Solving Differential Equations in Developmental Models of Multicellular Structures Expressed Using L−systems

Pavol Federl and Przemyslaw Prusinkiewicz
Department of Computer Science, University of Calgary
Calgary, Alberta, Canada T2N 1N4
e−mail: federl|pwp@cpsc.ucalgary.ca

Abstract

Mathematical modeling of growing multicellular structures creates the problem of solving systems of equations in which not only the values of variables, but the equations themselves, may change over time. We consider this problem in the framework of Lindenmayer systems, a standard formalism for modeling plants, and show how parametric context−sensitive L−systems can be used to numerically solve growing systems of coupled differential equations. We illustrate our technique with a developmental model of the multicellular bacterium *Anabaena*.

Reference

P. Federl and P. Prusinkiewicz: Solving differential equations in developmental models of multicellular structures expressed using L−systems. In M. Bubak, G. van Albada, P. Sloot and J. Dongarra (Eds.): *Proceedings of Computational Science. ICCS 2004* (Krakow, Poland, June 6−9, 2004), Part II, *Lecture Notes in Computer Science 3037*, Springer, Berlin, pp. 65−72.

Solving Differential Equations in Developmental Models of Multicellular Structures Expressed Using L-systems

Pavol Federl and Przemyslaw Prusinkiewicz

University of Calgary, Alberta, Canada

Abstract. Mathematical modeling of growing multicellular structures creates the problem of solving systems of equations in which not only the values of variables, but the equations themselves, may change over time. We consider this problem in the framework of Lindenmayer systems, a standard formalism for modeling plants, and show how parametric context-sensitive L-systems can be used to numerically solve growing systems of coupled differential equations. We illustrate our technique with a developmental model of the multicellular bacterium *Anabaena*.

1 Introduction

Recent advances in genetics have sparked substantial interest in the modeling of multicellular organisms and their development. Modeling information transfer through cell membranes is a vital aspect of these models. Diffusion of chemicals is one example of a transfer mechanism, and can be mathematically expressed as a system of ordinary differential equations (ODEs). Due to the developmental nature of the models, this system changes as the cells in the organism divide. Such *dynamically evolving systems of equations* are not easily captured by standard mathematical techniques [2].

The formalism of L-systems [6] lends itself well to modeling developmental processes in organisms. Prusinkiewicz et al. introduced *differential L-systems* (dL-systems) [10] as a notation for expressing developmental models that include growing systems of ODEs, but left open the problem of solving these equations. From the viewpoint of software organization these equations can be solved either using an external solver or within the L-system formalism itself. The first technique induces substantial overhead due to repetitive transfers of large amounts of data to and from the solver in each simulation step. As an alternative, we present a mechanism where the system of ODEs is internally maintained, updated, and solved by an L-system. We adapt to this end an implicit (Crank-Nicholson) integration scheme, whereas previous approaches only used simpler, explicit methods [2, 7]. We illustrate our solution by revisiting the diffusion-based developmental model of the blue-green alga *Anabaena catenula* [1, 8, 11], defined using a dL-system [10].

2 L-systems and the L+C Language

In the formalism of L-systems [6], growing biological structures are represented as strings of *modules* [11]. The initial structure is the *axiom*. An L-system describes the development of this structure in terms of *rewriting rules* or *productions*. Each production replaces its the *predecessor* module by zero, one, or more *successor* modules. For example, the production $A \rightarrow AB$ replaces module A by a structure consisting of a new module A and a new module B. In general, productions can be *context free*, with productions matched only to the predecessor module, or *context-sensitive*, with productions matched also to the predecessor's neighbors. The context-sensitive productions make it possible to simulate information transfer within developing structures.

The algorithms presented in this paper are specified in the L+C programming language [4, 5], which combines the declarative programming style of L-systems with C++ constructs. The L+C modules are declared by the keyword `module`, e.g. `module B(int, double)`. The initial string is preceded by the keyword `axiom`, e.g. `axiom: B(1,7.0)`. The body of a production, delimited by curly braces, may include any valid C++ statement. An example of a context-sensitive production is:

```
A(n) < B(i,j) > C() : { if (i < j) produce D(i+n,j-n); }
```

The body of this production is executed for every module `B` that has a module `A` on its left and `C` on its right side. If the parameter `i` of module `B` is less than its parameter `j`, the module `B` will be replaced by a module `D` with updated parameters. This is denoted by the keyword `produce` inside the `if` statement.

