Documentation for PHOENICS

TR 003

In-Form
(Version 2006)

Title:	In-Form
CHAM Ref:	CHAM/TR003
Document rev:	03
Doc. release date:	05 April 2005
Last revised date:	13 June 2006
Software version:	PHOENICS 2006
Responsible author:	D B Spalding
Other contributors:	
Editor:	J C Ludwig
Published by:	CHAM
Confidentiality:	
Classification:	Unclassified

CHAM, Bakery House, 40 High Street, Wimbledon, London SW19 5AU, UK
Telephone: 020 8947 7651 Fax: 020 8879 3497
E-mail: phoenics@cham.co.uk
Web site: http://www.cham.co.uk

CHAM

Computer Simulation of fluid flow, heat flow, chemical reaction and stresses in solids

In-Form: TR 003

The present document describes the new method of setting up flow-simulation problems which greatly extends the ability of PHOENICS users to introduce novelty and variety into their work.

From its inception in 1981, PHOENICS provided users with access to open-source coding, into which they could introduce what Fortran subroutines and functions they desired.

Then, in the mid-nineties, "PLANT" was introduced. This enabled the user to express his or her wishes by way of simple-syntax formulae, whereupon PHOENICS itself created the corresponding Fortran, and performed the necessary compilation and re-building.

Now, In-Form accepts an even wider range of formulae, with even simpler syntax; and it enables PHOENICS to interpret them without any compilation or executable-building at all!

The document consists of a reprint of the PHOENICS-Encyclopaedia article on In-Form, without however any of the graphical illustrations.

It has been included in the hard-copy documentation for the convenience of users who will, it is expected, make much use of the new feature.

In-Form; the Input of Data by way of Formulae

Summary

In-Form is a supplement to the PHOENICS Input Language (PIL) which facilitates the input of problem-defining data. Specifically, it allows users to express their requirements through algebraic formulae, whether for:

- space and time discretization,
- material properties,
- initial values,
- sources,
- boundary conditions,
- body shapes and motions, or
- special print-out features.

The formulae are placed in the data-input file, Q1 by way of a text editor, or via the graphical user interface of PHOENICS, the Virtual-Reality Editor.

The PHOENICS input module, Satellite, thereafter transmits the implications of the formulae to the solver module, EARTH. There they are recognised as instructions to perform the appropriate computations. The user has nothing more to do except await and inspect the computed results.

Note to users of earlier versions of PHOENICS:

Unlike the earlier PLANT feature of PHOENICS, In-Form creates no new Fortran coding and therefore needs no 're-compilable' version of the software.

PLANT still exists and functions as before, for users who have become accustomed to it; but In-Form can now do all that PLANT can do, and more.

Moreover, In-Form not only greatly increases the range of admissible new-to-PHOENICS features: it also allows already existing features to be introduced in simpler ways. In particular, it renders unnecessary the use of GRNDx as an option indicator; and even the long-established COVAL command can often be replaced by a simpler statement.

Contents

This page intentionally left blank.

1. Why In-Form has been created

1.1. Pre-In-Form methods of specifying flow-simulation tasks

(a) PIL, GROUND coding and PLANT

Prior to the introduction of In-Form, there were three methods for defining flow-simulation tasks for PHOENICS, namely:

1 by writing PHOENICS-Input-Language (PIL) statements in a Q1 file, so as to activate selected built-in features of the solver module, EARTH; this was always used, whether the Q1 was created by:

- o selection from a library of such files,
- o use of a text editor,
- o use of the Graphical User Interface, or
- o some combination of the above three;

2 by providing additional problem-specific Fortran-code modules, the so-called 'GROUND coding', which could be compiled and linked into a new EARTH executable;

3 by adding supplementary PLANT-activating statements to the Q1 file, which caused appropriate new Fortran to be written automatically, whereafter the new executable was created and run.

Method 1 has sufficed for the majority of users for most of their requirements, not least because the standard solver module is richly endowed with already-prepared GROUND-type sub-routines which can be switched on or of by pointers (the GRNDx flags) placed in Q1.

However, it is impossible to provide sufficient of these to satisfy all the needs which users may at some time express; therefore methods 2 or 3 have been extensively and successfully employed by many users throughout the years. Besides, keeping track of what all the GRNDx pointers mean is rather tiresome.

Method 2, of course, is accessible only to users with sufficient knowledge of Fortran, and the leisure to exploit it.

Method 3 requires no such knowledge; so it can be used by a wider circle of users. Nevertheless, like Method 2, it does necessitate the possession of a re-compilable version of PHOENICS, and of a compiler, both of which entail extra expense.

This is why In-Form has been introduced. It provides:

- the unlimited extensibility of simulation power which GROUND and PLANT were designed to supply; but
- it does so to the standard executable, and so necessitates neither new coding, nor compilation, nor new executable.

1.2. The new capabilities supplied by In-Form

In-Form has thus been created in order to provide the advantages of PLANT without the disadvantages**.** Specifically, the expressions placed in the Q1 file are now transmitted directly to EARTH; and EARTH has been equipped to interpret and work with them.

Moreover, the syntax of an In-Form expression, for a given requirement, is easier to read and write than that of the corresponding PLANT one.

Further, In-Form statements can do what was never possible for PLANT, namely apply initial values, sources and other attributes including those of motion, to **Virtual-Reality objects**.

Finally, pre-In-Form PIL-using practices which were satisfactory but clumsy, for example the long-established practice of using GRNDx pointers, can be replaced and generalised by easier-to-understand In-Form equivalents.

1.3. How In-Form works

The points which all users of In-Form should understand are as follows:

a) In-Form statements placed in the Q1 file have the typical form:

   (keyword what_where_when_indicators formula optional_conditions)

   English words such as 'at', 'of', 'is' and 'with' appear as meaningful separators.

b) There are only a few keywords, of which 'initial', 'property', 'source' and 'stored' are the most commonly used. A full list if supplied in section 2.2b .

c) The what_where_when items indicate, for example, to what VR-object and to what solved-for variable, the formula is to be applied.

d) The user's main concern is therefore to understand the syntax of In-Form sufficiently for the statement both to be accepted by PHOENICS and to express his or her intentions.

Users do not need, but may be interested, to know what SATELLITE does when it has read the statement. This is as follows:

- SATELLITE first subjects the statement to some legality checks; then, if satisfied, it creates a corresponding special-data (i.e. SPEDAT) statement of character-string type; this is passed to EARTH, via EARDAT, in the usual way.
- When EARTH begins execution, its first task is to read the SPEDATs and convert them, if it can, into instructions (to itself) to perform the corresponding mathematical operations.
- If it cannot, because of some illegality in the setting which SATELLITE could not detect, it will report what error it has found.
- Unless there is an error, little of this is visible to the user. However, the SPEDATs are written into, and can be inspected in, the files Q1EAR, EARDAT and (if accepted by EARTH) in RESULT also. To look at these files may be useful if all has not proceeded expected.

2. Introduction to In-Form

2.1. The setting of density, as an example

a. The PIL method

Consider the setting of the PIL variable RHO1, the first-phase domain-fluid density.

The line in Q1

```
RHO1 = 1.189
```

sets its value to a constant, namely that of 1-atmosphere 20-degree-Celsius air.

If some other setting is needed the Encyclopaedia article on RHO1 shows how alternative settings may be made.

For example, the setting:

```
RHO1=GRND1
```

leads to four options, the selection from which is effected by ascribing values to one or more of the PIL variables: RHO1A, RHO1B, RHO1C, IBUOYA, IBUOYB etc.

Further opportunities are opened by the settings: RHO1=GRND2

```
RHO1=GRND3
```

Until

```
RHO1=GRND10
```

The choice is thus rich, if somewhat bewildering; but, inevitably, it is still limited.

The dialog boxes of the PHOENICS GUI, of course, are designed to reduce the bewilderment, by presenting the alternatives in a systematic way; but they can do nothing to overcome the limitations.

b. The In-Form alternative

The In-form equivalent of the first of the above settings is:

```
(property RHO1 is 1.189)
```

the meaning of which is obvious enough.

Let us now consider the first of the GRND1 options, which the Encyclopaedia states to be:

RHO1 = RHO1 + RHO1B * H1

where RHO1A and RHO1B are constants to be set in the Q1 file, and H1 is the first-phase enthalpy which, it is supposed, is a solved-for variable.

Let its definition be such that H1 is at 20 degrees Celsius.

This option might be selected if one were seeking to represent the dependence on temperature of the density of atmospheric pressure air, which is most naturally expressed as:

RHO1 = 1.189 * [1.0 + H1 / (1005.0 * 293.0)

wherein:

1005.0 is the constant-pressure specific heat of air, and

293.0 is the absolute temperature corresponding to 20 degrees Celsius.

Of course, RHO1A and RHO1B have to be calculated as 1.0 and 1.0/(1005.0*293.0) respectively.

The In-Form equivalent of the above setting is simply:

(PROPERTY RHO1 IS 1.189*(1.0 + H1/(1005.0*293.0)))

This is easily written and easily understood; and its writer needs to make no visit to the Encyclopaedia in order to check that the right GRND number has been chosen.

In-Form thus allows users to express their requirements in a natural manner.

Moreover, if the user, enlarging his demands, wished to take into account a quadratic dependence of density on temperature, he would not need to search the Encyclopaedia for that option (which he would not find, as it happens).

Instead he could simply extend his formula thus:

(property RHO1 is 1.189*(1.0 + H1/(1005.0*293.0) + 1.e-6*H1^2))

[Here 1.e-6 is just an example, of course; any value could be used. The ^ is the In-Form symbol for exponentiation.]

Thus the word 'alternative' in the above title is an In-Form-underestimating misnomer; for In-Form, as well as providing an alternative way of doing what PIL can do, does also much that PIL can not.

c. Answers to initial questions about In-Form

Questions already may have arisen in the reader's mind, which it is as well to articulate and answer at once. Here are some:

- Both upper- and lower-case characters have appeared in the above In-Form statements. Is this significant?

  No. The Satellite converts all characters to upper-case before processing them. Some users prefer to place the In-Form-specific words 'property' and 'is' in lower case, and the ones which they have introduced (here 'RHO1' and 'H1') in upper case. Or they may use the opposite rule; or none at all. It is their choice.

- If the formula became very long, the 68-character limit to Q1-file lines would be exceeded. What then?

  Insertion of a $ sign in or before the 68th column will be interpreted by SATELLITE as an append-next-line command; so In-Form statements consisting of two lines (but no more) are allowed. However, if the formula is so long as to exceed this limit, it is necessary to construct it out of several smaller parts.

- All constants in the above examples have appeared in numerical form. Can symbols be employed?

- Yes. For example the PIL variable RHO1A could be introduced in place of the 1.189 of the formulae, if the setting

  RHO1A=1.189 preceded the In-Form statement thus:

  (property RHO1 is RHO1A)

- We have seen a keyword (property) and some formulae; and we suppose that RHO1 is a what_where_when example. What about showing some conditions?

  Good question! The answer will be provided in section 2.2 .

- Three specifications for RHO1 have been specified. Suppose that all three were included in the Q1 file; would the information about all three be transmitted to EARTH? And which one would be used to define the density?

  The answer is that all three specifications would be transmitted to and processed by EARTH; but it is only the last to appear in the Q1 which would actually be used for the flow-simulating calculation.

- The SATELLITE ordinarily gives RHO1 the default value 1.0 before the Q1 is read; so this is presumably superseded by the In-Form specification. Suppose however that a

PIL statement appeared in the Q1 after, i.e. below, the In-Form statement(s). Would this supersede the latter?

- The answer is negative: it is in EARTH that the decision is made; and it is always the In-Form statements which chooses to follow.

  The reader may care to check these statements by running PHOENICS with the following Q1:

```
TALK=f;RUN(1,1)
store(rho1)
(property rho1 is 1.0)
(property rho1 is 2.0)
(property rho1 is 3.0)
rho1=4.0
STOP
```

The RESULT file will show RHO1=4.0 in the GROUP 9 print-out; but the field print-out will show RHO1 to be 3.0.

It will also carry the printed line: "Formula used for setting RHO1" .

d. What has appeared in Q1EAR and EARDAT

For interested readers who have read the second part of section 1.3 (How In-Form works) the consequences for Q1EAR and EARDAT of the In-form statements of sub-section b above will now be shown.

New users will find it helpful to inspect the Encyclopaedia article on SPEDAT first.

- The first setting (namely of RHO1 to 1.189) leads to:

```
 SPEDAT(SET,PROPERTY,RHO1,C,=1.189)
```

in Q1EAR (and later in RESULT),

wherein the 'C,' is a reminder that what follows it must be treated as a character string.

The solver module, EARTH, does not read this however. It reads instead the EARDAT file, where the corresponding entry is:

```
PROPERTY  RHO1   C=1.189
```

The second setting (of RHO1 as linear in H1) places:

```
SPEDAT(SET,PROPERTY,RHO1,C,=1.189*(1.0+H1/(1005.0*293.0)))
```

in Q1EAR, and

```
PROPERTY  RHO1   C=1.189*(1.0+H1/(1005.0*293.0))
```

in EARDAT.

The third setting (of RHO1 as quadratic in H1) places:

```
SPEDAT(SET,PROPERTY,RHO1,C,=1.189*(1.0+H1/(1005.0*293.0)+1.E$)
SPEDAT(SET,PROPERTY,RHO1,C,-6*H1^2))
```

in Q1EAR, and

```
PROPERTY  RHO1   C=1.189*(1.0+H1/(1005.0*293.0)+1.E$
PROPERTY  RHO1   C-6*H1^2)
```

in EARDAT.

The following comments are appropriate:

- The entries in SPEDAT and EARDAT are a little harder to read than the original statements, but not much.
- The line-break ($) sign is placed somewhat differently; but otherwise SPEDAT and EARDAT reproduce faithfully the character strings in the originals.
- The SATELLITE (which wrote Q1EAR and EARDAT) performed no numerical evaluations, for example of 1005.0*293.0, leaving EARTH to do so later.
- The reader is reminded that there is no necessity to inspect Q1EAR or EARDAT, which can be regarded as belonging to transactions which are internal to PHOENICS.

2.2. The In-Form statement

a. General description

As has already been stated, the general form of an In-Form statement in a Q1 file consists of four items, enclosed between () brackets:

(keyword what_where_when_indicators formula optional_conditions)

This will now be explained, under the headings:

- keywords
- pre-formula conditions, i.e. what_where_when_indicators
- the formula, containing the expression to be evaluated by the solver
- post-formula options, i.e. optional_conditions

It should be noted that:

- The opening and closing brackets of the statement must always be present.
- The opening bracket must start in the first or second column.
- Lines must not extend belong the 68th column; but a dollar sign, $, in or before the 68th column will be taken as an append-next-line instruction.
- Characters may be upper- or lower-case without consequences.
- Blank spaces separate the items, several successive spaces having the same effect as one.

See also Appendix 2 for a succinct summary of In-Form-statement features, in which it will be seen (from the appearance of [] brackets, which items may be dispensed with).

b. Keywords

The keywords used in In-Form statements are:

- **PROPERTY**: This has already been encountered in section 2.1. It will be explained in connexion with all material properties used by PHOENICS in section 4.
- **STORED**: This keyword is used for the creation of auxiliary variables which can have distinct values for each cell in the domain.

  Such variables are of two main kinds, namely:

  - those which are used only for post-processing purposes, as when exact solutions are computed for comparison with the numerically-derived values: and
  - those which require to be updated continually during the computations because their values influence how the computation proceeds.

  These will be discussed in section 5.

- **MAKE**: This keyword is also concerned with the creation of auxiliary variables but of lesser dimensionality. It is also discussed in section 5.
- **STORE1**: This keyword concerns what is to be done with single auxiliary variables introduced by MAKE.
- **INITIA**L: This is concerned with the setting of initial values of solved-for or auxiliary values. Initial values can be set for:
    - the whole domain,
    - for limited volumes defined by patches (i.e. those of which the 'bounding boxes' are aligned with the computational grid)
    - for limited volumes defined by facetted objects (i.e. those of which the 'bounding boxes' are not wholly aligned with the computational grid)
    - for limited volumes defined by 'In-Form objects' the shapes and positions of which are defined by formulae.

  This keyword is discussed in section 6.
- **INFOB**: As its name suggests, this keyword indicates that the position and shape of an 'In-Form object' are to be defined. It also is described in section 6.
- **SOURCE**: This extremely important keyword, the subject of section 7, is used for introducing formulae defining the sources of mass, momentum, energy and other conserved properties.

  In accordance with the usual PHOENICS practices, boundary conditions are also introduced by its means.

  Sources may, like initial values, be applied to the whole domain or to variously-defined limited spaces.
- **MOVOB:** This keyword represents In-Form's contribution to MOFOR, the moving-frame-of-reference feature of PHOENICS.

  As originally introduced, the motion of objects had to be specified by way of a .bvh (later .mof) file. Whereas this was necessary and convenient for complex motions of articulated objects such as the 'dancing man', it was over-complicated for others, for which formulae suffice.

  The MOVOB keyword, discussed in section 8.1, makes this possible, indeed easy.
- **TGRID, XGRID, YGRID, ZGRID**: These four keywords, concerned with the specification of time intervals for transient calculations, and of longitudinal-distance interval and lateral grid expansion or contraction in steady parabolic flows, are discussed together in section 3.
- **PRINT** and **LONGNAME**: In-Form has greatly extended the ability of the user to print useful and easily-understandable information, whether to the RESULT file or to a special one called INFOROUT. The relevant keywords are explained in section 9.
- **READREALS**: This keyword provides the opportunity to convey to EARTH more information than SPEDAT can. It is described in section 10.

c. Pre-formula conditions; 'var', 'at'

'var', the 'what' indicator

In the statements for density in section 2.1a, the words between the keyword and the formula were 'RHO1 is'.

Here RHO1 is of the 'what_where_when' kind, and specifically 'what': it says: 'this formula applies to the variable with name RHO1' and, by implication, to no other. It can thus be regarded as a pre-formula condition.

As it happens the words: 'of RHO1 is', or 'var RHO1 is' would have had the same effect; but, since the 'of' and 'var' add little to the understandability of the statement, they are best omitted.

The word 'is' is a signal that the formula follows. It must not be omitted.

'at', the 'where_when' indicator applied to a patch

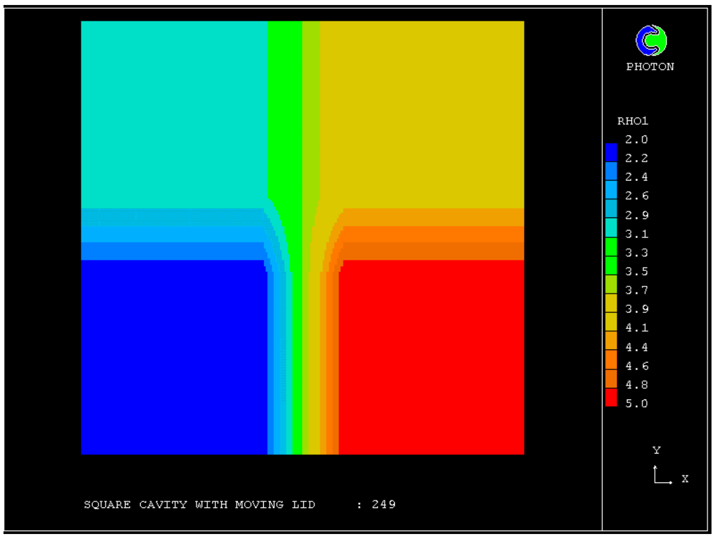
The applicability of formulae can be limited in space and time, by use of the word 'at' followed by the name of a patch or object, the names of which must appear in the Q1 file.

An example of a Q1 file which defines density differently for each of four different patches (SW, NW, NE, SW) now follows.

```
#249
   case 249 (square cavity with moving lid) has now been loaded
 STORE(RHO1)  ! in order that PHOTON will be able to display it
  ! now declare the patches to which 'at' can refer.
 PATCH(SW,CELL,1,NX/2,1,NY/2,1,1,1,1)   ! south-west quadrant
 (property RHO1 at SW is 2.0)

 PATCH(NW,CELL,1,NX/2,NY/2+1,NY,1,1,1,1)! north-west quadrant
 (property RHO1 at NW is 3.0)

 PATCH(NE,CELL,NX/2+1,NX,NY/2+1,NY,1,1,1,1) ! north-east quadrant
 (property RHO1 at NE is 4.0)

 PATCH(SE,CELL,NX/2+1,NX,1,NY/2,1,1,1,1)! south-east quadrant
 (property RHO1 at SE is 5.0)
```

Here, the 'where' is specified by the IXF, IXL, IYF etc arguments of the patch commands; and the 'when' by the final '1,1's which, since the case is a steady-state one, signify 'at all times.'

The result is as shown (with the smearing engendered by the very-coarse grid) in the following PHOTON plot.



Readers interested in the transmission details may like to know that the relevant EARDAT contains the lines:

```
PROPERTY  RHO1!SWC=2.0
PROPERTY  RHO1!NWC=3.0
PROPERTY  RHO1!NEC=4.0
PROPERTY  RHO1!SEC=5.0
```

wherein it will be noted that the exclamation mark, !, replaces 'at'; and

the corresponding lines in the Q1EAR and RESULT files are:

```
SPEDAT(SET,PROPERTY,RHO1!SW,C,=2.0)
SPEDAT(SET,PROPERTY,RHO1!NW,C,=3.0)
SPEDAT(SET,PROPERTY,RHO1!NE,C,=4.0)
SPEDAT(SET,PROPERTY,RHO1!SE,C,=5.0)
```

The same with more complex formulae

It might be argued that the same effect could have been arrived at without the use of In-Form; this is true; for RHO1 could have been set by means of four INIVAL-type patches.

However, INIVAL patches allow only uniform (or linear in x or y or z, via LINVLX, LINVLY and LINVLZ) values to be set, whereas In-Form can set any non-uniform ones which can be described by formulae, for example:

```
(property rho1 at sw is 100*(xg + yg))
(property rho1 at nw is 1.0 + 1.e-1*(u1+v1))
(property rho1 at ne is 10.0 + 0.0001*p1/h1)
(property rho1 at se is 5.0+0.1*((xg)^2 + (yg)^2))
```

from which it can be deduced that the formulae may involve other 3D-stored variables (u1, v1, p1, h1) and also geometrical quantities (xg, yg) .

[It should be noted, however, that the ability to set density freely in this manner does not signify that the setting makes physical sense, or that a converged solution will be obtained.]

A 'when' example

Core library case 756 simulates (rather crudely) the motion of a paddle in a chemical reactor. It does so by shifting the PRPS distribution be reference to different patches at different times.

In association with In-Form SOURCE settings which depend upon the PRPS value, to be described below, the fluid is set in motion as seen here.

'at', the 'where_when' indicator, applied to a faceted object

The name of an object may be substituted for the name of a patch, with the difference that the formula is then operative only in that part of the bounding box of the object which lies within the facet-described outer surfaces of the object.

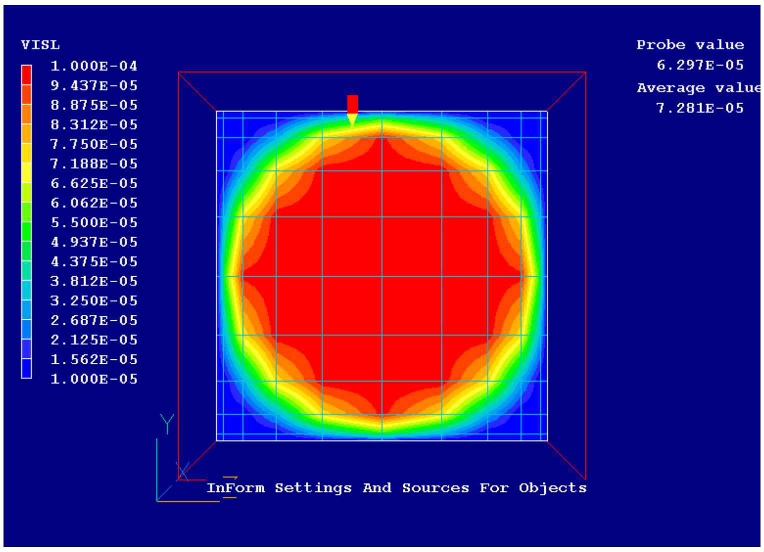
Input-Library case 764 shows how the viscosity can be set at different values according to whether the cell in question lies inside or outside a sphere defined by way of facets.

The sphere object is of course that defined by the line:

> OBJ, NAME, SPHERE

near the bottom of the Q1 file.

The resulting viscosity distribution is here shown in a VR-Viewer contour plot.



Further scrutiny of the Q1 file for case 764 reveals the 'at' condition in use also for statements with the keywords PROPERTY, INITIAL, STORED and SOURCE, in each case showing that the formula is to be applied to the sphere. Evidently it is very versatile.

A question of order

The alert reader may have noticed that, for the square-cavity case, the In-Form statement containing 'at SW' was placed below the PATCH(SW,.....) declaration, and that the same order (PATCH first, at later) was adopted for the three further statements.

This appeared natural, and conducive to orderly thought; but was it necessary?

