A Dynamic Library Implementation of the Lake States STEMS Growth Model

by

Thomas E. Burk

August 2013

Staff Paper Series No. 222

Department of Forest Resources

College of Food, Agricultural and Natural Resource Sciences
University of Minnesota
St. Paul, Minnesota

For more information about the Department of Forest Resources and its teaching, research, and outreach programs, contact the department at:

Department of Forest Resources
University of Minnesota
115 Green Hall
1530 Cleveland Avenue North
St. Paul, MN 55108-6112
Ph: 612.624.3400
Fax: 612.625.5212
Email: forest.resources@umn.edu
http://www.forestry.umn.edu/publications/staffpapers/index.html

Introduction

The STEMS distance independent, individual tree growth model (Belcher et al. 1982) is widely used in Lake States forestry practice. A number of computer implementations of the model are available including TWIGS (Miner et al. 1988) and most recently LS-FVS (Dixon and Keyser 2013). Most computer implementations force a user to adopt, or convert to, a particular input format and aren't general in the sense of fitting neatly into an organization's existing information technology software infrastructure. Here we present C++ code implementing the STEMS model that can be compiled into a Windows Dynamic Linked Library (DLL) or a Mac OS X dynamic library or a Unix shared object and utilized from any number of software platforms. We illustrate using DLL calls from Visual Basic 6 and Microsoft Excel. Though not illustrated, the compiled C++ code could also be used from statistical software such as R and SAS.

The Model

The particular STEMS model equations implemented are those compiled and described by Miner et al. (1988) with the addition of the regional height equations of Ek et al. (1981); calculations have been verified against the TWIGS software distributed by the USDA Forest Service (http://www.nrs.fs.fed.us/tools/software/). However, the modularity of our code allows for substitution of alternative component equations. The required equations predict crown ratio code, potential DBH growth, DBH growth modifier, DBH growth adjustment, probability of survival, and total tree height. Species-specific equation coefficients are used. An annual time step is used with the exception of total height prediction which is completed at the end of a projection interval. Survival (mortality) is modeled in a deterministic manner.

The Inputs

The required inputs at the stand level are age, not used in predictions, site index, site index species, and whether the stand is a red pine plantation or not. A representative tree list with species, DBH, total height, crown ratio code, and an expansion factor for each tree must also be supplied. The expansion factor is the number of trees per acre represented by the tree. Crown ratio code is 1 for crown ratios between 1 and 10%, 2 for crown ratios between 11 and 20%, etc. Total height and crown ratio code are both optional inputs. USDA Forest Service species codes are used by default, though our C++ code is setup so users can easily substitute their own set of input codes.

Library Call

The C++ function declaration is:

```
int LSGrowTrees(int iAge, int iSI, int iUserSISpp, int iNTrees, int iYearsGrow,
       bool isPlantation, int *iUserSppCode, double *DBHIn, double *HtIn, double *ExpFacIn,
       double *CRIn, double *DBHOut, double *HtOut, double *ExpFacOut, double *CROut)
```

The scalar variables passed are:

Variable	Type	Description
iAge	Integer	Stand age (years)
iSI	Integer	Site index, base age 50 (feet)
iUserSISpp	Integer	Species code for site index
iNTrees	Integer	Number of trees in tree list
iYearsGrow	Integer	Number of years to project tree list
isPlantation	Boolean	Is the stand a red pine plantation?

The pointers to one dimensional, zero-based arrays, one element per tree, passed are:

Array	Type	Description
iUserSppCode	Integer	Species code
DBHIn	Double	DBH (inches)
HtIn	Double	Total height (feet)
ExpFacIn	Double	Trees per acre represented
CRIn	Double	Crown ratio code

The pointers to one dimensional, zero-based arrays, one element per tree, returned are:

Array	Type	Description
DBHOut	Double	DBH (inches)
HtOut	Double	Total height (feet)
ExpFacOut	Double	Trees per acre represented
CROut	Double	Crown ratio code

The function call itself returns an integer value of 1 if no error occurs or a negative integer indicating a particular error.

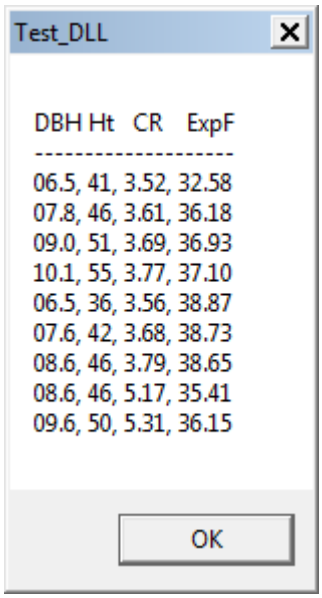
Sample Use I - Visual Basic 6

Appendix A presents an example of calling a Windows DLL compilation of the C++ code from Visual Basic 6.

The first lines of code make use of the compiler directive DevMode set using Project|Properties|Make in VB6. If DevMode is true, non-zero, the DLL is referenced with a full path; if it is false, zero, the DLL is assumed to reside in the Windows directory. The DLL reference parallels the function declaration presented above: note that the arrays must be passed by reference. An integer in C++ is equivalent to a long integer in Visual Basic 6, hence the "As Long" in the function definition. The DLL also contains the function VersionNumberString which returns the version number of the C++ code.

Private Sub cmdGo_Click is a simple example of setting up variables, calling the DLL, and displaying the result, a projected tree list, in a message box. Note that the arrays are zero-based and that it is the zeroth element of an array that is passed to the DLL (e.g. iUserSppCode(0)).

Running the Visual Basic 6 routine results in:



Sample Use II - Microsoft Excel (Visual Basic for Applications)

Appendix B presents an example of calling a Windows DLL compilation of the C++ code from an Excel macro. The macro is Visual Basic for Applications code which could be used in any Microsoft Office application including the Microsoft Access database.

The following illustrates how input data might be setup in an Excel worksheet:

	A	B	C	D	E	F	G	H	I	J	K
1	Stand age	38						Fill in B1:B3			
2	SI Spp.	746						Fill in tree list starting in A6			
3	SI	70						Press Ctrl/Shift/G and answer prompts			
4								Results are on "Output" sheet			
5	Tree	Species	DBH	Tot.Ht.	CR Code	Exp.Fac.					
6	1	746	6	38	-1	40					
7	2	746	7	43	-1	40					
8	3	746	8	48	-1	40					
9	4	746	9	52	-1	40					
10	5	375	6	34	-1	40					
11	6	375	7	40	-1	40					
12	7	375	8	44	-1	40					
13	8	12	8	44	-1	40					
14	9	12	9	48	-1	40					

The first lines of macro code declare how the DLL will be used. This basically parallels what was presented in the previous section. In this case a full path reference is provided.

Public Sub growTrees reads several inputs from the input worksheet and prompts the user for other inputs. The call to the DLL parallels that presented in the previous section. The results are written to a second worksheet in the Excel workbook:

	A	B	C	D	E	F	G
1	**Projected tree list**						
2							
3	**Tree**	**Species**	**DBH**	**Tot.Ht.**	**CR Code**	**Cd.Vol.**	**Exp.Fac.**
4	1	746	6.5	41	3.52	0.038	32.58
5	2	746	7.8	46	3.61	0.069	36.18
6	3	746	9.0	51	3.69	0.105	36.93
7	4	746	10.1	55	3.77	0.142	37.10
8	5	375	6.5	36	3.56	0.034	38.87
9	6	375	7.6	42	3.68	0.060	38.73
10	7	375	8.6	46	3.79	0.087	38.65
11	8	12	8.6	46	5.17	0.086	35.41
12	9	12	9.6	50	5.31	0.118	36.15

In this instance additional code was added to the macro to compute tree cordwood volume from DBH and total height, functions MerchVolume and VolumePct.

The C++ Code

Appendix C presents the C++ code. The particular syntax is that used in Microsoft Visual Studio C++. Annotation labels have been added to the code presentation to facilitate the following explanations.

