

FIDE
A REDUCE package for automation of FInite
difference method for partial Differential
Equation solving
Version 1.1

Richard Liska
Faculty of Nuclear Science and Physical Engineering
Technical University of Prague
Brehova 7, 115 19 Prague 1, Czechoslovakia
E-mail: tjerl@cspuni12.bitnet (EARN)
Fax: (42 - 2) 84 73 54
Tel: (42 - 2) 84 77 86

May 1991

User's Manual

Abstract

The FIDE package performs automation of the process of numerical solving partial differential equations systems (PDES) by means of computer algebra. For PDES solving finite difference method is applied. The computer algebra system REDUCE and the numerical programming language FORTRAN are used in the presented methodology. The main aim of this methodology is to speed up the process of preparing numerical programs for solving PDES. This process is quite often, especially for complicated systems, a tedious and time consuming task. In the process one can find several stages in which computer algebra can be used for performing routine analytical calculations, namely: transforming differential equations into different coordinate systems, discretization of differential equations, analysis of difference schemes and generation of numerical programs. The FIDE package consists of the following modules:

EXPRES	for transforming PDES into any orthogonal coordinate system.
IIMET	for discretization of PDES by integro-interpolation method.
APPROX	for determining the order of approximation of difference scheme.
CHARPOL	for calculation of amplification matrix and characteristic polynomial of difference scheme, which are needed in Fourier stability analysis.
HURWP	for polynomial roots locating necessary in verifying the von Neumann stability condition.
LINBAND	for generating the block of FORTRAN code, which solves a system of linear algebraic equations with band matrix appearing quite often in difference schemes.

Version 1.1 of the FIDE package is the result of porting FIDE package to REDUCE 3.4. In comparison with Version 1.0 some features has been changed in the LINBAND module (possibility to interface several numerical libraries).

References

[1]R. Liska, L. Drska: FIDE: A REDUCE package for automation of FInite difference method for solving pDE. In ISSAC '90, Proceedings of the International Symposium on Symbolic and Algebraic Computation, Ed. S. Watanabe, M. Nagata. p. 169-176, ACM Press, Addison Wesley, New York 1990.

Contents

1	EXPRESA Module for Transforming Differential Operators and Equations into an Arbitrary	
1.1	The specification of the coordinate system	3
1.2	The declaration of tensor quantities	3
1.3	New infix operators	3
1.4	New prefix operators	4
1.5	Tensor expressions	4
1.6	Assigning statement	5
2	IIMETA Module for Discretizing the Systems of Partial Differential Equations	6
2.1	Specification of the coordinates and the indices corresponding to them	6
2.2	Difference grids	6
2.3	Declaring the dependence of functions on coordinates	7
2.4	Functions and difference grids	8
2.5	Equations and difference grids	10
2.6	Discretization of basic terms	10

2.7	Discretization of a system of equations	16
2.8	Error messages	18
3	APPROXA Module for Determining the Precision Order of the Difference Scheme	18
3.1	Specification of the coordinates and the indices corresponding to them	18
3.2	Specification of the Taylor expansion	19
3.3	Function declaration	19
3.4	Order of accuracy determination	20
4	CHARPOLA Module for Calculating the Amplification Matrix and the Characteristic Polynom	
4.1	Commands common with the IIMET module	21
4.2	Function declaration	21
4.3	Amplification matrix	22
4.4	Characteristic polynomial	23
4.5	Automatic denotation	23
5	HURWPA Module for Polynomial Roots Locating	24
5.1	Conformal mapping	24
5.2	Investigation of polynomial roots	25
6	LINBANDA Module for Generating the Numeric Program for Solving a System of Linear A	
6.1	Program generation	25
6.2	Choosing the numerical library	27
6.3	Completion of the generated code	28

1 EXPRES

A Module for Transforming Differential Operators and Equations into an Arbitrary Orthogonal Coordinate System

This module makes it possible to express various scalar, vector, and tensor differential equations in any orthogonal coordinate system. All transformations needed are executed automatically according to the coordinate system given by the user. The module was implemented according to the similar MACSYMA module from [1].

1.1 The specification of the coordinate system

The coordinate system is specified using the following statement:

```
SCALEFACTORS <d>,<tr 1>,...,<tr d>,<cor 1>,...,<cor d>;
```

$\langle d \rangle ::= 2 \mid 3$	coordinate system dimension
$\langle tr\ i \rangle ::= \text{"algebraic expression"}$	the expression of the i-th Cartesian coordinate in new coordinates
$\langle cor\ i \rangle ::= \text{"identifier"}$	the i-th new coordinate

All evaluated quantities are transformed into the coordinate system set by the last SCALEFACTORS statement. By default, if this statement is not applied, the three-dimensional Cartesian coordinate system is employed. During the evaluation of SCALEFACTORS statement the metric coefficients, i.e. scale factors SF(i), of a defined coordinate system are computed and printed. If the WRCHRI switch is ON, then the nonzero Christoffel symbols of the coordinate system are printed too. By default the WRCHRI switch is OFF.

1.2 The declaration of tensor quantities

Tensor quantities are represented by identifiers. The VECTORS declaration declares the identifiers as vectors, the DYADS declaration declares the identifiers as dyads. i.e. two-dimensional tensors, and the TENSOR declaration declares the identifiers as tensor variables. The declarations have the following syntax:

```

<declaration> <id 1>,<id 2>,...,<id n>;
<declaration> ::= VECTORS | DYADS | TENSOR
<id i> ::= "identifier"

```

The value of the identifier V declared as vector in the two-dimensional coordinate system is (V(1), V(2)), where V(i) are the components of vector V. The value of the identifier T declared as a dyad is ((T(1,1), T(1,2)), (T(2,1), T(2,2))). The value of the tensor variable can be any tensor (see below). Tensor variables can be used only for a single coordinate system, after the coordinate system redefining by a new SCALEFACTORS statement, the tensor variables have to be re-defined using the assigning statement.

1.3 New infix operators

For four different products between the tensor quantities, new infix operators have been introduced (in the explaining examples, a two-dimensional coordinate system, vectors U, V, and dyads T, W are considered):

.	- scalar product	$U.V$	=	$U(1)*V(1)+U(2)*V(2)$
?	- vector product	$U?V$	=	$U(1)*V(2)-U(2)*V(1)$
&	- outer product	$U\&V$	=	$((U(1)*V(1),U(1)*V(2)),$ $(U(2)*V(1),U(2)*V(2)))$
#	- double scalar product	$T\#W$	=	$T(1,1)*W(1,1)+T(1,2)*W(1,2)+$ $T(2,1)*W(2,1)+T(2,2)*W(2,2)$

The other usual arithmetic infix operators +, -, *, ** can be used in all situations that have sense (e.g. vector addition, a multiplication of a tensor by a scalar, etc.).

1.4 New prefix operators

New prefix operators have been introduced to express tensor quantities in its components and the differential operators over the tensor quantities:

- VECT - the explicit expression of a vector in its components
- DYAD - the explicit expression of a dyad in its components
- GRAD - differential operator of gradient
- DIV - differential operator of divergence
- LAPL - Laplace's differential operator
- CURL - differential operator of curl
- DIRDF - differential operator of the derivative in direction (1st argument is the directional vector)

The results of the differential operators are written using the DIFF operator. DIFF(*scalar*,*cor i*) expresses the derivative of *scalar* with respect to the coordinate *cor i*. This operator is not further simplified. If the user wants to make it simpler as common derivatives, he performs the following declaration:

FOR ALL X,Y LET DIFF(X,Y) = DF(X,Y); .

