1968

QSLIN - A Fortran Subroutine Package for the Solution of Non-Linear Two-Point Boundary Value Problems

S. Silverston

Report Number:
68-017
QSLIN - A Fortran Subroutine Package
for the Solution of Non-Linear
Two-Point Boundary Value Problems

S. Silverston
April, 1968
CSD TR 17
QSLIN - A Fortran Subroutine Package for the Solution of Non-Linear Two-Point Boundary Value Problems

S. Silverston

QSLIN is used to solve non-linear two-point boundary value problems by the method of quasi-linearization. The linearized problem is solved by the method of superposition with orthonormalization using subroutine ORTNRM. Linear boundary conditions are assumed.

Let $x =$ independent variable

$u(x) =$ vector of $n$ dependent variables

$f(x,u) =$ a given vector of functions of $x$ and $u$,

\[ \text{in general non-linear} \]

$B =$ a constant $(n-k) \times n$ matrix

$D =$ a constant $k \times n$ matrix

$c_1 =$ a constant $(n-k)$-vector

$c_2 =$ a constant $k$-vector

We want to solve the problem:

\[ \frac{d}{dx} u(x) = f(x,u) \]

$Bu(a) = c_1, \; Du(f) = c_2$

The subroutine is entered by the statement:

CALL QSLIN
(INITL, NORM, W, OTHER, IT, CVCRIT, 
N,M,K,Y,DER,CD, 
A,NN,H,NP,NT, 
TEST, GX, NX, 
NFOO, NPC1, NPC2, ALT, 
NERR)
In the following discussion, we break the parameters into 6 groups. We refer throughout to the manual on Subroutine ORTNRM.

1. Iteration and convergence of linearized problem

\textbf{INITL,NORM, W, OTHER, IT, CVCRIT}

In the method of quasi-linearization, we start with an initial guess solution \( u^0(x) \), and solve successively the linear two-point boundary value problems

\[
\frac{d}{dx} u^i = \sum_{i=1}^{n} \frac{f(x, u^{i-1})}{u^{i-1}_j} (u^i_j - u^{i-1}_j) + f(x, u^{i-1}),
\]

\( i = 1, 2, \ldots \)

\( Bu^i(a) = c_1, \quad Du^i(b) = c_2. \)

After each iteration, we compute

\[
E_i = \|u^i - u^{i-1}\|_N
\]

in some norm \( N \). We consider the process to have converged when

\[
E_i < \text{crit.}
\]

where \( \text{crit.} \) is a pre-specified constant, and set

\[
u = u^i
\]

The iteration-and-convergence parameters should be set as follows:
3.

INITL: Name of subroutine for storing the initial guess solution \( u^0(x) \). To be called by a statement of the form

\[
\text{CALL INITL (UO) with:}
\]

\( \text{UO: Real array dimensioned } (NT,N) \quad (NT=\text{number of solution points, } N=\text{number of dependent variables } n. \quad \text{See ORTHNM write-up.) Values of } u^0(x) \text{ at the solution points.} \)

The subroutine should generate the values of \( u^0(x) \) for \( x \) equal to each of the NT solution points successively.

Note that here NT is the first subscript and N is the second.

NORM: Flag to indicate the norm to be used in computing the \( E_i \). There are three built-in norm options available, and also the option of the user's choosing a different one and coding it himself. The components \( (u^i - u^{i-1}) \) may be weighted differently if desired, using a weight vector \( w \) (see below).

\[
\begin{align*}
\text{NORM = 1, max norm,} \\
E_i &= \max_{1 \leq j \leq n} \left( \sum_{x \in \text{soln.pts}} w_j \right) \min \left\{ \max_{1 \leq j \leq n} \left| u^i_j(x) - u^{i-1}_j(x) \right| \right\} \\
&= 2, \text{ least squares norm,} \\
E_i &= \max_{1 \leq j \leq n} \left( \sum_{x \in \text{soln.pts.}} \frac{\left[ u^i_j(x) - u^{i-1}_j(x) \right]^2}{NT} \right)^{1/2}
\end{align*}
\]
\[ E_i = \max \left( w_j \sum_{x \in \text{soln.pts.}} \left| \frac{u_j^i(x) - u_j^{i-1}(x)}{\text{NT}} \right| \right) \]

\[ = 4, \text{ user-coded norm (See below: OTHER.)} \]

The "NT" in the above expressions is the number of solution points.

