Algorithms in the Nashlib set in various programming languages – Part 4

John C Nash, retired professor, University of Ottawa Peter Olsen, retired ??

11/01/2021

Contents

Abstract

Algorithms 24 and 25 from the book Nash (1979) are implemented in a variety of programming languages including Fortran, BASIC, Pascal, Python and R. These routines concern the use of iterative methods, in particular conjugate gradients, to solve linear algebra problems.

Overview of this document

This section is repeated for each of the parts of Nashlib documentation.

A companion document **Overview of Nashlib and its Implementations** describes the process and computing environments for the implementation of Nashlib algorithms. This document gives comments and/or details relating to implementations of the algorithms themselves.

Note that some discussion of the reasoning behind certain choices in algorithms or implementations are given in the Overview document.

Algorithm 24 – conjugate gradients for linear equations and least squares

Fortran

Listing

```
      SUBROUTINE A24CG(N,B,C,TOL,G,IPR,APR,V,T,IMULT)
C  ALGORITHM 24 CONJUGATE GRADIENT SOLUTION OF LINEAR EQUATIONS
C    HAVING POSITIVE DEFINITE COEFFICIENT MATRIX
C  J.C. NASH   JULY 1978, FEBRUARY 1980, APRIL 1989
C  N     = ORDER OF PROBLEM AND LENGTH OF VECTORS B,C,G,V,T
C  B     = INITIAL GUESS FOR SOLUTION (INPUT) MAY BE NULL
C        = SOLUTION (OUTPUT)
C  C     = RIGHT HAND SIDE OF EQUATIONS
C  TOL   = CONVERGENCE TOLERANCE ON SIZE OF RESIDUAL SUM OF SQUARES
C  G     = VECTOR OF APPROXIMATE RESIDUALS (OUTPUT)
C  IPR   = PRINTER CHANNEL. IF IPR.GT.0 PRINT TO CHANNEL IPR
C  APR   = NAME OF SUBROUTINE TO PUT A*T IN V
C         CALLING SEQUENCE:   CALL APR(N,T,V)
C  V,T   = WORK ARRAYS OF N ELEMENTS EACH
C  IMULT = NO. OF MATRIX MULTIPLICATIONS ALLOWED (INPUT) OR USED (OUT)
C  STEP 0
      INTEGER N,IPR,IMULT,LIMIT,I,ITN,COUNT
      REAL TOL,G2,G2L,K,T2,B(N),C(N),G(N),T(N),V(N)
      LIMIT=IMULT
      IMULT=0
C  STEP 1
   5  IMULT=IMULT+1
      IF(IMULT.GT.LIMIT)RETURN
      CALL APR(N,B,G)
      DO 10 I=1,N
        G(I)=G(I)-C(I)
  10  CONTINUE
C  STEP 2
  20  G2=0.0
      DO 25 I=1,N
        G2=G2+G(I)**2
        T(I)=-G(I)
  25  CONTINUE
      IF(IPR.GT.0)WRITE(IPR,950)IMULT,G2
 950  FORMAT(13H AFTER STEP 2,I4,15H PRODUCTS, RSS=,1PE16.8)
C  STEP 3
      IF(G2.LT.TOL)RETURN
C  STEP 4
      DO 110 ITN=1,N
C  STEP 5
        IMULT=IMULT+1
        IF(IMULT.GT.LIMIT)RETURN
      CALL APR(N,T,V)
C  STEP 6
        T2=0.0
        DO 60 I=1,N
          T2=T2+T(I)*V(I)
```

```
  60    CONTINUE
C  STEP 7
        K=G2/T2
        G2L=G2
C  STEP 8
        G2=0.0
        COUNT=0
        DO 80 I=1,N
          G(I)=G(I)+K*V(I)
          T2=B(I)
          B(I)=T2+K*T(I)
          IF(T2.EQ.B(I))COUNT=COUNT+1
          G2=G2+G(I)**2
  80    CONTINUE
C  STEP 9
      IF(COUNT.EQ.N)GOTO 5
C     IF(COUNT.EQ.N)GOTO 20
C       IF(G2.LT.TOL)RETURN
C   USING ALTERNATIVE STEP 9 ROUTE
        IF(G2.LT.TOL)GOTO 5
      IF(IPR.GT.0)WRITE(IPR,951)IMULT,G2
 951  FORMAT(13H AFTER STEP 9,I4,15H PRODUCTS, RSS=,1PE16.8)
C  STEP 10
        T2=G2/G2L
        DO 100 I=1,N
          T(I)=T2*T(I)-G(I)
 100    CONTINUE
C  STEP 11
 110  CONTINUE
C  STEP 12
      GOTO 5
C  NO STEP 13 SINCE RETURN USED
      END
      SUBROUTINE FRANK(N,T,V)
C  J.C. NASH   JULY 1978, APRIL 1989
      INTEGER N,I,J
      REAL T(N),V(N),S
      DO 10 I=1,N
        S=0.0
        DO 5 J=1,N
          S=S+AMIN0(I,J)*T(J)
   5    CONTINUE
        V(I)=S
  10  CONTINUE
      RETURN
      END
```

