Building 64-bit versions of the Acoustic Toolbox on Windows

After evaluating several alternatives to installing a gfortran build environment that targets 64-bit Windows, MSYS2 was a clear alternative, because it integrates a linux-like shell environment (msys2) with the Mingw-64 toolchain. The installation of MSYS2 was simple and installing new packages for the environment was made easier by MSYS2’s integrated package manager “pacman.”

Following are the steps I took to install MSYS2 on a Windows 7 computer. Note that I did not have administrator rights so did not take advantage of the MSYS2 Windows installer, which may have offered a cleaner install experience and may avoid problems down the line. Thus far my non-administrator method appears to provide a valid enough environment to build and run the Acoustic Toolbox programs.

Steps to install MSYS2:

1. You need to get an archive utility that can handle .xz format. I downloaded PeaZip Portable for this task:
	1. <http://www.peazip.org/peazip-portable.html>
	2. After unzipping the download to a location on my local disk, I just double-clicked on the peazip app icon to run it.
2. Download the MSYS2 base archive from here:
	1. <https://sourceforge.net/projects/msys2/files/Base/x86_64/>
	2. I downloaded the latest base archive called msys2-base-x86\_64-20160719.tar.xz
3. Use your archive utility to extract the archive to a folder; I chose an existing folder c:\apps because I could not create a folder off of c: without admin rights. The files are then extracted into c:\apps\msys64
4. In your msys64 folder, double-click on the file msys2\_shell.cmd. Msys2 will go through some configuration steps, and when complete, tell you to shut the shell down and run again (close the window with the “x” rather than typing “exit” at the command line).
5. Double-click on msys2\_shell.cmd again. The msys2 shell window comes up.
6. Now update the package databases:
	1. pacman -Syuu
	2. Type Y when asked if you want to proceed with installation
7. We can now use pacman to request the mingw64 toolchain package
	1. Type pacman -S mingw-w64-x86\_64-toolchain



Type Y when asked to install. The packages are downloaded; in some cases the packages won’t download due to slow mirror servers. I just re-issue the pacman command until they all download without error.

A few subpackages never seemed to download so I excluded them from the install by using ^4 ^8 to exclude subpackages 4 and 8 (ada and objective c I assume).

Once the toolchain is downloaded and installed, there is one last package to install:

pacman -S make

Once “make” is installed you can close the shell window (again, close the window by clicking the “x” instead of typing “exit” at the command line).

Now if you want to work with the 64-bit compilers and build environment you double-click on the mingw64 application from the msys64 directory.

I copied AT to my MSYS2 home folder (/home/tim.haverland).

Before compiling the AT code, you must first adjust two parameters in the Field3D.F90 program, MaxM and MaxSet, in order to provide enough array space to handle the problem at hand. MaxM is the maximum number of modes, and MaxSet is the maximum number of “mode sets.”

Through trial and error, we determined that the appropriate values for the Cape Cod demo were MaxM=891 and MaxSet=109,975 (see Appendix 1). These constants can be adjusted to fit other problems, but due to the amount of “static” data space that is available in Windows (<2GB), the product of these two constants must be less than around 124.352 million.

An alternative that may make Field3d.f90 more flexible is to change two of the program’s largest arrays, k and PhiR, from static arrays to ALLOCATEABLE arrays. ALLOCATEABLE arrays are placed within a dynamic data segment in Windows, which when built as a 64-bit program, allows addressing of 8TB of data, or the max free physical/virtual RAM on the machine, whichever is less. You can then set conservative values for MaxM and MaxSet so you can run a variety of problems within the allocated arrays.

Setting large values for MaxM and MaxSet does not appear to automatically use up the allocated memory in Windows; rather, it appears that the memory is “reserved” but only used when accessed or updated, thus right-sizing your arrays to fit the problem at hand. For the case of 200Hz and 200m, Field3D was using about 1.5GB of RAM despite having over 16GB of RAM allocated to the arrays k and phiR (5,000 MaxM \* 200,000 MaxSet \* 8 bytes for COMPLEX data type \* 2 arrays).

To change the arrays to ALLOCATEABLE, the beginning of the Field3d.f90 program must be edited to look like the following:

PROGRAM FIELD3D

 ! Generate file of replica vectors

 ! Multiple receivers depths are handled inefficiently.

 USE SdRdRMod

 USE beampatternMod

 USE ElementMod

 IMPLICIT NONE

 INTEGER, PARAMETER :: SHDFile = 25, MaxM = 5000, MaxSet = 200000 ! Setting MaxM and MaxSet to values that will accommodate a wide variety of problems.

