MULTI-SCALE CELL-BASED COMPUTATIONAL MODELS OF VERTABRATE SEGMENTATION AND SOMITOGENESIS ILLUMINATE COORDINATION OF DEVELOPMENTAL MECHANISMS ACROSS SCALES

Susan D Hester

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the Department of Biophysics

Indiana University

April 2012

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

James Glazier, Ph.D.

David Baxter, Ph.D.

John Beggs, Ph.D.

Joseph Pomerening, Ph.D.

Sima Setayeshgar, Ph.D.

Date of Oral Examination
September 9, 2011

To my parents, Jeff and Vicki Hester, and my sisters, Janice and Patricia, for the hours they have spent and the love they have extended supporting me over the last five years; and to Jason, for providing me much-needed perspective at every turn.

ACKNOWLEDGEMENTS

I thank my advisor, Dr. James A. Glazier, for his support, guidance and empathy. I will be eternally grateful to him for extending his expertise in the field of computational modeling, his enthusiasm for multidisciplinary research and collaboration, and his deep commitment as a mentor.

I would also like to thank the members of my committee, Dr. David Baxter, Dr. John Beggs, Dr. Sima Setayeshgar and Dr. Joe Pomerening, for extending their support and sharing their unique perspectives on this work.

I am indebted to those who collaborated with me and supported much of this work: Julio Belmonte, my modeling partner in crime; Dr. Scott Gens, who found many a thought-provoking article and who was always willing to have a conversation with a graduate student whose mental wheels had gotten stuck; and Dr. Sherry Clendenon, who provided both beautiful original experimental images and the intuition of an expert anatomist.

I extend special thanks to Dr. Maciek Swat, whose ongoing development of CompuCell3D and willingness to foster new users' development has made so much possible for many budding modelers, myself included. I would also like to acknowledge the other members of the CompuCell3D team during my time in the Glazier group, Benjamin Zaitlen, Randy Heiland and Dr. Mitja Hmeljak.

I owe thanks to Dr. Charles Little and Dr. Brenda Rongish for inviting me into their laboratory at the University of Kansas Medical Center to learn some finer points of chick embryo culture and manipulation, and Mike Filla and Tracey Cheuvront of KUMC for their hands-on instruction. I would also like to acknowledge Mike Hosek from Indiana University for first introducing me to experimental techniques with chick embryos.

I thank those who shared their work and insight on development and segmentation mechanics: Dr. Santiago Schnell, Dr. Philip Maini, Dr. Claudio Stern, Dr. Julian Lewis, Dr. Kees Weijer, Dr. Olivier Pourquié, Dr. Ertugrul Özbudak, and Dr. Bertrand Bénazéraf.

Finally, I extend my thanks to the agencies whose financial support made this research possible: the United States National Institutes of Health (under grants RM01 GM077138 and RM01 GM076692), the United States Environmental Protection Agency (under grants RD83500101 and R834289), the Indiana University College of Arts and Sciences Office of the Vice President of Research, the Indiana University Biocomplexity Institute and the Lilly Foundation for their support of my Biocomplexity Graduate Student Fellowship.

Sections of this thesis have been published in modified form in Hester, S. D., J. M. Belmonte, *et al.* (2011). "A multi-cell, multi-scale model of vertebrate segmentation and somite formation." PLoS Comput Biol 7(10): e1002155.

Susan D. Hester

MULTI-SCALE CELL-BASED COMPUTATIONAL MODELS OF VERTABRATE SEGMENTATION AND SOMITOGENESIS ILLUMINATE COORDINATION OF DEVELOPMENTAL MECHANISMS ACROSS SCALES

Somitogenesis, the formation of the body's primary segmental structure common to all vertebrate development, requires coordination between biological mechanisms at several scales. Explaining how these mechanisms interact across scales and how the events they generate are coordinated in space and time is necessary for a complete understanding of somitogenesis and its evolutionary flexibility. So far, mechanisms of somitogenesis have been studied independently. To test the consistency, integrability and combined explanatory power of current prevailing hypotheses, we built an integrated clock-and-wavefront model including submodels of the intracellular segmentation clock, intercellular segmentation-clock coupling via Delta/Notch signaling, an FGF8 determination front, delayed differentiation, clock-wavefront readout, and differential-cell-cell-adhesion-driven cell sorting. We identified inconsistencies between existing submodels and gaps in the current understanding of somitogenesis mechanisms, and proposed novel submodels and extensions of existing submodels where necessary. For reasonable initial conditions, two-dimensional simulations of our model robustly generate spatially and temporally regular somites, realistic dynamic morphologies and spontaneous emergence of anterior-traveling "pseudo waves" of Lfng. Our model is flexible enough to generate interspecies-like variation in somite size in response to changes in the PSM growth rate and segmentation-clock period, and in the number and width of Lfng stripes in response to changes in the PSM growth rate, segmentation-clock period and PSM length.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

Following is a list of the acronyms used in this work. The page(s) where the acronym first occurs and is defined in context is given in parentheses. A brief definition follows where appropriate.

2D (p. 5): Two-dimensional.

AP (p. 5): Antero-posterior. The major axis of the embryo that runs from the head (anterior) to the tail (posterior) of the embryo.

CC3D (p. 11): CompuCell3D. An open-source software package designed to simulate multi-cell Glazier-Graner-Hogeweg (GGH) (Glazier and Graner 1993) models of cell behaviors in conjunction with intracellular genetic-network or reaction-kinetic models and extracellular partial-differential-equation (PDE) models of tissue-level morphogen concentrations (Alber, Chen et al. 2006; Popławski, Shirinifard et al. 2008; Shirinifard, Gens et al. 2009). For information on running CC3D simulations, see (Swat, Hester et al. 2009).

DV (p. 5): Dorso-ventral. The axis of the embryo that runs from the top/"back" of the embryo (the dorsal side) to the bottom/"belly" of the embryo (the ventral side).

ECM (p. 14): Extracellular matrix. Material that is secreted by cells into the surrounding environment. The ECM can act as a scaffold across which cells can move; it can also act to restrict cell movement by mechanically impeding cells.

FGF8 (p. 16): Fibroblast growth factor 8. See KEY MOLECULAR PLAYERS.

GGH (p. 11, p. 40): Glazier-Graner-Hogeweg. The GGH model of cell dynamics represents space as a regular lattice. **Cell** dynamics in the GGH model provide a much simplified representation of cytoskeletally-driven cell motility using a stochastic modified Metropolis algorithm consisting of a series of index-copy attempts.

ML (p. 5): Medio-lateral. The minor axis of the embryo that runs from the long midline of the embryo (the medial line) to the edge of the embryo (the lateral line).

mRNA (p. 11): Messenger ribonucleic acid. This type of molecule is responsible for carrying a "transcript" of the information needed to manufacture a protein from the original DNA (deoxyribonucleic acid) molecule to the cellular protein-building machinery.

PSM (p. 3, p. 13): Presomitic mesoderm. The tissue from which the somites form. The PSM lies in two strips of mesenchymal cells along either side of the notochord, the central structure running along the AP axis of the embryo.

RA (p. 16): Retinoic acid. See KEY MOLECULAR PLAYERS.

KEY MOLECULAR PLAYERS

Following is a short list of key molecular players in somitogenesis. The page on which the molecule is introduced and described is given, along with a short description.

Axin2 (p. 19): Axin2 is regulated downstream of Wnt3a signaling. Axin2 displays periodic patterns of expression in the presomitic mesoderm in both space and time. Medio-lateral stripes of Axin2 mRNA seem to move anteriorly down the antero-posterior axis of the chick embryo during somitogenesis stages. Axin2 expression oscillations are approximately half a cycle out of phase with Lfng oscillations and Dusp6 oscillations.

Delta (p. 17): Delta is the membrane-bound ligand of the Notch receptor. Delta/Notch signaling plays an important role in synchronizing segmentation-clock oscillations in neighboring cells, as contact between the membrane-tethered ligand, Delta, and its receptor, Notch, is required to stimulate Delta/Notch signaling. Delta/Notch signaling plays a role in regulating the segmentation clock, and drives one of the loops in the segmentation-clock submodel.

EphA4 (p. 25): A membrane-bound receptor found in the anterior PSM and in the anterior of the somites. When an EphA4-expressing cell encounters an ephrinB2-expressing cell, there is effective repulsion between the two cells.

ephrinB2 (p. 25): A membrane-bound receptor found in the posterior of the somites. When an ephrinB2-expressing cell encounters an EphA4-expressing cell, there is effective repulsion between the two cells.

FGF8 (p. 16): Fibroblast growth factor 8. FGF8 is a cell-cell signaling molecule that diffuses a short distance away from the cell that secretes it and signals through a cell-surface receptor. FGF8 is implicated in the determination and differentiation fronts: FGF8 signaling keeps PSM cells in an immature state, and is expressed strongly in the posterior and in an anteriorly-decreasing gradient in the PSM. FGF8 signaling plays a role in regulating the segmentation clock, and drives one of the three loops in the segmentation-clock network submodel.

Lfng (p. 19): Lunatic fringe. Lfng is regulated downstream of Delta/Notch signaling. Lfng displays periodic patterns of expression in the presomitic mesoderm in both space and time. Medio-lateral stripes of Lfng mRNA seem to move anteriorly down the antero-posterior axis of the chick embryo during somitogenesis stages. Lfng expression oscillations are approximately in phase with Dusp6 oscillations and half a cycle out of phase with Axin2 oscillations.

N-cadherin (p. 25): A homophilic, membrane-bound adhesion protein expressed in the centers of the somites. Two cells expressing N-cadherin display strong adhesion towards one another. N-cadherin disruption leads to fragmentation of somites and separation of the anterior and posterior somite compartments (Horikawa, Radice *et al.* 1999).

N-CAM (p. 25): A homophilic, membrane-bound adhesion protein expressed in the anterior PSM and somites. N-CAM results in weaker adhesion between cells than N-cadherin.

Notch (p. 17): Notch signaling plays a role in regulating the segmentation clock, and drives one of the loops in the segmentation-clock submodel. Delta/Notch signaling plays an important role in synchronizing segmentation-clock oscillations in neighboring cells, as contact between the membrane-tethered ligand, Delta, and its receptor, Notch, is required to stimulate Delta/Notch signaling.

RA (p. 16): Retinoic acid. RA is a cell-cell signaling molecule that can diffuse long distances. RA acts as a maturation signal and antagonizes FGF8 signaling. RA is implicated in positioning the differentiation front for these reasons and because it is found in posteriorly-decreasing gradient in the anterior PSM.

Wnt3a (p. 16): Wnt3a is a cell-cell signaling molecule that diffuses a short distance away from the cell that secretes it and signals through a cell-surface receptor. Wnt3a is expressed strongly in the posterior and in an anteriorly-decreasing gradient in the PSM. Wnt3a signaling plays a role in regulating the segmentation clock, and drives one of the three loops in the segmentation-clock network submodel.

Chapter 1

INTRODUCTION

1.1 Multi-scale models of development

In biology, the term “model” traditionally refers to an animal or cell line used in a set of experiments. In this dissertation, I use the term differently, to refer to abstract representations of biological phenomena. I will distinguish types of models by their *level*, referring to their degree of abstraction (*not*, it should be noted, by the length scale with which they are concerned). In this lexicon, a *biological model* is a qualitative (possibly complex) description of a set of hypothesized mechanisms and relationships that seek to explain a biological phenomenon; a *mathematical model* is a formalized, quantifiable representation of a biological model, in which the rules (often sets of equations) governing the biological behavior are stated explicitly in a quantitative form; a *computational model* is an implementation of a mathematical model in the form of algorithms, which use particular methods and sets of (possibly method-dependent) parameters, initial conditions and boundary conditions; a *simulation* is an instance of a computational model expressed as executable code with specific parameter values (we employ families of simulations to evaluate a model’s response to changes in boundary conditions or initial conditions); and a *visualization* is a set of images presenting a selection of the data produced by a simulation. At each abstraction level and length scale, every model requires at least the following: *objects* (what physical components are being described), *behaviors* (the intrinsic properties of objects), *interactions* (how objects and their behaviors affect each other), *dynamics* (how objects, behaviors and interactions change in time) and *initial conditions* (the initial identity, configuration and state of all objects, behaviors and interactions) (**Figure 1.1**). Finally, at each level of abstraction we must explicitly state the simplifications and assumptions that we have made in addition to those already included in the previous levels of abstraction.

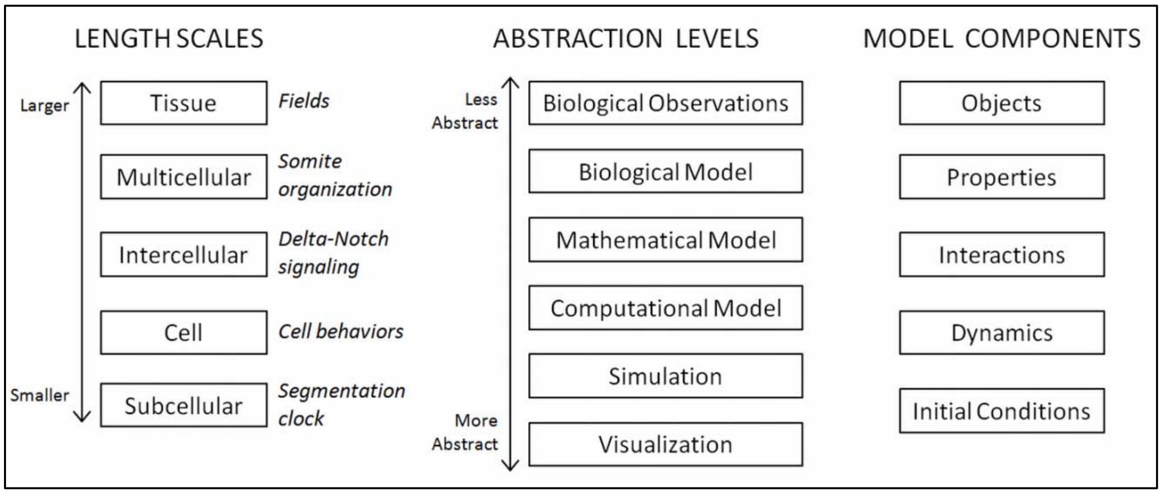

Figure 1.1: Multi-scale modeling over different length scales and abstraction levels.

Unless qualified, when I refer to a *model* I am referring to an ensemble of corresponding biological, mathematical and computational models and simulations (*i.e.*, to a particular set of hypotheses and their description at various levels of abstraction).

Models at any level of abstraction can be modular, combining two or more *submodels* (usually models addressing a single length scale or mechanism) into a multi-scale and/or multi-mechanism *composite model*.

1.1.1 Challenges of building multi-scale models

Building successful multi-scale composite models is challenging. We must first combine the biological submodels in a biologically consistent, plausible way, *i.e.*, we must infer biological connections between the mechanisms that the different submodels explain. We must then connect the corresponding mathematical submodels in a way that both reflects the biological connection and makes sense in the mathematical context of the submodels. Implementing the model computationally requires similar attention to consistency across submodels and between the composite model and the modeled biological phenomenon. Different submodels require different computational approaches to their solutions, especially when they address different spatial and/or temporal scales. In this case, interaction between submodels entails interaction between different computational methods. In essence, to connect two submodels requires an additional model of how the submodels interact, expressed at the biological, mathematical, computational and simulation abstraction levels.

1.2 Somitogenesis: a multi-scale problem

Somitogenesis, the developmental process during which the presomitic mesoderm (*PSM*) lying on either side of the central notochord divides into a series of roughly spherical epithelial *somites* (**Figure 1.2**), establishes the earliest evident segmentation in vertebrate embryos (Gossler and Hrabe de Angelis 1998). Somite formation is regular in both time and space, with a pair of somites (one on either side of the notochord) forming and separating from the anterior of the PSM approximately every 30 minutes in zebrafish, every 90 minutes in chick, and every 120 minutes in mouse. An intricate cellular dance characterizes somite formation, with cells at the interface between a forming somite and the anterior PSM rearranging and pulling apart to form two distinct tissues separated by an *intersomitic gap* (Kulesa and Fraser 2002).

The striking spatio-temporal periodicity and dynamic morphology of somitogenesis depend on mechanisms operating across a range of scales, as well as interactions between scales: genetic and protein oscillations and regulatory networks at the *subcellular* scale (Dequeant, Glynn et al. 2006); *juxtacrine* (contact-dependent) and *paracrine* (secretion-dependent) cell-cell signaling

(Lewis 2003; Wahl, Deng *et al.* 2007), and differential cell-cell adhesion at the *cellular* and *multicellular* scales (Duband, Dufour *et al.* 1987; Glazier, Zhang *et al.* 2008); and PSM-spanning gradients (Dubrulle, McGrew *et al.* 2001; Aulehla, Wehrle *et al.* 2003) and gene expression patterns (Palmeirim, Henrique *et al.* 1997) at the *tissue* scale.

Because somitogenesis involves interactions between many scales as well as coordination between events occurring in time and space, it is both uniquely interesting in its own right and a case study for the development of predictive and informative multi-scale models of development. Existing submodels addressing specific subcomponent mechanisms of somitogenesis have improved our understanding at individual scales and between scales, creating the impression that we are converging on a comprehensive understanding of somitogenesis. We have no assurance, however, that existing submodels are consistent and integrable with one another, or that, combined, they suffice to explain somitogenesis *in toto*.

Studying somitogenesis also provides insight into the evolvability of the vertebrate body plan. I will demonstrate that our integrated model of somitogenesis is robust and flexible enough that it can describe somitogenesis in animals as different in size, shape and gestation time as chickens, garden snakes, mice and zebrafish (Sections 4.3 and 4.4). Modeling how mechanisms interact in time and space and across scales clarifies how a single segmentation strategy is flexible enough to generate such variation.

In this dissertation, I focus on segmentation and somite formation in the chick embryo, and the features of somitogenesis that make it robust in the face of perturbations as well as flexible and evolvable enough to produce observed variations between species. Our model describes two-dimensional (*2D*) antero-posterior (*AP*) and medio-lateral (*ML*) effects of somitogenesis between roughly HH Stage 8 (4 somite pairs) and HH Stage 16 (26-28 somites pairs). During this period, the PSM is relatively flat and of constant length, allowing us to neglect dorso-ventral (*DV*) patterning. We did not model the initiation or termination of somitogenesis, the effects of cell division, or the formation of the specialized somites that form at the extreme anterior and posterior of the somitic tissues.

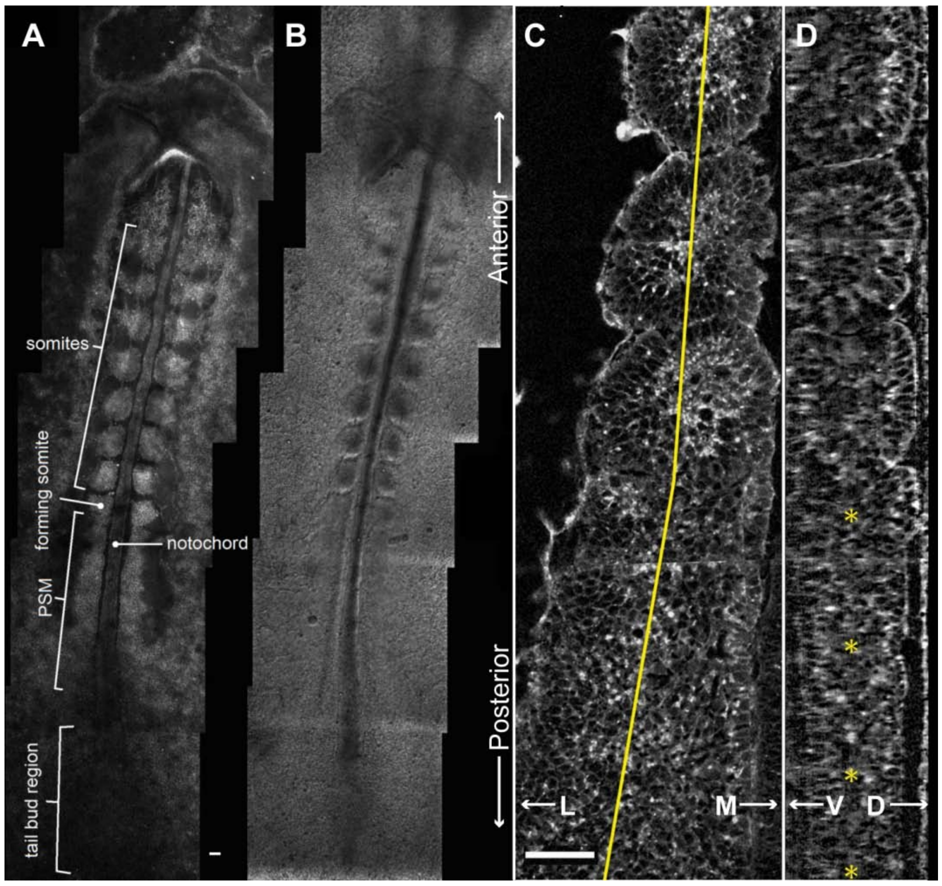

Figure 1.2: Chick embryo at HH Stage 10. (**A**) Image of a live HH Stage 10 chick embryo stained with Lens culinaris agglutinin-FITC. (**B**) DIC image of the same embryo, (**C**) Coronal (ML-AP) and (**D**) sagital (DV-AP) slices of a single strip of the PSM and the most recently formed somites of a chick embryo at HH Stage 10, stained with Lens culinaris agglutinin-FITC. The PSM is relatively flat at the posterior end, and gradually becomes thicker towards the anterior end. We measured PSM DV thickness at the PSM midline (yellow line in (**C**)). Yellow *s in (**D**) indicate points where the thickness was measured. Measured thickness, from posterior (bottom) to anterior (top): 61 μm, 67 μm, 73 μm and 95 μm. The thickness through the center of the forming somite is 98 μm. In all panels, the anterior (head) is at top, posterior (tailbud) at bottom. Scale bars 40μm. (Credit for experimental image: Dr. Sherry Clendenon, Indiana University).

