```
  1     !Polyize.f95
  2     !2021.04.06.1640cdt JMS- Traveler2/Athlon64/WinXPPro/APF9.0/Ogl1.2.1
  3                                                          !.1630 fixed bug in line #337
  4     !Table of Contents:
  5     !Module PolyDef
  6     ! Program Main
  7     !  Subroutine ImportBinFile(iP)
  8     !  Subroutine BinFileCsvReport(iP)
  9     !  Subroutine MatrixCsvWriter
  10    !  Subroutine ValuesSequence(iP)
  11    !  Subroutine ExampleIputs(iP)
  12    !  Subroutine DataInToCoesOut(iP)
  13    !  Subroutine CoesInToDataOut(iP)
  14    !  Subroutine CoefficientCount(iP)
  15    !  Subroutine DisclaimerEtc(iP)
  16    !-----------------------------------------------------------------------------7 9
  17    Module PolyizeDef
  18    !use PolyDef                                                    !JMS 2021.03.27
  19      integer*4            ::nNameChars!# of characters in the filename.
  20      character            ::cBinFilein*79!Filename of the binary input data
  21
  22     type ::  PolyizeRec
  23      integer*4            ::nVT       !Variables- total number [1,2,...,10]
  24      integer*4            ::nOrdMax   !Largest of the nOrd()'s
  25      integer*4            ::nCT       !Coefficients total number
  26      integer*4            ::nOrd(10)  !Even-orders of polynomial of each variable
  27      integer*4            ::iPolyClass!Coefficients type:
  28      integer*4            ::iCoeDigits!Number of digits for exact values
  29      integer*4            ::iCoeDeciPl!Number of decimal places
  30      real*16              :: CoeDiv   !Divisor of Coe()- real
  31      real*16              :: CoeMaxScaled!MaxAbs coefficient in .bin-
  32      integer*4            ::iPowDigits!Number of digits for exact values
  33      integer*4            ::iPowDeciPl!Number of decimal places
  34      real*16              :: PowDiv   !Divisor of Coe()- real
  35      real*16              :: PowMaxScaled!MaxAbs coefficient in .bin-
  36     end type PolyizeRec
  37     type(PolyizeRec)::Pr !This is the .bin file's front-end binary information.
  38
  39    ! --nVT and nOrd(1:nVT) determine nCT and the size of these arrays:
  40      real*16  ,allocatable::X( :,:)    !Coefficient values in [-1,1]
  41    !                  (0:nOrdMax,nVT)
  42      integer*4,allocatable::nVnC(:,:)  !Indexes & powers lookup table
  43    !                          (nVT,nCT)
  44      real*16  ,allocatable:: Coe(:,:)  !The matrix to be inverted
  45    !                          (nCT,nCT)
  46      real*16  ,allocatable:: Pow(:,:)  !The matrix to be inverted
  47    !                          (nCT,nCT)
  48      real*16  ,allocatable:: DataSeq(:)!The user's data vector
  49    !                               (nCT)
  50      real*16  ,allocatable:: CoeSeq(:) !The user's data vector
  51    !                             (nCT)
  52
  53    ! --The following often-used variables are copied out of the Pr record:
  54      integer*4            ::nVT       !Variables- total number [1,2,...,10]
  55      integer*4            ::nOrdMax   !Largest of the nOrd()'s
  56      integer*4            ::nCT       !Coefficients total number
  57      integer*4            ::nOrd(10)  !Even-orders of polynomial of each variable
```

```
  58      real*16              :: CoeDiv   !Divisor of Coe()- real
  59      real*16              :: PowDiv   !Divisor of Coe()- real
  60      integer*4            ::iDecPlaces=12
  61      integer*4            ::iDomain=1
  62      character*1    ::AugHex(0:20)=(/"0","1","2","3","4","5","6","7","8","9","a" &
  63                                     ,"b","c","d","e","f","g","h","i","j","k"/)
  64    !---------------------------------------
  65    ! --Runtime environment variables:
  66      character::cIniFIle*79  ='Polyize.ini'c
  67      character::cOutFile*79  ='Polyize.txt'c
  68    ! --Unit#'s for input/output:
  69      integer*4::uD  =14 !: Open(uD...) for reading & writing documentation
  70      integer*4::uP  =13 !: write(uP,...) Print to- file
  71      integer*4::uS  = 6 !: write(uS,...) Print to- screen
  72      integer*4::uT  = 0 ! [0,uP, or uS] temporarily requested outputs
  73    !                     if(uT>5) write(uT,...)
  74    ! --Memory allocation variables:
  75      logical  ::LExists
  76      integer*4::iAlloc       !Allocation flag: used with allocatable arrays
  77      integer*4::iPolyAlloc =0!:unAllocated; =1:Allocated
  78    ! --Timeing variables:
  79      real*16              ::TimeEstMin
  80      character            ::Chrono(0:9)*22!TimeStamps:
  81    !                                 (0):               for differential use
  82    !                                 (1):               Start
  83    !                                 (2):Data entry    - done
  84    !Contains
  85    End Module PolyizeDef
  86    !-------------------------------------------------------------------------------7 9
  87
  88    !-------------------------------------------------------------------------------7 9
  89    Program Main                                                        !2021.04.06.1640
  90    !2021.04.06.1640cdt JMS- Opens an output file & calls PolyGen()
  91     use PolyizeDef
  92    !--End Globals
  93     implicit none
  94     !--Arguments
  95     !--Internals
  96      integer*4::WhatNext != 0:end
  97    !                     = 1:provide a data template
  98    !                     = 2:compute poly coefficients
  99      character::cL*80
 100     !--EndDefs--------------------------
 101      uP = 13  ! = 6 ...for screen only (e.g. for no writes to a hard drive.)