Then, however, we must realize that if the scalars or tensor quantities do not directly explicitly depend on the coordinates, their dependencies have to be declared using the DEPEND statements, otherwise the derivative will be evaluated to zero. The dependence of all vector or dyadic components (as dependence of the name of vector or dyad) has to appear before VECTORS or DYADS declarations, otherwise after these declarations one has to declare the dependencies of all components. For formulating the explicit derivatives of tensor expressions, the differentiation operator DF can be used (e.g. the differentiation of a vector in its components).

1.5 Tensor expressions

Tensor expressions are the input into the EXPRES module and can have a variety of forms. The output is then the formulation of the given tensor expression in the specified coordinate system. The most general form of a tensor expression *tensor* is described as follows (the conditions (d=i) represent the limitation on the dimension of the coordinate system equalling i):

$\langle tensor \rangle ::= \langle scalar \rangle \mid \langle vector \rangle \mid \langle dyad \rangle$
 $\langle scalar \rangle ::=$ "algebraic expression, can contain $\langle scalars \rangle$ " \mid "tensor variable with scalar value" $\mid \langle vector 1 \rangle . \langle vector 2 \rangle \mid \langle dyad 1 \rangle \# \langle dyad 2 \rangle \mid (d=2) \langle vector 1 \rangle ? \langle vector 2 \rangle \mid \text{DIV} \langle vector \rangle \mid \text{LAPL} \langle scalar \rangle \mid (d=2) \text{ROT} \langle vector \rangle \mid \text{DIRDF}(\langle vector \rangle, \langle scalar \rangle)$
 $\langle vector \rangle ::=$ "identifier declared by VECTORS statement" \mid "tensor variable with vector value" $\mid \text{VECT}(\langle scalar 1 \rangle, \dots, \langle scalar d \rangle) \mid -\langle vector \rangle \mid \langle vector 1 \rangle + \langle vector 2 \rangle \mid \langle vector 1 \rangle - \langle vector 2 \rangle \mid \langle scalar \rangle * \langle vector \rangle \mid \langle vector \rangle / \langle scalar \rangle \mid \langle dyad \rangle . \langle vector \rangle \mid \langle vector \rangle . \langle dyad \rangle \mid (d=3) \langle vector 1 \rangle ? \langle vector 2 \rangle \mid (d=2) \langle vector \rangle ? \langle dyad \rangle \mid (d=2) \langle dyad \rangle ? \langle vector \rangle \mid \text{GRAD} \langle scalar \rangle \mid \text{DIV} \langle dyad \rangle \mid \text{LAPL} \langle vector \rangle \mid (d=3) \text{ROT} \langle vector \rangle \mid \text{DIRDF}(\langle vector 1 \rangle, \langle vector 2 \rangle) \mid \text{DF}(\langle vector \rangle, \text{"usual further arguments"})$
 $\langle dyad \rangle ::=$ "identifier declared by DYADS statement" \mid "tensor variable with dyadic value" $\mid \text{DYAD}(\langle scalar 11 \rangle, \dots, \langle scalar 1d \rangle, \dots, \langle scalar d1 \rangle, \dots, \langle scalar dd \rangle) \mid -\langle dyad \rangle \mid \langle dyad 1 \rangle + \langle dyad 2 \rangle \mid \langle dyad 1 \rangle - \langle dyad 2 \rangle \mid \langle scalar \rangle * \langle dyad \rangle \mid \langle dyad \rangle / \langle scalar \rangle \mid \langle dyad 1 \rangle . \langle dyad 2 \rangle \mid \langle vector 1 \rangle \& \langle vector 2 \rangle \mid (d=3) \langle vector \rangle ? \langle dyad \rangle \mid (d=3) \langle dyad \rangle ? \langle vector \rangle \mid \text{GRAD} \langle vector \rangle \mid \text{DF}(\langle dyad \rangle, \text{"usual further arguments"})$

1.6 Assigning statement

The assigning statement for tensor variables has a usual syntax, namely:

$\langle tensor variable \rangle := \langle tensor \rangle$
 $\langle tensor variable \rangle ::=$ "identifier declared TENSOR" .

The assigning statement assigns the tensor variable the value of the given tensor expression, formulated in the given coordinate system. After a change of the coordinate system, the tensor variables have to be redefined.

References

- [1] M. C. Wirth, On the Automation of Computational Physics. PhDr Thesis. Report UCRL-52996, Lawrence Livermore National Laboratory, Livermore, 1980.

2 IIMET

A Module for Discretizing the Systems of Partial Differential Equations

This program module makes it possible to discretize the specified system of partial differential equations using the integro-interpolation method, minimizing the number of the used interpolations in each independent variable. It can be used for non-linear systems and vector or tensor variables as well. The user specifies the way of discretizing individual terms of differential equations, controls the discretization and obtains various difference schemes according to his own wish.

2.1 Specification of the coordinates and the indices corresponding to them

The independent variables of differential equations will be called coordinates. The names of the coordinates and the indices that will correspond to the particular coordinates in the difference scheme are defined using the COORDINATES statement:

```
COORDINATES <coordinate 1>{,<coordinate i>} [ INTO  
<index 1>{,<index i>}];  
<coordinate i> ::= "identifier" - the name of the coordinate  
<index i> ::= "identifier" - the name of the index
```

This statement specifies that the $\langle \text{coordinate } i \rangle$ will correspond to the $\langle \text{index } i \rangle$. A new COORDINATES statement cancels the definitions given by the preceding COORDINATES statement. If the part [INTO ...] is not included in the statement, the statement assigns the coordinates the indices I, J, K, L, M, N, respectively. If it is included, the number of coordinates and the number of indices should be the same.

2.2 Difference grids

In the discretization, orthogonal difference grids are employed. In addition to the basic grid, called the integer one, there is another, the half-integer grid in each coordinate, whose cellular boundary points lie in the centers of the cells of the integer grid. The designation of the cellular separating points and centers is determined by the CENTERGRID switch: if it is ON and the index in the given coordinate is I, the centers of the grid cells are designated by indices I, I + 1, ..., and the boundary points of the cells by indices I + 1/2, ..., if, on the contrary, the switch is OFF, the cellular centers are designated by indices I + 1/2, ..., and the boundary points by indices I, I + 1, ... (see Fig. 2.1).

ON CENTERGRID

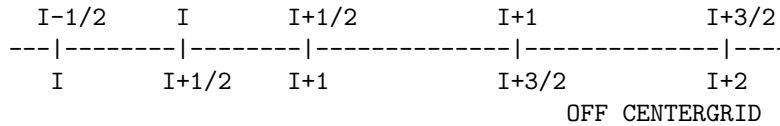


Figure 2.1 Types of grid

In the case of ON CENTERGRID, the indices $i, i+1, i-1, \dots$ thus designate the centers of the cells of the integer grid and the boundary points of the cells of the half-integer grid, and, similarly, in the case of OFF CENTERGRID, the boundaries of the cells of the integer grid and the central points of the half-integer grid. The meaning of the integer and half-integer grids depends on the CENTERGRID switch in the described way. After the package is loaded, the CENTERGRID is ON. Obviously, this switch is significant only for non-uniform grids with a variable size of each cell.

The grids can be uniform, i.e. with a constant cell size - the step of the grid. The following statement:

```
GRID UNIFORM, <coordinate>{, <coordinate>};
```

defines uniform grids in all coordinates occurring in it. Those coordinates that do not occur in the GRID UNIFORM statement are supposed to have non-uniform grids.