The code at A shows how user species codes are setup and given their STEMS codes equivalents (cf. Miner et al. 1988 for STEMS codes). Here, 40 USDA Forest Service codes are being used for user inputs, array SppCodes, with their STEMS code equivalents defined in array STEMSSppCodes. For example USDA Forest Service species code 743, bigtooth aspen, is STEMS species code 23. Note that STEMS codes here are numbered 0, ..., 30 in contrast to 1, ..., 31 used by Miner et al. (1988).

The constant declarations beginning at B give the equation coefficients for the 31 STEMS species codes. Note that two dimensional arrays in C++ are defined in row major order (i.e. row by row). The zeroth element for a species coefficients is not used (is set to 0 in each instance). For example the fourth height equation coefficient for STEMS species code 2 is 0.1622.

The code at C illustrates defining a function accessible in a DLL when using Microsoft Visual Studio C++: note the inclusion of the keyword _stdcall in the function definition. In this case the code version number is returned as a string in the function argument.

The function definitions beginning at D are for the component prediction equations of STEMS (plus the height equation). The keyword _stdcall is absent from the function definitions as these functions are only accessible internally.

The function at E, STEMS, grows a tree list one year. To do so it mainly calls the functions defining the component prediction equations (cf. D).

The function accessible via, for example, a DLL call, LSGrowTrees, is defined starting at F. The function's return value is 1 if no errors occurred; error codes are listed in the comments at the top of the function code. Most errors correspond to invalid inputs; simple checks are applied to inputs, for example site index must be between 20 and 100 (see G).

Project Web Site

The C++ code, in the form of a Microsoft Visual Studio project, the Visual Basic 6 example usage, and the Excel example usage are all available at http://www.tc.umn.edu/~tburk/. This document and a document describing how to setup DLLs in Microsoft Visual Studio are also available there.

References

Belcher, D.M., M.R. Holdaway, and G.J. Brand. 1982. STEMS - the stand and tree evaluation and modeling system. USDA Forest Service General Technical Report NC-79.

Dixon, G.E. and C.E. Keyser, comps. 2013. Lake States variant overview - Forest vegetation simulator. Internal Report Fort Collins, CO: USFA, Forest Service, Forest Management Service Center.

Ek, A.R., E.T. Birdsall, and R.J. Spears. 1981. Total and merchantable tree height equations for Lake States tree species. Staff Paper Series No. 27. St. Paul, MN: Department of Forest Resources, University of Minnesota.

Miner, C.L., N.R. Walters, and M.L. Belli. 1988. A guide to the TWIGS program for the North Central United States. USDA Forest Service General Technical Report NC-125.

Appendix A

```
#If (DevMode) Then   'Explicit path
    Private Declare Function VersionNumberString _
        Lib "D:\teb\Visual Studio 2008\Projects\LSPTG\Release\LSPTG.dll" _
        (ByVal myString As String) As Long
    Private Declare Function LSGrowTrees _
        Lib "D:\teb\Visual Studio 2008\Projects\LSPTG\Release\LSPTG.dll" _
        (ByVal iAge As Long, ByVal iSI As Long, ByVal iUserSISpp As Long, ByVal iNTrees As Long, _
        ByVal iYearsGrow As Long, ByVal isPlantation As Boolean, ByRef iUserSppCode As Long, _
        ByRef DBHIn As Double, ByRef HtIn As Double, ByRef ExpFacIn As Double, ByRef CRIn As Double, _
        ByRef DBHOut As Double, ByRef HtOut As Double, ByRef ExpFacOut As Double, _
        ByRef CROut As Double) As Long
#Else   'Relative path
    Private Declare Function VersionNumberString Lib "LSPTG.dll" (ByVal myString As String) As Long
    Private Declare Function LSGrowTrees Lib "LSPTG.dll" _
        (ByVal iAge As Long, ByVal iSI As Long, ByVal iUserSISpp As Long, ByVal iNTrees As Long, _
        ByVal iYearsGrow As Long, ByVal isPlantation As Boolean, ByRef iUserSppCode As Long, _
        ByRef DBHIn As Double, ByRef HtIn As Double, ByRef ExpFacIn As Double, ByRef CRIn As Double, _
        ByRef DBHOut As Double, ByRef HtOut As Double, ByRef ExpFacOut As Double, _
        ByRef CROut As Double) As Long
#End If

Private Sub cmdGo_Click()

    Dim LoopArr As Long
    Dim MakeString As String
    Dim ReturnValue As Long

    Dim iAge As Long, iSI As Long, iUserSISpp As Long, iNTrees As Long, iYearsGrow As Long
    Dim isPlantation As Boolean
    Dim iUserSppCode() As Long
    Dim DBHIn() As Double, HtIn() As Double, ExpFacIn() As Double, CRIn() As Double
    Dim DBHOut() As Double, HtOut() As Double, ExpFacOut() As Double, CROut() As Double

    iAge = 38
    iSI = 70
    iUserSISpp = 746
    iNTrees = 9
    iYearsGrow = 7
```

```
isPlantation = False
ReDim iUserSppCode(iNTrees - 1) As Long
ReDim DBHIn(iNTrees - 1) As Double
ReDim HtIn(iNTrees - 1) As Double
ReDim ExpFacIn(iNTrees - 1) As Double
ReDim CRIn(iNTrees - 1) As Double
ReDim DBHOut(iNTrees - 1) As Double
ReDim HtOut(iNTrees - 1) As Double
ReDim ExpFacOut(iNTrees - 1) As Double
ReDim CROut(iNTrees - 1) As Double
iUserSppCode(0) = 746: iUserSppCode(1) = 746: iUserSppCode(2) = 746: iUserSppCode(3) = 746
iUserSppCode(4) = 375: iUserSppCode(5) = 375: iUserSppCode(6) = 375: iUserSppCode(7) = 12
iUserSppCode(8) = 12
DBHIn(0) = 6: DBHIn(1) = 7: DBHIn(2) = 8: DBHIn(3) = 9: DBHIn(4) = 6: DBHIn(5) = 7: DBHIn(6) = 8
DBHIn(7) = 8: DBHIn(8) = 9
HtIn(0) = 38: HtIn(1) = 43: HtIn(2) = 48: HtIn(3) = 52: HtIn(4) = 34: HtIn(5) = 40: HtIn(6) = 44
HtIn(7) = 44: HtIn(8) = 48
ExpFacIn(0) = 40: ExpFacIn(1) = 40: ExpFacIn(2) = 40: ExpFacIn(3) = 40: ExpFacIn(4) = 40
ExpFacIn(5) = 40: ExpFacIn(6) = 40: ExpFacIn(7) = 40: ExpFacIn(8) = 40
CRIn(0) = -1: CRIn(1) = -1: CRIn(2) = -1: CRIn(3) = -1: CRIn(4) = -1: CRIn(5) = -1
CRIn(6) = -1: CRIn(7) = -1: CRIn(8) = -1

ReturnValue = LSGrowTrees(iAge, iSI, iUserSISpp, iNTrees, iYearsGrow, isPlantation, iUserSppCode(0), _
    DBHIn(0), HtIn(0), ExpFacIn(0), CRIn(0), DBHOut(0), HtOut(0), ExpFacOut(0), CROut(0))

'If ReturnValue is anything other than 1, some sort of error occurred
MsgBox ("I got this: " & ReturnValue)

MakeString = "DBH Ht   CR    ExpF" & vbCrLf & "--------------------" & vbCrLf
For LoopArr = 0 To iNTrees - 1
    MakeString = MakeString & Format(DBHOut(LoopArr), "0#.0") & ", " & _
        Format(HtOut(LoopArr), "##") & ", " & _
        Format(CROut(LoopArr), "#.00") & ", " & Format(ExpFacOut(LoopArr), "###.00") & vbCrLf
Next LoopArr

MsgBox (MakeString)

Dim myString As String * 6
VersionNumberString (myString)
MsgBox (myString)
```

```
End Sub

Private Sub cmdStop_Click()

    Unload Me

End Sub
```