 102                                                  call DateAndTime22(Chrono(1))
 103      !--- Open output
 104      if(uP.gt.12) then
 105        open( uP,file=cOutFile,action='write',err=10)
 106          cL='Opening output file: '//cOutFile    ;call TimeStamp(cL,chrono(1),uP)
 107          goto 20
 108      endif !uP>12
 109   10     uP=6
 110          cL='The computer monitor will be the output. 'c ;call WSF(cL,uP)
 111          cL=''c                                          ;call WSF(cL,uP)
 112   20 continue
 113      cL=''c                                           ;call WSF(cL,uP)
 114      cL='This is:    "Polyize.exe",         By: Jeff Setterholm'c
```

```
115                                                                  ;call WSF(cL,uP)
116      cL='        Version 1.1 2021.04.06    Lakeville, MN 55044 USA'c
117                                                                   call WSF(cL,uP)
118      cL=''c                                                      ;call WSF(cL,uP)
119
120      call DisclaimerEtc(uP) !/////////////////////
121
122      call ImportBinFile(uP) !/////////////////////
123
124      cL=''c                                                      ;call WSF(cL,uP)
125                                                    call DateAndTime22(Chrono(2))
126      write(cL,"('.bin file content - loaded, parsed, &  ready.')")
127                                                   call TimeStamp(cL,Chrono(2),uP)
128  !   call ElapsedTime(Chrono(1),Chrono(2),Chrono(9),TimeEstMin,0)
129  !   call TimeStamp( & !                                    |
130  !   "                                       task`s time: "c       ,Chrono(9),uP)
131
132    50 continue
133       cL=''c                                                    ;call WSF(cL,uP)
134       write( 6,"(' What next?   -3 : Write the `CoefficientCount` table.')")
135       write( 6,"(12x,'  -2 : *.bin -> *.csv `Comma Separated Values` file.')")
136       write( 6,&
137      "(12x,'  -1 : Write `ValuesSequenceInfo.txt`,`DataIn.txt`,&`PolysIn.txt.`')")
138       write( 6,"(12x,'   0 : Quit.')")
139       write( 6,"(12x,'  +1 : Process your `DataIn.txt`->`CoesOut.txt`.')")
140       write( 6,"(12x,'  +2 : Process your `CoesIn.txt`->`DataOut.txt`.')")
141       write( 6,"('Your choice = '\)")
142       read(5,*) WhatNext
143       write(uP,"('WhatNext=',i2)") WhatNext                     ;call WSF(cL,uP)
144                                                    call DateAndTime22(Chrono(3))
145       select case(WhatNext)
146         case(-3) ! = -3 : Write the 'Coefficient Count Table'.
147          call CoefficientCount(uP)
148
149         case(-2) !=-2 : *.bin -> *.csv `Comma Separated Values` file.
150          call BinFileCsvReport(uP) !/////////////////
151                                                    call DateAndTime22(Chrono(4))
152          call ElapsedTime(Chrono(3),Chrono(4),Chrono(9),TimeEstMin,0)
153          call TimeStamp( & !                                    |
154          "                                    task`s time: "c    ,Chrono(9),uP)
155
156         case(-1) != -1 :`ValuesSequenceInfo.txt`,'DataIn.txt,&'CoesIn.txt.
157          call ValuesSequence(uP) !//////////////////
158          call ExampleIputs(uP)   !//////////////////
159
160         case( 1) ! = +1 : Process your `DataIn.txt`->`CoesOut.txt`.
161          call DataInToCoesOut(uP)  !/////////////////
162
163         case( 2) ! = +2 : Process your `CoesIn.txt`->`DataOut.txt`.
164          call CoesInToDataOut(uP)  !/////////////////
165
166         case default !Quit
167          goto 100
168
169       end select!(WhatNext)
170       goto 50
171
```

```
  100 continue
     if(iPolyAlloc>0) then
       deallocate(X)
       deallocate(nVnC)
       deallocate(Pow)
       deallocate(Coe)
       deallocate(DataSeq)
       deallocate(CoeSeq)
       iPolyAlloc = 0
       if(uP>5) write(uP,"(1x,'Arrays deallocated')")
     endif!iPolyAlloc>0

     if(uP.gt.12) close(uP)
  !  pause 'Press enter to continue - thus closing this output screen & exiting.'
  End Program Main
  !-------------------------------------------------------------------------------7 9

  Subroutine ImportBinFile(iP)
  !2021.03.30.1545cdt JMS- Import the polynomial matrices to use.
   use PolyizeDef
  !--End Globals
   implicit none
   !--Arguments
    integer*4::iP           !Write enable>5: write(iP,...)
