Visual Basic – Exercises

Introduction

We will begin our writing of procedures in VB by some simple exercises on user-defined functions. These cannot easily be produced by using the macro recorder, so are required to be written by directly typing the code into a module.

As an example of a simple user-defined function, consider the scalar (dot) product of two three-dimensional vectors, $\mathbf{a} = (a_1, a_2, a_3)$ and $\mathbf{b} = (b_1, b_2, b_3)$ in Cartesian coordinates. Mathematically this is given by

$$\mathbf{a} \cdot \mathbf{b} = (a_1 b_1 + a_2 b_2 + a_3 b_3).$$

The function procedure `ScalarProduct` given below carries out this calculation. The two vectors are given as the arguments `aVector` and `bVector`.

```
' All variables must be declared before use.
Option Explicit
' Start arrays at index 1 (instead of 0).
Option Base 1
'
Function ScalarProduct(aVector, bVector) As Single
' Calculates scalar product of two vectors A and B.
    Dim product As Single, i As Integer
    product = 0
    For i = 1 To 3
        product = product + aVector(i) * bVector(i)
    Next
    ScalarProduct = product
End Function
```

Note that the procedure statement

`product = product + aVector(i) * bVector(i)`

would be equivalent, in mathematics, to

$$x = x + a \times b,$$

with $a \neq 0$, $b \neq 0$ and would be nonsense. However the computing expression is NOT a mathematical statement, but is an assignment statement. Thus in the code statements such as `xVal = xVal + aVal` means, starting on the right-hand side, take the value in the memory location named `xVal` and add to it the value in the location named `aVal`, then put the result back into the location named `xVal`. Hence if this statement is executed repeatedly (in a loop) it accumulates into `xVal` successive values of `aVal`, i.e. it sums the values.

The code segment

```
For i = 1 To 3
   product = product + aVector(i) * bVector(i)
Next
```

executes a loop. Initially, *product = 0*, so for *i = 1* the line

```
product = product + aVector(i) * bVector(i)
```
computes *product = aVector(1) * bVector(1)*, equivalent to the mathematical statement $product = 0 + a_1 b_1$. The next time through the loop, *i* has the value 2, *product* has the value $a_1 b_1$ and the line evaluates $product = a_1 b_1 + a_2 b_2$. The final value of *i* is 3, and now on the RHS *product* has the value $a_1 b_1 + a_2 b_2$ and so the line computes finally $product = a_1 b_1 + a_2 b_2 + a_3 b_3$. The penultimate statement `ScalarProduct = product` assigns the value of `product` to the name of the function, `ScalarProduct`.

Exercise 1

You will create a suite of procedures based on the example above.
The length of a vector **a** is given by

$$|\mathbf{a}| = \sqrt{a_1^2 + a_2^2 + a_3^2}$$

The angle, *q*, between the vectors **a** and **b** can found from

$$\mathbf{a} \cdot \mathbf{b} = |a||b| \cos q$$

The vector product of two vectors **a** and **b** is a vector **c** with components given by

$$c_1 = a_2 b_3 - a_3 b_2$$
$$c_2 = a_3 b_1 - a_1 b_3$$
$$c_3 = a_1 b_2 - a_2 b_1$$

The volume, *V*, of the parallelepiped with sides formed from the three vectors **a**, **b**, and **c** is given by

$$V = |\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})|.$$

If the three vectors are coplanar, $V = 0$.

- Open a new (blank) workbook and start the VB Editor (if not already running) using Alt+F11. Select "Module" from the "Insert" menu. This will create a new module, called "Module1" by default, into which you should paste the **Function** procedure `ScalarProduct` given above.
- You can now use the function `ScalarProduct` in your spreadsheet. Enter some numbers into cells A1:A3 and B1:B3. In cell A5 enter =Sca`larProduct(A1:A3,B1:B3`. If done correctly the spreadsheet should compute the value of the scalar product of the two vectors defined by components in cells A1:A3 and B1:B3. Check the result by entering the formula for the scalar product into some cell in the spreadsheet. What happens if you enter =Sca`larProduct(A1:C1,A2:C2)` in cell B5?
- Write a **Function** procedure, `VectorLength`, to calculate the length of a vector. (Note the VB function for square root is Sqr.)
- Write a **Function** procedure, `VectorAngle`, to calculate the angle (in degrees) between the vectors **a** and **b** making use of calls to the procedures `ScalarProduct` and `VectorLength`. (Note VB does not have a function for $\cos^{-1} q$ so you have to access the one provided in Excel by using the statement `Application.Acos(cosineOfAngle`. Other Excel functions, such as that converting radians to degrees, can also be used by prefixing their names with "`Application.`".

