

```

1  !jInvert.f95
2  !2026.04.19.1810cdt-    Complex Matrix Overwriting Inverter
3
4  !   Author- Jeffrey M. Setterholm, Lakeville,MN 55044 USA
5  !   IP Status- Free source code (e.g.: post copyright)
6  !
7  !   Computer- "T4"/HP-800-G4-Mini/i7-8700T/IntelUHD630/win10Pro-22H2
8  !               ^name ^Mfgr.Id   ^chipset ^graphics   ^OS
9  !               /AbsoftProFortran 21.0.2/OpenGL+Glut3.6
10 !               ^compiler      ^Fortran graphics
11 !Disclaimer:
12 !*****
13 !*****   Individual cognition is always flawed,   *****
14 !*****   including yours and mine.               *****
15 !*****   - So: -                                   *****
16 !*****   Use this code & program at your own risk. *****
17 !*****
18
19 !Table of Contents: ...use to search...
20 !Module jInvert_FoR      <- The "Frame-of-Reference"      @ 31
21 !End Module jInvert_FoR      @ 68
22 !Subroutine jInvert(Nin,C,iRW,iP)    <- the overwriting complex inverter @ 68
23 !Subroutine jInvert_Report(N,C,iP)   <- overwriter progress reports      @237
24 !Subroutine jInvert_Check(N,Cout,iP) <- quality confirmation              @270
25 !                                     @: approximate Line#'s ^^^^
26
27 !   See "jInvert-f95.pdf" for a color-coded-syntax version of this file.
28 !-----
29 !-----7 9
30
31 Module jInvert_FoR      !The "Frame-of-Reference" for all that follows.
32
33 !use jInvert_FoR !Complex overwriting Inverter variables      2026.04.19
34 !               ,only: ValMin,iRank,CdetN,iUsed,iRu,jCu,Temp !<-any subset allowed
35
36 !The key variables of "jInvert_Demo" which reads "Invert-Input.csv":
37 character(Len=40):: DataFileIn    = "jInvert-Input.txt"
38 character(Len=40):: DataFileOut    = "jInvert-Output.txt"
39 integer(4)      :: iRW              !Unit# for- file Reads & writes
40 integer(4)      :: iP              ! - writing DataFileOut
41 character(Len=79):: YourDataHeader &!First line of DataFileIn: description
42                                     = "Reading the First line failed."
43 integer( 4)      :: Nin              !The size of Cin      @ 98
44 integer(4)      :: iRankin          !Rank of the inverse @213
45 integer(4)      :: iRankOut
46
47 complex(16),allocatable:: C( :, :) !Input -> overwritten -> output @ 99
48
49 !The key variables of "jInvert()" are:
50 integer( 4) :: N      ! @ 51
51 real(16)    :: ValMin ! @117
52 real(16)    :: SNR    = -1._16 !Signal-to-Noise Ratio @152
53 integer(4)  :: iRank  !Tracks the growing rank of the inverse @168
54 complex(16) :: CdetN  !Abs(determinant product) for each iteration @169
55 integer(4),allocatable:: iUsed(:) ! @194
56                                     !Overwriting:
57 integer(4),allocatable:: iRu(:)   !Tracks used pivot- rows @175
58 integer(4),allocatable:: jCu(:)   ! - columns @178
59
60 complex(16),allocatable:: Temp(:) !Temporary use vector of size Nin ----
61 complex(16),allocatable:: Cin(:, :) !Used by jInvert_Check() @124
62 complex(16),allocatable:: CTest(:, :) !Used by jInvert_Check() @288
63
64 !Contains
65 End Module jInvert_FoR
66 !-----7 9
67
68
69

```

```

70 Subroutine jInvert(Nin,C,iRankOut,iRW,iP)
71 !2026.04.19.1815cdt JMS- Adapted for complex variables
72 !2006.05.25.0715cdt JMS- An instantiation of the "Setterholm Matrix Inverter"
73 !--Globals
74 use jInvert_FoR &!Complex overwriting Inverter variables 2026.04.19
75 ,only: ValMin,iRank,CdetN,iUsed,iRu,jCu,Temp,Cin,Ctest,SNR
76 !--End Globals
77 implicit none
78 !--Arguments These are arguments, not the Frame-of-Reference (FoR) variables
79 ! with the same names, which jInvert_Demo uses.
