Subject: Re: mpfit: multivariate fit
Posted by Vince Hradil on Tue, 08 May 2007 14:11:44 GMT
View Forum Message <> Reply to Message

On May 7, 11:19 pm, Dave <Confused.Scient...@gmail.com> wrote:
> Hi all,
>
> I'm trying to use MPFIT to numerically estimate a coordinate
> transformation matrix that relates two sensors. One set is
> uncalibrated, and the other has a known calibration. So, I have a set
> of observed vectors (xyz_obs) and a set of known vectors (xyz_known)
> and I'm trying to estimate (in a least squares sense) the
> transformation matrix T that relates them. Judiciously choosing the
> data in the fictional example below, I expect the transformation
> matrix to be 2 * identity(3,3).
>
> When I execute the code below, I get:
>

```
> IDL> .r foo
> % Compiled module: TRANS.
> % Compiled module: $MAIN$.
> Iter      1   CHI-SQUARE =       1278958.0          DOF = 291
>     P(0) =             1.00000
>     P(1) =             0.00000
>     P(2) =             0.00000
>     P(3) =             0.00000
>     P(4) =             1.00000
>     P(5) =             0.00000
>     P(6) =             0.00000
>     P(7) =             0.00000
>     P(8) =             1.00000
> % MPFIT: Error detected while calling MPFIT_FDJAC2:
> % MPFIT: Out of range subscript encountered: FJAC.
> % MPFIT: Error condition detected. Returning to MAIN level.
```

>
> Any ideas on what I'm doing wrong here?
>
> Thanks!
> Dave
>

```
> %%%
> % Contents of foo.pro
> %%%
>
> function trans, K, X=x, Y=y, err=err, forward=fw
>   model = K ## x
>   if keyword_set(fw) then return, model else return, (y-model)/err
> end
```

```
>
>  ; MAIN
>
>  ; Attempt to estimate the transformation matrix given a set
>  ; of observed Cartesian vectors and a set of known cartesian
>  ; vectors.
>
>  n = 100 ; number of 'observations'
>
>  v = [1.0d, 0.15, 0.5] ; template vector
>  xyz_obs = dblarr(n, 3) ; observations
>  for i=0, n-1 do $
>    xyz_obs[i,*] = reform( v+0.01*randomn(seed,3), 1, 3)
>
>  xyz_known = dblarr(n, 3) ; known values (trivial scaling)
>  for i=0, n-1 do $
>    xyz_known[i,*] = reform( v*2.0, 1, 3)
>
>  ; Estimate the transformation matrix, T
>  T0 = identity(3, /DOUBLE) ; initial guess transformation matrix
>  f = {x: xyz_obs, y: xyz_known, err: 0.01}
>  T = mpfit('trans', T0, functargs=f, COVAR=S2)
>
>  end
```

In the file mpfit.pro, in function mpfit_fdjac2() (line 1119) I changed fjac[0,j] = to fjac[*,j] = and it runs(?). I was getting a subscript out of range error before.

```
      if abs(dside[j]) LE 1 then begin
          ;; COMPUTE THE ONE-SIDED DERIVATIVE
          ;; Note optimization fjac(0:*,j)
-->           fjac[*,j] = (fp-fvec)/h[j]
```

The results are:

```
Iter      2   CHI-SQUARE =       377.98996          DOF = 291
    P(0) =              1.49186
    P(1) =             0.112292
    P(2) =             0.982288
    P(3) =             0.223779
    P(4) =            0.0168439
    P(5) =             0.147343
    P(6) =             0.745930
    P(7) =            0.0561463
    P(8) =             0.491144
```

I'm not sure if this helps. Maybe Craig can answer better.

