1976

Note on the Software Design for an Elliptic PDE Solver

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
76-193

NOTE ON THE SOFTWARE DESIGN FOR AN ELLIPTIC PDE SOLVER

John R. Rice
Mathematical Sciences
Purdue University

CSD-TR 193
May 15, 1976

ABSTRACT

There notes present a high level description of the structure and operation of a software package for solving linear elliptic partial differential equations on a general domain in two dimensions. The package is to be modularized and structured so that various experimental changes and additions can be implemented with ease. The three main modules described here are User Interface (which is to be implemented near the end), Region Discretization and Operator Approximation. The modules not discussed here are Linear Equation Solution and Output.

ELLIPTIC PDE SOLVER - ULTIMATE USER INTERFACE

We assume a preprocessor that translates the user's specifications into a standard form. The preprocessor is to create a Fortran program specifically tailored for the user's particular problem. In the development stage we use the output of this contemplated preprocessor as the input to the later modules in the package. The purpose of this note is to first indicate the nature of the ultimate user input by example and second to specify the output of this preprocessor and define the input of the remaining modules.

1. EXAMPLE PROBLEM SPECIFICATION. This sample problem is a variable coefficients operator on the region

with one coefficient defined by a Fortran subprogram and the forcing term tabulated data. Standard keywords and symbols of the systems are underlined in this example.

```
OPERATOR = UXX + SIN(X)*UYY + FORT(X)*UX = DATA(X,Y)
BOUNDARY = 1.  LINE.  (0,0) TO (0,1)
           2.  LINE.  (0,1) TO (.75,1)
           3.  CURVE.     X = 1-.25*COS(ALPHA)
                          y = 1-.25*SIN(ALPHA)
                          FOR ALPHA = 0 TO PI
```

4. LINE. (1.25,1) TO (2,1)

5. CURVE. Y = X**2/4. FOR X = 2. TO 0.

BOUNDARY CONDITIONS =

1. UX = 0

2. U = 2.-EXP(THETA*X)

3. U = 2-EXP(.75*THETA)

4. U = (1.+EXP(.75*THETA))/.75*X+2.*(1.-(1.+EXP(.75*THETA)/.75

5. U = EDGE(X,Y)

FUNCTIONS =

```
      FUNCTION FORT(X)
      REAL F,THETA,MASS
      SUM = 0.0
      DX = 0.08
      DO 10 K = 2,24
         X = DX*FLOAT(K-1)
10       SUM =   SUM + EXP(-THETA**)
      SUM = 2.*SUM + 1. + EXP(-2.*THETA)
      F = .04*SUM/MASS
      RETURN
      END

      FUNCTION EDGE(X,Y)
      EDGE = 1. + THETA*X + MASS*(1.-Y)
      EDGE = AMAX1(EDGE,1.5)
      RETURN
      END
```

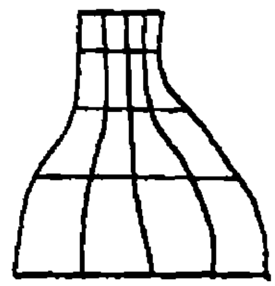
```
DATA = THETA = .0076
       PI = 3.141519265
       MASS = 1.46
     DATA = (TABLE(X,Y),FILE = INPUT2)

METHOD = COLLOCATION

GRID =     X:  0.0,0.25,0.5,0.75,0.9,1.05,1.2,1.35,1.5,1.6,1.7,1.9,2.
           Y:  0.0 TO 1.0 WITH 6 STEPS

OUTPUT = TIME SUMMARY
         MEMORY SUMMARY
         REGION PLOT
         RESIDUALS
         TABLE ON GRID
         CROSS SECTION PLOT:  X=0.,X=.5,X=1.0,X=1.5,Y=.25
```

2. EXAMPLES OF OTHER STATEMENTS NOT INCLUDED ABOVE.

```
METHOD = HODIE WITH 5-POINT STAR AND 7 POINTS
         FINITE DIFFERENCE WITH 9-POINT STAR

BOUNDARY = 1.  CURVE  X = 1 + 2.*COS(T)
                      Y = 1 + 2.*SIN(T)  FOR T=0 TO 3.14159265

           2.  BREAK

           3.  LINE.  (1,1) TO (1,0) TO (0,0) TO (1,1)
               This allows multiply connected regions
```

SPECIAL POINTS = 1. (1.2,1.2) REENTRANT CORNER
2. (.6,.4) SINGULARITY LIKE R**(.5)

This gives special behavior treatment implemented by using a small library of special elements or special difference formulas



VARIABLE COORDINATES =

CHANGEX = X/SHAPE(Y)

CHANGEY = ((Y+1.)**2-1.)/3.

This allows variable spacings in rectangular grids

There will also be provisions for a user to bypass parts of the preprocessor's automatic features and to directly provide the input for the next module in the program.

3. STANDARD OUTPUT IN FORM OF INPUT FOR NEXT MODULE

3.1 Operator. Coefficient function array

SUBROUTINE PDE(X,Y,COEFS)

COEFS(I) are coefficients of UXX,UXY,UYY,UX,UY,U and right side.

