Jason Byrne <jbyrne6@gmail.com>

derivative error discussion

4 messages

Jason Byrne <jbyrne6@gmail.com> **26 February 2008 16:10**
To: David Long <long.daithi@gmail.com>
Cc: Peter Gallagher <peter.gallagher@tcd.ie>, "R. T. James McAteer" <james.mcateer@tcd.ie>

... I hope I've got all this right in my head now.

deriv.pro works by 3 point Lagrangian interpolation, described here:
http://en.wikipedia.org/wiki/Lagrange_polynomial

The propagation of errors equation is at: (and in the handout photocopied from 1st year labs)
http://mathworld.wolfram.com/ErrorPropagation.html
The error equation is derived from the Taylor expansion of a function to the first degree.

derivsig.pro seems to apply the idea of the error propagation equation to the use of 3-point Lagrangian in deriv.pro. The basic formula becoming:

(sigd)^2 = [(sigy_n-1)^2 + (sigy_n+1)^2] / (x_n-1 - x_n+1)^2

ie: the error in the previous point plus the error in the following point, divided by the interval. (The squares are just relating variance instead of standard deviation: variance = sigma^2).
If the errors in the x interval aren't the same (aren't all equal to each other) it recalculates to include their individual effects and adds it on to the sigd.

Then for the edge points it does the same but with weightings (similar to how deriv works):

sigd[0] = (sigy[0]^2*9.0 + sigy[1]^2*16.0 + sigy[2]^2) / (x[2]-x[0])^2
sigd[n-1]=(sigy[n-1]^2*9.0 + sigy[n-2]^2*16.0 + sigy[n-3]^2) / (x[2]-x[0])^2

again recalculating for the x errors if needed.

```
;    See Bevington, "Data Analysis and Reduction for the Physical
;          Sciences," McGraw-Hill (1969), Chap 4.
```

Jason Byrne <jbyrne6@gmail.com> **24 June 2008 19:49**
To: Shane Maloney <shane.maloney98@gmail.com>, David Long <long.daithi@gmail.com>
Cc: Peter Gallagher <peter.gallagher@tcd.ie>, James McAteer <james.mcateer@tcd.ie>

Hey Shane, Dave,
There's been a lot of talk about error bars and below is an email I sent a while back to re-iterate the use of IDL's deriv and derivsig for obtaining kinematics, with a note at the bottom from looking at this today.
Adding to this was discussion on the possible need for hard-coding minimum/maximum errors since precision

cannot be higher than the resolution etc. This brought up questions of how imagers work, PointSpreadFunction awareness, in multiscale the idea of filter size, and in particular a rule by Nyquist that a sampling rate must be at least twice the maximum frequency in order to sufficiently detect it (nice little example at http://www.cs.cf.ac.uk/Dave/Multimedia/node149.html).
Hopefully it's all correct and straight-forward, though discussion/debate is of course welcome/inevitable ;) and I have Peter's "Bevington" book on my desk if anyone wants it. Tis important to get this stuff right, so do point out any flaws, because I'm finding small effects can drastically change the CME kinematics (similarly we saw the Dave Wave cadence issue?!).
Cheers,
Jason.

---------- Forwarded message ----------
From: **Jason Byrne** <jbyrne6@gmail.com>
Date: 2008/2/26
Subject: derivative error discussion
To: David Long <long.daithi@gmail.com>
Cc: Peter Gallagher <peter.gallagher@tcd.ie>, "R. T. James McAteer" <james.mcateer@tcd.ie>

... I hope I've got all this right in my head now.

deriv.pro works by 3 point Lagrangian interpolation, described here:
http://en.wikipedia.org/wiki/Lagrange_polynomial

The propagation of errors equation is at: (and in the handout photocopied from 1st year labs)
http://mathworld.wolfram.com/ErrorPropagation.html
The error equation is derived from the Taylor expansion of a function to the first degree.

derivsig.pro seems to apply the idea of the error propagation equation to the use of 3-point Lagrangian in deriv.pro. The basic formula becoming:

(sigd)^2 = [(sigy_n-1)^2 + (sigy_n+1)^2] / (x_n-1 - x_n+1)^2

ie: the error in the previous point plus the error in the following point, divided by the interval. (The squares are just relating variance instead of standard deviation: variance = sigma^2).
If the errors in the x interval aren't the same (aren't all equal to each other) it recalculates to include their individual effects and adds it on to the sigd.