In case 764, the lines with 'at sphere' all appear above the line which defines the sphere object. This suggests that the patch-first-at-later principle may not a necessity; and the suggestion can be proved to be correct by running the following Q1:

#249

```
   case 249 (square cavity with moving lid) has now been loaded
 STORE(RHO1)  ! in order that PHOTON will be able to display it
  ! now declare the patches to which 'at' can refer.
 (property RHO1 at SW is 2.0)
 (property RHO1 at NW is 3.0)
 (property RHO1 at NE is 4.0)
 (property RHO1 at SE is 5.0)
 PATCH(SW,CELL,1,NX/2,1,NY/2,1,1,1,1)   ! south-west quadrant
 PATCH(NW,CELL,1,NX/2,NY/2+1,NY,1,1,1,1)! north-west quadrant
 PATCH(NE,CELL,NX/2+1,NX,NY/2+1,NY,1,1,1,1) ! north-east quadrant
 PATCH(SE,CELL,NX/2+1,NX,1,NY/2,1,1,1,1)! south-east quadrant
```

which will be found to give the same results as before.

The patch-first-at-later prescription is a good rule to follow; but it is not required by In-Form.

d. The formula

Suppose that one wished to specify the density as proportional to the inverse square of the distance from the origin of the coordinate system.

Then any algebraically-literate person would recognise this as being expressed symbolically, with x, y and z as the cartesian coordinates, by:

density = 1 / (x**2 + y**2 + z**2)**0.5

with ** signifying exponentiation.

This is (almost) exactly how it is expressed in an In-Form statement, which might be: (property RHO1 is 1/(XG^2 + YG^2 + ZG^2)^0.5)

wherein xg, yg and zg denote the cartesian coordinates of the cell centres, and ^ replaces ** as the exponentiation sign.

If the distances were to be measured not from the origin but from some point with coordinates x0,y0,z0, these values would need to be declared in the Q1 file, and given some values, for example by:

REAL(X0,Y0,Z0)

X0=1.2; Y0=2.2; Z0=3.3

Then the formula could be expressed as:

1/((XG-X0)^2 + (YG-Y0)^2 + (ZG-Z0)^2)^0.5

This is still easy to read; but, if the pre-formula and post-formula options were at all extensive, the In-Form statement would have more than one line; then ease of reading would be diminished.

It is therefore useful to recognise that PIL is flexible enough to allow the formula to be built up in stages, each one being brief enough be easily understood.

For example, one might declare and define the character variables XDSQ, YDSQ and ZDSQ thus:

```
CHAR(XDSG,YDSQ,ZDSQ)XDSQ=(XG-X0)^2
YDSQ=(YG-Y0)^2
ZDSQ=(ZG-Z0)^2
```

Then the formula could be abbreviated to:

1/(:XDSQ: + :YDSQ: + :ZDSQ:)^0.5

wherein the colons signify that, because XDSQ etc are character variables, they must be evaluated as soon as read by SATELLITE.

Finally, one might decide to declare and define the character variable FORM thus: CHAR(FORM)

FORM=1/(:XDSQ: + :YDSQ: + :ZDSQ:)^0.5

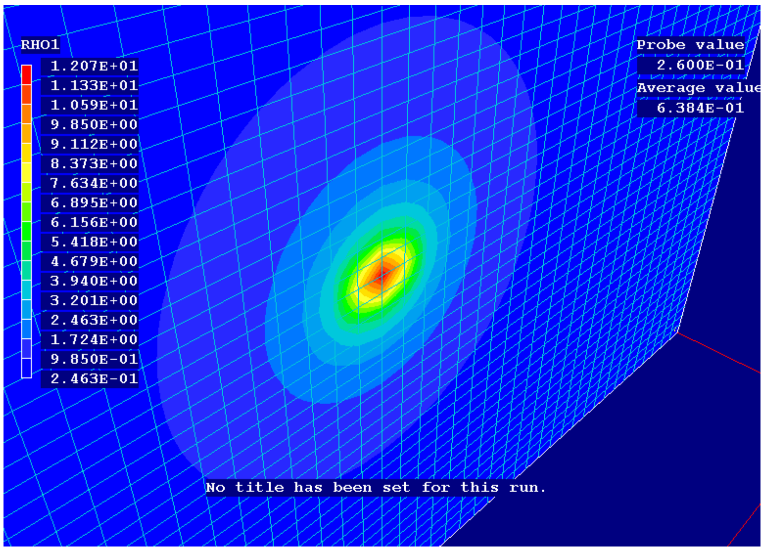
whereupon the whole In-Form statement would be reduced to: (property RHO1 is :FORM:)

The equivalence of all these statements can be tested by running PHOENICS from the following Q1 file, which contains all four of the above specifications:

```
talk=t;run(1,1)
store(rho1)
nx=50;ny=50;nz=50
xulast=2.4;yvlast=4.4;zwlast=6.6
#unigrid
  inform9begin
REAL(X0,Y0,Z0)
X0=1.2; Y0=2.2; Z0=3.3
  Case 1
(property of RHO1 is 1/((XG-X0)^2 + (YG-Y0)^2 + (ZG-Z0)^2)^0.5)
  Case 2
```

```
CHAR(XDSQ,YDSQ,ZDSQ)
XDSQ=(XG-X0)^2
YDSQ=(YG-Y0)^2
ZDSQ=(ZG-Z0)^2
(property of RHO1 is 1/(:XDSQ: + :YDSQ: + :ZDSQ:)^0.50
  Case 3
CHAR(FORM)
FORM=1/(:XDSQ: + :YDSQ: + :ZDSQ:)^0.5 ! colons are needed here too
  FORM=1/(XDSQ + YDSQ + ZDSQ)^0.5 ! This would be incorrect
(property of RHO1 is :FORM:)
  inform9end
```

If Group 19 of the RESULT file is examined, it will be seen that all four of the formulae have been converted into identical SPEDAT lines; and the VR-Viewer plot shown here indicates (qualitatively) that the resulting density field is as expected.



A comprehensive discussion of the many types of formulae allowed by In-Form is given in section 2.3 below. The present preliminary discussion, now concluded, has been inserted so as to make clear that even the most complex formulae, of which several will be shown below, can be broken into more easily digested fragments.

e. Post-formula options; 'with', '!', 'if' and 'infob'

There are numerous post-formula options; but many of them apply only to particular keywords, as is shown in Appendix 2.

Keyword-specific options will be dealt with keyword-by-keyword in section 4. However there are three which apply to many keywords, being characterised by IMAT, IF and INFOB.

These will be dealt with, in order, here.

Limitation to a particular material by use of 'with IMAT' etc

- It is possible to limit the action of a formula to those locations at which the material-property index, IMAT (alias PRPS), has a specific value.
- Core-library case 756, for example, uses this technique so as to apply a momentum source only to those cells, within the space occupied by the associated patch, in which the material-designating property PRPS (alias IMAT), is in excess of a prescribed value (namely 100).

The flow is transient; so this is a 'when' as well as a 'where' condition.

'with IMAT>=100.0' is what appears in the In-Form statement in question.

- The complete set of 'with IMAT' conditions allowable in In-Form statements is:

  'with IMAT>value'

  means 'for PRPS greater than value'

  'with IMAT<value'

  means 'for PRPS less than value'

  'with IMAT>=value'

  means 'for PRPS greater than or equal to value'

  'with IMAT<=value'

  means 'for PRPS less than or equal to value'

  'with IMAT=value'

  means 'for PRPS equal to value'

  'with IMAT!=value'

  means 'for PRPS not equal to value'

EARDAT manifestations; the use of '!'

Inspection of the corresponding entries written into EARDAT, when the SATELLITE has finished reading the Q1 of case 756, reveals that the counterparts of the In-Form statements just inspected, namely:

```
(SOURCE of U1 at I is 1.E5*(VEL*(YIC-YG)-U1) with IMAT>=100!LINE)
(SOURCE of V1 at I is 1.E5*(VEL*(XG-XIC)-V1) with IMAT>=100!LINE)
```

are (with $ and blanks removed for ease of reading):

```
 SOURCE U1!I  C=1.E5*(7.8540E-01*(10-YG)-U1)!IMAT>=100!LINE
  SOURCE V1!I  C=1.E5*(7.8540E-01*(XG-10)-V1)!IMAT>=100!LINE
```

Evidently the ' with ' has been replaced by an exclamation mark.

Consider now the '!LINE' which follows 100 in both the In-Form statements and their EARDAT counterparts. The following needs to be noted:

a) LINE is a further post-formula option, applicable only to the 'source' keyword;
b) its meaning is 'linearize the source', i.e. express it as:

   constant1*(constant2 - the as-yet-unknown value of the dependent variable in the cell in question);
c) despite the apparent equivalence of ' with ' and '!', substitution of the former for the latter in this In-Form statement is not allowed;
d) it can be concluded that, although several post-formula options can be applied simultaneously, only the first can be preceded by a ' with ', and the following ones must be separated by exclamation marks.

Other variables than PRPS can be used in a similar way. Thus, it is possible to:

- define a new whole-field-stored "MARK" variable;
- use In-Form to ascribe values for it over the whole field;

- ascribe to density one value where (and when) the marker exceeds a prescribed value, and another when it falls below the prescribed value.

IF

Another generally-applicable post-formula option is the "IF(condition)" construct, wherein *condition* can be a Fortran-like expression, as exemplified in library cases:

- 776, in connection with the ZGRID keyword;
- 777, in connection with the YGRID keyword;
- 778, in connection with the STORE1 keyword;
- 779, in connection with the SOURCE keyword;
- 781, in connection with the STORED keyword;
- 783, in connection with the several keywords;
- 785, in conjunction with INFOB; and
- 786, in connection with the SOURCE keyword;

It should be noted that either ' with ' or '!' must always precede the 'IF'.

INFOB_N

In-Form objects, created by the INFOB keyword, will not be discussed systematically until section 6. Here it suffices to say that their existence and location are indicated by the value of a 'marker variable' (usually MARK) in the cells which they occupy.

This value is usually 1, 2, 3 or other integer. Then the post-formula option 'with INFOB_2', say, signifies that the formula is to be applied to cells where MARK has the value 2 (strictly 2.0, because 'real' values are in question).

Library case 768 illustrates its use.

Specifically,

- 'with INFOB_1' will be seen to modify INITIAL of MARK;
- 'with INFOB_1' will be seen to modify a SOURCE of TEM1;
- 'with LINE!INFOB_1' will be seen to modify a SOURCE of U1.

It may be asked why, since this condition is of the 'what_where_when' variety, it is not among the 'pre-formula' options. The answer is that 'at' has already been used for a different purpose; for an In-Form object itself requires to be 'localised', which is why 'at PATCH1' appears in each of the In-Form statements in question.

2.3. Allowable formulae

a. General description

The formulae employed by In-Form, whether for setting properties, initial values, sources or anything else, are arrangements of **operators, functions** and **operands** which conform to rules which are similar to those of algebra and/or Fortran.

No significance attaches to whether upper- or lower-case characters are used.

b. Operators

The operators which may be used are:

- **= + - * / ( )** , all of which have their usual significances; and
- **^**, which represents exponentiation, thereby replacing the ****** of Fortran;

c. Functions

The functions, listed in alphabetical order in Appendix 1, are as follows:

- **conventional mathematical functions**:
    - o *ABS ACOS ASIN ATAN COS EXP MAX MIN SIN SQRT TAN*, which have their Fortran significances, except that it should be noted that:
        - § *MAX* and *MIN* operate on real values, thereby replacing the AMAX1 and AMIN1 of Fortran ;
    - o *LOGE* which stands for the natural logarithm, with base e;
    - o *LOG10* which stands for the Napierian logarithm, with base 10.0 .
- **formula-name functions**:
    - o *POL2 POL3 POL4 POL5 POL6*, which signify that a polynomial of the appropriate order is to be used:
    - o *PWL3*, which signifies a piece-wise-linear function, with three parts;
    - o *SPL5*, which signifies a cubic-interpolation spline function passing through five points; and
    - o *PWLF*, which signifies a piece-wise linear function of which the defining points are specified in a file;
- **neighbour-location functions**:
    - o *EAST, WEST, NORTH, SOUTH, HIGH, LOW* and *OLD*, which have meanings which are conventional in PHOENICS.
- **special functions**:
    - o *BOX* and *SPHERE*, which can be used for making geometrical shapes (more numerous than the names suggest); They will be discussed and exemplified in section 6.4.
    - o *SUM* and *SSUM*, which perform special summation operations, explained in Appendix 1. Examples of the use of SUM may be found in cases 362 345 781 785 and 786.

    The use of SSUM is exemplified in 363.

Syntax of POLx, PWL3, SPL5, PWLF

- **POLx** is followed by a bracket, then the name of the variable in question, then a comma or ampersand (i.e. , or &) followed by the requisite number, separated by further commas (or ampersands).

    Note: When written in the Q1EAR, EARDAT and RESULT files, only ampersands appear; but commas are preferable in Q1 files, because they are easier to read.

    Thus:

    POL2(x , a0 , a1 , a2)

        signifies a0+x*(a1+x*a2)

    POL3(x , a0 , a1 , a2 , a3)

        signifies a0+x*(a1+x*(a2+x*a3))

    POL4(x , a0 , a1 , a2 , a3 , a4)

        signifies a0+x*(a1+x*(a2+x*(a3+x*a4)))

    POL5(x , a0 , a1 , a2 , a3 , a4 , a5)

        signifies a0+x*(a1+x*(a2+x*(a3+x*(a4+x*a5))))

POL6(x , a0 , a1 , a2 , a3 , a4 , a5 , a6)

signifies a0+x*(a1+x*(a2+x*(a3+x*(a4+x*(a5+x*a6)))))

where:

- o x may be a constant or a variable, but
- o a0, a1, a2, etc must be constants.

Examples of use of polynomial functions are to be found in library cases: 701 for POL3; 345 for POL4; 089 for POL6, whereby it may be remarked that the practice of case 345, in which the coefficients are declared and allocated before being placed as symbols in the In-Form statement, is preferable from the view-point of clarity.

- The syntax of **PWL3** is similarly constructed.

  Thus, PWL3 is followed by a bracket, then the name of the variable in question, followed by a comma or an ampersand.

  Thereafter follow pairs of numbers, of which the first is the abscissa and the second is the corresponding ordinate, thus:

  PWL3(x , x0 , y0 , x1 , y1 , x2 , y2 , x3 , y3)

  represents y as a function of x consisting of straight lines which pass through (x0,y0), (x1,y1),(x2,y3),(x3,y3)

  Library case 763 illustrates the use of this function. It enables different formulae for setting each of four fluid properties to be compared.

  It calculates the four relevant properties of the fluid (ethylene glycol) in four different ways, namely by way of:

  - o the polynomial formulae in macro 089,
  - o three-part patchwise linear formulae,
  - o five-point cubic-spline formulae, and
  - o multi-part piece-wise linear formulae.

- The syntax of **SPL5** is similarly constructed.

  Thus, SPL5 is followed by a bracket, then the name of the variable in question, followed by a comma or an ampersand.

  Thereafter follow pairs of numbers, of which the first is the abscissa and the second is the corresponding ordinate, thus:

  SPL5(x , x0 , y0 , x1 , y1 , x2 , y2 , x3 , y3 , x4 , y4 , x5 , y5)

  represents y as a spline function of x which passes through (x0,y0), (x1,y1), (x2,y2), (x3,y3), (x4,y4), (x5,y5)

  Library case 763 illustrates the use of this function also.

- PWLF is followed by;

  - o an opening bracket,
  - o the name of the file (with full path, if it is not in the working directory).
  - o a comma,
  - o the name of the independent variable, and
  - o a closing bracket.

  Examples of the use of PWLF can be found in library case 763, in which the property values at a range of values are read from the files:

- o denprp for density
- o enuprp for kinematic viscosity
- o cpprp for specific heat, and
- o cndprp for thermal conductivity.

Syntax of SUM

SUM is followed by (*formula*) with *condition*. Examples of its use may be found in cases 785. and 786.

d. Operands

1. The NAME of any SOLVEd or STOREd variable, e.g. P1, H2, KE, ENUL or RHO1, may be an operand.

   If no square brackets follow the name of the variable, the value prevailing at the current cell is intended. However neighbour-cell or more-distant-cell values may be indicated in accordance with the following convention, using:

   - o either, fixed indices, whereby:
     - § PHI[1&2&3] is the value of the dependent variable PHI at the cell IX = 1; IY = 2 and IZ = 3.
     - § PHI[1] (or PHI[1&&]) is the value of the dependent variable at the cells of IX = 1; IY = IY and IZ = IZ.
     - § PHI[&3] (or PHI[&3&]) is the value of the dependent variable at the cells of IX = IX; IY = 3 and IZ = IZ.

     A core-library example which makes extensive use of indexing is case 759, where it is used for the creation of the pressure-gradient sources which are needed for the solution of the 'collocated-velocities' equations.

     Another is case 758, where it is used for the creation of an initial-value field on one slab by the superposition of the initial-value fields on two lower slabs.

     Another example is case 710, where indexing is used to enable the flow of tube-side fluid to be simulated for a shell-and-tube heat exchanger.

   - o or relative indices, whereby:
     - § PHI[+1] corresponds to EAST(PHI)
     - § PHI[&+1] corresponds to NORTH(PHI)
     - § PHI[&&+1] corresponds to HIGH(PHI)
     - § PHI[-1] corresponds to WEST(PHI)
     - § PHI[&-1] corresponds to SOUTH(PHI)
     - § PHI[&&-1] corresponds to LOW(PHI)

   Relative indices are used for a similar purpose in case 368, as a substitute for the 'neighbour-patch' technique which had been used in an earlier PLANT-based solution of the same problem.

2. Any of the following variables pertaining to a cartesian or cylindrical-polar grid may be an operand:

   NX, NY, NZ

   the number of intervals in the space directions

   XG, YG, ZG

   the coordinates of the centre point of a grid cell

XU, YV, ZW

the distances from the origin of the corresponding velocity-storage points

ZZW, ZWADD

quantities used in z-wise grid-expansion formulae

RG and RV

(for polar-coordinate grids) the radii of the cell centre and v-velocity node respectively

DXG, DYG, DZG

the distances between the cell centres in the three coordinate directions

DXU, DVY, DZW

the corresponding distances between the velocity nodes; and

VOL

the volume of the cell

AEAST, ANORTH, AHIGH

the easy, north and high areas of a cell

XULAST, YVLAST, ZWLAST

the total extents of the domain in the three coordinate directions

IXF, IXL, IYF, IYL, IZF, IZL

The first and last cells of a patch in the x- and y-directions

RINNER

The inner radius of a polar-coordinate grid velocity-storage points

3. Further permissible operands include the following variables pertaining to an expanding or contracting grid, of the kind sometimes used for parabolic flows:

IZ, IZSTEP

both of which represent the z-wise cell index number

XRAT, YRAT

the x- and y-wise expansion ratios

DZRAT

the corresponding z-step expansion ratio

DZ, DZL, DZGL

its previous value, and the distance between the cell centres corresponding

AZDZ, AZXU, AZYV, AZRI, AZAL, AZPH

various factors relating to non-uniform parabolic grids

SNALFA

grid-edge slope in parabolic grids

4. For time-dependent flow simulations, any of the following variables may be used as operands in In-Form formulae:

TIM

the time from the start of the flow process

DT

the current time step

TFIRST, TLAST

the starting and finishing times

5. For all simulations, any of the following indices and counters may be used as operands:

   INDVAR

   the index of the currently-solved variable, eg 3 for U1 or 14 for H1

   FSWEEP, ISWEEP, LSWEEP

   the first, current and last sweep number

   FSTEP, ISTEP, LSTEP

   the first, current and last time step

   ITHYD, LITHYD

   the current hydrodynamic iteration number and its upper limit

   IRUN, NRUN

   the current run number (in a multi-run calculation) and its upper limit

6. The final set of permissible operands comprises:

   TSURR

   the temperature of the surroundings

   PBAR

   the average downstream pressure in a parabolic calculation

   MASS1

   The mass of first- (or only-) phase material in the cell

   MASS2

   the mass of second-phase material in the cell.

   TINY, GREAT

   very small and very large numbers

e. Examples

Examples of the formulae which In-Form is capable of handling can be found in the Q1 files of the Core Input-file Library by clicking here.

When the list is displayed, click on 'Q1' to see the Q1 file, or the case number to see the Q1EAR file. The former shows the In-Form statement supplied by the user; the latter shows (in Group 19), their SPEDAT equivalents.

The list displayed may not fully correspond to your installation. To make a new one, run the library-search facility of the PHOENICS Commander.

The PIL-fragment macro which forms core-library case 089 contains a particularly rich collection of formulae. It is used for setting fluid properties.

3. Grid specification

3.1. Time discretization

In-Form allows time discretization to be dictated by formulae by means of the TGRID keyword, the syntax of which is most easily learned by inspection of library case 778.

3.2. Geometric grids for parabolic flows

In-Form permits the specification of expanding and contracting grids for parabolic calculations.

The variable XRAT can be set for each z-location. An example is:

(XGRID of XRAT is 1.+:POWER:*DZ/(ZWADD+ZZW))

Likewise, YRAT can be specified, for example as in case 777 as follows:

(YGRID of YRAT is 1.+:POWER:*DZ/(ZWADD+ZZW) with IF(IZ.LE.20.OR.IZ.GT.40))

Finally, In-Form can specify the forward step, DZ, as for example in case 776, where the IF conditions is used:

(ZGRID of DZ is 0.1*YVLAST with IF(IZ.LE.2))

4. Material properties in general

4.1. The PROPERTY keyword

The keyword **property** is used in In-Form statements for specifying distributions of any of the twenty quantities listed below. These include both genuine properties, such as density, specific heat and thermal-expansion coefficient, and others, such as turbulence length scale, which PHOENICS handles in a similar way, as explained in the Encyclopaedia article.

Whenever In-Form statements holding the PROPERTY keyword are present in the Q1 file, and are accepted as legitimate by EARTH, the specified quantity is computed and used at every appropriate point in the calculation.

This is true even when the same properties are nominally being computed in accordance with PIL settings: what In-Form specifies takes precedence.

A typical In-Form statement is:

(property EL1 at PATCH_1) is YG - YG^2.0 if(YG.LT.1.0))

which should make the first-phase turbulence length scale a quadratic function of the y-distance from the wall over half the grid.

Then the following simple Q1 file:

```
TALK=T;RUN(1,1)
STORE(EL1)
PATCH(PATCH1,INIVAL,1,1,1,5,1,1,1,1)
GRDPWR(Y,10,2,1)
(property EL1 at PATCH1 is YG - YG^2.0 with if(YG.LT.1.0))
NYPRIN=1
```

would cause EARTH to place in the RESULT file the following:

```
 Flow field at ITHYD=   1, IZ=   1, ISWEEP= 1, ISTEP=1
   IY EL1
10   0.0
 9   0.0
 8   0.0
 7   0.0
 6   0.0
 5   9.000E-02
 4   2.100E-01
 3   2.500E-01
 2   2.100E-01
 1   9.000E-02
```

from which it may be seen that, for YG in excess of 1.0, i.e. IY in excess of 5, EL1 has its default value of 0.0 because the In-Form statement made no prescription.

The last sentence made reference to both IY and YG because, tiresomely in this case, PHOENICS prints the grid coordinates in a different place. However (in anticipation of the not-yet-presented discussion of the keyword STORED) this may be easily remedied.

Thus, addition of the line:

```
(stored YGG is YG)
LSWEEP=2
```

causes the RESULT file to contain:

```
IY      YGG        EL1
10   1.900E+00   0.0
 9   1.700E+00   0.0
 8   1.500E+00   0.0
 7   1.300E+00   0.0
 6   1.100E+00   0.0
 5   9.000E-01   9.000E-02
 4   7.000E-01   2.100E-01
 3   5.000E-01   2.500E-01
 2   3.000E-01   2.100E-01
 1   1.000E-01   9.000E-02
```

It is worth explaining why YGG was stored, and not YG; and also why LSWEEP was increased to 2.

The reason for the first is that, although (stored YG ...) would have been accepted, PHOENICS would then become confused between the newly-stored YG and the already-present one.

The reason for the second is that, unless the (stored YGG ...) has a special post-formula option ('with SWPSTR', to be explained in section 5), YGG will not have been computed in time for first-sweep print-out.

4.2. Individual properties

The material properties used by PHOENICS are listed below and in the file \phoenics\d_earth\d_core\gxprutil.htm as may be seen by clicking here.

RHO1 DRH1DP RHO2 DRH2DP ENUT ENUL PRNDTL PHINT TMP1 TMP2 EL1 EL2 CP1 CP2 DVO1DT DVO2DT CFIPS CMDOT CINT CVM

Each of these may be set by way of In-Form, as will now be explained and exemplified, in order of IPROP value.

First however two properties which are new to PHOENICS will be mentioned, namely ENT1 and ENT2, which will be used for representing the enthalpies of the two phases when they are not being solved for directly.

This renders more symmetrical than hitherto the alternative treatments of temperature and enthalpy.

Thus ENT1 is to H1 as TMP1 is to TEM1; and

ENT2 is to H2 as TMP2 is to TEM2.

For historical reasons, the symmetry is not perfect; for H1 and H2 are "hard-wired" to variables 14 and 15, whereas TEM1 and TEM2 take the indices which the satellite ascribes to them when it encounters them in the Q1 file. However, there would be no advantage in enforcing symmetry further.

