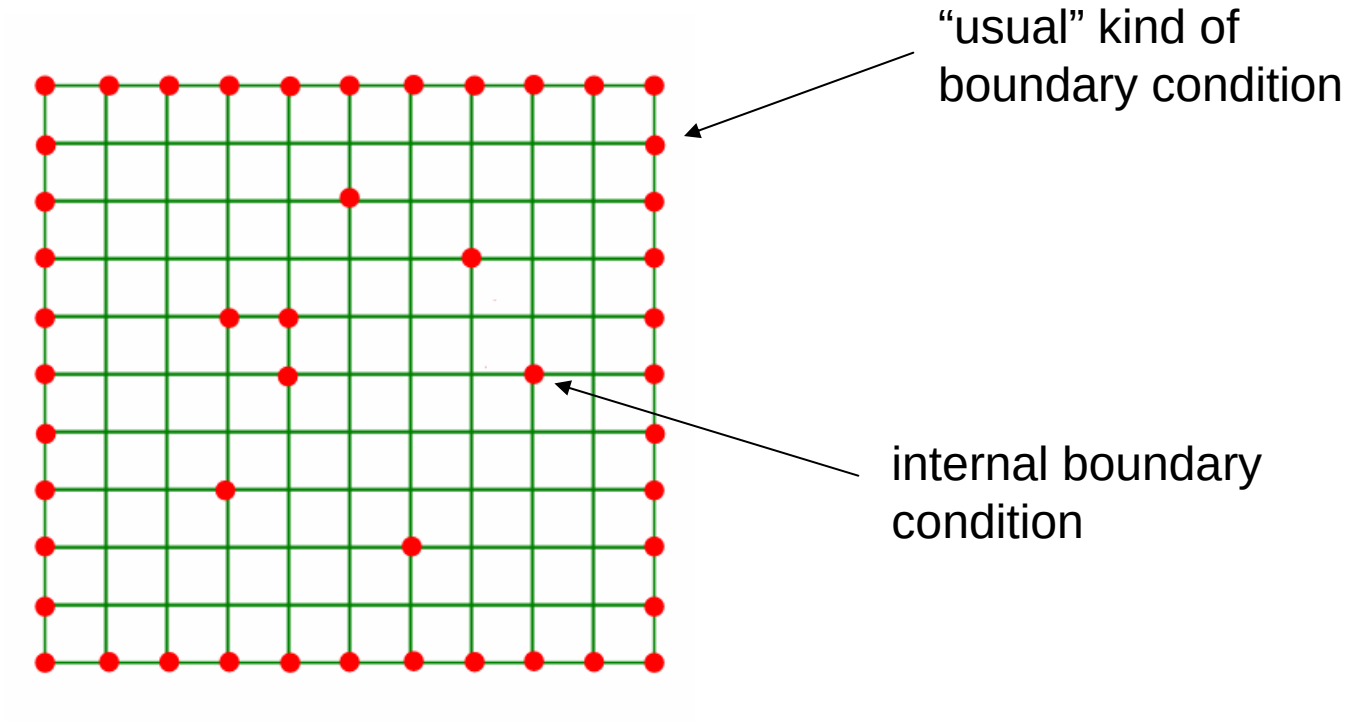
$$\frac{1}{\text{area}} \int_{\text{surface } \omega} y \, d\omega = y(\text{center})$$

So Laplace's equation is, in some sense, the perfect interpolator. It also turns out to be the one that minimizes the integrated square of the gradient,

$$\int_{\Omega} |\nabla y|^2 \, d\Omega$$

So the basic idea of Laplace interpolation is to set $y(\mathbf{x}_i) = y_i$ at every known data point, and solve $\nabla^2 y = 0$ at every unknown point.

