Subject: Yet another user with poly_fit problems
Posted by Gus on Mon, 30 Sep 2013 19:59:16 GMT
View Forum Message <> Reply to Message

Hello everyone,

I've read a few of the older posts on this topic, but their solution didn't really help me solve the problem that I am currently having with the poly_fit function. The set of coefficients generated by the function (a 4th degree polynomial) produces some rather absurd results. Here is a short version of the problem I am having.

X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]

C = poly_fit(X, Y, /double, yfit=D)

IDL generates the following coefficients (for C)

```
      15.940691
   0.0015355228
-3.0965110e-007
 1.1170193e-011
-6.6767399e-017
```

Yet, one will clearly see that this fit produces rather undesirable results since, within the same range of X values (0 to roughly 150,000), this fit will produce Y values that can be as high as 1600 and as low as -3000 (rather than between 15.9 and 20). Excel is generating better coefficients than IDL!

Here is what I have already tried to do (and did not solve the problem)

1) Double precision of X and Y prior to using the poly_fit function (notice that I am using the "/double" keyword function already in that function);

2) Subtracting the mean of X from that array, before fitting the data - suggested in previous posts;

3) Subtracting the value of X[0] from that array, before fitting the data;

4) Subtracting the mean of Y from that array, before fitting the data.

Does anyone know of any other solution to this problem?

Subject: Re: Yet another user with poly_fit problems
Posted by Gus on Mon, 30 Sep 2013 20:03:59 GMT
View Forum Message <> Reply to Message

Correction for 3), I meant subtracting Y[0] from the Y array.

On Monday, September 30, 2013 4:59:16 PM UTC-3, Gus wrote:
> Hello everyone,
>
>
>
> I've read a few of the older posts on this topic, but their solution didn't really help me solve the problem that I am currently having with the poly_fit function. The set of coefficients generated by the function (a 4th degree polynomial) produces some rather absurd results. Here is a short version of the problem I am having.
>
>
>
> X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
>
> Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]
>
>
>
> C = poly_fit(X, Y, /double, yfit=D)
>
>
>
> IDL generates the following coefficients (for C)
>
>
>
> 15.940691
>
> 0.0015355228
>
> -3.0965110e-007
>
> 1.1170193e-011
>
> -6.6767399e-017
>
>
>
> Yet, one will clearly see that this fit produces rather undesirable results since, within the same range of X values (0 to roughly 150,000), this fit will produce Y values that can be as high as 1600 and as low as -3000 (rather than between 15.9 and 20). Excel is generating better coefficients than IDL!
>
>
>
> Here is what I have already tried to do (and did not solve the problem)
>

>
>
> 1) Double precision of X and Y prior to using the poly_fit function (notice that I am using the "/double" keyword function already in that function);
>
>
>
> 2) Subtracting the mean of X from that array, before fitting the data - suggested in previous posts;
>
>
>
> 3) Subtracting the value of X[0] from that array, before fitting the data;
>
>
>
> 4) Subtracting the mean of Y from that array, before fitting the data.
>
>
>
> Does anyone know of any other solution to this problem?

Subject: Re: Yet another user with poly_fit problems
Posted by David Fanning on Mon, 30 Sep 2013 20:09:10 GMT
View Forum Message <> Reply to Message

Gus writes:

> I've read a few of the older posts on this topic, but their solution didn't really help me solve the problem that I am currently having with the poly_fit function. The set of coefficients generated by the function (a 4th degree polynomial) produces some rather absurd results. Here is a short version of the problem I am having.
>
> X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
> Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]
>
> C = poly_fit(X, Y, /double, yfit=D)
>
> IDL generates the following coefficients (for C)
>
> 15.940691
> 0.0015355228
> -3.0965110e-007
> 1.1170193e-011
> -6.6767399e-017

This code doesn't seem to work for me:

IDL> X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
IDL> Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]
IDL> C = poly_fit(X, Y, /double, yfit=D)
% Compiled module: POLY_FIT.
% Variable is undefined: NDEGREE.

Are you sure you are using the right POLY_FIT?