Then for the edge points it does the same but with weightings (similar to how deriv works):

sigd[0] = (sigy[0]^2*9.0 + sigy[1]^2*16.0 + sigy[2]^2) / (x[2]-x[0])^2
sigd[n-1]=(sigy[n-1]^2*9.0 + sigy[n-2]^2*16.0 + sigy[n-3]^2) / (x[2]-x[0])^2

again recalculating for the x errors if needed.

```
;    See Bevington, "Data Analysis and Reduction for the Physical
;          Sciences," McGraw-Hill (1969), Chap 4.
```

NOTE:

I can't find better reasoning than derivsig.pro propagation for unevenly spaced samples. Looking at deriv.pro again, it's that a Lagrange interpolation is performed before the derivative is taken, so the weighting between points is based upon how a fit would sit between them, and derivsig goes hand-in-hand with this assumption, *but* does appear to be determining the propagated error at the midpoint of the interval.

Peter Gallagher <gallagpt@tcd.ie> **24 June 2008 19:57**
Reply-To: peter.gallagher@tcd.ie
To: Jason Byrne <jbyrne6@gmail.com>

... thanks for send that email. We might want to give Peter Schuck a call at some point this week or next to bounce some of these ideas off him.

I think we now have a pretty thorough treatment of the errors. It will be interesting to see how it effects our other plots and the derived kinematics.

Cheers,

Peter.

Peter Gallagher PhD
www.SolarMonitor.org
Astrophysics Research Group
School of Physics
Trinity College Dublin
Dublin 2
Ireland
W: www.physics.tcd.ie/astrophysics
T: +353 (0)1 896 1300
F: +353 (0)1 671 1759

Quoting Jason Byrne <jbyrne6@gmail.com>:

> Hey Shane, Dave,
> There's been a lot of talk about error bars and below is an email I sent a
> while back to re-iterate the use of IDL's deriv and derivsig for obtaining
> kinematics, with a note at the bottom from looking at this today.
> Adding to this was discussion on the possible need for hard-coding
> minimum/maximum errors since precision cannot be higher than the resolution
> etc. This brought up questions of how imagers work, PointSpreadFunction
> awareness, in multiscale the idea of filter size, and in particular a rule
> by Nyquist that a sampling rate must be at least twice the maximum frequency
> in order to sufficiently detect it (nice little example at
> http://www.cs.cf.ac.uk/Dave/Multimedia/node149.html).
> Hopefully it's all correct and straight-forward, though discussion/debate is

> of course welcome/inevitable ;)
> and I have Peter's "Bevington" book on my desk if anyone wants it. Tis
> important to get this stuff right, so do point out any flaws, because I'm
> finding small effects can drastically change the CME kinematics (similarly
> we saw the Dave Wave cadence issue?!).
> Cheers,
> Jason.
>
>
> ---------- Forwarded message ----------
> From: Jason Byrne <jbyrne6@gmail.com>
> Date: 2008/2/26
> Subject: derivative error discussion
> To: David Long <long.daithi@gmail.com>
> Cc: Peter Gallagher <peter.gallagher@tcd.ie>, "R. T. James McAteer" <
> james.mcateer@tcd.ie>
>
>
> ... I hope I've got all this right in my head now.
>
>
>
> deriv.pro works by 3 point Lagrangian interpolation, described here:
> http://en.wikipedia.org/wiki/Lagrange_polynomial
>
> The propagation of errors equation is at: (and in the handout photocopied
> from 1st year labs)
> http://mathworld.wolfram.com/ErrorPropagation.html
> The error equation is derived from the Taylor expansion of a function to the
> first degree.
>
> derivsig.pro seems to apply the idea of the error propagation equation to
> the use of 3-point Lagrangian in deriv.pro. The basic formula becoming:
>
> (sigd)^2 = [(sigy_n-1)^2 + (sigy_n+1)^2] / (x_n-1 - x_n+1)^2
>
> ie: the error in the previous point plus the error in the following point,
> divided by the interval. (The squares are just relating variance instead of
> standard deviation: variance = sigma^2).
> If the errors in the x interval aren't the same (aren't all equal to each
> other) it recalculates to include their individual effects and adds it on to
> the sigd.
>
> Then for the edge points it does the same but with weightings (similar to
> how deriv works):
>
> sigd[0] = (sigy[0]^2*9.0 + sigy[1]^2*16.0 + sigy[2]^2) / (x[2]-x[0])^2
> sigd[n-1]=(sigy[n-1]^2*9.0 + sigy[n-2]^2*16.0 + sigy[n-3]^2) / (x[2]-x[0]
>)^2
>
> again recalculating for the x errors if needed.
>

