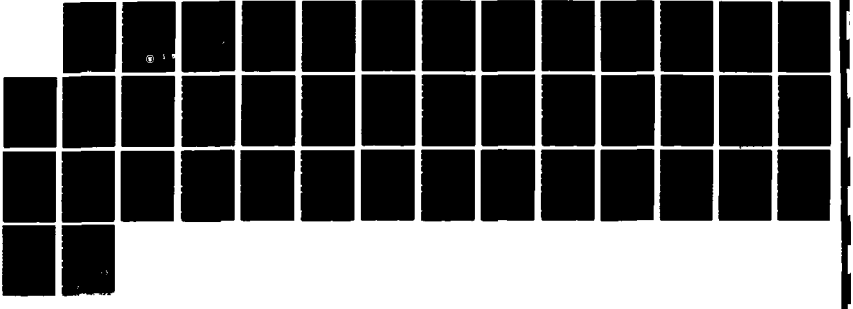
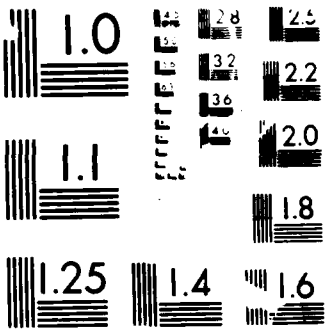
AD-A189 809 NFEARS (NONLINEAR FINITE ELEMENT ADAPTIVE RESEARCH SOLVER) A NONLINEAR AD (U) PITTSBURGH UNIV PA INST FOR COMPUTATIONAL MATHEMATICS AND APP 1/1

UNCLASSIFIED C K MESZTENYI ET AL DEC 87 F/G 12/5 NL





MICROCOPY RESOLUTION TEST CHART

NS 1963 A

INSTITUTE FOR COMPUTATIONAL MATHEMATICS AND APPLICATIONS

ICMA-87-114 December 1987

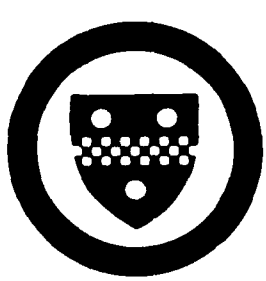
NFEARS
A Nonlinear Adaptive Finite Element Solver [1]

Part II: User's Manual

by

Charles K. Mesztenyi [2] and Werner C. Rheinboldt [3]

Department of Mathematics and Statistics
University of Pittsburgh



(2)

ICMA-87-114 December 1987

NFEARS
A Nonlinear Adaptive Finite Element Solver [1]

Part II: User's Manual

by

Charles K. Mesztenyi [2] and Werner C. Rheinboldt [3]

[1] This work was in part supported by the Office of Naval Research under contracts N-00014-80-C-9455 and N-00014-85-K-0169 and by the National Science Foundation under grant DCR-8309926. Acknowledgment is also made for the partial support of the Computer Science Center of the University of Maryland

[2] Computer Science Center, University of Maryland, College Park, MD 20742
[3] Department of Mathematics and Statistics, University of Pittsburgh, Pittsburgh, PA 15260

Contents

Preface to NFEARS ii

Part I. Mathematical Foundations

Introduction 1

I.1. Problem Class 4

I.2. Domains and Mappings 7

I.3. Finite Element Approximation 13

I.4. Mesh Representation and Densities 19

I.5. The Solution Manifold 26

I.6. The Continuation Process 29

I.7. Simplicial Approximation of the Solution Manifold 32

I.8. Error Estimation and Mesh Adaptation 36

I.9. References 42

Part II. NFEARS User's Manual

Preface to the User's Manual 45

II.1. NFEARS Program 46

II.2. Geometry Input Preparation 48

II.3. User Supplied Subroutines 55

II.4. Running NFEARS 60

II.5. NFEARS Commands 65

II.6. Region Subcommands 74

Preface to the User's Manual

This represents the second Part of the report on NFEARS, the "Nonlinear Finite Element Adaptive Research Solver" developed jointly by the Universities of Maryland and Pittsburgh. This part constitutes the User's Manual for the system. It was intended to describe all necessary aspects for running NFEARS successfully without requiring a detailed knowledge of the mathematical background given in Part I. However, the reader should be generally familiar with the aims and tasks of the program.

II.1. NFEARS Program

NFEARS is available for VAX and Unisys Computers. In order, to use the program the user is required to write subroutines in Fortran 77 describing the problem to be solved (see Sections I.1 and II.3) and to combine them with the NFEARS program. Although, for the most part, NFEARS is written in standard Fortran 77, some special features are assumed to be available in the compiler. The principal non-standard feature is the use of the INCLUDE statement which allows for the inclusion of program-segment-files.

NFEARS uses labeled common-storage areas extensively for its internal data structure. All of these labeled common-storage areas are defined by declarations in individual files which are then inserted into the program files by means of INCLUDE statements. One of these individual files, MAXDIM, declares parameter values which in turn are used for dimensioning various arrays. These parameter values limit the current size of the problem. If a given problem exceeds these limitations, MAXDIM should be edited for larger values, and NFEARS should be recompiled. The following limitations are presently set up in MAXDIM:

M0MAX	= 16	Maximum number of 0-D domains
M1MAX	= 12	Maximum number of 1-D domains
M2MAX	= 5	Maximum number of 2-D domains
M1TMAX	= 120	Maximum number of free 1-D nodes
M2TMAX	= 100	Maximum number of free nodes in one 2-D domain
M21MAX	= 200	Maximum number of free nodes in one 2-D domain and its boundary
M01DFX	= 5000	Maximum size of the Jacobian corresponding to the free nodes in the 0/1-D domains
M2DFMX	= 10000	Maximum size of the Jacobian corresponding to the free nodes in one 2-D domain and its boundary

Other possible machine or installation dependent parts of the NFEARS program occur in the program segment file IOPROG in connection with the handling of

disk files. When NFEARS is run the following disk files are used by unit numbers :

5	fs	System input file
6	fs	System output file
9	us	Used by SAVE/RESET to equate user's file
10	fs	Log-file
12	ud	2-D tree file
14	ud	2-D vector file
15	us	Neumann condition assembly file
16	us	Element assembly file
18	ud	2-D Jacobian file
20	s	Temporary file used to equate user's geometry input file and also used by the region calculation
21-29	us	Region center-point save files
31-...	us	Region output files

(u=unformatted, f=formatted, s=sequential, d=direct access)

Although NFEARS opens these files, it does not check whether they did exist before. Thus the user should not have cataloged and assigned files with the above unit numbers. Normally, NFEARS is used interactively, and, in order, to avoid excessive printout to the terminal and to keep a record of the run, a Log-file is established on unit 10 . When an NFEARS session is terminated with the QUIT command, this Log-file (10), as well as the Region files (31-...), are not saved separately. It is left to the user to print out the formatted, sequential Log-file and to save the unformatted Region files for post-processing.

Before any use of NFEARS the user is required to perform the following two steps:

(a) To prepare the geometry input describing the domain Ω, and

(b) to write the user supplied subroutines describing the mathematical problem and to combine them with NFEARS in the form of an executable module.

These steps are described in detail in Sections II.2and II.3

II.2. Geometry Input Preparation

The preparation of the geometry input consists of the following six steps:

(a) Subdivision of the domain Ω.
(b) Assignment of directions for the 1-D domains.
(c) Numbering of all subdomains.
(d) Definition of an initial mesh.
(e) Specification of an initial solution.
(f) Construction of the geometry input file.

These steps are illustrated with a simple example in Figures II.3.1 -II.3.3.

(a) Subdivision of the domain Ω:

As discussed in Section I.2, the domain Ω must be subdivided into generalized quadrilaterals each with four corner points and four sides. As before, we call the open quadrilaterals 2-D domains (Ω_k^2, k=1,...,N_2), the open sides 1-D domains (Ω_k^1, k=1,...,N_1), and the corner points 0-D domains (Ω_k^0, k=1,...,N_0). Any 1-D domain is either a side of exactly one 2-D domain in which case it is part of the external boundary of Ω, or it is a side of two 2-D domain in which case it is contained in the interior of Ω. As shown in Figure I.2.1, angles formed at the corner points of the 2-D domains should be between α and 180-α degrees with a suitable tolerance α to avoid numerical instabilities; a value of $\alpha \approx 15^o$ has been found adequate.