Subject: Re: mpfit: multivariate fit
Posted by Dave[3] on Tue, 08 May 2007 14:23:57 GMT
View Forum Message <> Reply to Message

Thanks hradilv, that does help. I also found that I could get it to
work if I expand the dimensions and make everything a vector prior to
fitting. I'd be interested in knowing from Craig if your fix is ok as
I greatly prefer it to the fiddling below.

```
function trans, K, X=x, Y=y, err=err, forward=fw

  kk = reform(K, 3, n_elements(k)/3)
  xx = reform(x, 3, n_elements(x)/3)

  model = KK # xx

  model = reform(model, n_elements(model))

  if keyword_set(fw) then return, model else return, (y-model)/err
end

; MAIN

; Attempt to estimate the transformation matrix given a set
; of observed cartesian vectors and a set of known cartesian
; vectors.

n = 1000 ; number of 'observations'

v = [1.0d, 0.15, 0.5] ; template vector
xyz_obs = dblarr(3,n) ; observations
for i=0, n-1 do $
  xyz_obs[*,i] = v+0.01*randomn(seed,3)

xyz_known = dblarr(3,n) ; known values (trivial scaling)
for i=0, n-1 do $
  xyz_known[*,i] = v*2.0d

; Estimate the transformation matrix, T
T0 = identity(3, /DOUBLE) ; initial guess transformation matrix
f = {x: reform(xyz_obs,3*n), $
     y: reform(xyz_known,3*n), $
     err: 0.01}
T = mpfit('trans', reform(T0,3*3), functargs=f, COVAR=S2, /NOCATCH)
T = reform(T, 3, 3)

; Residuals
res = trans(reform(T,3*3), X=reform(xyz_obs,3*n), /FORWARD) - $
  reform(xyz_known,3*n)
```

```
T_known = identity(3) * 2.0d
res_known = trans(reform(T_known,3*3), X=reform(xyz_obs,3*n), /
FORWARD) - $
  reform(xyz_known,3*n)
```

Subject: Re: mpfit: multivariate fit
Posted by David Fanning on Tue, 08 May 2007 14:45:30 GMT
View Forum Message <> Reply to Message

Dave writes:

> Thanks hradilv, that does help. I also found that I could get it to
> work if I expand the dimensions and make everything a vector prior to
> fitting. I'd be interested in knowing from Craig if your fix is ok as
> I greatly prefer it to the fiddling below.

I don't speak for Craig, but I'm going to guess that Vince's
"fix" is not going to go over well. :-)

Here is the line that breaks:

```
   fjac(0,j) = (fp-fvec)/h(j)
```

When it breaks, fjac is a [300,9] array. And what
you are trying to stuff into it is a [100,3] array.
But, j is 7, which means you are trying to do something
like:

```
   fjac[0:99, 7:9] = thisThing
```

Of course, that "9" is what is causing the grief.

I think Vince's fix just makes it possible to stuff
the data into fjac *somewhere*. This seems a dubious
proposition when you are looking for good results. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: mpfit: multivariate fit
Posted by Craig Markwardt on Fri, 11 May 2007 08:05:58 GMT
View Forum Message <> Reply to Message

Dave <Confused.Scientist@gmail.com> writes:

```
>  Thanks hradilv, that does help.  I also found that I could get it to
>  work if I expand the dimensions and make everything a vector prior to
>  fitting.  I'd be interested in knowing from Craig if your fix is ok as
>  I greatly prefer it to the fiddling below.
>
>  function trans, K, X=x, Y=y, err=err, forward=fw
>
>    kk = reform(K, 3, n_elements(k)/3)
>    xx = reform(x, 3, n_elements(x)/3)
>
>    model = KK # xx
>
>    model = reform(model, n_elements(model))
>
>    if keyword_set(fw) then return, model else return, (y-model)/err
>  end
```

David Fanning is right. You can't simply recode the function like you did. According to the documentation:

```
  ;  In general there are no restrictions on the number of dimensions in
  ;  X, Y or ERR.  However the deviates *must* be returned in a
  ;  one-dimensional array, and must have the same type (float or
  ;  double) as the input arrays.
```

The output of your function *must* be one dimensional :-)
Just reform() it before you return it.

By the way, I would say that fitting all 9 matrix components is a sure path to problems. At the very least, you should enforce the constraint that the matrix must be symmetric. If there is a simple rotation (no scale or skew factors), then I would suggest using Euler angles, or even better, use quaternions and fit the quaternion components. You can use QTVROT in my library to apply a quaternion rotation to a 3-vector.

Good luck!
Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------