80 ! In the same way, you can use the FoR variables when calling
81 ! jInvert().
82 integer(4) :: Nin
83 complex(16):: C(Nin,Nin)
84 integer(4) :: iRankOut
85 integer(4) :: iRW !A Fortran unit# to use for reading or writing
86 integer(4) :: iP
87 !--Internals
88 integer(4) :: N
89 integer(4) :: i,iu,iu2,j,ju,ju2,L,iuB,juB
90 complex(16)::Cmult
91 complex(16)::ValMax,ValMax2
92 integer(8) ::iBitFlag,jBitFlag
93 integer(4) ::iAlloc
94 complex(16):: Czero,Cone
95 !----- !end defs
96 if(iP>5) write(iP, "(/' {jInvert(Nin,C,iRankOut,iRW,iP):'&
97 &,45('-'),t40,' [allocating] ',t63,' jInvert.f95 @ 97')")
98 N = Nin
99 !---
100 Czero = ( 0._16, 0._16) !Complex: zero
101 Cone = (+1._16, 0._16) !Complex: real 1.0
102
103 allocate(iUsed(N),stat=iAlloc)
104 if(iAlloc.ne.0) stop 'jInverter: iUsed(N) allocation error. Halt.'
105 iUsed = 0
106 allocate(iRu(N),stat=iAlloc)
107 if(iAlloc.ne.0) stop 'jInverter: iRu(N) allocation error. Halt.'
108 iRu = 0
109 allocate(jCu(N),stat=iAlloc)
110 if(iAlloc.ne.0) stop 'jInverter: jCu(N) allocation error. Halt.'
111 jCu = 0
112 allocate(Temp(N),stat=iAlloc)
113 if(iAlloc.ne.0) stop 'jInverter: Temp(N) allocation error. Halt.'
114 Temp = Czero
115 allocate(Cin(N,N),stat=iAlloc)
116 if(iAlloc.ne.0) stop 'jInverter: Cin(N,N) allocation error. Halt.'
117 Cin = Czero
118 allocate(CTest(N,N),stat=iAlloc)
119 if(iAlloc.ne.0) stop 'jInverter: CTest(N,N) allocation error. Halt.'
120 CTest = Czero
121 !---
122 CdetN = Czero
123 !---
124 Cin = C !Saves C as Cin for use in analyzing Linear Dependence.
125 !C will be overwritten with its inverse when C is full-rank.
126 !"jInvert_Check()" aids interpreting less-than-full-rank results.
127
128 !This is a binary file of N and Cin. windows 10 security blocks ".bin" files
129 if(iP>5) write(iP, "(' Writing binary file jCin.bim' ,t75,' @129')")
130 if(iP>5) write(iP, "(' Fortran arrays store in column-major order.',/))")
131 open( unit=iRW, file="jCin.bim", action='write', form = 'unformatted' &
132 , access='sequential', status='replace')
133 write(iRW) n
134 write(iRW) C
135 close(iRW)
136
137 iUsed=0; iRu=0; jCu=0; ValMin=0._16; Temp=Czero; iRank=0
138 do i=1,N ;iRu(i)=i ;jCu(i)=i ;end do!i

```

```

139  if(iP.gt.5) call jInvert_Report(N,C,iP)                !Initial printout
140
141  ! Progressive inversion - from the abs(largest) pivot down to the noise,
142  ! ... a loop:
143  do L=1,N ;iU=0 ;jU=0 ;valMax =Czero
144  do i=1,N ;if(iRu(i).lt.0)                                cycle
145  do j=1,N ;if(jCu(j).lt.0)                                cycle
146  if(abs(valMin).gt.abs(C(i,j)))                           cycle
147  if(abs(valMax).gt.abs(C(i,j)))                           cycle
148  valMax=C(i,j) ;iU=i ;jU=j
149  enddo!j
150  enddo!i
151  if(L==1) then
152  valMin=abs(valMax)*1.e-24_16 !"noise floor" ...not cast in concrete.
153  ! works here because Fortran's complex(16)
154  ! components have 31-decimal-place precision.