Although L-systems have been defined as a *parallel* rewriting mechanism, they are commonly implemented by *sequentially* scanning the predecessor string to obtain the successor string. In L+C we take advantage of this fact. The scanning direction is chosen at each derivation step by calling functions `Forward()` or `Backward()`. As the successor string is being generated, the *newly created* modules in the string can be used for context matching, using the symbols '`<<`' for the *new left context* and '`>>`' for the *new right context*. For example, consider a string of modules `A` with integer parameters `A(1)A(2)A(3)A(4)A(5)` and a production

```
A(nc) << A(n) : produce A(nc+n);
```

This string will be transformed into `A(1)A(3)A(6)A(10)A(15)` in a single derivation step.

Productions in L+C can be arranged into *groups* using the construct `group <n>: <productions>`, where `n` is an integer identifier. The active group of productions can be selected dynamically by calling the function `UseGroup(n)`. The L+C constructs `Start` and `StartEach` are used to denote blocks of C++ code that are executed at the beginning of the simulation and at the beginning of each derivation step. Both constructs are useful when the derivation of the string is divided into separate phases.

3 The *Anabaena* Model

Anabaena is a bacterium that forms non-branching multicellular filaments consisting of two classes of cells: *vegetative cells* and *heterocysts* [8]. A vegetative cell may divide into two vegetative cells, or differentiate into a heterocyst. In spite of the cell division, the spacing between heterocysts is relatively constant, with approximately 10 vegetative cells separating the heterocysts on the average Mathematical models of this phenomenon [1, 7, 11] postulate that the distribution of heterocysts is regulated by a compound produced by the heterocysts, diffusing from cell to cell along the filament, and decaying in the vegetative cells. If the compound concentration in a vegetative cell falls below a threshold level, this cell differentiates into a heterocyst.

A model operating in continuous time according to this description can be captured by the following dL-system [10]:

axiom: $F_h(s_{max}, c_{max})F_v(s_{max}, c_{max})F_h(s_{max}, c_{max})$
$F(s_l, c_l) < F_v(s, c) > F(s_r, c_r)$:
    if $s < s_{max}$ & $c > c_{min}$
        solve $dc/dt = D \cdot (c_l + c_r - 2c) - \mu c$ and $ds/dt = rs$
    if $s = s_{max}$ & $c > c_{min}$
        produce $F_v(ks_{max}, c)F_v((1-k)s_{max}, c)$
    if $c = c_{min}$
        produce $F_h(s, c)$
$F_h(s, c)$:
        solve $ds/dt = r_s(s_{max} - s)$ and $dc/dt = r_c(c_{max} - c)$

According to this model, a vegetative cell F_v or heterocyst F_h is characterized by its length s and compound concentration c. If the vegetative cell length s is below the maximum value s_{max} and the compound concentration c is above the threshold c_{min}, the concentration changes according to the equation $dc/dt = D \cdot (c_l - 2c + c_r) - \mu c$. This equation combines diffusion of the compound c according to Fick's law with the decay of c. The second differential equation characterizing F_v describes exponential elongation of cells according to the equation: $ds/dt = rs$.

In addition to these differential equations, two productions describe the behavior of a vegetative cell. If the cell reaches maximum length s_{max} while the concentration c is still above the threshold c_{min}, the cell will asymmetrically divide into two vegetative cells of length ks_{max} and $(1-k)s_{max}$, with the compound concentration c inherited from their parent cell. Otherwise, if the concentration c drops down to the threshold c_{min}, the cell will differentiate into a heterocyst. The last line of the dL-system specifies the behavior of heterocysts. Their length and compound concentration converge exponentially to the limit values of s_{max} and c_{max}, according to the equations: $dc/dt = r_c(c_{max} - c)$ and $ds/dt = r_s(s_{max} - s)$. The heterocysts do not undergo any further division.

