Palaeontologia Electronica
http://palaeo-electronica.org

SOURCE CODE FOR THEORETICAL MORPHOLOGIC SIMULATION OF HELICAL COLONY FORM IN THE BRYOZOA

David M. Raup, George R. McGhee Jr., and Frank K. McKinney

ABSTRACT

The analytic techniques of theoretical morphology involve the computer simulation of both existent and nonexistent biological form, and the subsequent analysis of the distribution and evolution of biological form within theoretical morphospaces. In this contribution we make available the computer source code used in the simulation of hypothetical helical colony forms found in the Bryozoa—colony forms that have been convergently evolved multiple times in the evolutionary history of the phylum.

David M. Raup. Department of the Geophysical Sciences, University of Chicago, Chicago, Illinois 60637 USA draup@itol.com
George R. McGhee Jr. Department of Geological Sciences, Wright-Rieman Laboratories, Rutgers University, New Brunswick, New Jersey 08903 USA mcghee@rci.rutgers.edu correspondence author
Frank K. McKinney. Department of Geology, Appalachian State University, Boone, North Carolina 28608 USA mckinneyfk@appstate.edu

KEY WORDS: Theoretical morphology; helical colonies; form simulation; Bryozoa

INTRODUCTION: THE ANALYTICAL POTENTIAL OF THEORETICAL MORPHOLOGY

"*A second reason for giving theoretical morphology a good run for its money is that, in many ways, it is unlike other sciences. In* A New Kind of Science (2002) *Stephen Wolfram has controversially suggested that science in the future will be more concerned with algorithms than laws. One wonders how that could be true of all science, but if there is any science of which this is clearly true, that science is theoretical morphology.*"

Maclaurin (2003, p.465)

The inspiration for "theoretical morphology" (Raup and Michelson 1965) lies in the morphogenetic simulation of actual growth processes in nature and in the search for the general growth "rules" governing the generation of biological form. In this aspect of the discipline, it can be viewed as an algorithmic approach to the analysis of evolutionary development, as opposed to an experimental one. Yet theoretical morphology is much more than "Computer EvoDevo"! The developmental models created in a theoretical morphologic analysis can be used not only to simulate existing biological form, but also to generate hypothetical

PE Article Number: 9.2.7A
Copyright: Paleontological Society September 2006
Submission: 10 September 2004 Acceptance: 19 June 2006

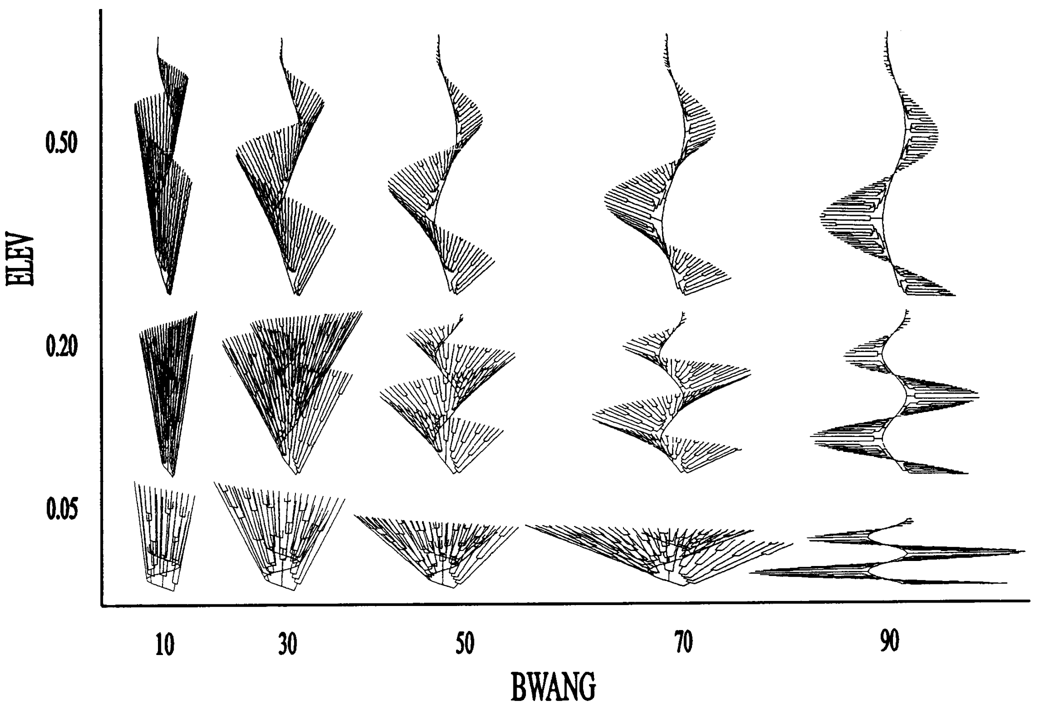

Figure 1. A two-dimensional slice through the three-dimensional theoretical morphospace of helical colony morphologies illustrating the range in form produced by varying the parameters *BWANG* and *ELEV*, where the parameter *XMIN* is held constant at 20. The simulations were produced by the computer program given in this paper. For purposes of visual presentation the morphospace axes are not arithmetic.

biological form that, although geometrically possible, has never been evolved by any known living organisms on Earth. The very existence of nonexistent form is of immediate significance to the evolutionary theorist—why hasn't nature produced this form? The answer to this question leads both the theorist and experimentalist to the analysis of the roles of adaptation, natural selection, and evolutionary constraint in the shaping of biological form (for a more extensive discussion of the analytic techniques of theoretical morphology see McGhee 1999, 2001, 2007).

Many years ago, we became interested in the intricate and elegant helical colony forms evolved by species of the bryozoan genus *Archimedes* (McKinney and Raup 1982). Although *Archimedes* is long extinct, similar helical colony forms have been convergently evolved in species of the living genus *Bugula turrita* (McKinney 1980). In fact, further analysis has revealed that helical colony forms have been convergently, independently, evolved no less that six separate times within the Phylum Bryozoa (McKinney and McGhee 2003). Why have bryozoan colonies repeatedly evolved helically coiled colonies? And how do they produce these colony forms? Lastly, are there geometrically possible helical forms that they have never produced, and why not? These questions are of the very essence of theoretical morphology.