Figure II.3.1 shows an example where the domain is a quarter disk with Dirichlet boundary conditions on the horizontal line (fixed boundary) and Neumann conditions on the rest of the boundary. The right side of the figure shows a possible subdivision into three 2-D domains, nine 1-D domains and seven 0-D domains. Thus, in this case, we have $N_2=3$, $N_1=9$ and $N_0=7$. It should be noted that, in line with our definition of the admissible meshes in Section I.3, a basic mesh Δ is automatically introduced on Ω once the initial subdivision is given; namely, the mesh consisting exactly of 4 superelements on each 2-D domain.

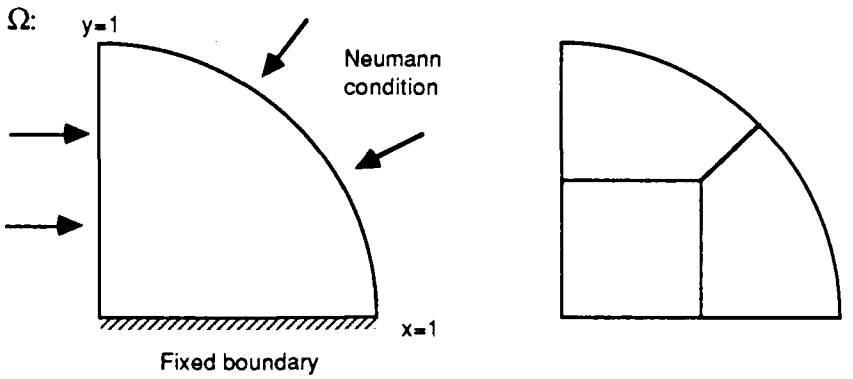


Figure II.3.1

(b) Assignment of directions for the 1-D domains.

As detailed in Section I.2, directions have to be assigned to all 1-D domains in order to define their tangent and normal vectors and also their curvature C. On 1-D domains which carry Neumann conditions this assignment must be uniform in the sense that all normals point either outward or inward to the domain Ω and, hence, are not mixed. As discussed in Section I.2, the normal vector is obtained from the tangent vector by rotating the latter counter-clockwise through 90°. Figure II.3.2 shows a possible assignmement of directions for our example.

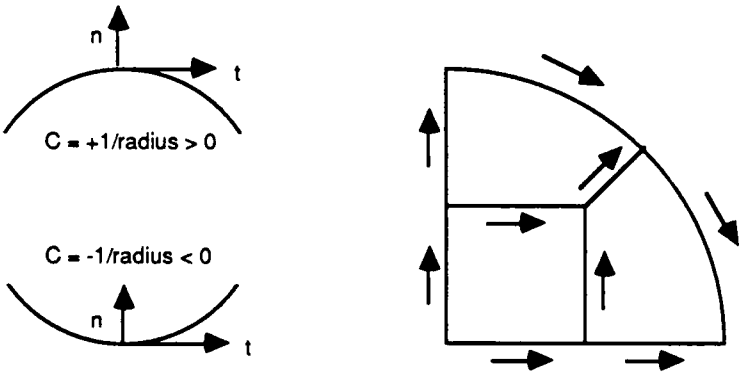


Figure II.3.2

(c) Numbering of all subdomains.

The next step is to number all 0-D, 1-D and 2-D domains. Within each group the numbers should start with 1, and end with N_0, N_1 and N_2, respectively. The order of the 0-D and 2-D domains is irrevelant. However, some savings in speed and memory space can be achieved if the 1-D domains are numbered as follows: Begin by numbering the 1-D domains which carry Dirichlet conditions, then continue with the others by using a "wavefront" to move over the subdivision. Figure II.3.3 shows such a numbering for our example.

(d) Definition of an initial mesh:

In NFEARS meshes are specified in terms of density functions and intensity values. Section I.4 presents the definition of the density function D_Ω on the domain Ω and gives an algorithm for the construction of the mesh from D_Ω and the given intensity $\Im$. The un-normalized density d_Ω is specified in terms of 29 coefficients $p_0,\ldots,p_9$ for each closed 2-D domain. In order to simplify the definition of the starting mesh, NFEARS reduces this input requirement by asking only for a few of these coefficients and by performing linear interpolation to get all others. More specifically, NFEARS requires one coefficient value for each 2-D domain, one for each 1-D domain, and two for each 0-D domain. The single coefficient values for the 2-D and 1-D domains are assigned to the mid-points of these sub-domains as their appropriate p_i-value, $1\le i\le 9$. The first of the two coefficient values for a 0-D domain is again used as their p_i-value, $1\le i\le 9$, while the second coefficient is the p_0 value which describes the singularity. Note that the p_0 values must be either zero or negative. Figure I.3.4 shows the mesh generated from the indicated initial density values for a closed 2-D domain. More specifically, the initial coefficients are shown adjacent to the corners for the 0-D domains (where the second line is p_0), along the sides for the 1-D domains, and near the mid-point of the 2-D domain. With each picture, the intensity is listed. It is advisable to start with uniform coefficient values, and then to increase the $p_1,\ldots,p_9$ values in areas where a singularity is expected.

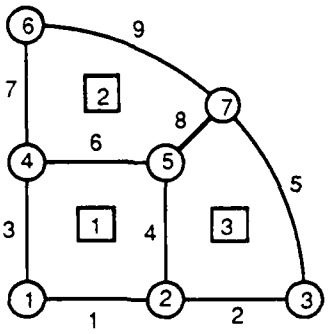
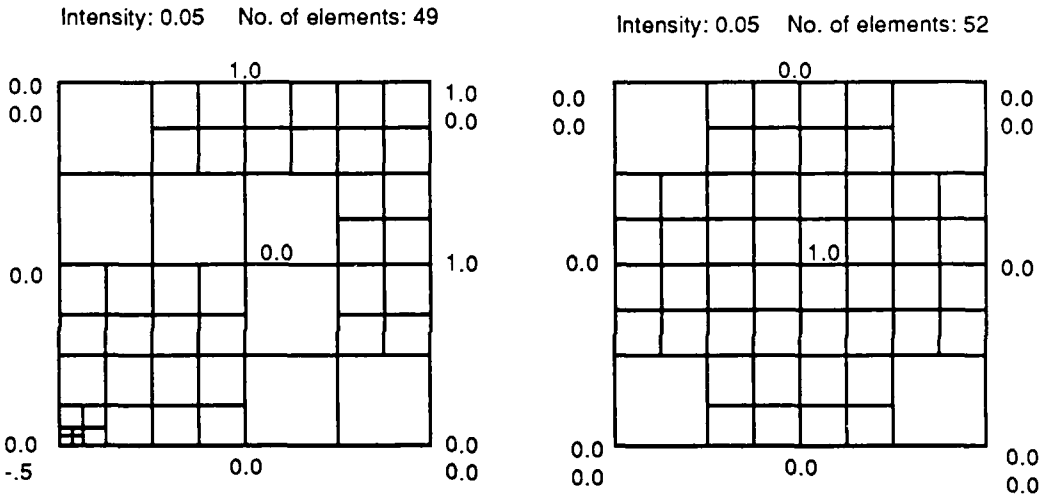


Figure II.3.3



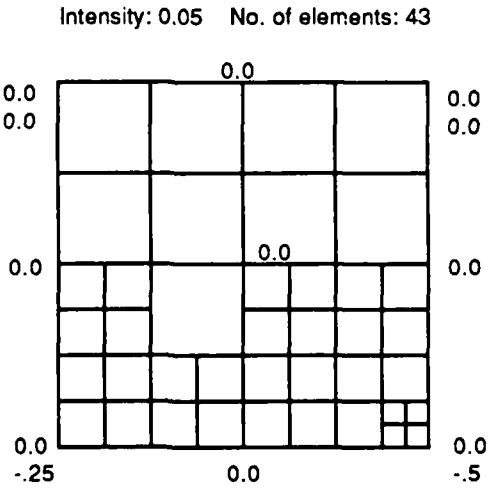


Figure II.3.4

(e) Specification of an initial solution.