1.2.1 Clock-and-wavefront model

The *clock-and-wavefront* model, initially proposed by Cooke and Zeeman in 1976, describes a smoothly varying intracellular oscillator (the *segmentation clock*) that interacts with a posterior-propagating front of cell maturation extending in the ML and DV directions in the PSM (the *wavefront*) to divide the PSM into periodic segments at regular spatio-temporal intervals (Cooke and Zeeman 1976). Experiments have since borne out the model's central predictions, identifying candidates for both the clock and wavefront components in the PSM. This validation has boosted the model's popularity and led to a family of clock-and-wavefront models at all abstraction levels (ranging from purely qualitative biological models to mathematical descriptions to computational implementations). Recently, Baker *et al.* have reviewed the various types of somitogenesis models including the clock-and-wavefront model (Baker, Schnell *et al.* 2006), and have implemented sophisticated 1D mathematical clock-and-wavefront models (Baker, Schnell *et al.* 2006; Baker, Schnell *et al.* 2007; Baker, Schnell *et al.* 2008). Clock-and-wavefront models differ in detail but adhere to the core idea of Cooke and Zeeman that an intracellular segmentation clock and a posteriorly advancing wavefront establish and coordinate the temporal and spatial periodicity of somitogenesis. **Figure 1.3** shows a schematic of the clock-and-wavefront model elements included in the composite model that I present here.

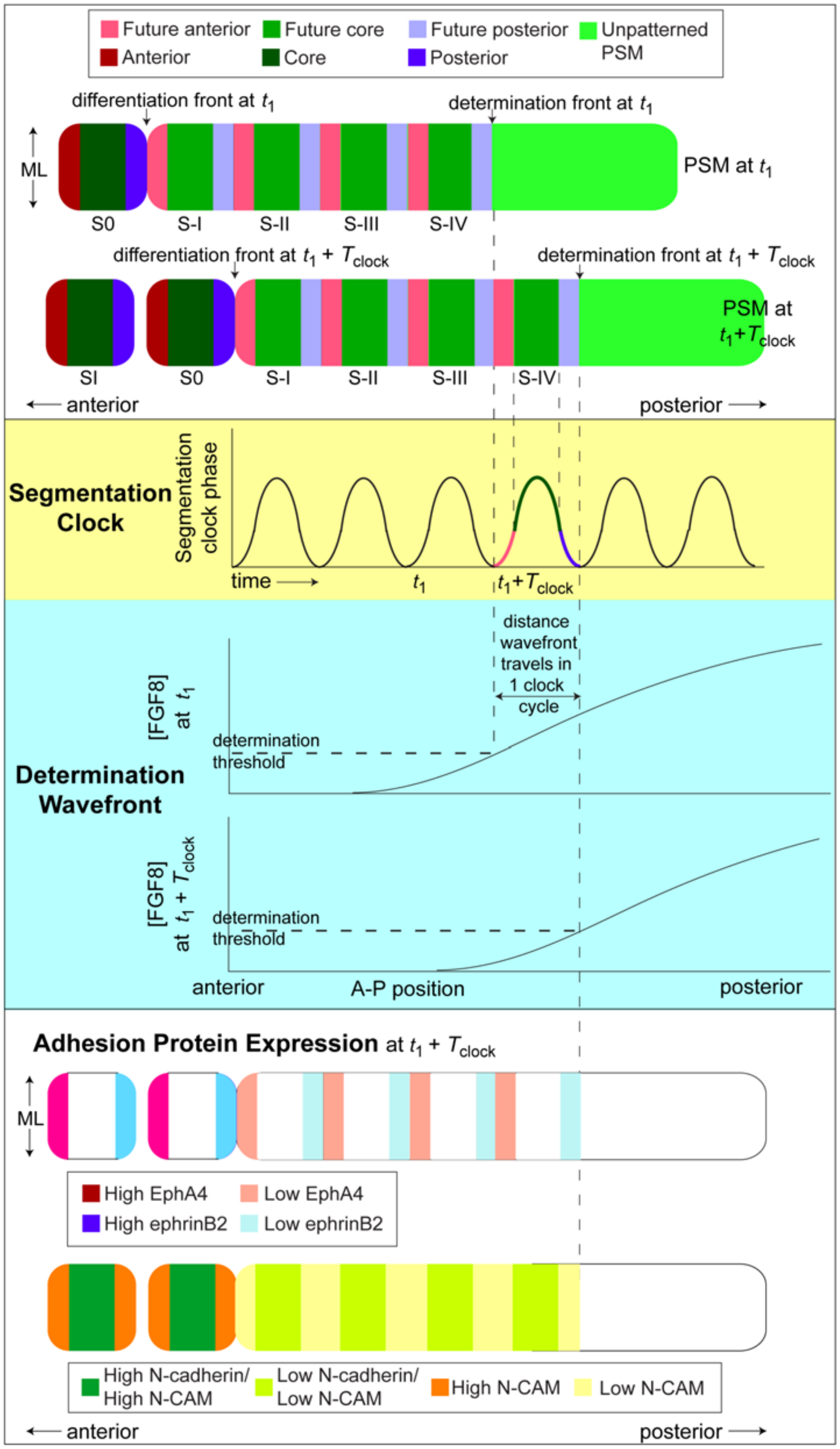
Future anterior
Future core
Future posterior
Unpatterned PSM
Anterior
Core
Posterior
differentiation front at t_1
determination front at t_1
ML
PSM at t_1
S0
S-I
S-II
S-III
S-IV
differentiation front at $t_1 + T_{clock}$
determination front at $t_1 + T_{clock}$
PSM at t_1+T_{clock}
SI
S0
S-I
S-II
S-III
S-IV
anterior
posterior
Segmentation Clock
Segmentation clock phase
time
t_1
t_1+T_{clock}
distance wavefront travels in 1 clock cycle
[FGF8] at t_1
determination threshold
Determination Wavefront
[FGF8] at $t_1 + T_{clock}$
determination threshold
anterior
A-P position
posterior
Adhesion Protein Expression at $t_1 + T_{clock}$
ML
High EphA4
Low EphA4
High ephrinB2
Low ephrinB2
High N-cadherin/ High N-CAM
Low N-cadherin/ Low N-CAM
High N-CAM
Low N-CAM
anterior
posterior

Figure 1.3: Schematic: A typical clock-and-wavefront model and its relationships to adhesion-protein expression. The AP position of a threshold concentration of temporally-decreasing FGF8 results in a posterior-propagating determination front, anterior to which a cell becomes competent to sense the state of its intracellular segmentation clock. At the determination front, a cell determines its fated somitic cell type (core, anterior or posterior) based on the state of its segmentation clock. Differentiation follows four segmentation clock periods (corresponding to four somite lengths) later. The PSM grows continuously in the posterior direction through addition of cells from the tailbud, maintaining its length. T_{clock} is the period of the segmentation clock. (*Below*) The clock-wavefront interaction results in the spatial pattern of adhesion protein expression that creates the differential adhesion between somitic cell types assumed in our computational implementation of the clock-and-wavefront model: EphA4 occurs in the anterior compartment of the forming somite and the anterior of the PSM; ephrinB2 occurs in the posterior compartment of the forming somite; N-CAM occurs throughout the anterior of the PSM and in the somites; and N-cadherin is strong in the core of forming and formed somites.

The basic clock-and-wavefront model, while powerful, is not a complete explanation of somitogenesis. It lacks molecular explanations for numerous mechanisms observed in somitogenesis, including the origins and behaviors of the clock and wavefront; how the intracellular segmentation clocks interact between cells to maintain synchrony and phase-locking despite molecular noise, cell movement and cell division; how the clock and wavefront interact to induce cell determination and differentiation; how oscillating segmentation-clock molecules cause stable expression and localization of structural proteins like cell adhesion molecules; and, finally, how the distribution of structural molecules leads to the dynamics of segmentation and epithelialization. Various existing submodels address one or more of these aspects: s*egmentation-clock submodels* address protein and mRNA oscillations within cells (Goldbeter and Pourquie 2008; Jensen, Pedersen et al. 2010); *synchronization submodels* address crosstalk, synchronization and phase-locking between cells' segmentation clocks (Lewis 2003; Uriu, Morishita *et al.* 2010); *determination front* and *differentiation submodels* address the spatial progression of PSM maturation and somite formation (Dubrulle, McGrew et al. 2001; Dubrulle and Pourquie 2004; Baker, Schnell et al. 2006; Baker, Schnell et al. 2006); *clock-wavefront readout submodels* address the signaling and genetic regulatory events through which the segmentation clock and determination front interact to create a stable segmental pattern of gene expression in the PSM (Oginuma, Niwa *et al.* 2008; Watanabe, Sato *et al.* 2009); and *cell adhesion submodels* address the cellular mechanics behind morphological changes during somite formation (Glazier, Zhang *et al.* 2008).

We drew on existing hypotheses for the intracellular segmentation-clock network from Goldbeter and Pourquié (Goldbeter and Pourquie 2008), Delta/Notch cell-cell segmentation-clock synchronization from Lewis (Lewis 2003), an Fgf8 threshold-positioned determination front from Dubrulle *et al.* (Dubrulle, McGrew et al. 2001; Dubrulle and Pourquie 2004) and differential-adhesion-mediated morphogenesis from Glazier *et al.* (Glazier, Zhang *et al.* 2008). As an example of the discriminatory power of building an integrated model, we found that we had to significantly alter and extend the Goldbeter-Pourquié intracellular segmentation-clock submodel (Goldbeter and Pourquie 2008) to make it compatible with Delta/Notch coupling and synchronization (based on (Lewis 2003)) between neighboring cells' segmentation clocks. Existing biological clock-and-wavefront readout submodels were insufficiently quantitative to allow creation of corresponding mathematical and computational models, so we developed our own readout submodel based on the available experimental data and previous speculative submodels.

Our composite computational model, simulated in the Glazier-Graner-Hogeweg (*GGH*)-model-based CompuCell3D (*CC3D*) simulation environment (Swat, Hester *et al.* 2009), reproduces spatially and temporally regular formation of an unlimited number of somites for biologically reasonable initial conditions and parameter values, somite-to-somite variation in somite shape and border morphology consistent with experiments, and anteriorly traveling ML stripes of expression of certain genes in the PSM. Changing certain model parameters changes the somite size, frequency of formation and shape, giving us insight into which mechanisms may be responsible for observed interspecies variation. Somite size in our model depends on both the segmentation-clock period and the PSM growth rate (which determines the rate of determination front progression), while somite formation frequency depends on the segmentation-clock period. The number of gene-expression stripes in our simulated PSM depends on the relationship between the segmentation-clock period, PSM growth rate and PSM length; and the relationship between the PSM growth rate and length depends on the mechanism that determines where and when cells differentiate.

1.3 Outline

In Chapter 2, BIOLOGICAL SUBMODELS OF SOMITOGENESIS PHENOMENA: BACKGROUND AND DEVELOPMENT, I will describe the development of the biological submodels that make up the full composite model of somitogenesis. For each submodel, I will describe the observations and experiments that supported the development of the submodel, previous incarnations of the submodel and the modifications that we made in order to include the submodel in the composite model. I will also state the simplifications and assumptions made for each submodel. Then, in Chapter 3, METHODS, I will describe how we translated the biological submodels into mathematical and computational submodels, including additional simplifications and assumptions, and describe how we carried out the perturbations reported in Chapter 4, RESULTS. In Chapter 4, RESULTS, I will present the outcomes of those perturbations. I will summarize the results and discuss their implications in Chapter 5, DISCUSSION. Also in Chapter 5, DISCUSSION, I will describe possibilities for future extensions of the composite model and make my concluding remarks.

Chapter 2

BIOLOGICAL SUBMODELS OF SOMITOGENESIS PHENOMENA: BACKGROUND AND DEVELOPMENT

In order to distinguish between the biological structures reflected in the model and the analogous features within the model itself, from this point on I will refer to model **cells**, **tissues**, **structures** and **cell-types** using **bold type** (their biological equivalents will remain in plain type).

2.1 Presomitic mesoderm

In contrast to the well-studied segmentation program in *Drosophila*, vertebrate somitogenesis segments a dynamic, morphologically non-uniform tissue that is cellular rather than syncytial and whose dimensions are much greater than the diffusion lengths of the graded patterning morphogens. A first step to describing the process of somitogenesis is to describe the unique tissue in which it occurs, the presomitic mesoderm (*PSM*). Thus, the first submodel that I will describe here is that of the growth and anatomy of the PSM itself.

The particular cells that comprise the PSM are continuously changing. New cells are constantly being added to the posterior of the PSM as they leave the highly proliferative tailbud. At the same time, a group of cells periodically leaves the anterior of the PSM, separating from the anterior tip to form a new somite. During the stages studied, cell addition and subtraction occur at similar rates, maintaining the PSM at a roughly constant length, so the PSM appears to travel posteriorly down the AP axis of the embryo, leaving a trail of somites in its wake. A cell's relative position within the PSM becomes progressively more anterior as the posterior and anterior borders of the PSM move posteriorly; for this reason, a cell's AP position in the PSM is correlated to the amount of time that it has spent as part of the PSM.

In the posterior-most region of the PSM, cells are loosely associated and highly motile; directly anterior to this region, cells are less motile, adhere more strongly to one another and pack more closely; as their position in the PSM becomes more anterior, cells become even less motile and begin to form stable neighbor relationships (Duband, Dufour et al. 1987; Kulesa and Fraser 2002; Delfini, Dubrulle et al. 2005; Benazeraf, Francois et al. 2010).

The shape of the PSM varies between the posterior and anterior regions as well. The posterior-most PSM is flat in the DV dimension and widely spread medio-laterally. Moving in the anterior direction, the PSM gradually extends dorso-ventrally (**Figure 1.2** (**C-D**)); at the same time, it becomes increasingly restricted medially by the notochord and the neural tube (the *midline structures*), and laterally by an enveloping network of fibronectin-rich extracellular matrix (*ECM*) that surrounds the PSM. This ECM thickens and organizes into a tubular structure around the more mature PSM and somites (Czirok, Zamir *et al.* 2006; Martins, Rifes *et al.* 2009). The PSM is further constrained dorsally and laterally by the epiblast and ventrally by the hypoblast.

Some cell proliferation occurs within the PSM: in zebrafish, an estimated 10-15% of PSM cells divide during a single roughly thirty-minute segmentation-clock oscillation (Horikawa, Ishimatsu *et al.* 2006), and the cell division time in the anterior chick PSM has been estimated at 10 hours (Stern, Fraser *et al.* 1988). Cell lineage studies in chick have shown that clones from cell divisions occurring in the anterior PSM typically become incorporated into the same somite (Stern, Fraser *et al.* 1988).

We modeled one column of the PSM as a 2D AP-by-ML strip of motile, non-proliferating **cells.** The modeled **PSM** is ten average **cell** diameters wide so it forms **somites** containing approximately 100 **cells**, corresponding to the roughly 100 cells in a 2D mid-plane AP-by-ML cross-section of a somite in the chick embryo (see **Figure 1.2**).

We neglected the DV extension of the anterior PSM (and consequently the rounding of the somites in this direction), as well as the ML asymmetries in signaling and morphology that result from the presence of the midline structures. While DV extension and ML asymmetries are significant in biological somitogenesis, in particular affecting the epithelialization of forming somites, their effects are sufficiently weak that treating the PSM and somites as two-dimensional and medio-laterally symmetric is reasonable at the level of detail of the model.

The focus of my work was on the patterning events in somitogenesis, rather than the formation of the PSM. Therefore, we modeled the PSM beginning seven or eight somite lengths posterior to the most recent somite, where cells adhere moderately to each other and pack closely, with little intercellular space (Duband, Dufour et al. 1987; Kulesa and Fraser 2002; Delfini, Dubrulle et al. 2005; Benazeraf, Francois et al. 2010), and neighboring cells' segmentation-clock oscillations have already synchronized (Mara, Schroeder *et al.* 2007) (**Figure 3.6**). We did not model the tailbud or the elaborate cell migration paths by which cells leave the tailbud to enter the extreme posterior of the PSM (Stern 1992; Knezevic, De Santo *et al.* 1998; Stern, Charite *et al.* 2006).

While there is a small amount of cell division in the anterior PSM, we assumed that cell division in this region contributes primarily to the DV extension of the PSM and that AP extension of the

PSM is due to the addition of cells from the posterior. Our model simplified growth in the modeled region of the PSM as occurring solely due to the steady addition of cells from the more posterior PSM. We also simplified cell behavior in the PSM by assuming relatively uniform cell motility in the modeled region of the PSM.

2.2 Establishment of morphogen gradients

At least three signaling molecules form developmentally important gradients in the PSM. *Fibroblast growth factor 8* (*FGF8*) and the *wingless* homolog *Wnt3a* are both present at high concentrations in the tailbud and posterior PSM, and decrease anteriorly (Dubrulle, McGrew *et al.* 2001; Aulehla, Wehrle *et al.* 2003). Raldh2, which synthesizes retinoic acid (*RA*), a differentiation promoter, is expressed in newly-formed somites and generates a posteriorly-decreasing retinoid signaling gradient in the anterior PSM (Blentic, Gale *et al.* 2003; Diez del Corral, Olivera-Martinez *et al.* 2003).

Reasonable estimates for the diffusion length of FGF8 in the PSM tissue are on the order of a micron (Goldbeter, Gonze *et al.* 2007). A classic source-sink diffusion model is therefore not reasonable for the establishment of the FGF8 gradient in the PSM—a gradient that spans the greater fraction of the AP length of the PSM, a length on the order of tenths of millimeters (Diez del Corral, Olivera-Martinez *et al.* 2003). The same is true of Wnt3a, another large signaling molecule (Christian 2000; Aulehla, Wehrle *et al.* 2003). A submodel that does not count on diffusion as the primary mechanism is necessary to explain the AP FGF8 and Wnt3a gradients observed in the PSM. In building our composite model, we adopted submodels for the FGF8 and Wnt3a gradients proposed by Dubrulle and Pourquié (Dubrulle and Pourquie 2004), and Aulehla *et al.* (Aulehla, Wehrle *et al.* 2003), respectively.

Dubrulle and Pourquié (Dubrulle and Pourquie 2004) proposed the following model for FGF8 gradient formation in the PSM: cells transcribe *fgf8* mRNA while residing in the tailbud and cease transcription once they enter the PSM, but continue FGF8 translation for as long as *fgf8* mRNA is present. Because cells move anteriorly relative to the PSM boundaries as they age, *fgf8* mRNA decay establishes an anteriorly-decreasing gradient that the FGF8 protein concentration mirrors. Results of their ensuing experiments strongly supported this model. Exonic probes for *fgf8* transcription only detected active transcription in the posterior-most part of the embryo, and treatment with a transcription inhibitor did not affect the presence of *fgf8* mRNA in the rest of the PSM. Furthermore, they established that a signal from the posterior PSM is not necessary for maintaining *fgf8* expression in the PSM: surgically removing the tailbud and posterior PSM did not change *fgf8* expression in the rest of the PSM (Dubrulle and Pourquie 2004).

Aulehla and colleagues (Aulehla, Wehrle *et al.* 2003) independently suggested that a related mechanism forms the Wnt3a signaling gradient in the PSM: *wnt3a* mRNA is expressed only in the tailbud and Wnt3a protein translation ceases in cells as they enter the PSM, so protein decay establishes a posterior-to-anterior signaling gradient. While their model is not as strongly backed by experimental results as has been the model of the FGF8 gradient, it is reasonable and consistent with other observations (Aulehla, Wehrle *et al.* 2003).

Unlike FGF8 and Wnt3a, RA is a small molecule with a large estimated diffusion length (Goldbeter, Gonze *et al.* 2007). Moreover, the observed length of the RA signaling gradient is shorter than that of either FGF8 or Wnt3a (Diez del Corral, Olivera-Martinez *et al.* 2003). Thus, a classic diffusion mechanism is a reasonable model for explaining the gradient of RA signaling in the anterior PSM.

We did not explicitly model RA signaling in the PSM. Instead, we made the simplifying assumption that the primary role of RA in the anterior PSM is as an FGF8 antagonist and differentiation promoter. In our model, RA is assumed to act in concert with FGF8 to trigger an abrupt differentiation event in cells at a particular FGF8 signaling threshold, as I will discuss in Section 3.2.

2.3 Segmentation clock

The first evidence for the segmentation-clock component of Cooke and Zeeman's clock-and-wavefront model was provided by Palmeirim *et al.* (Palmeirim, Henrique *et al.* 1997), who observed oscillations in *c-hairy1* expression (downstream of Notch signaling) in chick PSM. Following this observation, entire cohorts of mRNA and protein oscillations have been observed in the PSM (Forsberg, Crozet et al. 1998; Aulehla, Wehrle et al. 2003; Dequeant, Glynn et al. 2006). Her1 and Her7 oscillations downstream of Delta/Notch signaling are prominent in zebrafish (Jiang, Aerne et al. 2000; Horikawa, Ishimatsu et al. 2006; Mara, Schroeder et al. 2007; Ozbudak and Lewis 2008). In mouse, expression of genes downstream of FGF, Wnt and Delta/Notch signaling oscillate with the same period. Gene-expression oscillations downstream of FGF and Delta/Notch share a phase, and are half a period out of phase with gene-expression oscillations downstream of Wnt (Aulehla, Wehrle et al. 2003; Dequeant, Glynn et al. 2006). These oscillations can occur cell-autonomously, persisting even in dissociated PSM cells (Palmeirim, Henrique *et al.* 1997).