You may not be used to thinking of Laplace's equation as allowing isolated internal boundary conditions. But it of course does!



values are fixed on red dots

Lots of linear equations (one for each grid point)!

$$y_0 = y_{0(\text{measured})}$$

generic equation for a known point

$$y_0 - \tfrac{1}{4}y_u - \tfrac{1}{4}y_d - \tfrac{1}{4}y_l - \tfrac{1}{4}y_r = 0$$

generic equation for an unknown point
note that this is basically the mean value theorem

lots of special cases:

$$y_0 - \tfrac{1}{2}y_u - \tfrac{1}{2}y_d = 0 \qquad \text{(left and right boundaries)}$$

$$y_0 - \tfrac{1}{2}y_l - \tfrac{1}{2}y_r = 0 \qquad \text{(top and bottom boundaries)}$$

$$y_0 - \tfrac{1}{2}y_r - \tfrac{1}{2}y_d = 0 \qquad \text{(top-left corner)}$$

$$y_0 - \tfrac{1}{2}y_l - \tfrac{1}{2}y_d = 0 \qquad \text{(top-right corner)}$$

$$y_0 - \tfrac{1}{2}y_r - \tfrac{1}{2}y_u = 0 \qquad \text{(bottom-left corner)}$$

$$y_0 - \tfrac{1}{2}y_l - \tfrac{1}{2}y_u = 0 \qquad \text{(bottom-right corner)}$$

There is exactly one equation for each grid point, so we can solve this as a giant (sparse!) linear system, e.g., by the bi-conjugate gradient method.

Surprise! It's in NR3, as Laplace_interp, using Linbcg for the solution.

Easy to embed in a mex function for Matlab

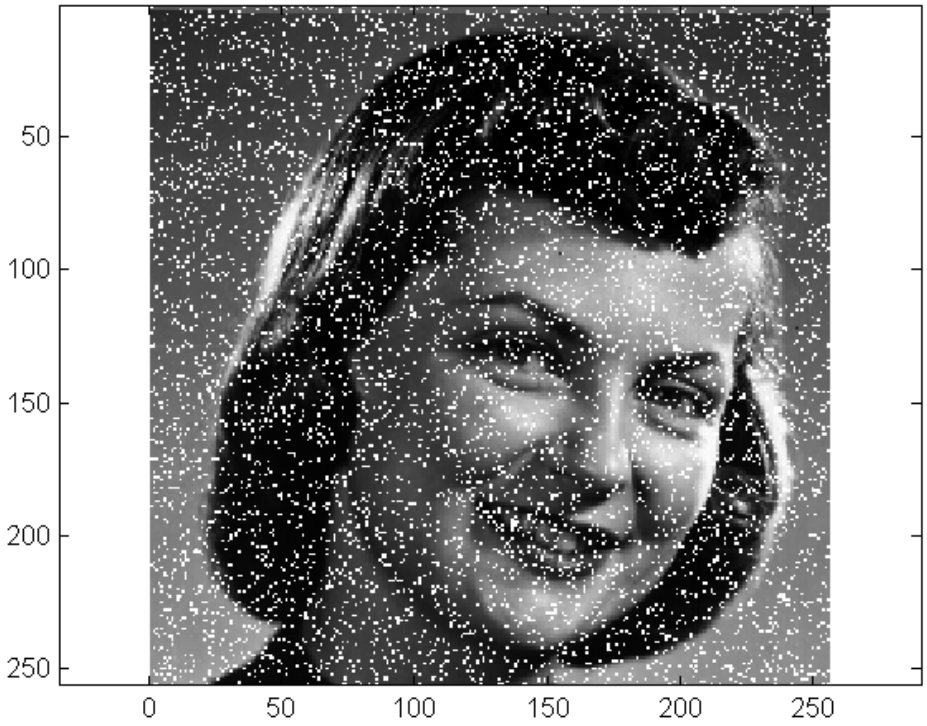
```
#include "..\nr3_matlab.h"
#include "linbcg.h"
#include "interp_laplace.h“

/* Usage:
          outmatrix = laplaceinterp(inmatrix)
*/

Laplace_interp *mylap = NULL;

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]) {
     if (nrhs != 1 || nlhs != 1) throw("laplaceinterp.cpp: bad number of args");
     MatDoub ain(prhs[0]);
     MatDoub aout(ain.nrows(),ain.ncols(),plhs[0]);
     aout = ain; // matrix op
     mylap = new Laplace_interp(aout);
     mylap->solve();
     delete mylap;
     return;
}
```