For all calculations NFEARS requires an starting solution on the nodes of the initial mesh (see Section I.6). Once again, in order to simplify the input, NFEARS asks only for a reduced number of solution values and uses biquadratic interpolation to determine the other ones. More specifically, one value is required for each 0-D domain, one each at the mid-points of the 1-D domains, and one each at the mid-points of the 2-D domains. The biquadratic interpolation is based on the local coordinate system as defined in Section I.2. It should be noted that the initial solution values also specify the Dirichlet boundary conditions on the relevant 1-D domains; in other words, they define the corresponding boundary functions b as quadratic functions in the local coordinates (see Section I.2). In our example, we assumed a zero initial solution and zero Dirichlet conditions.

(f) Construction of the geometry input file.

NFEARS permits either an interactive input of the geometry or a read-in of a prepared geometry input file. In order to avoid typing errors it is generally advisable to set up a geometry input file. This input file has to consist of $N_0+N_1+N_2+3$ data lines in free format where, again, N_0, N_1 and N_2 denote the number of 0-D, 1-D and 2-D domains, respectively. The general format is as follows:

N_0	N_0 = Number of 0-D domains;
$1,x_1,y_1,b_1,u_1,p_1,p_1^0$	index i of the i-th 0-D domain;
$2,x_2,y_2,b_2,u_2,p_2,p_2^0$	x_i,y_i = global coordinates of the 0-D domain;
.....	b_i = 0 if this 0-D domain is free,
.....	=1 if it carries a σ_4-dependent Dirichlet
.....	condition ,
.....	=2 if it carries a fixed Dirichlet condition ;
.....	u_i = initial solution value;
.....	p_i = density coefficient;
.....	p_i^0 = singularity coefficient
$N_0,x_{N0},y_{N0},b_{N0},u_{N0},p_{N0},p_{N0}^0$	
N_1	N_1 = Number of 1-D domains;
$1,J_1,K_1,b_1,C_1,u_1,p_1$	index i of the i-th 1-D domain;
$2,J_2,K_2,b_2,C_2,u_2,p_2$	J_i, K_i = indices of the adjacent 0-D domains
.....	(from - to),
.....	b_i = 0 this 1-D domain is free,
.....	= 1 if it carries a σ_4-dependent Dirichlet
.....	condition,
.....	= 2 if it carries a fixed Dirichlet condition,
.....	= -1 if it carries a Neumann condition;
.....	C_i = signed curvature of the 1-D domain;
.....	u_i = solution value at its mid-point
.....	p_i = coefficient of the density function at the
.....	mid-point;
$N_1,J_{N1},K_{N1},b_{N1},C_{N1},u_{N1},p_{N1}$	
N_2	N_2= Number of 2-D domains;
$1,I_1,J_1,K_1,L_1,u_1,p_1$	index i of the i-th 2-D domain;
$2,I_2,J_2,K_2,L_2,u_2,p_2$	I_i,J_i,K_i,L_i = indices of the adjacent 1-D domains
.....	ordered counter clockwwise;
.....	u_i = initial solution at the mid-point
.....	p_i = coefficient of the density function at the
.....	mid-point
$N_{N2},I_{N2},J_{N2},K_{N2},L_{N2},u_{N2},p_{N2}$	

Notes:

1. When a 1-D domain carries a Dirichlet boundary condition (b_i= 1 or 2), then the two bounding 0-D domains should have the same type of boundary condition.
2. The definition of the sign of the curvature for a 1-D domain is indicated in Figure II.3.2; that is, if we look from the starting 0-D domain , J_i, toward the terminating 0-D domain, K_i, then the positive (+) sign or negative (-) sign is to be used when the center of the circle is on the right or the left side, respectively. For straight lines, the value of the curvature is zero.
3. The indices of the four 1-D domains that bound a 2-D domain have to be listed in counter-clockwise order. When the 2-D domain is mapped into the units-quare, the first 1-D domain is mapped into the η axis and the second one onto the ξ axis (see Section I.2).

For our example, the input file has the following form:

```
7                                   Number of 0-D domains
1,0.,0.,2,0.,0.,-.5                 Data for the seven 0-D domain
2,.5,0.,2,0.,0.,0.
3,1.,0.,2,0.,0.,0.
4,0.,.5,0,0.,0.,0.
5,.5,.5,0,0.,0.,0.
6,0.,1.,0,0.,0.,0.
7,.70710678,.70710678,0,0.,0.,0.
9                                   Number of 1-D domains
1,1,2,2,0.,0.,0.                    Data for the nine 1-D domains
2,2,3,2,0.,0.,0.
3,1,4,-1,0.,0.,0.
4,2,5,0,0.,0.,0.
5,7,3,-1,1.,0.,0.
6,4,5,0,0.,0.,0.
7,4,6,-1,0.,0.,0.
8,5,7,0,0.,0.,0.
9,6,7,-1,1.,0.,0.
3                                   Number of 2-D domains
1,3,1,4,6,0.,0.                     Data for the three 2-D domains
2,7,6,8,9,0.,0.
3,4,2,5,8,0.,0.
```

II.3. User Supplied Subroutines

Before running NFEARS, the user must write subroutines which calculate the values and derivatives of the functions Φ, G_2, G_1 in the problem-definition of Section I.1, and which set up or modify certain parameters and print out appropriate headings. All these subroutines carry entry names beginning with USR... .Note also that all of them must be provided even if some are not in use, since the operating system usually does not handle missing subroutines. Once these routine have been written and compiled, they have to be combined with NFEARS to produce an executable module.

NFEARS provides a common block

/USRPAR/ FUSER(10,2), IUSER(10)

for up to 20 real and 10 integer valued parameters for use in the supplied subroutines. The subroutine USRFCT allows for storing of data in these two arrays during the initial call while USRMOD permits their later modification. These data are retained and may be used in all other user subroutines. They are also saved by a SAVE command , and read back by a RESET command.

The following subroutines must be provided:

USRFCT	This routine is called at initialization time. It may also be used to print out some captions.
USRINV	Routine to provide an initial solution for the problem. It is called by the "INVAL" command.
USRMOD	Routine to provide, change, or print user parameter in USRPAR during the process.
USRPH1	Routine to provide the first derivatives of the function Φ at specified points.
USRPH2	Routine to provide the second derivatives, including the derivatives by σ_1, of the function Φ at specified points.
USRG1	Routine to provide the values and derivatives by σ_3 of the function G_1 at specified points.
USRG2	Routine to provide the values and derivatives by σ_2 of the function G_2 at specified points.

The calling sequences are as follows:

SUBROUTINE USRFCT (I, NU, IDPR, IDUF, IDUFP, IDUG2, IDUG1)

This subroutine is called at the initialization of a problem either by an INIT command or a RESET command. It is expected to print out a caption for the run.

Input arguments:

I = 0 for initial start with INIT,
= 1 for recall of saved data with RESET

NU = Fortran unit number (6 or 10) where echo print should be directed

Output arguments (if I=0):

IDPR = Problem number

IDUF = ID number of the Φ function

IDUFP = 0 if Φ does not depend on σ_1, non-zero otherwise

IDUG2 = signed identification number of G_2 as follows:
= 0 if zero; that is, if the G_2 term does not exist
< 0 if G_2 is independent of σ_2
> 0 if G_2 depends on σ_2

IDUG1 = signed identification number of G_1 as follows:
= 0 if zero; that is, if the G_1 term does not exist
< 0 if G_1 is independent of σ_3
> 0 if G_1 depends on σ_3

As noted before, in all cases, the program should print a title for the run. The other arguments should be set by the user if I=0. They will have been set before when I=1, but may be reset by the routine. The integer valued identification numbers are provided for identification pruposes only. NFEARS merely checks whether they are zero, positive or negative integers.

