16.6 AVECTOR: A vector algebra and calculus package

This package provides REDUCE with the ability to perform vector algebra using the same notation as scalar algebra. The basic algebraic operations are supported, as are differentiation and integration of vectors with respect to scalar variables, cross product and dot product, component manipulation and application of scalar functions (e.g. cosine) to a vector to yield a vector result.

Author: David Harper.

16.6.1 Introduction

This package [2] is written in RLISP (the LISP meta-language) and is intended for use with REDUCE 3.4. It provides REDUCE with the ability to perform vector algebra using the same notation as scalar algebra. The basic algebraic operations are supported, as are differentiation and integration of vectors with respect to scalar variables, cross product and dot product, component manipulation and application of scalar functions (*e.g.* cosine) to a vector to yield a vector result.

A set of vector calculus operators are provided for use with any orthogonal curvilinear coordinate system. These operators are gradient, divergence, curl and del-squared (Laplacian). The Laplacian operator can take scalar or vector arguments.

Several important coordinate systems are pre-defined and can be invoked by name. It is also possible to create new coordinate systems by specifying the names of the coordinates and the values of the scale factors.

16.6.2 Vector declaration and initialisation

Any name may be declared to be a vector, provided that it has not previously been declared as a matrix or an array. To declare a list of names to be vectors use the VEC command:

```
VEC A,B,C;
```

declares the variables `A`, `B` and `C` to be vectors. If they have already been assigned (scalar) values, these will be lost.

When a vector is declared using the `VEC` command, it does not have an initial value.

If a vector value is assigned to a scalar variable, then that variable will automatically be declared as a vector and the user will be notified that this has happened.

[2] Reference: Computer Physics Communications, **54**, 295-305 (1989)

A vector may be initialised using the AVEC function which takes three scalar arguments and returns a vector made up from those scalars. For example

```
A := AVEC(A1, A2, A3);
```

sets the components of the vector A to A1, A2 and A3.

16.6.3 Vector algebra

(In the examples which follow, V, V1, V2 *etc* are assumed to be vectors while S, S1, S2 etc are scalars.)

The scalar algebra operators +,-,* and / may be used with vector operands according to the rules of vector algebra. Thus multiplication and division of a vector by a scalar are both allowed, but it is an error to multiply or divide one vector by another.

`V := V1 + V2 - V3;`	Addition and subtraction
`V := S1*3*V1;`	Scalar multiplication
`V := V1/S;`	Scalar division
`V := -V1;`	Negation

Vector multiplication is carried out using the infix operators DOT and CROSS. These are defined to have higher precedence than scalar multiplication and division.

`V := V1 CROSS V2;`	Cross product
`S := V1 DOT V2;`	Dot product
`V := V1 CROSS V2 + V3;`	
`V := (V1 CROSS V2) + V3;`	

The last two expressions are equivalent due to the precedence of the CROSS operator.

The modulus of a vector may be calculated using the VMOD operator.

```
S := VMOD V;
```

A unit vector may be generated from any vector using the VMOD operator.

```
V1 := V/(VMOD V);
```

Components may be extracted from any vector using index notation in the same way as an array.

```
V := AVEC(AX, AY, AZ);
V(0);                          yields AX
V(1);                          yields AY
V(2);                          yields AZ
```

It is also possible to set values of individual components. Following from above:

```
V(1) := B;
```

The vector V now has components AX, B, AZ.

Vectors may be used as arguments in the differentiation and integration routines in place of the dependent expression.

```
V := AVEC(X**2, SIN(X), Y);
DF(V,X);                              yields (2*X, COS(X), 0)
INT(V,X);                             yields (X**3/3, -COS(X), Y*X)
```

Vectors may be given as arguments to monomial functions such as SIN, LOG and TAN. The result is a vector obtained by applying the function component-wise to the argument vector.

```
V := AVEC(A1, A2, A3);
SIN(V);                        yields (SIN(A1), SIN(A2), SIN(A3))
```

16.6.4 Vector calculus

The vector calculus operators div, grad and curl are recognised. The Laplacian operator is also available and may be applied to scalar and vector arguments.

```
V := GRAD S;      Gradient of a scalar field
S := DIV V;       Divergence of a vector field
V := CURL V1;     Curl of a vector field
S := DELSQ S1;    Laplacian of a scalar field
V := DELSQ V1;    Laplacian of a vector field
```

These operators may be used in any orthogonal curvilinear coordinate system. The user may alter the names of the coordinates and the values of the scale factors. Initially the coordinates are X, Y and Z and the scale factors are all unity.

There are two special vectors : COORDS contains the names of the coordinates in the current system and HFACTORS contains the values of the scale factors.

The coordinate names may be changed using the COORDINATES operator.

```
COORDINATES R,THETA,PHI;
```

This command changes the coordinate names to R, THETA and PHI.

The scale factors may be altered using the SCALEFACTORS operator.

```
SCALEFACTORS(1,R,R*SIN(THETA));
```

This command changes the scale factors to 1, R and R SIN(THETA).

Note that the arguments of SCALEFACTORS must be enclosed in parentheses. This is not necessary with COORDINATES.

When vector differential operators are applied to an expression, the current set of coordinates are used as the independent variables and the scale factors are employed in the calculation. (See, for example, Batchelor G.K. 'An Introduction to Fluid Mechanics', Appendix 2.)