In our initial studies, we demonstrated that the helical colony form—although very intricate in appearance—could be computer simulated by a model having only five geometric parameters (McKinney and Raup 1982). Thus, far from being difficult to develop, the fact that the morphogenesis of helical colonies is algorithmically simple may be a large factor in its repeated evolution within the bryozoans. In our subsequent studies, we have taken the geometric parameters of the helical colony model and created a theoretical morphospace of hypothetical, and potential, bryozoan colony form (Figure 1). Exploration of this morphospace to date has revealed that it is very unevenly filled by actual bryozoans through geologic time, and that the "full" (Figure 2) versus "empty" (Figure 3) regions of the morphospace appear largely to be

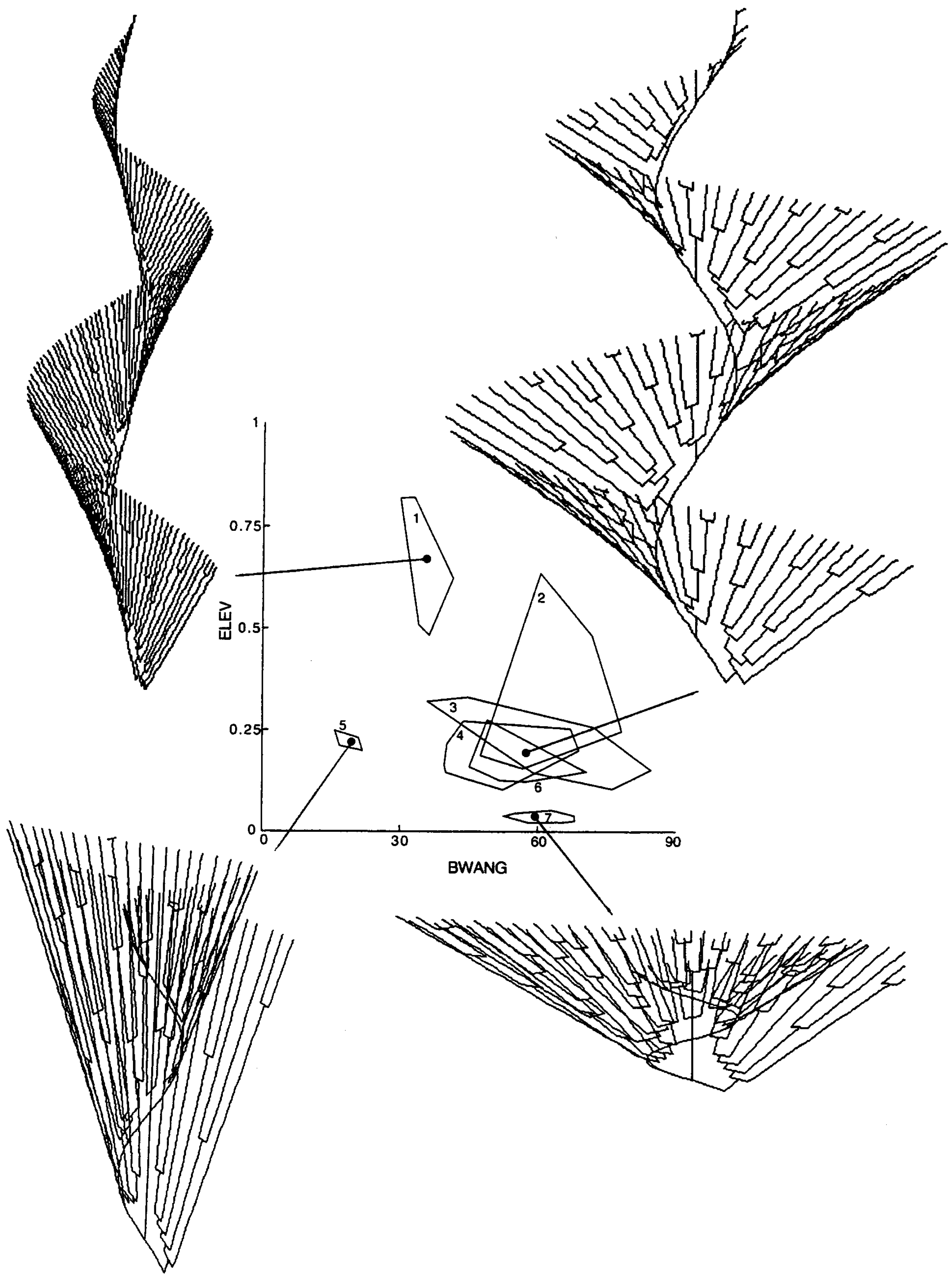

Figure 2. Simulations of existent helical colony morphologies in fossil and living bryozoans. The polygons in the morphospace illustrate the measurement field of measurements taken from: (1) Paleozoic *Archimedes laxus*, (2) Paleozoic basinal *Archimedes* ecomorphs, (3) Paleozoic back-shoal *Archimedes* ecomorphs, (4) living *Bugula* colonies, except *B. plumosa* colonies, (5) living *Bugula plumosa*, (6) Eocene *Crisidmonea archimediformis*, and (7) living *Retiflustra cornea*. Data polygons from McKinney and McGhee (2003).