The "w_j" in the above expressions are the weights for the individual components \( j = 1, 2, \ldots, n \). See below: W.

Note that the norm is taken considering the values of \( u^i \) and \( u^{i-1} \) only at the solution points. Thus we see that the choice of solution points in QSLIN has significance beyond that of simple output of solution. More care must be given to choice of solution points in using QSLIN than was true in using ORTNRM directly.

\[ W: \text{ Real array dimensioned (N). Set of weights for use in computing the norm } \]

\[ E_i = \| u^i - u^{i-1} \| \]

See under NORM above for specific use in formulas. The elements of \( W \) should of course all be \( \geq 0 \).

OTHER: Name of subroutine for computing the norms \( E_i \) by a formula other than those supplied. (See above under NORM, options 1-3.) To be called by a statement of the form

\[ \text{CALL OTHER (EI, UIDM1, UI, W, N, NT) with:} \]

\[ EI: \text{ Real. Value of } E_i, \text{ to be computed and stored by the subroutine.} \]
UIM1: Real array dimensioned (NT,N). The values of the n-vector $u_{i-1}(x)$ at the NT solution points.

UI: Real array dimensioned (N,NT). The values of the n-vector $u_{i}(x)$ at the NT solution points. Note the unfortunate inconsistency in the dimensioning of UI and UIM1: UIM1 is (NT,N), while UI is (N,NT).

W: Real array dimensioned (N). The weights $w_i$. See above discussion of W.

N: Integer. Number of dependent variables n.

NT: Integer . Number of solution points.

The values of UIM1, UI, W, N, and NT should not be changed by OTHER.

OTHER is used if and only if the flag NORM is set equal to 4. If NORM is not set to 4, a dummy argument may be used for OTHER in the CALL QSLIN statement.

IT: Integer. The max number of iterations, producing successively, $u_{1}, u_{2}, \ldots$, to be allowed. If convergence of $u_{i}$ has not occurred after IT iterations, a note is printed, the error flag is set (see below under NERR), and control is returned from QSLIN to the calling program.

CVCRIT: Real. The convergence criterion for convergence of the $u_{i}$. When

$$E_{i} < CVCRIT$$

convergence is considered to have occurred. Of course CVCRIT should be set > 0.
None of the parameters NORM, W, IT, or CVCRIT are changed by QSLIN.

II. System of Equations

These parameters are covered by exactly the same rules as the comparable set in ORTNRM. The system of equations considered here is the linearized set of equations.

\[
\frac{d}{dx} u^i = \sum_{j=1}^{N} f(x,u_{i-1}^j) u_j^i + \sum_{j=1}^{N} f(x,u_{i-1}^j) u_{i-1}^j + f(x,u_{i-1}^i)
\]

Here, the values of \( u_{i-1}^j \) are considered as functions of \( x \) in the equations. The homogeneous set of equations, to be used in computing base solutions, includes only the first term on the right side. The second and third terms on the right side are used just for computing the particular solution.

In the subroutine DER which evaluates the vector \( \frac{d}{dx} u^i \), the user obtains the values of \( u_{i-1}^j(x) \) for any value \( x \) by means of the statement

CALL APFX(X, UIM1), with:

X: Real. Independent variable \( x \).

UIM1: Real array dimensioned \( (N) \). The vector of values \( u_{i-1}^j(x) \).
The subroutine APROX will supply the values of $u^{i-1}$ at a given $x$, by interpolating in the table of $u^{i-1}(x)$ at the solution points. Generally, the CALL APROX statement should appear as the first executable statement in DER. The value of $x$ is not changed by APROX.

As in ORTNRM, none of the parameters $N$, $K$, $Y$ are changed by QSLIN.

III. Interval and Spacing

$A$, $NH$, $H$, $NP$, $NT$

The parameters are set exactly as for ORTNRM. Their choice is a little more significant here than in ORTNRM, however, as noted earlier. Proper choice of intervals may improve convergence as well as accuracy, since solution values of $u^1$ are stored, and hence used in the subsequent iteration, at solution points only. The variable-step-size option provided for ORTNRM has its major utility here. Parameters are not changed by QSLIN.

IV. Orthonormalization

$TEST$, $C$, $NX$

These parameters are set exactly as for ORTNRM, for application to the linearized system of equations. They are not changed by QSLIN.
Note that, for purposes of orthonormalization each iteration \( i = 1, 2, \ldots \) is completely distinct. Thus \( N_X \) is merely the upper limit on orthonormalizations for any one iteration. It should not be the total number of orthonormalizations for all iterations.