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: Yet another user with poly_fit problems
Posted by suicidaleggroll on Mon, 30 Sep 2013 20:18:03 GMT
View Forum Message <> Reply to Message

On Monday, September 30, 2013 2:09:10 PM UTC-6, David Fanning wrote:
> Gus writes:
>
>
>
>> I've read a few of the older posts on this topic, but their solution didn't really help me solve the problem that I am currently having with the poly_fit function. The set of coefficients generated by the function (a 4th degree polynomial) produces some rather absurd results. Here is a short version of the problem I am having.
>
>>
>
>> X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
>
>> Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]
>
>>
>
>> C = poly_fit(X, Y, /double, yfit=D)
>
>>
>
>> IDL generates the following coefficients (for C)
>

>>
>
>>   15.940691
>
>>   0.0015355228
>
>>   -3.0965110e-007
>
>>   1.1170193e-011
>
>>   -6.6767399e-017
>
>
>
> This code doesn't seem to work for me:
>
>
>
> IDL> X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
>
> IDL> Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]
>
> IDL> C = poly_fit(X, Y, /double, yfit=D)
>
> % Compiled module: POLY_FIT.
>
> % Variable is undefined: NDEGREE.
>
>
>
> Are you sure you are using the right POLY_FIT?
>
>
>
> 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.")

He just missed the "4" in the call (for a 4th order polynomial).

Gus - actually Excel gives the EXACT same answer as IDL, which, as you said, is completely ridiculous. The problem is you're fitting a 4th order polynomial to 5 data points. Because of this, the solution will be mathematically perfect (R^2 = 1), because the solution is not overdetermined and no least squares fitting can be performed.

You need more points in order to generate a "valid" 4th order poly fit so the "fit" can actually do some good, rather than just reproduce your 5 values exactly (with god knows what in between).

I've run into this in the past, and in that application it was reasonable to linearly interpolate my 5 points to, say, 1000 points, and then perform the poly fit on that.

For example:
X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]

```
newX = dindgen(1000)/999 * (max(X)-min(X)) + min(X)
newY = interpol(Y, X, newX)
C = poly_fit(newX, newY, 4, /double, yfit=D)
print, C
      17.356485
  0.00010074819
 -2.0171981e-09
  1.8082251e-14
 -5.6591322e-20
```

Subject: Re: Yet another user with poly_fit problems
Posted by David Fanning on Mon, 30 Sep 2013 20:23:13 GMT
View Forum Message <> Reply to Message

suicidaleggroll@gmail.com writes:

>
> On Monday, September 30, 2013 2:09:10 PM UTC-6, David Fanning wrote:
>> Gus writes:
>>
>>
>>
>>> I've read a few of the older posts on this topic, but their solution didn't really help me solve the problem that I am currently having with the poly_fit function. The set of coefficients generated by the function (a 4th degree polynomial) produces some rather absurd results. Here is a short

version of the problem I am having.
>>
>>>
>>
>>> X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
>>
>>> Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]
>>
>>>
>>
>>> C = poly_fit(X, Y, /double, yfit=D)
>>
>>>
>>
>>> IDL generates the following coefficients (for C)
>>
>>>
>>
>>>   15.940691
>>
>>>   0.0015355228
>>
>>>   -3.0965110e-007
>>
>>>   1.1170193e-011
>>
>>>   -6.6767399e-017
>>
>>
>>
>> This code doesn't seem to work for me:
>>
>>
>>
>> IDL> X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
>>
>> IDL> Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]
>>
>> IDL> C = poly_fit(X, Y, /double, yfit=D)
>>
>> % Compiled module: POLY_FIT.
>>
>> % Variable is undefined: NDEGREE.
>>
>>
>>
>> Are you sure you are using the right POLY_FIT?
>>

>>
>>
>> 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.")
>
>
> He just missed the "4" in the call (for a 4th order polynomial).
>
> Gus - actually Excel gives the EXACT same answer as IDL, which, as you said, is completely ridiculous. The problem is you're fitting a 4th order polynomial to 5 data points. Because of this, the solution will be mathematically perfect (R^2 = 1), because the solution is not overdetermined and no least squares fitting can be performed.
>
> You need more points in order to generate a "valid" 4th order poly fit so the "fit" can actually do some good, rather than just reproduce your 5 values exactly (with god knows what in between).
>
> I've run into this in the past, and in that application it was reasonable to linearly interpolate my 5 points to, say, 1000 points, and then perform the poly fit on that.
>
> For example:
> X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
> Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]
>
> newX = dindgen(1000)/999 * (max(X)-min(X)) + min(X)
> newY = interpol(Y, X, newX)
> C = poly_fit(newX, newY, 4, /double, yfit=D)
> print, C
> 17.356485
> 0.00010074819
> -2.0171981e-09
> 1.8082251e-14
> -5.6591322e-20

It is probably worth pointing out that the order of the coefficients in
the variable C are e, d, c, b, and a. That sometimes (nearly every time
with me) gets missed.