Appendix B

```
Option Explicit

'Hook to DLL (path to change or unneeded if in Windows directory)
Private Declare Function LSGrowTrees _
    Lib "D:\teb\Visual Studio 2008\Projects\LSPTG\Release\LSPTG.dll" _
    (ByVal iAge As Long, ByVal iSI As Long, ByVal iUserSISpp As Long, ByVal iNTrees As Long, _
    ByVal iYearsGrow As Long, ByVal isPlantation As Boolean, ByRef iUserSppCode As Long, _
    ByRef DBHIn As Double, ByRef HtIn As Double, ByRef ExpFacIn As Double, ByRef CRIn As Double, _
    ByRef DBHOut As Double, ByRef HtOut As Double, ByRef ExpFacOut As Double, _
    ByRef CROut As Double) As Long

'Called with Ctrl/Shift/G
Public Sub growTrees()

Dim i As Integer, k As Integer

Dim iAge As Long, iSI As Long, iUserSISpp As Long, iNTrees As Long, iYearsGrow As Long
Dim isPlantation As Boolean
Dim iUserSppCode() As Long
Dim DBHIn() As Double, HtIn() As Double, ExpFacIn() As Double, CRIn() As Double
Dim DBHOut() As Double, HtOut() As Double, ExpFacOut() As Double, CROut() As Double

'Read initial stand conditions from worksheet
iAge = Sheets("Input").Cells(1, 2)
iUserSISpp = Sheets("Input").Cells(2, 2)
iSI = Sheets("Input").Cells(3, 2)

'Ask for number of trees in tree list, reporting interval, number of intervals
iNTrees = InputBox("Number of trees in tree list? ", "Type A Number")

'Number of years will grow trees
iYearsGrow = InputBox("Number of years to project tree list? ", "Type A Number")
'Red pine plantation?
isPlantation = False

'Need to clear Output worksheet
Worksheets("Output").Range("A4:F500").ClearContents

ReDim iUserSppCode(iNTrees - 1) As Long
```

```
ReDim DBHIn(iNTrees - 1) As Double
ReDim HtIn(iNTrees - 1) As Double
ReDim ExpFacIn(iNTrees - 1) As Double
ReDim CRIn(iNTrees - 1) As Double
ReDim DBHOut(iNTrees - 1) As Double
ReDim HtOut(iNTrees - 1) As Double
ReDim ExpFacOut(iNTrees - 1) As Double
ReDim CROut(iNTrees - 1) As Double

'Get initial tree list
For i = 0 To iNTrees - 1
    iUserSppCode(i) = Sheets("Input").Cells(6 + i, 2)
    DBHIn(i) = Sheets("Input").Cells(6 + i, 3)
    HtIn(i) = Sheets("Input").Cells(6 + i, 4)
    CRIn(i) = Sheets("Input").Cells(6 + i, 5)
    ExpFacIn(i) = Sheets("Input").Cells(6 + i, 6)
Next i

'Call DLL to grow tree list
Dim ReturnValue As Long
ReturnValue = LSGrowTrees(iAge, iSI, iUserSISpp, iNTrees, iYearsGrow, isPlantation, iUserSppCode(0), _
    DBHIn(0), HtIn(0), ExpFacIn(0), CRIn(0), DBHOut(0), HtOut(0), ExpFacOut(0), CROut(0))

'TESTING
MsgBox ("I got this: " & ReturnValue)

Dim outputRow As Integer
outputRow = 4
For k = 0 To iNTrees - 1
    Sheets("Output").Cells(outputRow, 1).Value = k + 1
    Sheets("Output").Cells(outputRow, 2).Value = iUserSppCode(k)
    Sheets("Output").Cells(outputRow, 3).Value = DBHOut(k)
    Sheets("Output").Cells(outputRow, 4).Value = HtOut(k)
    Sheets("Output").Cells(outputRow, 5).Value = CROut(k)
    Sheets("Output").Cells(outputRow, 6).Value = MerchVolume(DBHOut(k), HtOut(k)) / 79
    Sheets("Output").Cells(outputRow, 7).Value = ExpFacOut(k)
    outputRow = outputRow + 1
Next k

End Sub
```

```
'****************************************
'
'Next two only needed if want tree volumes
'
'****************************************
Private Function MerchVolume(dblDBH As Double, dblHeight As Double) As Single
'Compute merchantable volume of a tree of DBH dblDBH and total height dblHeight.
'Merchantability specs are 3-inch top and min DBH of 5 inches.
'FOR NOW merch volume percents are assumed to have been placed in Public Public array
'intVolumePct initialized prior to call. intVolumePCT is based on Tables 10,
'11, and 9 of Bulletin 1104. This is standard cord calculation for company *******.

Dim intDBH As Integer
Dim sngTotVolume As Single
Dim sngBA As Single

sngBA = 0.005454 * dblDBH * dblDBH

If dblHeight < 30 Then
    sngTotVolume = (0.42 + 0.006 * (30 - dblHeight)) * sngBA * dblHeight
Else
    sngTotVolume = 0.42 * sngBA * dblHeight
End If

intDBH = CInt(dblDBH)
If intDBH < 5 Then
    intDBH = 4
ElseIf intDBH > 20 Then
    intDBH = 20
End If

MerchVolume = VolumePct(intDBH) / 100# * sngTotVolume

End Function

Private Function VolumePct(iDBH As Integer) As Single
'These are the default merchantable volume percents that ******** uses as a function
'of DBH. They are based upon Tables 10, 11, and 9 of Bulletin 1104. Min merch DBH is
'5 inches and top diameter limit is 3 inches. Trees larger than 20-inches DBH are
'assumed to "be like" 20-inch DBH trees
```

```
Select Case iDBH
    Case 4
        VolumePct = 0
    Case 5
        VolumePct = 72
    Case 6
        VolumePct = 77
    Case 7
        VolumePct = 82
    Case 8
        VolumePct = 85
    Case 9
        VolumePct = 87
    Case 10
        VolumePct = 87
    Case 11
        VolumePct = 88
    Case 12
        VolumePct = 89
    Case 13
        VolumePct = 89
    Case 14
        VolumePct = 90
    Case 15
        VolumePct = 92
    Case 16
        VolumePct = 93
    Case 17
        VolumePct = 93
    Case 18
        VolumePct = 94
    Case 19
        VolumePct = 94
    Case 20
        VolumePct = 95

End Select

End Function
```

Appendix C

```
// LSPTG.cpp : Defines the exported functions for the DLL application.
//

#include "stdafx.h"
#include "math.h"

// Number of model (STEMS) species codes
//    They are used as 0, 1, 2, ..., 28, 29, 30; e.g. aspen, STEMS species 25, is 24
//    Note: STEMS species 27 (26 here) is for red pine plantation (modifier function)
const int NumSTEMSSppCodes = 31;

// Translates user species codes into model species codes
//    NumSppCodes is how many user-defined species codes there are
//    SppCodes[] are the user-defined species codes
//    STEMSSppCodes[] are STEMS model species codes to use for the user-defined codes
//          Keeping in mind that the STEMS are 0-based so e.g. aspen, STEMS species 25, is 24
const int NumSppCodes = 40;
const int SppCodes[40] = {105,130,125,129,94,12,95,71,241,261,
      299,543,544,742,317,316,972,975,977,371,951,318,314,541,802,804,823,826,833,809,
      837,402,403,407,743,746,741,375,300,999};
const int STEMSSppCodes[40] = {0,0,1,2,3,4,5,6,7,8,9,10,10,11,12,13,14,14,14,15,
      16,17,17,18,19,19,19,19,20,21,21,22,22,22,23,24,24,25,29,30};

// Height equation coefficients. There are 6 coefficients, though coefficient 5 is not used
//    Coefficient b1 is in column 1, etc.
//    From Ek et al. 1981
const double HtCoeffs[31][7] = {0,16.934,0.12972,1,0.20854,0,0.12902,
      0,36.851,0.08298,1,0.00001,0,0.18231,0,16.281,0.08621,1,0.1622,0,0.23316,
      0,31.957,0.18511,1.702,0,0,0.162,0,14.304,0.19894,1.4195,0.23349,0,0.12399,
      0,20.038,0.18981,1.2909,0.17836,0,0.10159,0,13.62,0.24255,1.2885,0.25831,0,0.10771,
      0,8.2079,0.19672,1.3112,0.33978,0,0.11666,0,5.3117,0.10357,1,0.68454,0,0,
      0,16.934,0.12972,1,0.20854,0,0.12902,0,11.291,0.2525,1.5466,0.35711,0,0.06859,
      0,13.625,0.28668,1.6124,0.30651,0,0.0746,0,6.86,0.27725,1.4287,0.40115,0,0.12403,
      0,6.86,0.27725,1.4287,0.40115,0,0.12403,0,8.458,0.27527,1.9602,0.34894,0,0.12594,
      0,7.1852,0.28384,1.4417,0.38884,0,0.11411,0,6.3628,0.27859,1.8677,0.49589,0,0.05841,
      0,5.3416,0.23044,1.1529,0.54194,0,0.06372,0,8.1782,0.27316,1.725,0.38694,0,0.10847,
      0,9.2078,0.22208,1,0.31723,0,0.13465,0,6.6844,0.19049,1,0.43972,0,0.10806,
      0,3.8011,0.39213,2.9053,0.55634,0,0.09593,0,6.1034,0.17368,1,0.44725,0,0.1461,
      0,5.5346,0.22637,1,0.46918,0,0.11782,0,6.4301,0.23545,1.338,0.4737,0,0.08228,
```