Within the PSM, Delta/Notch signaling couples, synchronizes and maintains synchrony among neighboring PSM cells against noise from cell proliferation, stochastic gene expression and cell movement (Horikawa, Ishimatsu et al. 2006; Mara, Schroeder et al. 2007; Ozbudak and Lewis

2008). PSM cells initiate and synchronize their segmentation clocks during their sojourn in the tailbud and posterior-most PSM, before entering the more anterior region of the PSM (Mara, Schroeder *et al.* 2007).

Local synchronization of oscillations between neighboring PSM cells at the same AP position is crucial to segmentation (Ozbudak and Lewis 2008). However, PSM cells at different AP positions do not oscillate in phase. Instead, the phases of the oscillators display distinctive dynamic patterns across the PSM. In chick, lunatic fringe (*Lfng,* downstream of Notch signaling) is initially expressed in the posterior PSM as a broad ML stripe that travels anteriorly, gradually slowing and narrowing before finally arresting in the anterior PSM at the location of the next presumptive somite. Axin2 (downstream of Wnt3a signaling) is expressed in a similar pattern about half a segmentation-clock period after Lfng, so that Axin2 stripes appear to follow Lfng stripes down the AP axis of the embryo. Other oscillating molecules in the segmentation-clock network display similar patterns (Palmeirim, Henrique *et al.* 1997; Forsberg, Crozet *et al.* 1998; McGrew, Dale *et al.* 1998; Aulehla, Wehrle *et al.* 2003).

2.3.1 Single-cell network

Goldbeter and Pourquié (Goldbeter and Pourquie 2008) developed a detailed model of the mouse/chick segmentation-clock network in a single cell, including independent FGF, Wnt and Delta/Notch oscillator loops. Each oscillator loop pathway includes negative feedback in which a downstream target of the signaling pathway inhibits the signaling that promotes its own transcription. In the FGF loop, Dusp6 inhibits the activation of ERK, an early player in the cascade leading to Dusp6 activation. Axin2 works in complex with Gsk3β to promote phosphorylation and subsequent degradation of β-catenin, a component of canonical Wnt signaling that upregulates transcription of Axin2. Finally, Lfng inhibits Notch signaling, while Notch signaling upregulates Lfng. Two connections couple the pathway loops: an unknown transcription factor in the FGF pathway (designated X_a and thought to be a member of the E-twenty-six (*ETS*) family of transcription factors) upregulates Axin2 in the Wnt pathway, and uncomplexed Gskβ from the Wnt pathway inhibits Notch signaling (Goldbeter and Pourquie 2008). If the coupling terms are omitted, simulations of the uncoupled FGF, Wnt and Notch loops oscillate autonomously, each with a different frequency. With the inter-loop coupling proposed by Goldbeter and Pourquié (Goldbeter and Pourquie 2008), the simulated oscillations phase-lock, with Lfng (in the Notch pathway) and Dusp6 (in the FGF pathway) oscillating in phase with one another and out of phase with Axin2 (in the Wnt pathway), as observed experimentally (Aulehla, Wehrle et al. 2003; Dequeant, Glynn et al. 2006).

2.3.2 Cell-cell coupling and synchronization

Özbudak and Lewis demonstrated that synchronizing gene oscillations in the posterior of the zebrafish PSM (synchronization that persists into the anterior of the PSM and is crucial for proper segmentation) requires Delta/Notch signaling; when Notch signaling is inhibited with DAPT treatment, expression of the oscillating genes *her1* and *her7* in the posterior PSM assumes a salt-and-pepper pattern—more characteristic of the more posterior region where oscillations are not yet synchronized—in lieu of the wild-type uniform expression (Ozbudak and Lewis 2008). Following this work, Mara *et al.* identified regions in the posterior-most zebrafish PSM and anterior tailbud where segmentation-clock oscillations are primed, initiated and synchronized (Mara, Schroeder *et al.* 2007).

Motivated by the observation that Delta/Notch signaling is necessary for oscillator synchronization, Lewis (Lewis 2003) developed a biological model of the zebrafish segmentation clock composed of a single intracellular oscillatory network loop in the Delta/Notch signaling pathway that couples between contacting cells via juxtacrine Delta/Notch signaling[1]. In Lewis' segmentation-clock submodel, oscillations result from a negative feedback loop, with Her1/7 proteins downstream of Delta/Notch signaling repressing their own transcription. Neighboring cells' oscillation networks are "driven" by juxtacrine Delta signaling by their neighbors, coupling neighboring oscillators and ultimately allowing them to synchronize with one another. A schematic of Lewis' segmentation-clock network is shown in **Figure 2.1**.

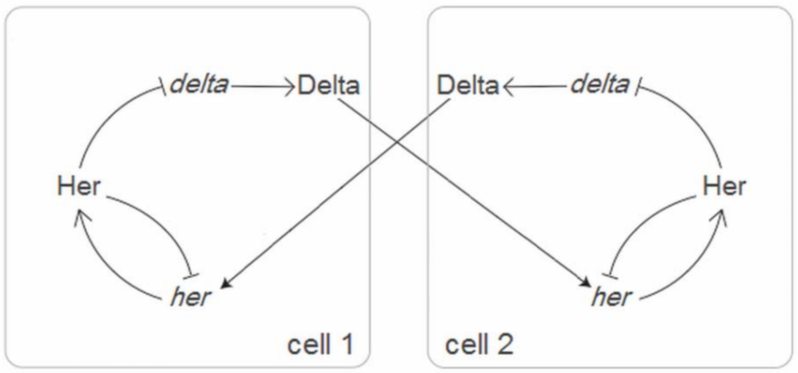

Figure 2.1: Schematic of Lewis-style coupling between neighboring cells' segmentation-clock networks via Delta/Notch signaling.

We took Lewis-style juxtacrine Delta/Notch signaling between neighboring cells as the prototype for intercellular segmentation-clock coupling in our extended submodel of the segmentation-clock network. It is worthwhile to note that, while observations directly supporting Delta/Notch-mediated synchronization are so far limited to the zebrafish, the phenotype resulting from inhibited Delta/Notch signaling in the posterior PSM—disruption of somite formation some time later, when the perturbed (previously posterior) region of the PSM

[1] Juxtacrine signaling is signaling that requires contact between cells. In the case of juxtacrine Delta/Notch signaling, both the ligand, Delta, and the receptor, Notch, are tethered to the cell membrane.

would otherwise be forming somites (Ozbudak and Lewis 2008)—is observed in chick and mouse as well (Conlon, Reaume *et al.* 1995; Hrabe de Angelis, McIntyre *et al.* 1997; Evrard, Lun *et al.* 1998; Barrantes, Elia *et al.* 1999; Jiang, Aerne *et al.* 2000), supporting a similar role for Delta/Notch oscillator synchronization in these organisms.

2.3.3 Extended segmentation-clock network submodel with cell-cell coupling

We incorporated the Goldbeter-Pourquié description of the segmentation clock into our integrated model rather than choosing a simpler model (such as a phase oscillator, simple sine wave or single negative feedback loop) for two related reasons: (1) a primary aim in this work is to integrate current models of somitogenesis mechanisms at different scales; and (2) the Goldbeter-Pourquié model allowed us to explicitly model the connections between the segmentation clock and local FGF, Wnt and Delta signaling (discussed in Section 3.2.4), and between the expression of oscillating clock molecules and the eventual differentiated states of cells (discussed in Section 3.2.5).

In a single cell, uninfluenced by outside factors, the loop-to-loop coupling mechanisms in the Goldbeter-Pourquié model maintain the appropriate phase relationships between the oscillators (Goldbeter and Pourquie 2008). They are not, however, sufficient to explain observed behaviors of multiple cells. Extending the Goldbeter- Pourquié model network to multiple cells required us to include Delta/Notch coupling and synchronization between neighboring cells. As currently understood, cell-cell synchronization occurs through the Notch pathway. Coupling between pathways in the Goldbeter-Pourquié model is directional: the FGF pathway feeds forward to the Wnt pathway, which feeds forward to the Notch pathway, with no feedback from the Notch pathway back to the FGF or Wnt pathways. Thus, the FGF and Wnt oscillators in Notch-coupled

Goldbeter-Pourquié networks cannot entrain within or between cells. Experimentally-observed FGF- and Wnt-oscillator entrainment requires at least one additional coupling within the intracellular segmentation-clock network to allow the Notch oscillator to entrain the FGF and Wnt oscillators, or additional or modified juxtacrine signaling to entrain the FGF oscillators between cells.

Experiments, while not conclusive, do suggest a feedback coupling from the Notch oscillator to the FGF oscillator. Hes7, a cycling gene downstream of Notch, regulates cyclic expression of Dusp4, a downstream FGF-signaling inhibitor exhibiting the same set of behaviors as Dusp6 in the segmentation clock (Niwa, Masamizu *et al.* 2007). We therefore introduced a generic Hes7-like inhibitory *Dusp modification factor* (*DMF*) into our submodel of the Notch signaling cascade, allowing the Notch loop to influence the FGF loop. Free Gsk3β generally inhibits Notch signaling (Espinosa, Ingles-Esteve *et al.* 2003), so adding DMF and Delta downstream of Notch also required us to model Gsk3β phosphorylation of the intracellular domain of cleaved Notch (*NICD*) in place of the direct inhibition of Lfng by Gsk3β previously modeled by Goldbeter and Pourquié.

In our composite model, the segmentation-clock network in each cell connects to FGF8 and Wnt3a signaling from the local environment and Delta signaling from the cell's immediate neighbors. **Figure 2.2** shows our segmentation-clock submodel and indicates our modifications to/extensions of the original Goldbeter-Pourquié model.

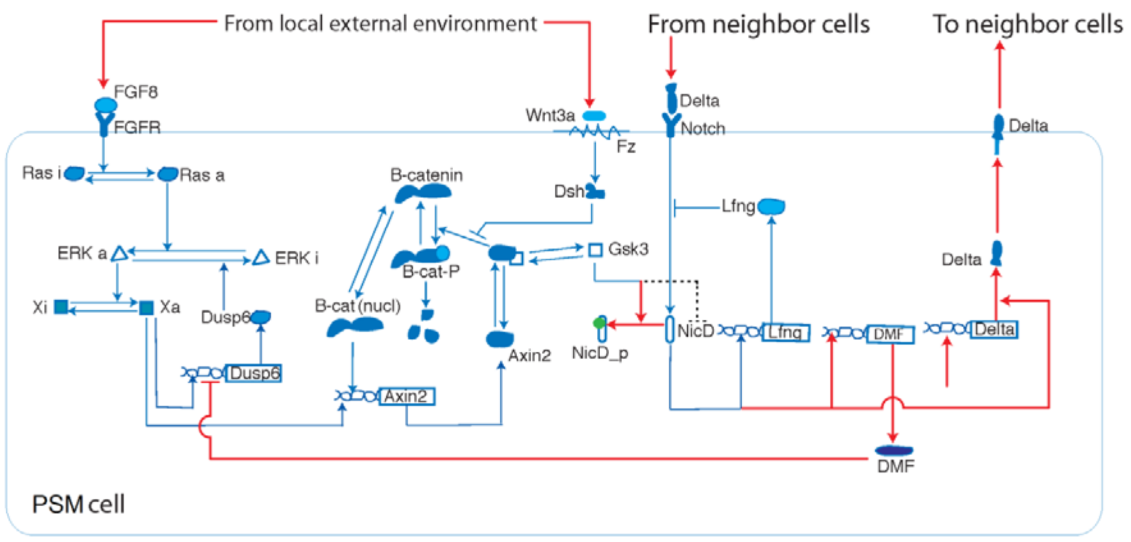

Figure 2.2: Schematic: Extended three-oscillator, externally-coupled biological sub-model for the segmentation-clock network. We adapted and extended the Goldbeter and Pourquié segmentation-clock biological model to include Delta signaling and to allow the experimentally-observed phase locking between the FGF, Wnt and Notch loops in multiple coupled cells. Red lines show connections/processes in our biological model that are not in the Goldbeter-Pourquié biological model and dotted lines show connections in the Goldbeter-Pourquié biological model not used in our biological model.

2.4 Differential-adhesion-driven cell sorting

Because the complex clock-wavefront interaction at cell determination is noisy, initial determination can result in significant misplacement of determined and differentiated cells. Forming repeatable somites of a specified size and shape that are separated by clean intersomitic gaps, as observed *in vivo*, requires a mechanism to refine the initial spatial distribution of cell types. One possible correction mechanism would be that cells remain labile after initial determination and can re-determine or re-differentiate in response to the predominant types of their neighbors. Another would be that misplaced cells undergo apoptosis in response to being surrounded by cells of a different type. Such mechanisms are potentially fast and work no matter how far a cell is misplaced from its appropriate location. However, neither of these mechanisms accounts for observed migrations of individual cells across the compartment boundaries within a forming somite or across a forming intersomitic boundary (Kulesa and Fraser 2002).

Glazier *et al.* (Glazier, Zhang *et al.* 2008) proposed that somite boundaries arise spontaneously through cell sorting due to intrinsic random cell motility and patterns of differential cell-cell adhesion resulting from adhesion-mediating molecules at cells' membranes.

2.4.1 Patterns of adhesion protein expression in the anterior PSM and somites

Cells at the anterior of the PSM express a variety of cell-surface adhesion molecules that modify cell-cell interactions during and after the period in which they reorganize to form a new somite, they. Adhesion molecules include homophilic neural cell-adhesion molecule (*N-CAM*) and neural cadherin (*N-cadherin*), and heterorepulsive EphA4 and ephrinB2 (see **Figure 1.3**). EphA4 and ephrinB2 are a complementary pair of surface receptors that are expressed in distinct ML bands in the forming somite and anterior PSM: EphA4 is expressed in the anterior compartments of forming somites and in the anterior tip of the PSM, while ephrinB2 is expressed in the posterior compartments of forming somites (Durbin, Brennan *et al.* 1998; Baker and Antin 2003; Barrios, Poole *et al.* 2003; Watanabe, Sato *et al.* 2009) (see **Figure 1.3**). When juxtaposed, cells from these two populations induce bidirectional signals that change cell morphology, leading to an effective retraction and "repulsion" between the signaling cells. The precise mechanism behind Eph/ephrin-mediated repulsion is unknown, but is likely due to mutually-induced collapse of the cortical actin cytoskeleton (Harbott and Nobes 2005).

N-CAM and N-cadherin are homophilic trans-membrane adhesion receptors that contribute to cell-cell adhesion. As a somite forms, N-cadherin expression increases and localizes predominantly to the apical surfaces of the epithelialized cells that form the outer layer of cells in the somite (contrary to the norm in most tissues, the apical surfaces face the somite core and the basal surfaces face the exterior of the somite) (Duband, Dufour *et al.* 1987). N-cadherin disruption leads to fragmentation of somites and separation of the anterior and posterior somite compartments (Horikawa, Radice *et al.* 1999). N-CAM, which results in weaker, less specific adhesion than N-cadherin, is expressed relatively uniformly throughout the anterior PSM and the somites (Duband, Dufour *et al.* 1987) (see **Figure 1.3**).

2.4.2 Submodel of adhesion-driven cell sorting at somite borders

According to the Glazier *et al.* submodel of adhesion-driven cell sorting, mutual repulsion between strongly EphA4- and ephrinB2-expressing cells leads to somite border and intersomitic gap formation. The anterior (EphA4) and posterior (ephrinB2) somite compartments adhere strongly to a core of highly adhesive cells with high concentrations of N-CAM and N-cadherin (Glazier, Zhang *et al.* 2008). Simulations of Glazier *et al.*'s differential adhesion model reproduce many of the characteristics of somitogenesis *in vivo*, including somite compartment separation in the absence of N-cadherin and adhesion-induced-cell-migration correction of indistinct somite boundaries (Glazier, Zhang *et al.* 2008).

Our adhesion- and motility-based submodel of somite formation is a simplification of the Glazier *et al.* model (Glazier, Zhang *et al.* 2008). While Glazier *et al.* used six somitic cell types, our differential cell-cell adhesion submodel uses three, representing cells with high EphA4 and high N-CAM, high ephrinB2 and high N-CAM, and high N-CAM and N-cadherin concentrations at their membranes. Furthermore, whereas Glazier *et al.* did not address the issues of cell-type determination prior to differentiation or the mechanisms that initially establish the spatial pattern of adhesion-protein expression, we combined the differential adhesion submodel with submodels of these mechanisms to address both of these issues.

2.5 Determination and differentiation fronts

In addition to the segmentation clock, the clock-and-wavefront model calls for a positional front at which cells undergo a maturing event that marks them as belonging to a particular somite or somite border and ultimately segments the tissue into periodic blocks.

One extension of the clock-and-wavefront model distinguishes between two positional "fronts" in the embryo at which cells undergo apparently irreversible changes on their way to becoming part of a somite—the *determination front,* at which cells acquire their eventual somitic fates, and the *differentiation front,* at which cells express the complement of adhesion and signaling proteins that lead to somite formation and cellular positions within the somites.

2.5.1 FGF8 threshold-based determination-front submodel

At the time that they proposed their model, Cooke and Zeeman suggested that the role of the wavefront could be played by a regressing gradient of some morphogen (Cooke and Zeeman 1976). Decades later, Dubrulle *et al.* proposed and supported a model in which the determination front was positioned by a threshold in FGF8 signaling (Dubrulle, McGrew *et al.* 2001). By electroporating chick PSM with an FGF8 expression vector, they demonstrated that high levels of FGF8 signaling maintain PSM cells in an immature state: cells under the influence of constitutive FGF8 signaling remained morphologically similar to posterior PSM cells, failing to form somites or somite borders. They also showed the opposite result: inhibiting FGF8 signaling induced early cell maturation.

In the same study, Dubrulle *et al.* demonstrated that, anterior to an AP position roughly four somite widths posterior to the most recent somite border, cells are no longer labile with regard to their somitic fates (Dubrulle, McGrew *et al.* 2001). When they inverted segments of PSM tissue posterior to this position, cells differentiated and incorporated into somites according to their new AP positions. When, on the other hand, they inverted segments of PSM tissue anterior to this position, cells differentiated according to their original AP positions (*i.e.*, inverting a somite-wide segment of tissue anterior to this position resulted in formation of a "backwards" somite). Because cell fates are apparently determined, though not yet realized, at this position, Dubrulle *et al.* dubbed this position the "differentiation front."

Based on their results, Dubrulle *et al.* (Dubrulle, McGrew *et al.* 2001) proposed a biological model in which the advancing FGF8 gradient serves as a determination front in the clock-and-wavefront model: below a threshold concentration of FGF8, cells become competent to respond to the states of their segmentation clocks. Cells posterior to the position of the FGF8 threshold have undetermined somitic fates and cells anterior to the position of the FGF8 threshold have determined somitic fates. We adopted an FGF8-threshold-positioned determination-front submodel in our composite model (see **Figure 1.3**).

2.5.2 Differentiation-front submodels

In vivo, cell differentiation involves continuous changes in cell properties, behaviors and interactions over a finite amount of time. We simplified differentiation in our biological model by describing it as occurring in two discrete, instantaneous steps. First, at determination, **cells** in our model begin to weakly exhibit the adhesion characteristics of their determined types, roughly approximating the early accumulation of adhesion-altering proteins at the membranes of biological cells and allowing some adhesion-mediated maintenance of future intrasomitic compartments prior to full differentiation. Some time later, at differentiation, **cells** undergo a second, more drastic change and assume the adhesion characteristics of their final differentiated states, which then drive **somite** formation, determine **somite** shape and maintain intrasomitic compartments.

The determination front model assumes that a second mechanism triggers differentiation some time after determination (approximately four segmentation-clock periods later in chick). In the course of our analysis, we considered two types of biological model for the delay between determination and differentiation: a cell-autonomous time delay and a positional differentiation signal.

2.5.2.1 Time-delayed differentiation-front submodel

In a *cell-autonomous delay model*, an intracellular "timer" counts down the time between cell determination and differentiation independent of the cell's external environment. Such a timer could represent, *e.g.*, the time a cell takes to express and manufacture adhesion proteins and localize them to the cell membrane, or the time the local FGF8 and/or Wnt3a concentrations take to fall below additional threshold concentrations that permit full differentiation. Short FGF8 (Goldbeter, Gonze *et al.* 2007) and Wnt3a (Christian 2000; Aulehla, Wehrle *et al.* 2003) diffusion lengths make local FGF8 and Wnt3a signaling nearly cell-autonomous; *fgf8* mRNA and/or Wnt3a protein decay effectively constitute the intracellular timer in such a scenario. We modeled both a generic intracellular timer that does not rely on any particular countdown mechanism and a second FGF8-concentration differentiation threshold. The choice of timer did not significantly affect our results.

2.5.2.2 Positional differentiation-front submodel

In a *positional differentiation-front* model, the position of the differentiation front anterior to the determination front results from a separate signaling threshold mechanism that triggers determined cells to undergo full differentiation. The likeliest candidate for such a positional differentiation signal is RA originating in the somites and the anterior tip of the PSM (Palmeirim, Dubrulle *et al.* 1998; Diez del Corral and Storey 2004; Goldbeter, Gonze *et al.* 2007). RA, which diffuses from the somites and the anterior tip of the PSM into more posterior regions, has a concentration that depends on the distance from the anterior tip of the PSM and is independent of cells' history or their distance from the tailbud, except to the extent that distance from the tailbud and distance from the anterior end of the PSM are correlated. *In vivo*, mutually antagonistic opposing gradients of FGF8 and RA probably cooperate in positioning the differentiation front (Goldbeter, Gonze *et al.* 2007). We did not explicitly model RA and RA-FGF8 interaction, choosing instead to model the simplest-case scenario involving morphogen-segmentation-clock interaction and threshold-positioned developmental fronts. Instead of explicitly modeling the RA concentration field when considering the positional differentiation-front model, we made the simplifying assumption that, once the PSM reaches its full length, a differentiation front begins at the anterior tip of the PSM and moves posteriorly at a constant speed equal to the rate of PSM growth, thus maintaining the PSM at a constant length. This assumption produces a differentiation front essentially indistinguishable from that for a simple two-gradient model that includes RA explicitly.