ENT1 and ENT2 are therefore added to the list to be discussed below.

In the following discussion, examples are provided for some, but not all material properties. The principles of setting are the same for all properties; so exemplification of each one would become tedious.

- **RHO1**; IPROP=1

  Water at and around 4 degrees Celsius, its maximum-density temperature

  As an example, suppose that, in a study of ice formation, it is desired to represent accurately the variation of the density of water between 0 and 10 degrees Celsius; then one way is to read the data from a table of numbers, thus:

```
** Temperature   density
0  0.9998681
1  0.9999267
2  0.9999679
3  0.9999922
4  1.0000000
5  0.9999919
6  0.9999682
7  0.9999296
8  0.9998764
9  0.9998091
10 0.9997282
```

Then, if this table is placed in a file with the complete pathname: sub-directory/water , say, then the appropriate In-Form statement is:

(property RHO1 is PWLF(sub-directory/water),TEM1).

This practice is exemplified in the following simple Q1:

```
TALK=T;RUN(1,1)
STORE(RHO1,TEM1)
GRDPWR(Y,11,1.1,1)
PATCH(WHOLE,CELL,1,NX,1,NY,1,NZ,1,1)
(property rho1 is PWLF(\phoenics\d_earth\d_core\inplib\water),TEM1)
(initial TEM1 at whole is YG*10.0-0.5)

(stored RHM1 is RHO1 - 0.999)
LSWEEP=2
NYPRIN=1
```

which, when run by PHOENICS, leads to a RESULT file containing the following:

```
 Y      RHM1        TEM1         RHO1
11   7.282E-04   1.000E+01   9.997E-01
10   8.091E-04   9.0         9.998E-01
 9   8.764E-04   8.0         9.999E-01
 8   9.296E-04   7.0         9.999E-01
 7   9.682E-04   6.0         1.0
 6   9.919E-04   5.0         1.0
 5   1.000E-03   4.0         1.0
 4   9.922E-04   3.0         1.0
 3   9.679E-04   2.0         1.0
```

```
2   9.267E-04   1.0         9.999E-01
1   8.681E-04   0.0         9.999E-01
```

which shows not only the density RHO1 for each of the temperatures but also RHM1, i.e. RHO1 - 1.0, in order that sufficient decimal places can be read.

Comparison with the original table of data will reveals that the original accuracy has been maintained.

Although it is the PROPERTY keyword that is here in the centre of attention, the keywords INITIAL and STORED have proved to be useful, in this example, for enabling the contents of the RESULT file to be easily interpreted.

The density of disturbed salty water

An example of some interest is that to be found in the multi-fluid-turbulence library case l302 which concerns the motion of a submarine vessel through water in which the initial salt distribution varies with height. As the vessel moves, the salt distribution is changed and, with it, the density distribution.

- **DRH1DP**; IPROP=2

  This represents the differential of the logarithm of the first-phase density with respect to pressure. It can be set via In-Form to any expression which the user desires; but of course it would be perverse to use a setting which conflicted with that already made for density.

  If, for example, the density setting is:

  (property rho1 is 1.0*(p1/1.e5)^1.4) ,

  as might be appropriate to isentropically-compressed air, a suitable setting for DRH1DP would be:

  (property drh1dp is 1.4/p1) .

- **RHO2**; IPROP=3

  The second-phase density can be set in the same manner as RHO1.

- **DRH2DP**; IPROP=4

  The remarks made about DRH1DP apply here too.

- **ENUT**; IPROP=5

  The contribution made by turbulence to the effective viscosity, which is represented by this PHOENICS variable, although not strictly-speaking a property of a material, can be set by In-Form as though it were one.

  Examples are:

  - (property enut is 0.5478*0.1643*ke^2/ep)

    which would correspond to what is conventionally used in the k-epsilon model; and

  - (property enut is 0.01*wdis*w1)

    which might appear in a primitive do-it-yourself version of the LVEL model.

- **ENUL**; IPROP=6, the laminar kinematic viscosity

  Laminar viscosities of fluids vary in complex ways, for which so many formulae are to be found in the literature that it would be impossible to build them all into PHOENICS. In-Form however enables them to be introduced easily.

  Here, for example is one which has been recommended for saturated water:

  (property enul is 1.0e7*exp((1.1246-0.012557*tem1)/(1. - 72.9679e-3*tem1))

In-Form handles this without difficulty.

- **PRNDTL(varname)**; IPROP=7, the Prandtl Number, thermal conductivity or material exchange coefficient.

  The Encyclopaedia entry PRANDTL explains how this variable, and the corresponding variable for the turbulent contribution, PRT(), are used and interpreted by PHOENICS.

  PRNDTL(varname) can be set by In-Form, but not PRT(varname). The reason is that, if ever anything more complex than a constant PRT(varname) is required, it is simplest to set PRT(varname) to zero and to use In-Form's great flexibility to set a single formula which expresses both the laminar and turbulent contributions.

- **PHINT(varname);** IPROP=8 and 9, the interface value for a first- or second-phase variable.

  Interface values of temperature, when heat transfer is in question, and of concentration when mass transfer occurs, have to be expressed in many different ways, in accordance with the circumstances.

  Often it can be presumed that the two phases are in thermodynamic equilibrium at the surface.

  Examples are:

  - o Heat transfer, when there is neither mass transfer (e.g. vaporization or condensation) nor chemical reaction (e.g. catalytic oxidation of a gaseous fuel). In this case the interface temperatures of both phases are equal to the weighted (by the heat-transfer coefficients) average of the local bulk temperatures of the two phases.
  - o Vaporisation of a liquid into its own vapour. In this case the interface temperature is a function of the local pressure, often expressed by way of the *Clausius-Clapeyron* equation.
  - o Vaporisation of a liquid into a gas. In this case the gas-phase interface concentration of the vaporising material is a function of the interface temperature and the local pressure. The temperature has to be computed by way of a local energy balance in which the latent heat of vaporization plays an important part.
  - o Combustion of carbon particles in air at high temperature. In this case the concentrations of oxygen and carbon dioxide on both sides of the interface can be taken as zero; the interface temperature then depends on a heat balance involving the rates of:
    - § mass-transfer of oxidant to the surface,
    - § conductive heat transfer to the interior of the particle,
    - § convective heat transfer to the bulk of the gas,
    - § radiative heat transfer to the surroundings, and
    - § the heat generated by the chemical reaction.

  In-Form enables all such relationships to be expressed with ease.

- **TMP1**; IPROP=10, the temperature of the first phase derived from the solved-for enthalpy.

  Even though temperature differences are the driving force for heat transfer by conduction and radiation, when convection is dominant it is often convenient to employ the enthalpy as the solved-for variable, because contributions of simultaneous mass transfer may be taken easily into account.

Therefore, from its earliest days, PHOENICS has been equipped with means for deducing the fluid temperature from solved-for values of the enthalpy, H1.

The earlier means of doing so are explained by the Encyclopaedia entry for TMP1.

- **TMP2**; IPROP=11, the temperature of the second phase derived from the solved-for enthalpy.

  What has been written about TMP1 applies also to TMP2, with the obvious changes.

- **EL1**; IPROP=12, the mixing length-scale of the first-phase fluid

  The ways in which distributions of this variable can be set by way of built-in coding can be learned from the relevant Encyclopaedia entry

  Inspecting that entry will reveal that, although much is provided, that provision is still meagre in comparison with what users may reasonably require.

  For example, selecting EL1=GRND8 activates the Nikuradze formula in terms of the y-coordinate. But what if it is the x-direction which measures the distance from the wall? More GROUND coding would, prior to PLANT and In-Form, have had to be introduced.

  In-Form allows such a desire to be easily satisfied, by writing in the Q1 file an expression such as the following:

  ```
  (property el1 is :xulast:*(0.14 - 0.08*(1 -2.0*xg/$
  :xulast:)^2 - 0.06*(1 - 2*xg/:xulast:)^4 ))
  ```

  Moreover, any other desired expression can be provided, without any enquiry as to what coding has been built in.

- **EL2**; IPROP=13, the mixing length-scale of the second-phase fluid

  What has been written about EL1, applies of course also to EL2.

- **CP1**; IPROP=14, the specific heat of the first phase.

  The PHOENICS Encyclopaedia contains articles on CP1 and on specific heats at constant pressure.

  Both make reference to the fact that PHOENICS can compute temperatures in two different ways, either:

  - directly, by solving for TEM1 or TEM2, in which case the enthalpy, if it appears at all is a derived quantity; or
  - indirectly, by solving for the enthalpy, H1 or H2, in which case it is the temperature which has to be derived, as was explained in the above discussion of TMP1.

  Because of the necessity to effect these derivations swiftly, and in both directions, PHOENICS makes use of an effective-specific-heat concept, defined in a non-standard manner, which however facilitates the easy deduction of enthalpy from temperature and of temperature from enthalpy.

  The "effective" specific heat can be regarded as an average value for the range of temperature from a reference value to the prevailing temperature. Care must therefore be used, when using published data, to establish the definition of the specific heat which the publication employs.

  The specific-heat data in the library case 089 are NOT, at the time of writing, consistent with the PHOENICS definition. Translation into "PHOENICS terms" remains to be accomplished. However, the error entailed by forgoing the translation will be small in most cases.

- **CP2**; IPROP=15, the specific heat of the second phase.

  What has been written about CP1 applies also to CP2, with the obvious changes.

- **DVO1DT**; IPROP=16, the first-phase volumetric thermal-expansion coefficient.

  The definition of this quantity is contained in the PHOENICS Encyclopaedia article.

  It is like DRH1DP in that it can be obtained by differentiating the expression for density, this time with respect to temperature.

  It is like DRH1DP also in that although it can be set independently of the density, it is wise to ensure that the two settings are consistent.

  However, complete consistency is not always needed. For example, the Boussinesq approximation, which is employed in buoyant-flow simulations when the temperature variations are not large, involves using a constant value of density for most of the calculation together with a finite value of the thermal expansion coefficient.

- **DVO2DT**; IPROP=17, the second-phase volumetric thermal-expansion coefficient.

  What has been written above about DVO1DT applies equally, mutatis mutandis, to DVO2DT.

- **CFIPS**; IPROP=21, the interphase friction coefficient.

  The PHOENICS Encyclopaedia article contains extensive discussion about the nature of this variable and about the built-in options which are provided for calculating it.

  There are many of these; but it is impossible to build in every variant that users are likely to need.

  In-Form has been devised so that users' needs can be satisfied easily. For example, library case 725 contains the simple statement:

  (PROPERTY CFIPS is 500.*R2*MASS1)

  This is equivalent of one of the GRNDx options and therefore does not require In-Form. However, In-Form allows the user to write:

  (PROPERTY CFIPS is 500.*MAX(R1,R2)*MASS1)

  or

  (PROPERTY CFIPS is 500.*R2*MIN(MASS1,MASS2))

  or

  (PROPERTY CFIPS is 500.*(R1*R2)^0.5*MASS1)

  or

  (PROPERTY CFIPS is 500.*R2*MASS1*KE/EP)

  or

  anything else he can think of.

  Then PHOENICS will respond accordingly.

- **CMDOT**; IPROP=22, the interphase mass-transfer rate

  The PHOENICS Encyclopaedia article describes this variable and the built-in options which are provided for calculating it.

  It is used in conjunction with the interphase-friction coefficient, FIP, which it multiplies.

In-Form enables the user to increase the range of options indefinitely, for example by embodying the full range of formulae having the form:

FIP multiplier = ln(1 + B),

where B is the "driving force for mass transfer" described in text-books on the subject.

See, for example, *Convective Mass Transfer* by DB Spalding, McGraw Hill, New York, 1963, p 154.

Typical expressions for B include:

```
  (phiG - phiS)  where phi = any conserved property
  ------------   G denotes the bulk-phase value,
  (phiS - phiT)  S denotes the interface value
 T denotes the value in the transferred substance;

  (TemG -TemS)*CPwhere TemG and TemS the temperatures in the bulk
  ---------------of the fluid and at the interface respectively
Latent heat  CP is the specific heat of the fluid and
 phase-change (eg vaporization) occurs;
```

and, for liquid-fuel combustion:

```
  [ CP*(TemG - TemS)  +  H * MoxG/S ] / Latent heat

   where H is the heat of combustion,
   MoxG is the oxygen mass fraction in the bulk of the gas,
   and S is the stoichiometric ratio.
```

- **CINT(varname)**; IPROP=23 and 24, the phase-to-interface transfer coefficient

  The PHOENICS Encyclopaedia article contains an extensive, but far from exhaustive discussion of this variable and the built-in options.

  Experts on the subject will recognise that what is provided is far from doing justice to the scientific knowledge (and conjecture) that is likely to be needed from time to time.

  The availability of In-Form enables the needs now to be met.

- **CVM**; IPROP=25, the virtual-mass coefficient

  As the encyclopaedia article makes plain this is a matter of active research and speculation.

  Knowledge is still too uncertain for any particular formulations of the "virtual-mass effect" to have become so well-established as to deserve permanent embodiment in PHOENICS.

  Therefore In-Form provides an ideal tool for researchers who wish to experiment quickly with one idea after another.

- **ENT1**; which has no IPROP value

  This property, and the reason for its introduction, has been described above.

  An example of its use in connexion with In-Form is to be found in library case 762

- **ENT2;** which has no IPROP value

  As for ENT1.

4.3. Further examples of use of (PROPERTY ...

Core-library case 105.htm is one of the oldest in the library, having been originally provided to enable the main ingredients of CFD (time-dependence, convective flow, diffusion and sources) to be studied in a one-dimensional situation.

It has now been supplied with the In-Form equivalents of the earlier data settings, which have however been left in place in order that the advantages of the new style can be perceived.

Case 763 concerns the square-cavity flow of case 762, but calculates the four relevant properties of the fluid (ethylene glycol) in four different ways, namely by way of:

- the polynomial formulae in macro 089,
- three-part patchwise linear formulae,
- five-point cubic-spline formulae, and
- multi-part piece-wise linear formulae, of which the property values at a range of values are read from the files:
  - denprp for density
  - enuprp for kinematic viscosity
  - cpprp for specific heat, and
  - cndprp for thermal conductivity.

Since the temperature range is not very large, the agreement is close, as may be seen from this extract from the result file

Use for SCRS, the simple chemical reaction

Core-library case 492 provides a good example of how the use of In-Form simplifies the setting of properties required in combustion simulations; for it contains both the pre-In-Form and post-In-Form settings.

In the former, RHO1, TMP1 and CP1 are all set to GRNDx values; then further information is conveyed by the setting of such variables as RHO1A, TEMP2B and CPC; this is easy to do, but only when one has checked the documentation for the meaning of each variable.

No such checking is needed when the In-Form route is taken; for the statements starting '(property TMP1 is' and '(property RHO1 is' are rather easy to interpret; and indeed to write also, once the rule regarding colons has been mastered.

It is only the sections numbered 1. and 2. which concern the property settings; however the opportunity has been taken, under the heading of 'The In-Form Alternative' to illustrate uses also of:

- 'source' which has the same effect as the built-in 'eddy-break-up combustion-rate coding';
- 'stored' which enables the built-in coding for calculating the concentration variables 'prod' and 'oxid' to be dispensed with, and also enables the volumetric combustion rate and the temperature in degrees Fahrenheit to be printed out; and
- 'longname', which makes the RESULT file easier to read.

Two-phase-flow cases

In case 702, In-Form is employed for the setting of the inter-phase heat-transfer coefficient and phase surface values

In case 726, these quantities are specified indirectly, by way of a source setting.

4.4. Choosing a fluid

Although users are enabled, as just explained, to use In-Form formulae for the setting of individual fluid properties, this is by no means always convenient.

Often the user prefers to specify the fluid by name and to rely on PHOENICS to use the properties which that fluid possesses.

The **fluid_name** feature of In-Form allows this.

Library case 761 exemplifies its use, showing that it is necessary to:

1. set the character variable fluid_name equal to one of the following:
    - liquids
        - saturated_water
        - SAE_5W-30_engine_oil
        - SAE_10W-30_engine_oil
        - SAE_20W-20_engine_oil
        - Ethylene_Glycol
        - Ethylene_Glycol_50%_by_volume_aqueous_solution
        - Gasoline
        - Glycerin
        - Refrigerant-12
        - Refrigerant-134a
        - Therminol_59
        - Therminol_66
        - Dowtherm_A
        - Syltherm_800
        - FC-72
        - HFE-7100
        - Mercury
    - gases
        - Air
        - Ammonia
        - Argon
        - Carbon_Dioxide
        - Carbon_Monoxide
        - Helium
        - Hydrogen
        - Methane
        - Nitrogen
        - Oxygen
        - superheated_water_vapor_at_atmospheric_pressure
        - saturated_water_vapor
        - saturated_vapor_Refrigerant-12
        - saturated_vapor_Refrigerant-134a
2. then load library case 089 which contains formulae for the relevant properties of all the above fluids, and
3. rely on that loading to include in the Q1-file the property formulae which are appropriate.

It will be seen that the property formulae which have been provided are both numerous and complex; but the user does not need to be concerned with the complexity; for In-Form and EARTH do all that is necessary.

Of course, if the user wishes to specify fluids which are not included in case 089, he or she will have to specify what the corresponding formulae should be, using the provided examples as templates.

When, as is true of the formulae in case 089, the formulae imply that the properties depend upon temperature and pressure, the user must ensure that these quantities are appropriately calculated, by setting SOLVE(P1,TEM1) and providing appropriate initial and boundary conditions.

5. Auxiliary variables

5.1. Whole-field, for viewing or post-processing

The (stored ....) keyword of In-Form enables whole-field variables to be created in accordance formulae specified by the user.

This facility may be used for variables which the user wishes to inspect, but which play no part in the calculation. RHM1, in an example in section 4.2, was of this character.

A solid-stress example

During development work, it is often desirable to perform numerical calculations for cases of which the analytical solution is known, in order that the accuracy of the numerical solution can be checked. The computation and printing of the exact values as auxiliary variables facilitates this.

Library case s603 can serve as an example. It concerns the heating of a solid block under various conditions of constraint.

The exact solutions are provided by In-Form statements here, wherein:

- sxth, syth and szth are the theoretical stresses in the x, y and z directions;
- exth, eyth and ezth are the theoretical strains in the x, y and z directions: and
- pth is the theoretical dilatation.

Whether the computed solutions are adequate can the be deduced by inspecting the RESULT file, of which a part is shown here.

Another exact-solution check

Case 718 provides an example of an artificially-contrived three-dimensional diffusion problem which possesses an exact solution.

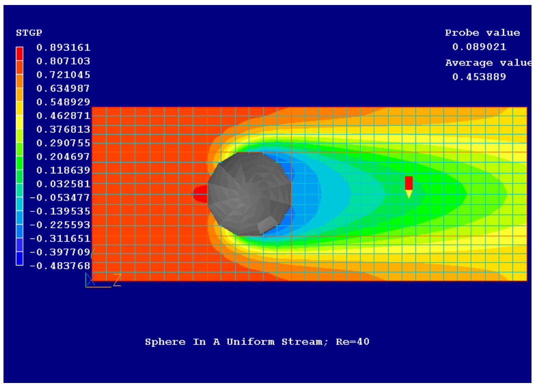
This case is, incidentally, the In-Form equivalent of a much-earlier PLANT-using example, Case z118, where the PLANT-style formula for the exact solution can be seen, for comparison with the In-Form-style one.

There is an obvious similarity; but the advantage of the In-Form approach is, of course, that no new Fortran coding is required.

Stagnation pressure

It may be interesting, for flows for which there should be large regions of uniform stagnation pressure, to check whether this expectation is confirmed by the numerical calculations. This can be done for library case 805, where the velocity squared is stored by means of STORE(VLSQ), and then STGP, i.e. P1 + 0.5*RHO1*VLSQ by means of an In-Form STORED statement.

The contour plot shown here reveals that there is indeed a substantial region of uniform stagnation pressure, despite the coarseness of the grid.



Use of LOG10 for display enhancement

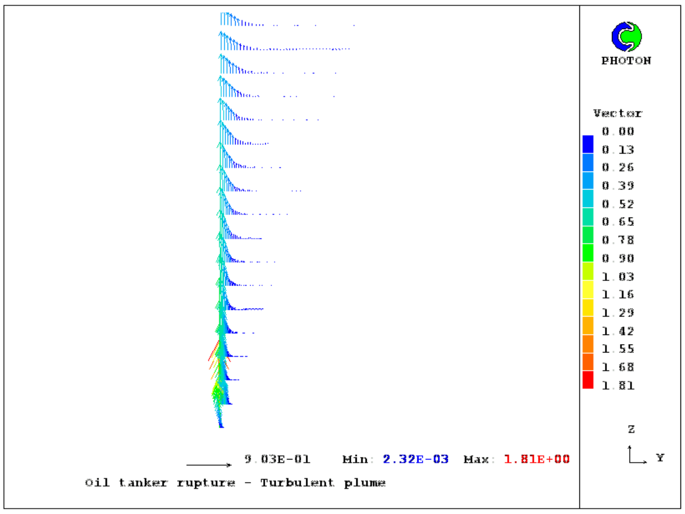
Library case 402 concerns the parabolic-mode simulation of the turbulent plume of oil-contaminated water which may rise above a leaking tanker on the ocean floor. Because of the large distances and volumes of water involved, the oil concentration varies over many orders of magnitude, which makes graphical display by way of contour plots rather difficult.

This is easily contrived by means of an In-Form stored statement,

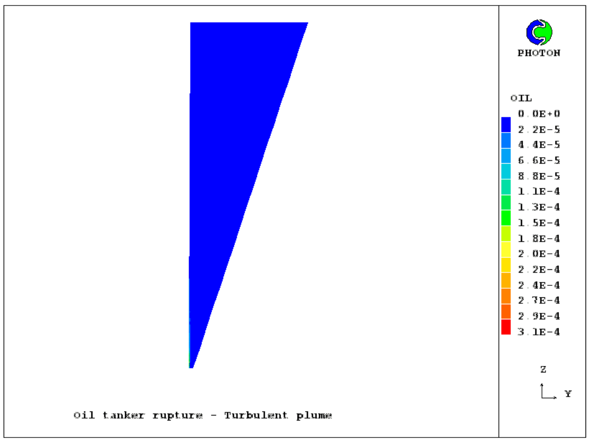
(stored var logo is log10(oil)).

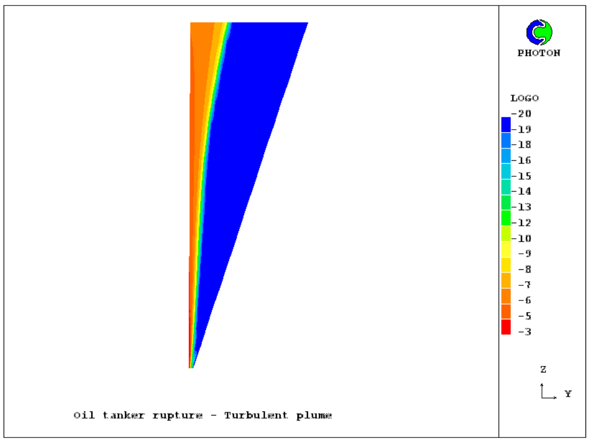
The following images show:

the shape of the plume displayed by way of velocity vectors;



the (almost invisible) contours of the oil concentration, oil ; and the (much more visible) contours of log10(oil).





If however the logarithm of the concentration has been computed, its variation, being less extreme, is easier to display.

Control of when and where the quantities are computed

The STORED-command formula can be optionally followed by 'with *condition*', where *condition* is one of:

ZSLSTR

for 'start of z-slab',

ZSLFIN

for 'finish of z-slab',

SWPSTR

for 'start of sweep',

SWPFIN

for 'finish of sweep',

TSTSTR

for 'start of time step',

TSTFIN

for 'finish of time step',

These may be understood by recalling that PHOENICS calculations are organised in nested iterative loops, whereby:

- the innermost loop visits all the cells in a "slab" of constant IZ;
- the intermediate loop "sweeps" through the whole domain, thereby visiting every slab; and
- the outermost loop, if the phenomenon simulated is time-dependent, updates variables at each time step.

These 'with' conditions act as economy devices, enabling values to be updated only when needed. If no condition is supplied, In-Form takes ZSLFIN as its default.

Inspection of the Q1 for case 805 will show that 'with SWPFIN' was used.

5.2. Whole-field, for participation in the calculation

Auxiliary variables may be stored whole-field, and computed within the main equation-solving loop, for many purposes, including those of acting as intermediaries in the calculation of sources or fluid properties.

Examples of use

Thus:

- The following lines extracted from the core-library case 701:

```
 * Temperature
(STORED of T1 is H1/CP)
 * Heat capacity
(STORED of CP is 4186.8*POL3(T1,.616,-.0040428,1.8333e-5,-2.38E-08))
```

enable the stored-only temperature, T1, to be computed from the solved-for enthalpy.

Here it should be noted that the calculation of T1 makes use of CP, and that of CP makes use of T1. Divergence is a possibility in such cases; but it does not arise in this case because the dependence of CP on T1 is small. Inspection of the Q1 file shows that the so-calculated T1 is used extensively for further property calculations.