A

B

```
      0,7.2773,0.22721,1,0.41179,0,0.11046,0,6.9572,0.26564,1,0.4866,0,0.01618,
      0,6.9572,0.26564,1,0.4866,0,0.01618,0,6.9572,0.26564,1,0.4866,0,0.01618,
      0,6.9572,0.26564,1,0.4866,0,0.01618,0,6.9572,0.26564,1,0.4866,0,0.01618};

// Crown ratio equation coefficients. There are 4 coefficients. Stored like height equation.
//    From Miner et al. 1988
const double CRCoeffs[31][5] = {0.000E+00,6.640E+00,1.350E-02,3.200E+00,-5.180E-02,
      0.000E+00,5.350E+00,5.300E-03,1.528E+00,-3.300E-02,
      0.000E+00,6.790E+00,5.800E-03,7.590E+00,-1.030E-02,
      0.000E+00,7.840E+00,5.700E-03,1.272E+00,-1.420E-01,
      0.000E+00,5.630E+00,4.700E-03,3.523E+00,-6.890E-02,
      0.000E+00,5.540E+00,7.200E-03,4.200E+00,-5.300E-02,
      0.000E+00,6.000E+00,5.300E-03,4.310E-01,-1.200E-03,
      0.000E+00,5.710E+00,7.700E-03,2.290E+00,-2.530E-01,
      0.000E+00,5.710E+00,7.700E-03,2.290E+00,-2.530E-01,
      0.000E+00,5.710E+00,7.700E-03,2.290E+00,-2.530E-01,
      0.000E+00,4.500E+00,3.200E-03,7.950E-01,-1.050E-01,
      0.000E+00,4.350E+00,4.600E-03,1.820E+00,-2.740E-01,
      0.000E+00,4.850E+00,5.000E-03,9.810E+00,-9.900E-03,
      0.000E+00,4.350E+00,4.600E-03,1.820E+00,-2.740E-01,
      0.000E+00,4.400E+00,2.500E-03,1.000E+00,-9.400E-02,
      0.000E+00,4.180E+00,2.500E-03,1.410E+00,-5.120E-01,
      0.000E+00,4.440E+00,3.700E-03,2.090E+00,-6.500E-02,
      0.000E+00,3.400E+00,6.600E-03,2.870E+00,-4.340E-01,
      0.000E+00,4.490E+00,2.900E-03,1.210E+00,-6.500E-02,
      0.000E+00,5.840E+00,8.200E-03,3.260E+00,-4.900E-02,
      0.000E+00,4.200E+00,1.600E-03,2.760E+00,-2.500E-02,
      0.000E+00,5.060E+00,3.300E-03,1.730E+00,-6.100E-02,
      0.000E+00,6.210E+00,7.300E-03,9.990E+00,-1.000E-02,
      0.000E+00,4.110E+00,5.400E-03,1.650E+00,-1.100E-01,
      0.000E+00,4.000E+00,2.400E-03,-2.830E+00,2.100E-02,
      0.000E+00,5.000E+00,6.600E-03,4.920E+00,-2.630E-02,
      0.000E+00,4.000E+00,2.400E-03,-2.830E+00,2.100E-02,
      0.000E+00,4.000E+00,2.400E-03,-2.830E+00,2.100E-02,
      0.000E+00,4.000E+00,2.400E-03,-2.830E+00,2.100E-02,
      0.000E+00,4.000E+00,2.400E-03,-2.830E+00,2.100E-02,
      0.000E+00,4.000E+00,2.400E-03,-2.830E+00,2.100E-02};

// Potential growth equation coefficients. There are 5 coefficients. Stored like height equation.
//    From Miner et al. 1988
const double PGCoeffs[31][6] = {0.0000E+00,1.6062E-01,-9.0000E-06,3.6245E+00,4.0000E-05,1.0000E+00,
```

```
      0.0000E+00,9.4460E-02,-1.2000E-04,2.0596E+00,3.5000E-04,2.4225E-01,
      0.0000E+00,2.5578E-01,-8.8000E-04,1.7263E+00,4.0000E-05,1.0000E+00,
      0.0000E+00,1.7056E-01,-1.4516E-02,1.0660E+00,5.2000E-04,2.7298E-01,
      0.0000E+00,1.2200E-01,-8.0000E-04,1.9890E+00,6.0000E-05,1.0000E+00,
      0.0000E+00,1.0713E-01,-1.0700E-03,2.0017E+00,6.0000E-05,9.1127E-01,
      0.0000E+00,1.1147E-01,-1.0000E-05,3.0685E+00,3.0000E-05,1.0000E+00,
      0.0000E+00,1.3403E-01,-1.0000E-06,3.6880E+00,2.0000E-05,1.0000E+00,
      0.0000E+00,1.6872E-01,-3.0000E-07,3.5738E+00,1.0000E-05,1.0000E+00,
      0.0000E+00,1.6062E-01,-9.0000E-06,3.6245E+00,4.0000E-05,1.0000E+00,
      0.0000E+00,5.8807E-02,-6.0000E-09,5.0559E+00,2.4000E-04,3.1553E-01,
      0.0000E+00,1.0948E-01,-4.0000E-05,2.2226E+00,5.0000E-04,6.6260E-02,
      0.0000E+00,1.0948E-01,-4.0000E-05,2.2226E+00,5.0000E-04,6.6260E-02,
      0.0000E+00,1.0948E-01,-4.0000E-05,2.2226E+00,5.0000E-04,6.6260E-02,
      0.0000E+00,2.8496E-01,-1.8200E-03,1.5297E+00,3.0000E-05,1.0000E+00,
      0.0000E+00,1.5155E-01,-3.0000E-06,3.3104E+00,7.0000E-05,5.7302E-01,
      0.0000E+00,2.5402E-01,-4.0000E-06,2.9396E+00,1.0000E-05,1.0000E+00,
      0.0000E+00,1.8772E-01,-7.0000E-06,2.5839E+00,2.8000E-04,8.3850E-02,
      0.0000E+00,2.1167E-01,-3.0000E-06,3.3131E+00,1.0000E-05,1.2430E-01,
      0.0000E+00,1.2654E-01,-4.0000E-06,2.7538E+00,5.0000E-05,6.2086E-01,
      0.0000E+00,1.5535E-01,-1.0000E-07,3.5367E+00,1.8000E-04,2.5900E-01,
      0.0000E+00,1.7358E-01,-7.0000E-05,2.4451E+00,3.0000E-05,9.5222E-01,
      0.0000E+00,1.6471E-01,-2.1610E-03,1.3949E+00,4.0000E-05,7.1628E-01,
      0.0000E+00,2.3490E-01,-9.4200E-03,1.1041E+00,3.6000E-04,1.5386E-01,
      0.0000E+00,2.1645E-01,-9.1000E-05,2.6030E+00,4.4000E-05,1.0000E+00,
      0.0000E+00,1.0971E-01,-3.2000E-04,2.0236E+00,3.4000E-04,2.1288E-01,
      0.0000E+00,3.7510E-01,-2.4790E-02,9.6288E-01,4.0000E-05,1.0000E+00,
      0.0000E+00,3.7510E-01,-2.4790E-02,9.6288E-01,4.0000E-05,1.0000E+00,
      0.0000E+00,3.7510E-01,-2.4790E-02,9.6288E-01,4.0000E-05,1.0000E+00,
      0.0000E+00,3.7510E-01,-2.4790E-02,9.6288E-01,4.0000E-05,1.0000E+00,
      0.0000E+00,3.7510E-01,-2.4790E-02,9.6288E-01,4.0000E-05,1.0000E+00};

// Competition modifier equation coefficients. There are 7 coefficients.
//    0th coefficient is BAmax, 1st - 4th coefficients are bs, 5th - 6th coefficients are cs.
//    Otherwise stored like height equation.
//    From Miner et al. 1988
const double CMCoeffs[31][7] = {225,1.78E+00,-3.00E+00,1.62E+01,2.27E-01,4.02E-01,2.30E-01,
      300,7.19E-01,-1.09E+01,1.69E+03,3.75E-01,2.03E+00,-3.54E-01,
      300,1.36E+00,-2.64E+00,1.15E+01,3.86E-01,9.70E-02,7.55E-01,
      350,5.00E+00,-1.01E+00,3.64E+00,0,1.507E+00,-5.20E-01,
      325,1.76E+00,-1.51E+00,2.63E+00,2.33E-01,9.27E-01,-2.99E-01,
      300,3.80E+00,-1.52E+00,6.54E+00,3.48E-01,5.22E-01,1.73E-01,
```

```
      250,1.78E+00,-3.00E+00,1.62E+01,2.27E-01,3.90E-02,1.00E+00,
      350,2.54E+00,-1.14E+00,2.26E+00,0,5.26E-01,1.36E-01,
      300,1.27E+00,-1.34E+00,1.05E+00,0,4.60E-02,1.00E+00,
      300,1.36E+00,-2.64E+00,1.15E+01,3.86E-01,9.70E-02,7.55E-01,
      250,5.00E+00,-5.68E-01,1.83E+00,6.30E-02,2.60E-01,4.19E-01,
      250,1.40E+00,-2.03E+00,1.04E+01,6.94E-01,1.81E-01,4.45E-01,
      250,1.40E+00,-2.03E+00,1.04E+01,6.94E-01,1.81E-01,4.45E-01,
      250,1.40E+00,-2.03E+00,1.04E+01,6.94E-01,1.81E-01,4.45E-01,
      250,5.00E+00,-9.70E-01,4.40E+00,2.68E-01,1.00E-01,6.29E-01,
      250,0.679,-10.97,1568,0.483,0.202,0.454,250,1.59,-3.27,26.7,0.412,0.353,0.182,
      250,1.17,-4.59,29.19,0.43,0.142,0.524,250,5,-1.38,8.26,0.326,0.453,0.34,
      250,1.98,-0.974,1.64,0,0.051,1,275,1.98,-0.974,1.64,0,0.278,0.365,
      250,1.98,-0.974,1.64,0,1.365,-0.208,250,1.66,-2.62,9.97,0.515,0.28,0.228,
      250,1.13,-4.64,164.6,0.648,0.093,1,250,1.08,-6.6,346.1,0.395,0.209,0.543,
      275,1.98,-1.75,3.67,0.232,0.11,0.678,350,2.31,-1.67,3.94,0,4.41E-01,1.73E-01,
      275,1.98E+00,-9.74E-01,1.64E+00,0,2.78E-01,3.65E-01,
      275,1.98E+00,-9.74E-01,1.64E+00,0,2.78E-01,3.65E-01,
      275,1.98E+00,-9.74E-01,1.64E+00,0,2.78E-01,3.65E-01,
      275,1.98E+00,-9.74E-01,1.64E+00,0,2.78E-01,3.65E-01};

// DBH growth adjustment equation coefficients. There are 3 coefficients. Stored like height equation.
//    From Miner et al. 1988
const double DACoeffs[31][4] = {0,-0.0174,0.00065,0.069,0,0,-0.00017,0.018,
      0,-0.0043,0.00011,0.029,0,0,0,0,0,-0.0112,0.0004,0.03,0,-0.0248,0.00162,0.037,
      0,-0.0093,0.00042,0.048,0,-0.005,0,-0.004,0,-0.0043,0.00022,0.039,0,0,0,0,
      0,-0.0115,0.00032,0.051,0,0,0,0,0,0.0109,0,-0.106,0,0.0008,0.00004,-0.017,
      0,-0.0102,0.00023,0.028,0,-0.0021,0.0001,0.001,0,0,0.00007,-0.037,0,0.0014,0.00002,-0.024,
      0,-0.0113,0.00066,-0.002,0,-0.0079,0.00039,0.048,0,0.0015,-0.00003,-0.026,
      0,0.0026,0,-0.088,0,0,0,0,0,0.0124,-0.00022,-0.079,0,-0.0132,0.00079,0.031,
      0,-0.0066,0.00038,0.011,0,0,0,0,0,0,0,0,0,0,0,0,0,0.0085,-0.00026,-0.088,0,-0.0046,0,-0.07};

// Probability survival equation coefficients. There are 7 coefficients. Stored like height equation.
//    From Miner et al. 1988
const double PSCoeffs[31][8] =
      {0.0000E+00,9.9660E-01,5.9020E-01,1.8000E+01,7.1100E-01,1.1440E-02,4.7640E+00,6.8340E-01,
      0.0000E+00,9.9970E-01,1.9953E+00,5.7970E+01,1.0120E+00,2.6480E-01,1.6260E+00,1.2730E-01,
      0.0000E+00,9.9890E-01,1.6150E+00,1.5680E+03,2.2680E+00,8.7120E-01,4.3200E-01,1.0120E-01,
      0.0000E+00,9.9940E-01,9.7240E-01,3.1420E+02,1.9150E+00,2.8386E-01,1.3021E+00,1.6831E-01,