2.6 Clock-wavefront readout

An explanatory clock-and-wavefront model of somitogenesis requires a mechanism by which the segmentation clock and advancing wavefront interact to induce cell determination and subsequent differentiation. In our case, the biological clock-wavefront readout submodel must translate the concentrations of oscillatory segmentation-clock players at the time a cell first experiences below-threshold FGF8 signaling into stable patterns of eventual EphA4, ephrinB2, N-CAM and/or N-cadherin expression (see **Figure 1.3**). In the absence of an existing model at the appropriate level of detail, we developed our own novel clock-wavefront readout submodel. Our clock-wavefront readout submodel depends on speculative connections between molecules downstream of FGF8, Wnt3a and Notch signaling, and the adhesion molecules EphA4, ephrinB2, N-CAM and N-cadherin. The speculative connections in our clock-wavefront readout submodel and our justifications for drawing them are described below and illustrated in **Figure 2.3**.

Mesp2 (cMeso in chick) is a basic helix-loop-helix (*bHLH*) transcription factor whose localization in the anterior PSM is controlled by Notch signaling (Takahashi, Inoue *et al.* 2003; Yasuhiko, Haraguchi *et al.* 2006). A ML stripe of Mesp2 mRNA and protein appears one somite length posterior to the anterior tip of the PSM, marking the location where the next somite border will form (Morimoto, Takahashi *et al.* 2005). Intersomitic border formation and normal intrasomitic compartmentalization both require Mesp2 (Takahashi, Koizumi *et al.* 2000; Takahashi, Inoue *et* *al.* 2003; Morimoto, Takahashi *et al.* 2005; Nakajima, Morimoto *et al.* 2006; Yasuhiko, Haraguchi *et al.* 2006; Saga 2007). FGF8 signaling inhibits Mesp2 expression (Delfini, Dubrulle *et al.* 2005), which is consistent with an FGF8 threshold-as-wavefront model in which Mesp2 is an important mediator between the clock, wavefront and final determination of EphA4-expressing cells. In our biological readout submodel, EphA4 expression is regulated by Notch signaling, with cMeso acting as an intermediary; when a cell's local FGF8 signaling falls below the determination threshold, high cMeso expression (driven by Notch signaling) leads to expression of EphA4.

Cytoplasmically available β-catenin directly affects N-cadherin- and N-CAM-mediated cell-cell adhesion: β-catenin recruited from the cytoplasm by a membrane-associated complex stabilizes N-cadherin and N-CAM at the plasma membrane (Ozawa, Baribault *et al.* 1989), and high levels of β-catenin saturate β-catenin binding to cadherin at the plasma membrane and increase cell-cell adhesion *in vivo* (Nelson and Nusse 2004). As one of the cycling components in the segmentation clock (see **Figure 3**), cytoplasmic β-catenin is thus an attractive potential link between the segmentation clock and the presence of active N-cadherin/N-CAM at the cell membrane. In our biological clock-wavefront readout submodel, a cell's concentration of cytoplasmic β-catenin at the time when the local FGF8 signaling falls below the determination threshold determines the amounts of stable N-CAM and N-cadherin that will be present on the cell's surface when it matures into a somitic cell.

AP compartmentalization and epithelialization of somites require the bHLH transcription factor Paraxis (Burgess, Rawls *et al.* 1996; Johnson, Rhee *et al.* 2001), which is expressed in the anterior-most PSM and somites (Burgess, Cserjesi *et al.* 1995; Dubrulle, McGrew *et al.* 2001). Paraxis is a target of β-catenin-dependent Wnt signaling (Linker, Lesbros *et al.* 2005), and FGF8 signaling restricts its expression to the anterior of the PSM (Dubrulle, McGrew *et al.* 2001), making it a potential player in the clock-wavefront interaction. In Paraxis-null mice, ephrinB2 transcription in the somites is diffuse rather than restricted to the posterior of the somites (Johnson, Rhee *et al.* 2001). The PSM of these mice partially segments, but intersomitic gaps do not form and the outer cells of the somites do not entirely epithelialize (Burgess, Rawls *et al.* 1996). These characteristics suggest that the Paraxis-null somitic phenotype is a result of disrupted Eph-ephrin signaling, which is involved in epithelialization and gap formation (Durbin, Brennan *et al.* 1998; Barrios, Poole *et al.* 2003; Watanabe, Sato *et al.* 2009), and which serves to segregate EphA4- and ephrinB2-expressing cell populations (Mellitzer, Xu *et al.* 1999; Glazier, Zhang *et al.* 2008). In our biological clock-wavefront readout submodel, when local FGF8 signaling falls below the determination threshold, the presence of a high concentration of Paraxis (downstream of Wnt3a signaling) leads to the expression and stabilization of ephrinB2.

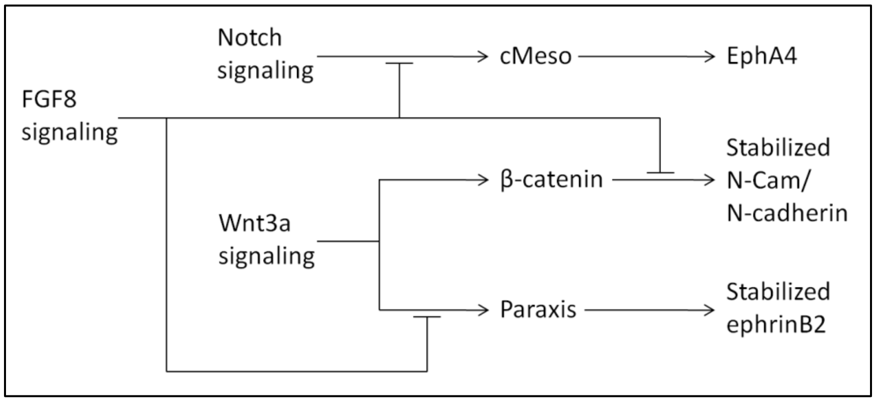

Figure 2.3: Clock-wavefront readout biological submodel. Notch signaling regulates EphA4 through cMeso (Mesp2), cytoplasmic β-catenin in the Wnt3a pathway stabilizes N-CAM and N-cadherin at the plasma membrane, and functional ephrinB2 signaling requires Paraxis, downstream of Wnt3a signaling. When FGF8 signaling decreases below a threshold, it releases the inhibition of cMeso, Paraxis and N-Cam/N-cadherin, leading to expression of adhesion proteins on the cell membrane.

2.7 Cell types

A ***cell type*** in our model denotes a collection of model **cells** that share a unique set of properties, interactions and dynamics. **Cells** in our model occupy space (as opposed to being points), are deformable and motile (unless specified otherwise), and have variable adhesivity to other **cells**. The **cells** in our model are nonpolar and, with the exception of **Source cells**, have a constant volume and do not divide.

Cell types in our model reflect the simplification that differentiation happens in two steps. At determination, **cells** assume **cell types** with adhesion properties that are intermediate between undetermined **PSM** and differentiated **somitic cell types**. At differentiation, **cells** assume **cell types** with appropriate adhesion properties to form and maintain **somites**. This reflects the premise that differential adhesion is a driving factor in the dynamics of somite and somite-border morphology.

Our model has ten **cell types**: **Medium**, **Wall**, and **Source cells** do not correspond to actual biological cells, but represent the environment and structures surrounding the PSM; **PSM cells** represent undetermined cells in the modeled region of the PSM; **pre_EphA4**, **pre_ephrinB2**, and **pre_Core cells** represent cells with determined somitic cell types; and **EphA4**, **ephrinB2** and **Core cells** represent the somitic cell types. **Cells** of different **cell types** can differ in size, motility, adhesion to other **cells**, subcellular properties, contribution and response to signaling, and, ultimately, their roles in **PSM** and **somite** dynamics.

(0) **Medium** represents ECM and fluid in the tissue that is not explicitly modeled, and occupies any space that is not otherwise occupied by **cells**.

(1) **Wall cells** are arranged in immobile columns on either side of the **PSM**, representing the medial and lateral structures and extracellular material constraining the PSM to form a single anterior-posterior band. **Wall cells** disappear at the differentiation front to allow relaxation of **somite** boundaries and because they are no longer necessary to constrain **PSM** growth. **Wall cells** are the only **cells** in the simulation that have inflexible shapes and do not move.

(2) **Source cells** represent the addition of new cells to the modeled region of the PSM from the posterior-most PSM and tailbud. **Source cells** grow and divide at a constant rate to produce **PSM cells**; they are the only **cells** in the model that divide. Each **Source cell** contains a segmentation-clock network and high concentrations of *fgf8* mRNA, FGF8 and Wnt3a, which its progeny inherit. **Cells** in the posterior-most layer (those in contact with **Medium** at the posterior end) are, by default, **Source cells**: **Source cells** that fall out of contact with **Medium** at the posterior end of the **PSM** become **PSM cells**, and **PSM cells** that come into contact with **Medium** at the posterior end of the **PSM** become **Source** cells.

The remaining **cell types** represent biological cells in the PSM and somites.

(3) **PSM cells** represent cells in the modeled region of the PSM that are posterior to the determination front, and therefore do not have assigned somitic cell-type fates. They have higher motility and weaker cell-cell adhesion than other **cells** in the model. **PSM cells** each contain a segmentation-clock network and are under the influence of FGF8, Wnt3a and Delta signaling from the local field environment and surrounding **cells**. **PSM cells** do not transcribe *fgf8* mRNA or translate new Wnt3a protein, but do produce and

secrete FGF8 from existing intracellular *fgf8* mRNA and signal with existing Wnt3a. **PSM cells** will become **pre_EphA4**, **pre_ephrinB2** or **pre_Core cells**, depending on the state of their intracellular segmentation clocks when they reach the determination front.

pre_EphA4, **pre_ephrinB2** and **pre_Core cells**, the *determined **cells***, represent cells in the PSM that are anterior to the determination front and posterior to the differentiation front (and so have been assigned fated somitic cells types but have not fully differentiated into their fated cell types). They are similar to **PSM cells**, with slightly lower motility and **cell-cell** adhesion strengths similar to those of **PSM cells**. Segmentation-clock networks in determined **cells** no longer oscillate, but do continue Delta signaling to neighboring **PSM cells**. In the differentiation signaling-threshold submodel, determined **cells** differentiate in response to the differentiation-threshold concentration of FGF8, but otherwise they do not respond to external signaling. Determined **cells** no longer secrete FGF8 or Wnt3a. Our results are not significantly influenced by discontinuing the segmentation-clock oscillations or FGF8, Wnt3a and Delta/Notch signaling in determined **cells**.

(4) **pre_EphA4 cells** represent PSM cells that are fated to express high concentrations of membrane-bound EphA4 and to localize to the anterior compartment of the somite. They adhere weakly to all **cell types** in the **PSM** and **somites**, but adhere slightly more strongly to **EphA4 cells** and other **pre_EphA4 cells** than to **cells** of other **cell types**. They will differentiate into **EphA4 cells** upon reaching the differentiation wavefront.

(5) **pre_ephrinB2 cells** represent PSM cells that are fated to express high concentrations of membrane-bound ephrinB2 and to localize to the posterior compartment of the somite. They adhere weakly to all **cell types** in the **PSM** and **somites**, but adhere slightly more

strongly to **ephrinB2 cells** and other **pre_ephrinB2 cells** than to **cells** of other **cell types**. They will differentiate into **ephrinB2 cells** upon reaching the differentiation wavefront.

(6) **pre_Core cells** represent PSM cells that are fated to express high concentrations of stabilized N-CAM and N-cadherin and relatively low concentrations of EphA4 or ephrinB2 at their membranes, and to localize to the center of the somite. They adhere moderately to **Core cells** other **pre_Core cells**, and weakly to other **cell types** in the **PSM**. They will differentiate into **Core cells** upon reaching the differentiation front.

Cells with the **somitic cell types EphA4**, **ephrinB2** and **Core** represent cells in the PSM and somites that are anterior to the differentiation front. They do not have segmentation-clock oscillations, nor do they secrete FGF8 or respond to FGF8, Wnt3a or Delta/Notch signaling. Their adhesive properties, which drive somite formation and maintenance, differ drastically from those of other **cell types**.

(7) **EphA4 cells** represent cells with high concentrations of EphA4 at their membranes, which make up the anterior compartments of the somites. They adhere moderately to **Core cells** and other **EphA4 cells**, and strongly repulse **ephrinB2 cells**. They adhere weakly to **cell types** in the **PSM**, but adhere slightly more strongly to **pre_EphA4 cells** than to **pre_ephrinB2 cells** and **PSM cells**.

(8) **ephrinB2 cells** represent cells with high concentrations of ephrinB2 at their membranes, which make up the posterior compartments of the somites. They adhere moderately to **Core cells** and other **ephrinB2 cells**, and strongly repulse **EphA4 cells**. They adhere weakly to **cell types** in the **PSM**, but adhere slightly more strongly to **pre_ephrinB2 cells** than to **pre_EphA4 cells** and **PSM cells**.

(9) **Core** cells represent cells with high concentrations of stabilized N-cadherin and N-CAM and relatively low concentrations of EphA4 or ephrinB2 at their membranes, which make up the centers of the somites. **Core cells** adhere moderately to **EphA4** and **ephrinB2 cells**, and strongly to other **Core cells**.

Table 2.1 shows the relative degrees of adhesion and repulsion between **cells** of different **cell types** in our biological model.

Table 2.1: Strengths of adhesion and repulsion between model cell types.

Cell type	**Medium**	**Wall**	**Source**	**PSM**	**pre_EphA4**	**pre_ephrinB2**	**pre_Core**	**EphA4**	**ephrinB2**	**Core**
Medium	--	--	N	MR	MR	MR	MR	wr	wr	MR
Wall		--	MR	MR	MR	MR	MR	MR	MR	MR
Source			wa	wa	wa	wa	wa	wa	wa	wa
PSM				wa	wa	wa	wa	wa	wa	wa
pre_EphA4					wa	wa	wa	wa	wa	wa
pre_ephrinB2						wa	wa	wa	wa	wa
pre_Core							MA	wa	wa	MA
EphA4								MA	**SR**	MA
ephrinB2									MA	MA
Core										**SA**

N=Neutral; wa=Weak Adhesion; MA=Moderate Adhesion; **SA**=Strong Adhesion; wr=Weak Repulsion; MR=Moderate Repulsion; **SR**=Strong Repulsion; -- = not applicable.

Chapter 3

METHODS

3.1 CompuCell3D

We implemented our computational models as simulations using CompuCell3D (*CC3D*) (**http://compucell3d.org**), an open-source software package designed to simulate multi-cell Glazier-Graner-Hogeweg (*GGH*) (Glazier and Graner 1993) models of cell behaviors in conjunction with intracellular genetic-network or reaction-kinetic models and extracellular partial-differential-equation (*PDE*) models of tissue-level morphogen concentrations (Alber, Chen *et al.* 2006; Popławski, Shirinifard *et al.* 2008; Shirinifard, Gens *et al.* 2009).

3.1.1 Cell-based modeling and simulation modules

A cell-based approach to multi-scale modeling allows for a biologically- and physically-motivated strategy for connecting scales. As is the case for biological somitogenesis, the **cell** is the natural unit of integration in our composite model. We implemented our computational models as simulations using Python scripts containing custom modules written as classes: the great majority of these simulation modules perform tasks to do with individual **cells**, *e.g.*, iteration of subcellular segmentation-clock networks within **cells**, protein secretion by **cells** into the extracellular space, signal-sensing by **cells**, or the alteration of **cells'** characteristics or behaviors. Ultimately, **cell** behaviors connect phenomena at the molecular and tissue scales.

A CompuCell3D simulation class can assign properties or attributes to **cells** (***cell** attributes*) that are stored by the main CompuCell3D program and are callable from other modules. Examples of

cell attributes range from Boolean variables indicating whether certain **cell** behaviors are active to an entire implementation of a subcellular genetic or reaction-kinetic network.

A separate simulation configuration script (in either Python or CC3DML) registers the modules, defines **cell types** and default **cell-type**-dependent contact energies, designates chemical fields, sets GGH-related parameters and boundary conditions, and specifies initial conditions.

3.1.2 Glazier-Graner-Hogeweg model

The Glazier-Graner-Hogeweg (*GGH*) computational model (Swat, Hester *et al.* 2009) represents space as a regular lattice of sites (or pixels). A GGH *generalized cell* may represent a biological cell, a subcellular compartment, a cluster of cells, or a piece of non-cellular material or surrounding medium. The **cells** from our biological model are simulated as GGH generalized cells. Each generalized cell is an extended domain of sites on a *cell lattice* that share a common index (referred to as the *cell index*, σ).The cell-lattice configuration corresponds to an *effective energy* (*H*), defined so that simulated **cells** have the desired properties, behaviors and interactions, implemented via constraint terms in *H*. The effective energy in GGH simulations is not the actual energy of the biological cells and tissue being modeled but a simple way to specify the factors that govern **cell** properties, behaviors and dynamics in the simulated biological model. In our biological model, **cells** have volumes and surface areas, and interact via adhesion and repulsion, so that *H* is given by the following equation:

$$H = \sum_{\substack{\vec{i},\vec{j}\\ \text{neighbors}}} J\left(\sigma(\vec{i}),\sigma(\vec{j})\right)\left(1-\delta\left(\sigma(\vec{i}),\sigma(\vec{j})\right)\right) + \sum_{\sigma}\left[\lambda_{\mathrm{vol}}(\sigma)\left(v(\sigma)-V_{\mathrm{t}}(\sigma)\right)^2 + \lambda_{\mathrm{surf}}(\sigma)\left(s(\sigma)-S_{\mathrm{t}}(\sigma)\right)^2\right]$$

(Equation 3.1)

The first sum, over all pairs of neighboring lattice sites $\vec{i}$ and $\vec{j}$, calculates the *boundary* or *contact energy* between neighboring **cells**. $J(\sigma(\vec{i}),\sigma(\vec{j}))$ is the boundary energy per unit contact area for **cells** $\sigma(\vec{i})$ and $\sigma(\vec{j})$ occupying sites $\vec{i}$ and $\vec{j}$, respectively, and the delta function restricts the contact-energy contribution to **cell**-**cell** interfaces. We specify $J(\sigma(\vec{i}),\sigma(\vec{j}))$ as a matrix according to the **cell types** of $\sigma(\vec{i})$ and $\sigma(\vec{j})$. Higher (more positive) contact energies between **cells** result in greater repulsion between the **cells** and lower (more negative) contact energies between **cells** result in greater adhesion between the **cells**.

The second sum in (Eq. 3.1), over all **cells**, calculates the effective energies due to the volume and surface-area constraints. Deviations of the volume or surface area of **cell** σ from its target values ($V_{\mathrm{t}}(\sigma)$ or $S_{\mathrm{t}}(\sigma)$, respectively), increase the effective energy, penalizing these deviations. On average, a **cell** will occupy a number of pixels in the cell lattice slightly smaller than its target volume due to surface tensions from the contact energies (J). The parameters λ_{vol} and λ_{surf} behave like Young's moduli, with higher values reducing fluctuations of a **cell**'s volume or surface area about its target values.

Cell dynamics in the GGH model provide a much simplified representation of cytoskeletally-driven cell motility using a stochastic modified Metropolis algorithm consisting of a series of index-copy attempts. Before each attempt, the algorithm randomly selects a target site, $\vec{i}$, and a neighboring source site, $\vec{i}\,'$. If different **cells** occupy those sites, the algorithm sets $\sigma(\vec{i}) = \sigma(\vec{i}\,')$ with probability $P(\sigma(\vec{i}) \to \sigma(\vec{i}\,'))$, given by the Boltzmann acceptance function:

$$P(\sigma(\vec{i}) \to \sigma(\vec{i}\,')) = \begin{cases} 1 & : \quad \Delta\mathrm{H} \le 0 \\ e^{-\frac{\Delta H}{T_{\mathrm{m}}}} & : \quad \Delta\mathrm{H} > 0 \end{cases}, \qquad \text{(Equation 3.2)}$$

where ΔH is the change in the effective energy if the copy occurs and T_{m} is a global parameter describing cell-membrane fluctuations that we will discuss momentarily. A Monte Carlo Step (*MCS*) is defined as *N* index-copy attempts, where *N* is the number of sites in the cell lattice, and sets the natural unit of time in the computational model.

The average value of the ratio $\Delta H / T_{\mathrm{m}}$ for a given **cell** determines the amplitude of fluctuations in the **cell** boundaries that are a simplified representation of the cytoskeletal fluctuations that drive cell motility. High $\Delta H / T_{\mathrm{m}}$ results in rigid, barely- or non-motile **cells** and little **cell** rearrangement. For low $\Delta H / T_{\mathrm{m}}$, large fluctuations allow a high degree of **cell** motility and rearrangement. For extremely low $\Delta H / T_{\mathrm{m}}$, **cells** may fragment in the absence of a constraint sufficient to maintain the integrity of the borders between them. Because $\Delta H / T_{\mathrm{m}}$ is a ratio, we can achieve appropriate **cell** motilities by varying either T_{m} or ΔH. Variations in T_{m} allow us to explore the impact of global changes in cytoskeletal fluctuations (*e.g.*, to mimic an experiment using cytochalasin). By changing ΔH, we can influence the relative motility of the **cell types** or of individual **cells** by varying, for example, the parameter λ_{surf}, the target surface areas (S_{t}) or the contact energies (J) between **cells**.