Example output

We solve the linear equations problem

$$Ax = b$$

for A being the Frank matrix and b the vector that is all $A1$, that is, A times a vector of ones. Different orders of problem are tried. This problem has an obvious solution, which is that

$$x_i = 1$$

and the elements of A are integers, so we can compute the error in the solution as well as the residual

$$b - Ax$$

```
A <- matrix(0, nrow=5, ncol=5)
b <- rep(1,5)

for (i in 1:5) {
   for (j in 1:5){
       A[i,j]=min(i,j)
   }
}
print(A)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1    1    1    1
## [2,]    1    2    2    2    2
## [3,]    1    2    3    3    3
## [4,]    1    2    3    4    4
## [5,]    1    2    3    4    5
```

```
C <- A %*% b
print(C)
```

```
##      [,1]
## [1,]    5
## [2,]    9
## [3,]   12
## [4,]   14
## [5,]   15
```

Note that the program, as given, is single precision, so some of the results are not terribly accurate. Also the control `IMULT` is an upper limit on the number of matrix multiplications before the process is artificially halted. We still get an approximate solution at this stage, and the result accuracy in terms of residuals and error.

```
gfortran ../fortran/dr24.f
mv ./a.out ../fortran/dr24le.run
../fortran/dr24le.run < ../fortran/a24f.in
```

```
##  ORDER=   10 IMULT=   50 TOLERANCE=  9.99999994E-09
##  MEAN ABSOLUTE -- RESIDUAL=  2.67982487E-05 -- ERROR=  8.17596883E-05
##  ORDER=   10 IMULT=   10 TOLERANCE=  9.99999994E-09
##  MEAN ABSOLUTE -- RESIDUAL=  3.41129315E-04 -- ERROR=  4.34631103E-04
##  ORDER=   10 IMULT=    5 TOLERANCE=  9.99999994E-09
##  MEAN ABSOLUTE -- RESIDUAL=  1.70371048E-02 -- ERROR=  2.66596377E-02
##  ORDER=   50 IMULT=   51 TOLERANCE=  9.99999994E-09
##  MEAN ABSOLUTE -- RESIDUAL=  3.52401723E-04 -- ERROR=  4.98390182E-05
##  ORDER=   50 IMULT=   10 TOLERANCE=  9.99999994E-09
##  MEAN ABSOLUTE -- RESIDUAL=  9.57564563E-02 -- ERROR=  1.86094921E-02
##  ORDER=   50 IMULT=  100 TOLERANCE=  9.99999975E-06
```

```
##  MEAN ABSOLUTE -- RESIDUAL=  2.42843627E-04 -- ERROR=  2.69281853E-04
##  ORDER=    0 IMULT=    0 TOLERANCE=  0.00000000E+00
```

Pascal

The 1990 edition of Compact Numerical Methods included three different examples of use of conjugate gradients for linear algebraic problems:

- linear equations

Algorithm 25 – Rayleight quotient minimization

Fortran

Listing

```
C&&& A25
C  TEST ALG 25 USING GRID (5 POINT)
C  J.C. NASH   JULY 1978, APRIL 1989
      LOGICAL IFR
      INTEGER N,M,NOUT,NIN,KPR,LIMIT,I
      EXTERNAL APR,BPR
C     REAL EPS,P0,X(N),S(N),T(N),U(N),V(N),W(N),Y(N),RNORM
      COMMON /GSZ/ M,IFR,R(1600)
      REAL EPS,P0,RNORM,VNORM,RNV
      REAL S(1600),T(1600),U(1600),V(1600),W(1600),X(1600),Y(1600)
C  I/O CHANNELS
      NIN=5
      NOUT=6
  1   READ(NIN,900)M,LIMIT
 900  FORMAT(2I4)
      N=M*M
      WRITE(NOUT,950)M,N,LIMIT
 950  FORMAT(' GRID ORDER',I4,' EQNS ORDER',I5,'  LIMIT=',I4)
      IF(M.LE.0)STOP
      IFR=.FALSE.
C  IBM MACHINE PRECISION
      EPS=16.0**(-5)
      KPR=LIMIT
      RNORM=1.0/SQRT(FLOAT(N))
      DO 10 I=1,N
        X(I)=RNORM
  10  CONTINUE
      CALL A25RQM(N,X,EPS,KPR,S,T,U,V,W,Y,P0,NOUT,APR,BPR)
      WRITE(NOUT,951)KPR,P0
 951  FORMAT(' RETURNED AFTER',I4,' PRODUCTS WITH EV=',1PE16.8)
      DO 20 I=1,N
        R(I)=-P0*X(I)
  20  CONTINUE
      CALL APR(N,X,V)
      RNORM=0.0
      VNORM=0.0
      DO 30 I=1,N
        RNORM=RNORM+(V(I)+R(I))**2
        VNORM=VNORM+X(I)**2
  30  CONTINUE
      RNORM=SQRT(RNORM/N)
      VNORM=SQRT(VNORM/N)
      RNV=RNORM/VNORM
      WRITE(NOUT,952)RNORM,VNORM,RNV
 952  FORMAT(' RESIDUAL NORM=',1PE16.8,' /',E16.8,'=',E16.8)
      GOTO 1
      END
      SUBROUTINE BPR(N,X,V)
```

```
C  J.C. NASH   JULY 1978, APRIL 1989
C  UNITM MATRIX * X  INTO V
      INTEGER N,I
      REAL X(N),V(N)
      DO 100 I=1,N
        V(I)=X(I)
 100  CONTINUE
      RETURN
      END
      SUBROUTINE APR(N,X,V)
C  J.C. NASH   JULY 1978, APRIL 1989
      LOGICAL IFR
      INTEGER N,I,J,M
      DOUBLE PRECISION S
      REAL X(N),V(N),D,Q
C  M BY M GRID OF A GEORGE  M=SQRT(N)
      COMMON /GSZ/ M,IFR,R(1600)
C     COMMON /GSZ/M
      D=4.0
      Q=-1.0
      DO 100 I=1,N
C  NOTE ALL INTEGERS
      J=I/M
      J=M*J
      S=D*X(I)
C  SUBTRACT RHS FOR RESIDUAL
      IF(IFR)S=S-R(I)
C  LEFT EDGE
      IF(I-J.EQ.1)GOTO 20
      S=S+Q*X(I-1)
C  RIGHT EDGE
  20  IF(I.GT.J)S=S+Q*X(I+1)
C  TOP EDGE
      IF(I.GT.M)S=S+Q*X(I-M)
C  BOTTOM EDGE
      IF(I.LE.N-M)S=S+Q*X(I+M)
      V(I)=S
  100 CONTINUE
      RETURN
      END
      SUBROUTINE A25RQM(N,X,EPS,KPR,Y,Z,T,G,A,B,PO,IPR,APR,BPR)
C  ALGORITHM 25 RAYLEIGH QUOTIENT MINIMIZATION BY CONJUGATE GRADIENTS
C  J.C. NASH   JULY 1978, FEBRUARY 1980, APRIL 1989
C    N   =  ORDER OF PROBLEM
C    X   =  INITIAL (APPROXIMATE?) EIGENVECTOR
C   EPS  =  MACHINE PRECISION
C&&& for Microsoft test replace with actual names
C   APR,BPR  ARE NAMES OF SUBROUTINES WHICH FORM THE PRODUCTS
C          V= A*X    VIA   CALL APR(N,X,V)
C          T= B*X    VIA   CALL BPR(N,X,T)
C   KPR  =  LIMIT ON THE NUMBER OF PRODUCTS (INPUT) (TAKES ROLE OF IPR)
C        =  PRODUCTS USED (OUTPUT)
C  Y,Z,T,G,A,B RE WORKING VECTORS IN AT LEAST N ELEMENTS
```

```
C   PO   =  APPROXIMATE EIGENVALUE (OUTPUT)