      0.0000E+00,9.9840E-01,1.9241E+00,2.8710E+01,1.0210E+00,4.4210E+00,3.6400E+00,1.6720E+00,
      0.0000E+00,9.9460E-01,1.6990E+00,5.3778E+01,1.2190E+00,6.8277E-01,9.5976E-01,2.2499E-01,
      0.0000E+00,9.9700E-01,1.6807E+00,5.3778E+01,1.2190E+00,6.8277E-01,9.5976E-01,2.2499E-01,
```

```
      0.0000E+00,9.9900E-01,2.2089E+00,8.6735E+01,1.0000E+00,1.9312E-01,1.5881E+00,2.1573E-01,
      0.0000E+00,9.9910E-01,3.0762E+00,2.7140E+01,1.1006E+00,6.6775E-02,3.4908E+00,4.8884E-01,
      0.0000E+00,9.9970E-01,1.9953E+00,5.7970E+01,1.0120E+00,2.6480E-01,1.6260E+00,1.2730E-01,
      0.0000E+00,9.9900E-01,1.6657E+00,5.3778E+01,1.2190E+00,6.8277E-01,9.5976E-01,2.2499E-01,
      0.0000E+00,9.9636E-01,1.0260E+00,3.6859E+02,1.7248E+00,3.4484E-01,1.9680E+00,2.9190E-01,
      0.0000E+00,9.9636E-01,1.0260E+00,3.6859E+02,1.7248E+00,3.4484E-01,1.9680E+00,2.9190E-01,
      0.0000E+00,9.9640E-01,1.0260E+00,3.6860E+02,1.7250E+00,3.4480E-01,1.9680E+00,2.9190E-01,
      0.0000E+00,9.9410E-01,5.7940E-01,1.1630E+01,6.4400E-01,2.6550E-01,2.9630E+00,4.7150E-01,
      0.0000E+00,9.9750E-01,2.2028E+00,1.9107E+01,8.2984E-01,1.5171E+00,2.1691E+00,7.9582E-01,
      0.0000E+00,9.9790E-01,1.1097E+00,2.8390E+01,1.0450E+00,4.0350E-01,2.0390E+00,2.7690E-01,
      0.0000E+00,9.9790E-01,2.4852E+00,6.0760E+01,1.2050E+00,8.5030E-01,1.0100E+00,2.3940E-01,
      0.0000E+00,9.9918E-01,1.3148E+00,1.3925E+03,2.4835E+00,3.4132E-02,4.9703E+00,8.1102E-01,
      0.0000E+00,9.9940E-01,2.2907E+00,1.0000E+03,1.8500E+00,6.5760E-02,4.2280E+00,7.3660E-01,
      0.0000E+00,9.9770E-01,5.6390E-01,2.9090E+01,1.1370E+00,1.0040E-02,3.8340E+00,3.1770E-01,
      0.0000E+00,9.9770E-01,5.6380E-01,2.9090E+01,1.1370E+00,1.0040E-02,3.8340E+00,3.1770E-01,
      0.0000E+00,9.9800E-01,2.0115E+00,3.1396E+02,2.4521E+00,1.1489E-02,4.1021E+00,3.4790E-01,
      0.0000E+00,9.9080E-01,3.7720E-01,3.4548E+01,1.0886E+00,9.3138E-02,3.4186E+00,5.3457E-01,
      0.0000E+00,9.9080E-01,3.7720E-01,3.4550E+01,1.0890E+00,9.3140E-02,3.4190E+00,5.3460E-01,
      0.0000E+00,9.9910E-01,1.9539E+00,8.3790E+00,3.9820E-01,1.7100E+00,2.4440E+00,8.9230E-01,
      0.0000E+00,9.9970E-01,1.9953E+00,5.7970E+01,1.0120E+00,2.6480E-01,1.6260E+00,1.2730E-01,
      0.0000E+00,1.0000E+00,9.9000E+01,1.0000E+00,1.0000E+00,1.0000E-01,1.0000E+00,1.0000E+00,
      0.0000E+00,1.0000E+00,9.9000E+01,1.0000E+00,1.0000E+00,1.0000E-01,1.0000E+00,1.0000E+00,
      0.0000E+00,9.9280E-01,2.1665E+00,8.8783E+01,1.6255E+00,2.9477E-03,4.8551E+00,5.6133E-01,
      0.0000E+00,9.9520E-01,2.0387E+00,1.5360E+02,1.2989E+00,3.3279E-01,3.0610E+00,8.2021E-01};

// Site index conversion equation intercepts
//    31 rows corresponding to 31 STEMS species codes; row 0 is STEMS species 1, etc.
//    Each row is for converting __TO THAT SPECIES__ from the species represented by the column (31 cols)
//    Column 0 is STEMS species 1, etc.
const double SIConvB0[31][31] = {0,-0.083,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-4.661,-4.661,-11.774,0,0,0,0,0,
      0.081,0,3.926,14.453,0,0,0,0,0,0,0,0,13.601,0,0,0,0,0,0,0,0,-20.194,-20.194,0,0,0,0,0,0,
      0,-4.094,0,10.9,0,0,0,0,0,0,0,0,5.264,0,0,0,0,0,0,4.678,4.678,0,-9.004,-9.004,0,0,0,0,0,0,
      0,-18.249,-13.974,0,0,0,0,-19.486,0,0,0,0,0,0,0,0,0,0,0,0,0,-7.893,-7.893,0,0,0,0,0,0,
      0,0,0,0,0,-7.212,0,-15.196,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1.366,-1.366,12.096,0,0,0,0,0,
      0,0,0,0,6.175,0,2.846,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,-3.7,0,-12.708,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,10.904,8.205,0,6.434,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,8.379,-0.953,-15.228,0.253,7.441,4.812,0,-37.214,-37.214,0,-4.346,-4.346,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,-17.873,-5.606,0,0,0,0,0,0,0,-9.376,0,0,0,6.729,-2.119,-2.628,3.4,11.937,0,-11.597,-11.597,0,-18.897,-18.897,-16.538,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0.946,0,0,-7.973,0,-6.136,-5.273,-1.384,3.405,0,-9.013,-9.013,0,-0.214,-0.214,-5.084,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,11.436,0,0,2.022,5.216,0,-4.758,2.045,1.885,0,-6.197,-6.197,0,-9.885,-9.885,-10.258,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,-0.256,0,0,2.652,4.856,4.768,0,4.931,9.642,0,-15.079,-15.079,0,3.165,3.165,-1.913,0,0,0,0,0,
```

```
	0,0,0,0,0,0,0,0,0,0,-7.746,0,0,-3.563,1.262,-2.119,-4.97,0,1.006,0,-11.199,-11.199,0,8.187,-2.435,-11.199,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,-5.428,0,0,-15.895,-3.717,-2.162,-11.645,-1.143,0,0,-34.762,-34.762,0,-23.105,-23.105,-16.344,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1.235,-1.235,0,0,0,0,0,0,0,0,0,
	0,0,-5.576,0,0,0,0,0,0,0,23.568,0,0,10.41,7.934,6.247,12.35,10.199,22.442,1.321,0,0,0,-8.189,-16.187,-16.401,0,0,0,0,0,
	0,0,-5.576,0,0,0,0,0,0,0,23.568,0,0,10.41,7.934,6.247,12.35,10.199,22.442,1.321,0,0,0,-8.189,-16.187,-16.401,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	4.385,16.194,8.276,7.893,1.605,0,0,0,0,0,4.818,0,0,16.179,0.239,9.945,-3.615,-11.564,19.238,0,7.992,7.992,0,0,2.223,-19.532,0,0,0,0,0,
	4.385,16.194,8.276,7.893,1.605,0,0,0,0,0,4.818,0,0,16.179,0.239,9.945,-3.615,2.868,19.238,0,13.856,13.856,0,-2.271,0,8.5,0,0,0,0,0,
	9.002,0,0,0,-17.109,0,0,0,0,0,0,0,0,13.612,4.769,9.696,1.911,10.199,13.342,0,13.069,13.069,0,13.926,-8.947,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

// Site index conversion equation slopes (see details for intercepts)
const double SIConvB1[31][31] = {0,1.028,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.063,1.063,1.308,0,0,0,0,0,
	0.973,0,0.959,0.792,0,0,0,0,0,0,0,0,0,0.761,0,0,0,0,0,0,0,0,0,1.247,1.247,0,0,0,0,0,0,
	0,1.043,0,0.78,0,0,0,0,0,0,0,0,0,0.939,0,0,0,0,0,0,0.839,0.839,0,1.088,1.088,0,0,0,0,0,0,
	0,1.263,1.282,0,0,0,0,1.787,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,
	0,0,0,0,0,1.168,0,1.852,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.851,0.851,0.707,0,0,0,0,0,
	0,0,0,0,0.856,0,0.769,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,1.3,0,1.975,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0.56,0.54,0,0.506,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0.894,1.007,1.332,0.99,0.961,0.887,0,1.579,1.579,0,0.902,0.902,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,1.314,1.065,0,0,0,0,0,0,0,1.119,0,0,0,0.844,1.048,0.991,0.954,0.751,0,1.114,1.114,0,1.168,1.168,1.215,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0.993,0,0,1.185,0,1.176,1.086,1.096,0.916,0,1.136,1.136,0,0.896,0.896,1.066,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0.751,0,0,0.954,0.85,0,0.998,0.965,0.872,0,0.992,0.992,0,0.994,0.994,1.058,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,1.01,0,0,1.009,0.921,1.002,0,0.992,0.828,0,1.221,1.221,0,0.876,0.876,1.001,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,1.041,0,0,1.048,0.912,1.036,1.008,0,0.88,0,1.098,1.098,0,0.708,0.849,1.098,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,1.128,0,0,1.332,1.092,1.147,1.208,1.136,0,0,1.549,1.549,0,1.201,1.201,1.225,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.935,0.935,0,0,0,0,0,0,0,0,0,
	0,0,1.192,0,0,0,0,0,0,0,0.633,0,0,0.898,0.88,1.008,0.819,0.911,0.646,1.07,0,1,0,1.025,1.168,1.255,0,0,0,0,0,
	0,0,1.192,0,0,0,0,0,0,0,0.633,0,0,0.898,0.88,1.008,0.819,0.911,0.646,1.07,1,0,0,1.025,1.168,1.255,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0.941,0.802,0.919,1,1.175,0,0,0,0,0,1.109,0,0,0.856,1.116,1.006,1.142,1.412,0.833,0,0.976,0.976,0,0,0.979,1.403,0,0,0,0,0,
	0.941,0.802,0.919,1,1.175,0,0,0,0,0,1.109,0,0,0.856,1.116,1.006,1.142,1.178,0.833,0,0.856,0.856,0,1.021,0,0.95,0,0,0,0,0,
	0.765,0,0,0,1.414,0,0,0,0,0,0,0,0,0.823,0.938,0.945,0.999,0.911,0.816,0,0.797,0.797,0,0.713,1.053,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

int _stdcall VersionNumberString(char * outString) {
	// From what I read, one should not try to pass a string back to VB from C++
	// Need extra terminating character in C++ strings, hence 6 versus 5

	strncpy_s(outString,6,"v1.00",6);
	return 1;
```