Several coordinate systems are pre-defined and may be invoked by name. To see a list of valid names enter

```
SYMBOLIC !*CSYSTEMS;
```

and REDUCE will respond with something like

```
(CARTESIAN SPHERICAL CYLINDRICAL)
```

To choose a coordinate system by name, use the command GETCSYSTEM.

To choose the Cartesian coordinate system :

```
GETCSYSTEM 'CARTESIAN;
```

Note the quote which prefixes the name of the coordinate system. This is required because GETCSYSTEM (and its complement PUTCSYSTEM) is a SYMBOLIC procedure which requires a literal argument.

REDUCE responds by typing a list of the coordinate names in that coordinate system. The example above would produce the response

```
(X Y Z)
```

whilst

```
GETCSYSTEM 'SPHERICAL;
```

would produce

```
(R THETA PHI)
```

Note that any attempt to invoke a coordinate system is subject to the same restric-

tions as the implied calls to COORDINATES and SCALEFACTORS. In particular, GETCSYSTEM fails if any of the coordinate names has been assigned a value and the previous coordinate system remains in effect.

A user-defined coordinate system can be assigned a name using the command PUTCSYSTEM. It may then be re-invoked at a later stage using GETCSYSTEM.

Example 5

We define a general coordinate system with coordinate names X,Y,Z and scale factors H1,H2,H3 :

```
COORDINATES X,Y,Z;
SCALEFACTORS(H1,H2,H3);
PUTCSYSTEM 'GENERAL;
```

This system may later be invoked by entering

```
GETCSYSTEM 'GENERAL;
```

16.6.5 Volume and Line Integration

Several functions are provided to perform volume and line integrals. These operate in any orthogonal curvilinear coordinate system and make use of the scale factors described in the previous section.

Definite integrals of scalar and vector expressions may be calculated using the DEFINT function.

Example 6

To calculate the definite integral of $\sin(x)^2$ between 0 and 2π we enter

```
DEFINT(SIN(X)**2,X,0,2*PI);
```

This function is a simple extension of the INT function taking two extra arguments, the lower and upper bounds of integration respectively.

Definite volume integrals may be calculated using the VOLINTEGRAL function whose syntax is as follows :

VOLINTEGRAL(integrand, vector lower-bound, vector upper-bound);

Example 7

In spherical polar coordinates we may calculate the volume of a sphere by integrating unity over the range r=0 to RR, θ=0 to PI, ϕ=0 to 2*π as follows :

```
VLB := AVEC(0,0,0);          Lower bound
VUB := AVEC(RR,PI,2*PI);     Upper bound in r, θ, φ respectively
VOLINTORDER := (0,1,2);      The order of integration
VOLINTEGRAL(1,VLB,VUB);
```

Note the use of the special vector VOLINTORDER which controls the order in which the integrations are carried out. This vector should be set to contain the number 0, 1 and 2 in the required order. The first component of VOLINTORDER contains the index of the first integration variable, the second component is the index of the second integration variable and the third component is the index of the third integration variable.

Example 8

Suppose we wish to calculate the volume of a right circular cone. This is equivalent to integrating unity over a conical region with the bounds:

z = 0 to H (H = the height of the cone)
r = 0 to pZ (p = ratio of base diameter to height)
phi = 0 to 2*PI

We evaluate the volume by integrating a series of infinitesimally thin circular disks of constant z-value. The integration is thus performed in the order : d(ϕ) from 0 to 2π, dr from 0 to p*Z, dz from 0 to H. The order of the indices is thus 2, 0, 1.

```
VOLINTORDER := AVEC(2,0,1);
VLB := AVEC(0,0,0);
VUB := AVEC(P*Z,H,2*PI);
VOLINTEGRAL(1,VLB,VUB);
```

(At this stage, we replace P*H by RR, the base radius of the cone, to obtain the result in its more familiar form.)

Line integrals may be calculated using the LINEINT and DEFLINEINT functions. Their general syntax is

LINEINT(vector-function, vector-curve, variable);

DEFLINENINT(vector-function, vector-curve, variable, lower-bound, upper-bound);

where

vector-function is any vector-valued expression;

vector-curve is a vector expression which describes the path of integration in terms of the independent variable;

variable is the independent variable;

lower-bound

`upper-bound` are the bounds of integration in terms of the independent variable.

Example 9

In spherical polar coordinates, we may integrate round a line of constant theta ('latitude') to find the length of such a line. The vector function is thus the tangent to the 'line of latitude', (0,0,1) and the path is `(0,LAT,PHI)` where `PHI` is the independent variable. We show how to obtain the definite integral *i.e.* from $\phi = 0$ to 2π :

```
DEFLINEINT(AVEC(0,0,1),AVEC(0,LAT,PHI),PHI,0,2*PI);
```

16.6.6 Defining new functions and procedures

Most of the procedures in this package are defined in symbolic mode and are invoked by the REDUCE expression-evaluator when a vector expression is encountered. It is not generally possible to define procedures which accept or return vector values in algebraic mode. This is a consequence of the way in which the REDUCE interpreter operates and it affects other non-scalar data types as well : arrays cannot be passed as algebraic procedure arguments, for example.

16.6.7 Acknowledgements

This package was written whilst the author was the U.K. Computer Algebra Support Officer at the University of Liverpool Computer Laboratory.