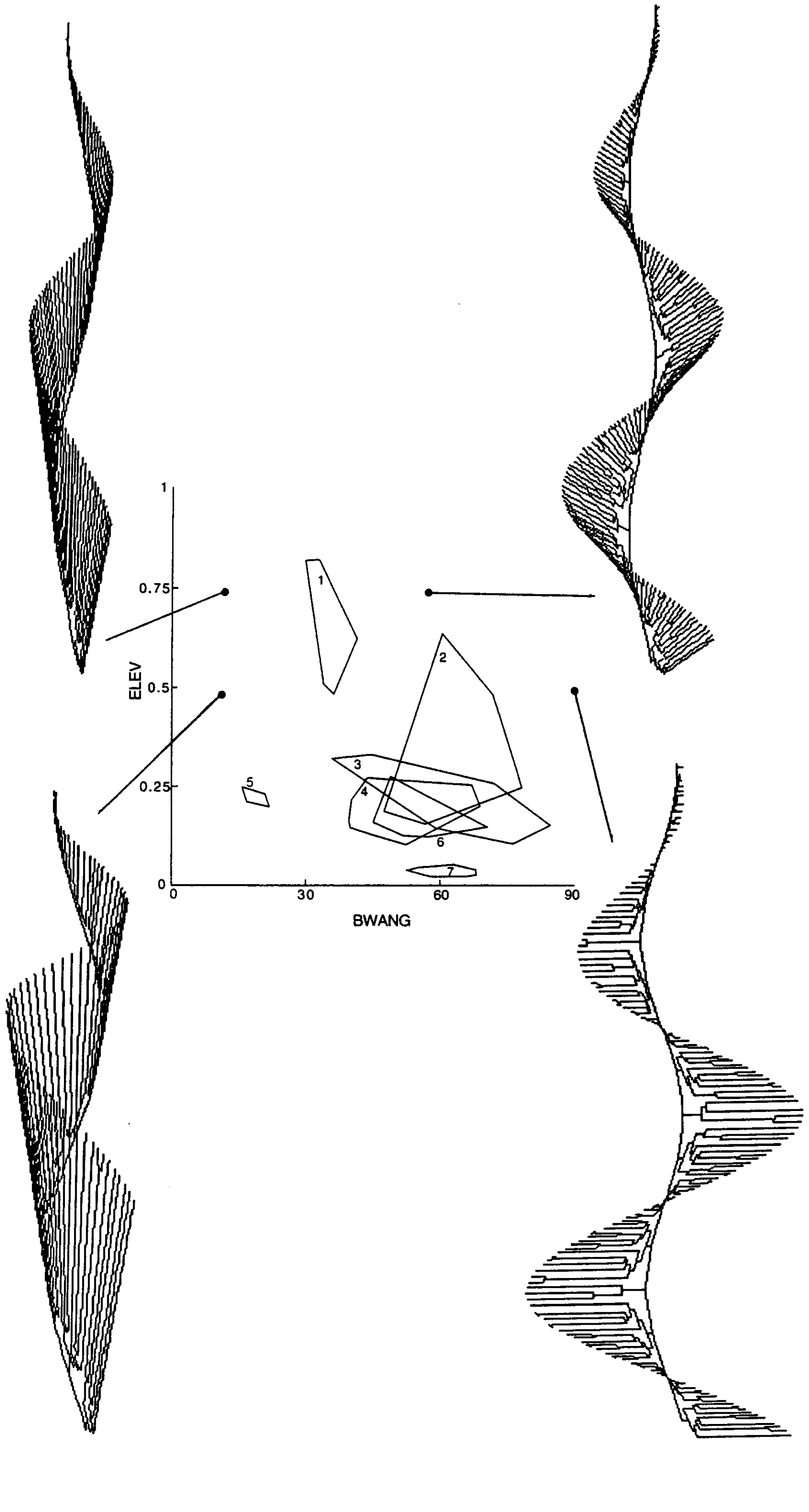

Figure 3. Simulation of nonexistent helical colony morphologies. Data polygons of real fossil and living bryozoan colonies as in Figure 2.

due to functional constraints – that is, the existent forms make efficient filtering geometries, and the nonexistent forms do not (McGhee and McKinney 2000, 2002; McKinney and McGhee 2003, 2004).

In the spirit of Stephen Wolfram's (2002) vision of a science "more concerned with algorithms than laws," we are making available the algorithms of the helical colonies model in this brief paper. We feel we have just scratched the surface of understanding why these intricate forms have convergently reappeared during bryozoan evolution, and we wish to make the code available for others to experiment with. We also encourage others to take the code and modify it to simulate other helical aspects of form in nature, beyond those found (or not found) within the Bryozoa.

THE HELICAL COLONY MODEL

The helical colony model has five basic geometric parameters (see McKinney and Raup 1982, figure 3). Two parameters determine the geometry of the central helix of the colony:

1. *ELEV*: the rate of climb of the helix (a measure of the repeat distance, or pitch, of the helix), and
2. *RAD*: the radius of the helix.
   Three parameters determine the geometric arrangement of the filtration-sheet whorls that are attached to the helix axis and the pattern of branching within the whorls:
3. *ANG*: the radial angle between the initial branches that originate from the central helix of the colony (a measure of the density of the initial branches in the core of the colony),
4. *BWANG*: the angle between the branches in the filtration-sheet whorls and the axis of the central helix of the colony, and
5. *XMIN*: the minimum distance between the outer two of three adjacent branches within the filtration-sheet whorls, at the point at which the central branch bifurcates. This parameter controls the frequency of branching, and hence the density of the branches, within the filtration-sheet whorls.

The geometric parameters *ELEV*, *BWANG*, and *XMIN* have been used to construct a three-dimensional morphospace of hypothetical helical colony forms in bryozoans (McKinney and Raup 1982), a morphospace that has subsequently been used in the analysis of actual helical colony forms in both extinct and living bryozoans (McGhee and McKinney 2000, 2002; McKinney and McGhee 2003, 2004).

THE HELICAL COLONY COMPUTER PROGRAM

The helical colony computer program is written in VISUAL BASIC, the code of which is provided in the Appendix (the code was originally written in VB 3.0, but subsequently modified and expanded, and now runs in VB 6.0). An executable version of the code is also available in the Appendix. This program was used in producing the simulations shown in Figure 1. The default values of the parameters are set so as to produce an initial simulation approximating that of the colony form illustrated in figure 2 of McKinney and Raup (1982); that is, *ELEV* = 0.17, *BWANG* = 60, *XMIN* = 20, *ANG* = 45, and *RAD* = 2. These parameter settings can be changed by simply pressing the desired button that appears on the monitor screen adjacent to each simulation, whereupon the program will prompt the user that the parameter values have been changed. Press the "Run Again" button, and the new simulation will appear on the screen.