Let's try it on our favorite face for filtering
(But this is interpolation, not filtering: there is no noise!)

```
IN = fopen('image-face.raw','r');
face = flipud(reshape(fread(IN),256,256)');
fclose(IN);
bwcolormap = [0:1/256:1; 0:1/256:1; 0:1/256:1]';
image(face)
colormap(bwcolormap);
axis('equal')
```

```
facemiss = face;
ranface = rand(size(face));
facemiss(ranface < 0.1) = 255;
image(facemiss)
colormap(bwcolormap)
axis('equal')
```

delete a random 10% of pixels



```
facemiss(facemiss > 254) = 9.e99;
newface = laplaceinterp(facemiss);
image(newface)
colormap(bwcolormap)
axis('equal')
```

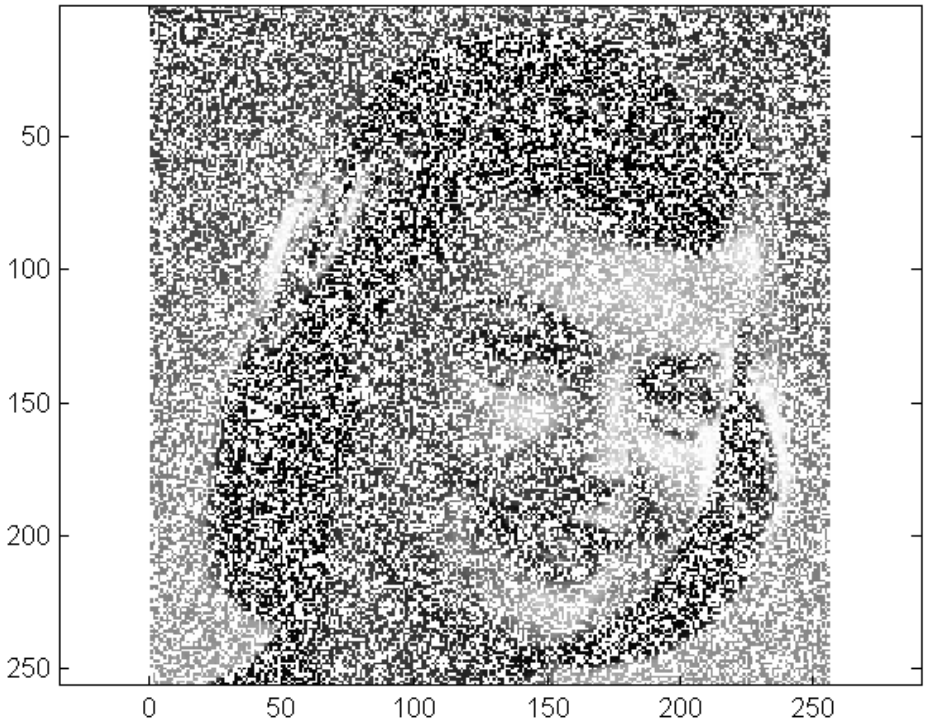
restore them by Laplace interpolation

this is the convention expected by laplaceinterp for missing data

pretty amazing!

```
facemiss = face;
ranface = rand(size(face));
facemiss(ranface < 0.5) = 255;
image(facemiss)
colormap(bwcolormap)
axis('equal')
```

delete a random 50% of pixels



```
facemiss(facemiss > 254) = 9.e99;
newface = laplaceinterp(facemiss);
image(newface)
colormap(bwcolormap)
axis('equal')
```