- In case 706 the lines :

```
(STORED VAR FLIQ IS MAX(1.e-5,MIN(1.,(TEMP-:RG(1):)/(:RG(2):$
-:RG(1):))))
PATCH (iMUSHY,VOLUME,1,NX,1,NY,1,NZ,1,1)
(SOURCE OF U1 AT iMUSHY IS (0-U1)*:RG(3):*(1-FLIQ)/FLIQ)
(SOURCE OF V1 AT iMUSHY IS (0-V1)*:RG(3):*(1-FLIQ)/FLIQ)
```

enable the liquid fraction to be deduced from the temperature during the casting of a metal alloy, and thereafter the momentum sinks to be computed.

- In case 752:

```
(STORED of DM is 1.-AM-BM-CM-EM)
```

deduces the concentration of a fifth not-solved-for concentration, DM, to be computed from the solved-for values of the four other components of a mixture.

This, it should be remarked, as a display-only action; for DM appears not to be used during the computation.

5.3. Other

Other auxiliary variables which In-Form can create are:

- single real variables,
- user-dimensioned real-variable arrays, and
- real variables having values for each cell in a patch.

Syntax for other auxiliary variables: the MAKE command

The MAKE keyword is used for the allocation of memory for storage of real variables required by the user.

The complete format of the In-Form MAKE statement is:

- (MAKE dimension OF varname IS formula)

  where:

  - o 'dimension' is an integer or integer expression,
  - o 'varname' is the variable name, and
  - o formula is the mathematical expression of the initial value of the variable, the default of which is 1.e-10 .

If 'dimension' is more than 1 it should be equal to the number of cells in one slab NX*NY. Its default value is unity.

If an expression is used, it must involve only quantities such as NX, LSTEP, NTPRIN. which are known at the start of the calculation; for that is the moment at which the storage has to be allocated.

When a single real variable is to be declared, it suffices to specify only its name. Thus, the following In-Form statements are all equivalent:

- (MAKE 1 OF varname)
- (MAKE OF varname)
- (MAKE VAR varname)
- (MAKE varname)
- (MAKE 1 varname)
- (MAKE varname IS 1.E-10)
- (MAKE 1 varname IS 1.E-10)

Single variables declared in this way can subsequently be used in any In-Form statement.

Examples of the use of a single variable are:

- In library case 781, which supersedes the earlier PLANT case z616.

  (MAKE RES)

  (STORE1 of RES is SUM(VOL*(GENG-2.*:RHO1:*EPKE*G)/(NY*NZ)))

  (STORE1 of RESREF(G) is RES)
- In library case 778, which supersedes the earlier PLANT case z619.

(MAKE of SIZMIN is GREAT)

(STORE1 of SIZMIN at PATCH2 is MIN(DZW,MIN(DYG,MIN(DXG,SIZMIN))) with IF(ISWEEP.EQ.LSWEEP))

and also in library case 778

(MAKE of VELMAX is TINY)

(STORE1 of VELMAX at PATCH2 is MAX(W1,MAX(V1,MAX(U1,VELMAX))) with IF(ISWEEP.EQ.LSWEEP))

(TGRID of DT at PATCH3 is SIZMIN/VELMAX)

An example of the use of MAKE for a real array is to be found In library case 786, which supersedes the earlier PLANT case z350:

(MAKE :NX:*:NY: of SWTH is 0)

(STORE1 SWTH is OLD(TEM1[:IXP:,:IYP:,])-:TLIM: with ZSLSTR)

(SOURCE var V1 at WWALF is -:FORM:*V1 with IF(SWTH.LE.0))

Other relevant cases in the Input-File Library include:

779 and 783.

6. Initial values

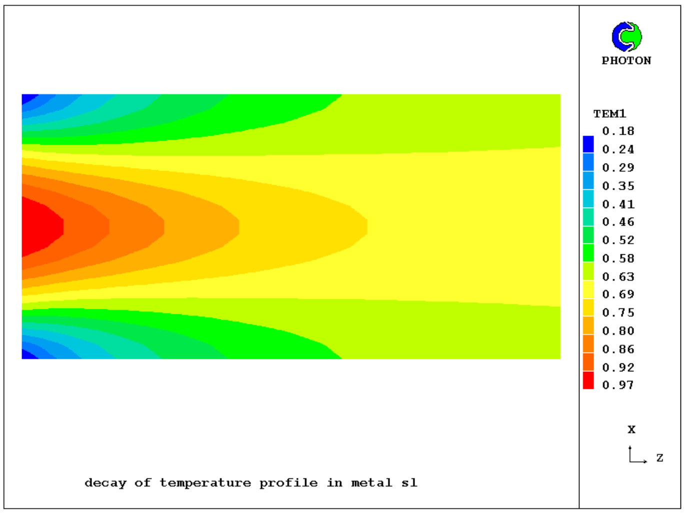
6.1. The INITIAL keyword

The (**initial** ...) keyword of In-Form is used for ascribing starting values of solved-for variables or auxiliary ones, i.e. those which have featured in a preceding SOLVE(), STORE() or SOLUTN() statement.

A **simple example** is provided by library case 707, in which a sinusoidal temperature profile reduces in amplitude while retaining its shape.

```
   ! sinusoidal temperature distribution set by In-Form
(initial of TEM1 is SIN(XG*3.14159))
```

The variation of temperature with time (plotted as the z-axis, increasing to the right) is shown by this contour plot.



In the just-shown In-Form statement, the formula for the initial profile was valid for the whole domain; there was therefore no need for an 'at' in the statement.

More often, however, different specifications of initial values are required for different parts of the domain. Localization by means of 'at' are therefore necessary.

Three distinct kinds of localization are possible:

- by reference to patches;
- by reference to VR (i.e. facetted) objects:
- by reference to 'In-Form objects' which are entities which are indeed defined by way of initial conditions.

These three topics will be discussed in turn.

6.2. Initial values for patches

The discussion will proceed by way of example:

a) **A simple example** is to be seen in library case 105 where indeed the old (PIL) and the new (In-Form) ways of doing the same thing are compared.

```
PATCH(INI1,INIVAL,3,8,1,1,1,1,1,1)
IF(INFORM) THEN
 (INITIAL OF H1 AT INI1 IS 1.0)
ELSE
 INIT(INI1,H1,0.0,1.0)
ENDIF
```

b) **Somewhat more interesting** are the following lines extracted from the core-library case 704

```
PATCH(IINIT,INIVAL,1,NX,1,NY,1,NZ,1,1)
(INITIAL of C1 at IINIT is XG+YG)
(INITIAL of EXAC at IINIT is XG+YG+:TLAST:)
```

for which no equivalent PIL statement exists. In this case, C1 is a solved-for variable; it is therefore given an initial distribution only for the first time step.

EXAC, by contrast, is an auxiliary variable which represents the exact solution of the problem; and, since its values vary with time, it has to be initialised at each time step, this being effected by the TLAST in the formula.

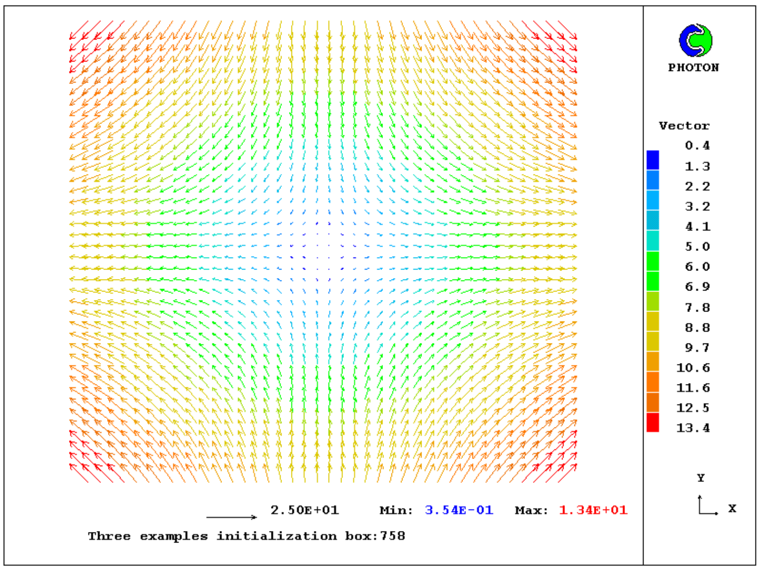
However, although 'at IINIT' does appear in the statement, examination of the arguments of the patch show that it extends over the whole domain; therefore, omitting all mention of IINIT would give the same result.

c) **A better example** is provided by case 758 which illustrates the use of the **initial** statement for the setting of three two-dimensional velocity fields, each in its own slab-shaped patch.

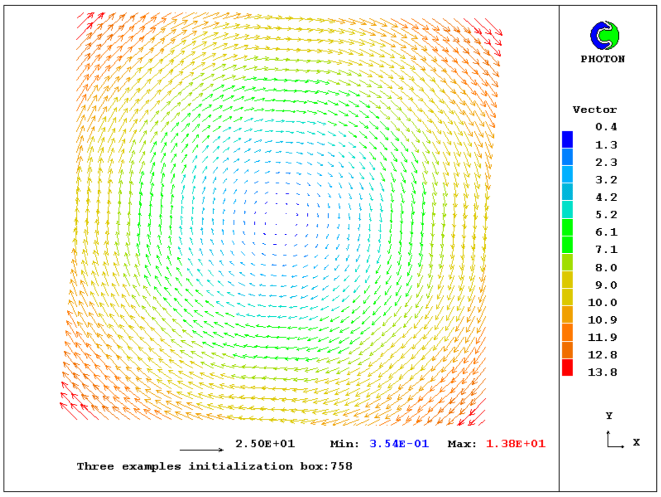
```
Example 1: Initialization of stagnation point flow
   =======================================
PATCH(INI1,INIVAL,1,NX,1,NY,1,1,1,1)
(INITIAL of U1 at INI1 is XU-10.)
(INITIAL of V1 at INI1 is -(YV-10.))

Example 2: Solid-body rotation flow
   ========================
PATCH(INI2,INIVAL,1,NX,1,NY,2,2,1,1)
(INITIAL of U1 at INI2 is YG-10.)
(INITIAL of V1 at INI2 is -(XG-10.))


Example 3:  Flow superposition
==================
PATCH(INI3,INIVAL,1,NX,1,NY,3,3,1,1)
(INITIAL of U1 at INI3 is -U1[&&-1]+U1[&&1])
(INITIAL of V1 at INI3 is -V1[&&2]+V1[&&-2])
```

The fields correspond to:

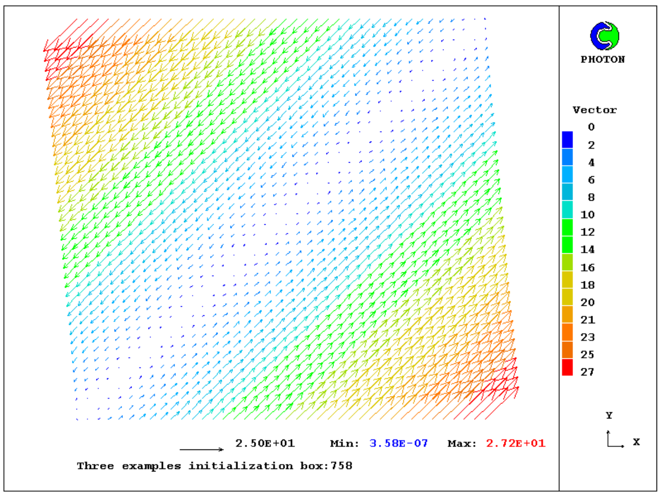
- a stagnation-point flow;



- a solid-body rotation; and



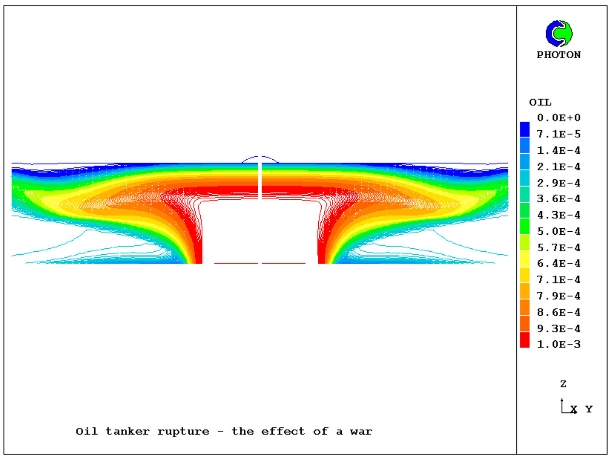
- the superposition of the second on the first.



- the first two fields have been placed on slabs IZ=1 and IZ=2, and
- the third, which to be placed on the IZ=3 slab, needs to make reference to the values which are present at the lower slabs.

d) An environmental example is provided by case 401 in which is simulated the rise of oil released from a wreck on the ocean floor. The deep water is initialised to 4 degrees Celsius and the near-water layer to 10 degrees.

As a consequence, the oil scarcely penetrates to the surface, as this contour plot shows.



6.3. Initial values for VR objects

(**initial** at OBJECT is FORMULA with OPTIONS)

is the syntax for specifying initial distributions for facetted objects, as is illustrated by case 764.

```
(INITIAL H1 at sphere is 9.9+100.0*XG with if(zg.gt..01+(zwlast/2)))
```

The object is a sphere, the material of which is the domain fluid (i.e. unusually, not a solid) and it is given a non-uniform initial enthalpy distribution by means of a formula with a 'with if' condition. This produces an initial field which is printed in the RESULT file (for iy=5) as:

```
 Field Values of H1
 IX=  10   0.0 0.0 0.0 0.0 0.0
 IX=   9   0.0 0.0 0.0 0.0 0.0
 IX=   8   0.0 1.860E+01   0.0 0.0 0.0
 IX=   7   0.0 1.740E+01   1.740E+01   0.0 0.0
 IX=   6   0.0 1.580E+01   1.580E+01   1.580E+01   0.0
 IX=   5   0.0 1.400E+01   1.400E+01   1.400E+01   0.0
 IX=   4   0.0 1.240E+01   1.240E+01   0.0 0.0
 IX=   3   0.0 1.120E+01   0.0 0.0 0.0
 IX=   2   0.0 0.0 0.0 0.0 0.0
 IX=   1   0.0 0.0 0.0 0.0 0.0
 IZ=  6   7   8   9  10
```

in which the specified variation with x and cut-off below iz=7 can be clearly seen.

It is not implied that anyone would desire such a bizarre initial enthalpy field, merely that, if anyone did, In-Form would make it possible, which PIL on its own, whether accessed via the VR-Editor or not, is unable to provide.

There is no need to present more examples of this capability of In-Form; for another of its capabilities, also connected with initial values deserves more attention.

6.4. Initial values for In-Form Objects

a. Overview

A new class of object

Since the introduction many years ago of the 'Virtual-Reality' user interface, PHOENICS has made extensive use of 'objects'. [Click here for the relevant Encyclopaedia entry]

Now In-Form allows the use of its own objects, the positions and shapes of which are marked by the cells which they occupy; and it provides formula-based means for defining those cells.

It should be recognised from the start that In-Form objects occupy cells either **wholly** or **not at all**; they cannot at present, as facet-described (i.e. 'VR') objects can, occupy only part of a cell. This limitation may be removed in the future.

A change of syntax

All that has been explained above about attachment of In-Form operations (INITIAL, SOURCE, etc) to the whole domain, to patches in space and time, and to VR-objects, applies also to In-Form objects, as will be illustrated below. However the syntax is a little different.

For example, an initial value is attached to an In-Form object by way of a statement such as:

(INITIAL of PRPS at PATCH1 is 0.0 with INFOB_1) wherein it can be seen that In-Form objects, here INFOB_1, are also associated with patches.

All this will be explained below. What must first be described however is how In-Form objects, of which the user is at present allowed up to eight in a given flow simulation, are actually created,

Object creation

In-Form objects are created by means of statements of the following format:

(INFOB at PATCHNAME is **Formula** with INFOB_**N**)

Here:

- '**INFOB**' is a keyword which indicates what is to follow;
- **'PATCHNAME**' is the name of a patch defined by a preceding (in the q1 file) PATCH statement, which defines the limits, in terms of ixf, ixl, iyf, iyl, izf, izl, itf and itl, within which the object may exist;
- **N** is the index of the object (from 1 to 8); and
- **'formula**' is a character string which defines the shape and position of the object in space and time.

The INFOB statement acts by 'tagging' the cells of which the centre lies within the object, recording at the same time which object is in question.

Additionally and optionally, if it is desired to indicate these cells via the RESULT file or PHOTON display, a 3D-stored variable (usually, but not necessarily, called 'MARK') is also given the infob index as its value.

To ensure this, the INFOB statement needs to be preceded by one such as:

(STORED of MARK at PATCH1 is 1.0 with INFOB_1)

which ensures that:

1. the variable MARK is indeed stored; and
2. that the value 1.0 is assigned to it in cells which are to be regarded as part of INFOB object number 1.

Is an INFOB PATCH a 'bounding box'

VR objects have 'bounding boxes'. These are the smallest volumes which can be defined by grid-aligned surfaces into which the object will fit.

The patch associated with an **In-Form** object, however, is usually larger than would be the bounding box of the cells comprising the objects. It is the volume within which such cells **may** exist; but it is rare for any of these cells to touch the bounding surfaces of the patch.

In one sense the INFOB PATCH is unnecessary; for, if the PATCH were always defined as the whole domain, the effects of the object would be the same. However its existence acts as an economy measure; it tells PHOENICS where searching for parts of the object would be a waste of time.

New functions

Two special functions, namely 'BOX' and 'SPHERE', are used in the formulae which follow the INFOB keyword.

These may create directly objects having the shapes suggested by their names; but, with appropriately devised arguments, they may also be used for the creation of objects of much more complex shape.

It should be noted that both these functions can be used with polar and body-fitted coordinate grids as well as cartesian ones.

b. The BOX function

The format governing the use of BOX in INFOB statements is indicated by the following example:

(INFOB at PATCH1 is BOX(arguments) with INFOB_1)

This, in conjunction with the (STORED of MARK.... statement, uses BOX(arguments) to indicate which are the cells in question.

Also permissible is the statement:

(INFOB at PATCH1 is ALL - BOX(arguments) with INFOB_1)

This indicates that the said value of MARK is to be placed at all cells in the patch **except** those within the space defined by BOX.

'ALL' can here be regarded as a function which has no arguments.

The arguments in question are as follows:

BOX(x0,y0,z0,xsize,ysize,zsize,alpha,beta,theta)

where

x0 = X-coordinate of west south low corner of box, in meters

y0 = Y-coordinate of west south low corner of box, in meters

z0 = Z-coordinate of west south low corner of box, in meters

xsize = X-size of box side, in meters

ysize = Y-size of box side, in meters

zsize = Z-size of box side, in meters

alpha = angle rotating around x axis, in radians

beta = angle rotating around y axis, in radians

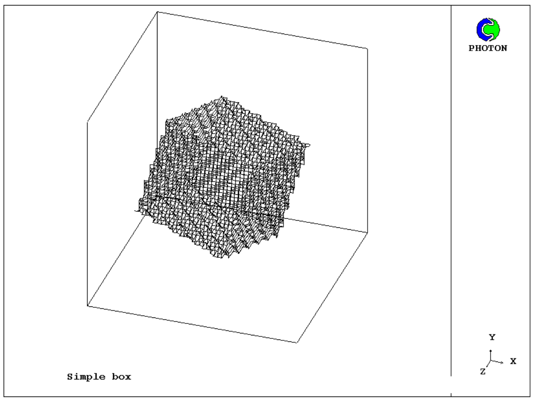
theta = angle rotating around z axis, in radians

Any one of the nine arguments can be expressed by way of a formula which may itself contain functions and constants.

Library case 383 illustrates the use of the box function for the creation of a box in a cubical domain, with all its rotations equal to 0.25 radians.

```
patch(patch1,volume,1,nx,1,ny,1,nz,1,1)
FIINIT(PRPS) =  0.000000E+00 !
  inform11begin
(stored of mark at patch1 is 1.0 with infob_1)
(initial of prps is 100 with infob_1)
real(x0,y0,z0,xs,ys,zs,al,be,th)
x0=xulast/4
y0=yvlast/4
z0=zwlast/4
xs=xulast/2
ys=yvlast/2
zs=zwlast/2
al=0.25
be=0.25
th=0.25
(infob at patch1 is box(x0,y0,z0,xs,ys,zs,al,be,th) with infob_1)
  inform11end
```

A photon plot is shown here, created by the sur mark x 0.99 and sur mark y 0.99 commands.



c. The SPHERE function

The format governing the use of SPHERE in INFOB statements is similar to that of BOX, namely:

(INFOB at PATCH1 is SPHERE(arguments) with INFOB_1)

It has fewer arguments however, namely:

SPHERE(xc, yc, zc, radius)

where

xc = x coordinate of the centre, in meters

yc = y coordinate of the centre, in meters

zc = z coordinate of the centre, in meters

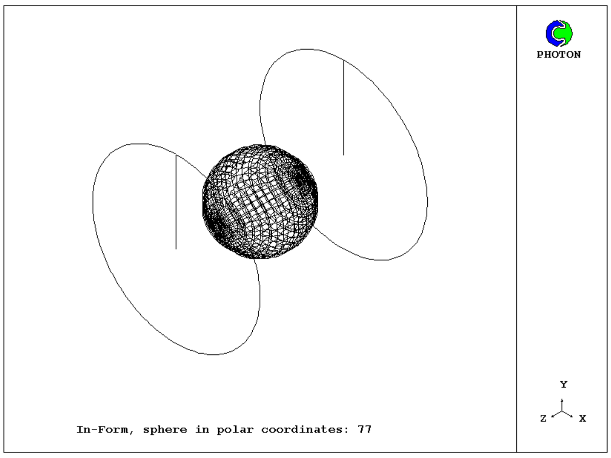
radius = radius of the sphere, in meters

'ALL - SPHERE' has the same significance as for 'BOX'.

Library case 772 creates an In-Form object, namely a sphere with its centre on the axis of a polar grid, by means of:

```
(INFOB at PATCH1 is SPHERE(10, 10, 10, 5) with INFOB_1)
```

The PHOTON plot displayed here reveals the result.



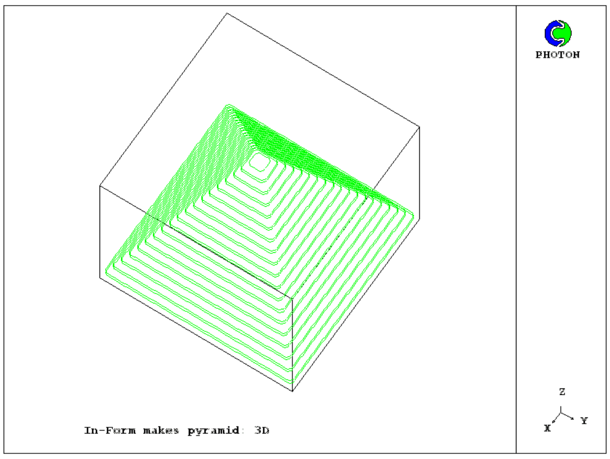
d. Simple shapes made with BOX

A pyramid

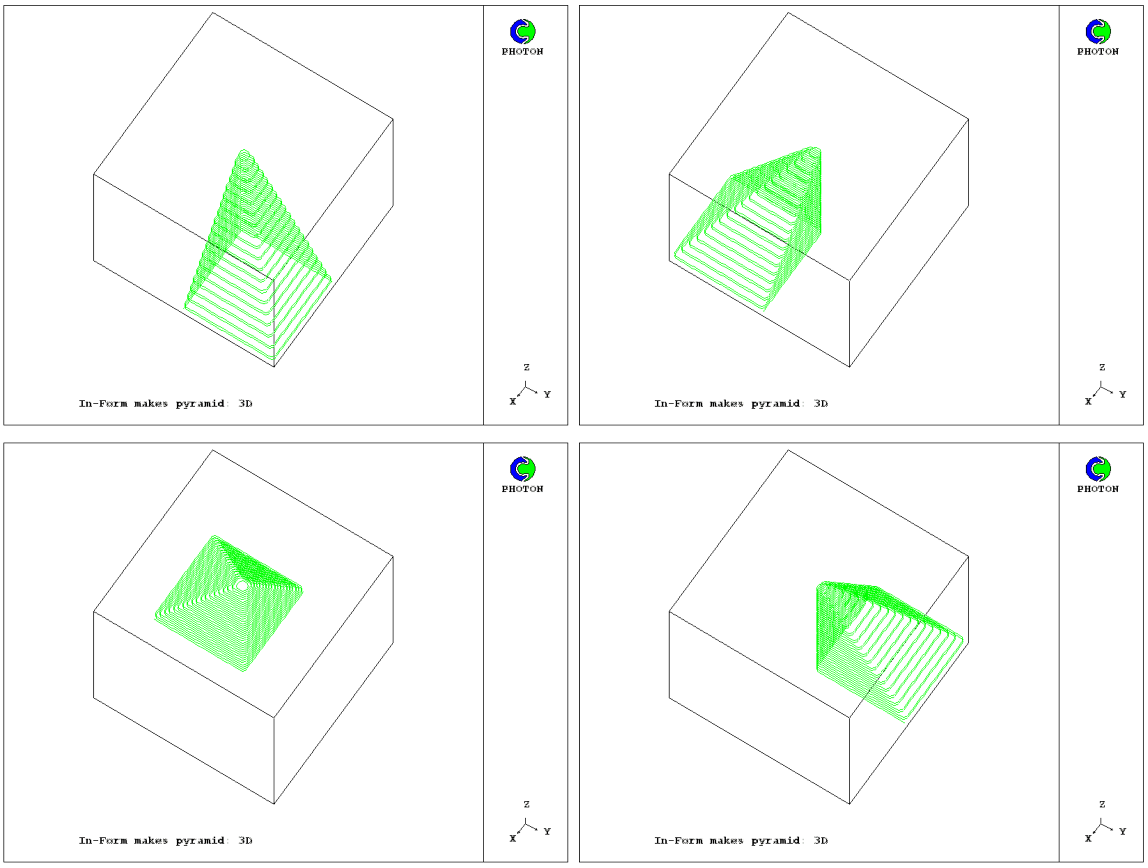
Core Library 769 shows how a pyramid can be created by a succession of In-Form statements employing the BOX function, in which the x-, y- and z-sizes vary with the value of **zg**.