4 Solving the dL-systems

A dL-system can be viewed as a formal statement of a problem that requires a solution. The simplest technique for numerically solving the ODEs from the previous section is the forward Euler method, where time derivatives are replaced with a forward finite difference approximation. This approach to solving differential equations in the framework of L-systems was presented in [3]; for the *Anabaena* dL-system it leads to the following system of equations:

$$(C^{i+1} - C^i)/\Delta t = D \cdot \left(C_l^i - 2C^i + C_r^i\right) - \mu C^i \ , \tag{1}$$

$$(S^{i+1} - S^i)/\Delta t = rS^i \ , \tag{2}$$

$$(C^{i+1} - C^i)/\Delta t = r_c(c_{max} - C^i) \ , \tag{3}$$

$$(S^{i+1} - S^i)/\Delta t = r_s(s_{max} - S^i) \ . \tag{4}$$

Here S^i and C^i denote the approximations of the cell length s and compound concentration c at time step i; time intervals have size Δt. Equations (1)-(4) define explicitly how the new values S^{i+1} and C^{i+1} are calculated using the previous values S^i and C^i. Subscripts l and r refer to the left and right neighbors of a given cell.

Although explicit integration is simple and can be easily expressed in L+C as a part of developmental models, it is inaccurate and prone to numerical instabilities. This is especially true for stiff differential equations, such as Equation (1) with a high diffusion constant D. Implicit schemes are much more appropriate for stiff equations. Discretizing the same ODEs using the implicit Crank-Nicholson scheme [9] yields:

$$(C^{i+1} - C^i)/\Delta t = D\frac{C_l^i - 2C^i + C_r^i + C_l^{i+1} - 2C^{i+1} + C_r^{i+1}}{2} - \mu\frac{C^i + C^{i+1}}{2} \ , \tag{5}$$

$$(S^{i+1} - S^i)/\Delta t = r(S^i + S^{i+1})/2 \ , \tag{6}$$

$$(C^{i+1} - C^i)/\Delta t = r_c\left(c_{max} - (C^i + C^{i+1})/2\right) \ , \tag{7}$$

$$(S^{i+1} - S^i)/\Delta t = r_s\left(s_{max} - (S^i + S^{i+1})/2\right) \ . \tag{8}$$

The relationships between the new and old values are now defined implicitly, by a system of linear equations. The coefficient matrix representing Equations (5) is tridiagonal, which leads to an efficient solution using a modified Gaussian elimination. Below we show that this elimination can be efficiently implemented using L-systems and L+C.

4.1 Solving Tridiagonal Systems of Linear Equations With L-systems

Let us consider a system of n linear equations with n unknowns described by a tridiagonal coefficient matrix. Such a system can be expressed as:

$$a_i X_{i-1} + b_i X_i + c_i X_{i+1} = y_i \ \text{ for } \ 1 \le i \le n \ , \tag{9}$$

where a_i, b_i, c_i and y_i are the coefficients of the equations, and X_i's denote the unknowns. This system of LEs can also be written in matrix form as $AX = Y$, where $X = [X_1, \cdots, X_n]^T$ is the column vector representing the unknowns, $Y = [y_1, \cdots, y_n]^T$ represents the constant terms, and A is the coefficient matrix:

$$A = \begin{pmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & a_n & b_n \end{pmatrix} \tag{10}$$

We solve this system of LEs in linear time using Gaussian elimination optimized to perform row subtractions only between non-zero coefficients. This process consists of two phases [9]. In the first phase, the coefficients below the diagonal are eliminated. Since the only coefficients of A below the diagonal are the coefficients a_i, this phase corresponds to calculating new values for coefficients a_i, b_i, c_i and y_i such that $a_i = 0$. To this end, the coefficients of each row are recalculated using the following substitutions:

$$a'_i \leftarrow 0 \ , \ b'_i \leftarrow b_i - \frac{a_i}{b'_{i-1}} c'_{i-1} \ , \ c'_i \leftarrow c_i \text{ and } y'_i \leftarrow y_i - \frac{a_i}{b'_{i-1}} y'_{i-1} \ . \tag{11}$$

These substitutions are applied iteratively, for i increasing from 2 to n. In the second phase the coefficients above the diagonal c_i are eliminated. This is accomplished by applying the following substitutions:

$$a'_i \leftarrow 0 \ , \ b'_i \leftarrow b_i \ , \ c'_i \leftarrow 0 \text{ and } y'_i \leftarrow y_i - \frac{c_i}{b'_{i+1}} y'_{i+1} \ . \tag{12}$$

These substitutions are applied in reverse order, for i decreasing from $n-1$ to 1. At the end of the second phase the coefficient matrix has non-zero entries strictly on the diagonal and the solution can be trivially found as $X_i = y_i / b_i$.