C   IPR  =  PRINT CHANNEL   PRINTING IF IPR.GT.0
C  STEP 0
      INTEGER N,LP,IPR,ITN,I,LIM,COUNT
      REAL X(N),T(N),G(N),Y(N),Z(N),PN,A(N),B(N)
      REAL EPS,TOL,P0,PA,XAX,XBX,XAT,XBT,TAT,TBT,W,K,D,V,GG,BETA,TABT,U
C  IBM VALUE   -  APPROX. LARGEST NUMBER REPRESENTABLE.
C&&&       PA=R1MACH(2)
      PA=1E+35
      LIM=KPR
      KPR=0
      TOL=N*N*EPS*EPS
C  STEP 1
  10  KPR=KPR+1
      IF(KPR.GT.LIM)RETURN
C  FIND LIMIT IN ORIGINAL PROGRAMS
      CALL APR(N,X,A)
      CALL BPR(N,X,B)
C  STEP 2
      XAX=0.0
      XBX=0.0
      DO 25 I=1,N
        XAX=XAX+X(I)*A(I)
        XBX=XBX+X(I)*B(I)
  25  CONTINUE
C  STEP 3
      IF(XBX.LT.TOL)STOP
C  STEP 4
      P0=XAX/XBX
      IF(P0.GE.PA)RETURN
      IF(IPR.GT.0)WRITE(IPR,963)KPR,P0
 963  FORMAT( 1H ,I4,' PRODUCTS, EST. EIGENVALUE=',1PE16.8)
C  STEP 5
      PA=P0
C  STEP 6
      GG=0.0
      DO 65 I=1,N
        G(I)=2.0*(A(I)-P0*B(I))/XBX
        GG=GG+G(I)**2
  65  CONTINUE
C  STEP 7
      IF(IPR.GT.0)WRITE(IPR,964)GG
 964  FORMAT(' GRADIENT NORM SQUARED=',1PE16.8)
      IF(GG.LT.TOL)RETURN
C  STEP 8
      DO 85 I=1,N
        T(I)=-G(I)
  85  CONTINUE
C  STEP 9
      DO 240 ITN=1,N
C  STEP 10
        KPR=KPR+1
        IF(KPR.GT.LIM)RETURN
```

```
        CALL APR(N,T,Y)
        CALL BPR(N,T,Z)
C  STEP 11
        TAT=0.0
        TBT=0.0
        XAT=0.0
        XBT=0.0
        DO 115 I=1,N
        TAT=TAT+T(I)*Y(I)
         XAT=XAT+X(I)*Y(I)
          TBT=TBT+T(I)*Z(I)
         XBT=XBT+X(I)*Z(I)
 115    CONTINUE
C  STEP 12
        U=TAT*XBT-XAT*TBT
        V=TAT*XBX-XAX*TBT
        W=XAT*XBX-XAX*XBT
        D=V*V-4.0*U*W
C  STEP 13
        IF(D.LT.0)STOP
C  MAY NOT WISH TO STOP
C  STEP 14
        D=SQRT(D)
        IF(V.GT.0.0)GOTO 145
        K=0.5*(D-V)/U
        GOTO 150
 145    K=-2.0*W/(D+V)
 150    COUNT=0
C  STEP 15
        XAX=0.0
        XBX=0.0
        DO 155 I=1,N
          A(I)=A(I)+K*Y(I)
          B(I)=B(I)+K*Z(I)
          W=X(I)
          X(I)=W+K*T(I)
          IF(W.EQ.X(I))COUNT=COUNT+1
          XAX=XAX+X(I)*A(I)
          XBX=XBX+X(I)*B(I)
 155    CONTINUE
C  STEP 16
        IF(XBX.LT.TOL)STOP
        PN=XAX/XBX
C  STEP 17
        IF(COUNT.LT.N)GOTO 180
        IF(ITN.EQ.1)RETURN
        GOTO 10
C  STEP 18
 180    IF(PN.LT.PO)GOTO 190
        IF(ITN.EQ.1)RETURN
        GOTO 10
C  STEP 19
 190    PO=PN
```

```
        GG=0.0
        DO 195 I=1,N
        G(I)=2.0*(A(I)-PN*B(I))/XBX
        GG=GG+G(I)**2
 195    CONTINUE
C  STEP 20
        IF(GG.LT.TOL)GOTO 10
C  STEP 21
        XBT=0.0
        DO 215 I=1,N
         XBT=XBT+X(I)*Z(I)
 215    CONTINUE
C  STEP 22
        TABT=0.0
        BETA=0.0
        DO 225 I=1,N
          W=Y(I)-PN*Z(I)
          TABT=TABT+T(I)*W
          BETA=BETA+G(I)*(W-G(I)*XBT)
 225    CONTINUE
C  STEP 23
        BETA=BETA/TABT
        DO 235 I=1,N
          T(I)=BETA*T(I)-G(I)
 235    CONTINUE
C  STEP 24
 240  CONTINUE
C  STEP 25
      GOTO 10
C  NO STEP 26  -  HAVE USED RETURN INSTEAD
      END
```