The pyramid is made in four parts, each of which is rotated around the vertical edge which they share.

The following images show: the whole pyramid,



then part 1, part 2, part 3 and part 4.



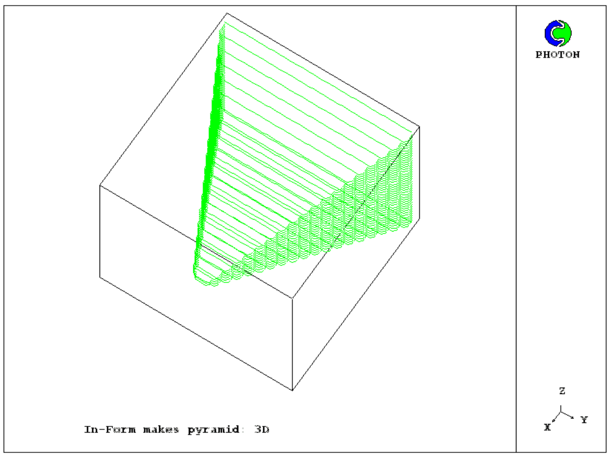
Inspection of the q1 file shows that each part of the pyramid is made by a different Infob statement, each of which however refers to the same PATCH1 and the same INFOB_1. The differences reside in their zangle arguments.

```
PATCH(PATCH1,CELL,1,NX,1,NY,1,NZ,1,LSTEP) ! patch1 occupies whole domain
  The pyramid is made in four parts, differing from each other in rotation about
   z-axis by 90 degrees i.e. PI/2.
  The xsize, ysize and zsize vary with ZG
  Part 1, zangle 0
(INFOB at PATCH1 is BOX(10,10,0,.5*(-ZG+20),.5*(-ZG+20),$
16*(-ZG+20.),0,0,0) with INFOB_1)
```

```
  Part 2, zangle PI/2
(INFOB at PATCH1 is BOX(10,10,0,.5*(-ZG+20),.5*(-ZG+20),$
16*(-ZG+20.),0,0,.5*PI) with INFOB_1)
  Part 3, zangle PI
(INFOB at PATCH1 is BOX(10,10,0,.5*(-ZG+20),.5*(-ZG+20),$
16*(-ZG+20.),0,0,PI) with INFOB_1)
  Part 4, zangle 3*PI/2
(INFOB at PATCH1 is BOX(10,10,0,.5*(-ZG+20),.5*(-ZG+20),$
16*(-ZG+20.),0,0,1.5*PI) with INFOB_1)
```

An x-direction pyramid

If a similar pyramid is required with its base on the X=0 surface rather than the Z=0 one, at it is necessary to replace **zg** by **xg** in the In-Form statements and to interchange the first, fourth and seventh arguments (which relate to x) with the third, sixth and ninth (which relate to z). Then the result is as shown here.



The pyramids have been created, as other examples will be, by making one or more arguments of the box function depend the grid-point coordinate.

The information is used at the time that each grid-point is visited at infob-scanning time; and it implies that the question "is this cell-centre in the box?" refers, in general, to a different box for each cell.

A single infob in more than one patch

It is permissible, and sometimes convenient, to associate an individual In-Form object with more than one patch, for example when different time periods are in question.

Library case 783 illustrates this, wherein PISTON1 is a patch which is active for the first time step only, while PISTON2 is used for describing the position of the same In-Form object at later times.

```
PATCH(PISTON1,CELL,1,NX,1,NY,1,NZ,1,1)
   **** Using BOX function for formation of piston geometry
(INFOB at PISTON1 is BOX(0.,0.,0.,.1,.1,Z1,0.,0.,0.$
) with INFOB_1)


PATCH(PISTON2,CELL,1,NX,1,NY,1,NZ,2,LSTEP)
(INFOB at PISTON2 is BOX(0.,0.,0.,.1,.1,ZWN,0.,0.,0.)$
 with INFOB_1)
```

Other simple shapes made by use of BOX

Library case 754 provides several examples of simple shapes made by the use of the BOX function.

e. Simple shapes made with SPHERE

Library case 749 provides several examples of simple shapes made by the use of the BOX function.

f. Some more-complex examples

Sphere connecting two tubes

Next 771 library case illustrates the creation of a Pump Chamber.

```
(INFOB at PATCH1 is SPHERE(10., 10., 10., 8.) with INFOB_1)
(INFOB at PATCH2 is SPHERE(XG, 16., 10., 2.) with INFOB_1)
(INFOB at PATCH3 is SPHERE(XG, 4., 10., 2.) with INFOB_1)
```

The first statement describes the central sphere, second - the top horizontal cylinder and third - the bottom cylinder.

Spheres inside wide channel

The creation of complex duct by means drilling of In-Form object in BFC is shown in 787 library case. At first a wide bent channel is created by means of BFC. The narrow channel is created from it by blocking any cells. Then two In-Form objects are described as spheres inside a wide channel.

```
(INFOB at PATCH1 is SPHERE(:XPP1:,:YPP1:,:ZPP1:,:RADIUS:) with INFOB_1)
(INFOB at PATCH2 is SPHERE(:XPP2:,:YPP2:,:ZPP2:,:RADIUS:) with INFOB_2)
```

Further inside each sphere values of PRPS variable are filled in with zero

```
(INITIAL of PRPS at PATCH1 is 0. with INFOB_1)
(INITIAL of PRPS at PATCH2 is 0. with INFOB_2)
```

A spiral heater

The 768 library case illustrates a method creation of a heating up spiral in a polar coordinate system.

At first it is required to create the In-Form object with the shape of a spiral with two coils. The drilling technique is applied for this purpose. The sphere with constant radius is used like as a drill bit. It will be moved on a trajectory of a spiral and to mark cells located inside this sphere. The whole set of marked cells will be one In-Form object.

The SPHERE(xce,yce,zce,radius) function creates In-Form object with the sphere shape where radius is a sphere radius (radius is equal 0.2 for this case) and xce, yce and zce are constants describing in X, Y and Z coordinates of the sphere centre in a own local cartesian coordinates system.

For cartesian examples this coordinates system coincides with coordinates system of domain. But they differ for polar cases. The local system is located so the XY plane of a polar coordinate system lies in positive part of the XY plane of the local cartesian system. Thus the origin of a local system will be always outside of the polar domain and X and Y the coordinates of the origin of the polar system will be equal YVLAST + RINNER in the local system. For this case they are equal 1. The Z coordinates of origins and the directions of Z axes of both coordinate systems coincide with one another.

If coordinates of the sphere centre will depend upon current coordinates of cells of the domain then the sphere will be moved inside the domain.

To create In-Form object with the ring shape it is enough to define X and Y coordinates of the sphere centre by the following formulas SPHERE(1+.5*sin(xg),1+.5*cos(xg), 1. , 0.2)

where 1 is the coordinates of the origin of the polar system, .5 is a radius of a circle on which the sphere will be moved and xg is current X polar coordinate of cell centre in radians. The Z coordinate will constant and be equal for example 1.

For creation of In-Form object with the spiral shape the Z coordinate of the sphere centre should depend upon cells coordinates also. Thus for creation of the spiral is offered to use following formula for the calculation of Z coordinates of the sphere centre

1+.5*xg

where 1 is Z coordinate of the start point of the spiral. The Z coordinate of the final point of the spiral will be

1+.5*XULAST=1+.5*6.28=4.13

The complete In-Form statement will look as

(INFOB at PATCH1 is SPHERE(1+.5*sin(xg),1+.5*cos(xg),1+.5*xg,.2) with INFOB_1)

Thus the spiral with one coil will be created. The additional In-Form statement is required for creation of the second spiral coil. That they were connected without a break the Z coordinate of the starting point of the second coil should be equal for the Z coordinate of the final point of the first coil. In this case Z coordinate will be calculated by the next formula

4.14+.5*xg

Thus the In-Form object with shape of the spiral with two coils can be created by following two In-Form statements

```
(INFOB at PATCH1 is SPHERE(1+.5*sin(xg),1+.5*cos(xg),1+.5*xg,.2) with INFOB_1)
(INFOB at PATCH1 is SPHERE(1+.5*sin(xg),1+.5*cos(xg),4.14+.5*xg,.2) with INFOB_1)
```

Moreover they can be incorporated in one statement like as

```
(INFOB at PATCH1 is SPHERE(1+.5*sin(xg),1+.5*cos(xg),1+.5*xg,$
.2)+SPHERE(1+.5*sin(xg),1+.5*cos(xg),4.14+.5*xg,.2) with INFOB_1)
```

Then the heat source and resistance are set inside spiral.

```
(SOURCE of TEM1 at PATCH1 is 2000. with INFOB_1)
(SOURCE of U1 at PATCH1 is -1000.*VABS*VOL*U1 with INFOB_1)
(SOURCE of V1 at PATCH1 is -1000.*VABS*VOL*V1 with INFOB_1)
(SOURCE of W1 at PATCH1 is -1000.*VABS*VOL*W1 with INFOB_1)
```

where VABS is the square root of symmetrically-computed velocity-squared.

7. Sources and boundary conditions

7.1. The SOURCE keyword

The **(source** ...) keyword of In-Form is used for ascribing sources of solved-for variables, i.e. those which have featured in a preceding SOLVE(), or SOLUTN(name,y,y,....) statement.

Since boundary conditions are, in PHOENICS, represented by way of sources, they too are the subject of the present discussion.

A **simple example** is provided by library case 105.

```
PATCH(X1,CELL,1,1,1,NY,1,NZ,1,LSTEP)
if(inform) then
 (SOURCE of H1 at X1 is 0.0 with FIXV)
else
 COVAL(X1,H1,FIXVAL,0.0)
endif


PATCH(WHOLE,VOLUME,1,NX,1,NY,1,NZ,1,LSTEP)
if(inform) then
 (SOURCE of H1 at WHOLE is 1.e4*(2.0-h1) with LINE)
else
 COVAL(WHOLE,H1,1.E4,2.0)
endif
```

In-Form statements containing the 'source' keyword commonly contain pre-formula and post-formula qualifiers. In the case just referred to:

- 'at X1' (a patch name) preceded the 'formula' (in actuality a simple constant); and
- 'with FIXV' followed it, signifying that fixed values were to be applied at the locations in question, corresponding to the FIXVAL argument of the COVAL statement which was being replaced. This 'source' is indeed a 'boundary condition'.
- However, there is also another source, attached to the patch called WHOLE, for which the option is 'with LINE', signifying that linearization should be applied. This is not a boundary condition; it represents a source of enthalpy within the flow domain.
- Here it may be remarked that 'linearization' was activated in the pre-In-Form (i.e. COVAL) formulation by the user's provision of separate CO and VAL arguments; and, if a truly non-linear source had been required, the user would have been at a loss.
- In-Form will however of its own accord linearize non-linear sources also, for example:

  (SOURCE of H1 at WHOLE is 1.e4*(2.0-h1)^3 with LINE)

  the non-linearity being caused by the '^3' insertion.

It does so however only when the source decreases with increase of the solved-for variable (i.e. H1 in this case); for otherwise numerical instability might occur.

Other options

The complete set of options which can be used with the 'source' keyword is:

- FIXFLU (fixed-flux, the default)
- FIXVAL (fixed-value)
- LINE (linearise)
- VOLU (for VR objects)
- MASS

- WHOLOB (for VR objects)
- INFOB_n (for In-form objects)
- IMAT=, or > or < or >=, or <=, or != iprp
- IF(condition)
- ONLYMS (for mass-flow patches)
- LAMWALL (for wall patches)
- BLASIUS (for wall patches)
- LOGLAW (for wall patches)

All these will be discussed by reference to the following examples, which are now distinguished by the means of localization employed, namely by connexion with:

- patches, in sub-section 7.2, or
- VR (i.e. facetted) objects 7.3, or
- In-Form objects 7.4.

These three topics will be discussed in turn.

7.2. Sources for patches

a. Sources of scalar quantities

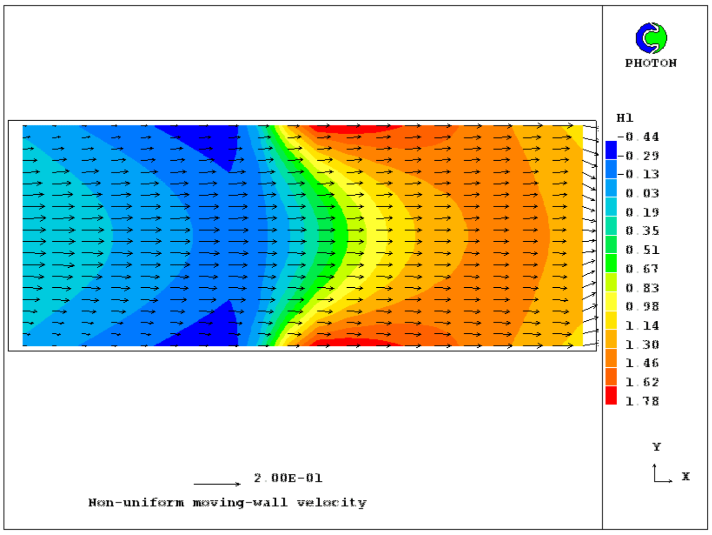
FIXVAL and FIXFLU

Two cases of the use of the In-Form SOURCE keyword to provide sources of a scalar were encountered in section 7.1 above, in connexion with library case 105. One employed the **fixval** option and had the nature of a boundary condition; the other employed the **fixflu** option, by default, and was qualified by the 'with LINE' condition.

Wall boundary conditions

In library case 745, a piece-wise-linear formula is used for wall enthalpy, first falling with longitudinal distance, then rising, then falling again.

The resulting enthalpy distribution in the duct is shown here.



Inspection of the relevant statements in the Q1 shows that:

```
CHAR(FORM)
FORM=pwl3(xg,0,0,.4*xulast,-.5*:twallmax:,.5*xulast,$
2.0*:twallmax:,xulast,:twallmax:)  ! piece-wise linear formula for H1
```

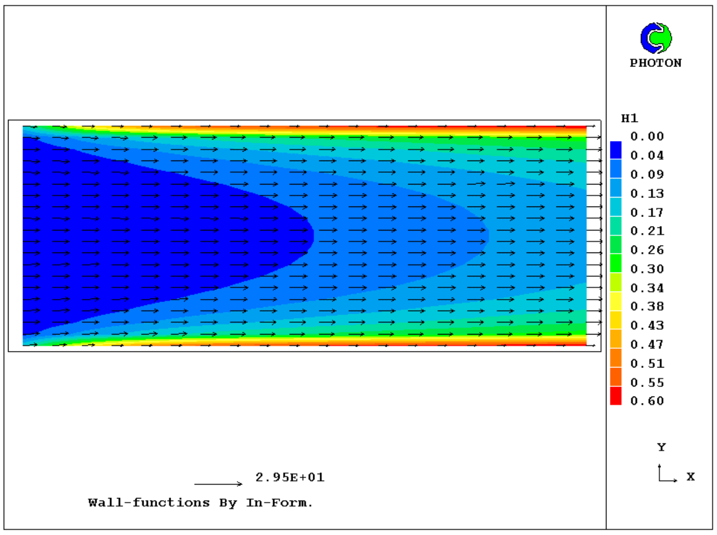
```
PATCH(SW,SWALL,1,NX,1,1,1,NZ,1,1)
(SOURCE of U1 at SW is UWALLMAX*XU/XULAST)
(SOURCE of H1 at SW is :FORM:)
PATCH(NW,NORTH,1,NX,NY,NY,1,NZ,1,1)
(SOURCE of U1 at NW is UWALLMAX*XU/XULAST with LAMWALL)
(SOURCE of H1 at NW is :FORM: with LAMWALL)
```

- The PATCH on the south wall of the duct has been given the 'type' SWALL, and the In-Form statement carries the formula for the wall temperature without qualification.
- The PATCH on the north wall of the duct, by contrast, and for the sake of variety, has been given the 'type' NORTH; for this reason 'with LAMWALL' has been appended to the formula as a condition, in order to ensure that the appropriate multiplication by the transport property and division by the node-to-wall distance should take place.
- An alternative to use of this qualifier would be to introduce the said multiplication and division into the formula.
- The symmetry of the solution shows that it does not matter which method is used.

In case 748, the 'In-Form' way of setting wall conditions is compared with the 'COVAL' way by using them for opposite walls of a plane-walled duct.

```
  South Wall !
PATCH(SW,SWALL,1,NX,1,1,1,NZ,1,1)
IF(ICASE.EQ.1) THEN
 COVAL(SW,U1,BLASIUS,0.0)
 COVAL(SW,H1,BLASIUS,1.0)
 COVAL(SW, A,BLASIUS,1.0)
 COVAL(SW, B,BLASIUS,1.0)
ELSE
 COVAL(SW,U1,LOGLAW,0.0)
 COVAL(SW,H1,LOGLAW,1.0)
 COVAL(SW, A,LOGLAW,1.0)
 COVAL(SW, B,LOGLAW,1.0)
ENDIF
  North Wall
PATCH(NW,NWALL,1,NX,NY,NY,1,NZ,1,1)
IF(ICASE.EQ.1) THEN
 (SOURCE of U1 at NW is 0.0 with BLASIUS)
 (SOURCE of H1 at NW is 1.0 with BLASIUS)
 (SOURCE of A  at NW is 1.0 with BLASIUS)
 (SOURCE of B  at NW is 1.0 with BLASIUS)
ELSE
 (SOURCE of U1 at NW is 0.0 with LOGLAW)
 (SOURCE of H1 at NW is 1.0 with LOGLAW)
 (SOURCE of A  at NW is 1.0 with LOGLAW)
 (SOURCE of B  at NW is 1.0 with LOGLAW)
ENDIF
```

The solution should be symmetrical, as indeed it is.



Scalar source with a mass source

Case 745 also contains a mass-flow source for enthalpy, about which the most interesting feature is the condition 'with ONLYMS'.

```
PATCH(INL,WEST,1,1,1,NY,1,1,1,1)  ! parabolic velocity, uniform H1
(SOURCE of P1 at INL is RHO1*VELMAX*(1.-(2.*YG/YVLAST-1)^2))
(SOURCE of U1 at INL is VELMAX*(1.-(2.*YG/YVLAST-1)^2) with ONLYMS)
(SOURCE of V1 at INL is 0.0 with ONLYMS)
(SOURCE of H1 at INL is TWALLMAX*0.2 with ONLYMS)
```

This signifies, as will be easily understood by those who recall that ONLYMS can also be the third argument of a COVAL statement, that the enthalpy flux consists of the enthalpy defined by the formula (here TWALLMAX*0.2) times the mass flux.

Use of SIN

In library case 717, which compares the numerical and analytical solutions of a steady-state two-dimensional diffusion problem, sources are introduced which involve the sine function.

There are three such sources, two for wall boundary conditions and the third within the domain.

```
PATCH(IYEQ1,NORTH,1,NX,NY,NY,1,1,1,1)
(SOURCE of H1 at IYEQ1 is SIN(1.)*SIN(XG) with LAMW)

PATCH(IXEQ1,EWALL,NX,NX,1,NY,1,1,1,1)
(SOURCE of H1 at IXEQ1 is SIN(1.)*SIN(YG))

PATCH(ISOURCE,VOLUME,1,NX,1,NY,1,1,1,1)
(SOURCE of H1 at ISOURCE is 2.*SIN(XG)*SIN(YG))
```

Dependence on time and position

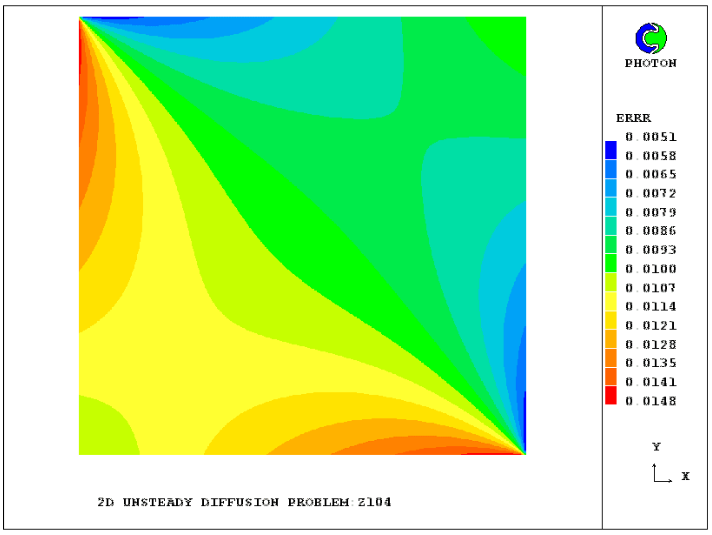
In library case 704 compares numerical and analytical solutions for a transient diffusion problem, for which the variable C1 has to be given boundary fluxes dependent on time, on position, and on the value itself.

```
PATCH(IWEST,WWALL,1,1,1,NY,1,NZ,1,LSTEP)
(SOURCE of C1 at IWEST IS (TIM+YG-C1)*2./DXG with line)

PATCH(IEAST,EWALL,NX,NX,1,NY,1,NZ,1,LSTEP)
(SOURCE of C1 at IEAST IS (TIM+1.0+YG-C1)*2./DXG with line)

PATCH(ISOUTH,SWALL,1,NX,1,1,1,NZ,1,LSTEP)
(SOURCE of C1 at ISOUTH IS (TIM+XG-C1)*2./DYG with line)
```

```
PATCH(INORTH,NWALL,1,NX,NY,NY,1,NZ,1,LSTEP)
(SOURCE of C1 at INORTH IS (TIM+1.0+XG-C1)*2./DYG with line)
```

```
PATCH(IERROR,CELL,1,NX,1,NY,1,NZ,1,LSTEP)
(SOURCE of ERRR at IERROR IS TIM+XG+YG-C1 WITH FIXVAL
```

Also computed, and given values by way of an In-Form source with fixval, is the variable named ERRR which represent the difference between the two solutions, a contour plot of which is shown here.



It is interesting to observe that the maximum and minimum errors (of the order of 1%) are to be found near the north-west and south-east corners.

b. Mass sources

Inlet mass-flux profile

- An input mass flow through the west face of a domain can be defined as follows:

  PATCH(INL,WEST,1,1,1,NY,1,1,1,1)

  (SOURCE of P1 at Patch_Name IS Input_mass_flow_formula)

- A **uniform** input mass flow can be set, for example, as

  (SOURCE of P1 at INL is RHO1*VELIN)

  where VELIN is input velocity, and RHO1 the density.

  An example can be found in library case 746 which is concerned especially with In-Form-based boundary conditions.

- The COVAL equivalent of this statement would be:

  COVAL(INL,P1,FIXFLU,RHO1*VELIN)

  to which the In-Form statement is not particularly superior.

- The **superiority of In-Form** appears when what is desired is a **non-uniform** inflow rate.

  For example a parabolic distribution of mass flow across the entrance to a channel can be prescribed via:

  (SOURCE of P1 at INL is RHO1*VELMAX*(1.-(2.*YG/YVLAST-1)^2))

  where:

  YG is the distance of the grid point from the south wall, YVLAST the distance from the south wall to the north and VELMAX is the Maximum velocity, as is to be found in library cases 745 and 747 .

An environmental example

Library case 401 concerns the flow induced in the upper layer of the ocean when a plume of somewhat-less-dense oil-droplet-carrying water rises from a tanker wreck on the ocean floor.

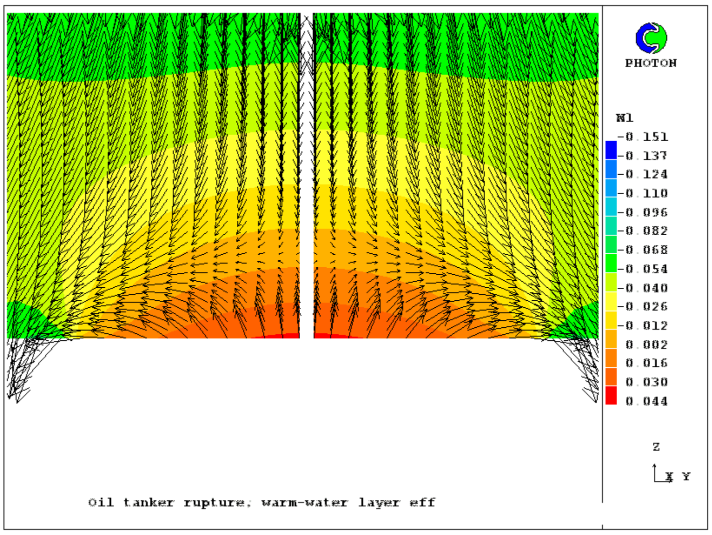
Separate (parabolic-mode) PHOENICS studies of the plume have shown that the plume has an approximately sinusoidal velocity profile; and it is therefore this which must be provided as an input to the (elliptic) upper-layer calculation.