The Metropolis algorithm evolves the cell-lattice configuration to simultaneously satisfy the constraints, to the extent to which they are compatible, with perfect damping (*i.e.*, average velocities are proportional to applied forces).

A potential index copy that increases the effective energy, *e.g.*, by increasing deviations from target values for **cell** volume or surface area or juxtaposing mutually repulsive **cells**, is improbable. Thus, the pattern evolves in a manner consistent with the biologically-relevant "guidelines" incorporated in the effective energy: **cells** maintain surface areas and volumes close to their target values, mutually adhesive **cells** (with low **cell**-**cell** contact energy) stick together, mutually repulsive **cells** separate, *etc*... Thus, the average time-evolution of the cell lattice corresponds to that achievable deterministically using finite-element or center-model methodologies with perfect damping.

3.1.2.1 GGH cell types

A GGH **cell type** distinguishes **cells** that share a unique set of behavioral mechanisms, parameters and submodels. Same-**type cells** may have additional parameters and variables which differ between **cells** of that **type**.

3.1.2.2 Cell motility

In contrast to many other models (Tiedemann, Schneltzer *et al.* 2007; Armstrong, Painter *et al.* 2009; Uriu, Morishita *et al.* 2009; Uriu, Morishita *et al.* 2010), our **cells** have explicit shapes and degrees of movement. We take advantage of the latter to study the effect of cell motility in somitogenesis. In our model we vary the motility of **cells** by varying the degree of **cell** membrane fluctuation, which is regulated by the parameter λ_{surf} (larger λ_{surf} leads to higher average ΔH, which reduces **cell** motility). This choice has a biological motivation; motility in biological cells is associated with the degree of membrane ruffling and higher λ_{surf} decreases the **cell**-boundary ruffling amplitude (Mombach, Glazier *et al.* 1995).

3.1.2.2 Time and length scales

The natural length and time scales in GGH computational models and simulations are pixels and MCS, respectively. We relate these to biologically-relevant units in such a way that events in time and space in our model correspond to those *in vivo*. Specifically, we use **cell** diffusion, morphogen diffusion, **cell** diameter, **somite** size and the segmentation-clock period to convert between “model time” and “biological time” in the following way: 1 pixel in our simulations corresponds to 1.43 µm (1 µm = 0.7 pixels) and 1 MCS corresponds to 0.015 min (6000 MCS = 90 min, the duration of one somite cycle).

3.1.3 Chemical fields

Chemical fields in CompuCell3D simulations evolve due to secretion and absorption by **cells**, and diffusion and decay in the extracellular medium, according to partial differential equations (*PDEs*). The standard CompuCell3D *PDE* solver, used in this work for the FGF8 field, updates fields at a user-defined interval in MCS using forward-Euler integration.

3.2 Mathematical and computational submodels of somitogenesis

3.2.1 Model GGH cell types

The **cell types** in our computational model correspond to those in our biological model. To develop our computational model, we assigned to each **cell type** GGH parameters for target volume (or surface area in two dimensions), target surface area (or boundary length in two dimensions), and volume and surface-area constraint parameters λ_{vol} and λ_{surf}. **Table 3.1** gives these parameters; **Table 3.2** lists other characteristics of our computational-model **cell types**.

Table 3.1: Parameters for model cell types.

Cell type	Diameter (µm)	Surface (µm)*	λ_{surf}
Medium	--	--	--
Wall	10	40	--
Source	10-20	40-80	15
PSM	10	40	15
pre_EphA4	10	40	15
pre_ephrinB2	10	40	15
pre_Core	10	40	15
EphA4	10	40	15
ephrinB2	10	40	15
Core	10	40	15

*In a 2D model, the Surface parameter is a cell boundary length with units of length, not area.

Table 3.2: Model cell behaviors.

Cell type	Diffusion constant (μm^2/min)	Grow/ Divide (Yes/No)	Secrete FGF8 (Yes/No)	Clock network (Yes/No)	Delta signal (Yes/No)	Respond to Delta signal (Yes/No)
Medium	--	--	--	--	--	--
Wall	--	N	N	N	N	N
Source	--	Y	Y	Y	Y	Y
PSM	1.08	N	Y	Y	Y	Y
pre_EphA4	1.01	N	Y	N	Y	N
pre_ephrinB2	1.01	N	Y	N	Y	N
pre_Core	0.98	N	Y	N	Y	N
EphA4	1.02	N	N	N	N	N
ephrinB2	1.02	N	N	N	N	N
Core	0.95	N	N	N	N	N

We also specified the *contact-energy matrix*, in which we designated the GGH contact energies that represent the adhesive and repulsive interactions between **cell types** (**Table 3.3**). We estimated GGH contact energies to approximate the relative adhesion and repulsion strengths between biological cells with different concentrations of adhesion molecules at their membranes. While we did not perform an exhaustive search over all possible contact energies, a modest exploration of contact energies did not significantly affect the resulting **cell** behaviors, provided that the contact energies maintained the hierarchy shown in **Table 3.3**.

Table 3.3: GGH contact energies between cell types for reference simulation.

Cell type	Medium	PSM	Source	pre_EphA4	pre_ephrinB2	pre_Core	EphA4	ephrinB2	Core	Wall
Medium	0	15	0	15	15	15	5	5	15	0
PSM		-20	-20	-20	-20	-20	-20	-20	-20	30
Source			-20	-20	-20	-20	-20	-20	-20	30
pre_EphA4				-25	-20	-20	-25	-20	-20	30
pre_ephrinB2					-25	-20	-20	-25	-20	30
pre_Core						-35	-20	-20	-20	30
EphA4							-25	80	-25	30
ephrinB2								-25	-25	30
Core									-40	30
Wall										--

Positive contact energies represent repulsive interactions; negative contact energies represent adhesive interactions. Larger contact energy magnitudes indicate stronger interactions.

3.2.2 Presomitic mesoderm growth

In our model, **PSM** extension occurs solely due to the addition of **cells** from the posterior (as opposed to cell division or cell addition from other sources). To represent the steady addition of cells to the modeled PSM region, we defined a layer of non-biological **Source cells** at the posterior end of the modeled **PSM** that grow and divide at a constant rate to produce new **PSM cells**.

3.2.3 Establishment of morphogen gradients

Source and **PSM cells** in the biological model secrete FGF8 and Wnt3a proteins that diffuse and decay in space. In the computational model, each **Source** and **PSM cell** has an internal concentration of *fgf8* mRNA (attached as a **cell** attribute in CC3D) that determines the **cell**'s FGF8 secretion rate. **Source cells** have a constant concentration of *fgf8* mRNA, $mfgf_0$, that **PSM cells** inherit from their parent **Source cells**. In **PSM cells**, *fgf8* mRNA decays exponentially in time with a decay constant k_{mfgf}:

$$\frac{d}{dt} mfgf_{\mathrm{cell}} = -k_{\mathrm{mfgf}} \cdot mfgf_{\mathrm{cell}} \quad . \qquad \text{(Equation 3.3)}$$

Figure 3.1 (**A**) shows the evolving cellular *fgf8* mRNA concentration averaged in the ML direction versus AP position.

In vivo, cells translate *fgf8* mRNA into FGF8 protein before secreting FGF8 into the intercellular space, where it binds to receptors on that and other cells' membranes to induce the intracellular FGF signaling cascade. We simplified this process in our computational model, first by setting

PSM and **Source cells**' FGF8 secretion rates directly proportional to their intracellular concentrations of *fgf8* mRNA:

$$S(\vec{\mathrm{x}}) = s_{\mathrm{fgf}} \cdot mfgf_{\mathrm{cell}(\vec{\mathrm{x}})} \quad , \qquad \text{(Equation 3.4)}$$

where $\vec{\mathrm{x}}$ is a field-lattice site corresponding to a cell-lattice site occupied by the **cell**, and each **PSM** and **Source cell** secretes FGF8 from every pixel it occupies. Second, **cells** in our model do not impede diffusion (**cells** and FGF8 co-occupy space), nor is FGF8 consumed during signaling, so that the local FGF8 concentration obeys the two-dimensional diffusion equation with secretion as specified in Equation 3.4:

$$\frac{\partial}{\partial t}\mathrm{FGF8} = D_{\mathrm{fgf}}\nabla^2\mathrm{FGF8} - k_{\mathrm{fgf}}\mathrm{FGF8} + S(\vec{\mathrm{x}}) \quad . \qquad \text{(Equation 3.5)}$$

Finally, we simplified FGF8 signaling in our mathematical and computational models by not modeling the interaction between extracellular FGF8 and cellular transmembrane FGF receptor proteins. Biologically, cells in the PSM generally express FGFR1 (Wahl, Deng *et al.* 2007); in constructing our computational model, we assumed that intracellular FGF signaling is proportional to the local FGF8 concentration and is not affected by the concentration of FGFR1 on a cell's surface. Because the intracellular segmentation clock is the only cellular property influenced by FGF8 in the posterior **PSM**, where FGF8 concentration is very high, and because the model clock's response to FGF8 saturates at a very low FGF8 concentration, possible effects of FGF receptor saturation are not a matter of concern. In our computational model, the FGF8 signaling experienced by a **cell** is the FGF8 concentration at the **cell**'s geometric center.

At the simulation level, a custom simulation class handles *fgf8* mRNA decay and **cell**-by-**cell** secretion, and the basic CC3D diffusion solver handles FGF8 diffusion and decay outside of **cells** every 16 MCS (because FGF8 varies smoothly and diffuses slowly, the simple CompuCell3D PDE solver is appropriate). Because FGF8 has a diffusion length shorter than a **cell** diameter (for the diffusion and decay constants estimated in (Goldbeter, Gonze *et al.* 2007)) and the FGF8 concentration field has decayed to zero near the simulation boundaries, the choice of global simulation boundary conditions does not significantly affect simulation results. FGF8 diffuses freely (**cells**, including **Wall** and **Medium**, do not impede diffusion) with Neumann boundary conditions[2]. **Figure 3.1** (**B**) shows the FGF8 field for a typical unperturbed simulation.

We used the FGF8 diffusion and decay constants estimated by Goldbeter *et al.* (Goldbeter, Gonze *et al.* 2007). We chose the initial intracellular *fgf8* mRNA concentration and *fgf8* mRNA decay rate, which determine the amplitude and shape of the AP FGF8 gradient, and the FGF8 concentration for **cell** determination to position the determination front roughly eight **somite** lengths anterior to the **Source cells**. Parameters governing the morphogen fields are presented in **Table 3.4**. Our model produces roughly exponential morphogen gradients, reflecting observations *in vivo* (Dubrulle and Pourquie 2004).

Cells carry their *fgf8* mRNA concentrations with them and secrete FGF8 at all sites within the **cell**, so the FGF8 source term (Wahl, Deng *et al.* 2007) in the apparently deterministic diffusion equation (Eq. 3.5) is stochastic, reflecting the stochasticity of **cell** motion and of the addition of **cells** to the **PSM** by the daughter **cells** of **Source cells**. Thus, the *fgf8* mRNA and FGF8 fields are

[2] Neumann boundary conditions specify the value of the spatial derivative of the field at the boundaries; in our case, we have set the value of the derivative to zero at all boundaries. Again, this choice of boundary conditions did not impact our results.

noisy. Noise in the *fgf8* mRNA and FGF8 fields increases with increasing **cell** motility, as expected, though diffusive smoothing decreases the noise in the FGF8 field compared to that of the *fgf8* mRNA field for all **cell** motilities (**Table 3.5**). Notably, the noise in *fgf8* expression in the simulated **PSM** is comparable to the salt-and-pepper pattern of *fgf8* expression in the *in vivo* PSM reported by Dubrulle *et al.* (Dubrulle and Pourquie 2004).

Table 3.4: Parameters for FGF8 and Wnt3a fields.

Parameter	**Value**
D_{FGF8}	0.6 μm^2/min
k_{FGF8}	0.2 min^{-1}
$mfgf_0$	5.0 nM
k_{mfgf}	0.005 min^{-1}
s_{fgf}	1.83 min^{-1}
C_{f2w}	0.32

Table 3.5: Noise in FGF8 and *fgf8* mRNA fields for different cell motilities.

	Low cell motility	**Reference cell motility**	**High cell motility**
Noise in [*fgf8* mRNA]	3.49 %	4.78 %	8.50 %
Noise in [FGF8]	2.11 %	3.96 %	6.38 %

We calculated noise as the standard percent deviation of the simulation data from the best-fit exponential function averaged over all times. Low motility: λ_{surf} = 25, D = 0.86 μm^2/min. Reference motility: λ_{surf} = 15, D = 1.08 μm^2/min. High motility: λ_{surf} = 5, D = 1.76 μm^2/min.

Finally, I reiterate that the AP gradient arises from **cell**-autonomous intracellular *fgf8* mRNA decay rather than diffusion of FGF8, which has a diffusion length much smaller than the length of the tissue, a phenomenon that has developmental and evolutionary implications that I will discuss in Sections 4.3 and 4.4.

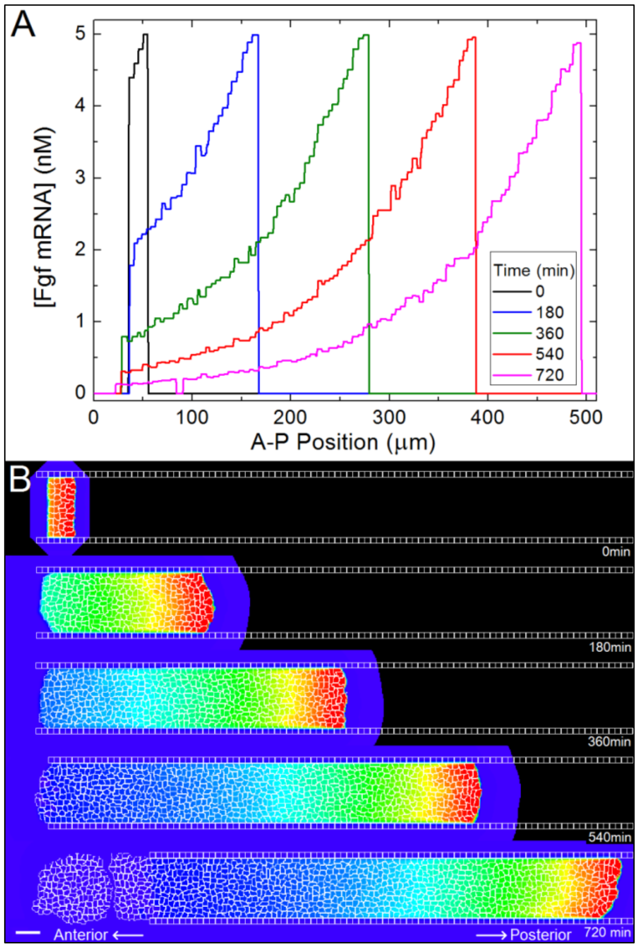

Figure 3.1: Typical FGF8 evolution of morphogen gradients in simulated PSM. (**A**) *fgf8* mRNA concentration along the A-P centerline of the simulated PSM at 0, 180, 360, 540 and 720 min. (**B**) FGF8 concentrations at the same times. The color scale is the same as in **Figure 3.6** (red corresponds to 45 nM and blue to 0 nM). Anterior to left. Direction of **PSM** growth to right (posterior). Scale bar 40 μm. Parameter values: D_{FGF8} = 0.6 μm^2/min; k_{FGF8} = 0.2min^{-1}; $mfgf_0$ = 5.0 nM; k_{mfgf} = 0.005 min^{-1}; s_{fgf} = 1.83 min^{-1}; C_{f2w} = 0.32; PSM growth rate = 1.63 μm/min.

To reduce computation time and because no experimental evidence suggests a more complex Wnt3a profile, we simplified our biological model of independent Wnt3a decay by setting each **cell**'s Wnt3a concentration proportional to its level of *fgf8* mRNA:

$$Wnt3a_{\mathrm{cell}} = C_{\mathrm{f2w}} \cdot mfgf8_{\mathrm{cell}} \quad . \qquad \text{(Equation 3.6)}$$

We neglected Wnt3a secretion and diffusion both for computational simplicity and because its very short diffusion length (Christian 2000; Aulehla, Wehrle *et al.* 2003) would effectively restrict it to the secreting **cell**'s immediate neighborhood. We made a similar simplification regarding Wnt3a to the one we made for FGF8: we took Wnt3a signaling to be proportional to the cellular concentration of Wnt3a, ignoring signal-receptor interaction and Wnt3a receptor saturation.

3.2.4 Segmentation-clock network

Goldbeter and Pourquié translated their biological segmentation-clock model into a mathematical model comprised of a set of ordinary differential equations (*ODE*s). In our mathematical model of the segmentation clock, we modified the ODEs of Goldbeter and Pourquié's mathematical segmentation-clock model to reflect our changes to their biological model (see **Figure 2.2**). Appendix A, SEGMENTATION CLOCK NETWORK SUBMODEL EQUATIONS, presents the full set of ODEs in our segmentation-clock mathematical submodel, and draws attention to the modifications we made to the original Goldbeter-Pourquié equations.

We neglected the numerous sources of intracellular fluctuations in real biological reaction networks, some of which we could implement in future at the computation level, *e.g.*, using a Gillespie method, both because they are computationally expensive to simulate and because the stochastic GGH model creates stochastic fluctuations that are large compared to the errors due to the ODE approximation. The lowest molecular concentrations dealt with in our extended clock model are of the order of 10^{-4} nM, corresponding to an average intermolecular distance of about 0.25 µm, two orders of magnitude lower than the average diameter of PSM cells (10 µm). In addition, such low concentrations occur for only a few molecular species and during only a fraction of the clock period, so we need not model the clock using stochastic methods at our current level of detail (Grima and Schnell 2008).

At the simulation level, we wrote a C++ class (**Oscillator**) to integrate the segmentation-clock network equations in each **cell** (Appendix C, SOURCE CODE). **Oscillator** stores current values for the molecular species and uses a fourth-order Runge-Kutta solver to integrate the equations for given values of local FGF8 signaling, local Wnt3a signaling and juxtacrine Delta signaling at each time step. We Python-wrapped **Oscillator** using *Simplified Wrapper and Interface Generator*, or *SWIG* (**http://www.swig.org**), to make it accessible in Python. Within a custom CC3D simulation class, we attached an instance of **Oscillator** to each **PSM cell** as a **cell** attribute. The class handles inputs from each **cell's** local environment (including surrounding **cells**) to the **cell's** instance of **Oscillator**, integrating the network equations and storing the values of the oscillating molecular species' concentrations for access by other simulation classes.

In a simulation of a single self-coupled **cell** (*i.e.*, the **cell** perceives incoming Delta signaling equal to its outgoing Delta signaling), our computational segmentation-clock submodel produces oscillations in Lfng, Axin2 and Dusp6 with the qualitative phase relationships seen *in vivo* (**Figure 3.2** (**A**)).

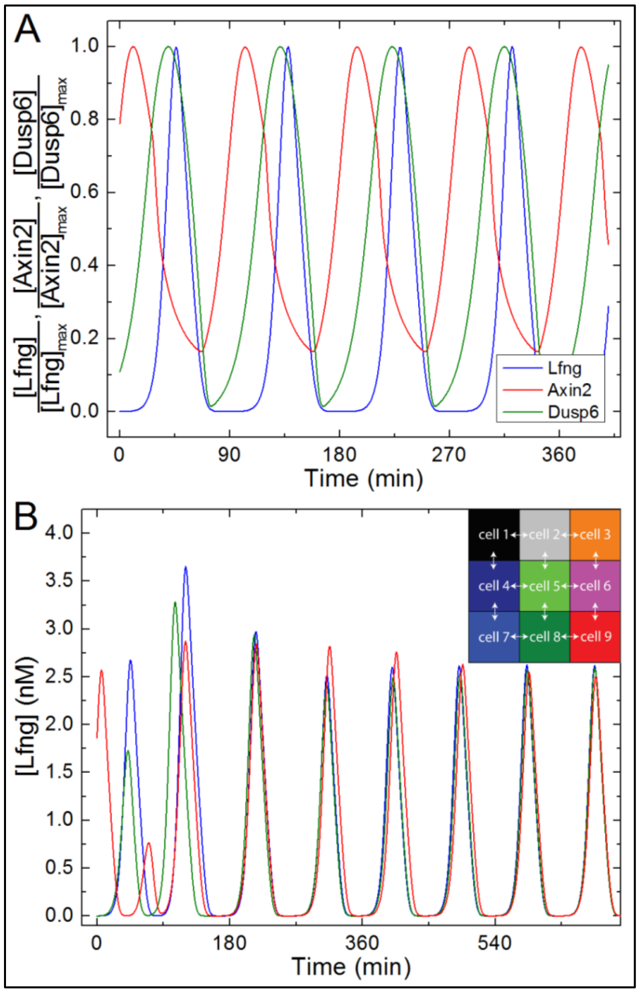

Figure 3.2: Simulated segmentation-clock behavior. (**A**) Normalized Lfng, Axin2 and Dusp6 concentrations in a single **cell** for the network shown in **Figure 2.2**. The **cell** is self-coupled, *i.e.*, its incoming Delta signal is set equal to its outgoing Delta signal, to reproduce the behavior of a **cell** in a neighborhood of **cells** of the same segmentation-clock phase. (**B**) Lfng concentration in nine coupled **cells** with the on-diagonal **cells** initially displaced in phase by 40%. After two segmentation-clock periods, the oscillations phase-lock (the time-averaged standard deviation over each subsequent cycle is less than 6% of the average of the amplitudes after the first two periods). Parameter values used are listed in Appendix B, SEGMENTATION CLOCK NETWORK SUBMODEL PARAMETERS AND INITIAL CONDITIONS.