In the outputs, the grid step is designated by the identifier that is made by putting the character H before the name of the coordinate. For a uniform grid, this identifier (e.g. for the coordinate X the grid step HX) has the meaning of a step of an integer or half-integer grids that are identical. For a non-uniform grid, this identifier is an operator and has the meaning of a step of an integer grid, i.e. the length of a cell whose center (in the case of ON CENTERGRID) or beginning (in the case of OFF CENTERGRID) is designated by a single argument of this operator. For each coordinate s designated by the identifier i , this step of the integer non-uniform grid is defined as follows:

$$Hs(i+j) = s(i+j+1/2) - s(i+j-1/2) \text{ at ON CENTERGRID}$$

$$Hs(i+j) = s(i+j+1) - s(i+j) \text{ at OFF CENTERGRID}$$

for all integers j ($s(k)$ designates the value of the coordinate s in the cellular boundary point subscripted with the index k). The steps of the half-integer non-uniform grid are not applied in outputs.

2.3 Declaring the dependence of functions on coordinates

In the system of partial differential equations, two types of functions, in other words dependent variables can occur: namely, the given functions, whose values are known before the given system is solved, and the sought functions, whose values are not available until the system of equations is solved. The functions

can be scalar, vector, or tensor, for vector or tensor functions the EXPRES module has to be applied at the same time. The names of the functions employed in the given system and their dependence on the coordinates are specified using the DEPENDENCE statement.

```
DEPENDENCE <dependence>{,<dependence>};
<dependence> ::= <function>([<order>],<coordinate>){,
<coordinate>}}
<function> ::= "identifier" - the name of the function
<order> ::= 1—2 tensor order of the function (the value of
the function is 1 - vector, 2 - dyad (two-
dimensional tensor))
```

Every *<dependence>* in the statement determines on which *<coordinates>* the *<function>* depends. If the tensor *<order>* of the function occurs in the *<dependence>*, the *<function>* is declared as a vector or a dyad. If, however, the *<function>* has been declared by the VECTORS and DYADS statements of the EXPRES module, the user need not present the tensor *<order>*. By default, a function without any declaration is regarded as scalar. In the discretization, all scalar components of tensor functions are replaced by identifiers that arise by putting successively the function name and the individual indices of the given component (e.g. the tensor component T(1,2), written in the EXPRES module as T(1,2), is represented by the identifier T12). Before the DEPENDENCE statement is executed, the coordinates have to be defined using the COORDINATES statement. There may be several DEPENDENCE statements. The DEPENDENCE statement cancels all preceding determinations of which grids are to be used for differentiating the function or the equation for this function. These determinations can be either defined by the ISGRID or GRIDEQ statements, or computed in the evaluation of the IIM statement.

The GIVEN statement:

```
GIVEN <function>{,<function>};
```

declares all functions included in it as given functions whose values are known to the user or can be computed. The CLEARGIVEN statement:

```
CLEARGIVEN;
```

cancels all preceding GIVEN declarations. If the TWOGRID switch is ON, the given functions can be differentiated both on the integer and the half-integer grids. If the TWOGRID switch is OFF, any given function can be differentiated only on one grid. After the package is loaded, the TWOGRID is ON.

2.4 Functions and difference grids

Every scalar function or scalar component of a vector or a dyadic function occurring in the discretized system can be discretized in any of the coordinates either on the integer or half-integer grid. One of the tasks of the IIMET module

is to find the optimum distribution of each of these dependent variables of the system on the integer and half-integer grids in all variables so that the number of the performed interpolations in the integro-interpolation method will be minimal.

Using the statement

SAME $\langle function \rangle \{, \langle function \rangle \};$

all functions given in one of these declarations will be discretized on the same grids in all coordinates. In each SAME statement, at least one of these functions in one SAME statement must be the sought one. If the given function occurs in the SAME statement, it will be discretized only on one grid, regardless of the state of the TWOGRID switch. If a vector or a dyadic function occurs in the SAME statement, what has been said above relates to all its scalar components. There are several SAME statements that can be presented. All SAME statements can be canceled by the following statement:

CLEARSAME;

The SAME statement can be successfully used, for example, when the given function depends on the function sought in a complicated manner that cannot be included either in the differential equation or in the difference scheme explicitly, and when both the functions are desired to be discretized in the same points so that the user will not be forced to execute the interpolation during the evaluation of the given function.

In some cases, it is convenient too to specify directly which variable on which grid is to be discretized, for which case the ISGRID statement is applied:

ISGRID $\langle s-function \rangle \{, \langle s-function \rangle \};$

$\langle s-function \rangle ::= \langle function \rangle ([\langle component \rangle], \langle s-grid \rangle \{, \langle s-grid \rangle \})$

$\langle s-grid \rangle ::= \langle coordinate \rangle .. \langle grid \rangle,$

$\langle grid \rangle ::= \text{ONE} - \text{HALF}$ designation of the integer

(ONE) and half-integer (HALF)

grids

$\langle component \rangle ::= \langle i-dim \rangle -$ for the vector $\langle function \rangle$

$\langle i-dim \rangle, \langle i-dim \rangle$ for the dyadic $\langle function \rangle$

it is not presented for the

scalar $\langle function \rangle$

$\langle i-dim \rangle ::= * -$ "natural number from 1 to the space dimension

the space dimension is specified in the EXPRES

module by the SCALEFACTORS statement, * means all

components

The statement defines that the given functions or their components will be discretized in the specified coordinates on the specified grids, so that, for example, the statement ISGRID U (X..ONE, Y..HALF), V(1, Z..ONE), T(*, 1, X..HALF); defines that scalar U will be discretized on the integer grid in the coordinate X, and on the half-integer one in the coordinate Y, the first component of vector

V will be on the integer grid in the coordinate Z, and the first column of tensor T will be on the half-integer grid in the coordinate X. The ISGRID statement can be applied more times. The functions used in this statement have to be declared before by the DEPENDENCE statement.

2.5 Equations and difference grids

Every equation of the system of partial differential equations is an equation for some sought function (specified in the IIM statement). The correspondence between the sought functions and the equations is mutually unambiguous. The GRIDEQ statement makes it possible to determine on which grid an individual equation will be discretized in some or all coordinates

```
GRIDEQ <g-function>{,<g-function>};
<g-function> ::= <function>(<s-grid>{,<s-grid>})
```

Every equation can be discretized in any coordinate either on the integer or half-integer grid. This statement determines the discretization of the equations given by the functions included in it in given coordinates, on given grids. The meaning of the fact that an equation is discretized on a certain grid is as follows: index I used in the DIFMATCH statements (discussed in the following section), specifying the discretization of the basic terms, will be located in the center of the cell of this grid, and indices I+1/2, I-1/2 from the DIFMATCH statement on the boundaries of the cell of this grid. The actual name of the index in the given coordinate is determined using the COORDINATES statement, and its location on the grid is set by the CENTERGRID switch.

2.6 Discretization of basic terms

The discretization of a system of partial differential equations is executed successively in individual coordinates. In the discretization of an equation in one coordinate, the equation is linearized into its basic terms first that will be discretized independently then. If D is the designation for the discretization operator in the coordinate x, this linearization obeys the following rules:

1. $D(a+b) = D(a)+D(b)$
2. $D(-a) = -D(a)$
3. $D(p.a) = p.D(a)$ (p does not depend on the coordinate x)
4. $D(a/p) = D(a)/p$

The linearization lasts as long as some of these rules can be applied. The basic terms that must be discretized after the linearization have then the forms of the following quantities:

1. The actual coordinate in which the discretization is performed.
2. The sought function.

3. The given function.
4. The product of the quantities 1 - 7.
5. The quotient of the quantities 1 - 7.
6. The natural power of the quantities 1 - 7.
7. The derivative of the quantities 1 - 7 with respect to the actual coordinate.