SUBROUTINE USRINV (N, XG, U)
DIMENSION XG(2,N), U(N)

This subroutine provides initial solution values U at N points in one 2-D domain with the coordinates x,y specified in the array XG. This routine is called by the INVAL command. The routine is called first with N=0 to allow for any initialization, such as the input of a file of data. Thereafter it is called for all regular free nodes of the problem. Note that an initial solution can be specified by the geometry input in which case USRINV may be a dummy subroutine. But then the INVAL command should never be used.

SUBROUTINE USRMOD (NU)

This subroutine allows for the modification and print-out of the parameter values in the common block /USRPAR/. It is called by the command UMOD. The input integer NU is the Fortran unit number (6 or 10) where the printed output should be directed.

SUBROUTINE USRPH1 (N, IX2, S1, XG, U, P)
DIMENSION S1(2), XG(2,N), U(0:2,N), P(0:2,N)

This subroutine evaluates the first derivatives of Φ at N points in one 2-D domain, and returns the results in the array P.

Input arguments:

N = number of points where the first derivatives of Φ are to be evaluated

IX2 = index value of the 2-D domain containing the points

S1 = the components of the parameter σ_1

XG(1,K), XG(2,K)
= global coordinates x,y of the point K (K = 1,...,N)

U(0,K) = the solution value u at the point K

U(1,K) = the value of the derivative $u_x = \partial u/\partial x$ at the point K

U(2,K) = the value of the derivative $u_y = \partial u/\partial y$ at the point K

Output arguments:

P(0,K) = $\partial\Phi/\partial u$ at the point K (K=1,...,N)

P(1,K) = $\partial\Phi/\partial(u_x)$ at the point K (K=1,...,N)

P(2,K) = $\partial\Phi/\partial(u_y)$ at the point K (K=1,...,N)

SUBROUTINE USRPH2 (N, IX2, S1, XG, U, PU, PL)
DIMENSION S1(2), XG(2,N), U(0:2,N), PU(0:2,0:2,N), PL(2,0:2,N)

This subroutine evaluates the second derivatives of Φ at N points in one 2-D domain, and returns the results in the arrays PU and PL.

Input arguments:

N = number of points where the second derivatives of Φ are to be evaluated

IX2 = index value of the 2-D domain containing the points

S1 = the components of the parameter σ_1

XG(1,K), XG(2,K) = global coordinates x,y of the point K (K = 1,...,N)

U(0,K) = the solution value u at the point K

U(1,K) = the value of the derivative $u_x = \partial u/\partial x$ at the point K

U(2,K) = the value of the derivative $u_y = \partial u/\partial y$ at the point K

Output arguments:

PU(I,J,K) = $\partial^2\Phi/\partial U(I,K)\,\partial U(J,K)$, (I,J=0,1,2; K=1,...N)

PL(I,J,K) = $\partial^2\Phi/\partial S1(I)\,\partial U(J,K)$, (I=1,2; J=0,1,2; K=1,...,N)

SUBROUTINE USRG1 (N, IX1, S3, XG, CN, G, GL)
DIMENSION S3(2), XG(2,N), CN(2,N), G(N), GL(2,N)

This subroutine calculates the function G_1 defining the Neumann boundary conditions, and its derivatives by σ_3, at N points in one 1-D domain, and returns the results in the arrays G and GL.

Input arguments:

N = number of points where the evaluation is to take place

IX1 = index value of the 1-D domain containing the points

S3 = the components of the parameter σ_3

XG(1,K),XG(2,K)
= global coordinates x,y of the point K (K = 1,...,N)

CN(1,K),CN(2,K)
= components of the normal unit vector at K in the global coordinate system.

Output arguments:

G(K) = the value of G_1 at the point K

GL(J,K) = the derivatives $\partial G(K)/\partial S3(J)$, J=1,2 of G_1 by σ_3 at the point K.

SUBROUTINE USRG2 (N, IX2, S2, XG, G, GL)
DIMENSION S2(2), XG(2,N), G(N), GL(2,N)

This subroutine evaluates the value of the function G_2 and its derivatives by σ_2 at N points in one 2-D domain, and returns the results in the arrays G and GL.

Input arguments:

N = number of points where the evaluation is to take place

IX2 = index value of the 2-D domain containing the points

S2 = the components of the parameter σ_2

XG(1,K),XG(2,K)
= global coordinates x,y of the point K (K = 1,...,N)

Output arguments:

G(K) = the value of G_2 at the point K

GL(J,K) = the derivatives $\partial G(K)/\partial S2(J)$, J=1,2 of G_2 by σ_2 at the point K.

While the above subroutines must be supplied by the user for any problems, two other subroutines are used in the program file REGION under the tag R.14. These subroutines specify the number of data (RNODI) and the actual data (RNODD) to be written out as added node data for a region file. Default subroutines for this are included in NFEARS. Any change of these routine requires a knowledge of the NFEARS data structure.

II.4. Running NFEARS

NFEARS works interactively in response to "commands" given by the user during a run. The command names, shown in the command flow-chart in Figure II.4.1, can be typed in lower or upper case. Once a command is given, the program may prompt for further input, specific to that command. After a command has been successfully executed, NFEARS prints out the execution-time and then prompts for a new command input. Initially, the program asks whether all input should be echoed back or not. In the case of a batch run, this indicator should always be set to one to ensure such an echo; there will be no echo if the indicator equals zero.

The output produced by NFEARS appears in two places, namely, (1) at the terminal, and (2) in the log-file with unit number 10. The amount of output can be controlled by the user. The prompt for a command-input, the time spent for the execution of a command, and any potential error messages will always show up at the terminal. The log-file (Fortran unit 10) will be created by NFEARS as a new, sequential, formatted file. Thereafter, output strings are written onto it, and an end-of-file termination occurs when the QUIT command is given. It is the user's responsibility to print or discard this file after termination of the NFEARS run. This file provides a record of the NFEARS session and may contain a large amount of data which could not be handled conveniently on the terminal's screen. The amount of output can be set and modified by the "TRACE" command.

In this Section we summarize briefly the essential aspects of these commands and refer to Section II.5 for more detailed descriptions of each of them.

Command Flowchart:

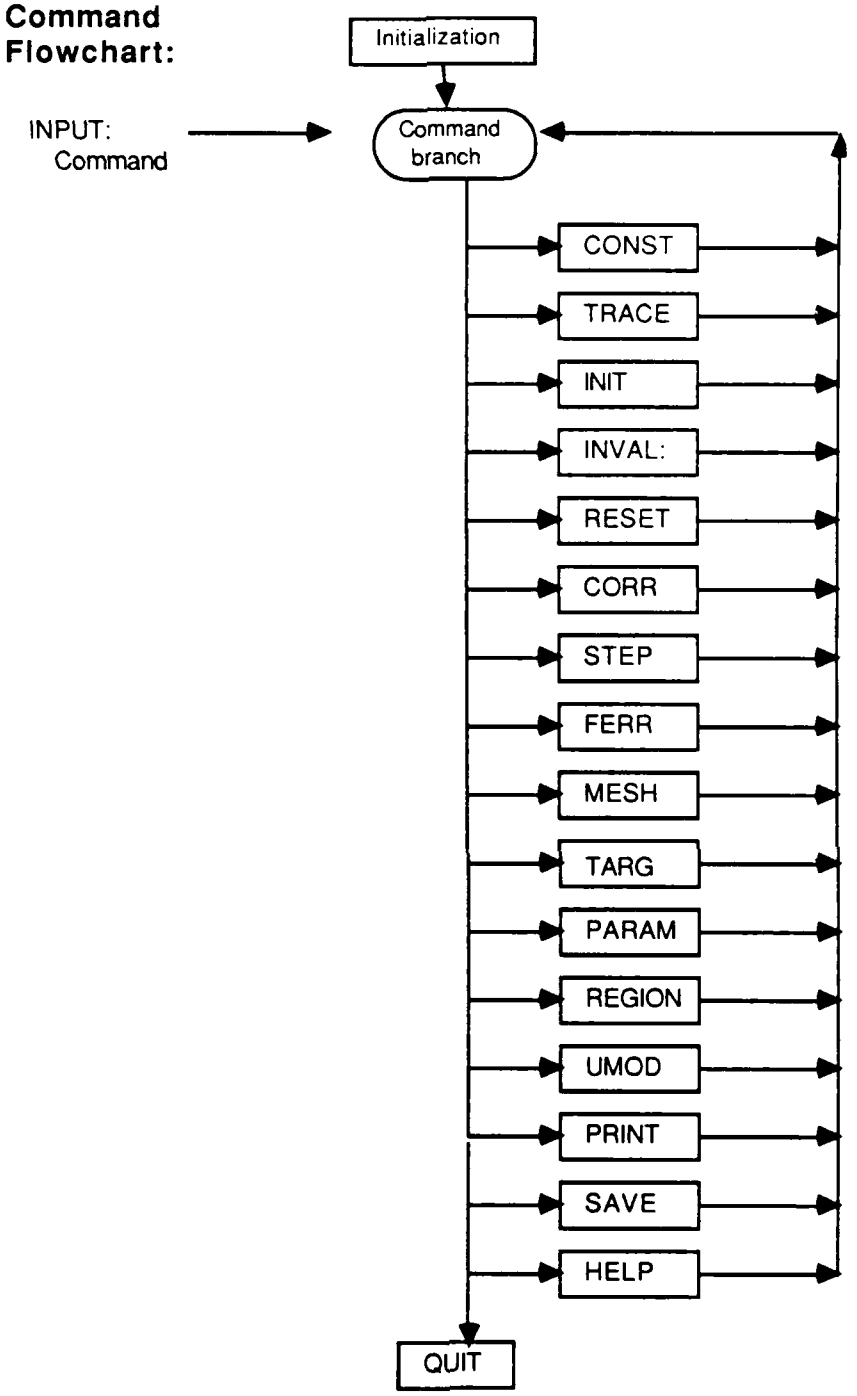


Figure II.4.1

CONST	to set certain constants for the continuation algorithm and for the mesh modifications, and to re-define the mesh intensity $\mathfrak{Z}$,
INIT	to initiaize a problem,
INVAL	to change existing values of the initial solution,
HELP	to print out all command names as well as the last command,
TRACE	to specify the amount of output,
RESET	to reset NFEARS from a previously saved file,
CORR	to correct the initial solution with the corrector process of the continuation algorithm,
STEP	to step along the solution path with the continuation algorithm,
FERR	to calculate estimates of the discretization errors of a solution and to determine the corresponding ideal density function,
MESH	to modify the mesh,
TARG	to request a target and/or limit point calculation,
UMOD	to call the user supplied subroutine USRMOD,
PARAM	to modify certain parameters,
PRINT	to print out solutions, errors, meshes, etc,
SAVE	to save present data on a file,
REGION	to calculate an approximation of some region of the solution manifold,
QUIT	to terminate the current run with the program.

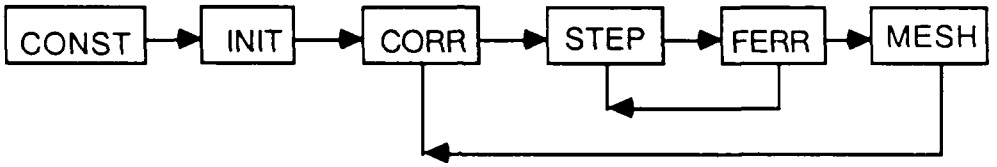
Order of Commands:

Although NFEARS will, in general, execute commands as they are given, there are certain logical restrictions which must be observed in the sequence of the commands. First of all, one has to establish the necessary constants for the corrector iteration by means of the CONST command if the default values are deemed to be inappropriate. In fact, it is always advisable to start with this command since it prints out the values of these constants before asking for any changes and hence provides a printed record of them in the log-file. After this, one can either initialize a new problem with INIT or continue with a previously saved problem by commanding a RESET. The INVAL command may be given at this point if a different initial solution is needed. In any case, this initialization is typically followed by a CORR command. The HELP, TRACE and QUIT commands can be used at any time, but obviously with QUIT the run will

terminate. The position of the UMOD command depends on the user supplied subroutine USRMOD. Typically, this routine is used to modify some of the parameters needed for the evaluation of certain functionals intrinsic to the problem. In that case, it should be followed by a CORR command to correct the last solution obtained by NFEARS.

Figure II.4.2 shows two frequently used command sequences, the first for single parameter continuation, the second for switching over to a calculation of a simplicial approximation of a (two-dimensional) region of the manifold.

Normal order for single parameter continuation:



Switching to two parameter Region calculation:

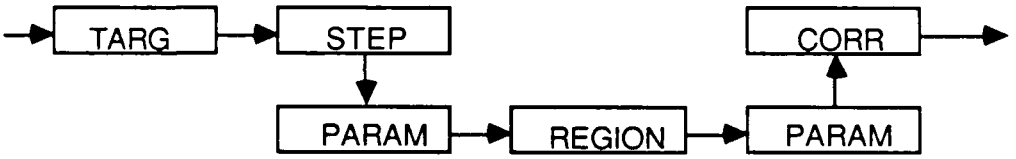


Figure 4.2

The basic data structures of NFEARS consist of several summary records and the tree-structures described in Section I.4. With the tree-structure, storage is provided for one set of solution data and the corresponding error indicators and densities. In addition, a temporary data structure is available during the corrector process for two sets of solution data used in the continuation algorithm and in the region calculation. This temporary data structure consists of the assembly files (units 15 and 16) and the vector and matrix arrays which are partly in files 14 and 18.; it is set up by the CORR command. More specifically, this command opens the assembly files, copies the solution currently in the tree data structure to the "predictor"-location of the temporary data structure, calculates its Jacobians and the tangent vector of the continuation path, and finally starts the corrector iteration to improve that solution. The CORR and STEP commands always leave the calculated solutions in the "current"-location

of the temporary storage. After "STEP" there may be two such solutions, namely, the "current" solution and, if applicable, the solution obtained at a target or limit point which is then contained in the "predictor"-location. Hence, unless the user wants to continue with another STEP command, one of these solutions should be transfered back to the tree storage structure. This is accomplished by the FERR command before its calculation of the discretization errors and of the ideal density function, and, accordingly, this command asks first which one of the two solutions in the temporary data structure is to be transfered.

The PRINT, MESH and SAVE commands always assume that the solution is in the tree data structure. In particular, the temporary data structure is not saved by the SAVE command; thus after a RESET command, the user should use a CORR command to re-establish it.

II.5. NFEARS Commands

In this section we discuss the individual NFEARS commands in detail.

5.1 The CONST Command:

The CONST command permits a change of certain control parameters for the continuation algorithm (CORR, STEP commands), the region calculation (REGION command) and the mesh modification (MESH command). When this command is invoked, the user is asked to opt either for the constants of group 1 used in CORR, STEP, REGION, or for those of group 2 needed in MESH, and allowed to change there values. Initially NFEARS sets these to some default values. The various constants, and, in parentheses, their default values are as follows:

(a) Group 1: Constants for the CORR, STEP and REGION commands:

Maximum number of steps allowed per STEP call (5)
Starting step size (0.01)
Maximum step size (1.0)
Minimum step size (0.0001)
Maximum number of steps in the corrector iteration (10)
Frequency of Jacobian evaluation (3)
Absolute error tolerance for the corrector iteration (10*C)
Relative error tolerance for corrector iteration (10*C)
Minimum pivot value allowed in matrix decomposition (C)

where C is the smallest number on the computer such that 1.0+C is different from 1.0.

(b) Group 2: Constants for the MESH command:

Mesh-modification mode: "manual" or "automatic" (automatic)
Control of the "automatic" mesh-modification: "by error size" or "by density" (by density)
Tolerances for automatic mesh-modification by "error size":
Refinement tolerance: If the error indicator of an element exceeds this tolerance then the element is subdivided. De-refinement tolerance: If ω is an element of a previous mesh for which all four

sons are elements of the current mesh and their combined error indicators fall below this tolerances, then ω is de-refined.

Intensity, initially set by user's input.

For both groups of constants, the program prints out the presently set values, and then asks if any of them should be changed.