In addition to the original five geometric parameters of McKinney and Raup (1982), this version of the program also allows the user to specify the number of whorls produced in the simulation and the number of growth increments (defaults are set at 2.8 whorls and 25 growth increments). The user may also change the growth gradient at the distal (uppermost) whorl of the colony simulation. The default value is set at "Growth gradient" = 1, which tapers the length of the branches in the distal whorl from the colony diameter down to zero at the uppermost tip of the simulation. This gradient in branch length reduction may be reduced by setting the growth gradient to a value < 1; a value of zero eliminates the growth gradient, and the branches in the distal whorl will grow to the same diameter as the rest of the colony.

The parameter *XMIN* is subject to Gaussian random variation to simulate natural growth variation seen in actual helical colonies (see McKinney and Raup 1982, p. 105). Each time the program is run, a new random number seed is used. The user may also specify the random number seed to be used in the simulation by pressing the "Random number seed" button.

Three-dimensional stereo pairs of a simulation (see McKinney and Raup 1982, figure 2) may be produced by rotating the simulation about the z-axis. This can be accomplished by using the

"Rotation" button: first, run one simulation with Rotation = +3 (degrees), print it, and then run a second simulation with Rotation = –3 (degrees). The perspective from which the simulation is viewed may also be changed by using the "Tilt" button, which rotates the simulation either clockwise or anticlockwise on the screen. The values for the angular program variables *BWANG*, *ANG*, Rotation, and Tilt are all input as degrees, but all internal computations in the program are in radians.

Scaling for the screen display is in twips, using part of the screen area (7000 high by 7000 wide). The program executes a "trial" run to determine the limits of the simulation picture and hence its scaling. The program is written to assume a standard monitor of 1024 by 768 pixel area, with 15 twips per pixel. Other monitors will change the graphic output and font size; check (with WINDOWS) to set screen resolution at 1024 by 768 for best graphic results. If the user further desires, the position of the simulation on the computer screen can also be moved by altering the final two lines of code in the Convert Subroutine (i.e., xp = 4000 + (xp – Hmin) * Pscale, and zp = 8000 – (Vmax – zp) * Pscale), which determine the X-Y plot coordinates on the screen (default values are 4000 and 8000, respectively).

HOW THE ALGORITHMS WORK

The program first executes a trial run of the computations in order to scale the simulation to the available computer screen area (discussed above). This scaling run, in which nothing is plotted, is accomplished by the outermost Do...While loop of the program in which the value of "trial" and "scaled" go from a value of zero to one. Actual plotting of the computations begins when the value of "trial" exceeds one, during which the Convert Subroutine is called in order to convert the cylindrical coordinate computations into cartesian coordinates for screen plotting.

The growth of the central helical axis in the colony simulation, and position of the nodes on the helix for the basal branches of the colony, is calculated in two sequential For...Next loops in the "Make Central Helix" and "Make Initial Growing Tips" sections of the program. The geometry of the helix, and placement of the basal branch nodes, is determined by the geometric parameters *ELEV*, *RAD*, *ANG*, *WHORLS*, and *GROW*. The values of the model parameters *ELEV*, *ANG*, and *WHORLS* (the number of whorls in the helix) can be specified by the user for each simulation. The parameters *RAD* and *GROW* (the linear magnitude of each growth increment) are set as constants in the program (at the value of 2 and 0.5, respectively). If the user wishes to change these default values, they may be changed directly in the code.

The growth of the branches in the colony simulation is calculated in the "BEGIN MAIN PROGRAM" section of the code. This entire section consists of two subsections: (1) algorithms determining the growth of each branch tip, and (2) algorithms determining when a branch bifurcates to produce two new branches:

1. In the first subsection, the growth of all branch tips "j" within a given growth increment "k" is calculated in two sequential For...Next loops: the first calculates and plots the positions of the new tips relative to that of the old tips, and the second rewrites the old tip positions in terms of the new so that the next round of new tip positions may be calculated.
2. In the second subsection, bifurcations of the growing branches are calculated in the "Loop for Bifurcations" section of the code. This section consists of an outer Do...Until loop, and three sequential inner For...Next loops. The first For...Next loop calculates the spatial positions of the growing branch tips and determines when a tip must bifurcate (i.e., when the model parameter distance *XMIN* is reached). Also within this For...Next loop is contained the subroutine that subjects the value of *XMIN* to Gaussian random variation (as discussed above). This section of the code calculates the positions of the two new tips that bifurcate from the old tip and draws the simulation branch crossbar between them. The next two For...Next loops save the positions of the new tips and rewrite the old tip positions in terms of the new so that the next round of new tip positions may be calculated. The outer Do...Until loop iterates these three inner For...Next loops, conducting a census across all growing tips within the colony. When this census is complete, the Do...Until loop ends, and the outermost For...Next loop of the program begins again for the next growth increment "k."

SETTING UP THE PROGRAM

To set up the program, follow standard Visual Basic procedure: create a Form and in its Properties Box set its BackColor property to white. On the left margin of the Form add 13 small Command buttons, and label their Caption properties consecutively "Run," "Exit," "Print," "ANG," "XMIN," "BWANG," "ELEV," "Number of whorls," "Number of growth increments," "Random number seed," "Rotation," "Tilt," and "Growth gradient (1 = full, 0 = none)." Adjacent to the Command buttons add a small Label Box, and label it "Parameters changed – Press Run Again button for new morphologic simulation."

After the Form has been configured, double-click on the Command1 button ("Run"), and type the code in the Code Box. Then click on the Start button on the tool bar (or in the Run pull-down menu) to run the program. If you leave the AutoRedraw property of the Form at the default setting of "False" the simulation will grow in real time on the computer screen. In order to print the simulation, you must set the AutoRedraw property of the Form to "True," but this results in the simulation appearing on the screen only in its final growth stage.

FUTURE WORK

"[T]he development of software for specific use as a teaching aid in theoretical . . . morphology is as essential as leading-edge research to ensure the continued development of this field."