C

```
}

int ConvertSppCode(int iCodeIn) {
      // iCodeIn is a user species code. Return it's model code equivalent or -1.

      for (int i=0; i<NumSppCodes; i++) {
            if(SppCodes[i] == iCodeIn) return STEMSSppCodes[i];
      }
      return -1;
}

double CrownRatioCode(int iSTEMSCode, double DBH, double RunningAvgG) {
      // Predict a tree's crown ratio code
      // iSTEMSCode is STEMS species code for tree
      // DBH is tree DBH, RunningAvgG is current running average basal area/acre

      int j = iSTEMSCode;
      double CR = CRCoeffs[j][1]/(1+CRCoeffs[j][2]*RunningAvgG)+CRCoeffs[j][3]*
            (1-exp(CRCoeffs[j][4]*DBH));
      if(CR < 0.5) CR=0.5;
      if(CR > 9.5) CR=9.5;
      return CR;
}

double PotentialDBHGrowth(int iSTEMSCode, double DBH, double TreeSI, double CR){
      // Predict a tree's potential DBH growth
      // iSTEMSCode is STEMS species code for tree
      // DBH is tree's DBH, TreeSI is tree's site index, CR is tree's crown ratio code

      int j = iSTEMSCode;
      double PotDBHGrow = PGCoeffs[j][1]+PGCoeffs[j][2]*pow(DBH,PGCoeffs[j][3])+
            PGCoeffs[j][4]*TreeSI*CR*pow(DBH,PGCoeffs[j][5]);
      if(PotDBHGrow < 0.0) PotDBHGrow=0.0;
      return PotDBHGrow;
}

double CompetitionModifier(int iSTEMSCode, double DBH, double StandG, double AADBH) {
      // Predict competition modifier for tree
      // iSTEMSCode is STEMS species code for tree
      // DBH is tree's DBH
      // StandG is stand basal area per acre, AADBH is arithmetic mean DBH of stand
```