155  ! Can be overwritten by SNR...
156  if(SNR>0._16) then
157  valMin=abs(valMax)/SNR                                     !2026.02.24
158  if(iP>5) &
159  write(iP,"(34x,'User SNR =',e15.6,' Signal-to-Noise Ratio')") SNR
160  endif!(SNR>0._16)
161  if(iP>5) &
162  write(iP,"(34x,' valMin =',e15.6,' scaled by pivot: '/')") valMin
163  CdetN=valMax
164  endif!(L==1)
165
166  if(iU==0)                                                Exit
167  if(abs(valMax).lt.valMin)                                Exit
168  iRank=iRank+1
169  if(L>1) CdetN=CdetN*valMax
170  valMax=C(iU,jU) ;iU2=jCu(jU) ;jU2=iRu(iU)
171  if(iP.gt.5) write(iP, &
172  "((' iU =',i3,' jU =',i3,16x,'Pivot =',sp,' (',f13.6,',',f13.6,'j)'))"&
173  "      iU      jU,                                valMax
174  Temp(1:N)=C(1:N,jU)                                ;i=iRu(iU)
175  C(1:N,jU)=C(1:N,jU2)                                ; iRu(iU)=iRu(iU2)
176  C(1:N,jU2)=Temp(1:N)                                ; iRu(iU2)=-abs(i)
177  Temp(1:N)=C(iU,1:N)                                ;j=jCu(jU)
178  C(iU,1:N)=C(iU2,1:N)                                ; jCu(jU)=jCu(jU2)
179  C(iU2,1:N)=Temp(1:N)/valMax ; jCu(jU2)=-abs(j)
180  do i=1,N ;if(i.eq.iU2) cycle ;Cmult=C(i,jU2)
181  do j=1,N ;C(i,j)=C(i,j)-C(iU2,j)*Cmult ;enddo!j
182  C(i,jU2)=-Cmult/valMax
183  enddo; C(iU2,jU2)=1._16/valMax
184  if(iP.gt.5) call jInvert_Report(N,C,iP)                !Loop printout
185  enddo !Look for the next largest remaining pivot.
186
187  !Done trying to invert C. Report how it turned out:
188  if(iP>5) write(iP, &
189  "(13x,'sign untracked:',9x,'CdetN =',sp,' (',f13.6,',',f13.6,'j)'))" CdetN
190  iBitFlag=0; jBitFlag=0
191  do i=1,N
192  if((iRu(i).lt.0).and.(i.lt.62)) iBitFlag=iBitFlag+2**(i-1)
193  if((jCu(i).lt.0).and.(i.lt.62)) jBitFlag=jBitFlag+2**(i-1)
194  if(iRu(i).lt.0)iUsed(i)=1
195  ! Zero linearly-dependent rows and columns, if any:
196  if(iRu(i).gt.0) C(i,1:N)=Czero
197  if(jCu(i).gt.0) C(1:N,i)=Czero
198  enddo!i
199
200  if((iP >5).and.(iRank.lt.N)) then
201  write(iP," (/ ^:Linearly Dependent. Rows and Columns with&
202  & positive indices (above) failed to invert,')")
203  write(iP," (' The corresponding rows & columns are zeroed.',\)")
204  write(iP," (' BitFlags: 1-used, 0-discarded [N->1]')")
205  write(iP,"(['b63.6,'] input')") iBitFlag !For first 61
206  write(iP,"(['b63.6,'] output')") jBitFlag !For first 61
207  call jInvert_Report(N,C,iP) !Linearly-dependence printout

```

```

208     endif!iP>5 & iu==0
209
210     !if(iRank<Nin) call jInvert_Check(N,C,iP)
211         call jInvert_Check(N,C,iP)
212
213     iRankOut = iRank
214     !This is a binary file of N iRank, and Cout.
215     if(iP>5) write(iP,"(/' Writing binary file jCinv.bim '      ,t75,' @214')")
216     if(iP>5) write(iP,"(' Fortran arrays store in column-major order.')"")