Savazzi (1995, p. 238)

The code given in this paper was written by two theoretical morphologists (Raup, McGhee) in close interaction with a bryozoan biologist (McKinney) and has been produced by an evolutionary process of trial-and-error in writing algorithms that would best simulate the organic forms actually seen in helical bryozoans in nature. We hope that it will not only stimulate further research in the analysis of the evolution of helical colony form, but that it will also be used as a teaching aid in illustrating the analytical techniques of theoretical morphology. Specifically, future instructors might begin with the existing code, and seek to even more closely model the form seen in helical bryozoans in a series of laboratory exercises. One obvious aspect of bryozoan form that has not been simulated at present is the presence of dissepiments – cross-bars between the branches in the colony. Another instructive approach would be to explore just how far the limits of the program could be pushed in producing nonexistent form. For example, the maximum values of *BWANG* shown in Figure 1 stop at 90 degrees – yet the program is capable of simulating angles that exceed that value.

Yet another future possible avenue of both research and pedagogical significance would be to see if similar simulations of form could be produced with totally different algorithmic approaches, such as the usage of L-systems (Prusinkiewicz and Lindenmayer 1996; for a detailed discussion of the potential uses of L-systems in theoretical morphology see McGhee 1999). L-systems have been used to produce very realistic simulations of branching-form in plants, and we see no reason why the approach could not also fruitfully be applied to branching-form in animals. Lastly, it might also be instructive to see if the algorithms could be simplified by using existing modeling software such as *Mathematica* (Wolfram 2002).

ACKNOWLEDGMENTS

We thank an anonymous reviewer for helpful comments on the manuscript, which enabled us to improve the paper. Acknowledgment also is made to the Donors of the Petroleum Research Fund, administered by the American Chemical Society, for partial support of this research, and to the National Science Foundation.

REFERENCES

Maclaurin, J. 2003. The good, the bad and the impossible. *Biology and Philosophy*, 18:463-476.

McGhee, G.R. 1999. *Theoretical Morphology: the Concept and Its Applications*. Columbia University Press, New York.

McGhee, G.R. 2001. Exploring the spectrum of existent, nonexistent and impossible biological form. *Trends in Ecology and Evolution*, 16:172-173.

McGhee, G.R. 2007. *The Geometry of Evolution: Adaptive Landscapes and Theoretical Morphospaces*. Cambridge University Press, Cambridge (England).

McGhee, G.R. and McKinney, F.K. 2000. A theoretical morphologic analysis of convergently evolved erect helical colony form in the Bryozoa. *Paleobiology*, 26:556-577.

McGhee, G.R. and McKinney, F.K. 2002. A theoretical morphologic analysis of ecomorphologic variation in *Archimedes* helical colony form. *Palaios*, 17:556-570.

McKinney, F.K. 1980. Erect spiral growth in some living and fossil bryozoans. *Journal of Paleontolog*y, 54:597-613.

McKinney, F.K. and McGhee, G.R. 2003. Evolution of erect helical colony form in the Bryozoa: phylogenetic, functional, and ecological factors. *Biological Journal of the Linnean Society*, 80:235-260.

McKinney, F.K. and McGhee, G.R. 2004. Erratum: Evolution of erect helical colony form in the Bryozoa. *Biological Journal of the Linnean Society* 81:619-620.

McKinney, F.K. and Raup, D.M. 1982. A turn in the right direction: simulation of erect spiral growth in the bryozoans *Archimedes* and *Bugula*. *Paleobiology*, 8:101-112.

Prusinkiewicz, P. and Lindenmayer, A. 1996. *The Algorithmic Beauty of Plants*. Springer-Verlag, Berlin.

Raup, D.M. and Michelson, A. 1965. Theoretical morphology of the coiled shell. *Science*, 147:1294-1295.

Savazzi, E. 1995. Theoretical shell morphology as a tool in constructional morphology. *Neues Jahrbuch für Geologie und Paläontologie, Abhandlungen* 195:229-240.

Wolfram, S. 2002. *A New Kind of Science*. Wolfram Media, Champaign, IL.

APPENDIX: THE CODE[1]

```
’ In order of appearance in program:
Dim trial, scaled, CH
Dim ELEV, BWANG, ANG, XMIN
Dim GRAD, NG, WHORLS, StartSeed
Dim Yr, YrD, Zr, ZrD  ’ For Sub Convert
Dim sigma, GROW, RAD, aXMIN
Dim seed, Bseed, Nit
Dim Pscale
Dim Hmin, Hmax, Vmin, Vmax
Dim pi, x, y, z
Dim j As Single
Dim jr    ’ To carry angle "j" to Sub Convert
Dim jj, k, kk
Dim tip(2000, 4), Ntip(2000, 4), Ttip(2000, 4)
’ 4 tip subscripts = x, y, z, rotation angle "j"
Dim xp, yp, zp’ Plotting coordinates from Sub Convert
Dim g
Dim X1, Y1, z1
Dim xx, yy, zz
Dim A1, A2
Dim tip1x, tip1y, tip2x, tip2y

Private Sub Command1_Click()
trial = 0
Cls
scaled = 0
Do
   WindowState = 2
   Label1.Visible = False
   If CH <> 1 Then’ CH = 1 indicates default values changed by user input
’----------Set Default Parameter and Variable Values----------
       ELEV = 0.17’ MODEL PARAMETERS of McKinney & Raup (1982)
       BWANG = 60
       ANG = 45
       XMIN = 20
                 ’ OTHER VARIABLES:
       GRAD = 1’ Growth gradient at distal tip of helix
       NG = 25   ’ Number of growth increments
       WHORLS = 2.8’ Number of whorls in the helix
       StartSeed = 54321’ Initial random number seed
       YrD = 0   ’ Initial rotation around the y-axis
       ZrD = 0   ’ Initial rotation around the z-axis
   ElseIf CH = 1 Then End If
’----------Constants (can be changed in the source code)----------
   sigma = 0.05’ Limit for random number generator
   GROW = 0.5’ Linear growth per increment
   RAD = 2   ’ Radius of central helix
   aXMIN = XMIN / 2’ Scaled to McKinney & Raup (1982) simulations
’----------Increment random number seed ("Try again" button)----------
   Bseed = StartSeed + Nit
   seed = Bseed
   Nit = Nit + 1
   Randomize (seed)
’----------Scaling Trial Run (without plotting; checks screen scale)----------
   trial = trial + 1
```