D

```
      int j = iSTEMSCode;
      double temp = CMCoeffs[j][0] - StandG;
      double CompMod=0;
      if(temp > 0){
         temp=sqrt(temp/StandG);
         double R = DBH/AADBH;
         double fR = CMCoeffs[j][1]*pow(1-exp(CMCoeffs[j][2]*R),CMCoeffs[j][3])+CMCoeffs[j][4];
         double gAD = CMCoeffs[j][5]*pow(AADBH+1,CMCoeffs[j][6]);
         CompMod=1-exp(-fR*gAD*temp);
      }
      return CompMod;
}

double DBHGrowthAdjustment(int iSTEMSCode, double DBH){
      // Compute additive adjustment factor for DBH growth
      //    There's a bound on DBH for computing adjustment
      // iSTEMSCode is STEMS species code for tree
      // DBH is tree's DBH

      int j = iSTEMSCode;
      double DBHceil=DBH;
      if((j==5 || j==30) && DBH > 10.0)
         // black spruce and non-commercial
         DBHceil=10.0;
      else
         if(DBH > 20.0) DBHceil=20.0;
      double DBHAdj = DACoeffs[j][1]*DBHceil+DACoeffs[j][2]*DBHceil*DBHceil+DACoeffs[j][3];
      return DBHAdj;
}

double ProbabilitySurvival(int iSTEMSCode, double DBH, double DBHGrow){
      // Compute tree probability of survival
      // iSTEMSCode is STEMS species code for tree
      // DBH is tree's DBH
      // DBHGrow is tree's predicted annual DBH growth
      // According to TWIGS (LSGROW.FOR) code, DBH for survival calculation
      //    should be DBH after one year's growth

      int j = iSTEMSCode;
      DBH += DBHGrow;
```

```
      double temp = 0;
      if(DBH > 1)
         temp=PSCoeffs[j][5]*pow(DBH-1,PSCoeffs[j][6])*exp(-PSCoeffs[j][7]*(DBH-1));
      temp += PSCoeffs[j][2]+PSCoeffs[j][3]*pow(DBHGrow,PSCoeffs[j][4]);
      return PSCoeffs[j][1] - (1.0/(1.0+exp(temp)));
}

double TreeTotalHeight(int iSTEMSCode, double DBH, double TreeSI, double StandG){
      // Estimate total tree height
      // iSTEMSCode is STEMS species code for tree
      // DBH is tree's DBH, TreeSI is site index for tree of species iSTEMSCode
      // StandG is stand basal area per acre

      int j = iSTEMSCode;
      return 4.5 + HtCoeffs[j][1]*pow(1.0-exp(-HtCoeffs[j][2]*DBH),HtCoeffs[j][3]) *
                  pow(TreeSI,HtCoeffs[j][4])*pow(StandG,HtCoeffs[j][6]);
}

int STEMS(int iNTrees, double StandG, double AADBH, double RunningAvgG, bool isPlantation,
      int *iSppCode, double *TreeSI, double *DBHProj, double *CRProj, double *ExpFacProj) {

/*    Grow the tree list one year.
      When come in, DBHProj and ExpFacProj are current DBH and expansion factor for tree. When
            leave, they have projected values. CRProj is current crown ratio code for tree; it
            isn't projected because need new running average basal area
      iNTrees = number of trees in tree list
      StandG = current stand basal area
      AADBH = current arithmetic average DBH
      RunningAvgG = running average G for up to last 10 years
      iSppCode = tree internal (converted from user) species codes
      TreeSI = tree (species) site index

      Returns 1 if successful, 0 if an exception occurs
*/
      try {
         for (int i=0; i<iNTrees; i++) {
               int j = iSppCode[i];
               double DBH = DBHProj[i];

               // Compute potential DBH increment
               double PotDBHGrow = PotentialDBHGrowth(j,DBH,TreeSI[i],CRProj[i]);
```