 INTEGER :: M( MaxSet ), MS( 3 ), Msource, Mlimit, iostat

 INTEGER :: iCorner1, isx, isy, isd1, ird1, itheta, NSets, iRec, IElementSource, INode, IdentifySourceElement

 REAL :: Rmin, Rmax, freq, atten = 0

 REAL (KIND=8) :: TStart, TEnd

 **!Moving k and phiR from static declared arrays to ALLOCATEABLE arrays**

 **!COMPLEX :: k( MaxM, MaxSet ), phiR( MaxM, MaxSet ), kS( MaxM, 3 ), phiST( MaxM, 3 )**

 **COMPLEX :: kS( MaxM, 3 ), phiST( MaxM, 3 )**

 **COMPLEX, ALLOCATABLE :: k( :, : ), phiR( :, : )**

 CHARACTER (LEN= 7) :: Option

 CHARACTER (LEN=80) :: Title, TitleEnv, FileRoot, SHDFileName

 CHARACTER (LEN=10) :: PlotType = 'TL '

 COMPLEX, ALLOCATABLE :: P( :, : ), phiS( :, :, : )

 CALL CPU\_TIME( Tstart ) ! log the start time

 ! get the file root for naming all input and output files

 ! should add some checks here ...

 CALL GET\_COMMAND\_ARGUMENT( 1, FileRoot )

 SHDFileName = TRIM( FileRoot ) // '.shd'

 ! Open the print file

 OPEN( UNIT = PRTFile, FILE = 'field3d.prt', STATUS = 'UNKNOWN', IOSTAT = iostat )

 WRITE( PRTFile, \* )

 WRITE( PRTFile, \* ) '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

 WRITE( PRTFile, \* ) 'Running FIELD3D'

 WRITE( PRTFile, \* ) 'Sums modes, producing pressure'

 WRITE( PRTFile, \* )

 CALL READIN( FileRoot, Title, Option, Mlimit, Rmin, Rmax ) ! Read in all the user input

 ALLOCATE( phiS( MaxM, Pos%Nsd, 3 ) )

 **!New ALLOCATE statement for k and phiR arrays**

 **ALLOCATE(k( MaxM, MaxSet ), phiR( MaxM, MaxSet ))**

Once Field3d.f90 has been edited if needed, all of the AT programs can be compiled.

To compile the tools, double click on the mingw64 application, then change your working directory to /home/<user>/at, and followed the “make” instructions from the SoundScape guide:

make clean

make all

make install

To run the AT programs outside of MSYS2 (from the Windows command prompt or from Matlab, add the location of the mingw64 bin folder (in my case C:\apps\msys64\mingw64\bin) to your Windows PATH variable.

This is usually done by choosing Start>Rt.-click on “Computer”>Choose Properties>Advanced system settings>Advanced tab>Environment variables. Then under System variables, highlight the Path variable and click Edit and append a “;” plus the path of your mingw64 bin path.

Appendix 1: Examining the maximum number of modes and “unique mode sets” for the Cape Cod Demo:

Two large arrays in the program Field3D, k and phiR, are sized based on two constants, MaxM (the maximum number of modes), and MaxSet (the maximum number of unique mode sets). These constants must be tailored to the size/complexity of the spatial ocean environment.

Laurel Henderson of HLS advised that the number of “nodes” in a problem (number of x-ordinates \* number of y-ordinates) is an upper limit for MaxSet. For the Cape Cod demo this is 300 \* 462 = 138,600. Upon examination of the code, however, it appears that MaxSet is actually compared against the number of “unique mode sets,” which can be less than the number of nodes. This can’t be estimated ahead of time but is reported in the .prt files produced by the Field3d program. For the Cape Cod demo the number of unique mode sets is 109,975. I have verified that field3d will run with a MaxSet of 109,975 but fail with a MaxSet of 190,974 (“Too many mode sets” reported by ReadAllModeSets).

MaxM can’t be estimated ahead of time but can be reported precisely after running the Kraken program on all .env files to create .mod files. I wrote a short Matlab script that cycles through all .mod files in a folder and reports the maximum “M” value (number of modes). Here are the M values for three frequencies:

50Hz: 222

100Hz: 446

200Hz: 891

This script takes about an hour to run for each frequency, so it’s a significant price to pay to get the max number of modes.

Similar to MaxSet, I have confirmed that field3d will produce errors with a MaxM of 890 (“too many modes”) but succeed with a MaxM of 891.

A more flexible solution described above is to change the k and phiR arrays to ALLOCATEABLE, and set MaxM and MaxSet to generous values that will accommodate a wide variety of problems.