Example output

?? explanation needed

```
gfortran ../fortran/a25.f
mv ./a.out ../fortran/a25.run
../fortran/a25.run < ../fortran/a25.in
```

```
##  GRID ORDER   3 EQNS ORDER    9  LIMIT= 100
##     1 PRODUCTS, EST. EIGENVALUE=  1.33333337E+00
##  GRADIENT NORM SQUARED=  1.77777791E+00
##     5 PRODUCTS, EST. EIGENVALUE=  1.17157304E+00
##  GRADIENT NORM SQUARED=  5.56937471E-13
##  RETURNED AFTER   5 PRODUCTS WITH EV=  1.17157304E+00
##  RESIDUAL NORM=  1.31790827E-07 /  3.43119442E-01=  3.84096069E-07
##  GRID ORDER  10 EQNS ORDER  100  LIMIT= 400
##     1 PRODUCTS, EST. EIGENVALUE=  4.00000244E-01
##  GRADIENT NORM SQUARED=  1.28000033E+00
##    15 PRODUCTS, EST. EIGENVALUE=  1.62028164E-01
##  GRADIENT NORM SQUARED=  7.54772778E-09
##  RETURNED AFTER  15 PRODUCTS WITH EV=  1.62028164E-01
##  RESIDUAL NORM=  5.59246428E-06 /  1.13465175E-01=  4.92879371E-05
##  GRID ORDER  -1 EQNS ORDER    1  LIMIT=   0
```

Pascal

Listing

```
procedure rqmcg( n : integer;
            A, B : rmatrix;
          var X : rvector;
          var ipr : integer;
          var rq : real);

var
  count, i, itn, itlimit : integer;
  avec, bvec, yvec, zvec, g, t : rvector;
  beta, d, eps, gg, pa, pn, step : real;
  ta, tabt, tat, tbt, tol, u, v, w, xat, xax, xbt, xbx : real;
  conv: boolean;

begin
  writeln('alg25.pas -- Rayleigh quotient minimisation');
  itlimit := ipr;
  conv := false;
  ipr := 0;
  eps := calceps;
  tol := n*n*eps*eps;

  pa := big;
  while (ipr<=itlimit) and (not conv) do
  begin
    matmul(n, A, X, avec);
    matmul(n, B, X, bvec);
    ipr := ipr+1;

    xax := 0.0; xbx := 0.0;
    for i := 1 to n do
    begin
      xax := xax+X[i]*avec[i]; xbx := xbx+X[i]*bvec[i];
    end;
    if xbx<=tol then halt;
    rq := xax/xbx;
    write(ipr,' products -- ev approx. =',rq:18);
    if rq<pa then
    begin
      pa := rq;
      gg := 0.0;
      for i := 1 to n do
      begin
        g[i] := 2.0*(avec[i]-rq*bvec[i])/xbx; gg := gg+g[i]*g[i];
      end;
      writeln(' squared gradient norm =',gg:8);
      if gg>tol then

      begin

        for i := 1 to n do t[i] := -g[i];
```

```
    itn := 0;
    repeat
      itn := itn+1;
      matmul(n, A, t, yvec);
      matmul(n, B, t, zvec); ipr := ipr+1;
      tat := 0.0; tbt := 0.0; xat := 0.0; xbt := 0.0;
      for i := 1 to n do
      begin
        xat := xat+X[i]*yvec[i]; tat := tat+t[i]*yvec[i];
        xbt := xbt+X[i]*zvec[i]; tbt := tbt+t[i]*zvec[i];
      end;

      u := tat*xbt-xat*tbt; v := tat*xbx-xax*tbt;
      w := xat*xbx-xax*xbt; d := v*v-4.0*u*w;
      if d<0.0 then halt;

      d := sqrt(d);
      if v>0.0 then step := -2.0*w/(v+d) else step := 0.5*(d-v)/u;

      count := 0;
      xax := 0.0; xbx := 0.0;
      for i := 1 to n do
      begin
        avec[i] := avec[i]+step*yvec[i];
        bvec[i] := bvec[i]+step*zvec[i];
        w := X[i]; X[i] := w+step*t[i];
        if (reltest+w)=(reltest+X[i]) then count := count+1;
        xax := xax+X[i]*avec[i]; xbx := xbx+X[i]*bvec[i];
      end;
      if xbx<=tol then halt
              else pn := xax/xbx;
      if (count<n) and (pn<rq) then
      begin
        rq := pn; gg := 0.0;
        for i := 1 to n do
        begin
          g[i] := 2.0*(avec[i]-pn*bvec[i])/xbx; gg := gg+g[i]*g[i];
        end;
        if gg>tol then
        begin
          xbt := 0.0; for i := 1 to n do xbt := xbt+X[i]*zvec[i];

          tabt := 0.0; beta := 0.0;
          for i := 1 to n do
          begin
            w := yvec[i]-pn*zvec[i]; tabt := tabt+t[i]*w;
            beta := beta+g[i]*(w-g[i]*xbt);
          end;
          beta := beta/tabt;

          for i := 1 to n do t[i] := beta*t[i]-g[i];
        end;
      end
```

```
        else
        begin
          if itn=1 then conv := true;
          itn := n+1;
        end;
      until (itn>=n) or (count=n) or (gg<=tol) or conv;
    end
    else conv := true;
  end
  else
  begin
    conv := true;
  end;
  ta := 0.0;
  for i := 1 to n do ta := ta+sqr(X[i]); ta := 1.0/sqrt(ta);
  for i := 1 to n do X[i] := ta*X[i];
end;
if ipr>itlimit then ipr := -ipr;
writeln;
end;
```