  !--Internals
    integer*4::i,iFileBytes,nB
    character::cL*80
    !--EndDefs---------------------------
     open(unit=14,file=cIniFIle,action='read')
       read(14,*) cBinFileIn
     close(14)

     nNameChars=len_trim(cBinFileIn)
     write(Cl,"('Reading: ',79(a1,:))")(cBinFileIn(i:i),i=1,nNameChars)
                                                                  call WSF(cL,uP)

     open(unit=14,file=cBinFileIn,form='binary',action='read')
       read(14)  Pr                          ! ...a known read

       nVT     = Pr.nVT
       nOrdMax = Pr.nOrdMax
       nCT     = Pr.nCT
       nOrd    = Pr.nOrd
       CoeDiv  = Pr.CoeDiv
       PowDiv  = Pr.PowDiv

  !    --Allocate the arrays:
       allocate(X(0:nOrdMax,nVT),stat=iAlloc);       if(iAlloc /= 0) then
         pause 'X(0:nOrdMax,nVT) alloc. error. `Enter` to exit.'     ;stop; endif
       X        = 0._16                               ;iPolyAlloc = iPolyAlloc+1
       allocate(nVnC(nVT,nCT),stat=iAlloc);          if(iAlloc /= 0) then
         pause 'nVnC(nVT,nCT) alloc. error. `Enter` to exit.'        ;stop; endif
       nVnC      = 0                                 ;iPolyAlloc = iPolyAlloc+1
       allocate(Pow(nCT,nCT),stat=iAlloc);           if(iAlloc /= 0) then
           pause 'Pow(nCT,nCT) alloc. error. `Enter` to exit.'       ;stop; endif
       Pow       = 1._16                             ;iPolyAlloc = iPolyAlloc+1
       allocate(Coe(nCT,nCT),stat=iAlloc);           if(iAlloc /= 0) then
```

```
229           pause 'Coe(nCT,nCT) alloc. error. `Enter` to exit.'       ;stop; endif
230       Coe      = 1._16                                    ;iPolyAlloc = iPolyAlloc+1
231 !     --Arrays needed to finish the .bin file read have been allocated.
232 !     --Now allocate the Data Sequence & Coefficient Sequence vectors:
233       allocate(DataSeq(nCT),stat=iAlloc);           if(iAlloc /= 0) then
234           pause 'DataSeq(nCT) alloc. error. `Enter` to exit.'       ;stop; endif
235       DataSeq  = 1._16                                    ;iPolyAlloc = iPolyAlloc+1
236       allocate(CoeSeq(nCT),stat=iAlloc);            if(iAlloc /= 0) then
237           pause 'CoeSeq(nCT,nCT) alloc. error. `Enter` to exit.'    ;stop; endif
238       CoeSeq   = 1._16                                    ;iPolyAlloc = iPolyAlloc+1
239
240     write(  cL,  &
241     "('Binary file data write/read sequence- integer*4`s & real*16`s:')")
242                                                                   call WSF(cL,uP)
243     nB = sizeof(nVT);     iFileBytes =            nB              ;write(cL, &
244     "(i12,', i* 4 nVT      -# of independent variables<=10    fixed size')") nB
245                                                                   call WSF(cL,uP)
246     nB = sizeof(nOrdMax); iFileBytes = iFileBytes+nB              ;write(cL, &
247     "(i12,', i* 4 nOrdMax  -Highest order variable    <=20     `     `'    )") nB
248                                                                   call WSF(cL,uP)
249     nB = sizeof(nCT);     iFileBytes = iFileBytes+nB              ;write(cL, &
250     "(i12,', i* 4 nCT      -Number of polynomial coefficients   `     `'    )") nB
251                                                                   call WSF(cL,uP)
252     nB = sizeof(nOrd);     iFileBytes = iFileBytes+nB             ;write(cL, &
253     "(i12,', i* 4 nOrd(10) -Order of each variable  (1:nVT)     `     `'    )") nB
254                                                                   call WSF(cL,uP)
255     nB = sizeof(Pr.iPolyClass);iFileBytes = iFileBytes+nB         ;write(cL, &
256     "(i12,', i* 4 iPolyClass-=1:exact, =2:unscaled              `     `'    )") nB
257                                                                   call WSF(cL,uP)
258     nB = sizeof(Pr.iCoeDigits);iFileBytes = iFileBytes+nB         ;write(cL, &
259     "(i12,', i* 4 iCoeDigits- +***.  <-count                    `     `'    )") nB
260                                                                   call WSF(cL,uP)
261     nB = sizeof(Pr.iCoeDeciPl);iFileBytes = iFileBytes+nB         ;write(cL, &
262     "(i12,', i* 4 iCoeDeciPl-number of digits after .           `     `'    )") nB
263                                                                   call WSF(cL,uP)
264     nB = sizeof(Pr.CoeDiv);   iFileBytes = iFileBytes+nB          ;write(cL, &
265     "(i12,', r*16 CoeDiv   -Coe[,] matrix dividing factor       `     `'    )") nB
266                                                                   call WSF(cL,uP)
267     nB = sizeof(Pr.CoeMaxScaled);   iFileBytes = iFileBytes+nB    ;write(cL, &
268     "(i12,', r*16 CoeMaxScaled -AbsMax Coe[,] in .bin           `     `'    )") nB
269                                                                   call WSF(cL,uP)
270     nB = sizeof(Pr.iPowDigits);iFileBytes = iFileBytes+nB         ;write(cL, &
271     "(i12,', i* 4 iPowDigits- +***.  <-count                    `     `'    )") nB
272                                                                   call WSF(cL,uP)
273     nB = sizeof(Pr.iPowDeciPl);iFileBytes = iFileBytes+nB         ;write(cL, &
274     "(i12,', i* 4 iPowDeciPl-number of digits after .           `     `'    )") nB
275                                                                   call WSF(cL,uP)
276     nB = sizeof(Pr.PowDiv);   iFileBytes = iFileBytes+nB          ;write(cL, &
277     "(i12,', r*16 PowDiv   -Pow[,] matrix dividing factor       `     `'    )") nB