The way of discretizing these basic terms, while the functions are on integer and half-integer grids, is determined using the DIFMATCH statement:

```

DIFMATCH <coordinate>,<pattern term>,{<grid specification>,<number of interpolations>,<discretized term>};
<coordinate> ::= ALL — "identifier" - the coordinate name from the COORDINATES statement
<pattern term> ::= <pattern coordinate>—
<pattern sought function>—
<pattern given function>—<pattern term> *
<pattern term>—<pattern term> / <pattern term>—
<pattern term> ** <exponent>—
DIFF(<pattern term>,<pattern coordinate>[,<order of derivative>])—
<declared operator>(<pattern term>{,<pattern term>})
<pattern coordinate> ::= X
<pattern sought function> ::= U — V — W
<pattern given function> ::= F — G
<exponent> ::= N — "integer greater than 1"
<order of derivative> ::= "integer greater than 2"
<grid specification> ::= <pattern function>=<grid>
<pattern function> ::= <pattern sought function>—
<pattern given function>
<number of interpolations> ::= "non-negative integer"
<discretized term> ::= <pattern operator>(<index expression>)—
"natural number"—DI—DIM1—DIP1—DIM2—DIP2—
<declared term> — - <discretized term> —
<discretized term> + <discretized term> —
<discretized term> * <discretized term> —
<discretized term> / <discretized term> —
(<discretized term>) —
<discretized term> **<exponent>
<pattern operator> ::= X — U — V — W — F — G
<index expression> ::= <pattern index> —
<pattern index> + <increment> —
<pattern index> - <increment>
<pattern index> ::= I
<increment> = "rational number"
DIFCONST <declared term>{,<declared term>};
<declared term> ::= "identifier" - the constant parameter of the difference scheme.

```

DIFFUNC $\langle \text{declared operator} \rangle \{ \langle \text{declared operator} \rangle \}$;
 $\langle \text{declared operator} \rangle ::=$ "identifier" - prefix operator, that can appear in discretized equations (e.g. SIN).

The first parameter of the DIFMATCH statement determines the coordinate for which the discretization defined in it is valid. If ALL is used, the discretization will be valid for all coordinates, and this discretization is accepted when it has been checked whether there has been no other discretization defined for the given coordinate and the given pattern term. Each pattern sought function, occurring in the pattern term, must be included in the specification of the grids. The pattern given functions from the pattern term can occur in the grid specification, but in some cases (see below) need not. In the grid specification the maximum number of 3 pattern functions may occur. The discretization of each pattern term has to be specified in all combinations of the pattern functions occurring in the grid specification, on the integer and half-integer grids, that is 2^n variants for the grid specification with n pattern functions (n=0,1,2,3). The discretized term is the discretization of the pattern term in the pattern coordinate X in the point X(I) on the pattern grid (see Fig. 2.2), and the pattern functions occurring in the grid specification are in the discretized term on the respective grids from this specification (to the discretized term corresponds the grid specification preceding it).

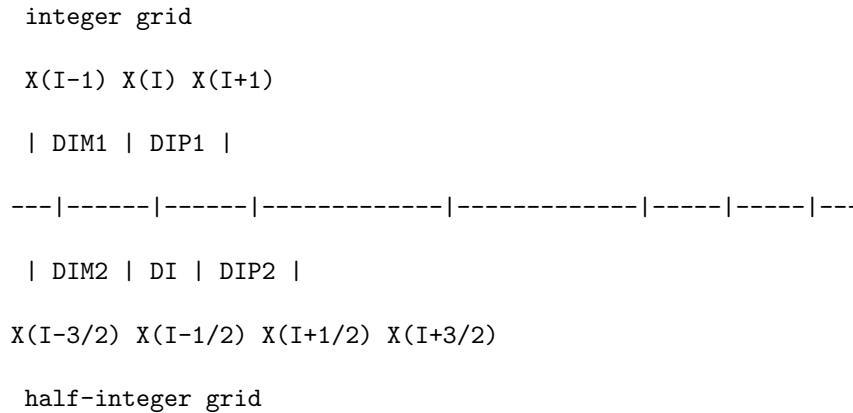


Figure 2.2 Pattern grid

The pattern grid steps defined as

$$\begin{aligned} \text{DIM2} &= X(I - 1/2) - X(I - 3/2) \\ \text{DIM1} &= X(I) - X(I - 1) \\ \text{DI} &= X(I + 1/2) - X(I - 1/2) \\ \text{DIP1} &= X(I + 1) - X(I) \end{aligned}$$

$$DIP2 = X(I + 3/2) - X(I + 1/2)$$

can occur in the discretized term.

In the integro-interpolation method, the discretized term is specified by the integral

$$\langle \text{discretized term} \rangle = 1/(X(I+1/2)-X(I-1/2)) * DINT(X(I-1/2), X(I+1/2), \langle \text{pattern term} \rangle, X),$$

where DINT is operator of definite integration DINT(from, to, function, variable). The number of interpolations determines how many interpolations were needed for calculating this integral in the given discrete form (the function on the integer or half-integer grid). If the integro-interpolation method is not used, the more convenient is the distribution of the functions on the half-integer and integer grids, the smaller number is chosen by the user. The parameters of the difference scheme defined by the DIFCONST statement can occur in the discretized expression too (for example, the implicit-explicit scheme on the implicit layer multiplied by the constant C and on the explicit one by (1-C)). As a matter of fact, all DIFMATCH statements create a base of pattern terms with the rules of how to discretize these terms in individual coordinates under the assumption that the functions occurring in the pattern terms are on the grids determined in the grid specification (all combinations must be included).

The DIFMATCH statement does not check whether the discretized term is actually the discretization of the pattern term or whether in the discretized term occur the functions from the grid specification on the grids given by this specification. An example can be the following definition of the discretization of the first and second derivatives of the sought function in the coordinate R on a uniform grid:

```
DIFMATCH R,DIFF(U,X),U=ONE,2,(U(I+1)-U(I-1))/(2*DI);
U=HALF,0,(U(I+1/2)-U(I-1/2))/DI;
DIFMATCH R,DIFF(U,X,2),U=ONE,0,(U(I+1)-2*U(I)+U(I-1))/DI**2,
U=HALF,2,(U(I+3/2)-U(I+1/2)-U(I-1/2)+U(I-3/2))/(2*DI**2);
All DIFMATCH statements can be cleared by the statement
```

```
CLEARDIFMATCH;
```

After this statement user has to supply its own DIFMATCH statements. But now back to the discretizing of the basic terms obtained by the linearization of the partial differential equation, as mentioned at the beginning of this section. Using the method of pattern matching, for each basic term a term representing its pattern is found in the base of pattern terms (specified by the DIFMATCH statements). The pattern matching obeys the following rules:

1. The pattern for the coordinate in which the discretization is executed is the pattern coordinate X.
2. The pattern for the sought function is some pattern sought function, and this correspondence is mutually unambiguous.