> ; See Bevington, "Data Analysis and Reduction for the Physical
> ; Sciences," McGraw-Hill (1969), Chap 4.
>
>
>
>
> NOTE:
>
> I can't find better reasoning than derivsig.pro propagation for unevenly
> spaced samples. Looking at deriv.pro again, it's that a Lagrange
> interpolation is performed before the derivative is taken, so the weighting
> between points is based upon how a fit would sit between them, and derivsig
> goes hand-in-hand with this assumption,
> *but* does appear to be determining the propagated error at the midpoint of
> the interval.
>

Jason Byrne <jbyrne6@gmail.com> **26 June 2008 10:23**
To: JayDog <jbyrne6@gmail.com>

There is an extra part to derivsig to account for including an error in the x-direction of the points (normally time). This means the final equation for computing the standard deviation on a point determined from using deriv on a previous three points (eg: obtained velocity point n from derivative of three height measurements at points n-1, n, n+1)

sigd^2 = ((sigy_n+1)^2 + (sigy_n-1)^2) / dx^2 + [((sigx_n+1)^2 + (sigx_n-1)^2) / dx^2] * [dy^2 / dx^2]

2008/6/24 Jason Byrne <jbyrne6@gmail.com>:

> Hey Shane, Dave,
> There's been a lot of talk about error bars and below is an email I sent a while back to re-iterate the use of IDL's deriv and derivsig for obtaining kinematics, with a note at the bottom from looking at this today. Adding to this was discussion on the possible need for hard-coding minimum/maximum errors since precision cannot be higher than the resolution etc. This brought up questions of how imagers work, PointSpreadFunction awareness, in multiscale the idea of filter size, and in particular a rule by Nyquist that a sampling rate must be at least twice the maximum frequency in order to sufficiently detect it (nice little example at http://www.cs.cf.ac.uk/Dave/Multimedia/node149.html).
> Hopefully it's all correct and straight-forward, though discussion/debate is of course welcome/inevitable ;) and I have Peter's "Bevington" book on my desk if anyone wants it. Tis important to get this stuff right, so do point out any flaws, because I'm finding small effects can drastically change the CME kinematics (similarly we saw the Dave Wave cadence issue?!).
> Cheers,
> Jason.
>
>
>
> ---------- Forwarded message ----------
> From: **Jason Byrne** <jbyrne6@gmail.com>

Date: 2008/2/26
Subject: derivative error discussion
To: David Long <long.daithi@gmail.com>
Cc: Peter Gallagher <peter.gallagher@tcd.ie>, "R. T. James McAteer" <james.mcateer@tcd.ie>

... I hope I've got all this right in my head now.

deriv.pro works by 3 point Lagrangian interpolation, described here:
http://en.wikipedia.org/wiki/Lagrange_polynomial

The propagation of errors equation is at: (and in the handout photocopied from 1st year labs)
http://mathworld.wolfram.com/ErrorPropagation.html
The error equation is derived from the Taylor expansion of a function to the first degree.

derivsig.pro seems to apply the idea of the error propagation equation to the use of 3-point Lagrangian in deriv.pro. The basic formula becoming:

(sigd)^2 = [(sigy_n-1)^2 + (sigy_n+1)^2] / (x_n-1 - x_n+1)^2

ie: the error in the previous point plus the error in the following point, divided by the interval. (The squares are just relating variance instead of standard deviation: variance = sigma^2).
If the errors in the x interval aren't the same (aren't all equal to each other) it recalculates to include their individual effects and adds it on to the sigd.

Then for the edge points it does the same but with weightings (similar to how deriv works):

sigd[0] = (sigy[0]^2*9.0 + sigy[1]^2*16.0 + sigy[2]^2) / (x[2]-x[0])^2
sigd[n-1]=(sigy[n-1]^2*9.0 + sigy[n-2]^2*16.0 + sigy[n-3]^2) / (x[2]-x[0])^2

again recalculating for the x errors if needed.

```
;    See Bevington, "Data Analysis and Reduction for the Physical
;          Sciences," McGraw-Hill (1969), Chap 4.
```

NOTE:

I can't find better reasoning than derivsig.pro propagation for unevenly spaced samples. Looking at deriv.pro again, it's that a Lagrange interpolation is performed before the derivative is taken, so the weighting between points is based upon how a fit would sit between them, and derivsig goes hand-in-hand with this assumption,
but does appear to be determining the propagated error at the midpoint of the interval.