217     open( unit=iRW, file="jCinv.bim" , action='write', form = 'unformatted' &
218           , access='sequential', status='replace')
219     write(iRW) n
220     write(iRW) iRank
221     write(iRW) C
222     close(iRW)
223
224     !Housekeeping:
225     deallocate(iUsed)
226     deallocate(iRu)
227     deallocate(jCu)
228     deallocate(Temp)
229     deallocate(Cin)
230     deallocate(CTest)
231     if(iP>5) write(iP,&
232         "(' jInvert} ',60('-'),t40,' [deallocated] '      ,t70,' Exit @230',/)"
233         return
234 End Subroutine jInvert
235 !-----7-9
236
237 Subroutine jInvert_Report(N,C,iP)
238 !2026.01.31.0745cst JMS- For complex matrices
239 !2018.09.04.1340cdt JMS- Prints Invert's progress.
240 !           Expects,uses, & returns real(16)'s (~QUAD precision).
241 !--- No globals
242 !--Globals
243 use jInvert_FoR &!Complex overwriting Inverter variables      2026.04.19
244 ,only: ValMin,iRank,CdetN,iRu,jCu
245 !--End Globals
246 implicit none
247 !--Arguments
248 integer(4) :: N
249 complex(16):: C(N,N)
250 real(16) :: Noise
251 integer(4) :: iP
252 !--Internals
253 integer(4)::i,j
254 !--EndDefs-----
255 Noise = ValMin
256 write(iP,"('@ Rank =',i3,21x,'abs(CdetN) =',e23.10/22x,' ',\)" &
257         iRank,abs(CdetN) ;if(iP < 6) return
258 do j=1,N; write(iP,"(sp,i6,24x,' ',\)" jCu(j)
259 enddo; write(iP,"(')")
260 do i=1,N; write(iP,"(i,' ',\)" iRu(i)
261 do j=1,N; write(iP,"(sp,' (',f13.6,' ',f13.6,'j)',\)" C(i,j)
262 enddo; write(iP,"(')")
263 enddo!i
264 write(iP,"(13(' '),\)"
265 do i=1,N; write(iP,"(31('-'),\)"
266 enddo; write(iP,*);
267 return
268 !-----7-9
269
270 Subroutine jInvert_Check(N,Cout,iP)
271 !2026.04.19.1605cdt JMS- Prints C*Cin and Cin*C
272 !--Globals
273 use jInvert_FoR &!Complex overwriting Inverter variables      2026.04.19
274 ,only: Cin,CTest,iRank
275 !--End Globals
276 implicit none

```

```

277  !--Arguments
278  integer(4) :: N
279  complex(16) :: Cout(N,N)
280  integer(4) :: iP
281  !--Internals
282  complex(16) :: Cz = ( 0., 0.)
283  integer(4) :: i,j,k
284  !--EndDefs-----
285  CTest = Cz
286  do i = 1,N
287    do j = 1,N
288      do k = 1,N
289        CTest(i,j) = CTest(i,j) + Cout(i,k)*Cin(k,j)
290      enddo!k
291    enddo!j
292  enddo!i
293  if(iP>5) write(iP,("/'Cinv*Cin: maps inputs back into inputs.'&
294            &,' An Identity matrix if full-rank.'))
295  do i=1,N
296    do j=1,N; write(iP,"(sp,' (',e13.4,',',e13.4,'j)',',\')") CTest(i,j)
297    enddo; write(iP,"(')")
298  enddo!i
299
300  CTest = Cz
301  do i = 1,N
302    do j = 1,N
303      do k = 1,N
304        CTest(i,j) = CTest(i,j) + Cin(i,k)*Cout(k,j)
305      enddo!k
306    enddo!j
307  enddo!i
308  if(iP>5) write(iP,("/'Cin*Cinv: maps outputs back into outputs.'&
309            &,' An Identity matrix if full-rank.'))
310
311  do i=1,N
312    do j=1,N; write(iP,"(sp,' (',e13.4,',',e13.4,'j)',',\')") CTest(i,j)
313    enddo; write(iP,"(')")
314  enddo!i
315  if((iP>5).and.(iRank<N)) &
316  write(iP,"( /'Consider these results tentative for linearly dependent&
317            & complex matrices.',/, ' I have not applied them yet.'&
318            &,'For real matrices the results have confirmed utility.'))
319
320  return
321 End Subroutine jInvert_Check
322 !-----7-9
323

```