3. The pattern for the given function is some pattern given function, or, in case the EQFU switch is ON, some pattern sought function, and, again, the correspondence of the pattern with the given function is mutually unambiguous (after loading the EQFU switch is ON).
4. The pattern for the products of quantities is the product of the patterns of these quantities, irrespective of their sequence.
5. The pattern for the quotient of quantities is the quotient of the patterns of these quantities.
6. The pattern for the natural power of a quantity is the same power of the pattern of this quantity or the power of this quantity with the pattern exponent N.
7. The pattern for the derivative of a quantity with respect to the coordinate in which the discretization is executed is the derivative of the pattern of this quantity with respect to the pattern coordinate X of the same order of differentiation.
8. The pattern for the sum of the quantities that have the same pattern with the identical correspondence of functions and pattern functions is this common pattern (so that it will not be necessary to multiply the parentheses during discretizing the products in the second and further coordinates).

When matching the pattern of one basic term, the program finds the pattern term and the functions corresponding to the pattern functions, maybe also the exponent corresponding to the pattern exponent N. After determining on which grids the individual functions and the individual equations will be discretized, which will be discussed in the next section, the program finds in the pattern term base the discretized term either with pattern functions on the same grids as are the functions from the basic term corresponding to them in case that the given equation is differentiated on the integer grid, or with pattern functions on inverse grids (an inverse integer grid is a half-integer grid, and vice versa) compared with those used for the functions from the basic term corresponding to them in case the given equation is differentiated on the half-integer grid (the discretized term in the DIFMATCH statement is expressed in the point $X(I)$, i.e. on the integer grid, and holds for the discretizing of the equation on the integer grid; with regard to the substitutions for the pattern index I mentioned later, it is possible to proceed in this way and not necessary to define the discretization in the points $X(I+1/2)$ too, i.e. on the half-integer grid). The program replaces in the thus obtained discretized term:

1. The pattern coordinate X with the particular coordinate s in which the discretization is actually performed.

2. The pattern index I and the grid steps DIM2, DIM1, DI, DIP1, DIP2 with the expression given in table 2.1 according to the state of the CENTERGRID switch and to the fact whether the given equation is discretized on the integer or half-integer grid (i is the index corresponding to the coordinate s according to the COORDINATES statement, the grid steps were defined in section 2.2)
3. The pattern functions with the corresponding functions from the basic term and, possibly, the pattern exponent with the corresponding exponent from the basic term.

		the equation discretized on	
		the integer grid	the half-integer grid
		CENTERGRID	CENTERGRID
		OFF	ON
I	i		i+1/2
DIM2	$(Hs(i-2)+Hs(i-1))/2$	$Hs(i-1)$	$(Hs(i-1)+Hs(i))/2$
DIM1	$Hs(i-1)$	$(Hs(i-1)+Hs(i))/2$	$Hs(i)$
DI	$(Hs(i-1)+Hs(i))/2$	$Hs(i)$	$(Hs(i)+Hs(i+1))/2$
DIP1	$Hs(i)$	$(Hs(i)+Hs(i+1))/2$	$Hs(i+1)$
DIP2	$(Hs(i)+Hs(i+1))/2$	$Hs(i+1)$	$(Hs(i+1)+Hs(i+2))/2$

Table 2.1 Values of the pattern index and the pattern grid steps.

More details will be given now to the discretization of the given functions and its specification. The given function may occur in the SAME statement, which makes it bound with some sought function, in other words it can be discretized only on one grid. This means that all basic terms, in which this function occurs, must have their pattern terms in whose discretization definitions by the DIFMATCH statement the pattern function corresponding to the mentioned given function has to occur in the grid specification. If the given function does

not occur in the SAME statement and the TWOGRID switch is OFF, i.e. it can be discretized only on one grid again, the same holds true. If, however, the given function does not occur in the SAME statement and the TWOGRID switch is ON, i.e. it can be discretized simultaneously on the integer and the half-integer grids, then the basic terms of the equations including this function have their pattern terms in whose discretization definitions the pattern function corresponding to the mentioned given function need not occur in the grid specification. If, however, in spite of all, this pattern function in the discretization definition does occur in the grid specification, it is the alternative with a smaller number of interpolations occurring in the DIFMATCH statement that is selected for each particular basic term with a corresponding pattern (the given function can be on the integer or half-integer grid).

Before the discretization is executed, it is necessary to define using the DIFMATCH statements the discretization of all pattern terms that are the patterns of all basic terms of all equations appearing in the discretized system in all coordinates. The fact that the pattern terms of the basic terms of partial equations occur repeatedly in individual systems has made it possible to create a library of the discretizations of the basic types of pattern terms using the integro-interpolation method. This library is a component part of the IIMET module (in its end) and makes work easier for those users who find the pattern matching mechanism described here too difficult. New DIFMATCH statements have to be created by those whose equations will contain a basic term having no pattern in this library, or those who need another method to perform the discretization. The described implemented algorithm of discretizing the basic terms is sufficiently general to enable the use of a nearly arbitrary discretization on orthogonal grids.

2.7 Discretization of a system of equations

All statements influencing the run of the discretization that one want use in this run have to be executed before the discretization is initiated. The COORDINATES, DEPENDENCE, and DIFMATCH statements have to occur in all applications. Further, if necessary, the GRID UNIFORM, GIVEN, ISGRID, GRIDEQ, SAME, and DIFCONST statements can be used, or some of the CENTREGRID, TWOGRID, EQFU, and FULLEQ switches can be set. Only then the discretization of a system of partial differential equations can be started using the IIM statement:

```
IIM <array>{,<sought function>,<equation>};
<array> ::= "identifier" - the name of the array for storing
the result
<sought function> ::= "identifier" - the name of the function
whose behavior is described by the
equation
```

$\langle equation \rangle ::= \langle left\ side \rangle = \langle right\ side \rangle$
 $\langle left\ side \rangle ::=$ "algebraic expression" , the derivatives are
 designated by the DIFF operator
 $\langle right\ side \rangle ::=$ "algebraic expression"

Hence, in the IIM statement the name of the array in which the resulting difference schemes will be stored, and the pair sought function - equation, which describes this function, are specified. The meaning of the relation between the sought function and its equation during the discretization lies in the fact that the sought function is preferred in its equation so that the interpolation is not, if possible, used in discretizing the terms of this equation that contain it. In the equations, the functions and the coordinates appear as identifiers. The identifiers that have not been declared as functions by the DEPENDENCE statement or as coordinates by the COORDINATES statement are considered constants independent of the coordinates. The partial derivatives are expressed by the DIFF operator that has the same syntax as the standard differentiation operator DF. The functions and the equations can also have the vector or tensor character. If these non-scalar quantities are applied, the EXPRES module has to be used together with the IIMET module, and also non-scalar differential operators such as GRAD, DIV, etc. can be employed.

The sequence performed by the program in the discretization can be briefly summed up in the following items:

1. If there are non-scalar functions or equations in a system of equations, they are automatically converted into scalar quantities by means of the EXPRES module.
2. In each equation, the terms containing derivatives are transferred to the left side, and the other terms to the right side of the equation.
3. For each coordinate, with respect to the sequence in which they occur in the COORDINATES statement, the following is executed:
 - a) It is determined on which grids all functions and all equations in the actual coordinate will be discretized, and simultaneously the limits are kept resulting from the ISGRID, GRIDEQ, and SAME statements if they were used. Such a distribution of functions and equations on the grids is selected among all possible variants that ensures the minimum sum of all numbers of the interpolations of the basic terms (specified by the DIFMATCH statement) of all equations if the FULLEQ switch is ON, or of all left sides of the equations if the FULLEQ switch is OFF (after the loading the FULLEQ switch is ON).
 - b) The discretization itself is executed, as specified by the DIFMATCH statements.
4. If the array name is A, then if there is only one scalar equation in the IIM statement, the discretized left side of this equation is stored in A(0) and the discretized right side in A(1) (after the transfer mentioned in item 2), if there are more scalar equations than one in the IIM statement, the discretization of the left side of the i-th scalar equation is stored in A(i,0) and the discretization of the right side in A(i,1).