V. Output Options

NPOO, NPO1, NPO2, ALT

The user may or may not want a full print-out of \( \bar{u}^i \) for every iteration. He may want to print only the last value of \( \bar{u}^i \), after convergence.
The same options for intermediate solution print-out, alternate solution, etc., that were available for ORTNRM are available here.

NPOO: Integer. Flag for print-out of the solutions $u^i$, $i = 1, 2, \ldots$

- $NPOO \leq 0$, no print-out of any of the $u^i$. This holds regardless of the value of $NP02$ (see below).
- $NPOO > 0$, print-out of $u^i$ given every $NPOO$ iterations, (i.e., for the $NPOO'$th, $2NPOO'$th, \ldots) and for the last solution generated (whether convergence occurred or the iteration limit IT was exceeded).

With each solution print-out, the error norm $E_1$ is also printed.

NPO1: Integer. Flag for print-out of intermediate vectors. Options are the same as for ORTNRM. Intermediate vector print-out is given for a particular iteration only when solution print-out for that iteration is flagged via $NPOO$.

NP02: Integer. Flag for print-out of solution vectors $u^i$. Options are the same as for ORTNRM, subject to the following exceptions and remarks:

1) The normalization of $u$ for homogeneous systems, as flagged in ORTNRM by setting $NP02$ negative, is not available for QSLIN.
2) If an alternate solution print-out is desired (and flagged by setting NPO2 = 3), it is given \textit{only} with the final solution print-out, i.e., after convergence.

3) A solution $u$ is always generated. NPO2 = 0 is considered the same as NPO2 = 1.

4) If solution print-out for a particular iteration is indicated as per NPOO but NPO2 is set to 0 or 1, no solution print-out is given, but the error norm is printed for that iteration.

\textbf{ALT:} Name of subroutine for computing an alternate solution. Same as for ORTNRM. If NPO2 ≠ 3, a dummy name may be used for ALT in the CALL ORTNRM statement.

The parameters NPOO, NPO1, and NPO2 are not changed by QSLIN.

VI. \textbf{Error Flag}

\textbf{NERR}

There are two error conditions which may be flagged in QSLIN. If there is no error, NERR will have been set to 0 on return from QSLIN.

\textbf{NERR:} Integer.

- Set to 0 if no error was encountered.
- Set to 1 if more orthonormalizations than the number specified by NX were required for some iteration.
- Set to 2 if convergence of the $u^i$ did not occur after the number of iterations specified by IT.
The user must include the names for INITL, DER, CO, and, if the respective options are used, OTHER and ALT in an EXTERNAL statement.

Storage Space

The user must allocate working storage space for use by QSLIN. As with ORTNRM, this is done via the labeled COMMON block /SCRATCH/. The QSLIN package will use the first LQ locations of /SCRATCH/, where

\[ LQ = (NT+6)\times N \times M + (3 \times K + K(K-1)/2 + 1) \times N \times X + K + (NT+1)(N+1) \]

Note that this is just \((NT+1)\times N\) more locations than are needed for ORTNRM. (As for ORTNRM, for homogeneous systems \(K\times N\times X\) less locations are necessary; however, it is not likely that QSLIN will be used for a homogeneous system.)

Upon return from QSLIN, the first \((NT,N)\) locations of /SCRATCH/ will contain the solution \(u\) at the solution points. For example, suppose that

\[ N = 4, M = 3, K = 2, NT = 11, NX = 25. \]

The following COMMON statement might be employed to get at the solution \(u\):

\[ \text{COMMON /SCRATCH/ U(11,4), S(422)} \]

Note here the following unfortunate inconsistency between QSLIN and ORTNRM: in ORTNRM, the first \((N,NT)\), rather than \((NT,N)\), locations of /SCRATCH/ contained \(u(x)\).
Auxiliary COMMON blocks

QSLIN uses, in addition to /SCRATCH/, COMMON blocks named

/KKKK/ and /MMMM/,

just like ORTNRM. Similarly, QSLIN does not use blank COMMON.

Alternate Entry

No alternate entry in QSLIN itself. SOLN is not pertinent to QSLIN, since solution is always generated and flag NPO2=0 is ignored. PRM may be used as with ORTNRM. See the ORTNRM manual under Alternate Entry for a discussion of SOLN and PRM.