278                                                                   call WSF(cL,uP)
279     nB = sizeof(Pr.PowMaxScaled);   iFileBytes = iFileBytes+nB    ;write(cL, &
280     "(i12,', r*16 PowMaxScaled   -AbsMax Pow[,] in .bin         `     `'    )") nB
281                                                                   call WSF(cL,uP)
282     nB = sizeof(X);        iFileBytes = iFileBytes+nB             ;write(cL, &
283     "(i12,', r*16 X[0:nOrdMax,nVT] -Variable values to use    allocated'  )") nB
284                                                                   call WSF(cL,uP)
285     nB = sizeof(nVnC);     iFileBytes = iFileBytes+nB             ;write(cL, &
```

```
286      "(i12,', i* 4 nVnC[nVT   ,nCT] -Values combinations sequence  `'      )") nB
287                                                                    call WSF(cL,uP)
288      nB = sizeof(Coe);      iFileBytes = iFileBytes+nB              ;write(cL, &
289      "(i12,', r*16 Coe[ nCT   ,nCT] -The coefficients solver       `'      )") nB
290                                                                    call WSF(cL,uP)
291      nB = sizeof(Pow);      iFileBytes = iFileBytes+nB              ;write(cL, &
292      "(i12,', r*16 Pow[ nCT   ,nCT]      Coe[,] = inverse(Pow[,])  `'      )") nB
293                                                                    call WSF(cL,uP)
294      write(  cL,"(i12,', Total Bytes')") iFileBytes;  call WSF(cL,uP)
295      cL=''c                                                       ;call WSF(cL,uP)
296
297      write(cL,"('All arrays needed have been successfully allocated')")
298                                                                    call WSF(cL,uP)
299    !    Cseq() = Coe[,] * Dseq()
300    !    Dseq() = Pow[,] * Cseq()
301    !    Dseq & Cseq are populated using nVnC[,] referencing the X[,] domain values
302
303    !    Complete the read of the .bin file...
304         read(14)   X, nVnC, Coe, Pow
305      close(14)
306                                                                                return
307    End Subroutine ImportBinFile
308    !-------------------------------------------------------------------------------7 9
309
310    Subroutine BinFileCsvReport(iP)
311    !2021.04.06.1630cdt JMS- Polyize's Comma Separated Value Summary. (Bug L#337)
312     use PolyizeDef                                                 !  Fixed
313    !--End Globals
314     implicit none
315     !--Arguments
316      integer*4::iP           !Write enable>5: write(iP,...)
317     !--Internals
318      integer*4::i,nV,N,nCol
319      character::cL*80
320     !--EndDefs---------------------------
321         write(cL,"('')")                                    ;call WSF(cL,iP)
322         write(cL,"('PolyizeCsvReport():')")
323                                                              call WSF(cL,iP)
324      cL=char(0); cL(           1:nNameChars-3) = cBinFilein(1:nNameChars-3)
325                  cL(nNameChars-2:nNameChars  ) ="csv"c
326      !Use a spreadsheet to export perfect inverses as .csv's:
327      open(unit=uD,file=cL,action='write')
328      write(uD,"('Generated by : Polyize.exe|BinFileCsvReport'\)")
329      write(uD,"('      runtime ',a22)") Chrono(3)
330      write(uD, &
331             "('Label indices are augmented hexadecimals: { 0,15}->{0:f}')")
332      write(uD,"(19x,'-&-  16->g 17->h 18->i 19->j     20 ->   k')")
333      write(uD,"(/'`Polyize.ini` has the name of the .bin file to read...'/)")
334      write(uD,"('Reading: ',79(a1,:))")(cBinFileIn(i:i),i=1,len_trim(cBinFileIn))
335      write(uD,"( i6.1,',:nVT        = # of Variables Total [1:7]')") nVT
336      write(uD,"( i6.1,',:nOrdMax    = highest order variable'   )") nOrdMax
337      write(uD,"( i6.1,',:nCT        = # of Coefficients Total'  )") nCT!x nOrdMax
338      do nV=nVT,1,-1; write(uD,"(i6.1,\)") nOrd(nV); enddo!nV
339      write(uD,"(',:nOrd - Order of each variable  <- sequence 7654321')")
340      write(uD,"( i6.1,',:iPolyClass = 1:exact, =2:unscaled'    )") Pr.iPolyClass
341      write(uD,"( i6.1,',:iCoeDigits = `+***.` <-count '        )") Pr.iCoeDigits
342      write(uD,"( i6.1,',:iCoeDeciPl =number of digits after .' )") Pr.iCoeDeciPl
```

```
343      write(uD,"(f30.9,',:CoeDiv = Coe[,] matrix dividing factor')")Pr.CoeDiv
344      write(uD,"(f30.9,',:CoeMaxScaled = AbsMax Coe[,] in .bin' )")Pr.CoeMaxScaled
345      write(uD,"( i6.1,',:iPowDigits = `+***.` <-count '        )") Pr.iPowDigits
346      write(uD,"( i6.1,',:iPowDeciPl =number of digits after .' )") Pr.iPowDeciPl
347      write(uD,"(f30.9,',:PowDiv = Coe[,] matrix dividing factor')")Pr.PowDiv
348      write(uD,"(f30.9,',:PowMaxScaled = AbsMax Coe[,] in .bin' )")Pr.PowMaxScaled
349
350      write(uD,"('The domain(s) of the sample points is/are:')")
351                          write(uD,"('       N,' \)")
352      do nV = nVT,1,-1; write(uD,"('  X(',i2,'),'\)") nV                ;enddo!nV
353                          write(uD,"(' X(0:nOrd(nV),nVT)')")
354      do N = 0,nOrdMax
355           write(uD,"(i8,',',4x,\)") N
356        do nV = nVT,1,-1
357          if(N>nOrd(nV)) then
358            write(uD,"('       ,'\)")
359           else
360            write(uD,"(f7.3,','\)") X(N,nV)
361          endif!N>nOrd(nV)
362        enddo!nd
363           write(uD,"('')")
364      enddo!N
365        if(iDomain==1) &
366        write(uD,"(/'Each domain is [-1. to +1.]')")