The IIM statement can be used more times during one program run, and between its calls, the discretizing process can be altered using other statements of this module.

2.8 Error messages

The IIMET module provides error messages in the case of the user's errors. Similarly as in the REDUCE system, the error reporting is marked with five stars : "*****" on the line start. Some error messages are identical with those of the REDUCE system. Here are given some other error messages that require a more detailed explanation:

***** Matching of X term not found - the discretization of the pattern term that is the pattern of the basic term printed on the place X has not been defined (using the DIFMATCH statement)

***** Variable of type F not defined on grids in DIFMATCH - in the definition of the discretizing of the pattern term the given functions were not used in the grid specification and are needed now

***** X Free vars not yet implemented - in the grid specification in the DIFMATCH statement more than 3 pattern functions were used

***** All grids not given for term X - in the definition of the discretization of the pattern of the basic term printed on the place X not all necessary combinations of the grid specification of the pattern functions were presented

3 APPROX

A Module for Determining the Precision Order of the Difference Scheme

This module makes it possible to determine the differential equation that is solved by the given difference scheme, and to determine the order of accuracy of the solution of this scheme in the grid steps in individual coordinates. The discrete function values are expanded into the Taylor series in the specified point.

3.1 Specification of the coordinates and the indices corresponding to them

The COORDINATES statement, described in the IIMET module manual, specifying the coordinates and the indices corresponding to them is the same for this program module as well. It has the same meaning and syntax. The present module version assumes a uniform grid in all coordinates. The grid step in the input difference schemes has to be designated by an identifier consisting of the

character H and the name of the coordinate, e.g. the step of the coordinate X is HX.

3.2 Specification of the Taylor expansion

In the determining of the approximation order, all discrete values of the functions are expanded into the Taylor series in all coordinates. In order to determine the Taylor expansion, the program needs to know the point in which it performs this expansion, and the number of terms in the Taylor series in individual coordinates. The center of the Taylor expansion is specified by the CENTER statement and the number of terms in the Taylor series in individual coordinates by the MAXORDER statement:

```
CENTER  $\langle center \rangle \{, \langle center \rangle\};$   
 $\langle center \rangle ::= \langle coordinate \rangle = \langle increment \rangle$   
 $\langle increment \rangle ::=$  "rational number"  
MAXORDER  $\langle order \rangle \{, \langle order \rangle\};$   
 $\langle order \rangle ::= \langle coordinate \rangle = \langle number of terms \rangle$   
 $\langle number of terms \rangle ::=$  "natural number"
```

The increment in the CENTER statement determines that the center of the Taylor expansion in the given coordinate will be in the point specified by the index $I + \langle increment \rangle$, where I is the index corresponding to this coordinate, defined using the COORDINATES statement, e.g. the following example

```
COORDINATE T,X INTO N,J;  
CENTER T = 1/2, X = 1;  
MAXORDER T = 2, X = 3;
```

specifies that the center of the Taylor expansion will be in the point $(t(n+1/2), x(j+1))$ and that until the second derivatives with respect to t (second powers of ht) and until the third derivatives with respect to x (third powers of hx) the expansion will be performed. The CENTER and MAXORDER statements can be placed only after the COORDINATES statement. If the center of the Taylor expansion is not defined in some coordinate, it is supposed to be in the point given by the index of this coordinate (i.e. zero increment). If the number of the terms of the Taylor expansion is not defined in some coordinate, the expansion is performed until the third derivatives with respect to this coordinate.

3.3 Function declaration

All functions whose discrete values are to be expanded into the Taylor series must be declared using the FUNCTIONS statement:

```
FUNCTIONS  $\langle name of function \rangle \{, \langle name of function \rangle\};$ 
```

$\langle name\ of\ function \rangle ::= \text{"identifier"}$

In the specification of the difference scheme, the functions are used as operators with one or more arguments, designating the discrete values of the functions. Each argument is the sum of the coordinate index (from the COORDINATES statement) and a rational number. If some index is omitted in the arguments of a function, this functional value is supposed to lie in the point in which the Taylor expansion is performed, as specified by the CENTER statement. In other words, if the COORDINATES and CENTER statements, shown in the example in the previous section, are valid, then it holds that $U(N+1) = U(N+1, J+1)$ and $U(J-1) = U(N+1/2, J-1)$. The FUNCTIONS statement can declare both the sought and the known functions for the expansion.

3.4 Order of accuracy determination

The order of accuracy of the difference scheme is determined by the APPROX statement:

APPROX ($\langle diff.\ scheme \rangle$);

$\langle diff.\ scheme \rangle ::= \langle l.\ side \rangle = \langle r.\ side \rangle$

$\langle l.\ (r.)\ side \rangle ::= \text{"algebraic expression"}$

In the difference scheme occur the functions in the form described in the preceding section, the coordinate indices and the grid steps described in section 3.1, and the other symbolic parameters of the difference scheme. The APPROX statement expands all discrete values of the functions declared in the FUNCTIONS statement into the Taylor series in all coordinates (the point in which the Taylor expansion is performed is specified by the CENTER statement, and the number of the expansion terms by the MAXORDER statement), substitutes the expansions into the difference scheme, which gives a modified differential equation. The modified differential equation, containing the grid steps too, is an equation that is really solved by the difference scheme (into the given orders in the grid steps).

The partial differential equation, whose solution is approximated by the difference scheme, is determined by replacing the grid steps by zeros and is displayed after the following message:

"Difference scheme approximates differential equation"

Then the following message is displayed:

"with orders of approximation:"

and the lowest powers (except for zero) of the grid steps in all coordinates, occurring in the modified differential equation are written. If the PRAPPROX switch is ON, then the rest of the modified differential equation is printed. If this rest is added to the left hand side of the approximated differential equation,

one obtain modified equation. By default the PRAPPROX switch is OFF. If the grid steps are found in some denominator in the modified equation, i.e. with a negative exponent, the following message is written, preceding the approximated differential equation:

”Reformulate difference scheme, grid steps remain in denominator”
and the approximated differential equation is not correctly determined (one of its sides is zero). Generally, this message means that there is a term in the difference scheme that is not a difference replacement of the derivative, i.e. the ratio of the differences of the discrete function values and the discrete values of the coordinates (the steps of the difference grid). The user, however, must realize that in some cases such a term occurs purposefully in the difference scheme (e.g. on the grid boundary to keep the scheme conservative).

4 CHARPOL

A Module for Calculating the Amplification Matrix and the Characteristic Polynomial of the Difference Scheme

This program module is used for the first step of the stability analysis of the difference scheme using the Fourier method. It substitutes the Fourier components into the difference scheme, calculates the amplification matrix of the scheme for transition from one time layer to another, and computes the characteristic polynomial of this matrix.

4.1 Commands common with the IIMET module

The COORDINATES and GRID UNIFORM statements, described in the IIMET module manual, are applied in this module as well, having the same meaning and syntax. The time coordinate is assumed to be designated by the identifier T. The present module version requires all coordinates to have uniform grids, i.e. to be declared in the GRID UNIFORM statement. The grid step in the input difference schemes has to be designated by the identifier consisting of the character H and the name of the coordinate, e.g. the step of the time coordinate T is HT.