Backward Integration

As in QSLIN, solution is generated in the direction of increasing x, regardless of the sign of the step-size H. In this regard, the user must remember to write subroutine INITL so as to store $u^0$ in the array UO(NT,N) in order of increasing x. I. e.,

$$UO(1,1) = u_1^0(\min(a,b))$$ etc.

Deck Set-up

The QSLIN package consists of the following subroutines:

QSLIN
XLIST
APROX
CNORK
PIF4
The ORTNRM package consisting of:

ORTNRM
ORTSUB
RUNKUT
NUGO
ARRAY
RND
FLIP
BLOCK DATA

The package uses COMMON blocks named

/SCRATCH/ (discussed above)
/KKKK/
/MMMM/

The QSLIN-ORTNRM package should be placed after the calling program in the deck to allow proper loading of COMMON blocks.

References


APPENDIX

Fortran listing of QSLIN package
SUBROUTINE QSLIN
PARAMETERS GOVERNING ITERATION AND CONVERGENCE
I: INITL, NORM, W, OTHER, IT, CVCRIT,
PARAMETERS SPECIFYING SYSTEM OF EQUATIONS
N, M, K, Y, DER, CO,
PARAMETERS DEFINING INTERVAL AND SPACING
A, NN, H, NP, NT,
PARAMETERS SPECIFYING ORTHONORMALIZATION
C, NX,
SPECIFICATION OF USERS OUTPUT OPTIONS
NPOO, NPO1, NPO2, ALT,
ERROR FLAG
NERR
DIMENSION Y(N*M),W(N),CN(2)
COMMON /SCRATCH/ S(l), NTK, NTK, NK, NTK, NT, DONE, PR, PRINT, DATA CN(5H), NO, 1H, /
EXTERNAL DER, CO, ALT, OTHER
NTK = NT
NK = N
DONE = .FALSE.,
PRINT = NPOO .GT. 0
PO = NPO1 .GT. 0 OR NPO2 .GT. 1
N2 = MINO(MAXO(NPO2+1),2)
CALL INITL(S)
L = NT*N + 1
CALL XLIST(I, S(L), NT, A, NN, H, NP)
K2 = L + NT
K1Q = K2 - 1
I = 0
10 I = I + 1
K1 = K1Q
NP1 = 0
NP2 = 1
PR = MOD(I, NPOO) .EQ. 0 .AND. PRINT
IF (.NOT. PR) GO TO 30
NP1 = NPO1
NP2 = N2
IF (PO) WRITE (6, 20) I
20 FORMAT (6H1GS|_I N  // 13H  ERROR NORM = E10.3)
IF (DONE) GO TO 55
K1 = 0
IF (NERR .NE. 0) GO TO 100
CALL CNORM(5H, N, NT, NORM, ENRM, OTHER)
IF (PR .AND. PO) WRITE (6, 40); ENRM
40 FORMAT (14HITERATION NO. 14 // 13H ERROR NORM = E10.3)
IF (DONE) GO TO 55
KV = 1
IF (ENRM .LT. CVCRIT) KV = 2
IF (KV .EQ. 1 .AND. IT .GT. 1) GO TO 10
NERR = 2 - KV
IF (.NOT. PRINT) RETURN
WRITE (6, 50); ENRM
50 FORMAT (1H1 A6, 17HCONVERGENCE AFTER 14, 11H ITERATIONS //
13H ERROR NORM = E10.3 //
13H SOLUTION PRINT-OUT FOR ONE ADDITIONAL ITERATION FOLLOWS.)
NP1 = NPO1
NP2 = MAXO(NPO2, 2)
I = I + 1
DONE = *TRUE*
PR = *TRUE*
PD = *TRUE*
K1 = K10
GO TO 50
50 IF (K1*EQ.2) RETURN
WRITE (6,60) I
60 FORMAT (52H0QSLIN COMPUTATION TERMINATED, TOO MANY ITERATIONS (  
1 14, 2H) )
NERR = 2
RETURN
100 WRITE (6,110)
110 FORMAT (30H0QSLIN COMPUTATION TERMINATED )
RETURN
END

SUBROUTINE XLIST (XL,NT,XO,NN,H,NP)
DIMENSION XL(NT),NN(I),H(I),NP(I)
XL(1) = XO
NNC = 1
DX = H(NNC)*FLOAT(NP(NNC))
NPC = 0
1 = 1
10 IF (NPC+LT,NNC(I)) GO TO 15
NPC = NPC + 1
NNC = NNC + 1
DX = H(NNC)*FLOAT(NP(NNC))
15 NPC = NPC + 1
II = 1 + 1
XL(II) = XL(I) + DX
I = II
1F (I+LT,NT) GO TO 10
RETURN
END