1.Zipped program files are available from the PE website at this URL: http://palaeo-electronica.org/2006_2/helical/index.html

```
    If scaled = 0 Then
        Pscale = 1
        Hmin = 10000: Hmax = -10000: Vmin = 10000: Vmax = -10000
    ElseIf scaled <> 0 Then End If
    pi = 4 * Atn(1)' Establish value of pi
    Offset = pi / 4' Default orientation, McKinney & Raup (1982, fig. 2)
    Command1.Caption = "Please wait . . . "
'----------Make Central Helix----------
    x = RAD * Cos(Offset)
    y = -RAD * Sin(Offset)
    z = -ELEV * 360 / 5 * Offset / (2 * pi)
    jr = Offset: Call convert
    If trial > 1 Then Line (xp, zp) - (xp, zp)
    For j = Offset To WHORLS * 2 * pi + 2 * pi / 36 + 0.01 Step 2 * pi / 36
    ' Line segments 10 degrees each
        x = RAD * Cos(j)
        y = -RAD * Sin(j)
        z = -ELEV * 360 / 5 * j / (2 * pi)
        jr = j: Call convert
        If trial > 1 Then Line  - (xp, zp)
        jj = j
    Next j
'----------Make Initial Growing Tips ANG degrees apart----------
'----------(jj is total rotation through all whorls in radians)----------
    k = 1
    For j = Offset to jj + 0.01 Step 2 * pi / (360 / ANG)
        k = k + 1
        x = (GROW + RAD) * Cos(j)' New tip
        y = -(GROW + RAD) * Sin(j)
        jr = j: Call convert: If trial > 1 Then Line (xp, zp) - (xp, zp)
        If trial <= 1 Then
            If zp < Vmin Then Vmin = zp: If zp > Vmax Then Vmax = zp
            If xp > Hmax Then Hmax = xp: If xp < Hmin Then Hmin = xp
        ElseIf trial > 1 Then End If
        tip(k, 1) = x' Record growing tip
        tip(k, 2) = y
        tip(k, 4) = j' Rotation angle for this tip
        x = RAD * Cos(j)' Point on helix
        y = -RAD * Sin(j)
        jr = j: Call convert: If trial > 1 Then Line  -(xp, zp)
        If trial <= 1 Then
            If zp < Vmin Then Vmin = zp: If zp > Vmax Then Vmax = zp
            If xp > Hmax Then Hmax = xp: If xp < Hmin Then Hmin = xp
        ElseIf trial > 1 Then End If
        kk = j
    Next j
'----------(invisible boundaries to avoid edge effects at termini of helix)----------
' start boundary
    x = (GROW + RAD) * Cos(Offset - pi / (360 / ANG))
    y = -(GROW + RAD)  * Sin(Offset  - pi / (360 / ANG))
' record starting boundary
    tip(1, 1) = x: tip(1, 2) = y: tip(1, 4) = -pi / (360 / ANG) + Offset
' end boundary
    x = (GROW + RAD) * Cos(kk + pi / (360 / ANG))
    y = -(GROW + RAD) * Sin(kk + pi / (360 / ANG))
' record ending boundary
    tip(k + 1, 1) = x: tip(k + 1, 2) = y: tip(k + 1, 4) = kk + pi / (360 / ANG)
    Ntips = k + 1
'----------BEGIN MAIN PROGRAM (Grow All Tips in Colony)----------
```

```
    For k = 2 To NG’ Calculate each growth increment "k"
        For j = 1 To Ntips’ For each growing tip at "j"
            If tip(j, 4) < jj - 2 * pi Or jj < 2 * pi Then
’ Normal growth if WHORLS <= 1
               Ntip(j, 1) = (k * GROW + RAD)  * Cos(tip(j, 4))
               Ntip(j, 2) = -(k * GROW + RAD) * Sin(tip(j, 4))
            ElseIf tip(j, 4) >= jj - 2 * pi And GRAD > 0 Then
’ Reduced growth for final whorl (using GRAD)
               Ntip(j, 1) = (k * GROW * (1 - GRAD * (tip(j, 4) - _
                  (jj - 2 * pi)) / (2 * pi)) + RAD) * Cos(tip(j, 4))
               Ntip(j, 2) = -(k * GROW * (1 - GRAD * (tip(j, 4) - _
                  (jj - 2 * pi)) / (2 * pi)) + RAD) * Sin(tip(j, 4))
            End If
            Ntip(j, 4) = tip(j, 4)
’ Plot new tip, end of old tip
            x = tip(j, 1): y = tip(j, 2): jr = tip(j, 4): Call convert
            If j > 1 And j < Ntips And trial > 1 Then Line (xp, zp) - (xp, zp)
            If trial <= 1 Then
               If zp < Vmin Then Vmin = zp
If zp > Vmax Then Vmax = zp
               If xp > Hmax Then Hmax = xp
If xp < Hmin Then Hmin = xp
            ElseIf trial > 1 Then End If
’ Start of new tip
            x = Ntip(j, 1): y = Ntip(j, 2): jr = tip(j, 4): Call convert
            If j > 1 And j < Ntips And trial > 1 Then Line  -(xp, zp)
            If trial <= 1 Then
               If zp < Vmin Then Vmin = zp: If zp > Vmax Then Vmax = zp
               If xp > Hmax Then Hmax = xp: If xp < Hmin Then Hmin = xp
            ElseIf trial > 1 Then End If
        Next j
’ Rewrite tip(j, 1), tip(j, 2), & tip(j, 4)
        For j = 1 To Ntips
            tip(j, 1) = Ntip(j, 1)
            tip(j, 2) = Ntip(j, 2)
            tip(j, 4) = Ntip(j, 4)
        Next j
’----------Loop for Bifurcations due to XMIN----------
        LastSplit = 0
        Do     ’ Census all growing tips for those that need to bifurcate
            For j = LastSplit + 2 To Ntips - 1
               x = tip(j - 1, 1): y = tip(j - 1, 2): jr = tip(j - 1, 4): Call convert
’ x-y-z position of point at angle = j - 1
               X1 = xp: Y1 = yp: z1 = zp