4.2 Function declaration

The UNFUNC statement declares the names of the sought functions used in the difference scheme:

```
UNFUNC <function>{,<function>}
```

$\langle function \rangle ::=$ "identifier" - the name of the sought function

The functions are used in the difference schemes as operators with one or more arguments for designating the discrete function values. Each argument is the sum of the index (from the COORDINATES statement) and a rational number. If some index is omitted in the function arguments, this function value is supposed to lie in the point specified only by this index, which means that, with the indices N and J and the function U, it holds that $U(N+1) = U(N+1, J)$ and $U(J-1) = U(N, J-1)$. As two-step (in time) difference schemes may be used only, the time index may occur either completely alone in the arguments, or in the sum with a one.

4.3 Amplification matrix

The AMPMAT matrix operator computes the amplification matrix of a two-step difference scheme. Its argument is an one column matrix of the dimension (1,k), where k is the number of the equations of the difference scheme, that contains the difference equations of this scheme as algebraic expressions equal to the difference of the right and left sides of the difference equations. The value of the AMPMAT matrix operator is the square amplification matrix of the dimension (k,k). During the computation of the amplification matrix, two new identifiers are created for each spatial coordinate. The identifier made up of the character K and the name of the coordinate represents the wave number in this coordinate, and the identifier made up of the character A and the name of the coordinate represents the product of this wave number and the grid step in this coordinate divided by the least common multiple of all denominators occurring in the scheme in the function argument containing the index of this coordinate. On the output an equation is displayed defining the latter identifier. For example, if in the case of function U and index J in the coordinate X the expression $U(J+1/2)$ has been used in the scheme (and, simultaneously, no denominator higher than 2 has occurred in the arguments with J), the following equation is displayed: $AX: = (KX*HX)/2$. The definition of these quantities As allows to express every sum occurring in the argument of the exponentials as the sum of these quantities multiplied by integers, so that after a transformation, the amplification matrix will contain only $\sin(As)$ and $\cos(As)$ (for all spatial coordinates s). The AMPMAT operator performs these transformations automatically. If the PRFOURMAT switch is ON (after the loading it is ON), the matrices H0 and H1 (the amplification matrix is equal to $-H1**(-1)*H0$) are displayed during the evaluation of the AMPMAT operator. These matrices can be used for finding a suitable substitution for the goniometric functions in the next run for a greater simplification.

The TCON matrix operator transforms the square matrix into a Hermit-conjugate matrix, i.e. a transposed and complex conjugate one. Its argument is the square matrix and its value is Hermit-conjugate matrix of the argument. The Hermit-conjugate matrix is used for testing the normality and unitarity of the amplification matrix in the determining of the sufficient stability condition.

4.4 Characteristic polynomial

The CHARPOL operator calculates the characteristic polynomial of the given square matrix. The variable of the characteristic polynomial is designated by the LAM identifier. The operator has one argument, the square matrix, and its value is its characteristic polynomial in LAM.

4.5 Automatic denotation

Several statements and procedures are designed for automatic denotation of some parts of algebraic expressions by identifiers. This denotation is namely useful when we obtain very large expressions, which cannot fit into the available memory. We can denote subparts of an expression from the previous step of calculation by identifiers, replace these subparts by these identifiers and continue the analytic calculation only with these identifiers. Every time we use this technique we have to explicitly survive in processed expressions those algebraic quantities which will be necessary in the following steps of calculation. The process of denotation and replacement is performed automatically and the algebraic values which are denoted by these new identifiers can be written out at any time. We describe how this automatic denotation can be used.

The statement DENOTID defines the beginning letters of newly created identifiers. Its syntax is

```
DENOTID  $\langle id \rangle$ ;  
 $\langle id \rangle ::=$  "identifier"
```

After this statement the new identifiers created by the operators DENOTEPOL and DENOTEMAT will begin with the letters of the identifier $\langle id \rangle$ used in this statement. Without using any DENOTID statement all new identifiers will begin with one letter A. We suggest to use this statement every time before using operators DENOTEPOL or DENOTEMAT with some new identifier and to choose identifiers used in this statement in such a way that the newly created identifiers are not equal to any identifiers used in the expressions you are working with.

The operator DENOTEPOL has one argument, a polynomial in LAM, and denotes the real and imaginary part of its coefficients by new identifiers. The real part of the j -th LAM power coefficient is denoted by the identifier $\langle id \rangle R0j$ and the imaginary part by $\langle id \rangle I0j$, where $\langle id \rangle$ is the identifier used in the last DENOTID statement. The denotation is done only for non-numeric coefficients. The value of this operator is the polynomial in LAM with coefficients constructed from the new identifiers. The algebraic expressions which are denoted by these identifiers are stored as LISP data structure standard quotient in the LISP variable DENOTATION!* (assoc. list).

The operator DENOTEMAT has one argument, a matrix, and denotes the real and imaginary parts of its elements. The real part of the (j,k) matrix element is denoted by the identifier $\langle id \rangle Rjk$ and the imaginary part by $\langle id \rangle Ijk$. The

returned value of the operator is the original matrix with non-numeric elements replaced by $\langle id \rangle R_{jk} + I^* \langle id \rangle I_{jk}$. Other matters are the same as for the DENOTEPOL operator.

The statement PRDENOT has the syntax

PRDENOT;

and writes from the variable DENOTATION!* the definitions of all new identifiers introduced by the DENOTEPOL and DENOTEMAT operators since the last call of CLEARDENOT statement (or program start) in the format defined by the present setting of output control declarations and switches. The definitions are written in the same order as they have been entered, so that the definitions of the first DENOTEPOL or DENOTEMAT operators are written first. This order guarantees that this statement can be utilized directly to generate a semantically correct numerical program (the identifiers from the first denotation can appear in the second one, etc.).

The statement CLEARDENOT with the syntax

CLEARDENOT;

clears the variable DENOTATION!*, so that all denotations saved earlier by the DENOTEPOL and DENOTEMAT operators in this variable are lost. The PRDENOT statement succeeding this statement writes nothing.

5 HURWP A Module for Polynomial Roots Locating

This module is used for verifying the stability of a polynomial, i.e. for verifying if all roots of a polynomial lie in a unit circle with its center in the origin. By investigating the characteristic polynomial of the difference scheme, the user can determine the conditions of the stability of this scheme.

5.1 Conformal mapping

The HURW operator transforms a polynomial using the conformal mapping $LAM=(z+1)/(z-1)$. Its argument is a polynomial in LAM and its value is a transformed polynomial in LAM ($LAM=z$). If P is a polynomial in LAM, then it holds: all roots LAM1i of the polynomial P are in their absolute values smaller than one, i.e. $—LAM1i— < 1$, iff the real parts of all roots LAM2i of the HURW(P) polynomial are negative, i.e. $Re(LAM2i) < 0$.

The elimination of the unit polynomial roots ($LAM=1$), which has to occur before the conformal transformation is performed, is made by the TROOT1 operator. The argument of this operator is a polynomial in LAM and its value is a polynomial in LAM not having its root equal to one any more. Mostly, the investigated polynomial has some more parameters. For some special values of

those parameters, the polynomial may have a unit root. During the evaluation of the TROOT1 operator, the condition concerning the polynomial parameters is displayed, and if it is fulfilled, the resulting polynomial has a unit root.

5.2 Investigation of polynomial roots

The HURWITZP operator checks whether a polynomial is the Hurwitz polynomial, i.e. whether all its roots have negative real parts. The argument of the HURWITZP operator is a polynomial in LAM with real or complex coefficients, and its value is YES if the argument is the Hurwitz polynomial. It is NO if the argument is not the Hurwitz polynomial, and COND if it is the Hurwitz polynomial when the conditions displayed by the HURWITZP operator during its analysis are fulfilled. These conditions have the form of inequalities and contain algebraic expressions made up of the polynomial coefficients. The conditions have to be valid either simultaneously, or they are designated and a proposition is created from them by the AND and OR logic operators that has to be fulfilled (it is the condition concerning the parameters occurring in the polynomial coefficient) by a polynomial to be the Hurwitz one. This proposition is the sufficient condition, the necessary condition is the fulfillment of all the inequalities displayed.