367        if(iDomain==2) &
368        write(uD,"(/'Each domain is [-N/2. to +N/2.]')")
369
370        write(uD, &
371          "(/'To exercise your model in the sequence needed by the Coe matrix,')")
372        write(uD, &
373     "('  -or- to synthesize an accurate dataset from polynomial coefficients,')")
374        write(uD, &
375    "('  use NvnC() below to select X() combinations above, or specific A()`s.')")
376        write(uD,"('Sample # sequencer: nVnC(1:nVT,1:nCT)'\)")
377        write(uD,"('   X(1) varies the most rapidly.')")
378        do nCol = 1,nCT; write(uD,"(i8,','\)") nCol
379          do nV = nVT,1,-1; write(uD,"( i7,','\)") nVnC(nV,nCol)       ;enddo!nV
380                            write(uD,"(''      )")                     ;enddo!nCol
381
382   !write(uD,"(/ e12.3,', :[Coe*Pow-I] r*16 128-bit AbsMax inversion error'  )") &
383   !               CoePowMax
384
385   !---------------------------
386        write(uD,"(/f25.3,', :Coefficients- AbsMax value (below)' )") &
387                                                                 Pr.CoeMaxScaled
388        write(6,"('Coe[nCT,nCT] write progress bar...')")
389        call MatrixCsvWriter( nCT,Coe,CoeDiv,'A','X' &
390                             ,Pr.iCoeDigits,Pr.iCoeDeciPl,iP)
391   !---------------------------
392        write(uD,"(/f25.3,', :Powers      - AbsMax value (below)' )") &
393                                                                 Pr.PowMaxScaled
394        write(6,"('Pow[nCT,nCT] write progress bar...')")
395        call MatrixCsvWriter( nCT,Pow,PowDiv,'X','A' &
396                             ,Pr.iPowDigits,Pr.iPowDeciPl,iP)
397   !---------------------------
398        call DisclaimerEtc(uD)
399
```

```
400      close(uD)                                                              ;return
401    End Subroutine BinFileCsvReport
402   !-------------------------------------------------------------------------7 9
403
404   Subroutine MatrixCsvWriter( nTot,A,ADiv,c1rows,c1cols,iADigits,iADeciPl,iP)
405   !2021.04.01.0850cdt JMS- Writes matrices Pow() & Coe() for export
406    use PolyizeDef
407   !--End Globals
408    implicit none
409    !--Arguments
410     integer*4::nTot
411     real*16  ::A(nTot,nTot) !Pow(,) or Coe(,)
412     real*16  ::ADiv
413     character::c1Rows*1     !"a" or "x"
414     character::c1cols*1     !"a" or "x"
415     integer*4::iADigits     !Number of digits for exact values
416     integer*4::iADeciPl     !Number of decimal places
417     integer*4::iP           !Write enable>5: write(iP,...)
418   !--Internals
419     character*16::Format1
420     character*13::Format2
421     integer*4::nRow,nCol,nV,j
422     integer*4::OutLength
423     character::PrintString*40
424     !--EndDefs---------------------------
425      if(Pr.iPolyClass==1) &
426      write(uD,"(f25.3,', :Matrix entry Exact integer divisor')") ADiv
427      if(Pr.iPolyClass==2) &
428      write(uD,"(f25.3,', :divisor       ...Unscaled ...non-exact')") ADiv
429
430   !  --For writing the matrix coefficient:
431            Format1=char(0)
432      if(iADeciPl<10) then
433        write(Format1,"(a1,'sp',a1,' f',i2.2,a1,i1.1,a1,3a1,2a1)") &
434          char(40),char(44) &
435         ,iADigits+iADeciPl,char(46),iADeciPl,char(44) &! (sp,f__.0,
436         ,char(39),char(44),char(39),char(92),char(41)           !        ','\)
437        else
438         write(Format1,"(a1,'sp',a1,'f',i2.2,a1,i2.2,a1,3a1,2a1)") &
439          char(40),char(44) &
440         ,iADigits+iADeciPl,char(46),iADeciPl,char(44) &! (sp,f__.0,
441         ,char(39),char(44),char(39),char(92),char(41)           !        ','\)
442      endif!(iADeciPl<10)
443   !  write(uD,*) Format1  !e.g.: (sp,f25.22,','\)
444
445   !  --For removing trailing zeros:
446        Outlength=iADigits+iADeciPl
447        write(Format2,"(a1,'a',i2.2,a1,3a1,2a1)") &
448        char(40), OutLength,char(44) &               !  (a__,
449       ,char(39),char(44),char(39),char(92),char(41) !       ','\)
450   !  write(uD,*) Format2  !e.g.: (a25,','\)
451
452      write(uD,"('The Column and row label indices that follow'\)")
453      write(uD,"(' are single-digit augmented hexadecimals.')")