To implement this two-phase process using an L-system, we represent the system of tridiagonal LEs by a string of modules `C`. Each module `C` has a single parameter `p` of type `struct CD` that holds the non-zero entries of one row:

```
struct CD { double a, b, c, y; };
module C (struct CD);
```

The following L+C productions implement the two phases specified by Equations (11) and (12):

```
group 1: C(pl) << C(p) : { p.y=p.y-pl.y*p.a/pl.b;
           p.b=p.b-pl.c*p.a/pl.b; produce C(p); }
group 2: C(p) >> C(pr) : { p.y=p.y-pr.y*p.c/pr.b;
           produce C(p); }
```

Each phase is performed in a single sequential derivation step. We use a global variable `phase` to set the direction of processing at the beginning of each derivation step and to decide which group of productions will be applied during that phase:

```
int phase;
Start: { phase=1; }
StartEach: {
    if (phase == 1) { Forward (); UseGroup (1); phase=2; }
    else { Backward (); UseGroup (2); phase=1; } }
```

4.2 Implementation

The complete developmental model of *Anabaena* based on Equations (5)-(8) is obtained by complementing the solution to the diffusion Equation (5), presented above, with the solution to Equations (6)-(8). To solve Equation (5) we transform it to the form of Equation (9) using the substitutions:

$$a_0 = c_0 = a_n = c_n = 0 \ , \ b_0 = b_n = 1 \ , \tag{13}$$

$$y_0 = y_n = \frac{2C^i + 2s_{\max} r_s \Delta t - C^i \Delta t r_s}{2 + r_s \Delta t} \ , \tag{14}$$

$$a_i = c_i = -D \ , \ b_i = \frac{1}{\Delta t} + 2D + \mu \ , \tag{15}$$

$$y_i = \frac{2C^i}{\Delta t} + D \cdot \left(C_l^i - 2C^i + C_r^i\right) - \mu C^i \ . \tag{16}$$

for $1 < i < n$. The resulting system of LEs is solved using the method outlined in Sec. 4.1. Since no information transfer is involved in Equations (6)-(8), they can be expressed in closed form as:

$$S_{veg}^{i+1} = \frac{2 + r\Delta t}{2 - r\Delta t} S^i \ , \tag{17}$$

$$C_{het}^{i+1} = \frac{2C^i + 2c_{max} r_c \Delta t - C^i r_c \Delta t}{2 + r_c \Delta t} \ , \tag{18}$$

$$S_{het}^{i+1} = \frac{2S^i + 2s_{max} r_s \Delta t - S^i r_s \Delta t}{2 + r_s \Delta t} \ . \tag{19}$$

To implement the solution based on these formulas using an L-system, we represent the growing filament again as a string of modules `C`. The parameter of the module `C` is of type `struct CD`, declared as:

```
struct CD { double s, con, a, b, c, y; bool h; };
```

The fields `s` and `con` denote the size of the cell and the concentration of the diffusing compound, respectively. The fields `a`, `b`, `c` and `y` represent the coefficients of the system of LEs described in Sec. 4.1. We use five logical phases to update the state of this filament in a time step Δt. In the first phase the new cell sizes are calculated using (17), and the system of LEs is initialized according to Equations (13)-(16). This phase is implemented by two productions:

```
group 1: C(pl) < C(p) > C(pr): {
    if (p.h) {
        p.y=(2*p.con+2*cmax*dt*rc-p.con*dt*rc)/(2+dt*rc);
        p.s=(2*p.s+2*smax*dt*rs-p.s*dt*rs)/(2+dt*rs);
        p.a=0; p.b=1; p.con=0; produce C(p); }
    else {
        p.a=-D; p.b=2/dt+2*D+mu; p.c=-D;
        p.y=2*p.con/dt+D*(cl.con-2*p.con+cr.con) - mu*p.con;
        p.s=(2+r*dt)/(2-r*dt)*p.s; produce C(p); } }
  C(p) : { p.a=p.c=0; p.b=1; p.y=p.con; produce C(p); }
```

The first production is applied to every module except the first and last. The second production is applied to the first and last modules. The LEs are then solved in the next two phases, using productions identical to those given in the previous section. In the fourth phase the solution is extracted and the fifth phase implements division and differentiation of cells:

```
group 4: C(p): { p.con=p.y/p.b; produce C(p); }
group 5: C(pl) < C(p) > C(pr) : {
    if (p.h) { produce C(p); }
    else if (p.s >= smax && p.con > cmin) {
        CD p1=p; p1.s=k*p.s; CD p2=p; p2.s=(1-k)*p.s;
        produce C(p1) C(p2); }
    else if (p.con < cmin) { p.h=true; produce C(p); }
}
```

The switching between the five phases is accomplished by extending the method discussed in Sec. 4.1. The initial steps of a simulation obtained using the above L-system are shown in Fig.1.