If the HURWITZP operator is called interactively, the user is directly asked if the inequalities are or are not valid. The user responds "Y" if the displayed inequality is valid, "N" if it is not, and "?" if he does not know whether the inequality is true or not.

6 LINBAND

A Module for Generating the Numeric Program for Solving a System of Linear Algebraic Equations with Band Matrix

The LINBAND module generates the numeric program in the FORTRAN language, which solves a system of linear algebraic equations with band matrix using the routine from the LINPACK, NAG ,IMSL or ESSL program library. As input data only the system of equations is given to the program. Automatically, the statements of the FORTRAN language are generated that fill the band matrix of the system in the corresponding memory mode of chosen library, call the solving routine, and assign the chosen variables to the solution of the system. The module can be used for solving linear difference schemes often having the band matrix.

6.1 Program generation

The program in the FORTRAN language is generated by the GENLINBAND-SOL statement (the braces in this syntax definition occur directly in the program

and do not have the usual meaning of the possibility of repetition, they designate REDUCE lists):

```

GENLINBANDSOL (<n-lower>,<n-upper>,{<system>});
<n-lower> ::= "natural number"
<n-upper> ::= "natural number"
<system> ::= <part of system> — <part of system>,<system>
<part of system> ::= {<variable>,<equation>} — <loop>
<variable> ::= "kernel"
<equation> ::= <left side> = <right side>
<left side> ::= "algebraic expression"
<right side> ::= "algebraic expression"
<loop> ::= {DO,{<parameter>,<from>,<to>,<step>},<c-system> }
<parameter> ::= "identifier"
<from> ::= <i-expression>
<to> ::= <i-expression>
<step> ::= <i-expression>
<i-expression> ::= "algebraic expression" with natural value
(evaluated in FORTRAN)
<c-system> ::= <part of c-system> — <part of c-system>,<c-system>
<part of c-system> ::= {<variable>,<equation>}

```

The first and second argument of the GENLINBANDSOL statement specifies the number of the lower (below the main diagonal) and the upper diagonals of the band matrix of the system. The system of linear algebraic equations is specified by means of lists expressed by braces { } in the REDUCE system. The variables of the equation system can be identifiers, but most probably they are operators with an argument or with arguments that are analogous to array in FORTRAN. The left side of each equation has to be a linear combination of the system variables, the right side, on the contrary, is not allowed to contain any variables of the system. The sequence of the band matrix lines is given by the sequence of the equations, and the sequence of the columns by the sequence of the variables in the list describing the equation system.

The meaning of the loop in the system list is similar to that of the DO loop of the FORTRAN language. The individual variables and equations described by the loop are obtained as follows:

1. $\langle parameter \rangle = \langle from \rangle$.
2. The $\langle parameter \rangle$ value is substituted into the variables and equations of the $\langle c-system \rangle$ loop, by which further variables and equations of the system are obtained.
3. $\langle parameter \rangle$ is increased by $\langle step \rangle$.
4. If $\langle parameter \rangle$ is less or equal $\langle to \rangle$, then go to step 2, else all variables and equations described by the loop have already been obtained.

The variables and equations of the system included in the loop usually contain the loop parameter, which mostly occur in the operator arguments in the REDUCE language, or in the array indices in the FORTRAN language. If $NL = \langle n\text{-lower} \rangle$, $NU = \langle n\text{-upper} \rangle$, and for some loop $F = \langle from \rangle$, $T = \langle to \rangle$, $S = \langle step \rangle$ and N is the number of the equations in the loop $\langle c\text{-system} \rangle$, it has to be true that

$$UP(NL/N) + UP(NU/N) < DOWN((T-F)/S)$$

where UP represents the rounding-off to a higher natural number, and $DOWN$ the rounding-off to a lower natural number. With regard to the fact that, for example, the last variable before the loop is not required to equal the last variable from the loop system, into which the loop parameter equal to $F-S$ is substituted, when the band matrix is being constructed, from the FORTRAN loop that corresponds to the loop from the specification of the equation system, at least the first NL variables-equations have to be moved to precede the FORTRAN loop, and at least the last NU variables-equations have to be moved to follow this loop in order that the correspondence of the system variables in this loop with the system variables before and after this loop will be secured. And this move requires the above mentioned condition to be fulfilled. As, in most cases, NL/N and NU/N are small with respect to $(T-F)/S$, this condition does not represent any considerable constrain.

The loop parameters $\langle from \rangle$, $\langle to \rangle$, and $\langle step \rangle$ can be natural numbers or expressions that must have natural values in the run of the FORTRAN program.

6.2 Choosing the numerical library

The user can choose the routines of which numerical library will be used in the generated FORTRAN code. The supported numerical libraries are: LINPACK, NAG, IMSL and ESSL (IBM Engineering and Scientific Subroutine Library). The routines DGBFA, DGBSL (band solver) and DGTSL (tridiagonal solver) are used from the LINPACK library, the routines F01LBF, F04LDF (band solver) and F01LEF, F04LEF (tridiagonal solver) are used from the NAG library, the routine LEQT1B is used from the IMSL library and the routines DGBF, DGBS (band solver) and DGTF, DGTS (tridiagonal solver) are used from the ESSL library. By default the LINPACK library routines are used. The using of other libraries is controlled by the switches NAG,IMSL and ESSL. All these switches are by default OFF. If the switch IMSL is ON then the IMSL library routine is used. If the switch IMSL is OFF and the switch NAG is ON then NAG library routines are used. If the switches IMSL and NAG are OFF and the switch ESSL is ON then the ESSL library is used. During generating the code using LINPACK, NAG or ESSL libraries the special routines are use for systems with tridiagonal matrices, because tridiagonal solvers are faster than the band matrix solvers.

6.3 Completion of the generated code

The GENLINBANDSOL statement generates a block of FORTRAN code (a block of statements of the FORTRAN language) that performs the solution of the given system of linear algebraic equations. In order to be used, this block of code has to be completed with some declarations and statements, thus getting a certain envelope that enables it to be integrated into the main program.

In order to be able to work, the generated block of code has to be preceded by:

1. The declaration of arrays as described by the comments generated into the FORTRAN code (near the calling of library routines)
2. The assigning the values to the integer variables describing the real dimensions of used arrays (again as described in generated FORTRAN comments)
3. The filling of the variables that can occur in the loop parameters.
4. The filling or declaration of all variables and arrays occurring in the system equations, except for the variables of the system of linear equations.
5. The definition of subroutine ERROUT the call to which is generated after some routines found that the matrix is algorithmically singular.

The mentioned envelope for the generated block can be created manually, or directly using the GENTRAN program package for generating numeric programs. The LINBAND module itself uses the GENTRAN package, and the GENLINBANDSOL statement can be applied directly in the input files of the GENTRAN package (template processing). The GENTRAN package has to be loaded prior to loading of the LINBAND module.

The generated block of FORTRAN code has to be linked with the routines from chosen numerical library.

References

- [1] R. Liska: Numerical Code Generation for Finite Difference Schemes Solving. In IMACS World Congress on Computation and Applied Mathematics. Dublin, July 22-26, 1991, Dublin,(In press).

33