E

```
            // Compute Competition modifier
            // If species is red pine (1), and in a red pine plantation, use species code 26
            //       (STEMS code 27) for competition modifier computation only
            if(j==1 && isPlantation) j=26;

            double CompMod = CompetitionModifier(j,DBH,StandG,AADBH);

            // Set species back to 1 if "red pine in red pine plantation"
            if(j==26) j=1;

            // Compute DBH growth adjustment
            double DBHAdj = DBHGrowthAdjustment(j,DBH);

            // Compute realized DBH growth
            double DBHGrow=PotDBHGrow*CompMod+DBHAdj;
            if(DBHGrow <= 0) DBHGrow = 0.000001;

            // Compute probability of survival
            double ProbSurv = ProbabilitySurvival(j,DBH,DBHGrow);

            // Compute new expansion factor
            ExpFacProj[i] *= ProbSurv;

            // Compute new DBH
            DBHProj[i] += DBHGrow;

            // Compute new crown ratio code
            // Done in calling routine AFTER new running average basal area found

         }
      }
      catch (...) {
            // If any sort of error occurs in STEMS code, come here and indicate by returning 0
            return 0;
      }

      return 1;
}

//    This is the routine that gets called to project the tree list
```

```
      /*    iAge = current stand age (NOT CURRENTLY USED)
            iSI = site index
            iUserSISpp = user species code of site index species
            iNTrees = number of trees in tree list
            iYearsGrow = number of years tree list is to be projected
            isPlantation = TRUE if red pine plantation
                  For a red pine tree in a red pine plantation, modifier is different
            One-dimensional arrays, all zero-based, length iNTrees, all type double except iSppCode
                  iUserSppCode - user species code of tree
                  DBHIn - current DBH of tree
                  HtIn - current height of tree: negative number means not measured
                  CRIn - current crown ratio code for tree: negative number means not measured
                  ExpFacIn - current expansion factor (trees/acre) of tree
                  DBHOut - projected DBH of tree
                  HtOut - projected height of tree
                  ExpFacOut - projected expansion factor (trees/acre) of tree
      */
/*    Return codes:
       1 = Success
      -1 = Invalid SI (user) species code
      -2 = Invalid SI
      -3 = Invalid number of trees in tree list
      -4 = Invalid number of years to grow trees
      -5 = Invalid tree (user) species code
      -6 = Invalid tree DBH
      -7 = Invalid tree height
      -8 = Invalid tree expansion factor
      -9 = Invalid tree crown ratio code

      -20 = STEMS growth routine encountered an error
*/

int _stdcall LSGrowTrees(int iAge, int iSI, int iUserSISpp, int iNTrees, int iYearsGrow,
      bool isPlantation, int *iUserSppCode, double *DBHIn, double *HtIn, double *ExpFacIn,
      double *CRIn, double *DBHOut, double *HtOut, double *ExpFacOut, double *CROut) {

      if(iNTrees < 1 || iNTrees > 200) return -3;
      if(iYearsGrow < 1 || iYearsGrow > 50) return -4;

      // Use user supplied site index (for one SI species) to compute ALL OTHER species SIs
      // Actual array. NumSTEMSSppCodes is __const__.
```

```
//    If don't put enough numbers in initialization brackets, rest set to 0.
double SppSIs[NumSTEMSSppCodes] = {0};

int iSISpp = ConvertSppCode(iUserSISpp);

if(iSISpp < 0) return -1;

if(iSI < 20 || iSI > 100) return -2;

SppSIs[iSISpp] = (double) iSI;

for (int i=0; i<NumSTEMSSppCodes; i++) {
      if(SppSIs[i] < 0.0001) {
            SppSIs[i] = SIConvB0[i][iSISpp] + SIConvB1[i][iSISpp]*SppSIs[iSISpp];
      }
}
// If that didn't work, try converting through aspen __IF__ aspen has been assigned
if(SppSIs[24] > 0.0001) {
      for (int i=0; i<NumSTEMSSppCodes; i++) {
            if(SppSIs[i] < 0.0001) {
                  SppSIs[i] = SIConvB0[i][24] + SIConvB1[i][24]*SppSIs[24];
            }
      }
}
// Any species without a SI set gets site index that user provided
for (int i=0; i<NumSTEMSSppCodes; i++) {
      if(SppSIs[i] < 0.0001) {
            SppSIs[i] = SppSIs[iSISpp];
      }
}

// These are deallocated at end of routine
double *TreeSI = new double[iNTrees];
double *CRAdjust = new double[iNTrees];
int *iSppCode = new int[iNTrees];

double StandG=0;
double SumDBH=0;
double SumExpFac=0;
// Loop through all trees in tree list to set things up
for (int i=0; i<iNTrees; i++) {
```

G

```
	// Convert species codes from user codes to model codes
	int iSpp = ConvertSppCode(iUserSppCode[i]);
	if(iSpp >= 0)
		iSppCode[i]=iSpp;
	else
		return -5;
	// Give tree a site index based on the tree's species
	TreeSI[i]=SppSIs[iSpp];

	double DBH = DBHIn[i];
	if(DBH < 1.0 || DBH > 35.0) return -6;
	DBHOut[i] = DBH;
	// Negative total height means wasn't measured
	if(HtIn[i] >= 0.0)
		if(HtIn[i] < 15.0 || HtIn[i] > 120.0) return -7;
	double ExpFac = ExpFacIn[i];
	if(ExpFac < 0.5 || ExpFac > 150.0) return -8;
	ExpFacOut[i] = ExpFac;
	double CR = CRIn[i];
	// Negative crown ratio code means wasn't measured
	// Crown ratio code further dealt with below after get stand BA
	if(CR >= 0.0)
		if(CR < 0.5 || CR > 9.5) return -9;

	StandG +=  DBH*DBH*ExpFac;
	SumDBH += DBH*ExpFac;
	SumExpFac += ExpFac;
}
double AADBH = SumDBH/SumExpFac;
StandG *= 0.005454;

// Store initial PREDICTED heights in HtOut
// Deal further with crown ratio code inputs
for (int i=0; i<iNTrees; i++) {
	int j = iSppCode[i];
	HtOut[i] = TreeTotalHeight(j,DBHOut[i],TreeSI[i],StandG);

	double CRPred = CrownRatioCode(j,DBHOut[i],StandG);
	double CR = CRIn[i];
	if(CR < 0) {
		CROut[i] = CRPred;
```

```
            CRAdjust[i] = 0;
      } else {
            CROut[i] = CR;
            CRAdjust[i] = CR - CRPred;
      }
}

//Initialize running average basal area array
double RunningBA[11] = {0,0,0,0,0,0,0,0,0,0,0};
RunningBA[1]=StandG;
double RunningAvgBA=StandG;

// Grow the trees in DBH, one year at a time
for (int i=1; i<=iYearsGrow; i++) {
      int ierr =
            STEMS(iNTrees,StandG,AADBH,RunningAvgBA,isPlantation,iSppCode,TreeSI,
                  DBHOut,CROut,ExpFacOut);
      if(ierr != 1) return -20;

      // New stand level attributes
      StandG=0;
      SumDBH=0;
      SumExpFac=0;
      for (int j=0; j<iNTrees; j++) {
            double DBH = DBHOut[j];
            double ExpFac=ExpFacOut[j];
            SumExpFac += ExpFac;
            SumDBH += DBH*ExpFac;
            StandG += DBH*DBH*ExpFac;
      }
      AADBH = SumDBH/SumExpFac;
      StandG *= 0.005454;

      // Update running average basal area array and compute new RunningAvgBA
      int k=10;
      if(i <= 9) k=i+1;
      RunningAvgBA=0;
      for (int j=k; j>=2; j--) {
            RunningBA[j] = RunningBA[j-1];
            RunningAvgBA += RunningBA[j];
      }
```

```
            RunningBA[1]=StandG;
            RunningAvgBA = (RunningAvgBA+StandG)/(double)k;

            // Now get new crown ratio codes for all trees
            for (k=0; k<iNTrees; k++){
                  double CR = CrownRatioCode(iSppCode[k],DBHOut[k],RunningAvgBA) + CRAdjust[k];
                  if(CR < 0.5) CR=0.5;
                  if(CR > 9.5) CR=9.5;
                  CROut[k] = CR;
            }

      }     // Loop through years of growth

      // Predict the heights of the grown trees
      for (int k=0; k<iNTrees; k++) {
            int j = iSppCode[k];
            double Ht = TreeTotalHeight(j,DBHOut[k],TreeSI[k],StandG);
            // If height was measured, new height is initial height plus change in predicted heights
            // HtOut[] was initially filled with initial predicted heights
            // If height wasn't measured, new height is just prediction
            if(HtIn[k] >= 0.0)
                  HtOut[k] = HtIn[k] + (Ht - HtOut[k]);
            else
                  HtOut[k] = Ht;
      }

      //Deallocate stuff setup with __new__ above
      delete[] TreeSI;
      TreeSI = NULL;
      delete[] CRAdjust;
      CRAdjust = NULL;
      delete[] iSppCode;
      iSppCode = NULL;

      return 1;
}
```