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: Yet another user with poly_fit problems
Posted by Heinz Stege on Mon, 30 Sep 2013 22:46:58 GMT
View Forum Message <> Reply to Message

On Mon, 30 Sep 2013 12:59:16 -0700 (PDT), Gus wrote:

> Hello everyone,
>
> I've read a few of the older posts on this topic, but their solution didn't really help me solve the problem that I am currently having with the poly_fit function. The set of coefficients generated by the function (a 4th degree polynomial) produces some rather absurd results. Here is a short version of the problem I am having.
>
> X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
> Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]
>
> C = poly_fit(X, Y, /double, yfit=D)
>
> IDL generates the following coefficients (for C)
>
> 15.940691
> 0.0015355228
> -3.0965110e-007
> 1.1170193e-011
> -6.6767399e-017
>
> Yet, one will clearly see that this fit produces rather undesirable results since, within the same range of X values (0 to roughly 150,000), this fit will produce Y values that can be as high as 1600 and as low as -3000 (rather than between 15.9 and 20). Excel is generating better coefficients than IDL!
>
> Here is what I have already tried to do (and did not solve the problem)
>

> 1) Double precision of X and Y prior to using the poly_fit function (notice that I am using the "/double" keyword function already in that function);
>
> 2) Subtracting the mean of X from that array, before fitting the data - suggested in previous posts;
>
> 3) Subtracting the value of X[0] from that array, before fitting the data;
>
> 4) Subtracting the mean of Y from that array, before fitting the data.
>
> Does anyone know of any other solution to this problem?

As far as I can see, it is not possible to get better results fitting the given data with a polynomal. The curve you need to fit the data has to be very steep near x=0 and very flat for x > 20000. I don't see how to manage this by a polynomal. Please show us the coefficients from Excel. I'm really interested to see them.

A solution for you may be a function like
  y = (p0 + p1*x + p2*x^2) / (x - x0)
I get the coefficients
  x0=-1543.59
  [p0,p1,p2]=[24607.0, 18.9245, 8.39527e-006]
with a self-written fit routine for non-linear functions. You can try curvefit (built-in) or mpcurvefit (from Craig Markwardt).

HTH, Heinz

Subject: Re: Yet another user with poly_fit problems
Posted by Gus on Mon, 30 Sep 2013 23:15:22 GMT
View Forum Message <> Reply to Message

Problem has been solved by suicida's reply (THANK YOU VERY MUCH).

Firstly, I forgot to add then "4" when I typed up the original post, but needless to say, I was using that correctly in IDL. I also had the coefficient in the right order (the last one is this case is coefficient for the "X^4"). Thanks David.

Anyhow, as I typed the original message here I had begun to wonder whether my error was indeed caused by the fact I had so few points in my array to execute a 4th order polynomial fit. Before I had come to a solution on how to test for that, I got the answer here. Increasing my data size through linear interpolation, seems to do the trick.

Heinz - Suicida's post solved the problem I have. I had thought about using a different method for fitting, but given that my problem is just a tiny snip of my code, which is used for a variety of different data points, changing the fitting methodology would create some comparative problems in its application. So, I need to stick to a single fitting method. Also, I totally misspoke with the better results for Excel (forgive my blasphemous accusation). I had, inadvertently, used a different dataset for the fitting in Excel.

Thank you everyone for the quick contribution!

On Monday, September 30, 2013 4:59:16 PM UTC-3, Gus wrote:
> Hello everyone,
>
>
>
> I've read a few of the older posts on this topic, but their solution didn't really help me solve the problem that I am currently having with the poly_fit function. The set of coefficients generated by the function (a 4th degree polynomial) produces some rather absurd results. Here is a short version of the problem I am having.
>
>
>
> X = [0.000000, 11.6667, 822.914, 3458.85, 27703.4, 133928.]
>
> Y = [15.9000, 16.0000, 17.0000, 18.0000, 19.0000, 20.0000]
>
>
>
> C = poly_fit(X, Y, /double, yfit=D)
>
>
>
> IDL generates the following coefficients (for C)
>
>
>
>
> 15.940691
>
> 0.0015355228
>
> -3.0965110e-007
>
> 1.1170193e-011
>
> -6.6767399e-017
>
>
>
> Yet, one will clearly see that this fit produces rather undesirable results since, within the same range of X values (0 to roughly 150,000), this fit will produce Y values that can be as high as 1600 and as low as -3000 (rather than between 15.9 and 20). Excel is generating better coefficients than IDL!
>
>
>