The relevant lines are:

```
patch(leak,low,1,1,1,12,1,1,1,1)
(source of p1 at leak is 1000*0.03*sin((532-yg)/338.7) with fixflu)
```

from which it can be deduced that the half-width of the sine curve is 532 meters.

The resulting w1 contours and velocity vectors near the base of the layer can be seen here.



The contours show that the fluid entering from below quickly changes its direction to horizontal and then downward, no doubt because of the influence of the density gradients.

This case will be referred to again under the heading of 'momentum sources'.

Injection through a wall

Library case 747 illustrates flow in a duct with a porous wall, through which additional fluid is injected.

```
  Injection
PATCH(NINJ,NORTH,1,NX,NY,NY,1,NZ,1,1)
(SOURCE of P1 at NINJ is RHO1*VELINJ*XG/XULAST)
(SOURCE of V1 at NINJ is -VELINJ*XG/XULAST with ONLYMS)
(SOURCE of H1 at NINJ is 1.0 with ONLYMS)
```

The injection rate increases linearly with distance along the duct.

c. Momentum sources

Inlet velocity profile

The cases used for the illustration of mass-flow boundary conditions, namely:

745, 746 and 747

illustrate momentum sources also, of which the interesting ones are those for U1, the velocity normal to the inlet surface.

Of these, case 746 is the simplest; for the 'formula' of the In-Form source statement is simply: 'VELIN with ONLYMS', the latter condition signifying that the momentum flux into each cell equals VELIN times the mass flow per unit area.

```
  Inlet
PATCH(IN,WEST,1,1,1,NY,1,1,1,1)
(SOURCE of P1 at IN is RHO1*VELIN)
(SOURCE of U1 at IN is VELIN with ONLYMS)
(SOURCE of V1 at IN is 0.0 with ONLYMS)
(SOURCE of H1 at IN is 0.0 with ONLYMS)
```

The 'with ONLYMS' condition is identical in its action to that of ONLYMS as the third argument of a PIL COVAL command.

Had the condition not been supplied, the formula would have had to be written as:

RHO1*VELIN^2 .

Similar remarks are appropriate about cases 745 and 747.

Wall-friction examples

Laminar wall friction is represented in cases 746, 745 and 747. In the first case, the walls are at rest.

```
  Walls
PATCH(SW,SWALL,1,NX,1,1,1,NZ,1,1)
(SOURCE of U1 at SW is 0.0)
(SOURCE of H1 at SW is 1.0)

PATCH(NW,NORTH,1,NX,NY,NY,1,NZ,1,1)
(SOURCE of U1 at NW is 0.0 with LAMWALL)
(SOURCE of H1 at NW is 1.0 with LAMWALL)
```

Therefore the 'formula' is either '0.0' (for the patch of SWALL type) or '0.0 with LAMWALL' for that of NORTH type.

The COVAL command of PIL could express this condition perfectly well; but this is not true of case 745, in which the walls are moving at velocities which are proportional to XU, the distance along the duct. For this, In-Form's capabilities are needed.

```
PATCH(NW,NORTH,1,NX,NY,NY,1,NZ,1,1)
(SOURCE of U1 at NW is UWALLMAX*XU/XULAST with LAMWALL)
```

An environmental example

Library case 401 has already been referred to in respect of mass sources. Now the momentum source is considered.

Inspection of the two lines:

```
(source of p1 at leak is 1000*0.03*sin((532-yg)/338.7) with fixflu)

(source of w1 at leak is 0.03*sin((532-yg)/338.7) with onlyms)
```

reveals that the 'formula' in the second, which represents the momentum source, is the same as that in the first, apart from the absence of the 1000.0 which represents the density.

Coupled with the condition 'with ONLYMS', this entails that the momentum flux per unit area is:

density * velocity ** 2 .

which is what is required.

Stirring caused by a paddle

Library case 756 illustrates momentum sources which vary with both position and time. This is effected by the In-Form statements for sources of U1 and V1:

```
PATCH(ALL,CELL,1,NX,1,NY,1,NZ,1,LSTEP)

(SOURCE of U1 at ALL is 1.E5*(VEL*(YIC-YG)-U1) with IMAT>=PAD!LINE)

(SOURCE of V1 at ALL is 1.E5*(VEL*(XG-XIC)-V1) with IMAT>=PAD!LINE)
```

which signify that, over the patch named ALL (which extends over the whole domain), the sources are proportional to the difference between the prevailing velocity at the point and the position-dependent quantities:

VEL*(YIC-YG) and VEL*(XG-XIC).

The constant multiplying the velocity differences has been made large, so as to cause U1 and V1 to be close to the desired values.

The **time**-dependence is brought about by clever use of the STORED keyword, which, being placed in a DO loop in the Q1 file, provides a different distribution of IMAT at the start of each times step.

This is not the most economical method of simulating a moving object (see section 8 for better ones); but it well illustrates the power of well-chosen In-Form statements.

The condition IMAT>=PAD, signifies that the source is to be applied only to those cells for which the PRPS-value, IMAT, is greater than or equal to PAD, the material index of the paddle, which happens to be 100., which is greater than the 67.0 (signifying water) which prevails elsewhere in the domain.

Following the delimiter is a second condition, namely "LINE", which is an abbreviation of "LINEARISED".

This is an instruction which requires that the source is introduced in the linearised manner which ensures that it becomes zero when the velocity equals the desired value.

Without this condition, the source would have been created in the "fixed-flux" manner, which, in COVAL statements, is specified by setting FIXFLU as the third argument.

Frictional resistance

In case 709, In-Form is used for introducing a frictional resistance at the bottom of a channel

```
PATCH(IBOTSTR,WEST,1,1,1,NY,1,NZ,1,1000)
(SOURCE of W1 at IBOTSTR is -0.003*1000.*VABS*W1 with LINE)4
(SOURCE of V1 at IBOTSTR is -0.003*1000.*VABS*V1 with LINE)
```

In the formulae used, VABS stand for the absolute velocity at the centre of the scalar cells, 1000. for the presumed fluid velocity and 0.003 for the friction factor which it is desired to introduce.

Therefore a formula which would be somewhat closer to the evident intention would be, for the source of W1, to replace VABS by 0.5*(VABS+HIGH(VABS)), with a corresponding modification for the source of V1.

Such refinements may not have been thought to be worthwhile by the author of the library case; but In-Form could provide them were they required.

d. Other sources

Pressure gradients

An interesting use of the In-Form source statement is provided by library case 759 which contrives, in a very simple manner, to provide a solution for the so-called "co-located velocities" (ie those prevailing at the cell centres), for the well-known "square-cavity" problem.

```
  ** Pressure gradient sources
  *** For the central cells
PATCH(DPDX1,EAST,2,NX-1,1,NY,1,1,1,1)
(SOURCE of UC1 at DPDX1 is (P1[-1]-P1[+1])/2.)

PATCH(DPDY1,NORTH,1,NX,2,NY-1,1,1,1,1)
(SOURCE of VC1 at DPDY1 is (P1[&-1]-P1[&+1])/2.)
```

The indexing technique is employed, because it is necessary to introduce pressure gradients into the formulae.

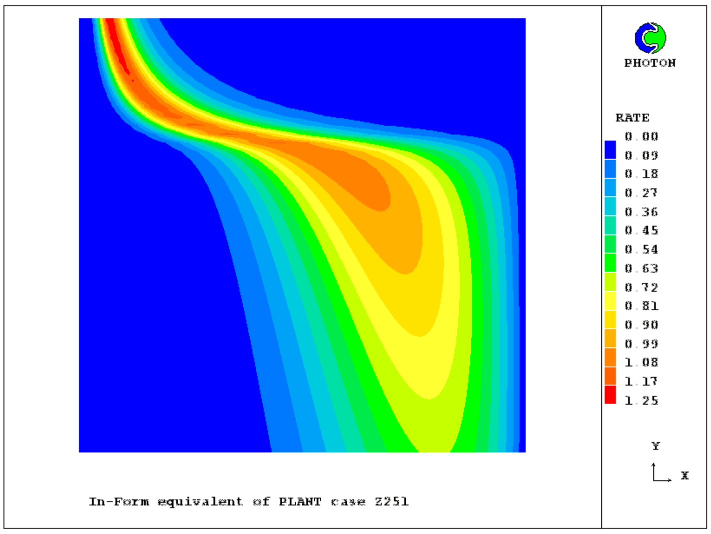
Chemical reaction

Case 751 illustrates how In-Form can be used for the creation of chemical-reaction sources, in this case for a reaction rate which depends both on the value of the dependent variable and on the x- and y-coordinates.

```
PATCH(IREACRAT,VOLUME,1,NX,1,NY,1,1,1,1)
(SOURCE of RCTD at IREACRAT is RCONST*RCTD^EXPO*XG^(-2)$
*(1.0+YG)*(1-RCTD) with LINE)
(SOURCE of RATE at IREACRAT is RCONST*RCTD^EXPO$
*(1.0+YG)*(1-RCTD) with FIXV)
```

A single differential equation is being solved, namely that for the 'reactedness', RCTD; but a second variable, namely RATE, is computed for each location; it is the volumetric reaction rate.

It is the distribution of the latter with x and y which is displayed in this contour plot.



Case 492 illustrates the use of In-Form to simulate chemical reaction in a turbulent gas by way of the 'eddy-break-up' formula.

```
PATCH(WHOLE,PHASEM,1,NX,1,NY,1,NZ,1,1)
(source of FUEL at WHOLE is -CEBU*EPKE*FUEL with line)
```

Velocity potential and stream function

Since the usability of PHOENICS for solving ideal-fluid problems is sometimes overlooked, cases 126 and 127 may be interesting.

In-Form is used for setting the **velocity potential**,

```
PATCH(NORTH,NORTH,2, NX-1,NY,NY,1,1,1,1)
PATCH(SOUTH,SOUTH,2, NX-1,1 ,1 ,1,1,1,1)
PATCH(EAST ,EAST ,NX,NX,  1 ,NY,1,1,1,1)
PATCH(WEST ,WEST ,1 , 1,  1 ,NY,1,1,1,1)
  INFORM13BEGIN
REAL(AA,BB)
AA=-1; BB=-1
(source of POT at north is aa*xg + bb*yg with fixval)
(source of POT at south is aa*xg + bb*yg with fixval)
(source of POT at east  is aa*xg + bb*yg with fixval)
(source of POT at west  is aa*xg + bb*yg with fixval)
```

in the former, and the **stream function**, in the latter, at the boundaries of a domain in which a plate is held at an angle to the stream in which it is immersed.

The computed velocity vectors for the two cases are shown on-line here and here.

7.3. Sources for VR-objects

a. Syntax

Method 1

In-Form can set the sources for a VR-type object in a manner similar to that used for patches, simply replacing the PATCH name by the OBJECT name, thus:

(SOURCE of VAR_NAME at VR_OBJECT_NAME is FORMULA)

The VR_OBJECT_NAME, like a patch name, marks the part of the domain inside which the source is to be applied; but it does so in geometry-related terms by way of lines in the Q1 file starting with >OBJ.

The object-describing lines usually appear below the source-describing lines in the Q1 file; but this is not necessary.

Method 2

In-Form sources for a VR-type object can also be set by editing them into the VR section of the Q1 file:

The source formulas are placed in the lines which describes attributes of VR objects, thus:

> OBJ, INFSRC_VARNAME, FORMULA

If the length of a line exceeds 68 symbols then the formula continuation is recorded in the next line, as follows:

> OBJ, INFSRC_VARNAME, FORMULA_BEGINNING$

> OBJ, INFSRC_VARNAME, FORMULA_CONTINUATION

[(SOURCE , it should be mentioned, is not the only In-Form keyword which can be treated by this second method. Thus INFINI_, INFSTO_, INFST1 and INFMAK_ have the same effects as the In-Form statements starting: (INITIAL , (STORED , (STORE1 and (MAKE .]

The In-Form object-defining lines may be written to the Q1 file by way of any word-processor.

Also the In-Form object-defining expressions can be located in '.DAT' or '.POB' files. In this case SATELLITE reads them from these files and places them appropriately in the Q1 file.

8. Moving objects and patches

8.1. Patches

How the motion is described

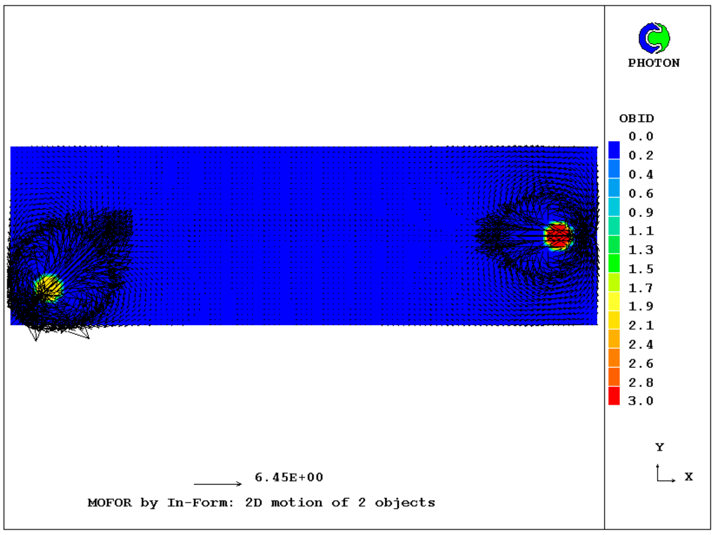
The MOVOB keyword of In-Form enables the motion of VR objects to be described by way of formulae. How it is used can be seen by inspection of core library case 360, which shows two objects moving along different trajectories through a fluid.

MOVOB statements make use of the POS function for setting the X, Y and Z coordinates and the rotation angles about the X, Y and Z axes of a moving object as functions of the time variable, TIM:

In case 360, the relevant statements are:

```
char(vel,gravt,times)
vel=10.; gravt=9.81; times=tim

(MOVOB of SPHERE1 is POS(:times:*:vel:,:times:*:vel:-0.5*:gravt:*:t$
imes:^2,0,0,0,0))
(MOVOB of SPHERE2 is POS(-:times:*:vel:,0,0,0,0,0))
```

These dictate a motion which, after the first time step appears thus.



Since the connexion between the two is not perfectly obvious, the following discussion is in order:

- Examination of the Q1 shows that the time interval is 0.1 second and that the domain size is 22 meters long and 7 meters high.
- Since the velocity has been set to 10 m/s, the x-coordinate of sphere1 is expected to be at 0.1*10 (i.e. :times:*:vel:) metres from its starting position; so it will have travelled a fraction 1/22 of the length of the domain.

  Its distance of vertical travel will be about the same, because the acceleration term has not had time to become significant.

  The presence of one sphere near the bottom left-hand corner in the just-seen picture suggests that this indeed sphere 1.
- Where should sphere2 be? Its x-coordinate and y-coordinate arguments in the (MOVOB....) statement are -1 (i.e. -:times:*:vel:) and zero. Yet, in the PHOTON

picture, it appears to be near the extreme right of the domain, and halfway up. What is the explanation?

- To discover this, one must look at the object-describing part of the Q1 file, where are to be found the lines:

```
> OBJ,NAME,SPHERE1
> OBJ,POSITION,0.000000E+00, 0.000000E+00, 0.000000E+00
> OBJ,NAME,SPHERE2
> OBJ,POSITION,2.100000E+01, 3.000000E+00, 0.000000E+00
```

in which the 'position' lines show that sphere1 starts at the origin of the coordinate system, whereas sphere2 starts at x=21 and y=3.

- From these observations it can be concluded that the x,y and z coordinates in the MOVOB statement are measured relative to the positions of the objects in question laid down in the '>OBJ' lines.

Relation to MOFOR

The movement of the objects causes motions in the fluid also. This is achieved by the MOFOR feature of PHOENICS, about which the following remarks may suffice for present purposes:

- In the Q1 file for case 360, MOFOR is activated by the line:

```
SPEDAT(SET,MOFOR,MOFFILE,C,NOTSET)
```

This provides two pieces of information, namely:

1. that the MOFOR mechanism is to be activated; and
2. that a 'MOFFILE' is not to be used.

- The explanation of the second item is that, when MOFOR was first introduced to PHOENICS, the motion of objects was conveyed by means of an ASCII file with extensions .mof (earlier, .bvh). This means is still operative.

  However, the moffile approach is rather too clumsy for motions which are simple enough to be described by formulae. This is indeed the reason for the introduction of the In-Form MOVOB keyword.

- The .mof/.bvh file format is provided with means of describing the motion of articulated objects, arranged in a 'hierarchy', such for example as the torso, upper arms, lower arms, hands, and fingers of a human being, which are 'offset' relative to one another, and have limited freedom of relative movement.

  As will be seen, the 'hierarchy' and 'offset' concepts are also embodied in MOVOB.

The OFFSET function

POS is not the only function which can appear in a MOVOB statement. Another is OFFSET which is used when several objects move in pre-defined relative motion.

It is exemplified in library cases 381 and 382.

8.2. Moving VR objects

In-Form can calculate coordinates of a moving body position by the formula specified in "(MOVOB" In-Form statements which uses POS and OFFSET special functions. "(MOVOB" statement has next format

```
 (MOVOB of VR_OBJECT_NAME is special_FUNCTION() with PARENT=PARENT_NAME)
```

where VR_OBJECT_NAME is a name of VR object name which is connected to a appropriate object-related coordinate system for which the current attributes are set.

The special In-Form flag "!PARENT=PARENT_NAME" set the name of parent object-related coordinate systems.

The OFFSET function describes the hierarchy part of the attribute settings. OFFSET is In-Form special function of declaring the frames of reference of object-related coordinate systems. Each new OFFSET function declares a new frame coordinate system and describes its position relative to its parent system. OFFSET has next format

```
OFFSET(Xorigin, Yorigin, Zorigin)
```

where Xorigin, Yorigin, Zorigin are formulas for calculation of coordinates of the origin position of the rotation axis relative to its parent.

POS is an In-Form function which describes the coordinates of the position of a moving object in appropriate object-related coordinate system. POS has the format

```
POS(Xpos, Ypos, Zpos, Xang, Yang, Zang)
```

where

Xpos, Ypos, Zpos are formulas for calculation of X, Y, Z coordinates of the position of moving VR object in meters.

Xang, Yang, Zang are formulas for setting the rotation angle of moving VR object about X, Y, Z axis in degrees.

Each formula can contain the TIM variable which is current time in seconds at the current time step.

VEL is an In-Form function which describes the velocity of a moving object in appropriate object-related coordinate system. VEL has the format

```
VEL(Xvel, Yvel, Zvel, Xrot, Yrot, Zrot)
```

where

Xvel , Yvel, Zvel are formulas for setting the X, Y, Z linear velocity components of moving VR object, m/s;

Xrot, Yrot, Zrot are formulas for setting the rotation velocities of moving VR object about X, Y, Z axis, 1/s.

ACC is an In-Form function which describes the acceleration of a moving object in appropriate object-related coordinate system. ACC has the format

```
ACC(Xacc, Yacc, Zacc, Xracc, Yracc, Zracc)
```

where

Xacc, Yacc, Zacc are formulas for setting the acceleration of linear motion of moving VR object in X, Y, Z directions, m/s^2;

Xracc, Yracc, Zracc are formulas for setting the acceleration of rotation motion of moving VR object about X, Y, Z axis, 1/s^2.

The examples of In-Form use for MOFOR attributes settings are submitted below.

Elementary motion: linear movement

The task is to provide the attributes for a simple motion of a 2D rectangular object called "BLOCK" with uniform velocity of 3m/s in Z-direction of computational domain starting from a given initial, stationary position of the object.

The description the hierarchy part is not mandatory as there is only one moving object. Therefore it is enough to define the Z position of moving object in each time step.

```
(MOVOB of BLOCK is POS(0, 0, 3*TIM, 0, 0, 0))
```

The movement is along Z-axis. The initial position and the sizes of the BLOCK are specified in the Q1 file. Z coordinates of moving BLOCK object are described as

Z velocity component[m/s] * current time[s].

Composite movement: rolling block

The case exemplifies the specification of the attributes for composite movement: the translation of rotating block. It results in the effect of the rolling 2D rectangular object with the longitudinal (X-direction) velocity component of 0.5 m/s and the angular rotation about block centre line (Z-axis) of 30 degrees per seconds. The movement starts from the initially steady position defined in the Q1 file.

The hierarchy part is described as follows:

```
(MOVOB of CHAM is OFFSET(0&0&0))
(MOVOB of BLOCK is OFFSET(0.55&0.9&0) with PARENT=CHAM)
```

The OFFSET defines the position of the rotation axis relative to the root frame: it places the Z-axis in the middle of BLOCK initial position.

The movement is along X-axis and the rotation about Z-axis. X coordinates of moving BLOCK object and rotation angles are described as

```
(MOVOB of BLOCK is POS(0.5*tim&0&0&0&0&-30*tim))
```

Independent motions: crossing paths

The case exemplifies the setting-up the attributes for the independent movements of two objects following their own linear trajectories in 2D, X-Y, computational space.

The movement of the first object, called SPHERE1, starts from its stationary position at the west-south corner and follows a prescribed parabolic trajectory. The velocity of SPHERE1 is 10 m/s and is defined by interaction of gravitation force.

The second object, SPHERE2, starts at east-south corner of the domain and follows the linear trajectory crossing the one from right to left. It has the constant velocity component of -10 m/s in X-direction and 3 m/s in Y-direction.

The movement starts from the initially steady position defined in the Q1 file.

The description the hierarchy part is not mandatory as two moving objects move inside one root coordinates system. Therefore it is enough only to define the position of moving objects in each time step.

The movement of SPHERE1 object is along diagonal of XY plane. X and Y coordinates of moving object are described as

```
vel=10.; gravt=9.81
(MOVOB of SPHERE1 is POS(tim*:vel:&tim*:vel:-0.5*:gravt:*tim^2&0&0&0&0))
```

The movement of SPHERE2 object is along X-axis from right to left. X coordinates of moving object are described as

```
(MOVOB of SPHERE2 is POS(-tim*:vel:&0&0&0&0&0))
```

Connected objects: falling of a cracked wall

The case exemplifies the setting-up the attributes for the connected movements of two objects following their relative rotation in 2D, Y-Z, computational space.

The movement of the first object, called BLOCK, starts from its stationary position as a vertical wall with its base placed next to the middle of the bottom domain boundary. BLOCK is allowed to fall. It does so by rotating clockwise about X-axis of its south-high corner. The angular velocity of the fall is ,5 degrees per second.

Initially, the second smaller object, TIP, sits stationary on the top of BLOCK and can be regarded as a part of the wall. Once the whole wall starts to fall, it cracks and TIP begins to move in opposite direction by rotating counter clockwise about X- axis of the north-low corner of the BLOCK. The angular velocity of the TIP relative to the BLOCK is 180 degree per second.

The hierarchy part is described as follows:

```
(MOVOB of BLOCK is OFFSET(0&0&2.1))
(MOVOB of TIP is OFFSET(0&2.5&-0.5) with PARENT=BLOCK)
```

Two frames are defined: BLOCK is a parent joint, and TIP is the BLOCK's child.

The OFFSET of BLOCK defines the position of the rotation axis relative to the root frame: it places the origin of parent related coordinate frame at the south-high corner of BLOCK initial position.

The OFFSET of TIP defines the position of the rotation axis relative to the BLOCK frame: it places the origin of its coordinate frame at the north-low corner of BLOCK initial position.

The movement of BLOCK object is a rotation clockwise about X plane. Rotation angle is described as

```
(MOVOB of BLOCK is POS(0&0&0&75*tim&0&0))
```

The movement of SPHERE2 object is a rotation counter clockwise about X plane. X coordinates of moving object are described as

```
(MOVOB of TIP is POS(0&0&0&-180*tim&0&0))
```

Agitated reactor

Impellers are the units one most commonly associates with stirred reactors. The particular design considered here consists of the ROD, mounted on rotating vertical SHAFT. The rod carries two PADDLEs, which agitate the flow. As the impeller rotates it forces the surrounding fluid to rotate with it. The design with rotating paddles is used when "in-depth" mixing is a must.

The task of the exercise is to provide the attributes for connected movements of SHAFT-ROD-PADDLEs assembly following their relative rotation in 3D computational space. The angular velocity of the shaft is 60 rpm.

The hierarchy part is described as follows:

```
(MOVOB of CHAM is OFFSET(0&0&0))
(MOVOB of SHAFT is OFFSET(0.15&0.25&0.25) with PARENT=CHAM)
(MOVOB of ROD is OFFSET(0&0&0) with PARENT=SHAFT)
(MOVOB of PADDLE1 is OFFSET(0&0&0) with PARENT=ROD)
(MOVOB of PADDLE2 is OFFSET(0&0&0) with PARENT=PADDLE1
```

Four frames are defined: SHAFT is a parent joint, and ROD, PADDLE1 and PADDLE2 are the SHAFT's children.