- Select “Procedure” from the “Insert” menu. Make sure “Sub” is selected as the procedure type, and type “VectorProduct” in the “name” box. This will create the outline of a procedure called `VectorProduct`, which you can fill in with your own Visual Basic code.
- Using this outline, write a **Sub** procedure `VectorProduct` taking three arguments, which are arrays for the vectors **a**, **b** (as input) and **c** (as output). (Notice that it cannot be a **Function** procedure as it computes three quantities and so does not return a single value).
- Write a **Function** procedure `Volume`, with three arguments, that computes the volume of the parallelepiped with sides formed from three given vectors, making use of calls to the procedures `VectorProduct` and `ScalarProduct`.
- Enter some numbers in cells C1:C3. Treating A1:A3, B1:B3 and C1:C3 as the components of three vectors **a**, **b** and **c**, respectively, use your functions `VectorLength`, `VectorAngle` and `Volume` in your spreadsheet to calculate these quantities using these vectors.
- Verify that your results are correct by using formulae in the spreadsheet cells to calculate the same quantities without using your macros.

To solve a scientific problem it is often broken down into smaller, and hopefully, solvable elements. The programme elements, `ScalarProduct`, `VectorLength` and `VectorProduct`, which you have created can be used in such cases.

Problem 1

A straight line, in the direction of a vector **u** and passing through the point P with position vector $\mathbf{r}_1$ can be expressed as $\mathbf{r}(t) = \mathbf{r}_1 + \mathbf{u}t$ where t is a parameter. A second line $\mathbf{r}(t) = \mathbf{r}_2 + \mathbf{v}t$ passes through $\mathbf{r}_2$ and is parallel to **v**. If these two lines do not meet they are called skew lines. If the two lines are parallel, $\mathbf{u} \times \mathbf{v} = 0$, the shortest difference between them is given by

$$d_{skew} = \frac{\left|(\mathbf{r}_2 - \mathbf{r}_1) \times \mathbf{u}\right|}{\left|\mathbf{u}\right|} = \frac{\left|(\mathbf{r}_2 - \mathbf{r}_1) \times \mathbf{v}\right|}{\left|\mathbf{v}\right|}.$$

If the lines are not parallel the shortest distance between them can be calculated from

$$d_{skew} = \frac{(\mathbf{r}_2 - \mathbf{r}_1) \cdot (\mathbf{u} \times \mathbf{v})}{\left|\mathbf{u} \times \mathbf{v}\right|}.$$

Write a **Function** procedure to calculate d_{skew} for varying choices of $\mathbf{r}_1$, $\mathbf{r}_2$, **u** and **v**. It will be helpful to first write a **Sub** procedure to obtain the components of the difference between two vectors. Then use this along with your other programme elements to construct the procedure to solve the problem. Note you will have to test if the directions are parallel in order to know which formula to use.

Problem 2

If **u** and **v** are interpreted as velocities of objects travelling along the lines and t as time, the closest distance between the objects is given by

$$d_{\min}(t_{\min}) = \frac{\left|(\mathbf{v}-\mathbf{u})\times(\mathbf{r}_2-\mathbf{r}_1)\right|}{\left|\mathbf{v}-\mathbf{u}\right|},$$

and occurs at a time

$$t_{\min} = -\frac{(\mathbf{v}-\mathbf{u})\cdot(\mathbf{r}_2-\mathbf{r}_1)}{\left|\mathbf{v}-\mathbf{u}\right|^2},$$

provided that $\mathbf{u} \neq \mathbf{v}$. If $\mathbf{u} = \mathbf{v}$ then the separation of the objects is constant equal to its initial value $\left|\mathbf{r}_2 - \mathbf{r}_1\right|$ at $t = 0$.

Write a **Function** procedure to evaluate $d_{\min}$ and t. Ordinarily a function procedure can only return one value associated with its name but it could, of course, return either the value of $d_{\min}$ or t depending on the value of one of its arguments, e.g.

```
Function
Approach(positionR1,positionR2,uVector,vVector,index),
```

could return the value of $d_{\min}$ if index = 0 , and t if index = 1.

As a particular example: At some time , , $t = 0$ the coordinates of two aeroplanes with respect to an airport are $\mathbf{r}_1 = (20,5,0.6)$ and $\mathbf{r}_2 = (32,-5,2)$ km and they are moving with velocities defined by $\mathbf{u} = (0,-250,0)$ and $\mathbf{v} = (-394.45,61.33,-24.65)$ kph. (The second plane is approaching to land; the first is passing the air port.) Use these data in a spreadsheet and your macros to find the planes' closest distance of approach and the time taken to reach that position, i.e. find the maximum time the pilots have in which to avoid a "near-miss" incident.

Save your spreadsheet and its associated macros as "*username-Vectors*".