3.2.4.1 Coupling neighboring cells' segmentation clocks

Each time the **Oscillator** class in each **cell** integrates the segmentation-clock equations, it takes as an input the value of Delta signaling from surrounding **cells**. The value of Delta signaling is calculated by a loop over the **cell's** immediate neighbors that sums total membrane-bound Delta from all neighbors and divides that value by the number of neighbors (*i.e.*, the value of Delta signaling is taken to be the average of the neighbors' ability to Delta signal). When we simulated multiple segmentation-clock networks with different initial phases coupled via Delta/Notch signaling, they phase locked to the same phase, while maintaining the desired intracellular FGF-Wnt-Notch phase relationships (**Figure 3.2** (**B**)).

3.2.4.2 Coupling the segmentation clock to the morphogen fields

At the biological level, FGF8 and Wnt3a interact with the FGFR1 and Frizzled receptors, respectively, leading to intracellular signal transduction in the FGF and Wnt pathways and driving the segmentation clock within a cell (see **Figure 2.2**). Thus, the local concentrations of FGF8 and Wnt3a potentially affect the amplitudes and/or oscillation periods of cells' segmentation clocks.

In our mathematical submodel of segmentation-clock-morphogen interaction, we simplified Wnt3a signaling by assuming that the concentration of disheveled (*Dsh*), which interferes with β-catenin phosphorylation as a downstream effect of signaling through Frizzled, is proportional to the degree of Wnt3a signaling. We further simplified Wnt3a and FGF8 signaling at the computational level, where we took the degree of FGF8 signaling within a **cell** to be equal to the

FGF8 concentration at the center of the **cell** and the degree of Wnt3a signaling to be cell-autonomous.

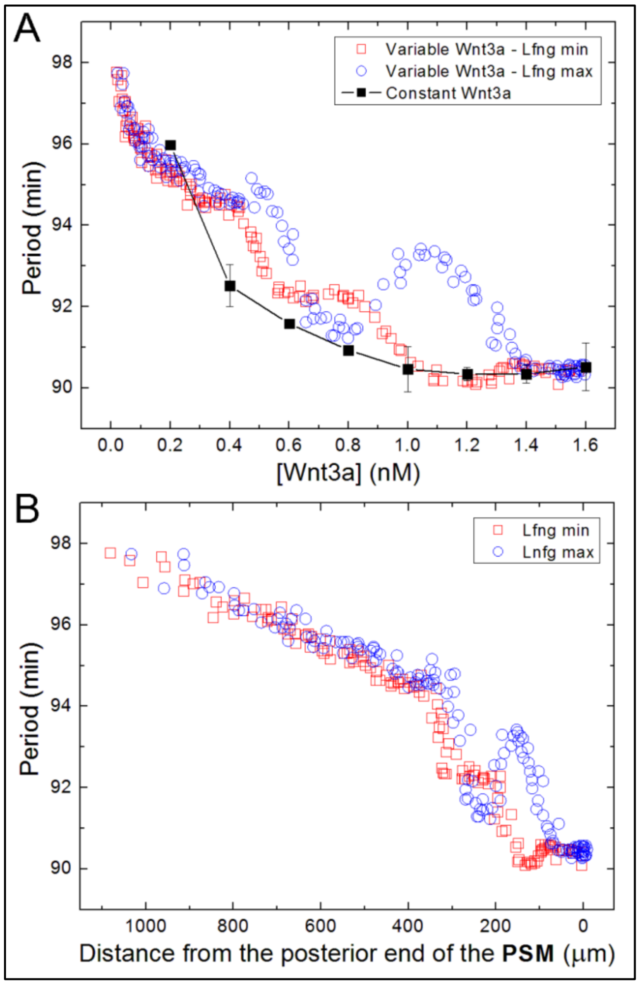

Figure 3.3: Segmentation-clock period versus Wnt3a concentration in simulated PSM. (**A**) Segmentation-clock period versus Wnt3a concentration in the simulated **PSM** (red squares and blue circles) and for **cells** with a constant Wnt3a concentration (connected black squares with error bars). (**B**) Segmentation-clock period as a function of **cell** position along the AP axis, measured by the anterior distance from the posterior (right) end of the simulated **PSM**. Slower oscillations in the anterior (left) simulated **PSM** are consistent with similar observations *in vivo* (Gomez, Ozbudak et al. 2008). Red squares indicate the period measured between times of minimum Lfng concentration and blue circles indicate the period measured between times of maximum Lfng concentration. Parameters are the same as in the reference simulation (**Figure 4.1**).

We found that the period of segmentation-clock oscillations increases with decreasing Wnt3a both in the simulated **PSM** and in sets of simulations of **cells** exposed to different, but constant, concentrations of Wnt3a (**Figure 3.3** (**A**)). In the case of the simulated **PSM**, this effect results in an anteriorly-increasing segmentation-clock period (**Figure 3.3** (**B**)), consistent with observations *in vivo* of segmentation-clock oscillations slowing within cells as they approach the anterior of the PSM (Gomez, Ozbudak et al. 2008; Gibb, Zagorska et al. 2009). In the regime where the model segmentation-clock network produces stable oscillations (for all but very low FGF8 and/or Wnt3a concentrations), the segmentation-clock period is independent of the FGF8 concentration, which is also consistent with experimental observations that the segmentation-clock period appears FGF8-independent (Gibb, Zagorska *et al.* 2009).

3.2.5 Clock-wavefront readout

In our biological model, three proteins serve as intermediaries between the segmentation-clock-wavefront interaction and eventual expression of adhesion proteins prior to and during somite formation: cMeso (Mesp2), downstream of Notch; cytoplasmic β-catenin modulated by Wnt3a signaling; and Paraxis, downstream of Wnt3a/β-catenin signaling (see **Figure 2.3**). Because the exact regulation of Mesp2 and Paraxis is unknown, we did not explicitly mathematically or computationally model their expression. We correlated the activity of each one with the concentration of an oscillatory component within our segmentation-clock submodel that is plausibly under similar regulation. We correlated cMeso (Mesp2) activity with Lfng concentration, as both are downstream of active Notch signaling and both repress Notch signaling; similarly, we correlated Paraxis activity with Axin2 concentration because both are downstream targets of Wnt/β-catenin signaling (see **Figure 2.2** and **Figure 2.3**). We had two motivations for this strategy. It is consistent with our simplified biological submodel, and the choice of the Lfng/Axin2/β-catenin trio allowed us to take advantage of a convenient characteristic of the time-series behavior of the segmentation-clock submodel.

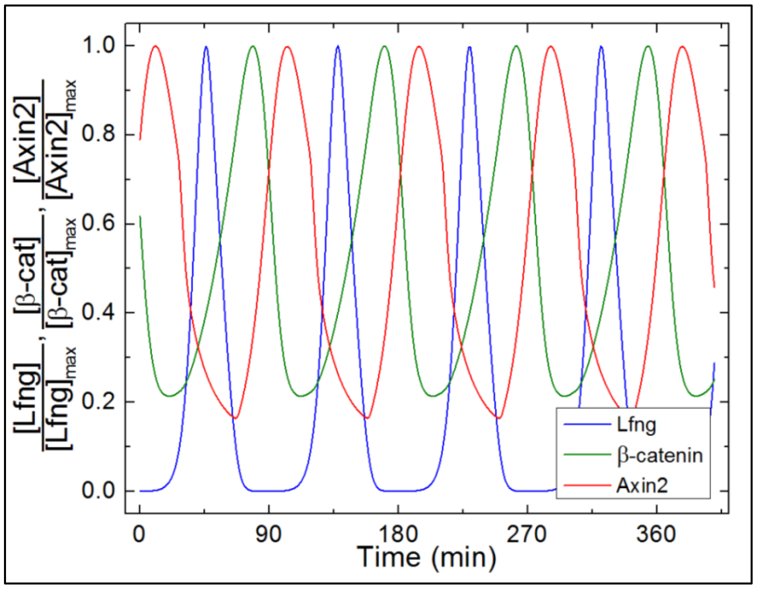

Figure 3.4: Segmentation-clock oscillations at the determination front. Time series of normalized determination-factor concentrations in a simulated **PSM cell** when [FGF8]=13.9 nM and [Wnt3a]=0.55 nM. When the external FGF8 concentration drops below the determination threshold of 13.9 nM, a **cell** decides its fate depending on both the relative and absolute concentrations of Lfng, β-catenin and Axin2. Segmentation-clock parameters are given in Appendix B, SEGMENTATION CLOCK NETWORK SUBMODEL PARAMETERS AND INITIAL CONDITIONS. The simulation shown is of a self-coupled **cell**, *i.e.*, the **cell** perceives incoming Delta signaling equal to its outgoing Delta signaling.

For external FGF8 and Wnt3a concentrations close to their values at the determination front, a **PSM cell**'s Lfng, Axin2 and β-catenin concentrations form temporally distinct peaks (**Figure 3.4**), allowing us to express our biological submodel of determination as a simple Boolean readout that assigns each **PSM cell** a determined **cell type** at the determination front (**Figure 3.5**). When the FGF8 concentration drops below the determination threshold, the readout algorithm determines whether the **cell** belongs in the core of the **somite** by comparing the concentration of β-catenin to a semi-arbitrary threshold for N-cadherin stabilization. Above this threshold, the **cell** chooses a **Core cell** fate; otherwise, the **cell** chooses a **peripheral cell** fate. **Peripheral cells** with [Lfng] > k_1[Axin2] choose anterior compartment (**EphA4**) fates, while those with [Lfng] ≤ k_1[Axin2] choose posterior compartment (**ephrinB2**) fates.

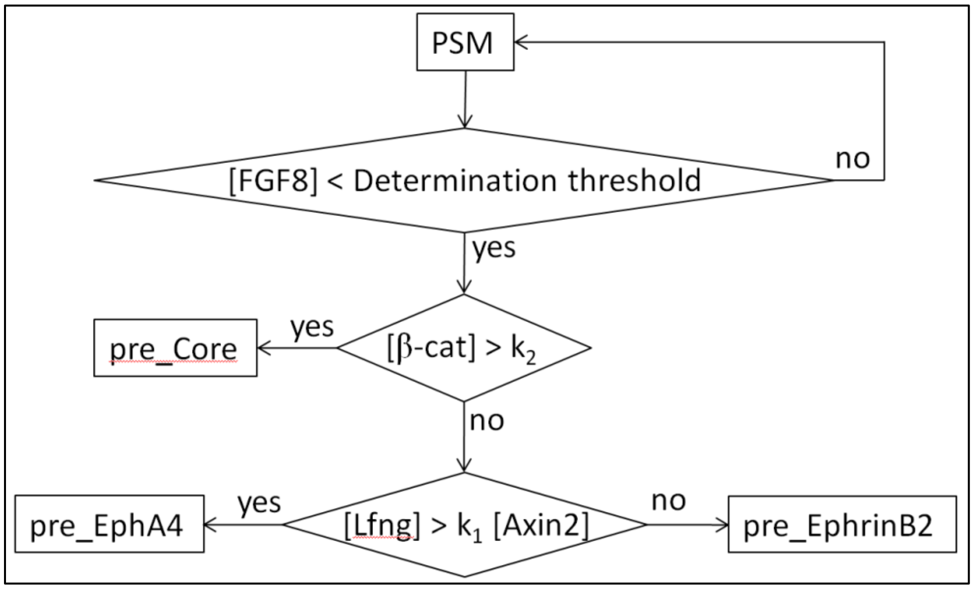

Figure 3.5: A schematic of the Boolean cell-type determination network submodel implemented in our computational model. When the external FGF8 concentration falls below the determination threshold, the relative and absolute concentrations of Lfng, β-catenin and Axin2 determine the fate of the **cell** in our computational model. The computational submodel is a simplified implementation of the biological submodel in **Figure 2.3**. k_1 = 21.28 and k_2 = 0.406 nM.

3.2.6 Differentiation

We implemented our simplified submodel of two-step differentiation by assigning new **cell types** to **cells** in the modeled **PSM** twice in the course of a simulation. At determination, we reassigned **PSM cells** to **pre_EphA4**, **pre_ephrinB2** or **pre_Core cell types** according to the criteria shown in **Figure 3.5**. Determined **cell types** have adhesion properties that are intermediate between **PSM** and **somitic cell types**. Later, at differentiation, the determined **cells** change their **cell types** to **EphA4**, **ephrinB2** or **Core**, which have the adhesion properties that drive **somite** formation and maintenance (see **Table 3.3**).

In the reference simulations (and where not otherwise stated), we implemented the cell-autonomous delay submodel of differentiation by attaching a “ticker” attribute to each **cell** at determination and incrementally increasing its value until four segmentation-clock periods had elapsed, at which time we assigned the **cell** a **somitic cell type**. Setting a second FGF8 concentration threshold to position the differentiation front four somite lengths anterior to the determination front in lieu of a differentiation “ticker” does not significantly alter results.

To explore the dynamic patterns of Lfng expression in the simulated **PSM**, we compared the cell-autonomous delay submodel to the positional-differentiation-front model. Instead of modeling the RA concentration field explicitly to implement the positional-differentiation-front submodel, we made the simplifying assumption that once the **PSM** reaches its full length, the differentiation front begins at the anterior tip of the **PSM** and advances at a constant speed equal to the **PSM** growth rate, maintaining the **PSM** at a constant length. We made this simplification because modeling the detailed mechanisms that position the differentiation front would not impact the simulated Lfng expression patterns at the level of detail we considered in our investigations.

3.2.7 Initial conditions

We modeled somitogenesis beginning after the formation of the first four somites, when the PSM has already grown to the length it will maintain through the subsequent formation of 22-24 additional somites. To avoid biasing the evolution of the model with a pre-imposed pattern, however, we initialized the model with only four layers of **cells** between two columns of confining **Wall cells** and allowed the **PSM** to grow to its full length from those initial conditions (**Figure 3.6**).

We initially defined two columns of **Wall cells** that run the length of the simulation and represent the medial and lateral structures confining the growth of the PSM, three layers of **PSM cells** spanning the ML space between the **Wall** columns and, posterior to them, a single layer of **Source cells**. We initialized the **Source cells** with concentrations of 5 nM *fgf8* mRNA and 45 nM FGF8, and the first three layers of **PSM cells** with 4.8 nM *fgf8* mRNA and 43.2 nM FGF8, 4.6 nM *fgf8* mRNA and 41.4 nM FGF8, and 4.4 nM *fgf8* mRNA and 39.6 nM FGF, moving from the posterior layer to the anterior layer, to emulate the progressive decay of *fgf8* mRNA and FGF8 in these **cells** after leaving the tailbud. We initialized the segmentation-clock networks within the **Source** and **PSM cells** with identical phases; we did not impose synchrony on the **Source cells**, but allowed them to interact with their neighbors. Our initial conditions result in the formation of a single ill-formed **somite** once the model **PSM** reaches its full length, after which normal **somites** form.

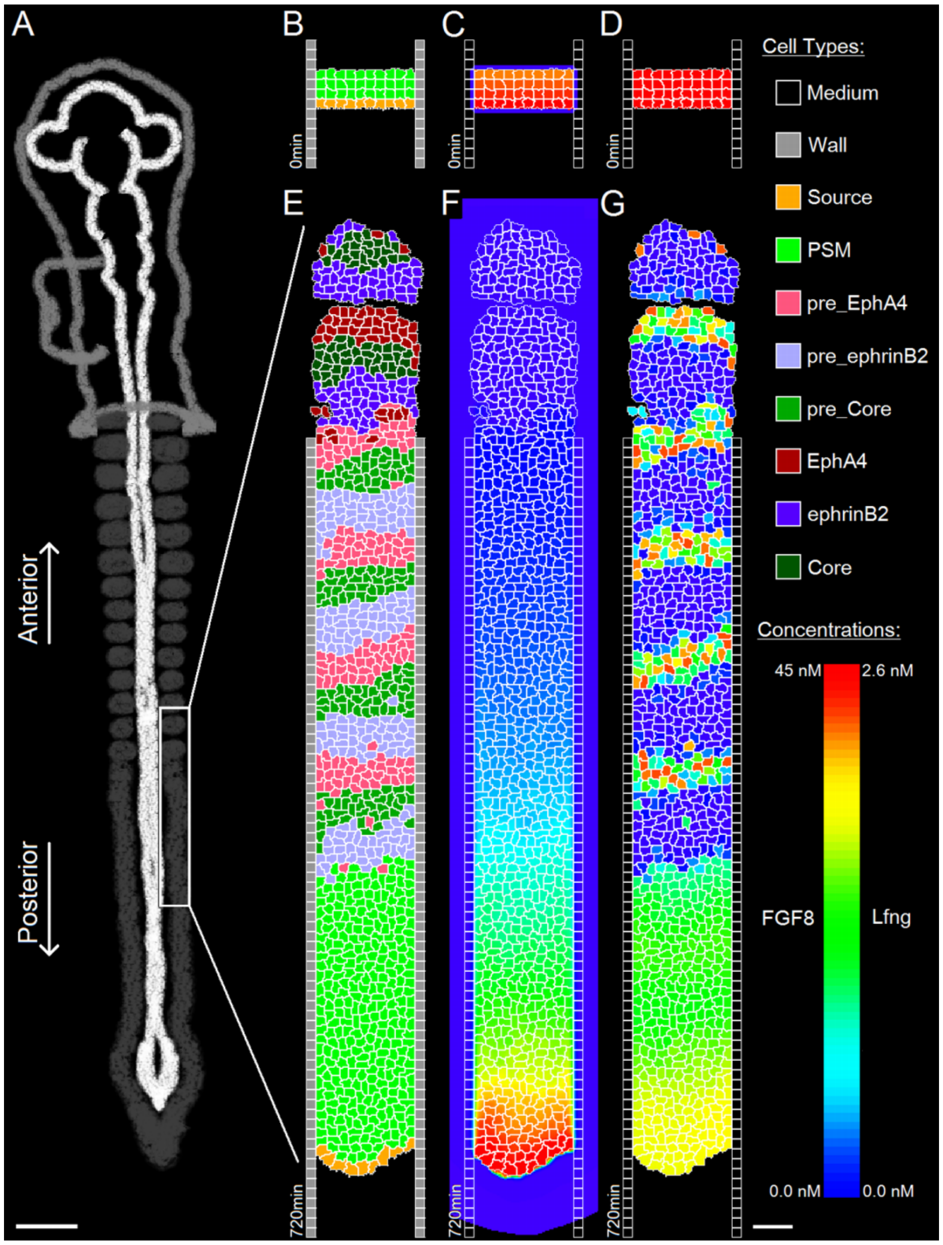

Figure 3.6: Initial conditions. (**A**) Sketch of an experimental image of a chick embryo at HH stage 10 (dorsal view). (**B**, **C**, **D**) Initial model conditions, visualizing: (**B**) **cell** types, (**C**) [FGF8] and (**D**) [Lfng]. Not shown: initially, the constraining **walls** extend the full AP length of the simulation. (**E**, **F**, **G**) The modeled **PSM** after reaching its full length (at 720 min), visualizing: (**E**) **cell** types, (**F**) [FGF8] and (**G**) [Lfng]. The patterns present in the full-length **PSM** arise spontaneously from the model's behavior. Parameters are the same as in the reference simulation (**Figure 4.1**). Scale bars: (**A**) 330 μm (**B**-**G**) 40 μm.

3.3 Perturbations

3.3.1 Altering the segmentation-clock period

We manipulated the period of the segmentation clock in our simulations by varying how frequently we updated the clock per unit time (*i.e.,* the number of clock iterations per MCS); we chose this method in order to predictably alter the segmentation-clock period without changing internal clock parameters or the clock's response to FGF8, Wnt3a or Delta/Notch signaling. Characteristics such as the clock's sensitivity to Wnt3a signaling and association constants between interacting clock components influence the clock period in a more biologically realistic way, but are less predictable in their additional influences on clock behavior.

3.3.2 Altering the PSM growth rate

We altered the **PSM** growth rate in a straightforward way: we increased the doubling time for the **Source cells**. Because the formation of the AP FGF8 gradient is tied to *fgf8* mRNA decay, altering the **PSM** growth rate also alters the rate at which the FGF8 determination front progresses (see **Figure 4.6**).

3.3.3 Altering PSM length

We allowed the length of the **PSM** to be determined in one of two ways, according to either the cell-autonomous or positional submodels for positioning the differentiation front.

In the case of the cell-autonomous differentiation-front submodel, the length of the **PSM** is determined by the distance between the posterior of the **PSM** where [FGF8] is at a maximum and the point in the anterior **PSM** where [FGF8] reaches the differentiation-threshold value (or some set distance/time after determination, if the ticker approach is implemented rather than the second threshold): a shallower FGF8 gradient will result in a longer **PSM**, and a steeper FGF8 gradient will result in a shorter **PSM**.

In the positional differentiation-front submodel, the length of the **PSM** is independent of the FGF8 gradient.

3.3.4 Altering cell motility

In chick, experimentally-observed cell motility is much greater in the tailbud and posterior PSM than in the more anterior regions of the PSM, and is graded from high in the posterior PSM to low in the anterior PSM (Benazeraf, Francois et al. 2010). We simplified our model by assuming that all **PSM cells** in the modeled region have the same motility. To examine the effect of global **cell** motility on **somite** formation in our model, we varied the parameter λ_{surf}, which changes the motility of the **cells**, as measured by their diffusion rate D_{cell} (**Table 3.6**). In all simulations, **PSM cells** subdiffuse, *i.e.*, their mean square displacement versus time $\left\langle r^2(\Delta t)\right\rangle = D_{\text{cell}}\Delta t^{\alpha}$, has an exponent α lower than that expected for a pure random walk (α=1).