454
455   !    --Export the signed coefficients:
456        do nCol   = 1,nCT
```

```
  457         do nV   = 1,iADigits-nVT-1;write(uD,"(' '\)")                ;enddo!nV
  458                                    write(uD,"(a1,\  )") c1cols
  459         do nV   = nVT,1,-1;  write(uD,"(a1,\)") AugHex(nVnC(nV,nCOl));enddo!nV
  460         do nV   = 1,iADeciPl;write(uD,"(' '\)")                      ;enddo!nV
  461                                    write(uD,"(',\)")                 ;enddo!nCol
  462                                    write(uD,"('')")   ;call Beamer(0   ,nCT,6)
  463       do nRow   = 1,nCT                               ;call Beamer(nRow,nCT,6)
  464         do nCol = 1,nCT;     write(PrintString,Format1) A(nRow,nCol) !/ADiv
  465  !      Replace trailing 0`s with spaces
  466         do j=OutLength,1,-1; if(PrintString(j:j) /= "0") exit
  467                                 PrintString(j:j)  = " "              ;enddo!j
  468                                    write(uD,Format2) PrintString     ;enddo!nCol
  469                                    write(uD,"(a1,\  )") c1rows
  470         do nV   = nVT,1,-1;  write(uD,"(a1,\)") AugHex(nVnC(nV,nRow));enddo!nV
  471                                    write(uD,"('')")                  ;enddo!nRow
  472                                                                          return
  473  End Subroutine MatrixCsvWriter
  474  !-------------------------------------------------------------------------------7 9
  475
  476  Subroutine ValuesSequence(iP)
  477  !2021.03.30.1545cdt JMS- Show the values sequence for the particular .bin file.
  478   use PolyizeDef
  479  !--End Globals
  480   implicit none
  481   !--Arguments
  482    integer*4::iP           !Write enable>5: write(iP,...)
  483  !--Internals
  484    integer*4::i,N,nV,nC
  485    character::cL*80
  486    !--EndDefs---------------------------
  487     cL='Exporting `ValuesSequenceInfo.txt` for your use.'    ;call WSF(cL,iP)
  488     open(unit=uD,file='ValuesSequenceInfo.txt',action='write')
  489      write(uD,"(79(a1,:))") (cBinFileIn(i:i),i=1,len_trim(cBinFileIn))
  490      write(uD,"(i3,', nVT = # of Variables Total. The orders are:' )")nVT
  491      write(uD,"(7(i3.1,',',:))") nOrd(1:nVT)
  492      do nV=1,nVT
  493        write(uD,"('  N, Domain of X(',i2,'):')") nV
  494        do N=0,nOrd(nV)
  495          write(uD,"(i3.1,',',f18.12)") N,X(N,nV)
  496        enddo!N
  497      enddo!nV
  498      write(uD,"(i6.1,',nCT = # of Coefficints Total. sequence entries:')")nCT
  499      write(uD, &
  500          "('Seq. #, X1, X2,etc, using the domain values listed above.')")
  501      do nC=1,nCT
  502        write(uD,"(i6.1,',',7(i3.1,',':) )") nC,nVnC(1:nVT,nC)
  503      enddo!nC
  504      write(uD,"(/' A(1:nCT) = Coe[1:nCT,1:nCT] * Values(1:nCT)/CoeDiv')")
  505      write(uD,"(/'The sequence entries above are also the respective orders')")
  506      write(uD,"( '  of the polynomial coefficients for synthesizing data.')")
  507      write(uD,"( ' X(1:nCT) = Pow[1:nCT,1:nCT] * A(1:nCT)/PowDiv')")
  508      write(uD,"(/'Run Polyize.exe with WhatNext = -2 to write a .csv'\)")
  509      write(uD,"( ' version of the .bin file.')")
  510      call DisclaimerEtc(uD)
  511     close(uD)
  512                                                                          return
  513  End Subroutine ValuesSequence
```

```
!-------------------------------------------------------------------------------7 9

Subroutine ExampleIputs(iP)
!2021.04.05.0605cdt JMS- Write example DataIn.txt & PolysIn.txt files.
 use PolyizeDef
!--End Globals
 implicit none
 !--Arguments
  integer*4::iP           !Write enable>5: write(iP,...)
!--Internals
  integer*4::i,nV,nC1,nC2
  character::cL*80
  !--EndDefs--------------------------
   cL='Creating example files `DataIn.txt` & CoesIn.txt`.'     ;call WSF(cL,iP)
!  --Synthesize a polynomial coefficient dataset:
   CoeSeq = 0._16; CoeSeq(nCT) = 1._16; DataSeq = 0._16
!  --...and solve it:
   do nC1 = 1,nCT; DataSeq(nC1)=0._16
     do nC2 = 1,nCT;
       DataSeq(nC1) = DataSeq(nC1)+Pow(nC1,nC2)*CoeSeq(nC2)/PowDiv
   enddo; enddo!nC1
!  --Write DataIn.txt:
   open(unit=uD,file='DataIn.txt',action='write')
     write(uD,"(79(a1,:))") (cBinFileIn(i:i),i=1,len_trim(cBinFileIn))
                       write(uD,"(i6,',',i3,', :nCT & nVT') )") nCT,nVT
     do nC1 = 1,nCT;   write(uD,"(i6,',',\)") nC1
      !do nV = 1,nVT; write(uD,"(i3,',',\)") nV            ;enddo!nV
       do nV = 1,nVT; write(uD,"(i3,',',\)") nVnC(nV,nC1)  ;enddo!nV
                      write(uD,"(sp,f40.30)") DataSeq(nC1);enddo!nC1
       write(uD,"(a1,'(sp,f40.30)',a1,'c, :Fortran output write format.')") &
                 char(34)      ,char(34)
   close(uD)
!  --Write PolysIn.txt:
   open(unit=uD,file='CoesIn.txt',action='write')
     write(uD,"(79(a1,:))") (cBinFileIn(i:i),i=1,len_trim(cBinFileIn))
                       write(uD,"(i6,',',i3,', :nCT & nVT') )") nCT,nVT
     do nC1 = 1,nCT;   write(uD,"(i6,',',\)") nC1
       do nV = 1,nVT; write(uD,"(i3,',',\)") nV            ;enddo!nV
                      write(uD,"(sp,f40.30)") CoeSeq(nC1);enddo!nC1
       write(uD,"(a1,'(sp,f40.30)',a1,'c, :Fortran output write format.')") &
                 char(34)      ,char(34)
   close(uD)
                                                                        return
End Subroutine ExampleIputs
!-------------------------------------------------------------------------------7 9

Subroutine DataInToCoesOut(iP)
!2021.04.05.0605cdt JMS- Read DataIn.txt and write CoesOut.txt
 use PolyizeDef
!--End Globals
 implicit none
 !--Arguments
  integer*4::iP           !Write enable>5: write(iP,...)