SUBROUTINE APROX (X,F)
DIMENSION F(I)
COMMON /SCRATCH/ S(I)
DO 20 I = 1,N
M = NT*I - NT + 1
20 F(I) = PSI(MAX,ABS(Y(I,J) - Z(J,I)))
RETURN
END

SUBROUTINE CNORM (Z,Y,W,N,NT,NORMO,ENRM,ORIG)
DIMENSION Z(NT,N),Y(N,NT),W(N)
ENRM = 0.
GO TO (10,20,30,40),NORMO
MAX NORM
10 DO 15 I = 1,N
8 = 0.
15 DO 13 J = 1,NT
B = AMAX1(B,ABS(Y(I,J) - Z(J,I)))
13 Z(J,I) = Y(I,J)
15 ENRM = AMAX1(ENRM,W(I)*B)
RETURN
LEAST SQUARES
20 DO 25 I = 1,N
B = 0.
C
25 GO TO (10,20,30,40),NORMO
MAX NORM
30 C
40 C

DO 23 J = 1, NT
B = B + (Y(I,J) - Z(I,J))**2
23 Z(I,J) = Y(I,J)
RETURN

C
SUM OF ABS. VAL.
DO 30 I = 1, N
B = 0.
DO 33 J = 1, NT
B = B + ABS(Y(I,J) - Z(I,J))
33 Z(I,J) = Y(I,J)
35 ENRM = AMAX1(ENRM, W(I) * SQRT(B / FLOAT(NT)))
RETURN

C
USER-CODED NORM
CALL ORIG(ENRM, Z, Y, W, N, NT)
DO 45 J = 1, NT
45 Z(I,J) = Y(I,J)
RETURN

END
FUNCTION PIF4(X, XLIST, N, FLIST)
DIMENSION XLIST(N), FLIST(N)
BLI(Q004FL, Q001FL, Q002FL, Q003FL, Q004FL) = (Q001FL - Q000FL) * (Q003FL - Q004FL)
Q004FL = Q000FL
DO 8 I = 1, N
8 CONTINUE
CONTINUE
I = N - 1
GO TO 5
1 IF (X - XLIST(1)) 4, 4, 6
4 I = 1
5 K = 1
GO TO 30
6 DO 8 I = 1, N
9 IF (X - XLIST(I)) 9, 9, 8
8 CONTINUE
I = 0
9 I = I - 1
K = 0
30 BLIF1 = BLI(X, XLIST(I), XLIST(I + 1), FLIST(I), FLIST(I + 1))
IF (K - 1) 11, 10, 11
10 PIF4 = BLIF1
RETURN
11 IF ((I + 2) - N) 12, 12, 13
12 IF ((I - 1) - 1) 14, 16, 16
13 L = 1
GO TO 15
14 L = I + 2
15 K = 2
GO TO 17
16 L = I + 2
17 BLIF2 = BLI(X, XLIST(I), XLIST(L), FLIST(I), FLIST(L))
BLIF3 = BLI(X, XLIST(I + 1), XLIST(L), BLIF1, BLIF2)
IF (K - 2) 19, 18, 19
18 PIF4 = BLIF3
RETURN
19 BLIF4 = BLI(X, XLIST(I), XLIST(I - 1), FLIST(I), FLIST(I - 1))
BLIF5 = BLI(X, XLIST(I + 1), XLIST(I - 1), BLIF1, BLIF4)
BLIF6 = BLI(X, XLIST(I + 2), XLIST(I - 1), BLIF3, BLIF5)
IF ((I + 3) - N) 20, 20, 23
20 IF ((I-2)-1) 21,21,21
21 IF (ABS(X-XLIST(I+3)) - ABS(X-XLIST(I-2))) 22,22,23
22 M=I+3
23 GO TO24
24 BLIF7 =BLI(X,XLIST(I+1),XLIST(M),FLIST(I),FLIST(M))
BLIF8 =BLI(X,XLIST(I+1),XLIST(M),FLIST(M),BLIF7)
BLIF9 =BLI(X,XLIST(I+2),XLIST(M),BLIF3,BLIF8)
PIF4 = BLI(X,XLIST(I-1),XLIST(M),BLIF6,BLIF9)
RETURN
END