> Here is what I have already tried to do (and did not solve the problem)
>
>
>
> 1) Double precision of X and Y prior to using the poly_fit function (notice that I am using the "/double" keyword function already in that function);
>
>
>
> 2) Subtracting the mean of X from that array, before fitting the data - suggested in previous posts;
>
>
>
> 3) Subtracting the value of X[0] from that array, before fitting the data;
>
>
>
> 4) Subtracting the mean of Y from that array, before fitting the data.
>
>
>
> Does anyone know of any other solution to this problem?

Subject: Re: Yet another user with poly_fit problems
Posted by Ken G on Wed, 02 Oct 2013 17:35:35 GMT
View Forum Message <> Reply to Message

I feel compelled to jump in here and say that the interpolation method is mathematically dicey and I wouldn't do it. What you're essentially doing is weighting your data points in a very non-linear way. We see that the first few points are clustered closely together in x; the last few points are widely separated. The uniform x spacing in the linear interpolation therefore devotes *many* more points to the large-x-value region relative to the number of y points out there to support the data. So in your fit result, you're biasing or weighting the data as if the large y were the most significant point by far. You can see this in the results if you plot newX, D . There's a downward bulge in D between the last two points as the weighting pulls the curve down toward the linear interpolation. Plus, D fails to come close to the first 2 points.

Subject: Re: Yet another user with poly_fit problems
Posted by suicidaleggroll on Wed, 02 Oct 2013 19:31:58 GMT
View Forum Message <> Reply to Message

On Wednesday, October 2, 2013 11:35:35 AM UTC-6, Ken G wrote:
> I feel compelled to jump in here and say that the interpolation method is mathematically dicey and I wouldn't do it. What you're essentially doing is weighting your data points in a very

non-linear way. We see that the first few points are clustered closely together in x; the last few points are widely separated. The uniform x spacing in the linear interpolation therefore devotes *many* more points to the large-x-value region relative to the number of y points out there to support the data. So in your fit result, you're biasing or weighting the data as if the large y were the most significant point by far. You can see this in the results if you plot newX, D . There's a downward bulge in D between the last two points as the weighting pulls the curve down toward the linear interpolation. Plus, D fails to come close to the first 2 points.

That is true. I mentioned in my post that this approach was suitable for the application in which I used it (my points were more or less evenly spaced), but you're right that it might not be ideal for Gus. The same kind of approach could probably still be used though with different abscissa values to prevent the heavy weighting at higher X values.

Subject: Re: Yet another user with poly_fit problems
Posted by Heinz Stege on Wed, 02 Oct 2013 19:57:38 GMT
View Forum Message <> Reply to Message

On Wed, 2 Oct 2013 12:31:58 -0700 (PDT), suicidaleggroll@gmail.com
wrote:

> On Wednesday, October 2, 2013 11:35:35 AM UTC-6, Ken G wrote:
>> I feel compelled to jump in here and say that the interpolation method is mathematically dicey and I wouldn't do it. What you're essentially doing is weighting your data points in a very non-linear way. We see that the first few points are clustered closely together in x; the last few points are widely separated. The uniform x spacing in the linear interpolation therefore devotes *many* more points to the large-x-value region relative to the number of y points out there to support the data. So in your fit result, you're biasing or weighting the data as if the large y were the most significant point by far. You can see this in the results if you plot newX, D . There's a downward bulge in D between the last two points as the weighting pulls the curve down toward the linear interpolation. Plus, D fails to come close to the first 2 points.
>
> That is true. I mentioned in my post that this approach was suitable for the application in which I used it (my points were more or less evenly spaced), but you're right that it might not be ideal for Gus. The same kind of approach could probably still be used though with different abscissa values to prevent the heavy weighting at higher X values.

I also fully agree with Ken. And I tried to get the newX values
without different weighting for the X values (except for the first and
the last point). I ran:

```
newX=X
for i=0,4 do begin &$
  newX=[newX,(newX+newX[1:*])/2.] &$
  newX=newX[sort(newX)] &end
newY = interpol(Y, X, newX)
```

The result is still as Ken describes: Very bad matching for the first

four points and a downward bulge between the last two points.