The OFFSET of SHAFT defines the position of the rotation axis relative to the ROOT frame: it places the origin of parent related coordinate frame at the middle of the SHAFT bottom face of its initial position.

The ROD and PADDLEs rotate about the same axis and has no freedom to move relatively to SHAFT. The OFFSETs of ROD and both PADDLEs are zero.

The movement of SHAFT object is a rotation clockwise about X plane. Rotation angle is described as

```
(MOVOB of SHAFT is POS(0&0&0&360*tim&0&0))
```

8.3. Moving In-Form objects

Introduction

As has just been seen, In-Form's MOVOB keyword enables the motion of VR objects to be described; then activation of MOFOR sets the fluid in motion.

However, even before the existence of MOVOB, In-Form had a means of causing its own In-Form to move, and to move the fluid with them. How this works will now be illustrated.

The fluid motion is achieved by attaching momentum sources to the In-Form objects. This was not in evidence when the MOVOB keyword was being used; but such sources were being introduced, nevertheless, behind the scenes.

Studying the 'source' statements associated with moving In-Form objects therefore throws some indirect light on how MOFOR works

a. Moving spheres

So the 765 library case sets the linear motion of two hot spherical bodies to follow a horizontal path as

```
char(xce,yce,zce,radius;
xce=.5+.5*(tim/100.-1); yce=0.1 + 01*(tim/100.-1); zce=.05; radius= .25
(INFOB at FIRST is SPHERE(:xce:,:yce:,:zce:,:radius:) with INFOB_1)
xce=.5+.25*(tim/100.-1); yce=1.5; zce=.05; radius= .25
(INFOB at SECOND is SPHERE(:xce:,:yce:,:zce:,:radius:) with INFOB_2)
```

It should notice that the '(INFOB' statement can contain several long formulas. However the maximum length of a Q1 line is 68 symbols. In addition a PIL command can consist of two parts connected by the '$' symbol. As a result the maximum length of a PIL command is 136 symbols. The possible maximum length of a formula in any In-Form statement given by default is 1000 symbols. In such cases for the setting of the long formulas it should use intermediate symbolical variables as was shown above.

Inside objects are present a heat sources.

```
(SOURCE of TEM1 at FIRST is 100. with INFOB_1)
(SOURCE of TEM1 at SECOND is 100. with INFOB_2)
```

b. Moving blades

The 770 library case set steady rotated In-Form object for simulation two paddle-stirred reactor. The rotated paddle is described by In-Form:

```
REAL(ANGVL); ANGVL=4.*PI/1. ! Number of revolutions, 1/s.
REAL(X0,Y0); X0=0.5; Y0=0.5 ! X and Y coordinate of centre of impellor
char(ang,dx1,dx2,dy1,dy2); ang=:ANGVL:*TIM;
dx1=:hsid2:*SIN(:ang:); dx2=:hsid1:*COS(:ang:)
dy1=:hsid2:*COS(:ang:); dy2=:hsid1:*SIN(:ang:)
(INFOB at PATCH1 is BOX(:x0:-:dx1:-:dx2:,:y0:-:dy1:+:dy2:,0,0.1,0.7,10,0,0,:ang:)
with INFOB_1)
```

The cartesian components of paddle velocity are set

```
(SOURCE of U1 at PATCH1 is :ANGVL:*(YG-:Y0:) with INFOB_1!FIXV)
(SOURCE of V1 at PATCH1 is :ANGVL:*(:X0:-XG) with INFOB_1!FIXV)
```

c. Moving valve

The inclined linear motion of rectangular object is submitted in 784 library case. Here four immobile In-Form objects (1, 2, 3 and 4) describe the geometry of blading sections of the chamber.

```
(INFOB at WHOLE is BOX(0.,.07,0.,.04,.03,1.,0.,0.,0.) with INFOB_1)
(INFOB at WHOLE is BOX(.07,0.,0.,.03,.04,1.,0.,0.,0.) with INFOB_2)
(INFOB at WHOLE is BOX(.04,.07,0.,.05,.03,1.,0.,0.,:-PI/4:) with INFOB_3)
(INFOB at WHOLE is BOX(.07,.04,0.,.03,.05,1.,0.,0.,:PI/4:) with INFOB_4)
```

Each of them is a solid body with appropriate installations inside objects

```
(STORED of VPOR at WHOLE is 0. with INFOB_1)
(SOURCE of U1 at WHOLE is 0. with INFOB_1!FIXV)
(SOURCE of V1 at WHOLE is 0. with INFOB_1!FIXV)
(STORED of VPOR at WHOLE is 0 with INFOB_2)
(SOURCE of U1 at WHOLE is 0. with INFOB_2!FIXV)
(SOURCE of V1 at WHOLE is 0. with INFOB_2!FIXV)
(STORED of VPOR at WHOLE is 0. with INFOB_3)
(SOURCE of U1 at WHOLE is 0. with INFOB_3!FIXV)
(SOURCE of V1 at WHOLE is 0. with INFOB_3!FIXV)
(STORED of VPOR at WHOLE is 0. with INFOB_4)
(SOURCE of U1 at WHOLE is 0. with INFOB_4!FIXV)
(SOURCE of V1 at WHOLE is 0. with INFOB_4!FIXV)
```

Two next In-Form objects (5 and 6) describe the geometry of a valve and a valve holder.

Description of a geometry of a valve

X and Y coordinates of a valve in the first time step

```
REAL(XVAL,YVAL); XVAL=0.065; YVAL=0.035
```

Moving of a valve during one time step

```
REAL(DMOV); DMOV=THROW/LSTEP; DMOV=DMOV/1.414
INTEGER(HLS,HLS1,HLS2); HLS=LSTEP/2; HLS1=HLS+1; HLS2=HLS+2
CHAR(CXP,CYP);
CXP=:XVAL:-:DMOV:*(:HLS:-ABS(ISTEP-:HLS1:));
CYP=:YVAL:-:DMOV:*(:HLS:-ABS(ISTEP-:HLS1:))
(INFOB at WHOLE is BOX(:CXP:,:CYP:,0.,.01,.042,1.,0.,0.,:-PI/4:) with INFOB_5)
```

Description of a geometry of the valve holder

X and Y coordinates of the valve holder in the first time step

```
REAL(DXH,DYH); DXH=0.0113; DYH=0.0252
(INFOB  at  WHOLE  is  BOX(:CXP:-:DXH:,:CYP:+:DYH:,0.,.01,.11,1.,0.,0.,:PI/4:)  with
INFOB_6)
```

Fixation of the valve velocities into valve

```
(SOURCE of U1 at WHOLE is :UP: with INFOB_5!FIXV)
(SOURCE of V1 at WHOLE is :VP: with INFOB_5!FIXV)
```

Fixation of the valve velocities into valve holder

```
(SOURCE of U1 at WHOLE is :UP: with INFOB_6!FIXV)
(SOURCE of V1 at WHOLE is :VP: with INFOB_6!FIXV)
```

d. Football trajectory, 2D

The 766 library case illustrates the use of the In-Form 'SPHERE' function to simulate the effect on the motion of the air of a football following a prescribed parabolic trajectory. The In-Form object is presented as

```
char(xce,yce,zce,radius,gravt,vel,times); gravt=9.81;vel=14.14;
times=tim;
xce=0.5+:times:*:vel:/1.414
yce=0.5+:times:*:vel:/1.414-0.5*:gravt:*:times:^2;
zce=.05; radius=.5
(INFOB at PATCH1 is SPHERE(:xce:,:yce:,:zce:,:radius:) with INFOB_1)
```

where xce, yce and zce are the x, y and z coordinates. They are character variables which are evaluated in the In-Form statements because they are enclosed within colons.

```
Setting of U1 and V1 values into SPHERE is
char(usour,vsour); usour=:vel:/1.414; vsour=:vel:/1.414-:gravt:*:times:
(SOURCE of U1 at PATCH1 is :usour: with INFOB_1!FIXV)
(SOURCE of V1 at PATCH1 is :vsour: with INFOB_1!FIXV)
```

e. Football trajectory, 3D

The modelling of motion of spherical object in 3D space is submitted in 767 library case.

The In-Form object is presented as

```
char(xce,yce,zce,radius,gravt,vel,times) gravt=9.81; vel=14.14; times=tim;
xce=0.5+:times:*:vel:/1.414-0.5*:gravt:*:times:^2; yce=0.;
zce=0.5+:times:*:vel:/1.414; radius=.5
(INFOB at PATCH1 is SPHERE(:xce:,:yce:,:zce:,:radius:) with INFOB_1)
```

Setting of U1, V1 and W1 values into SPHERE is

```
char(usour,wsour); usour=:vel:/1.414-:gravt:*:times:; wsour=:vel:/1.414
(SOURCE of U1 at PATCH1 is :usour: with INFOB_1!FIXV)
(SOURCE of V1 at PATCH1 is 0. with INFOB_1!FIXV)
(SOURCE of W1 at PATCH1 is :wsour: with INFOB_1!FIXV)
```

9. Print-out-related capabilities

9.1. LONGNAME

Variables which are introduced by way of In-Form statements beginning:

(STORED

are computed at each sweep through the solution domain. However, it sometimes occurs that values are required only for print-out purposes.

In these cases, In-Form makes use of the longname feature, which slightly pre-dates In-Form itself.

Syntax The syntax of longname statements can be deduced from the following example extracted from Input library case 763:

```
   The LONGNAME feature provides reminders at RESULT-reading time
(LONGNAME of L3RH print as 3-Piece-Wise_Linear_Density)
(LONGNAME of L3EN print as 3-Piece-Wise_Linear_Viscosity)
(LONGNAME of L3CP print as 3-Piece-Wise_Linear_Specific_Heat)
(LONGNAME of L3CN print as 3-Piece-Wise_Linear_Conductivity)
```

This illustrates the use of longname as a reminder, which will be printed in the RESULT file as a field-values title, of what formula has been used for the calculation of density and other properties.

The similarity of the property and longname entries in this example should not however lead to the supposition that their treatment by PHOENICS is similar.

On the contrary; whereas what follows "is" in the property statement **must** a meaningful formula, that in the longname statement can be **any** string of characters. Equally acceptable, for example, would be:

```
 (LONGNAME of VISL is my_new_formula)
```

9.2. PRINT

Another feature for a dump of variables values in a separate file is the '(PRINT' In-Form statement. It prints fields of standard dependent variables, or of user-defined single real variables calculated by Formula, in an INFOROUT file, which it places in the working directory.

Its format is:

```
(PRINT String_15 [at PatchName] is Formula)
```

where

String_15 is a character name with a length of 15 symbols;

PatchName is the name of a PATCH or VR object.

The length of String_15 together with PatchName is limited 15 symbols according to SPEDAT format.

The String_15 string is a comment of dumping variables.

Examples of use are:

```
(make pinlet) ! make user-defines single real variable
(store1 of pinlet is old(p1[1,1,1]) with if(isweep.eq.1)) ! define pinlet
(print of pin is pinlet) ! dump in INFOROUT file the pinlet value
(stored of pold is old(p1) with if(isweep.eq.1)) ! calculate POLD variable
(print of p1_old is pold) ! dump in INFOROUT file the fields of POLD
```

The 362 and 363 library cases use for a print-out of user-defined single real variables ASUM and TSUM.

```
(PRINT of Whole_high_area is ASUM)
(PRINT of IZ=NZ__Sum_TEM1 is TSUM)
```

9.3. Coefficients, residuals, corrections and exchange coefficients (gammas)

a. Coefficients

Users who are interested in the finite-volume equations solved by PHOENICS may print, and manipulate, terms which enter those equations.

The typical form these equations is seen by clicking here.

The terms which may be accessed are:

- the north, south, east, west, high and low coefficients:

  aN, aS, aE, aW, aH, aL;
- aP, which is added to the denominator; and
- S, which is the so-called "residual", i.e. the current imbalance in the equation.

The In-Form statements which enable them to be accessed have the form:

```
(STORED of name is AXCO(varname))
```

where:

- *name* is the name chosen by the user for the coefficient in question, for example E_V1 if the east coefficient of variable V1 is in question
- *X* is N, S, E, W etc according to the coefficient which is required; and
- *varname* is the name of the solved-for variable which is in question, i.e. V1 in the just-mentioned case.

A complete set of examples can be found in input-library case 788

b. Residuals

The equation imbalances, i.e. the residuals, may be obtained in a similar manner; but for them the In-Form Statement is:

```
(STORED of name is RESI(varname))
```

Such statements are also to be seen in case 277.

c. Corrections

Solving the finite-volume equations produces "corrections", i.e. quantities which should be added, cell-by-cell, to the values of the solved-for variables in order to reduce the imbalances in the equations.

To gain access to them, the appropriate statement is:

```
(STORED of name is CORR(varname))
```

Examples of accessing residuals and corrections are to be found in library cases: 249, concerned with 2D flow in a square cavity, and 768, concerned with 3D flow in a water heater.

If on-line, click here for the temperature field, the residual field, and the correction field for case 249.

Because the solution is well-converged, the values of both the residuals and the corrections are very small. This accounts for the irregularity of the residual field, of which the values have been reduced to the round-off error of the computer.

This method of displaying residuals and corrections is more convenient than the older-established method, which involved giving the variable of which residuals and corrections were required a special name, ending with the % sign. That method however is still available.

Warning

It will sometimes occur that the corrections are printed as zero. The explanation is that the iterative computation will have stopped before the prescribed value of LSWEEP because the residuals had fallen below the reference value; therefore the solver, which is where the corrections are computed, may not have been entered.

The cure is to set NPRINT to some value well below that of the number of sweeps at which solution terminated.

d. Exchange coefficients (gammas)

Also of interest to those who study the workings of the numerical-solution process are the exchange coefficients, i.e. diffusivities times densities, which enter (together with convection contributions) into the calculation of aN, aS, aE, aW etc. These too may be accessed by way of an In-Form statement, which is in this case:

```
(STORED of name is GAMM(varname))
```

This feature is illustrated in case 788.

e. Manipulating the coefficients, etc

Like other 3D-stored variables, the coefficients and other quantities mentioned in the present section may be:

- given initial values via FIINIT,
- confined between upper and lower values by way of VARMAX and VARMIN, and
- "relaxed" via RELAX(*name*,LINRLX*, factor*).

These possibilities give the user great power to intervene, if he or she wishes, in the solution process.

9.4. Eliciting debug

In-Form is supplied with extensive debug facilities, for the investigation of suspected errors.

These facilities are activated by use of the "read Q1" facility. Specifically, the Q1 file is supplied with, starting in column 3 or greater:

- the starting line:

  informbegin
- a second line:

  debug t
- a succession of further lines, of the kind:

  feature t

  or

  feature f

  where *feature* is one of the list given below
- concluded by the finishing line:

  informend

The full list of features is:

FULL

    for all the following

FORMULA

    for displaying the formula being parsed

INITIAL

    for initial-value settings

STORED

    for stored-value settings

MAKE

    for the MAKE facility

PROPERTY

    for property settings

SOURCE

    for sources

LINE

    for linearization aspects

XGRID

    for grid-related settings

YGRID

ZGRID

TGRID

INFOB

    for In-Form objects

MOFOR

    for moving objects

As is obvious, debug of a particular kind is activated when **t** follows the feature name, and deactivated by **f**.

An **f** following debug deactivates all debug features.

All debug features are **f** by default.

In-Form debug print-out is governed by the same indices as is the older-established debug from EARTH, i.e. that elicited by the PIL variable debug, namely izdb1, izdb2, etc.

Library case 768 provides an example.

10. Replacing READQ1

10.1. READREALS, READINTS and READLOGS

The READQ1 features of PHOENICS allow variables which do not form part of the PIL set to be transmitted to EARTH. However, each variable requires one extra line to be inserted in the Q1 file.

In-Form allows up to 20 variables to be transmitted in a single line (possibly with 'continuation' effected by the $ sign), by means of the READREALS, READINTS, and READLOGS statements.

Their use is illustrated by the facility (introduced with PHOENICS-3.5) for changing turbulence-model constants, which is described here,

Suppose, for example, it were desired to employ the different constants:

CMU	**=0.6**	**CD**	**=0.2**	**C1E**	**=1.5**
C2E	**=1.92**	**AK**	**=0.4**	**EWM**	**=9.0**

This could be effected by inserting in the Q1 file the line:

```
(READREALS turconst 0.6 0.2 1.5 1.92 0.4 9.0)
```

When the satellite reads this line, it places in Q1EAR the line:

```
 SPEDAT(SET,REALREAD,TURCONST,C,0.6&0.2&1.5&1.92&0.4&9.0)
```

which is echoed in RESULT; and in EARDAT the line:

```
 REALREAD  TURCONST   C0.6&0.2&1.5&1.92&0.4&9.0
```

This is then acted upon by the sequence of coding starting with IF GETREALS in the subroutine INIVST,

```
        IF(GETREALS('TURCONST')) THEN
          CALL SUB3R(CMU,REALS(1),CD,REALS(2),C1E,REALS(3))
          CALL SUB3R(C2E,REALS(4),AK,REALS(5),EWAL,REALS(6))
```

which duly interprets what it finds and records its work by printing the following in the RESULT file:

```
 CMU = 6.000000E-01 CD  = 2.000000E-01
 C1E = 1.500000E+00 C2E = 1.920000E+00
 AK  = 4.000000E-01 EWAL= 9.000000E+00
```

This example has shown how **real** variables may be transmitted. **Integer** and **logical** variables are transmitted by use of READINTS and READLOGS statements in the corresponding manner.

10.2. Discussion

The following points need to be understood by those wishing to exploit this powerful new means of transmitting simulation-defining data:

- Of course, inserting (READREALS etc) in a Q1 file is of no effect unless there exist corresponding GETREALS in the EARTH coding; and at present (April 2005) there are few of these.

  The new technique is therefore mainly of use to those having access to re-compilable versions of PHOENICS.

- Whereas the READQ1 facility did involve the reading of Q1 by the EARTH module, this is not true of In-Form's READxxx; for this uses EARDAT as the information-transmitting file.

- It also makes use of SPEDAT (which is true of all In-Form features); but it differs from the conventional uses of SPEDAT for transmitting single real, integer and logical values both:
    - in **capability**, in that it can convey **many** values, and
    - in **method**, in that it uses SPEDAT in **character-string** mode, which is signalled by the C in the Q1EAR, RESULT and EARDAT lines).
- The GETREALS, GETINTS and GETLOGS logical functions make no protest if, when they inspect the SPEDAT transmissions, they find no entry corresponding to their first argument (namely 'TURCONST' in the above example). They simply return their logical value as .FALSE. .

11. In- Form and the VR-Editor

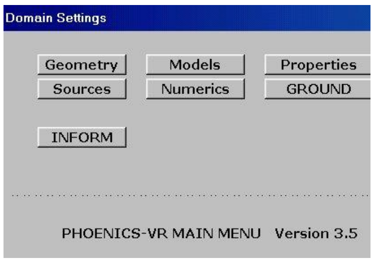
11.1. Historical background

- In-Form was conceived as an **extension** to the PHOENICS Input Language, PIL. The VR-Editor, on the other hand, was conceived as a **replacement** for PIL, and moreover that of earlier years.
- Even as a replacement, the VR-Editor has some deficiencies; for example it lacks the ability to 'declare' variables, or even to retain declarations already in the Q1.
- As has been seen, In-Form makes much use of declared-and-set character variables, which the VR-Editor could not retain.
- It is for this reason that **informXbegin** and **informXend** markers have had to be placed at the head and end of every set of In-Form statement, if the Q1 is to be processed by the VR-Editor.
- **X**, in these markers, is a number indicating to the VR-Editor into which 'Group' it should place the In-Form statements when writing an end-of-session Q1, namely:
    - o Group **7** for storing whole-field variables, whether stored-only or also solved;
    - o Group **9** for property-related settings,
    - o Group **11** for initial values, and
    - o Group **13** for sources and boundary conditions.
- However, this protection still leaves open the question: How can the user see and, if desired, modify, the In-Form settings?

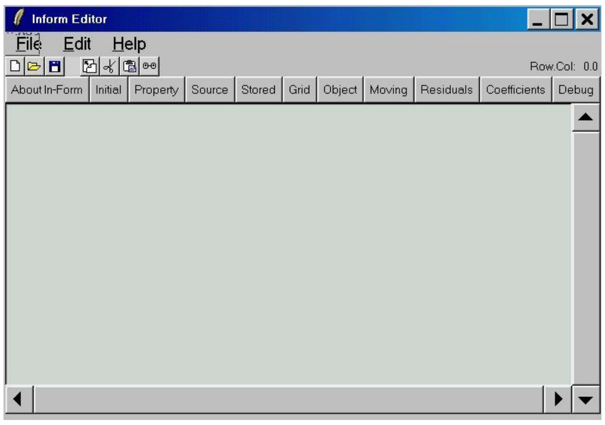
11.2. Editing In-Form statements via the menu

The question has been answered by providing "In-Form" buttons in all appropriate menu panels of the VR-Editor, which access a newly-created 'In-form editor'.

One such button is shown here, for the top panel.



When that button is pressed, the Inform Editor appears.

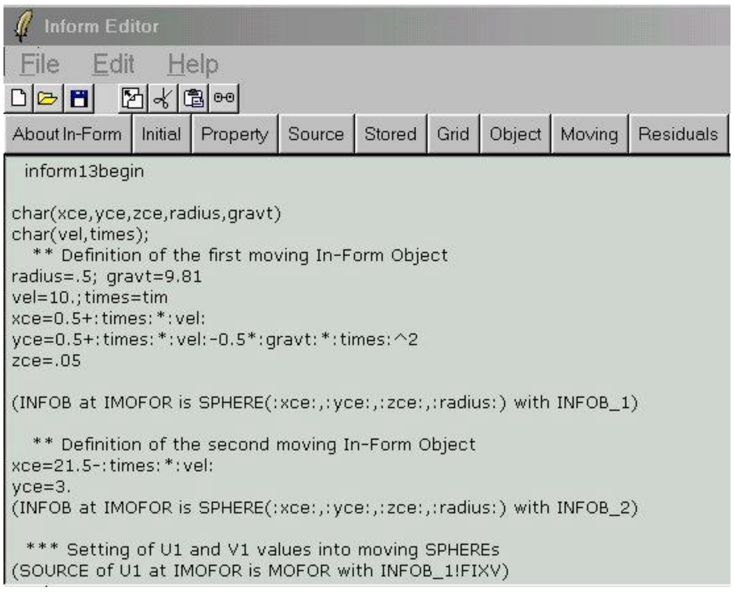


Similar buttons appear on lower-level panels of the VR-Editor user-interface.

11.3. Example; library case 360

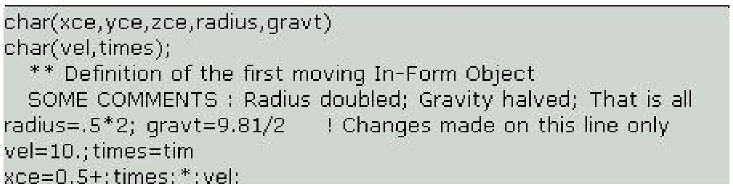
There now follow several screen-capture images which show stages in the editing of the moving-body trajectory.

The existing Group 13 statements are viewed here



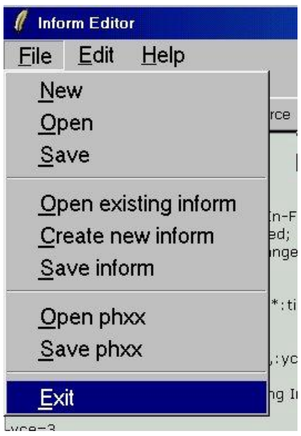
```
 inform13begin

char(xce,yce,zce,radius,gravt)
char(vel,times);
   ** Definition of the first moving In-Form Object
radius=.5; gravt=9.81
vel=10.;times=tim
xce=0.5+:times:*:vel:
yce=0.5+:times:*:vel:-0.5*:gravt:*:times:^2
zce=.05

(INFOB at IMOFOR is SPHERE(:xce:,:yce:,:zce:,:radius:) with INFOB_1)

   ** Definition of the second moving In-Form Object
xce=21.5-:times:*:vel:
yce=3.
(INFOB at IMOFOR is SPHERE(:xce:,:yce:,:zce:,:radius:) with INFOB_2)

  *** Setting of U1 and V1 values into moving SPHEREs
(SOURCE of U1 at IMOFOR is MOFOR with INFOB_1!FIXV)
```

They are then changed by way of Notepad-like editing actions to this.



```
char(xce,yce,zce,radius,gravt)
char(vel,times);
  ** Definition of the first moving In-Form Object
  SOME COMMENTS : Radius doubled; Gravity halved; That is all
radius=.5*2; gravt=9.81/2    ! Changes made on this line only
vel=10.;times=tim
xce=0.5+:times:*:vel:
```

When they are what the user wants, they are saved.