5 Conclusions

The modeling of developing organisms raises the problem of formulating mathematical models in which the set of variables that describes the system, and the system of equations that relate these variables, dynamically change [2]. We presented a method for automatically modifying and solving these equations using the formalism of L-systems. The underlying numerical integration method is the Crank-Nicholson method [9]. We illustrated our approach using a model of a filamentous bacterium *Anabaena*. This organism captures the essential elements of the development of a multicellular structure: division and differentiation of cells, and signaling between cells. Although we have only considered a very simple model for illustrative purposes, the use of the implicit method for solving the ODEs is essential to extensions of this model. For example, the process of gradual cell division, during which the diffusion constants between incompletely divided cells are large, can easily be incorporated into the model. Other potential applications of the proposed method include biomechanical and functional-structural models of plant architecture.

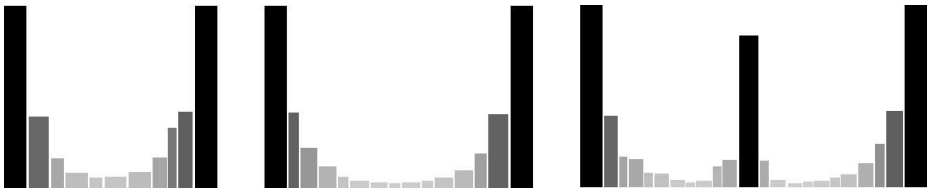

Fig. 1. Simulation results obtained using the implementation based on the Crank-Nicholson integration scheme. The following simulation constants were used: $s_{max} = 1$, $c_{max} = 255$, $c_{min} = 5$, $D = \mu = 0.03$, $r = 0.01$, $k = 0.37$, $r_s = 0.1$, $r_c = 0.15$, $\Delta t = 28.57$. Each cell is represented by a rectangle whose width and height correspond to the cell's size and concentration of the compound, respectively. The tallest dark bars represent the heterocysts, the lighter shaded bars represent vegetative cells.

6 Acknowledgments

We thank Brendan Lane, Lynn Mercer and Colin Smith for editorial help. The support from the Human Frontier Science Program and the National Sciences and Engineering Research Council of Canada is gratefully acknowledged.

References

1. C. G. de Koster and A. Lindenmayer. Discrete and continuous models for heterocyst differentiation in growing filaments of blue-green bacteria. *Acta Biotheoretica*, 36:249–273, 1987.
2. J.-L. Giavitto and O. Michel. Modeling the topological organization of cellular processes. BioSystems 2003, 70:149-163.
3. M. Hammel and P. Prusinkiewicz. Visualization of developmental processes by extrusion in space-time. *Proceedings of Graphics Interface '96*, pp 246–258, 1996.
4. R. Karwowski. *Improving the process of plant modeling: The L+C modeling language*. PhD thesis, University of Calgary, 2002.
5. R. Karwowski and P. Prusinkiewicz. Design and implementation of the L+C modeling language. *Electronic Notes in Theoretical Computer Science*, 86.2, 2003.
6. A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18:280–315, 1968.
7. A. Lindenmayer. Adding continuous components to L-systems. In G. Rozenberg and A. Salomaa, editors, *L Systems*, Lecture Notes in Computer Science 15, pages 53–68. Springer-Verlag, Berlin, 1974.
8. G. J. Mitchison and M. Wilcox. Rules governing cell division in *Anabaena*. *Nature*, 239:110–111, 1972.
9. W. H. Press, S. A. Teukolsky, and W. T. Wetterling. *Numerical recipes in C: The art of scientific computing. Second edition*. Cambridge University Press, 1988.
10. P. Prusinkiewicz, M. Hammel, and E. Mjolsness. Animation of plant development. Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993). ACM SIGGRAPH, New York, 1993, pp. 351–360.
11. P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag, New York, 1990. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer.