

```

1  !Tweak-Begin.F95
2  !2018.10.09.1015cdt JMS- Introducing: "Tweak-Engine",
3  !*****
4  !                                     &: Global Variables.
5  !
6  ! Jeffrey M Setterholm, 8095 230th St. E., Lakeville, Minnesota 55044, USA
7  ! I have authored the four Fortran *.f95 source code files listed below.
8  ! I hereby place these four files:
9  ! Tweak-Begin.f95, Tweak-Engine.F95, Tweak-User.F95, & Tweak-Vis.f95
10 ! and the algorithms which are demonstrated therein,
11 ! in the public domain (a.k.a.: "free").
12 !Disclaimer:
13 !*****
14 !                               Individual cognition is always flawed,
15 !                               including yours & mine.
16 !                               - So: -
17 !                               Use this code at your own risk.
18 !*****
19 !
20 !This code realizes my present understanding of:
21 !
22 !           Best-Fitting the Continuous Parameters of
23 ! Your      : Linear/Non-Linear Algorithmic Models of
24 ! Your      : Arbitrarily Large Datasets
25 !           with Extreme Accuracy.
26 !
27 !When YouTweak.f95 is tailored for your purposes:
28 !
29 !           You can determine your math model's adequacy,
30 !           or need for improvement, in accounting for your data.
31 !
32 !If you enjoy searching, -and-
33 ! when these algorithms are subsequently perfected,
34 ! you'll find that:
35 !
36 !           Mother nature's error evaluations of math models
37 !           are delightfully impartial, non-intrusive, and deeply insightful...
38 !           much like the smiles & frowns of the great teachers in the Himalayas.
39 !
40 !*****
41 !
42 !Tweak-Begin.f95 organizes the key variables within Tweak.exe
43 !An overview of the Engine (timestamps & sizes are approximate):
44 !
45 !           Lines      Bytes
46 ! 10/08/2018 05:30 AM      3,560 Tweak.gui      <- Absoft's GUI
47 ! 10/08/2018 11:54 AM    ~470  23,907 Tweak-Begin.f95  <- This file
48 !                                     Module TweakRec
49 !                                     Module KPrec
50 !                                     Module LZrec
51 !                                     Module MDrec
52 !                                     Module MDrec
53 !                                     Subroutine TweakAllocateAll
54 ! 10/08/2018 08:11 AM    ~1260  60,540 Tweak-Engine.F95  Includes:
55 !                                     Program Tweak
56 !                                     Subroutine EvalFit(RSSL, iP)
57 !                                     Subroutine PrintIter(iP)
58 !                                     Subroutine RSSPartials(jpUL)
59 !                                     Subroutine DatapointPartials
60 !                                     Subroutine Invert(N, A, ValMin, iRank, ...
61 !                                     Subroutine PrintA(N, A, Noise, iRank, ...
62 !                                     Subroutine SelectStepMult(iP)
63 !                                     Function om(Value1, Value0, iP)
64 !                                     Subroutine FloatWrite(R16In, a40out)
65 !                                     Subroutine FDate23(Date)
66 !                                     Subroutine Beamer(n, nTot)
67 ! 10/08/2018 08:14 AM    ~400   19,167 Tweak-User.F95  <- Your Model goes here
68 !                                     Subroutine YouTweak(Mode)
69 !                                     Subroutine YouTweak3D
70 !

```

```

71 !10/08/2018 07:22 AM ~850 50,436 Tweak-Vis.f95 <- 3D Visualization
72 ! Subroutine Draw3dJTL(iColor,...
73 ! Subroutine Morph3dJTL(TLin,iP)
74 ! Function Log10LF(Value1,iP)
75 ! Module CharDef
76 ! ~an English character vector font
77 ! Subroutine AlphaJS(cLabelL,PosLLCq...
78 ! Subroutine Model7DoF(Xyzh,Rpyh,S,...
79
80 !10/08/2018 12:27 PM 1,037,824 Tweak.exe
81 !10/08/2018 11:58 AM 1,621 Tweak-User.nml <- For mods @ runtime
82
83 !10/08/2018 12:03 PM 141,391 Tweak.txt <- Reports the process
84 !10/08/2018 06:18 AM 1,301 Tweak-Params.txt <- Reports parameters
85
86 !10/08/2018 12:03 PM 1,412,272 Tweak-3dDraw.3dv <- 3D output.
87
88 ! ~9 File(s) ~3,089,439 bytes
89
90 !Subdirectory /Images has 23 selfies of five solution trajectories.
91
92 !*****
93
94 !Five "modules" follow herein. The summarizing 'use' statements are:
95 !--- globals
96 !use Tweakrec, only: jPhase, jMode, cVersion, cDateTime & !Tweak's FYI
97 ! , jB0n, jUnClamp, jStepMult, jPrev & !Solution strategy
98 ! , jPU10, jPU, jPD, cFloat40 & !Printout-Alphanumeric
99 ! , jPU3d, TLrec, TL, TLprev, TL2, TLsave & !Printout-3D
100 ! , jTLmorph, TLiter, omj Save & !
101 ! , jItertot, jIter, jDone & !Iteration control
102 ! , RSS, RSSbase, Weight, Delta, offon & !Tweaking & errors
103 ! , AbsDet, NoiseFloor, iRank, kPChanged & !Inverter outputs
104 ! , jMintot, jMin, StepMult, omj & !Minimization passes
105 ! , B, BtB, BtZ & !Allocated matrices
106 ! , What, Why, How, Who, When, Where1 & !Project context
107 ! , jUserPhase, jUserConfig, cjUserFile & !Use in YouTweak()
108 ! , TweakNml !Runtime reconfig.
109 !Use KPrec, only: Kptot, Kp, Kp2, Pr, Pu, PstepMag !Parameters -to fit to-
110 !Use LZrec, only: Lztot, Lz, Zr, Zu, Zu2, Z0 !Outputs - of your -
111 !use MRec, only: Mdtot, Md, MdMax, Datae !Dataset
112 !use NXrec, only: Nxtot, Nx, Xr !Independent variables
113 ! ^^:These are allocated records.
114 ! ^^:At the outset you size these four values.
115 !---
116
117 !Table of Contents:
118 !Module TweakRec !Program control and various summary values:
119 !Module KPrec !Parameters:
120 !Module LZrec !Outputs:
121 !Module MRec !dataset storage & access:
122 !Module MRec !dataset storage & access:
123 ! &
124 !Subroutine TweakAllocateAll !Allocates Tweak's arrays.
125 !-----7 9
126 !////////////////////
127
128 Module TweakRec !Program control and various summary values: !use TweakRec
129 ! = j- integers & various reals
130
131 !Tweak's FYI
132 character::jPhase*80 !Marks key phases of the fitting process. Purely: FYI
133 ! For Your Information
134 ! ="0: Allocates records based on Kptot, Lztot, Mdtot, & Nxtot" c
135 ! in YouTweak(0)
136 ! ="1: You quantify your system model in the allocated arrays" c
137 ! in YouTweak(1)
138 ! YouTweak(2, 3, & 4) - exercised by the Tweak engine.
139 ! ="2: Top-of-each iteration: establish baseline residual errors" c
140 ! ="3: Populate BtB(Kptot, Kptot) and BtZ(Kptot)" c

```

```

141 !           ="4: Normalize and invert BtB(Kptot, Kptot)"c
142 !           ="5: Inverters stepsize: Pr.Pistep = BtBinv(denormalized)*BtZ"c
143
144 !           ="3, 4, & 5: RSSPartials uses RSS error gradient instead."c
145
146 !           ="6: Find an error-reducing actual stepsize"c
147 !           ="7: Iterative pass done. Your intervention opportunity:"c
148 !                                     in YouTweak(5)
149 !           ="8: Tweak is done interating. Export the results & close"c
150 !                                     in YouTweak(6)
151
152 integer*4::jMode !Interface of Tweak() to YouTweak().           Purely: FYI
153 !           = 0: Define Nxtot, Lztot, Mdtot, & Kptot:
154 !           = 1: Initialize Xr(), Zr(), Dr(), Pr(), & Datae():
155 !           = 2: Md=0 ~ "Rewind" your dataset:
156 !           = 3: Md=Md+1 - Datae(*, Md) -> Xr.Xdata and Zr.Zdata:
157 !           = 4: Exercise your system. Compute Zr.Z using Pr.P:
158 !           = 5: End of iterative pass -intervention opportunity:
159 !           = 6: Iterating done. Final printout(s) opportunity:
160
161 character::cVersion*60 & !Present software version
162 !           ="Tweak.Exe, version: 1.0, 2018.10.09, JMS, Public Domain"c
163
164 character::cDateTime*23!Current date & time.
165
166 !Solution strategy:
167 integer*4::jBOn      = 3 !Iteration# to start calling DatapointPartials()
168 !                   For jIter<jBOn the call is to RSSPartials() instead
169 !                   = 1 :Will probably work well for linear models.
170 !                   > 1 :A few iterations using RSSPartials()
171 !                   hastens the convergence of non-linear models.
172
173 integer*4::jUnClamp = 3 !Number of iterations to restrict Pr.Pistep to <= 1.
174
175 integer*4::jStepMult= 2!Minimization algorithm - computes StepMult.
176 !           = 1! - StepMult=1.0 .e.g.: using inverter's result
177 !           = 2! - calls SelectStepMult()
178
179 character*9::omj & !Expresses the value1/Value0 ratio of positive numbers
180 !           ='undefined' ! in exponential form, & places the signed exponent
181 !                   ! on the left side of the result.
182 !                   ! Exponents > 0 represent ratios >= 1.
183 !                   ! Exponents <=0 represent ratios < 1.
184 !                   ! 1. > mantissa >= .1, except when Value1 = 0.
185 !                   ! "__sme.nnn"
186 !                   ! "__smme.nnn"
187
188 integer*4::jPrev     =-1!Use previous parameter values in Tweak-Params.txt?
189 !                   =-1: no
190 !                   = 0: ask
191 !                   = 1: yes
192 !                   Ordinarily you'll solve each problem from scratch.
193 !                   For problems whose iteration time is more than a few
194 !                   minutes, easy resumption is convenient.
195
196 !Printout-Alphanumeric:
197 integer*4::jpu10    =10 !Unit number- for Tweak's parameter reads & writes
198 !                   to/from "Tweak-Params.txt"
199 integer*4::jpu      =11 !Unit number- Output of general information
200 !                   = 6 : outputs to DOS screen
201 !                   >=11 : writes/appends "Tweak.txt"
202 integer*4::jpd      = 4 !Print Detail level, use: "if(jpd >= ___)"
203 !                   = 0 : Disabled, e.g: when jpu<6
204 !                   = 1 : User selected important information/program controls
205 !                   = 2 : Enables screen/keyboard/program interactions
206 !                   = 3 : Report/summarize- final iteration
207 !                   = 4 : - each iteration
208 !                   = 5 : - some main program info.
209 !                   = 6 : - completed matrices & minimization
210 !                   = 7 : - some subroutine details

```

```

211 !           = 8 :           - in-depth details
212   character:: cFloat40*40 !used for: call FloatWrite(R16In, cFloat40) printing
213
214 !Printout-3D: -----
215 !YouTweak can interface with 3D visual data collector Draw3dJTL().
216 ! A minimal interface follows.
217 !If no calls to Draw3dJTL() are made, then TweakVis.f95 isn't needed
218 ! to compile & link Tweak.exe.
219   integer*4:: jpu3d = 13 !Unit number- for exporting 3D viewing coordinates.
220
221   integer*4:: jTlmorph = 1!Morph3dJTL() mode control.
222   !           Morph3dJTL() centralizes the arbitrary rescale
223   !           (~morphing) of TL.XYZ plot points.
224   !           Example uses:
225   !           = 0: Z-axis- no change
226   !           !           (<, -1.) [-1., 1.] ((1., >) :Z
227   !           = 1: - -log10(-Z) | linear | +log10(+Z)
228   !           = 2: - 2.-1./Z | linear | -2.+1./Z
229   !           = 3: - User defined morph of X, Y, & Z.
230
231   type:: TLrec           !"Thick Line" record
232   integer*4:: iC         ! DOS color of the line [0,15] to the next dot
233   !           = 0: move-to without drawing
234   !           = -1: closes the file
235   real*4   :: Th         ! Line thickness [1.0 - 20.0] & width           in pixels
236   real*16  :: XYZ(3)    ! (X, Y, Z) "next dot"
237   end type TLrec;
238   type(TLrec):: TL       !Draw3dJTL- Your "next dot" input
239   type(TLrec):: TLprev  !           - the most recent dot added
240   type(TLrec):: TL2     !           - "next dot" used by AlphaJS
241   type(TLrec):: TLsave  !A place to save TLprev during dot digressions.
242   type(TLrec):: TLiter(0:100)
243   character*9:: omjSave(0:100) !omj values of the first 100 iterations
244   !-----
245
246 !Iteration control:
247   integer*4:: jItertot=30!Fit iterations- total
248   integer*4:: jIter    !           - current #
249   integer*4:: jDone    != 0: will do another iteration
250   !> 0: reduced error not found.
251
252 !Evaluation:
253   real*16  :: RSS       !Root-Sum-Squared- present
254   real*16  :: RSSbase  !           - previous iteration
255 !Numerical Partial Derivative default values:
256   real*16  :: Weight   = 1._16 !Nominal output weighting factor
257   !           = 0._16 !           - set individually
258   real*16  :: Delta    = 1.e-6_16 !Numerical partial tweak-
259   !           - adjusted in YouTweak(5)
260   !           = 0._16 !           - set individually
261   real*16  :: offon    = 1._16 !Defaults to all- on
262   !           = 0._16 !           - off- set individually
263
264 !Inversion output values:
265   real*16  :: AbsDet    !Determinant- absolute value
266   real*16  :: NoiseFloor !Inversion noise floor
267   integer*4:: iRank     !BtB's rank, < Kptot if linearly dependent
268   integer*4:: kPChanged !# of parameters whose values changed.
269
270 !Minimization passes to determine the step multiplier:
271   integer*4:: jMintot =20
272   integer*4:: jMin
273   real*16  :: StepMult  !Pr. Pnew = Pr. Pbase + SetpMult * Pr. Pstep
274
275   real*16  , allocatable:: B( , , ) ! (Lztot, Kptot) Numerical partials array
276   real*16  , allocatable:: BtB( , , ) ! (Kptot, Kptot) Outer product array
277   real*16  , allocatable:: BtZ( , ) ! (Kptot
278
279 !Project Context: Consider having your dataset(s) define these:
280   character:: What *60 ! & !Title = "Title this particular parameter fit."c

```



```

421 !---
422 implicit none
423 !---
424 integer*4::iAlloc
425 !---
426 if(Kptot<=0) then; write(6, "(1x, 'Tweak halt! Kptot=', i9)") Kptot ; stop; endif
427 if(Lztot<=0) then; write(6, "(1x, 'Tweak halt! Lztot=', i9)") Lztot ; stop; endif
428 if(Mdtot<=0) then; write(6, "(1x, 'Tweak halt! Mdtot=', i9)") Mdtot ; stop; endif
429 if(Nxtot<=0) then; write(6, "(1x, 'Tweak halt! Nxtot=', i9)") Nxtot ; stop; endif
430 !
431 allocate(Pr(Kptot), stat=iAlloc) !Parameters array
432 if(iAlloc /= 0) then
433     pause 'TweakAllocateAll: Pr(Kptot) allocation error. `Enter` to exit.'
434     stop
435 endif!(iAlloc/=0)
436 Pr(1:Kptot)=P0; Pr.Pname=char(32)
437 !
438 allocate(Zr(Lztot), stat=iAlloc) !Outputs array
439 if(iAlloc /= 0) then
440     pause 'TweakAllocateAll: Zr(Lztot) allocation error. `Enter` to exit.'
441     stop
442 endif!(iAlloc/=0)
443 Zr(1:Lztot)=Z0; Zr.Zname=char(32)
444 !
445 allocate(Datae(0:Nxtot+Lztot, Mdtot), stat=iAlloc) !Data array
446 if(iAlloc /= 0) then
447     pause 'TweakAllocateAll: Datae() allocation error. `Enter` to exit.'
448     stop
449 endif!(iAlloc/=0)
450 Datae=0._16
451 !
452 allocate(Xr(0:Nxtot), stat=iAlloc) !Independent variables array
453 if(iAlloc /= 0) then
454     pause 'TweakAllocateAll: Xr(Nxtot) allocation error. `Enter` to exit.'
455     stop
456 endif!(iAlloc/=0)
457 Xr(0:NxTot)=X0; Xr.Xname=char(32)
458 !
459 allocate(B(Lztot, Kptot), stat=iAlloc) !B: Numerical partials array
460 if(iAlloc /= 0) then
461     pause 'TweakAllocateAll: B(Lztot, Kptot) alloc. error. `Enter` to exit.'
462     stop
463 endif!(iAlloc/=0)
464 B=0._16
465 !
466 allocate(BtB(Kptot, Kptot), stat=iAlloc) !Bt*B: Outer product array
467 if(iAlloc /= 0) then
468     pause 'TweakAllocateAll: B(Lztot, Kptot) alloc. error. `Enter` to exit.'
469     stop
470 endif!(iAlloc/=0)
471 BtB=0._16
472 !
473 allocate(BtZ(Kptot), stat=iAlloc) !Bt*Z array
474 if(iAlloc /= 0) then
475     pause 'TweakAllocateAll: BtZ(Kptot) allocation error. `Enter` to exit.'
476     stop
477 endif!(iAlloc/=0)
478 BtZ=0._16
479 ! ----- Array allocations completed.
480
481 End Subroutine TweakAllocateAll
482

```

return