5.2 The TRACE Command:

This command sets the Indicator for the amount of output from NFEARS and may be invoked at any time. It prompts for two, free-formatted input lines:

(i) The first line consists of two integer values:

ECHO , STATUS

where

ECHO = 1 if all inputs are to be echoed to the system ouput file 6,

= 0 if the inputs are not to be reprinted.

STATUS ≥ 0 indicator of the amount of print-out to the system ouput file. A zero value keeps it to a minimum, and for increasing positive values of STATUS the amount of output is increased.

(ii) The second line consists of six non-negative integers corresponding to 6 specific parts of NFEARS:

I1 , I2 , I3 , I4 , I5 , I6

These 6 parts are identified as follows:

I1: Execution of the INIT and RESET commands

I2: Corrector iteration

I3: Execution of the MESH command

I4: Execution of the CORR and STEP commands

I5: Execution of the FERR command

I6: Execution of the REGION Command

The integer value I_k ($1 \leq k \leq 6$) specifies the amount of output from Part k that is to be generated in the log-file on unit 10, and, when STATUS $\geq I_k$, also on the system output file 6.

5.3 The INIT Command:

This command initializes a new problem. NFEARS first calls the user-subroutine USRFCT with the first argument set to zero. This call allows for any set-up of parameter values that may needed in other user-subroutine and also for the print-out of a problem title.

Upon return from USRFCT, the program asks for the name of the file containing the geometry data. If the answer is "5", then these geometry data are to be given interactively. Otherwise, the answer is assumed to be the file-name where the geometry data reside with the format descibed in Section II.2.

After the geometry input, the program asks for the intensity $\Im$ of the initial mesh. This value can be changed again by the CONST command.

Thereafter, the program asks for the coefficients of the effective parameters. First, the coefficients δ_1 and δ_2 with $\delta_1+\delta_2 = 1.0$ of the linear functions $\lambda_1 = \delta_1\lambda$, $\lambda_2 = \delta_2\lambda$ are requested which specify the continuation-path (see Section I.5). Then the eight coefficients α and β are expected that define λ_1 and λ_2 in terms of the problem parameters (see Section I.1).

In summary, the input for the INIT command is as follows:

1. Input from USRFCT;
2. name of the geometry file or "5" for interactive input;
3. geometry data if the input is to be interactive;
4. intensity $\Im$
5. δ_1 , δ_2
6. k, α_k^1 , β_k^1 , k=1,2,3,4

   k, α_k^2 , β_k^2 , k=1,2,3

After the INIT command, the user should apply a CORR command to establish the initial solution in the temporary data structure and to compute its Jacobian

and the tangent of the specified path. If the initial solution is not adequate, the INVAL command may be used prior to CORR.

5.4 The INVAL Command:

This command is used to supply an initial solution through the USRINV subroutine. When this command is invoked, USRINV is called first with the input variable N set to zero to allow for any initialization that may be needed in this subroutine. Thereafter, USRINV is called repeatedly with N>0 to obtain initial values U at N nodes with global coordinates (x,y). Note that this command cannot be used before the geometry is established by the INIT or RESET commands. Moreover, the INVAL command should always be followed by the CORR command to correct the supplied solution and to establish it and its related data in the temporary storage area.

5.5 The RESET Command:

The RESET command causes the data structure from a previously saved disk-file, generated by a SAVE command, to be read back into the program. The user must supply the name of the file. After the file is read, NFEARS calls USRFCT with the first argument set to one to allow for a print-out of a problem title and, if desired, of any saved parameters in the common block /USRPAR/.

This command should be followed by a CORR command to establish the solution and its related data in the temporary data structure and to ensure its correctness. Once again, the INVAL command may be applied prior to CORR.

5.6 The CORR Command:

This command establishes the current solution and its Jacobian in the temporary data structure; it then, applies the corrector process to it , and calculates the tangent vector of the specified path in preparation for the continuation algorithm. This command should be used after any INIT or RESET command, and also after the calculation of an approximation of a region of the manifold by a REGION command (see Figure II.4.2).

5.7 The STEP Command:

The STEP command invokes the continuation algorithm and prompts the user for the number of steps that are to be taken. It terminates in either one of the following three modes:

(a) The specified number of steps have been taken. The "current" location contains the "point" on the path reached at the last step.

(b) During the step-calculation, a previously called-for target or limit point is detected between two successively computed points on the path. The solution corresponding to this target or limit point is returned in the "predictor" location of the temporary data structure while the computed point on the path just beyond it is in the "current" location.

((c) The corrector iteration failed to converge.

A print-out informs the user which of these modes applies, and, in particular, whether a target or limit point has been found. The STEP command is usually followed by a FERR command. It is advisable not to take too many steps with one STEP command, since error indicators are not calculated during the continuation and hence it may happen that the discretization errors become undesirably large. When the corrector iteration fails, the user may try to decrease the minimum step size of the continuation algorithm by means of the CONST command. Another remedy might be to establish a more suitable scaling of all variables and, especially, of the parameters λ_1 and λ_2; but, of course, this requires some modifications in the user-subroutines.

5.8 The FERR Command:

This command calculates the error indicators for a solution obtained by a CORR or STEP command and determines the ideal density. More specifically, the error estimates are computed for the solution in the current location, unless a target or limit point has been found which, of course, is then contained in the "predicted" location. In this case the user is asked whether the error calculation should be performed for the values in the "predicted" or the "current" location. The appropriate solution is then transfered to the tree storage area and the error and density calculation is performed. This command should be given before

any MESH, PRINT or SAVE commands since all of these commands work with the solution and its error indicators in the tree storage area.

5.9 The MESH Command:

This command checks the present meshes on the 2-D domains and modifies them in accordance with the mesh-modification-mode set by the CONST command. The program tests first if any de-refinement is needed; that is, whether any four elements obtained by a prior refinement of an element are to be contracted again into one element. Once all de-refinements, if any, are completed, the program checks if any refinement is to be performed; that is, whether any element is to be subdivided into four elements.

Modification is performed in accordance with the mode set by the CONST command. "Automatic" refinement decision can be made on the basis of the error-sizes or of the density and intensity. The refinement by error-size uses two error tolerances supplied by the CONST command and for details of the refinement by density/intensity we refer to Section I.8. "Manual" modification is performed interactively. The user is asked for each element, which is a candidate for de-refinement or refinement, whether the particular operation should be performed or not. In all cases, the total number of de-refinements and refinements is provided as output.

Interpolation is used to obtain the solution values for any nodes that may have been newly established by any refinement. The resulting approximate solution should always be corrected by a CORR or STEP command.

5.10 The TARG Command:

This command prints out any presently set target and/or limit point indicators, if there are any, and then asks for new indicators if these are to be established. A target or limit point indicator always consists of the index value of the variable or parameter variable, and, in addition -- for a target point -- of the desired target value. The following variable indices are allowed:

A 0-D domain not carrying a Dirichlet boundary condition,
the mid-point of a 1-D domain not carrying a Dirichlet boundary condition,
the mid-point of a 2-D -domain,

any one of the active λ or σ parameter variables.

5.11 The PARAM Command:

This command prints out the present values of the effective parameter values λ_1, λ_2 and of the coefficients δ_1, δ_2, α and β. It then asks if the δ and β values are to be changed. The following answers are allowed:

0 , 0	(two zeros) no change,
δ_1 , δ_2	($\delta_1+\delta_2 = 1.0$) change the previous δ_1 and δ_2 values,
1 , β	reset the value of β_i^k for the printed i,k indices.

In either case, new values of the α_i^k are determined which ensure that the current values of the program parameters σ_i^k are not changed, and then the values of the effective parameters λ_1,λ_2 are set to zero. More specifically -- after all changes are provided -- the parameter values will be as follows

$$\alpha_i^k \text{ (new)} = \alpha_i^k \text{ (old)} + \beta_i^k \text{ (old) } \lambda_k \text{ (old)} \quad , \quad i=1,...,4; \ k=1,2$$

$$\lambda \text{ (new)} = \lambda_k \text{(new)} = 0, \quad \beta_i^k \text{ (new)} = \text{input provided by the user.}$$

In addition, any previously set target or limit point indicators are erased. Thus , if desired, such target and limit point indicators have to be reset in terms of the new λ-values initialized to 0.