Note that the preprocessor could insert actual code as needed and thus avoid computing and using zero coefficients.

3.2 .BOUNDARY

NBOUND = number of boundary pieces

BCOORD(P,X,Y) = X,Y coordinates with boundary parameter value P along the IPIECE piece of the boundary (IPIECE is set in COMMON)

BRANGE(I,2) = First and last values for I-th boundary piece
All boundary segments are given with a clockwise orientation, the first closed curve is the exterior boundary.

3.3 Boundary Conditions

BCTYPE(I) = type of boundary conditions on the I-th piece

BCFUNK(I,X,T,VALU) = function specifying the boundary conditions on the I-th piece (VALU is an array of length 2 for some conditions)

3.4 Method

METHOD = value from COLLOC,HODIE,FDIFF,GALERK,LSTSQS
There are two associated integer variables METH1,METH2

LINSOL = value from DIRCT,ITER. There are associated parameters OMEGA1, OMEGA2

3.5 Grid.

NGRIDX,NGRIDY = number of X and Y grid lines

XGRID(I),YGRID(I) = coordinates of the grid lines

3.6 Special Points

NUMBSP = number of special points

SPTYPE(I) = Type of special points (values not yet determined)

SPX(I),SPY(I) = Coordinates of special points, constrained to be compatible with grid.

3.7 Change of Variables.

CHANGE = Logical switch to indicate change of variable

CHANGX(X,Y) = Function which specifies change in X-variable

CHANGY(X,Y) = Function which specifies change in Y-varible

3.8 Output Keywords Switches (Tentative List)

OUTTIM = Timing output

OUTRES,OUTSUL,OUTERR = Output of residuals, solution and error on the grid

OUTXP = Output of residual, solutions and error at extra points. NUMBXP = number of points XXP(I),YXP(I))

PLTREG = Region plot

PLTCS = Cross section plots of solution
NPLTCS = number along lines ax+by+c=0 with arrays CROSSA(I),CROSSB(I),CROSSC(I)

3.9 Development Input and Switches (Tentative List)

LEVEL = switch to control printed output

-1 no output
0 minimal printed output
1 summary of solution obtained
2 more meaningful DEBUG output
3 All DEBUG output

TRUE(X,Y) = true solution

ELLIPTIC PDE SOLVER - REGION DISCRETIZATION MODULE

1. OUTPUT

We first describe the output in two parts.

1.1 Grid Definition.

This is a set of variables with two indexes (suppressed below) with range 1 to NGRIDX,1 to NGRIDY

XGRID,YGRID = coordinates of the grid lines (this is also input to the module)

GTYPE = integer type indicator with certain information encoded as follows:

EXT = -2 = exterior point

INT = -1 = interior point

Index of boundary point if this grid point is on the boundary

Index of boundary point + K*1000 + J*10,000 when J = 1,2,3,4 if 1st clockwise boundary point is at noon, 3:00, 6:00 or 9:00

K = number of adjacent boundary points (clockwise)

1.2 Boundary Definitions:

This is a set of variables indexed from 1 to NBNDPT (supposed below)

XBOUNDMYBOUND = coordinates of boundary point

IPIECE = index of piece to which the point belongs (smallest number for corner points)

BPTYPE = type indicator = HORZ,VER7,BOTH,INTOR the type is negative if the point is a corner.

BNEIGH = J + 1000*K where J = X-index of neighboring
grid point, K = Y-index of neighboring
point (1st clockwise for BOTH case)

DISTBG = Distance from boundary to grid point
Neighbor.

2. MODULE STRUCTURE

```
Locate first boundary point within the grid
For K = 1 to NBOUND do
Compute Boundary-Grid Intersections
        |
        v
Check that the boundary closes
at all corners and at the end
        |
        v
For K = 1 to NBNDPT do
    Mark adjacent grid points as EXT, INT or compute their
        GTYPE value
    Check for "almost boundary points" and adjust if needed
    Determine BTYPE, BNEIGH and DISTBG for all boundary points
        |
        v
Mark all other grid points
as EXT or INT
```

ELLIPTIC PDE SOLVER - OPERATOR APPROXIMATION MODULE

1. OUTPUT. The output of this module is a particular representation of the sparse matrix of the linear system derived from the problem. This representation is directly tied to the geometry.