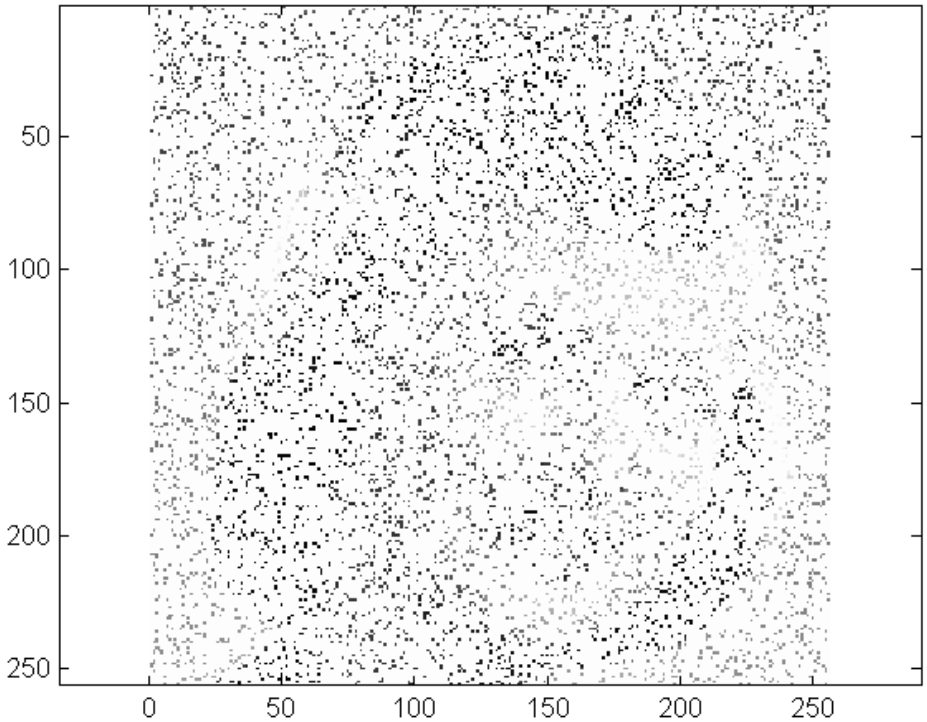
restore them by Laplace interpolation

starting to see some degradation

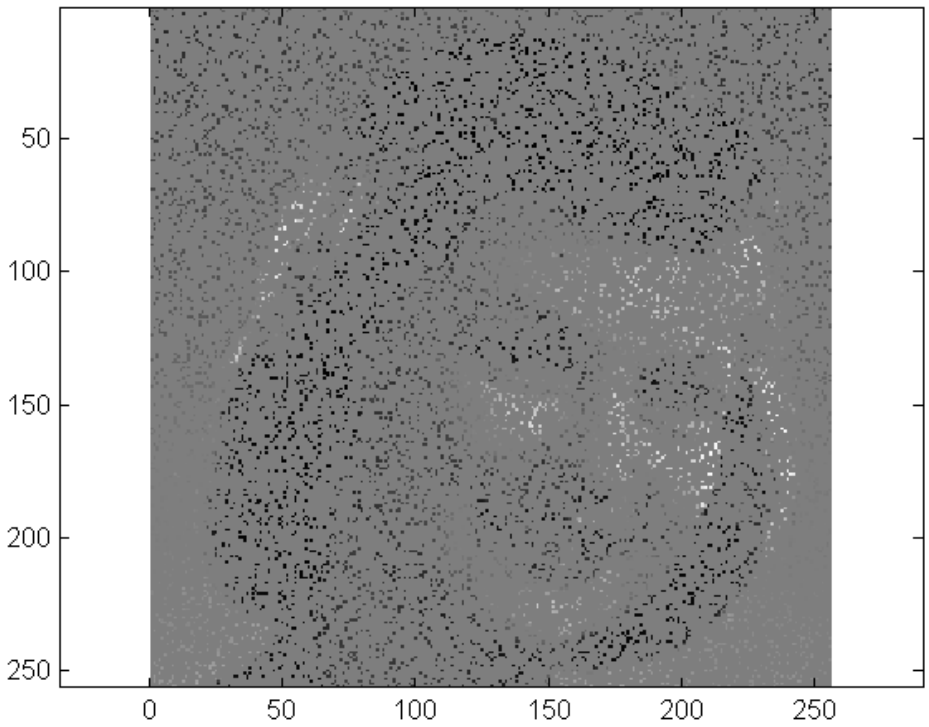
```
facemiss = face;
ranface = rand(size(face));
facemiss(ranface < 0.9) = 255;
image(facemiss)
colormap(bwcolormap)
axis('equal')
```

delete a random 90% of pixels

(well, it's cheating a bit, because your eye can't see the shades of grey in the glare of all that white)



This is a bit more fair…



```
facemiss(facemiss > 254) = 9.e99;
newface = laplaceinterp(facemiss);
image(newface)
colormap(bwcolormap)
axis('equal')
```

restore by Laplace interpolation

still pretty amazing (e.g., would you have thought that the individual teeth were present in the sparse image?)