5.12 The REGION Command:

This command invokes the algorithm for the calculation of a simplicial approximation of an open region of the manifold, and enters into a sub-command mode which is described in the next section. Before invoking this command, the user should ensure that both effective parameter variables, λ_1 and λ_2 are active. This can be guaranteed with the PARAM command by providing that at least one β_i^1 and at least one β_i^2 are non-zero. Accordingly, upon exit from the

REGION sub-command mode, the user may wish to invoke the PARAM command to choose a new relation between the two parameter variables.

5.13 The UMOD Command:

This command invokes a call to the user-subroutine USRMOD. Accordingly, any action taken depends on this routine.

5.14 The PRINT Command:

This command prints out the current solution in the tree-storage area and its associated error and density values. It should be used after a FERR command, otherwise the program may print a previous result. When this command is given, the program asks whether output should be on-line (system output 6). If the answer is "No", then the full solution, error and density values will be printed in the log-file (unit 10). If the answer is "Yes" (on-line), then the program will prompt further for a specification of the desired segments of the solution, error and density data which are to be printed out for each subdomain.

5.15 The SAVE Command:

This command saves the present data on a disk-file which can be used later to resume the computation by means of a RESET command. The program asks for the name of the file (which should not exist) to which the output is to be directed. The program saves all problem data including the parameter values in the user-subroutines. The data in the temporary storage area; that is, in particular, the Jacobian and the path-tangents, are not saved. Accordingly, this command should be given after a FERR or MESH command. It is the user's responsibility to save this disk file permanently after the termination of an NFEARS session.

5.16 The HELP Command:

This command prints out the names of the commands and the name of the last executed command.

5.17 The QUIT Command:

This command terminates the NFEARS session and prints out the number of region-output files (see Section II.6) that have been generated. These files and any files generated by a SAVE command should be saved by the user, and , in addition, the log-file (unit 10) should be printed out.

II.6. REGION Subcommands

The REGION command calculates a simplicial approximation on the manifold M in an neighborhood of the "current solution" called, in this context, the reference point. If a presently available target point is to be used as the reference point, then it has to be placed into the "current" location by issuing a FERR and CORR command prior to the REGION command. The Region sub-program is implemented as an interactive routine which is controlled by user supplied sub-commands each of which consists of a single character. When the REGION command is invoked, the program checks whether both effective parameters are active, if not thena request for a change of the parameter dependence is issued. When this condition is satisfied, the program resets the values of the two effective parameters λ_1 and λ_2 to zero.

The basic principle of the region calculation is discussed in Section I.7. As noted there a Kuhn triangulation in R^2 is used as the reference triangulation and we work with rectangular patches of eight triangles each containing one center node and eight boundary nodes. As outlined in Section I.7, the algorithm begins by mapping a first patch onto the tangent plane of the reference point and by projecting the nodes from there onto the manifold. Then, under user control, the program proceeds to transfer an adjacent patch, next to the first one, onto the manifold, etc. In other words, the REGION command maps a sequence of "patches" onto the manifold, and their connectivity pattern defines the desired simplicial approximation of the particular region.

The output from the REGION command consists of a region-file which has as its first record a connectivity matrix NODE(i,j), i,j=1,...,MAXNOD (=21), followed by a sequence of records containing the attributes of the calculated nodal points (solution values, error indicators, etc.). The entries of the connectivity matrix correspond to the nodes of a 20 by 20 subset of triangles of the reference triangulation (see Figure I.7.1); the triangular connections are not stored explicitly. Originally, this matrix is set to zero, except for the entry NODE(10,10) which corresponds to the node that is mapped into the reference point and is set to 1. When a node of any patch has been mapped successfully onto M and its attributes have been calculated, then the node receives a positive index which is recorded in the matrix. At the same time, the attribute record of that

point is saved. A negative index is used in the connectivity matrix when an attempt has been made to transfer the point to M but the corrector iteration has failed. In the matrix, the center-points of the patches correspond to the "even" entries, NODE(2i,2j) ($1 \le i,j \le 10$), and the boundary points of the patch are the eight adjacent matrix entries. Thus a patch is identified by nine entries NODE(2i+m,2j+n), m,n=-1,0,+1. The adjacent patches, of course, share the boundary nodes.

After the initialization, the program enters into the sub-command mode. The order in which patches are to be transfered onto M is controlled by the sub-commands L(eft), R(ight),D(own) and U(p), i.e. the patch to be used next is the one adjacent to the "current" patch in the specified direction. Figure II.6.2 shows the sequence of patches corresponding to the commands L,D,R,R,R. The "current" patch is usually the last calculated one. But, the program also provides nine temporary locations, indexed 1 to 9, where the center points on M of a successfully mapped patch can be saved by means of the sub-command S(ave). The reference point is initially saved in location 1. During the calculation another sub-command allows for a previously "saved patch" to replace the "current" one. When the region computation terminates, control is returned to the main program with a "current" solution. At that time, the user also has the choice which of these saved center points should become the "current" solution.

Generation of patches:

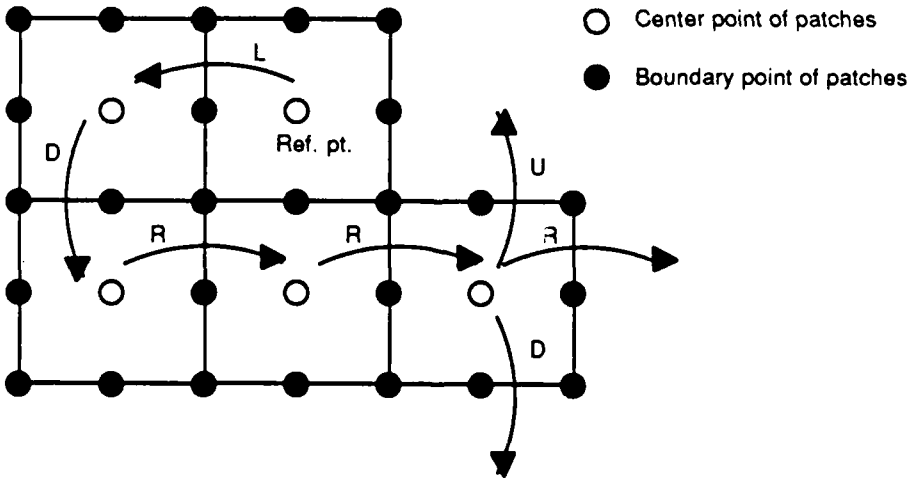


Figure II.6.2

The computation of the center-point needed for mapping a new patch onto M and the process of transfering a node from R^2 onto the tangent plane and of projecting it from there onto M was discussed in Section I.7. Default step sizes are based on the last stepsize of the continuation algorithm and can be changed by the user. For the center- point calculation the modified Newton process is the same as in the continuation algorithm, but the program always re-evaluates the Jacobian after the solution has been obtained. The calculation of the boundary points of the new patch uses a chord Newton method based on the Jacobian at the center point of the patch. A flow chart of the region-subprogram is given in Figure II.6.3.

A character matrix picture of the calculated patches is printed at the terminal. The entries in this matrix correspond to the center points of the patches. and consist of a character followed either by a blank or a single digit between 1 and 9. Patches which were not yet mapped onto M are represented by a period followed by blank. Instead of the full 10 by 10 array of patches, only the used patches are shown with one row and/or column of periods on the four sides where applicable. The characters representing these patches are as follows:

"r" or "R"	=	Initial reference patch
"a" or "A"	=	Successfully transfered patch for which all 9 nodes have been mapped onto M.
"b" or "B"	=	Unsuccessfully used patch for which the corrector diverged at one or more nodes

The "current" patch is indicated by a capital letter and all others by small letters. A digit between 1 and 9 following the patch character indicates that the center-point for that patch has been saved in the temporary location with the same index.