’ x-y-z position of point at angle = j + 1
               x = tip(j + 1, 1): y = tip(j + 1, 2): jr = tip(j + 1, 4): Call convert
               xx = xp - X1: yy = yp - Y1: zz = zp - z1
’ Distance between points in 3D
               dist = Sqr(xx * xx + yy * yy + zz * zz)
’ Get a random number from normal distribution
               Call Rng
’ Limit deviation from XMIN to 0.5
               If Abs(g) > 0.5 Or g = 0 Then Call Rng
’ Bifurcate if the following condition holds:
               If dist / Pscale > (1 - g) * (aXMIN / 5) Then Exit For
            Next j
            If j >= Ntips - 1 Then Exit Do
’ Calculate bifurcate at "j":
            A1 = tip(j, 4) - (1 / 6) * (tip(j + 1, 4) - tip(j - 1, 4))
```

```
          A2 = tip(j, 4) + (1 / 6) * (tip(j + 1, 4) - tip(j - 1, 4))
          If tip(j, 4) < jj - 2 * pi Or jj < 2 * pi Then
' Normal growth if WHORLS <= 1
             tip1x = (k * GROW + RAD) * Cos(A1)
             tip1y = -(k * GROW + RAD) * Sin(A1)
             tip2x = (k * GROW + RAD) * Cos(A2)
             tip2y = -(k * GROW + RAD) * Sin(A2)
          ElseIf tip(j, 4) >= jj - 2 * pi And GRAD > 0 Then
' Reduced growth for final whorl (using GRAD)
             tip1x = (k * GROW * (1 - GRAD * (tip(j, 4) - _
                (jj - 2 * pi)) / (2 * pi)) + RAD) * Cos(A1)
             tip1y = -(k * GROW * (1 - GRAD * (tip(j, 4) - _
                (jj - 2 * pi)) / (2 * pi)) + RAD) * Sin(A1)
             tip2x = (k * GROW * (1 - GRAD * (tip(j, 4) - _
                (jj - 2 * pi)) / (2 * pi)) + RAD) * Cos(A2)
             tip2y = -(k * GROW * ( 1 - GRAD * (tip(j, 4) - _
                (jj - 2 * pi)) / (2 * pi)) + RAD) * Sin(A2)
          End If
' Draw crossbar
          x = tip1x: y = tip1y: jr = A1: Call convert
          If k < NG And trial > 1 Then Line (xp, zp) - (xp, zp)
          x = tip2x: y = tip2y: jr = A2: Call convert
          If k < NG And trial > 1 Then Line  -(xp, zp)
' Save tips in Ttip( )
          For m = 1 To Ntips
             Ttip(m, 1) = tip(m, 1)
             Ttip(m, 2) = tip(m, 2)
             Ttip(m, 4) = tip(m, 4)
          Next m
' Rewrite tip( ), inserting new tips
          tip(j, 1) = tip1x: tip(j, 2) = tip1y: tip(j, 4) = A1
          tip(j + 1, 1) = tip2x: tip(j + 1, 2) = tip2y: tip(j + 1, 4) = A2
          For n = j + 1 To Ntips
             tip(n + 1, 1) = Ttip(n, 1)
             tip(n + 1, 2) = Ttip(n, 2)
             tip(n + 1, 4) = Ttip(n, 4)
          Next n
          Ntips = Ntips + 1
          LastSplit = j + 1
' Return to bifurcation census:
       Loop Until j = Ntips - 1
' Return for next growth increment:
   Next k
   If trial = 1 Then scaled = 1
' Scaling trial complete:
Loop While trial = 1
'----------END MAIN PROGRAM----------
'----------Screen display of parameter values----------
CurrentX = 100: CurrentY = 8000
Print "   ANG = "; ANG
Print "   XMIN = "; XMIN
Print "   BWANG = "; BWANG
Print "   ELEV = "; ELEV
Print " "
Print "   Whorls: "; WHORLS
Print "   Number of growth increments: "; NG
Print "   Growth gradient: "; GRAD
Print "   Rotation: "; ZrD
Print "   Tilt: "; YrD
Print "   Starting Seed = "; StartSeed
CurrentY = 7500
Beep
Command1.Caption = "Run Again"
Command2.Visible = True
Command3.Visible = True
Command4.Visible = True
Command5.Visible = True
Command6.Visible = True
Command7.Visible = True
Command8.Visible = True
Command9.Visible = True
Command10.Visible = True
Command11.Visible = True
Command12.Visible = True
Command13.Visible = True
scaled = 0
End Sub

’----------COMMAND BUTTONS----------
Private Sub Command10_Click()
’----------Random Number Seed Button----------
Lt = InputBox ("Random number seed", "HIT ENTER for no change", StartSeed, 4000, 4000)
If Len(Lt) = 0 Then Exit Sub
CH = 1 ’ bypass default value of seed
StartSeed = Lt
Command1.SetFocus
Label1.Visible = True
End Sub

Private Sub Command11_Click()
’----------Rotation Angle (degrees) Button----------
Lt = InputBox("Rotation in degrees about vertical axis (negative for clockwise)", _
       "HIT ENTER for no change", ZrD, 4000, 4000)
If Len(Lt) = 0 Then Exit Sub
CH = 1 ’ bypass default value of rotation
ZrD = Lt
Command1.SetFocus
Label1.Visible = True
End Sub

Private Sub Command12_Click()
’----------Tilt Angle (degrees) Button----------
Lt = InputBox("Tilt in degrees (negative for clockwise)", _
       "HIT ENTER for no change", YrD, 4000, 4000)
If Len(Lt) = 0 Then Exit Sub
CH = 1 ’ bypass default value of tilt
YrD = Lt
Command1.SetFocus
Label1.Visible = True
End Sub

Private Sub Command13_Click()
’----------Growth Gradient Button----------
’ GRAD = 0 for no growth gradient
’ GRAD = 1 for full gradient (i.e., zero growth at distal tip)
Lt = InputBox("Growth gradient", "HIT ENTER for no change", GRAD, 4000, 4000)