1.1 Matrix Representations. There is a 2-dimensional set of indices (of grid points or elements) and the equations naturally associated with each grid point or element are described as follows (the two indexes are suppressed)

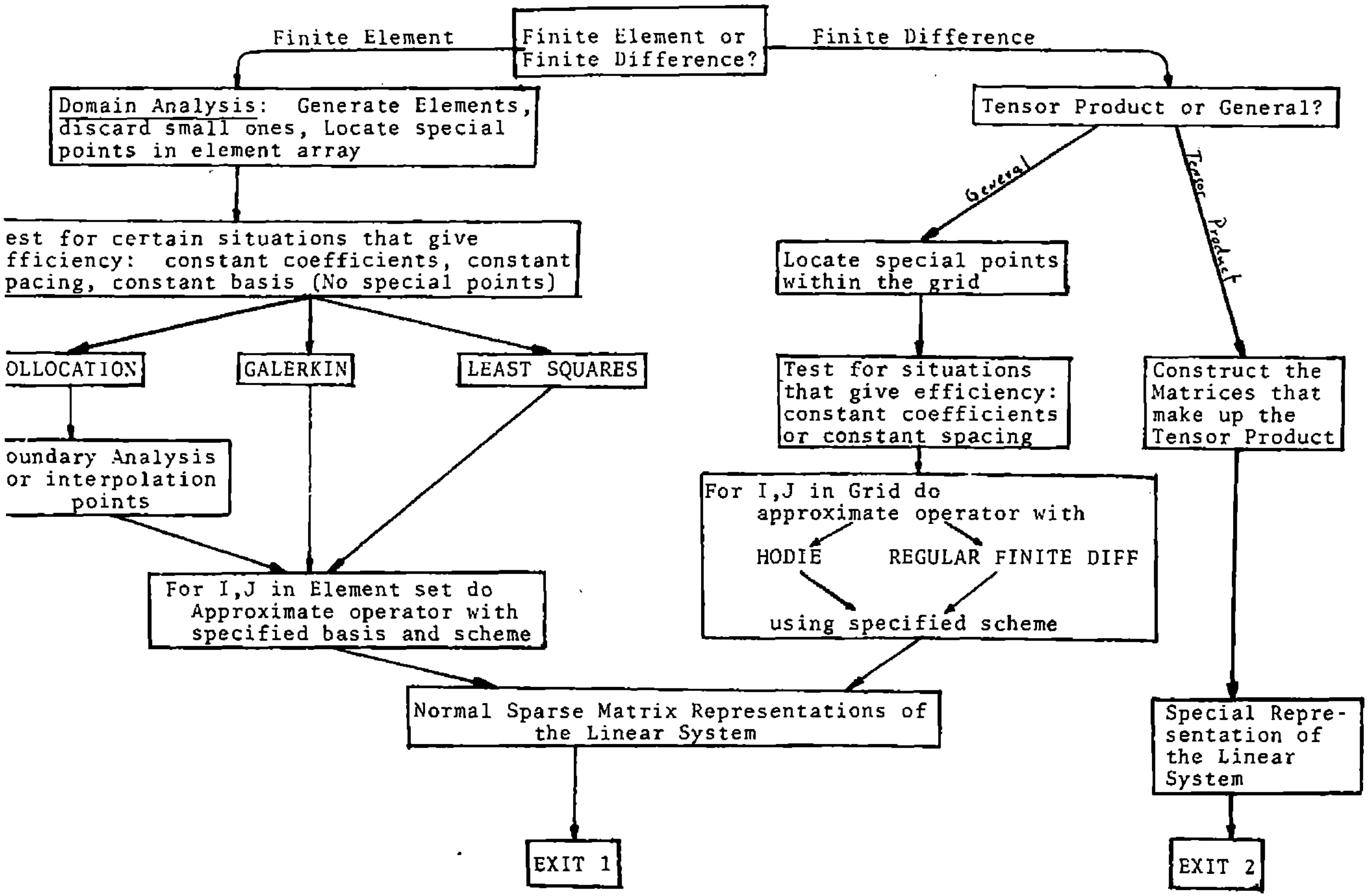
NVAR = number of variables associated with this index pair

NLCOEF = number (max) of linear equation coefficients associated with this index

KX(L),KY(L),KV(L),COELEQ(L),RHS(L)} for L = 1 to NVAR where KX,KY,KV describe the x-index, y-index and variable number of an unknown variable in the L-th equation. COELEQ is the coefficient of this unknown and RHS is the right side of the equation.

Note that (NVAR and NLCOEF) and (KX,KY,KV) can easily be packed into two words if storage is crucial rather than index processing time. Exterior points have NVAR = 0.

2. MODULE STRUCTURE. There is primarily a separate module for each operator approximation scheme. However, a certain amount of duplication can be avoided with the following structure.


Finite Element or Finite Difference?
Finite Element
Finite Difference
Domain Analysis: Generate Elements, discard small ones, Locate special points in element array
est for certain situations that give fficiency: constant coefficients, constant pacing, constant basis (No special points)
OLLOCATION
GALERKIN
LEAST SQUARES
oundary Analysis or interpolation points
For I,J in Element set do Approximate operator with specified basis and scheme
Tensor Product or General?
General
Tensor Product
Locate special points within the grid
Test for situations that give efficiency: constant coefficients or constant spacing
For I,J in Grid do approximate operator with
HODIE
REGULAR FINITE DIFF
using specified scheme
Construct the Matrices that make up the Tensor Product
Normal Sparse Matrix Representations of the Linear System
Special Representation of the Linear System
EXIT 1
EXIT 2