Example output

We use the same example as for Algorithm 15, with A the unit matrix and B the Frank matrix, here of order 5.

Note that we could modify the program to work with $-A$ to get the negative of the largest eigenvalue. Various shifting strategies might be used to get other solutions, but they are rather inconvenient and not recommended.

```
fpc ../Pascal2021/dr25.pas
# copy to run file
mv ../Pascal2021/dr25 ../Pascal2021/dr25.run
../Pascal2021/dr25.run <../Pascal2021/dr25p.in >../Pascal2021/dr25p.out
```

```
## Free Pascal Compiler version 3.0.4+dfsg-23 [2019/11/25] for x86_64
## Copyright (c) 1993-2017 by Florian Klaempfl and others
## Target OS: Linux for x86-64
## Compiling ../Pascal2021/dr25.pas
## dr25.pas(184,3) Note: Local variable "avec" not used
## dr25.pas(186,10) Note: Local variable "s" is assigned but never used
## Linking ../Pascal2021/dr25
## /usr/bin/ld.bfd: warning: link.res contains output sections; did you forget -T?
## 346 lines compiled, 0.2 sec
## 2 note(s) issued
```

```
Order of problem =5
Matrix A (unit)
Metric matrix B (frank)
Initial eigenvector approximation is all 1s
alg25.pas -- Rayleigh quotient minimisation
1 products -- ev approx. = 9.0909090909E-002 squared gradient norm = 7.2E-004
7 products -- ev approx. = 8.1014052771E-002 squared gradient norm = 3.2E-019
10 products -- ev approx. = 8.1014052771E-002 squared gradient norm = 1.6E-025
```

```
Solution after 11 products. Est. eigenvalue = 8.1014052771005207E-002
 0.1698911  0.3260187  0.4557341  0.5485287  0.5968848
Residuals
 5.61E-015  4.59E-014  7.77E-015 -1.24E-014 -2.09E-014
Sum of squared residuals = 2.7845218395467097E-027
```

Cleanup of working files

The following script is included to remove files created during compilation or execution of the examples.

```
## remove object and run files
cd ../fortran/
echo `pwd`
rm  *.o
rm  *.run
rm *.out
cd ../Pascal2021/
echo `pwd`
rm  *.o
rm  *.run
rm *.out
cd ../BASIC
echo `pwd`
rm *.out
cd ../Documentation
## ?? others
```

```
## /versioned/Nash-Compact-Numerical-Methods/fortran
## rm: cannot remove '*.o': No such file or directory
## rm: cannot remove '*.out': No such file or directory
## /versioned/Nash-Compact-Numerical-Methods/Pascal2021
## /versioned/Nash-Compact-Numerical-Methods/BASIC
## rm: cannot remove '*.out': No such file or directory
```

References

Nash, John C. 1979. *Compact Numerical Methods for Computers : Linear Algebra and Function Minimisation*. Book. Hilger: Bristol.