This action places them in the phxx file which the VR Editor uses to store items which must be written to the final Q1, the relevant part of which is seen here:

```
  Echo InForm settings for Group 13
  inform13begin
char(xce,yce,zce,radius,gravt)
char(vel,times);
   ** Definition of the first moving In-Form Object
   SOME COMMENTS : Radius doubled; Gravity halved; That is all
radius=.5*2; gravt=9.81/2 ! Changes made on this line only
vel=10.;times=tim
xce=0.5+:times:*:vel:
```

11.4. The future of the Inform Editor

- The VR and In-Form Editors work satisfactorily together.
- However, it will take some time for the maturity of the former to be emulated by the latter, and for the In-Form Editor's greater flexibility to be utilised by VR.
- The In-Form Editor needs at least two additions before it can be considered mature, namely:
    1. It must enable suitable formulae to be selected from lists in drop-down menus; and
    2. it must subject what is typed in to immediate validity checks, and then reject, with advice, syntactical errors.
- What it can offer to the VR-Editor is the ability to handle:
    - PIL declarations,
    - IF statements and
    - DO loops; and indeed
    - all the excellent PIL features which the current VR-Editor disregards.

  Now that the PHOENICS Commander, which also employs the Tcl/Tk package, has been adopted as the main introduction to and manager of PHOENICS, these additions to the In-Form Editor will swiftly be made.

Appendix 1. In-Form functions

Operators

+- operation of addition;

-- operation of subtraction;

*- operation of multiplication;

/- operation of division;

^- operation of raising to a power;

Functions

ALL

function sets BOX In-Form object which fills the whole boundary box. The function has no arguments.

ABS(arg1)

function of calculation of absolute value. The function has 1 argument which is a constant or a stored variable. Examples are 731, 734, 781 and 784 library cases.

ACOS(arg1)

function of calculation of arccosine. The function has 1 argument which is a constant or a stored variable.

AECO(arg1)

function gets east coefficients of finite-volume equations for arg1 solved variable. The function has 1 argument which is a solved variable. Example is 703 library case.

AHCO(arg1)

function gets high coefficients of finite-volume equations for arg1 solved variable. The function has 1 argument which is a solved variable.

ALCO(arg1)

function gets low coefficients of finite-volume equations for arg1 solved variable. The function has 1 argument which is a solved variable.

ANCO(arg1)

function gets north coefficients of finite-volume equations for arg1 solved variable. The function has 1 argument which is a solved variable. Example is 703 library case.

APCO(arg1)

function gets AP coefficients of finite-volume equations for arg1 solved variable. The function has 1 argument which is a solved variable. Example is 703 library case.

ARR(arg1)

function of calculation of "Arrhenius" value from next formulae

EXP(arg1/(R*T))

where R is the universal gas constant = 8314.31, T is the absolute temperature and arg1 is a constant or a stored variable.

ASCO(arg1)

function gets south coefficients of finite-volume equations for arg1 solved variable. The function has 1 argument which is a solved variable. Example is 703 library case.

ASIN(arg1)

function of calculation of arcsine. The function has 1 argument which is a constant or a stored variable.

ATAN(arg1)

function of calculation of arctangent. The function has 1 argument which is a constant or a stored variable.

AWCO(arg1)

function gets west coefficients of finite-volume equations for arg1 solved variable. The function has 1 argument which is a solved variable. Example is 703 library case.

BOX(arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9)

function sets BOX In-Form object. The function has 9 arguments which are constants or stored variables and where

arg1	- X-coordinate of west south low corner of box, m;
arg2	- Y-coordinate of west south low corner of box, m;
arg3	- Z-coordinate of west south low corner of box, m;
arg4	- X-size of box side, m;
arg5	- Y-size of box side, m;
arg6	- Z-size of box side, m;
arg7	- angle rotating around x axis, rad;
arg8	- angle rotating around y axis, rad;
arg9	- angle rotating around z axis, rad.

Examples are 769, 770, 783 and 784 library cases.

CORR(arg1)

function gets corrections values for arg1 solved variable. The function has 1 argument which is a solved variable. Examples are 768 and 249 library cases.

COS(arg1)

function of calculation of cosine. The function has 1 argument which is a constant or a stored variable in radian. Examples are 712, 768, 770 and 783 library cases.

COSH(arg1)

function of calculation of hyperbolic-cosine by next formula

0.5*(exp(arg1)+exp(-arg1))

The function has 1 argument which is a constant or a stored variable.

EAST(arg1)

function of calculation of value at neighbouring cell beside east face. The function has 1 argument which is a stored variable only. "EAST(arg1)" is equivalent of "arg1[+1]". Examples are 367 and 368 library cases.

EXP(arg1)

function of calculation of exponential value. The function has 1 argument which is a constant or a stored variable. Examples are 089, 700, 711, 712, 713, 720 and 740 library cases.

FACET(arg1)

function sets In-Form object from facetted object. The function has 1 arguments which is a character name of facetted object.

GAMM(arg1)

function gets exchange coefficients for arg1 solved variable. The function has 1 argument which is a solved variable. Example is 788 library case.

HIGH(arg1)

function of calculation of value at neighbouring cell beside high face. The function has 1 argument which is a stored variable only. "HIGH(arg1)" is equivalent of "arg1[,,+1]". Examples are 710, 722 and 735 library cases.

LOG10(arg1)

function of calculation of the Napierian logarithm, with base 10.0. The function has 1 argument which is a constant or a stored variable. Example is 089 library case.

LOGE(arg1)

function of calculation of the Napierian logarithm, with base "e". The function has 1 argument which is a constant or a stored variable.

LOW(arg1)

function of calculation of value at neighbouring cell beside low face. The function has 1 argument which is a stored variable only. "LOW(arg1)" is equivalent of "arg1[,,-1]". Examples are 366, 710, 722 and 735 library cases.

MAX(arg1,arg2)

function of maximum calculation. The function has 2 arguments which are constants or stored variables. Examples are 706, 778, 779 and 781 library cases.

MIN(arg1,arg2)

function of minimum calculation. The function has 2 arguments which are constants or stored variables. Examples are 706, 740, 741, 778 and 779 library cases.

NORTH(arg1)

function of calculation of value at neighbouring cell beside north face. The function has 1 argument which is a stored variable only. "NORTH(arg1)" is equivalent of "arg1[,+1]".

OFFSET(arg1,arg2,arg3)

function describes the hierarchy part of the MOFOR attribute settings. OFFSET is In-Form special function of declaring the frames of reference of object-related coordinate systems. Each new OFFSET function declares a new frame coordinate system and describes its position relative to its parent system. The arg1 ,arg2 and arg3 are formulas for calculation of coordinates of the origin position of the rotation axis relative to its parent.

OLD(arg1)

function of calculation of value at previous time step. The function has 1 argument which is a stored variable only. Examples are 368 and 786 library cases.

POL2(arg1,arg2,arg3,arg4)

function of calculation of the polynomial by next formula

arg2+arg1*(arg3+arg1*arg4)

The function has 4 arguments where arg1 may be a constant or a stored variable, but arg2, arg3 and arg4 must be constants. Example is 089 library case.

POL3(arg1,arg2,arg3,arg4,arg5)

function of calculation of the polynomial by next formula

arg2+arg1*(arg3+arg1*(arg4+arg1*arg5))

The function has 5 arguments where arg1 may be a constant or a stored variable, but arg2, arg3, arg4 and arg5 must be constants. Examples are 089, 701 and 763 library cases.

POL4(arg1,arg2,arg3,arg4,arg5,arg6)

function of calculation of the polynomial by next formula

arg2+arg1*(arg3+arg1*(arg4+arg1*(arg5+arg1*arg6)))

The function has 6 arguments where arg1 may be a constant or a stored variable, but arg2, arg3, arg4, arg5 and arg6 must be constants. Example is 089 library case.

POL5(arg1,arg2,arg3,arg4,arg5,arg6,arg7)

function of calculation of the polynomial by next formula

arg2+arg1*(arg3+arg1*(arg4+arg1*(arg5+arg1*(arg6+arg1*arg7))))

The function has 7 arguments where arg1 may be a constant or a stored variable, but arg2, arg3, arg4, arg5, arg6 and arg7 must be constants. Examples are 089 and 760 library cases.

POL6(arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8)

function of calculation of the polynomial by next formula

arg2+arg1*(arg3+arg1*(arg4+arg1*(arg5+arg1*(arg6+arg1*(arg7+arg1*arg8)))))

The function has 8 arguments where arg1 may be a constant or a stored variable, but arg2, arg3, arg4, arg5, arg6, arg7 and arg8 must be constants. Example is 089 library case.

POS(Xpos,Ypos,Zpos,Xang,Yang,Zang)

function sets the position and rotation of moving VR object by In-Form. This function has next parameters:

Xpos, Ypos, Zpos are formulas for setting X, Y, Z coordinate of moving VR object position, meters;

Xang, Yang, Zang are formulas for setting the rotation angle of moving VR object about X, Y, Z axis, degrees.

All parameters can be function of TIM variable which is current time in seconds at the current time step. This function is used in (MOVOB In-Form statement only.

PWL3(arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9)

signifies a piece-wise-linear function, with three parts. The syntax of PWL3 is following constructed. Thus, PWL3 is followed by a bracket, then the name of the variable in question, followed by a comma or an ampersand. Thereafter follow pairs of numbers, of which the first is the abscissa and the second is the corresponding ordinate, thus:

PWL3(x , x0 , y0 , x1 , y1 , x2 , y2 , x3 , y3)

represents y as a function of x consisting of straight lines which pass through (x0,y0), (x1,y1),(x2,y3),(x3,y3).

The function has 9 arguments where arg1 may be a constant or a stored variable, but arg2, arg3, arg4, arg5, arg6, arg7, arg8 and arg9 must be constants. Example is 763 library case.

PWLF(arg1,arg2)

signifies a piece-wise linear function of which the defining points are specified in a file. The function has 2 arguments. The syntax of PWLF is followed by an opening bracket, the name of the file (with full path, if it is not in the working directory), a comma, the name of the independent variable, and a closing bracket. Example is 763 library case.

RESI(arg1)

function gets residuals values for arg1 solved variable. The function has 1 argument which is a solved variable. Example is 768 library case.

SIN(arg1)

function of calculation of sine. The function has 1 argument which is a constant or a stored variable in radian. Examples are 712, 717, 720, 768, 770 and 783 library cases.

SINH(arg1)

function of calculation of hyperbolic-sine by next formula

0.5*(exp(arg1)-exp(-arg1))

The function has 1 argument which is a constant or a stored variable.

SOUTH(arg1)

function of calculation of value at neighbouring cell beside south face. The function has 1 argument which is a stored variable only. "SOUTH(arg1)" is equivalent of "arg1[,-1]".

SPHERE(arg1,arg2,arg3,arg4)

function sets SPHERE In-Form object. The function has 4 arguments which are constants or stored variables and where

arg1 - X-coordinate of a sphere centre, m;

arg2 - Y-coordinate of a sphere centre, m;

arg3 - Z-coordinate of a sphere centre, m;

arg4 - radius of a sphere, m.

Examples are 360, 765, 766, 767, 768 and 772 library cases.

SPL5(arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10,arg11)

signifies a cubic-interpolation spline function passing through five points. The syntax of SPL5 is following constructed. Thus, SPL5 is followed by a bracket, then the name of the variable in question, followed by a comma or an ampersand. Thereafter follow pairs of numbers, of which the first is the abscissa and the second is the corresponding ordinate, thus:

SPL5(x , x0 , y0 , x1 , y1 , x2 , y2 , x3 , y3 , x4 , y4 , x5 , y5)

represents y as a spline function of x which pass through (x0,y0), (x1,y1), (x2,y2), (x3,y3), (x4,y4), (x5,y5)

The function has 11 arguments where arg1 may be a constant or a stored variable, but arg2, arg3, arg4, arg5, arg6, arg7, arg8, arg9, arg10 and arg11 must be constants. Example is 763 library case.

SQRT(arg1)

function of calculation of value in 0.5 degree. The function has 1 argument which is a constant or a stored variable. Examples are 709, 726, 779, 780, 781, 783 and 785 library cases.

SSUM(arg1)

function of summation at one current IZ slab. The function has 1 argument which is a constant or a stored variable.

SUM(arg1)

function of global summation at all IZ slabs. The function has 1 argument which is a constant or a stored variable. Examples are 781, 785 and 786 library cases.

TAN(arg1)

function of calculation of tangent. The function has 1 argument which is a constant or a stored variable in radian.

TANH(arg1)

function of calculation of hyperbolic-tangent by next formula

(exp(arg1)-exp(-arg1))/(exp(arg1)+exp(-arg1))

The function has 1 argument which is a constant or a stored variable.

WEST(arg1)

function of calculation of value at neighbouring cell beside west face. The function has 1 argument which is a stored variable only. "WEST(arg1)" is equivalent of "arg1[-1]". Examples are 367, 368, 710, 722 and 735 library cases.

Appendix 2. The list of In-Form statements

[ ] - designation of a dispensable element of a statement.

```
(SOUR[CE] of Var at PatchName is Formula [with Options])
```

sets a source by Formula calculated for Var variable at region described by PATCH command with PatchName name. Var is 3D-stored solved variable. The "with Options" element contains options which specify the action of statement:

FIXFLU (by default) – fixed source

FIXVAL	- fixed value
ONLYMS	- value is convected in by associated mass source
LAMWALL	- laminar wall
BLASIUS	- turbulent wall using Blasius wall-function
LOGLAW	- turbulent wall using logarithmic wall-function
LINE	- linearised source
INFOB_n	- for moving InForm object
IMAT=iprp	- for cells with material number iprp
IF(condition)	- general logical condition

```
(INIT[IAL] of Var [at PatchName] is Formula [with Options])
```

sets the initialization of Var variable at region described by PATCH command with PatchName name by Formula calculated. Var is any 3D-stored variable. The "with Options" element contains options which specify the action of statement:

INFOB_n	- at moving InForm object
IMAT=iprp	- for cells with material number iprp
IF(condition)	- general logical condition

```
(STOR[ED] of Var [at PatchName] is Formula [with Options])
```

sets a account of Var variable at region described by PATCH command with PatchName name by Formula calculated. Var is any 3D-stored variable. If Var is a single user-real then this format should be used

```
  (STORE1 of Var [at PatchName] is Formula [with Options])
```

The "with Options" element contains options which specify the action of statement:

ZSLFIN (by default) – end of IZ slab

ZSLSTR	- start of IZ slab
SWPSTR	- start of sweep
SWPFIN	- end of sweep
TSTSTR	- start of time step
TSTFIN	- end of time step
RESIDU	- ar residual-calculation time

CORREC	- at correction-calculation
INFOB_n	- for moving InForm object
IMAT=iprp	- for cells with material number iprp
IF(condition)	- general logical condition

```
(PROP[ERTY] of Var [at PatchName] is Formula [with Options])
```

sets a property values of Var variable at region described by PATCH command with PatchName name by Formula calculated. Var is any variable name from this list:

RHO1 or DEN1 - first-phase density;

DRH1DP - differential of the logarithm of the first-phase density;

RHO2 or DEN2 - second-phase density;

DRH2DP - differential of the logarithm of the first-phase density;

ENUT or VIST - turbulent kinematic viscosity;

ENUL or VISL - laminar kinematic viscosity;

PRNDTL(var_name) - Prandtl Number;

PHINT(var_name) - interface value for a first- or second-phase variable;

TMP1 or TEMP - temperature of the first phase;

TMP2 - temperature of the second phase;

EL1 or LEN1 - mixing length-scale of the first-phase fluid;

EL2 or LEN1 - mixing length-scale of the second-phase fluid;

CP1 or SPH1 - specific heat of the first phase;

CP2 or SPH2 - specific heat of the second phase;

DVO1DT - first-phase volumetric thermal-expansion coefficient;

DVO2DT - second-phase volumetric thermal-expansion coefficient;

CFIPS or CFIP or INTF - interphase friction coefficient;

CMDOT or MDOT or CMDO or INTM - interphase mass-transfer rate;

CINT(var_name) - phase-to-interface transfer coefficient;

CVM - virtual-mass coefficient;

where var_name is 3D-stored solved variable. The "with Options" element contains options which specify the action of statement:

INFOB_n	- at moving InForm object
IMAT=iprp	- for cells with material number iprp
IF(condition)	- general logical condition

```
(INFO[B] [at PatchName] is Formula with INFOB_n[!Options])
```

sets formula-set object with "n" number by Formula calculated at region described by PATCH command with PatchName name. "n" is positive integer number less than 9. The Formula can contains ALL, BOX(), FACET() and/or SPHERE() function. The "with Options" element contains options which specify the action of statement:

INFOB_n	- at moving InForm object

IMAT=iprp	- for cells with material number iprp
IF(condition)	- general logical condition

```
(LONG[NAME] of Var [at PatchName] is Formula [with Options])
```

sets a account of Var variable at region described by PATCH command with PatchName name by Formula calculated at moment before print-out. For a designation variable field is used a expression of the formula. Var is any 3D-stored variable. The "with Options" element contains options which specify the action of statement:

INFOB_n	- at moving InForm object
IMAT=iprp	- for cells with material number iprp
IF(condition)	- general logical condition

If it is necessary only to use a long name without accounts by the Formula the following statement is possible

```
  (LONGNAME of Var is Long_Name)
```

where Long_Name is characters string without blanks.

```
(TGRI[D] of Var [at PatchName] is Formula [with Options])
```

sets a account of Var variable at region described by PATCH command with PatchName name by Formula calculated. The account begin at moment after calling of second group of GROUND if TLAST=GRND. Var is any 3D-stored variable or some EARTH reals. The "with Options" element contains options which specify the action of statement:

INFOB_n	- at moving InForm object
IMAT=iprp	- for cells with material number iprp
IF(condition)	- general logical condition

```
(XGRI[D] of Var [at PatchName] is Formula with Options)
```

sets a account of Var variable at region described by PATCH command with PatchName name by Formula calculated. The account begin at moment after calling of third group of GROUND if AZXU=GRND. Var is any 3D-stored variable or some EARTH reals. The "with Options" element contains options which specify the action of statement:

INFOB_n	- at moving InForm object
IMAT=iprp	- for cells with material number iprp
IF(condition)	- general logical condition

```
(YGRI[D] of Var [at PatchName] is Formula with Options)
```

sets a account of Var variable at region described by PATCH command with PatchName name by Formula calculated. The account begin at moment after calling of fourth group of GROUND if AZYV=GRND. Var is any 3D-stored variable or some EARTH reals. The "with Options" element contains options which specify the action of statement:

INFOB_n	- at moving InForm object
IMAT=iprp	- for cells with material number iprp
IF(condition)	- general logical condition

`(ZGRI[D] of Var [at PatchName] is Formula with Options)`

sets a account of Var variable at region described by PATCH command with PatchName name by Formula calculated. The account begin at moment after calling of fifth group of GROUND if AZDZ=GRND. Var is any 3D-stored variable or some EARTH reals. The "with Options" element contains options which specify the action of statement:

INFOB_n	- at moving InForm object
IMAT=iprp	- for cells with material number iprp
IF(condition)	- general logical condition

`(MAKE of Var is Formula)`

declare storage of VAR single real variable. Its initial values are calculated by Formula.

`(MOVOB of VRObName is Formulas [with PARENT=VRPar_Name])`

sets a position of moving VR object with VRObName name by Formulas calculated. This statement is used for the MOFOR purposes. The PARENT=VRPar_Name option set the name of parent reference of object-related coordinate system.

`(PRIN[T] String_15 [at PatchName] is Formula)`

dumps the fields of dependent variables or user-defined single real variables calculated by Formula in INFOROUT file of which places in the working directory. The String_15 string is a comment of dumping variables. String_15 is a character name by a length of 15 symbols. PatchName is a name of PATCH or VR object.

`(REAL[READ] SetName is RealVarSet)`

transmits to EARTH the real variables set with SetName name by means SPEDAT lines. SetName is a character string by a length of 15 symbols. RealVarSet is the set of real numbers separated by blanks among themselves.

`(INTR[EAD] SetName is IntegerVarSet)`

transmits to EARTH the integer variables set with SetName name by means SPEDAT lines. SetName is a character string by a length of 15 symbols. RealVarSet is the set of integer numbers separated by blanks among themselves.

`(LOGR[EAD] SetName is LogicalVarSet)`

transmits to EARTH the logical variables set with SetName name by means SPEDAT lines. SetName is a character string by a length of 15 symbols. RealVarSet is the set of logical numbers (T or F) separated by blanks among themselves.

`(MODS[OL] of Var [at PatchName] is Formula)`

sets a recalculation of Var variable at region described by PATCH command with PatchName name recalculated by Formula. The account begin at moment after getting equation solution. Var is any 3D-stored variable or some EARTH reals.

```
(MODG[EOM] of Var [at PatchName] is Formula)
```

sets a recalculation of the volume and areas modification at region described by PATCH command with PatchName name recalculated by Formula. Var variable can be VOLU name

for cell volume or EARE, NARE, HARE names for east, north, high cell areas. The account begin at moment after account of all geometrical parameters at each time step.

Appendix 3. In-Form options

The following paragraphs explain the significance of the 'options' which may appear in In-Form statements following 'with' or '!'.

The options are introduced in alphabetical order.

- AREA dictates that the source should be calculated as 'per unit area' of a plane VR object.
- BLAS dictates that the source calculated by the formula is to be multiplied by the surface area times the relevant effective exchange coefficient (viscosity for velocities, thermal conductivity for temperature, etc), calculated from the Blasius formula.

  This option is used only for SOURCE In-Form statement and for patches of wall type.
- CORREC dictates that that the formula should be evaluated at the moment of calculation of corrections.

  This option is used only for STORED In-Form statement and when the formula contains the CORR()function.
- EQCOEF dictates that the formula should be evaluated at the moment of calculation of the coefficients of the finite-volume equations .

  This option is used only for the STORED In-Form statement and when the formula contains AECO(), AWECO(), ANCO(), ASCO(), AHCO(), ALCO() or APCO() function.
- FIXFLU dictates that the source is of a fixed magnitude.

  This option is used only for the SOURCE In-Form statement, for which it is the default condition.
- FIXVAL dictates that the source is such as to fix the value of the relevant dependent variable, which is achieved by making the source equal to:
  2.0e+10 * (the fixed value minus the current value)

  This option is used only for the SOURCE In-Form statement.
- GAMCOF dictates that the formula should be evaluated at the moment of calculation of the exchange coefficients .

  This option is used only for the STORED In-Form statement and when the formula contains the GAMM() function.
- IF(condition) dictates that the formula should be evaluated only when and where "condition" is true. The format of the "condition" expression is the same as that of Fortan.

  This option can be used for all In-Form statements.
- IMAT=iprp dictates that the formula should be evaluated only for cells where the PRPS variable, which indicates the material index IMAT, is equal to a prescribed value: iprp.

  This option can be used for all In-Form statements. Other possible relations between IMAT and iprp may be expressed as:
  - IMAT>iprp greater than iprp
  - IMAT<iprp less than value iprp
  - IMAT>=iprp greater than or equal to iprp
  - IMAT<=iprp less than or equal to iprp
  - IMAT!=iprp not equal to iprp

- INFOB_n dictates that the formula should be evaluated only for cells occupied by the In-Form formula-set object with "n" number. "n" is a positive integer number.

  This option can be used for all In-Form statements.
- LAMW dictates that the source calculated by the formula is to be multiplied by the surface area times the relevant laminar exchange coefficient (viscosity for velocities, thermal conductivity for temperature, etc).

  This option is used only for SOURCE In-Form statement and for patches of wall type.
- LINE dictates that the value of the source is to be linearised with respect to the dependent variable in question.

  This option is used only for SOURCE In-Form statement. At should be used whenever the formula implies that the source diminishes when the dependent variable increases.
- LOGL dictates that the source calculated by the formula is to be multiplied by the surface area times the relevant effective exchange coefficient (viscosity for velocities, thermal conductivity for temperature, etc), calculated from the logarithmic-profile formula.

  This option is used only for SOURCE In-Form statement and for patches of wall type.
- ONLYMS dictates that the source calculated by the formula is to be multiplied by the inflowing mass of the relevant phase.

  This option is used only for SOURCE In-Form statement.
- RESIDU dictates that the formula should be evaluated at the moment of calculation of the residuals.

  This option is used only for then STORED In-Form statement and when the formula contains the RESI() function.
- SWPFIN dictates that the formula should be evaluated at the finish of the SWEEP loop.

  This option is used only for STORED In-Form statement.
- SWPSTR dictates that the formula should be evaluated at the start of the SWEEP loop.

  This option is used only for STORED In-Form statement.
- TSTFIN dictates that the formula should be evaluated at the finish of the time step.

  This option is used only for the STORED In-Form statement.
- TSTSTR dictates that the formula should be evaluated at the start of the time step.

  This option is used only for the STORED In-Form statement.
- VOLU dictates that the source should be calculated as 'per unit volume' of the relevant VR object.
- WHOL dictates that the source calculated by the formula is the total for whole VR object, distributed between cells in proportion to the volume of the cell which is occupied by the object.
- ZSLFIN dictates that the formula should be evaluated at the finish of the operations at the particular IZ slab.

  This option is used only for STORED In-Form statement, and is the default.
- ZSLSTR dictates that the formula should be evaluated before the start operations at the particular IZ slab.

  This option is used only for the STORED In-Form statement.