Subject: Reconstruct surface from gradient field?
Posted by dg86 on Thu, 13 Feb 2014 22:24:22 GMT
View Forum Message <> Reply to Message

Dear Folks,

I have measurements of the gradient field (tangent vectors) associated with
a two-dimensional surface, and would like to compute the height of the surface
itself. I understand that there are standard algorithms to do this, and wonder if
any of them have been implemented in IDL. If so, I'd be grateful for a pointer.

More specifically, I have values grad_x(x,y) and grad_y(x,y) that represent the
gradient of an unknown function z(x,y) measured at irregularly gridded locations
x and y in the plane. I can use griddata() to interpolate the gradient
functions smoothly across the plane. I want to calculate z(x,y) from grad_x and grad_y.
Moreover, I want to do this as accurately as possible.

Simple line integration is OK, but yields errors in the estimate for z(x,y) that vary
widely across the field of view. I've also taken a first shot at using FFTs to solve
Poisson's equation for z(x,y). Tests with simulated data were not encouraging.
These disappointing first efforts make me very hopeful indeed that somebody else
has a good solution already coded up, and is willing to share!

All the best,

David

Subject: Re: Reconstruct surface from gradient field?
Posted by Craig Markwardt on Sat, 15 Feb 2014 01:22:46 GMT
View Forum Message <> Reply to Message

On Thursday, February 13, 2014 5:24:22 PM UTC-5, David Grier wrote:
> Dear Folks,
>
>
>
> I have measurements of the gradient field (tangent vectors) associated with
>
> a two-dimensional surface, and would like to compute the height of the surface
>
> itself. I understand that there are standard algorithms to do this, and wonder if
>
> any of them have been implemented in IDL. If so, I'd be grateful for a pointer.

I don't have any existing code.

I think you should look at this as a least squares problem. You have measurements, presumably with some measurement error, and you want a surface that fits those measurements within the error. I think that is best expressed as a least squares (fitting) problem.

In least squares, you are trying to minimize the chi square value,

```
  CHI_SQUARE = TOTAL((grad_x(xj,yj) - grad_x_measured(xj,yj))^2) + $
             TOTAL((grad_y(xj,yj) - grad_y_measured(xj,yj))^2)
```

where XJ and YJ are the (x,y) position of every measured point and GRAD_{X,Y} is the model-calculated gradient and GRAD_{X,Y}_MEASURED is the measured value. You can easily add measurement errors to this equation in the standard way for chi-square.

Now it's "just" a matter of defining a function F whose gradients GRAD_X and GRAD_Y you can calculate, and then adjusting the the function until you get a good fit by minimizing CHI_SQUARE. Sounds easy!

You will need to consider the nature of your surface. There are an infinite number of surfaces F that will produce your measured gradients, to within the measurement range. Do you need some smoothness to your function? Is a bilinear interpolation enough? Does it need to have continuous derivatives everywhere? Do you need the function on a regular grid or an irregular grid?

Let's say you have a small regular grid of NX x NY points and you are satisfied with a piecewise bilinear function (small flat plate segments between the points). Then this is really a function with NX x NY parameters, and you want to solve for each of these parameters. NX x NY better be small or you will run into problems.

Then the gradient at any point (xj,yj) will be a simple finite difference,

```
  grad_x(xj,yj) = (F(xj+dx,yj)-F(xj-dx,yj))/(2*dx)
```

and similar for GRAD_Y. If you substitute this back into the expression for CHI_SQUARE, you have an equation you want to minimize that depends on the measured values and the parameterized F() function.

To solve this as a least squares problem, you will take the derivative of CHI_SQUARE with respect to each parameter and set to zero, which is the standard way to solve least squares problems (see Numerical Recipes). That gives you NX x Ny equations. For this simple bilinear problem, you will get a very sparse matrix that depends only on (i-2, i, i+2) in each {X,Y} direction.