For me it is a fact, that the given data points can not be fairly
fitted by a polynomial.

Heinz

Subject: Re: Yet another user with poly_fit problems
Posted by Gus on Wed, 02 Oct 2013 22:14:56 GMT
View Forum Message <> Reply to Message

Alright, I can see now that I do not have a perfect solution. For the purposes of my application, the bulging seen at the end actually concerns me less than the fact that the plotted curve is crossing the ordinate at roughly 17.3. In my application, very small variations in the value of Y (say +/- 0.2) Nevertheless, I was far happier to see reasonable Y values being generated by the new polynomial fit, as opposed to the ridiculous ones I had before.

To contextualize the discussion, I just wanted to say that the actual mathemathical equation used in the fitting is not so important for the engineering application for which I use it for. What is important is that, whichever method I use, it should be consistently applied for future datasets. Moreover, I expect to always have evenly spaced Y values, but increasingly spaced X values. Perhaps I need to look into the different fitting methods and determine whether one of them is acceptable for all future datasets.

At any rate, this was a very useful discussion because it made people in my office think about a whole new set of situations where we can run into rather problematic results when we have few points to execute a fitting (something that we expect to happen).

Gus

Subject: Re: Yet another user with poly_fit problems
Posted by Yngvar Larsen on Thu, 03 Oct 2013 11:00:17 GMT
View Forum Message <> Reply to Message

On Thursday, 3 October 2013 00:14:56 UTC+2, Gus wrote:

> To contextualize the discussion, I just wanted to say that the actual mathemathical equation used in the fitting is not so important for the engineering application for which I use it for. What is important is that, whichever method I use, it should be consistently applied for future datasets.
>
> Moreover, I expect to always have evenly spaced Y values, but increasingly spaced X values. Perhaps I need to look into the different fitting methods and determine whether one of them is acceptable for all future datasets.

How about fitting Y versus log(X+X0) then?

X = [0.000000d, 11.6667d, 822.914d, 3458.85d, 27703.4d, 133928d]
Y = [15.9000d, 16.0000d, 17.0000d, 18.0000d, 19.0000d, 20.0000d]
X0 = - min(X) + 1d0 ; "Randomly" chosen to make min(X+X0)>0 for the ALOG operation.

C = poly_fit(alog(X+X0), Y, 3, /double, yfit=D)

plot, X+X0, Y, /xlog
oplot, X+X0, D, col='ff'x

I don't know what X and Y values you expect, but this seems to work reasonably well for the ones you specify in your original post.

--
Yngvar

Subject: Re: Yet another user with poly_fit problems
Posted by on Thu, 03 Oct 2013 11:06:39 GMT
View Forum Message <> Reply to Message

Den torsdagen den 3:e oktober 2013 kl. 00:14:56 UTC+2 skrev Gus:
>
> Moreover, I expect to always have evenly spaced Y values, but increasingly spaced X values. Perhaps I need to look into the different fitting methods and determine whether one of them is acceptable for all future datasets.

This suggests to me that you should fit x as a function of y. (Although your example data set is not evenly spaced in y - perhaps the first y value should be 15.0 and not 15.9?)

Anyway, this works pretty nicely also with the data you specified, at least between the outermost y values:

c=poly_fit(y,x,4,/double)
cgplot,y,x,psym=9
yy=findgen(110)/20.+15
cgplot,yy,c[0]+yy*c[1]+yy^2*c[2]+yy^3*c[3]+yy^4*c[4],/over

or

cgplot,x,y,psym=9,/yno
cgplot,c[0]+yy*c[1]+yy^2*c[2]+yy^3*c[3]+yy^4*c[4],yy,/over

Subject: Re: Yet another user with poly_fit problems
Posted by Gus on Wed, 16 Oct 2013 17:12:16 GMT
View Forum Message <> Reply to Message

Forgive my absence, I was away on vacation. I am going to give these 2 a shot and see what results I get.

Mats - yes, I can make the first value 15.0 and not alter my results. The measurements we obtain for that variable are always spaced every 1 m (so 15, 16, 17, 18, ... for Y values would be normal). And, I actually generate equations in both forms, y = f(x) and x = f(y).

--
Gus