Table 3.6: Measured diffusion of PSM cells for different values of λ_{surf}

λ_{surf}	**Cell diffusion constant (D_{cell})**	**Diffusion exponent (α)**
5	1.76 μm^2/min	0.91
15	1.08 μm^2/min	0.79
25	0.86 μm^2/min	0.73

Chapter 4

RESULTS

4.1 Reference simulations reproduce key features of somitogenesis *in vivo*

Simulations of our integrated model produce a single irregular **somite** (reflecting the initial conditions, see **Figure 3.6**) after the **PSM** first reaches its full length. Then, at the frequency of the segmentation clock (as measured in the posterior **PSM** where the frequency is greatest), our model forms an unlimited series of **somites** with consistent size, shape, and anterior, core and posterior compartments. **Figure 4.2** shows an image of a longer-than-typical simulation, demonstrating the reference model's ability to steadily produce an unlimited number of **somites**.

4.1.1 Morphology

Simulated (*in silico*) avian **PSM** and **somite** tissue morphologies closely resemble those *in vivo* (**Figure 4.1**). In both, somites are initially block-like and gradually round up as they mature. A gap that is narrow at the center line and more pronounced and notch-shaped at the medial and lateral edges separates adjacent somites. The notch widens as the somites mature.

Because our model omits epithelialization, ML asymmetry, recruitment of **Core cells** to the **somite** border and three-dimensional effects, the detailed organization and shape of **cells** in the simulated **somites** differ slightly from those for somitic cells *in vivo*. Segmentation, border formation and border maturation in the simulations, which include only adhesion-related changes in **cell** morphologies, however, closely resemble those *in vivo,* suggesting that our model captures the primary mechanisms of somitogenesis and that these omitted effects play more limited roles in the modeled stages of somitogenesis. Our model's ability to control and assay the importance of individual biological mechanisms is particularly useful for understanding the mechanisms of somitogenesis, because such control is lacking experimentally, *e.g.*, separating adhesion from epithelialization *in vivo* is difficult because Eph-ephrin signaling, which is responsible for cell-cell repulsion at the somite border, is also implicated in epithelialization (Watanabe, Sato *et al.* 2009).

In chick embryos, the cells belonging to the forming somite and the anterior PSM intermingle across the presumptive intersomitic border, so the intersomitic border does not initially form a smooth ML line. Kulesa and Fraser (Kulesa and Fraser 2002) reported that PSM cells at the ML edges and the center of the presumptive intersomitic border are initially several cell diameters anterior to the eventual intersomitic border, while presumptive somite cells between the ML edges and the center of the forming border are initially posterior to the eventual intersomitic border, forming a distinct "W" shape. As the border matures, the two cell populations separate from one another, causing the border to smooth and flatten (**Figure 4.1** (**A**-**F**)).

In our model, intrinsic noise due to **cell** motility initially generates **cell** mixing across the presumptive somite border. As a rule, this mixing is laterally homogeneous, sometimes resulting in the border shape described by Kulesa and Fraser (Kulesa and Fraser 2002) and sometimes resulting in other patterns. Adhesion-driven **cell** sorting and border smoothing follow determination, forming clear intersomitic gaps and rounded **somites** despite initial intermingling of **cells** across the border. In cases where the initial pattern of mixing across the border resembles that described by Kulesa and Fraser, the ensuing morphological events are also similar (**Figure 4.1** (**G**-**N**)), suggesting that, while our model does not incorporate all of the mechanisms responsible for the initial pattern of cell determination/differentiation at the presumptive intersomitic gap *in vivo*, it does include plausible mechanisms for producing the ensuing cell migrations.

Glazier *et al.* (Glazier, Zhang *et al.* 2008) studied the effect of noisy initial border specification on somitogenesis in an adhesion-driven model with six model **somitic cell types** and achieved similar results (Figures 8 and 9 of (Glazier, Zhang *et al.* 2008)). Whereas Glazier *et al.* specified noisy presumptive borders as an initial condition, in our model noise arises from **cell** mixing prior to differentiation. Our model's ability to reproduce realistic dynamic border morphology with only three **somitic cell types** in place of the six **somitic cell types** used by Glazier *et al.* indicates that adhesion-mediated cell sorting can explain somite gap formation and somite rounding even in the simplest case of repulsive anterior and posterior compartments and a highly adhesive core connecting the two within a somite, strengthening the argument for a differential-adhesion-driven model of somite formation.

One feature of our simulations that has not been extensively studied *in vivo* is the persistence of dislocated cell types well into somitogenesis (**Figure 4.1** (**P**)). This feature distinguishes cell-sorting-driven correction of "fuzzy borders" from possible mechanisms that either re-differentiate or kill misplaced cells, and thus stands as a prediction of the submodel of differential-adhesion-mediated cell-sorting as the primary border-correction mechanism.

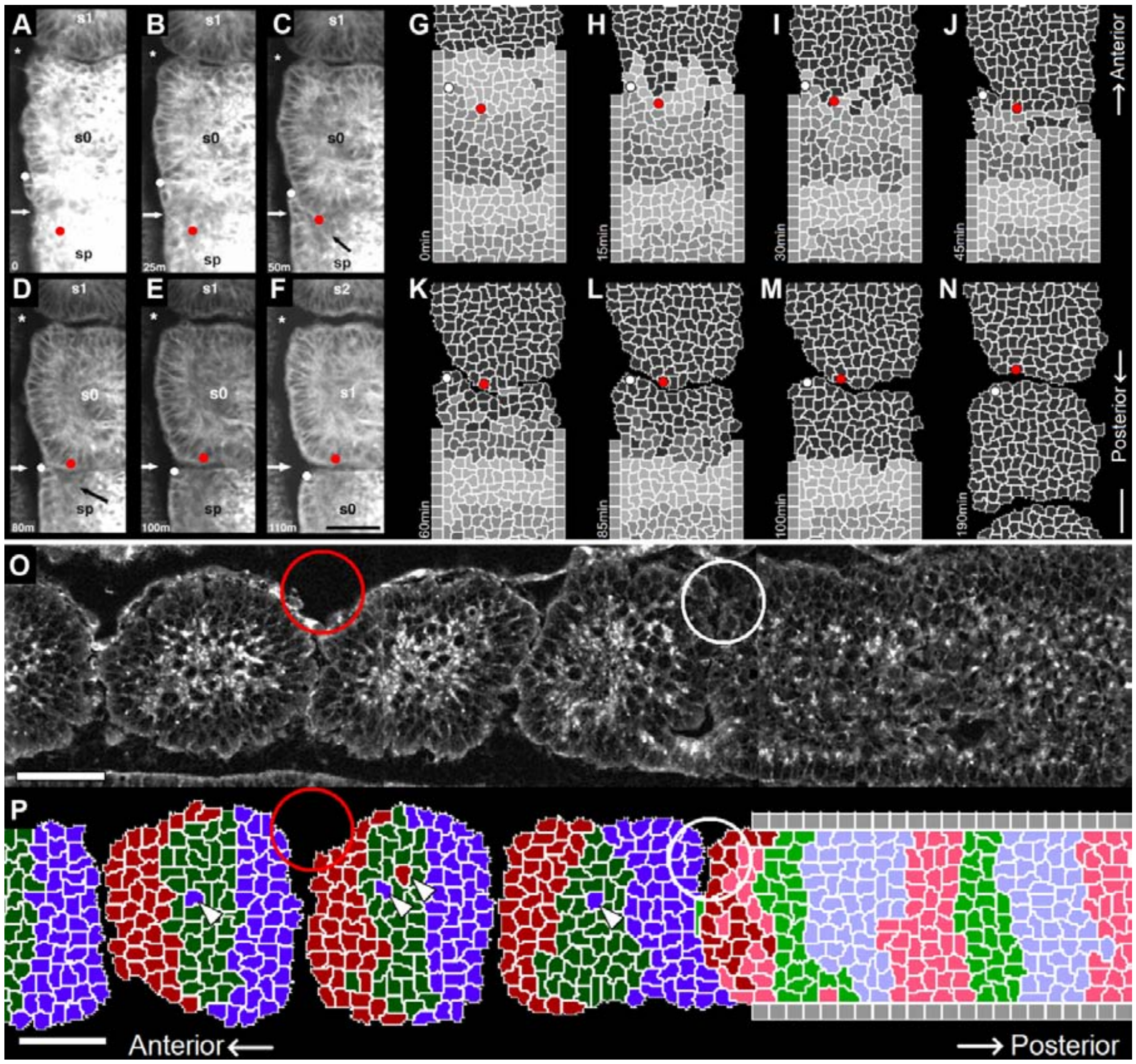

Figure 4.1: Comparison of reference simulation results with *in vivo* observations. (**A-F**) Experimental images from Kulesa and Fraser (Kulesa and Fraser 2002), taken at 0, 25, 50, 80, 100 and 110 minutes (reproduced with permission). Scale bar 50 μm. (**G-M**) Snapshots of a simulation reproducing the "ball and socket" morphology described by Kulesa and Fraser (Kulesa and Fraser 2002), taken at 0, 15, 30, 45, 60, 85, 100 and 190 minutes. Scale bar 40 μm. Initially, a "sleeve" of cells that will eventually be posterior to the forming border cradles presumptive somite cells that will eventually be anterior to the forming border (**A-C**, **G-J**). As the intersomitic border continues to develop, these two populations of cells move relative to each other to position themselves on the appropriate

sides of the border (**D-E**, **K-M**). The "sleeve" then retracts, leading to a rounded intersomitic border (**F**, **N**). The white and red dots in the simulations correspond to those in the experimental images. (**O**) Confocal image of one half of the PSM in a live chick embryo at HH Stage 10, stained with the cell-surface lipid label BODIPY ceramide. (**P**) Simulation detail at the corresponding time point. Simulated morphology closely resembles that observed *in vivo*, including the initially narrow gap separating adjacent somites (white circles), the block-like shape of the newly forming somite, the gradual rounding of more mature somites, and the resulting notch-like intersomitic clefts at the medial and lateral edges of maturing somites (red circles). Another notable feature of the simulation is the persistence of misplaced **cell** types after differentiation (white arrow heads). Model **cell type** colors are identical to those in **Figure 3.6**. Scale bars 50 μm. Reference simulation parameters: segmentation-clock period = 90 min; **PSM** growth rate = 1.63 μm/min; **Table 3.4** (FGF8 and Wnt3a); **Table 3.3** (cell-cell adhesion); **Table 3.1** and **Table 3.2** (**cell** sizes, motility and behaviors); and Appendix B, SEGMENTATION CLOCK NETWORK SUBMODEL PARAMETERS AND INITIAL CONDITIONS.

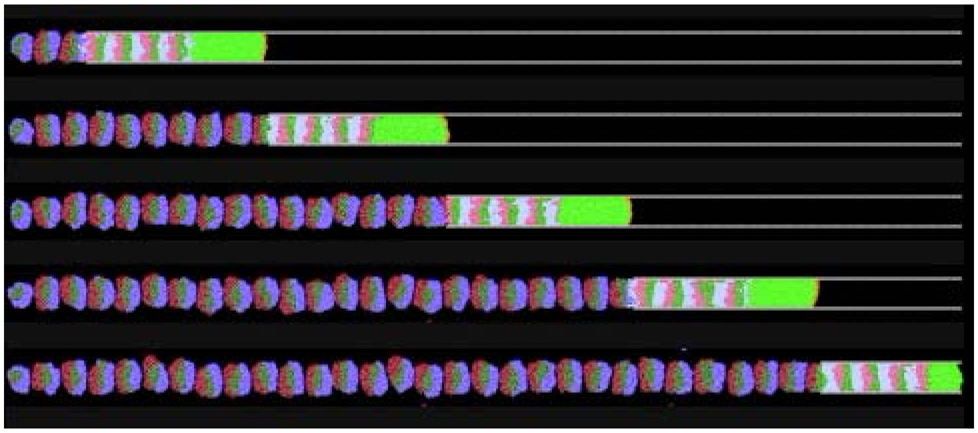

Figure 4.2: Simulations of the composite model with reference parameters make an indefinite number of somites. Anterior to the left. All parameters the same as in reference simulations (**Figure 4.1**).

4.1.2 Dynamic patterns of gene expression

In addition to realistic morphology, the model produces traveling stripes of Lfng expression that are similar to those observed *in vivo*. In the presence of a Wnt3a gradient, stripes of high Lfng concentration appear to form in the posterior **PSM** and travel in the anterior direction, narrowing in the AP direction as they do (**Figure 4.3**, **Figure 4.10** (**E**)). Lfng stripes do not occur for constant Wnt3a signaling, in either the presence or absence of an FGF8 gradient (see **Figure 4.10** (**D**)). We would expect this lack of response to variations in FGF8, given the insensitivity of the segmentation-clock period to the FGF8 concentration.

Our results are consistent with descriptions of the characteristic traveling stripes of gene expression as pseudo-waves arising from an AP gradient of the segmentation-clock periods in the PSM (Tiedemann, Schneltzer *et al.* 2007; Gibb, Zagorska *et al.* 2009; Morelli, Ares *et al.* 2009), rather than as propagating waves or the result of a conserved phase offset. **PSM cells** in

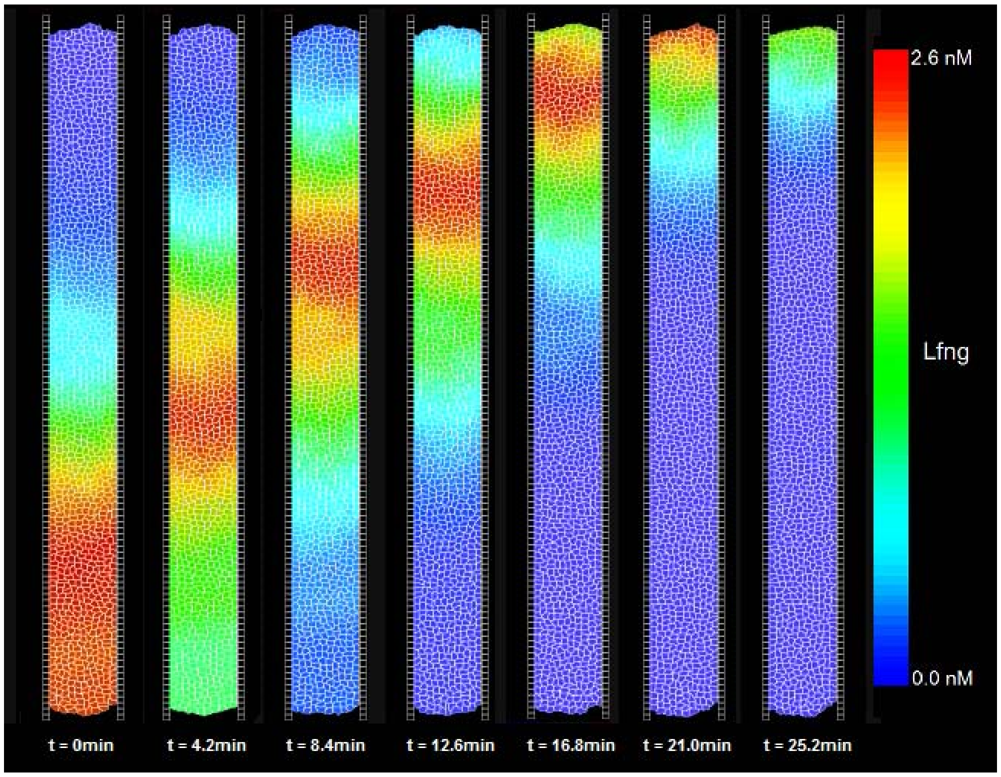

Figure 4.3: Traveling stripes of Lfng expression in the *in silico* PSM. In the presence of a [Wnt3a] gradient, broad stripes of Lfng expression originate in the posterior of the **PSM** and move anteriorly, narrowing as they do so. In order to aid visualization of the Lfng patterns, **cell** differentiation is turned off. Other parameters are equal to those in the reference simulations (**Figure 4.1**).

the computational model inherit their segmentation-clock phases from their parent **Source cells**. Because our reference simulations begin with the segmentation clocks in all **Source cells** in phase with one another, in the absence of any additional influences all **cells'** segmentation clocks would oscillate in phase and Lfng concentration would be spatially uniform throughout the **PSM** (as is the case for constant Wnt3a signaling, see **Figure 4.10** (**D**)). Thus, the pseudo-waves of Lfng in our simulations must result from interactions with a factor external to individual segmentation-clock networks. This factor is the Wnt3a signal which, while external to the segmentation-clock network, is internal to the **cell**. Simulations in which the segmentation clocks run cell-autonomously without Delta/Notch coupling between neighboring **cells** produce traveling Lfng expression stripes similar to those in simulations with **cell-cell** coupling, although noisier in the anterior since they lack **cell-cell** Delta/Notch coupling to compensate for **cell** mixing (**Figure 4.4**). Such cell-autonomous stripe behavior could not occur for our initial conditions were the stripes propagating waves. Our composite model differs from models of travelling waves (Tiedemann, Schneltzer *et al.* 2007; Santillán and Mackey 2008; Morelli, Ares *et al.* 2009; Uriu, Morishita *et al.* 2009) where the period-/phase-altering factor is either completely external to the cells or imposed as a gradient of an internal parameter.

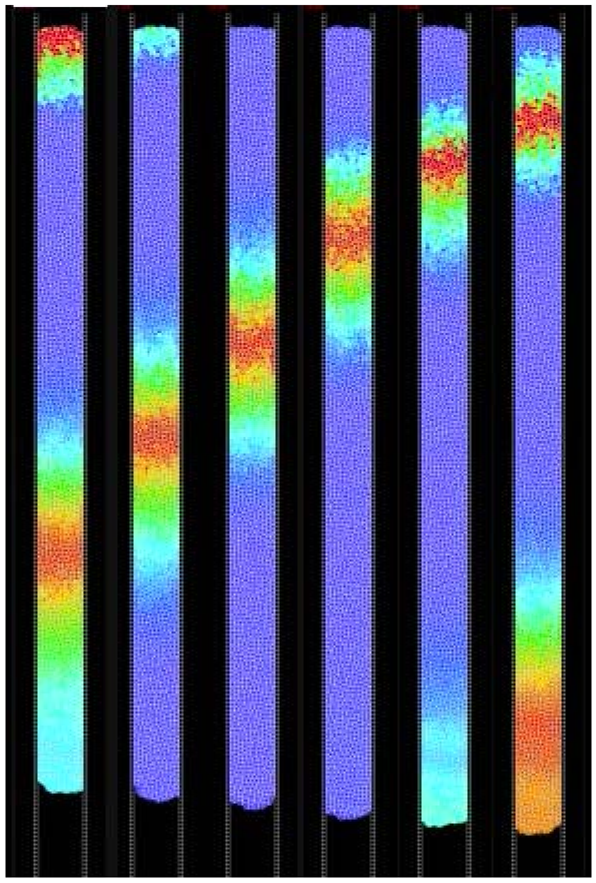

Figure 4.4: In the absence of Delta/Notch coupling between segmentation-clock networks, cell mixing leads to noisy Lfng expression stripes in the anterior PSM. With the exception of the lack of Delta/Notch signaling and the increased length of the **PSM** (to aid visualization), parameters are the same as in **Figure 4.3**. The color scale is also the same as that in **Figure 4.3**.

4.2 Cell motility affects somite border formation and morphology *in silico*

Somitogenesis in our simulations is sensitive to moderate (70%) changes in **cell** diffusion constants. Decreased **cell** motility in the **PSM** (D_{cell} = 0.86 µm^2/min) hinders **somite** rounding (**Figure 4.5** (**A-B**)) and prevents the variations in somite shape observed *in vivo* and in our reference simulations. Increased **cell** motility (D_{cell} = 1.76 µm^2/min) leads to over-mixing of **cells**, which can move significant distances in the AP direction, making adhesion-driven **cell** sorting insufficient to reform clean intersomitic boundaries and resulting in fused **somites** (**Figure 4.5** (**C**)). Our reference simulations use an intermediate **cell** motility for **PSM cells** (D_{cell} = 1.08 µm^2/min) comparable to the average cell motility measured in the PSM of chick embryos (D_{cell} ~ 1.0 µm^2/min) (Benazeraf, Francois et al. 2010). This degree of **cell** motility is high enough to allow **cell** sorting at the forming intersomitic borders, **somite** rounding, and variation in **somite** shapes, yet not so high that **cell** motion prevents appropriate border determination and **somite** separation (**Figure 4.1** and **Figure 4.5** (**B**)). *In vivo*, PSM cell motility in chick, as estimated from Figure 2 of (Benazeraf, Francois et al. 2010), ranges between approximately 2.75 µm^2/min in the extreme posterior PSM to approximately 0.25 µm^2/min in the extreme anterior PSM, so the range of **cell** motilities used in our simulations is experimentally reasonable.

In experiments, cells involved in somite border formation transiently increase their motility (Kulesa and Fraser 2002). This momentary increase also occurs in our simulations and arises spontaneously due to adhesion-driven **cell** sorting following differentiation. Measured diffusion constants for **cells** at the forming border can be as high as 3.0 µm^2/min, as is the case for the **cell** marked with a red dot in **Figure 4.1** (**G**-**M**), while the diffusion constant is only a third of this for a typical **cell** in the same simulation (D_{cell} = 1.08 µm^2/min for λ_{surf} = 15). Their higher motility results from directed migration of the misplaced **cells** to their final positions in response to changes in their local environments, *i.e.,* neighboring **cells**' adhesion properties.