A standard least squares solver can do this as long as the number of parameters is small. In fact, if the number of parameters is small, you can chose from some nice functions like bicubic spline or thin plate splines which will give you a smooth behavior. (Even better if you can just put the control points for your spline at the places where you know the function is varying). You can easily compute the gradients by finite difference techniques.

Otherwise you will need to develop a specialized matrix solver which can handle sparseness of this kind.   Since X and Y are coupled, it's sparse, but there will be lots of off-diagonal terms. (you can't put both X and Y on the diagonal at the same time)

I said it was "just" a least squares problem, but if you have a large grid, it could get problematic!

There might be some heuristic way to iterate towards the solution, I'm not sure. Poisson's equation has a nice solution like that, but you don't have a Poisson's equation unless you take a (even more noisy) derivative of your measured data.

Craig

Subject: Re: Reconstruct surface from gradient field?
Posted by Craig Markwardt on Sat, 15 Feb 2014 01:26:25 GMT
View Forum Message <> Reply to Message

On Friday, February 14, 2014 8:22:46 PM UTC-5, Craig Markwardt wrote:
> On Thursday, February 13, 2014 5:24:22 PM UTC-5, David Grier wrote:

> There might be some heuristic way to iterate towards the solution, I'm not sure. Poisson's equation has a nice solution like that, but you don't have a Poisson's equation unless you take a (even more noisy) derivative of your measured data.

This page might have some interesting ideas.
 http://dsp.stackexchange.com/questions/2859/how-do-i-numeric
ally-calculate-a-function-from-its-noisy-gradient

... especially the Frankot-Chellappa discussion. If you solve this by FFT, you will still need some kind of spatial filter to de-weight the measurement noise.

Craig

Subject: Re: Reconstruct surface from gradient field?
Posted by David Fanning on Sat, 15 Feb 2014 01:38:28 GMT
View Forum Message <> Reply to Message

Craig Markwardt writes:

> ... especially the Frankot-Chellappa discussion. If you solve this by FFT, you will still need some kind of spatial filter to de-weight the measurement noise.

Oh! Uh, ... sorry I asked. :-(

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.

Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: Reconstruct surface from gradient field?
Posted by dg86 on Sat, 15 Feb 2014 02:42:53 GMT
View Forum Message <> Reply to Message

On Friday, February 14, 2014 8:26:25 PM UTC-5, Craig Markwardt wrote:
> On Friday, February 14, 2014 8:22:46 PM UTC-5, Craig Markwardt wrote:
>
>> On Thursday, February 13, 2014 5:24:22 PM UTC-5, David Grier wrote:
>
>
>
>> There might be some heuristic way to iterate towards the solution, I'm not sure. Poisson's equation has a nice solution like that, but you don't have a Poisson's equation unless you take a (even more noisy) derivative of your measured data.
>
>
>
> This page might have some interesting ideas.
>
> http://dsp.stackexchange.com/questions/2859/how-do-i-numeric
ally-calculate-a-function-from-its-noisy-gradient
>
>
>
> ... especially the Frankot-Chellappa discussion. If you solve this by FFT, you will still need some kind of spatial filter to de-weight the measurement noise.
>
>
>
> Craig

Dear Craig,

Thanks for all of your helpful suggestions. Your comments about noise and the Poisson problem were spot on, and led to a better solution than I'd found previously. Rather than computing the divergence of the gradient field by finite differences, I'm calculating it in reciprocal space, with noise suppression:

Z(kx,ky) = -i [kx Gx(kx,ky) + ky Gy(kx,ky)] / [k^2 + eps^2]

Z(kx,ky) is the Fourier transform of the desired solution. Gx and Gy are the Fourier transforms of the gradients, computed with FFT(). The key thing is to choose the parameter eps to suppress noise. It's sort of like a Wiener filter. The solution then is the real part of the inverse FFT of Z(kx,ky).

This is a lot better than my previous effort, but still not good enough. Next, I'm going to try your least-squares approach.

TTFN,

David