If Len(Lt) = 0 Then Exit Sub
CH = 1 ’ bypass default value of gradient
```

```
GRAD = Lt
Command1.SetFocus
Label1.Visible = True
End Sub

Private Sub Command2_Click()
End
End Sub

Private Sub Command3_Click()
Command1.Visible = False
Command2.Visible = False
Command3.Visible = False
Command4.Visible = False
Command5.Visible = False
Command6.Visible = False
Command7.Visible = False
Command8.Visible = False
Command9.Visible = False
Command10.Visible = False
Command11.Visible = False
Command12.Visible = False
Command13.Visible = False
Form1.PrintForm
Printer.EndDoc
Command1.Visible = True
Command2.Visible = True
Command3.Visible = True
Command4.Visible = True
Command5.Visible = True
Command6.Visible = True
Command7.Visible = True
Command8.Visible = True
Command9.Visible = True
Command10.Visible = True
Command11.Visible = True
Command12.Visible = True
Command13.Visible = True
End Sub

Private Sub Command4_Click()
'----------Model Parameter ANG Button----------
Lt = InputBox("ANG in degrees", "HIT ENTER for no change", ANG, 4000, 4000)
If Len(Lt) = 0 Then Exit Sub
CH = 1 ' bypass default value of ANG
ANG = Lt
Command1.SetFocus
Label1.Visible = True
End Sub

Private Sub Command5_Click()
'----------Model Parameter XMIN Button----------
Lt = InputBox("XMIN", "HIT ENTER for no change", XMIN, 4000, 4000)
If Len(Lt) = 0 Then Exit Sub
CH = 1 ' bypass default value of XMIN
XMIN = Lt
aXMIN = XMIN / 2
Command1.SetFocus
Label1.Visible = True
```

```
End Sub

Private Sub Command6_Click()
'----------Model Parameter BWANG Button----------
Lt = InputBox("BWANG in degrees", "HIT ENTER for no change", BWANG, 4000, 4000)
If Len(Lt) = 0 Then Exit Sub
CH = 1 ' bypass default value of BWANG
BWANG = Lt
Command1.SetFocus
Label1.Visible = True
End Sub

Private Sub Command7_Click()
'----------Model Parameter ELEV Button----------
Lt = InputBox("ELEV", "HIT ENTER for no change", ELEV, 4000, 4000)
If Len(Lt) = 0 Then Exit Sub
CH = 1 ' bypass default value of ELEV
ELEV = Lt
Command1.SetFocus
Label1.Visible = True
End Sub

Private Sub Command8_Click()
'----------Number of Whorls Button----------
Lt = InputBox("Number of Whorls", "HIT ENTER for no change", WHORLS, 4000, 4000)
If Len(Lt) = 0 Then Exit Sub
CH = 1 ' bypass default value of WHORLS
WHORLS = Lt
Command1.SetFocus
Label1.Visible = True
End Sub

Private Sub Command9_Click()
'----------Number of Growth Increments----------
Lt = InputBox("Number of growth increments", "HIT ENTER for no change", NG, 4000, 4000)
If Len(Lt) = 0 Then Exit Sub
CH = 1 ' bypass default value of increments
NG = Lt
Command1.SetFocus
Label1.Visible = True
End Sub

Private Sub convert()
'----------Convert Subroutine----------
' Given x, y, BWANG, and angle in radians (jr), returns plotting coordinates: xp & zp
' plus yp for evaluating distance  between branches (and for rotation and tilt)
xh = RAD * Cos(jr)'  Point on helix at this rotation angle (jr)
yh = -RAD * Sin(jr)
zh = -ELEV * 360 / 5 * jr / (2 * pi)' Elevation of point on helix
d = Sqr((y - yh) ^ 2 + (x - xh) ^ 2)' Distance from x, y to point on helix
dp = d * Cos(pi * (90 - BWANG) / 180)' Distance from x, y to helix after elevation
zp = zh - d * Sin(pi * (90 - BWANG) / 180)' z-value for plotting
xp = xh + dp * Cos(jr)' x-value for plotting
yp = yh - dp * Sin(jr)' y-value for computing 'dist' and plotting (IF NEEDED)
' Rotation around y-axis (Yr) and/or z-axis (Zr):
Yr = YrD      ' Negative for clockwise
Zr = ZrD      ' Negative for clockwise
Yr = Yr - 90: Yr = pi * Yr / 180: Zr = Zr + 180: Zr = pi * Zr / 180
yyy = xp * Sin(Zr) + yp * Cos(Zr)
```

```
xxx = xp * Cos(Zr) * Sin(Yr) - yp * Sin(Zr) * Sin(Yr) + zp * Cos(Yr)
zzz = xp * Cos(Zr) * Cos(Yr) - yp * Sin(Zr) * Cos(Yr) - zp * Sin(Yr)
xp = xxx: yp = yyy: zp = zzz
If scaled = 0 Then Exit Sub
' Scale xp, zp for plotting:
If Abs(Hmax - Hmin) > Abs(Vmax - Vmin) Then _
   Pscale = 7000 / (Hmax - Hmin) Else Pscale = 7000 / (Vmax - Vmin)
xp = 4000 + (xp - Hmin) * Pscale
zp = 8000 - (Vmax - zp) * Pscale
End Sub

Private Sub Form_Load()
WindowState = 2
Command2.Visible = False
Command3.Visible = False
Command4.Visible = False
Command5.Visible = False
Command6.Visible = False
Command7.Visible = False
Command8.Visible = False
Command9.Visible = False
Command10.Visible = False
Command11.Visible = False
Command12.Visible = False
Command13.Visible = False
Label1.Visible = False
End Sub

Private Sub Rng()
'----------Random Number Subroutine----------
' Random number generator (crude but works).
' Yields normally distributed values ('g') with mean of zero and
'   standard deviation = 'sigma'.
r3 = 2
Do Until r3 < 1
   r1 = 2 * Rnd - 1
   r2 = 2 * Rnd - 1
   r3 = r1 * r1 + r2 * r2
Loop
r3 = Sqr((-2 * Log(r3)) / r3)
g = sigma * (r1 * r3)
End Sub
```