!--Internals
  integer*4::i,j,nV,nC1,nC2,nCTL,nVTL
  character::cL*80
  character::Format1*80
```

```
   !--EndDefs---------------------------
    cL='Process your `DataIn.txt` -> `CoesOut.txt`.'            ;call WSF(cL,iP)
!   --CoeSeq() = Coe[,] * DataSeq():
    cL=Char(0)
    open(unit=uD,file='DataIn.txt',action='read')
      read(uD,*) cL(1:nNameChars)   ;write(6,*) cL
      read(uD,*) nCTL,nVTL          ;write(6,*) 'nCT=',nCTL,' nVT=',nVTL
      if((nCT/=nCTL).or.(nVT/=nVTL)) return
!     --Import the data:
      DataSeq(nC1)=0._16
      do nC1 = 1,nCT; read(uD,*) nC2,(i,j=1,nVT),DataSeq(nC2)         ;enddo!nC1
      read(uD,*) Format1; write(6,*) Format1
    close(uD)
!   --- compute CoeSeq:
    CoeSeq=0._16
    do nC1=1,nCT
      do nC2 = 1,nCT
      CoeSeq(nC1) = CoeSeq(nC1)+Coe(nC1,nC2)*DataSeq(nC2)/CoeDiv
    enddo; enddo!nC1
!   --Write CoesOut.txt:
    open(unit=uD,file='CoesOut.txt',action='write')
      write(uD,"(79(a1,:))") (cBinFileIn(i:i),i=1,len_trim(cBinFileIn))
                       write(uD,"(i6,',',i3,', :nCT & nVT') )") nCT,nVT
      do nC1 = 1,nCT;  write(uD,"(i6,',''\)") nC1
        do nV = 1,nVT; write(uD,"(i3,',''\)") nVnC(nV,nC1)  ;enddo!nV
                       write(uD,Format1) CoeSeq(nC1);enddo!nC1
        write(uD,*)  Format1
    close(uD)
                                                                          return
End Subroutine DataInToCoesOut
!-----------------------------------------------------------------------------7 9

Subroutine CoesInToDataOut(iP)
!2021.04.05.0605cdt JMS- Read DataIn.txt and write CoesOut.txt
 use PolyizeDef
!--End Globals
 implicit none
 !--Arguments
  integer*4::iP           !Write enable>5: write(iP,...)
!--Internals
  integer*4::i,j,nV,nC1,nC2,nCTL,nVTL
  character::cL*80
  character::Format1*80
   !--EndDefs---------------------------
    cL='Process your `CoesIn.txt` -> `DataOut.txt`.'            ;call WSF(cL,iP)
!   --CoeSeq() = Coe[,] * DataSeq():
    cL=Char(0)
    open(unit=uD,file='CoesIn.txt',action='read')
      read(uD,*) cL(1:nNameChars)   ;write(6,*) cL
      read(uD,*) nCTL,nVTL          ;write(6,*) 'nCT=',nCTL,' nVT=',nVTL
      if((nCT/=nCTL).or.(nVT/=nVTL)) return
!     --Import the data:
      CoeSeq(nC1)=0._16
      do nC1 = 1,nCT; read(uD,*) nC2,(i,j=1,nVT),CoeSeq(nC2)         ;enddo!nC1
      read(uD,*) Format1; write(6,*) Format1
    close(uD)
!   --- compute DataSeq:
```

```
628      DataSeq=0._16
629      do nC1=1,nCT
630        do nC2 = 1,nCT
631        DataSeq(nC1) = DataSeq(nC1)+Pow(nC1,nC2)*CoeSeq(nC2)/PowDiv
632      enddo; enddo!nC1
633   !  --Write CoesOut.txt:
634      open(unit=uD,file='DataOut.txt',action='write')
635        write(uD,"(79(a1,:))") (cBinFileIn(i:i),i=1,len_trim(cBinFileIn))
636                               write(uD,"(i6,',',i3,', :nCT & nVT') )") nCT,nVT
637        do nC1 = 1,nCT;  write(uD,"(i6,',',\)") nC1
638         !do nV = 1,nVT; write(uD,"(i3,',',\)") nV            ;enddo!nV
639          do nV = 1,nVT; write(uD,"(i3,',',\)") nVnC(nV,nC1)  ;enddo!nV
640                         write(uD,Format1) DataSeq(nC1);enddo!nC1
641          write(uD,*)  Format1
642      close(uD)
643                                                                        return
644    End Subroutine CoesInToDataOut
645    !-------------------------------------------------------------------------7 9
646    Subroutine CoefficientCount(iP)
647    !2021.04.06.0555cdt JMS- # of coeff`s of even-order multi-variable Polynomials
648     use PolyizeDef, only:uD
649     implicit none
650     !--Arguments
651      integer*4 ::iP           !Write enable>5: write(iP,...)