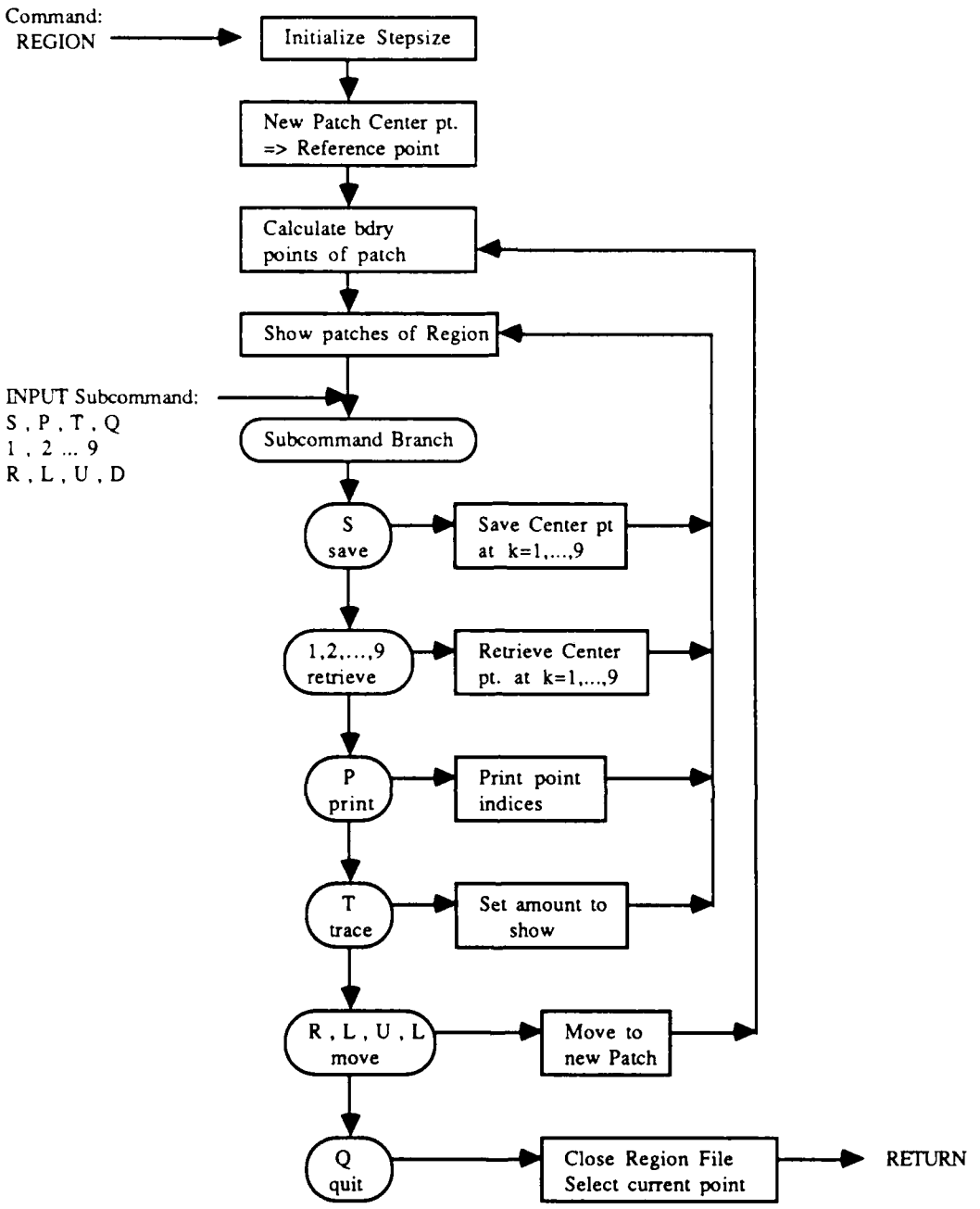


Figure II.6.3

Example:

Initial ouput picture:

```
.     .     .
.     R1    .
.     .     .
```

The current patch is the reference patch; its center point is the reference point which is always saved in location 1. The next patch may be chosen in any one of the four directions (Left, Right, Up, Down). Suppose that at a later time the picture looks as follows:

```
.     b     a     a     A     .
.     a     a     a     .     .
.     a     a     a     .     .
.     a2    a     a     .     .
.     a     r1    a     .     .
.     .     .     .     .     .
```

Only two movements are possible from the current patch "A", namely -- as indicated by the missing period in the top line -- the R(ight) and the D(own) movement . The center points of two patches were saved in locations 1 and 2. All patches were mapped successfully onto M, except the one in the upper left corner.

Below the picture, the program prints out a line of summary data, including the number of patches used so far, number of nodes transfered onto M, etc, and, optionally, three 3 by 3 matrices which give some information about the current patch calculation. For details we refer to the P(rint) sub-command below. These matrices can also be printed automatically with a proper setting of the T(race) sub-command.

After printing the picture the program expects another one-character (sub)command from the user. Following is a list of all these subcommands:

"S" = Save the current patch: The center-point of the current patch is saved in the temporary location with the next available index. If all 9 locations have been used then the user will be asked which one of them is to be replaced. The index of all patches for which the center points have been saved can be seen in the next picture.

"1", "2", ..., "9"
= Use the "saved patch" in this location as the "current patch"; an error message results if the indicated location is empty.

"P" = Prints three 3 by 3 matrices are printed side by side which give information about the last patch calculation. The first matrix contains the indices of the nodes that were mapped onto M, the second one shows the number of corrector steps for their calculation , and the third matrix contains the distance, in the maximum norm, from the predicted point on the tangent space to the computed node on M.
When the Print command is given, the program asks whether the above output should be directed to the terminal or to file 10. It may be noted that the T(race) command also allows for the automatic generation of the same output, except that then it will always be directed to the terminal.

"L", "R", "U" or "D"
= Calculates a new patch adjacent to the current patch by moving in the indicated direction L(eft), R(ight), U(p) or D(own). If the patch in that position has already been mapped onto M, or if it is outside of the 10 by 10 patch region, an error message is issued. Once the new patch has been transfered onto M, it will become the new current patch.

"T" = Trace command provides for certain optional outputs. It requires two integers as inputs, which are the same as the trace switches of the main program for the amount of terminal output. A 1,1 input produces an automatic output of the same three matrices on the terminal as the P(rint) command, while 0,0 suppresses this print-out.

"Q" = Quit command: It establishes a region-file before returning control to the main program. The user has a choice which of the saved center-points of the patches is to become the "current" solution. This solution should be "corrected" by a CORR command.

Region-file

A region-file is a sequentiai, un-formatted file with unit-number 30+i where i=1 at the first call of the REGION command, and, thereafter, i is incremented for each subsequent call of the REGION program. Up to 9 region files may be established. The first record of a region-file is the region-record containing the connectivity matrix NODE, all subsequent ones are the point attribute records.

Region-record:

NDTOT,NR,NPTOT,MXN,NODE

where

NDTOT = number of nodes mapped onto M = number of subsequent records,

NR = number of data in the record of each point

NPTOT = number of patches mapped onto M

MXN = dimension of the NODE matrix (=21)

NODE(MXN,MXN)

= connectivity matrix with entries i as follows

i > 0 index number of the point

i = 0 point not calculated

i < 0 iteration failed for the point

Point attribute records (length NR):

FERROR = the error estimator at the point

ENERGY = linear energy term

ERRMAX = maximum elemental error indicator

CINTY = current intensity

DINTY = ideal intensity

DNRM = ideal density norm

VECTP(1),VECTP(2)
= values of the effective parameters λ_1 and λ_2 (recall that these are relative to the reference point)

VECTP(3) = single effective parameter λ value (this has no real meaning in this context)

CPAR = values of the problem parameters σ_j^i (i=1,2; j=1,2,3,4), note that $\sigma_4^2 = 0$ is included which is not used

V(i), i=1,...,NR-18 (= N_0)
= These are optional output data obtained through a subroutine RNODD which also has an entry RNODI(n) that returns the number of data values "n". At present, NFEARS contains a default subroutine R.14 which gives n = N_0 data values at the 0-D domains, (including those carrying Dirichlet conditions). With the 18 previous data values, we have NR = 18+N_0.

END
DATE
FILMED
3-88
DTIC