

Help for .NET Developers

Background

The .NET platform offers application developers the means to simplify the code base, reduce development time and make portable, interoperable code. Its ultimate goal is to provide users access to their information anywhere, anytime and on any platform or device. While we are used to accessing information or services on one machine or even on a local network, .NET aims to make it simple to write applications that can access information or services anywhere on the World Wide Web. Building upon the Java experiences, Microsoft has designed an entire technology in an integrated and coherent manner.

Issue - VB vs. VB.NET

NAG has followed the development of .NET since its beginnings with particular interest in the languages provided by Microsoft: C#, managed C++ and of course VB.NET. NAG is also looking closely at the third-party compilers, especially the various Fortran.Net compilers now available.

.NET requires each of these languages to generate the same intermediate code and to share a common runtime library so that different routines may be compiled in different languages before being combined into one assembled application. Because of this there have been some sharp changes in the VB language. In consequence VB.NET is different in a number of aspects from the traditional (VB6) language. If you use (or plan to use) a NAG Library, the changes in storage conventions for array storage and the changed default calling convention could affect your application.

VB.NET arrays are now stored by row, which means that a two by two array array A(1,1) - arrays are zero based in .NET - has elements A(0,0), A(0,1), A(1,0), A(1,1) in consecutive locations. In traditional VB the storage of arrays would have meant that A(0,0), A(1,0), A(0,1), A(1,1) were in consecutive locations. This is significant if you have been using third party DLLs within VB. Typically the actual argument passed to satisfy an array argument would be the first element of the array. Storage association would then allow the routine to correctly access other elements of the array. Fortran was the ideal language to complement traditional VB, for its storage convention was similar to that of VB; under VB.NET this is no longer the case and some matrix transposition has to take place if a Fortran DLL is to be used. The row-orientated languages such as C become much suitable for VB.NET.

Traditional VB users might also be taken aback when they realize that the default calling convention is by ByVal, rather than ByRef. This means that if you are used to declaring a routine in a DLL for use in VB using a statement of the form:

```
Declare Sub A02AAF Lib "DLL20DDS.DLL" (xxr As Double,xxi As Double, yr As Double,yi As Double)
```

then you must change this to

```
Declare Sub A02AAF Lib "DLL20DDS.DLL" (ByRef xxr As Double,ByRef xxi As Double, ByRef yr As Double,ByRef yi As Double)
```

for use in VB.NET. This is because the defaults have changed and because the DLL is Fortran based and therefore expects its arguments to be passed by address, not value.

NAG can provide you with the requisite VB.NET declaration statements for incorporation of the [NAG Fortran DLL](#) routines into your VB.NET program. Equally we can show you how easy it is to build managed C++ wrappers around either the C or Fortran DLLs to harness the full power of these libraries from within the .NET environment.