652     !--Internals
653      integer*4 ::nMax=7,j,nOrder(10)=1,nCh,iUsed,nLine
654      integer*4 ::i1,i2,i3,i4,i5,i6,i7
655      integer*4 ::j1,j2,j3,j4,j5,j6,j7
656      character::cL*100
657      !--EndDefs-----------------------------------------------
658   !  --Write DataIn.txt:
659      cL='Write the `Coefficient Count Table`.'                    ;call WSF(cL,iP)
660      open(unit=uD,file='CoefficientCount.txt',action='write')
661      cL=''c ;write(uD,*) CL
662      cL= &
663      '    Number of polynomial coefficients <= 9999'// &
664      '  for 7 variables up to order 20'c
665      write(uD,*) CL
666      cL= &
667      '            Order of v1 = 0    2    4    6    8|'// &
668      '  10   12   14   16   18   20'c
669      write(uD,*) CL
670      cL= &
671      '                      ----------------- <-exact|unscaled->'// &
672      ' ------------------'c
673      write(uD,*) CL
674                                  nLine=1
675      do i1=0,20,2                 ; j1 = i1+1;
676        do i2=i1,20,2              ; j2 = i2+1;
677          do i3=i2,20,2            ; j3 = i3+1;
678            do i4=i3,20,2          ; j4 = i4+1;
679              do i5=i4,20,2        ; j5 = i5+1;
680                do i6=i5,20,2      ; j6 = i6+1;
681                                     j  = j1*j2*j3*j4*j5*j6
682                                   if(J>9999) exit
683                    cL=char(0);  iUsed = 0;           nCh = 22
684                    write(cL(1:22),"(i3,1x,6i3.0\)") nLine,i1,i2,i3,i4,i5,i6
```

```
                                    if(nLine==  1) cL(17:22) = 'V2 = 0'
                                    if(nLine== 11) cL(14:19) = 'V3 = 0'
                                    if(nLine== 66) cL(11:16) = 'V4 = 0'
                                    if(nLine==132) cL( 8:13) = 'V5 = 0'
                                    if(nLine==161) cL( 5:10) = 'V6 = 0'
                                    if(nLine==171) cL( 4: 7) = 'V7=0'
                do i7=2,i6,2;                write(cL(nCh+1:nCh+5),"('     '\)")
                                                     nCh = nCh+5        ;enddo!i7
                do i7=i6,20,2;  j7 = i7+1;
                                   j  = j1*j2*j3*j4*j5*j6*j7
                                if(J>9999) exit
                                iUsed = iUsed+1
                                             write(cL(nCh+1:nCh+5),"(i5\)") j
                                                     nCh = nCh+5
                enddo!i7
                             if(iUsed==0) exit
                               nLine=nLine+1
                                                        cL(48:48)='|'
                                             write(uD,*) CL
              enddo!i6
            enddo!i5
          enddo!i4
        enddo!i3
      enddo!i2
    enddo!i1
    cL= &
    '                        ----------------- <-exact|unscaled->'// &
    ' ------------------'c
    write(uD,*) CL
    cL=''c ;write(uD,*) CL
    call DisclaimerEtc(uD)
    close(uD)
                                                                          return
End Subroutine CoefficientCount
!-------------------------------------------------------------------------------7 9

Subroutine DisclaimerEtc(iP)
!2021.04.01.0855dt JMS- Disclaimer, etc.
!Elapsed is updated first,
!       so using the same argument twice results in returning the DaTime value.
!--Arguments
 implicit none
  integer*4::iP           !Write enable>5: write(iP,...)
 !--Internals
  character::cL*80
  !--EndDefs-----------------------------------------------
 cL=""                                                        ;call WSF(cL,iP)
 cL="   Link: http://ftp.setterholm.com/ExactInversePolynomials/Polygen.zip"c
                                                               call WSF(cL,iP)
 cL="Analyst: Jeff Setterholm  Lakeville MN 55044 USA   Tuesday 2020.04.06"c
                                                               call WSF(cL,iP)
 cL=""                                                        ;call WSF(cL,iP)
 cL="  ********, Polyize.exe is post-copyright i.e.:FREE ,*********"c
                                                               call WSF(cL,iP)
 cL=""                                                        ;call WSF(cL,iP)
 cL="  My legal disclaimer:"c                                 ;call WSF(cL,iP)
 cL="  ************************************************************"c
```

```
                                                                    call WSF(cL,iP)
  cL="  ********,  Individual cognition is always flawed  ,********"c
                                                                    call WSF(cL,iP)
  cL="  ********,          including yours & mine.        ,********"c
                                                                    call WSF(cL,iP)
  cL="  ********,                 - So: -                 ,********"c
                                                                    call WSF(cL,iP)
  cL="  ********,    Use these results at your own risk.  ,********"c
                                                                    call WSF(cL,iP)
  cL="  ************************************************************"c
                                                                    call WSF(cL,iP)
  cL="           Mitigate malfunctions. Nurture synergies."c
                                                                    call WSF(cL,iP)
  cL=""                                                            ;call WSF(cL,iP)
                                                                        return
 End Subroutine DisclaimerEtc
 !-------------------------------------------------------------------------------7 9

```