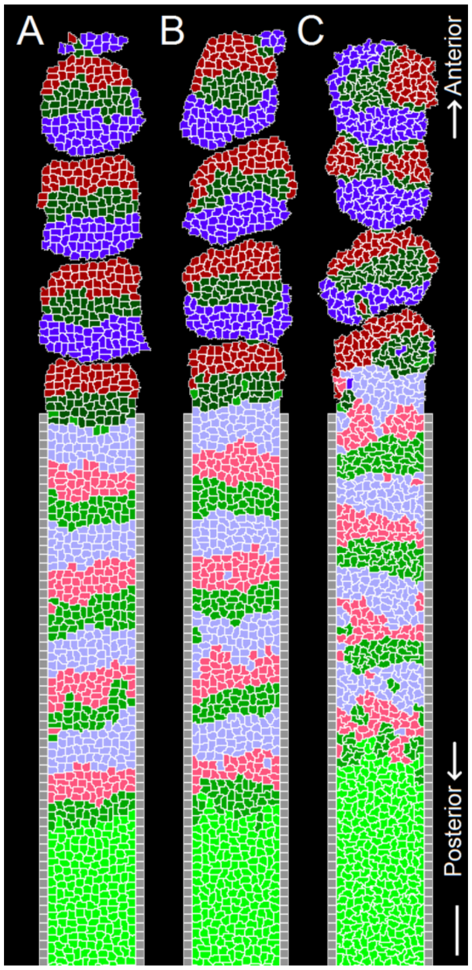

Figure 4.5: Somite quality dependence on cell motility *in silico*. We regulated **cell** motility by adjusting λ_{surf} . (**A**) Low **PSM cell** motility (λ_{surf} = 25, D_{cell} = 0.86 μm²/min): **somite** borders form, **somites** round up slowly compared to the reference simulation, and **somite** shape varies less than in the reference simulations. (**B**) Reference simulation, moderate **PSM cell** motility (λ_{surf} = 15, D_{cell} = 1.08 μm²/min): **cell** sorting corrects small amounts of initial **cell** mixing across presumptive **somite** borders, **somites** round up within a short time (about one segmentation-clock period after formation) and **somite** shape is variable. (**C**) High **PSM cell** motility compared to the reference simulation (λ_{surf} = 5, D_{cell} = 1.76 μm²/min): excessive mixing of **cell** types across presumptive **somite** borders leads to fused **somites**. Parameters, when not otherwise noted, are equal to those in the reference simulation (**Figure 4.1**). Anterior at the top. **Cell** colors are the same as in **Figure 3.6**. Scale bar 40 μm.

4.3 The segmentation-clock period and PSM growth rate regulate somite size *in silico*

Species vary greatly in their size and number of somites (**Table 4.1**). The clock-and-wavefront model can produce variation in somite size by varying the period of the segmentation clock and/or the speed of the advancing wavefront. Because the clock-and-wavefront model can continue to produce somites indefinitely, the number of somites is determined primarily by factors in the tailbud external to the PSM.

Table 4.1 Properties of somitogenesis during the modeled stages, by species.

	Zebrafish	Mouse	Chicken	Corn snake
AP somite length	50 μm	115 μm	150 μm	40 μm
Segmentation-clock period	30 min	120 min	90 min	100 min
PSM growth rate (AP)	2.49 μm/min	0.96 μm/min	1.66 μm/min	0.47 μm/min
PSM length	600 μm	1100 μm	1400 μm	1200 μm
Number of somites	31	65	55	315 ± 10%

Values estimated from Figure 4 in (Gomez, Ozbudak et al. 2008). The AP somite length and PSM growth rate are for stage HH 12+ (17 out of 52 somite pairs) in chicken and at the stages corresponding to the same fraction of total somites formed, in zebrafish, mouse and corn snake.

Predictably, decreasing the segmentation-clock period decreases **somite** size and increasing the segmentation-clock period increases **somite** size (**Figure 4.6**) (Baker, Schnell *et al.* 2006). If the rate of wavefront progression is unchanged, then the wavefront will travel a smaller distance during a shorter segmentation-clock period and a larger distance during a longer segmentation-clock period.

Because the FGF8 gradient is produced by gradual intracellular *fgf8* mRNA decay, rather than extracellular protein diffusion, increasing the rate of **PSM** growth “stretches” the shape of the FGF8 gradient (**Figure 4.7**) and thus increases the speed of the advancing determination wavefront. If the segmentation-clock period is unchanged, then a quickly-progressing wavefront

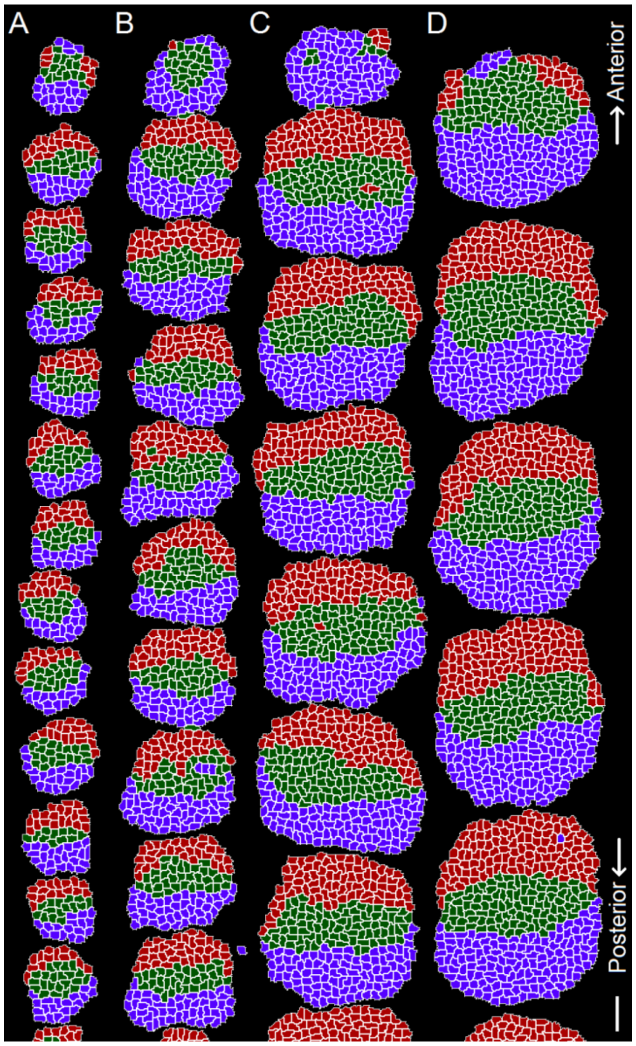

Figure 4.6: Somite size versus segmentation-clock period. Decreasing the period of the segmentation clock to 67.5 min shrinks **somites** (**A**) compared to the reference (chick) simulations with a segmentation-clock period of 90 min (**B**). Increasing the period of the segmentation clock to 135 minutes (**C**) or 180 min (**D**) forms proportionally larger **somites**. Well-formed smaller **somites** require decreased **cell** motility (λ_{surf} = 20; D_{cell} = 0.945 μm^2/min for **PSM cells** in (**A**)); larger **somites** form using the reference motility parameters (λ_{surf} = 15; D_{cell} = 1.08 μm^2/min for **PSM cells** in (**B-D**)). In each case, we adjusted the ML dimension to produce roughly circular **somites**. Segmentation and **somite** separation, however, succeed both for smaller and larger ML widths. Scale bar 40 μm. All other parameters are equal to those in the reference simulation (**Figure 4.1**). **Cell** colors are the same as in **Figure 3.6**.

will travel a greater distance during a segmentation-clock period and result in larger **somites**, and a slowly progressing wavefront will travel a smaller distance over a segmentation-clock period and result in smaller **somites** (**Figure 4.8**).

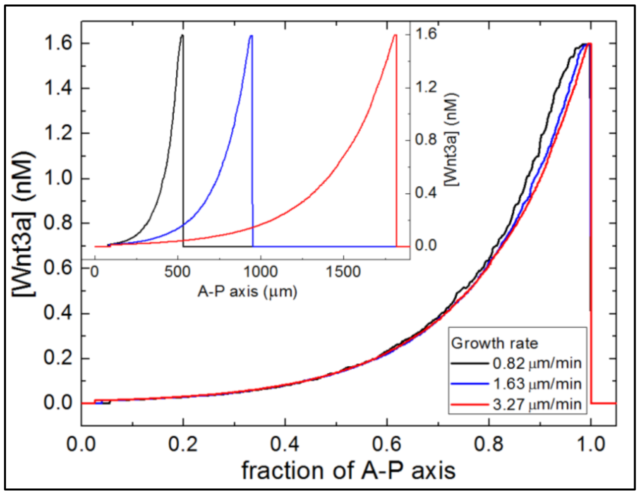

Figure 4.7: Effect of PSM growth rate on the Wnt3a profile in the simulated PSM. Faster (slower) **PSM** growth lengthens (shortens) the **PSM**, leaving the anterior and posterior concentrations of Wnt3a unchanged (inset). Normalizing the AP position by the total **PSM** length, the Wnt3a profiles for different growth rates are nearly identical. The AP position of the anterior of the **PSM** is defined to be zero. **PSM** growth rates: (black line) 0.82 μm/min; (blue line) 1.63 μm/min; (red line) 3.27 μm/min. All other parameters are equal to those in the reference simulation (**Figure 4.1**).

Well-defined larger **somites** formed without further adjustments to the reference simulation parameters, but formation of smaller **somites** with well-defined borders and clean intersomitic gaps required smaller **cell** motility than the reference simulation. **Cell** mixing easily disrupts extremely narrow compartments of **pre_EphA4**, **pre_ephrinB2** and **pre_ Core cells**, interfering with future apposition of **EphA4** and **ephrinB2 cell** compartments after differentiation. If a continuous cluster of **pre_Core cells** breaks through an adjacent compartment of **pre_EphA4** or **pre_ephrinB2 cells**, adhesion-mediated border correction, gap formation and compartment maintenance fail and the **somites** fuse (**Figure 4.9** (**A**)). For large **somites**, if a single **pre_Core cell** or a small cluster of **pre_Core cells** breaks through a neighboring compartment into the

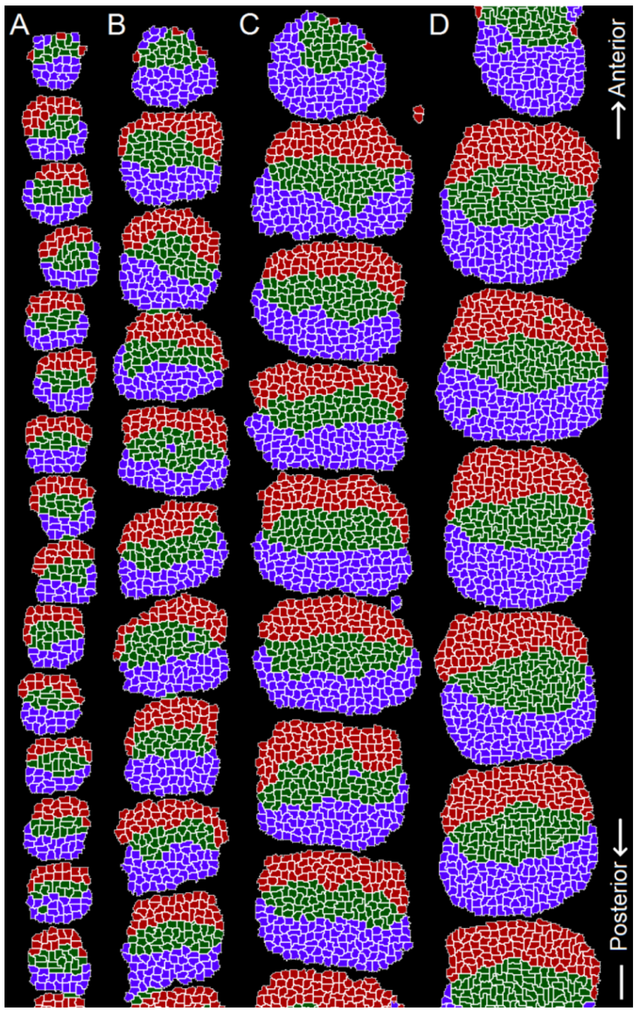

Figure 4.8: Somite size versus the rate of PSM growth. (**A**) Decreasing the rate of **PSM** growth to 1.08 μm/min compared to the reference simulation growth rate of 1.63 μm/min shrinks **somites**. (**B**) Reference simulation. (**C**,**D**) Increasing the rate of **PSM** growth to 2.04 μm/min (**C**) or 2.72 μm/min (**D**) forms proportionally larger **somites**. Well formed smaller **somites** require decreased **cell** motility (λ_{surf} = 25 in (**A**)); larger **somites** form using the reference **cell** motility parameters (λ_{surf} = 15 for (**B**-**D**)). In each case, we adjust the ML dimension to produce roughly circular **somites**. Segmentation and **somite** separation, however, both succeed for smaller and larger ML widths (data not shown). Scale bar 40 μm. All other parameters are equal to those in the reference simulation (**Figure 4.1**). **Cell** colors are the same as in **Figure 3.6**.

future intersomitic gap, then the two **somites** will be joined by the **Core cells**, which will prevent complete formation of the intersomitic gap, but the two joined **somites** will otherwise form normally and maintain their intrasomitic compartments (**Figure 4.9** (**B**)). For small **somites**, even a single breakthrough **pre_Core cell** may be enough to fuse adjacent **somites**, allowing us to predict that cell motility will be lower compared to the segmentation-clock period in species with very small somites than in species with large somites.

Our ability to control simulated **somite** size and formation rate by adjusting distinct mechanisms *in silico* allows us to explore the plasticity and robustness of the modeled somitogenesis mechanisms and provides a convenient tool for exploring the development and evolution of interspecies variability.

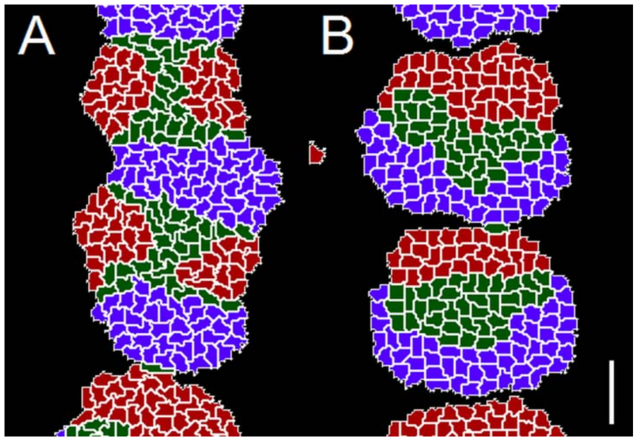

Figure 4.9: Somitogenesis defects. (**A**) A group of **Core cells** breaks through an adjacent EphA4 or ephrinB2 compartment, leading to fused **somites**. **Somite** fusing is a defective phenotype that does not occur in normal *in vivo* or *in silico* somitogenesis. (**B**) A single **Core cell** is stranded in the naturally acellular perisomitic ECM. Such stranded cells occasionally occur in normal *in vivo* somitogenesis. **Cell** colors are the same as in **Figure 3.6**. Scale bar 40 μm.

4.4 The number of Lfng expression stripes *in silico* depends on the segmentation-clock period, PSM growth rate and PSM length

In addition to affecting **somite** size, varying the segmentation-clock period and morphogen distribution changes the patterns of segmentation-clock protein and mRNA concentrations across the **PSM** *in silico*. *In vivo*, the number of high-Lfng-concentration stripes simultaneously present in the PSM varies between species, ranging from one or two in chick to as many as nine in the corn snake (Gomez, Ozbudak et al. 2008).

To explore the relationship between segmentation-clock period, PSM growth rate and number of Lfng stripes in the PSM, we first visualized Lfng concentration in simulations with a fixed rate of **PSM** growth and varying segmentation-clock periods (**Figure 4.10**). As expected, faster segmentation clocks produce greater numbers of stripes than slower clocks. When multiple stripes are present simultaneously, the distance between stripes narrows as the stripes move anteriorly down the **PSM**, as seen in experiments. In conjunction with the segmentation-clock period gradient along the **PSM** (increasing anteriorly along the tissue), this observation supports the methods Gomez *et al.* and Giudicelli *et al.* (Giudicelli, Ozbudak et al. 2007; Gomez, Ozbudak et al. 2008) used to calculate the segmentation-clock period at different positions in experimental tissues.

Next, we varied the **PSM** growth rate while holding the segmentation-clock period fixed. We did so in simulations implementing both the cell-autonomous differentiation and positional differentiation-front submodels.

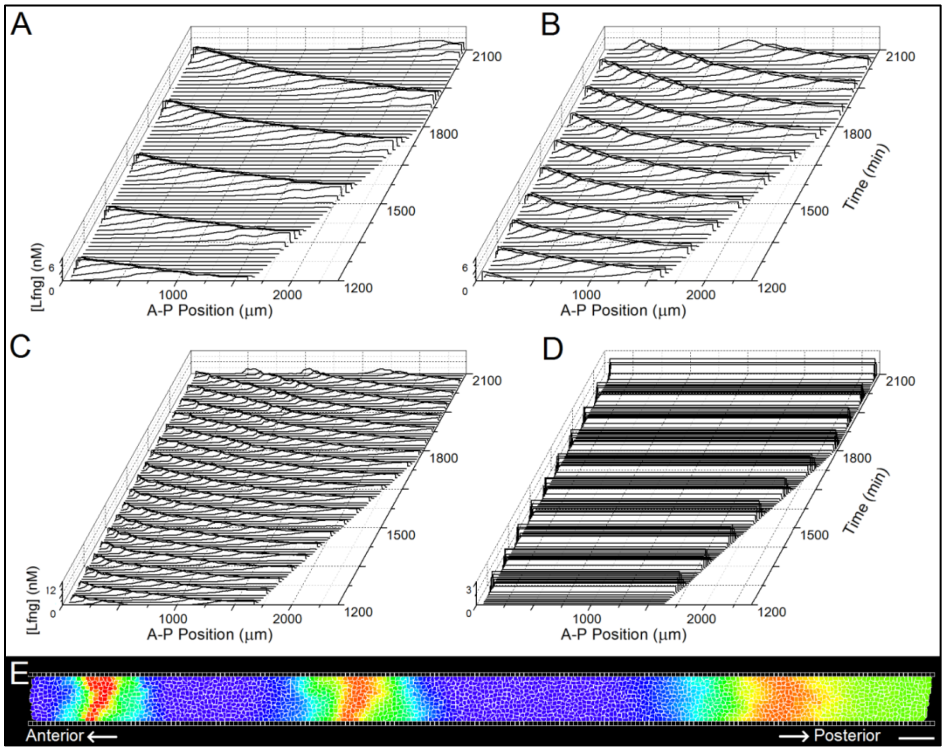

Figure 4.10: Anteriorly traveling Lfng stripes and segmentation-clock period. (**A-C**) Lfng expression versus AP position and time for different segmentation-clock periods. (**A**) Increasing the segmentation-clock period to 180 min from the reference simulation period of 90 min decreases the spatial and temporal frequency of Lfng stripes compared to the reference simulation (**B**). (**C**) Decreasing the segmentation-clock period to 45 min increases the spatial and temporal frequency of Lfng stripes compared to the reference simulation ([Lfng] axis rescaled for clarity). (**D**) For a uniform Wnt3a concentration of 0.5 nM, **cells**' segmentation-clocks oscillate in phase with a period of 90 min. (**E**) Lfng concentration in a simulation with a segmentation-clock period of 45 min. The distance between the center and anterior (left) peaks is shorter than the distance between the center and posterior (right) peaks. Scale bar 40 μm. Parameters, when not otherwise noted, are equal to those in the reference simulation (**Figure 4.1**). The color scale is the same as that in **Figure 3.6**.

Because the shapes of the FGF8 and Wnt3a concentration gradients depend on intracellular mRNA decay rather than on the protein diffusion lengths (Aulehla, Wehrle et al. 2003; Dubrulle and Pourquie 2004; Goldbeter, Gonze et al. 2007), if the parameters governing FGF8 and Wnt3a production and decay are unchanged, faster simulated **PSM** growth stretches the Wnt3a and FGF8 concentration gradients along the AP axis relative to the reference simulation, while slower **PSM** growth compresses the FGF8 and Wnt3a concentration gradients. In a cell-autonomous differentiation model, the minimum (anterior) concentration of Wnt3a (which is cell-autonomous in our model and nearly cell-autonomous *in vivo*) is unchanged from the reference simulation, so when the AP position is normalized by the **PSM** length, the Wnt3a concentration gradient is unchanged (see **Figure 4.6**). Changing the AP Wnt3a concentration gradient correspondingly changes the patterns of Lfng concentration: faster **PSM** growth broadens stripes of high Lfng concentration and slower **PSM** growth narrows them, leaving the number of Lfng stripes unchanged (**Figure 4.11** (**A-D**)).

When fixing the length of the **PSM** and allowing the minimum (anterior) concentrations of FGF8 and Wnt3a to vary with the **PSM** growth rate, slower **PSM** growth increases the number of Lfng stripes compared to the reference simulation and faster **PSM** growth decreases the number of Lfng stripes compared to the reference simulation (**Figure 4.11** (**E-G**)), consistent with the *in vivo* observations of Gomez *et al.* (Gomez, Ozbudak et al. 2008) that the number of Lfng stripes in the PSM depends on the segmentation-clock period relative to the PSM growth rate and that PSM length is comparable between organisms with significantly different growth rates (see **Table 4.1**).

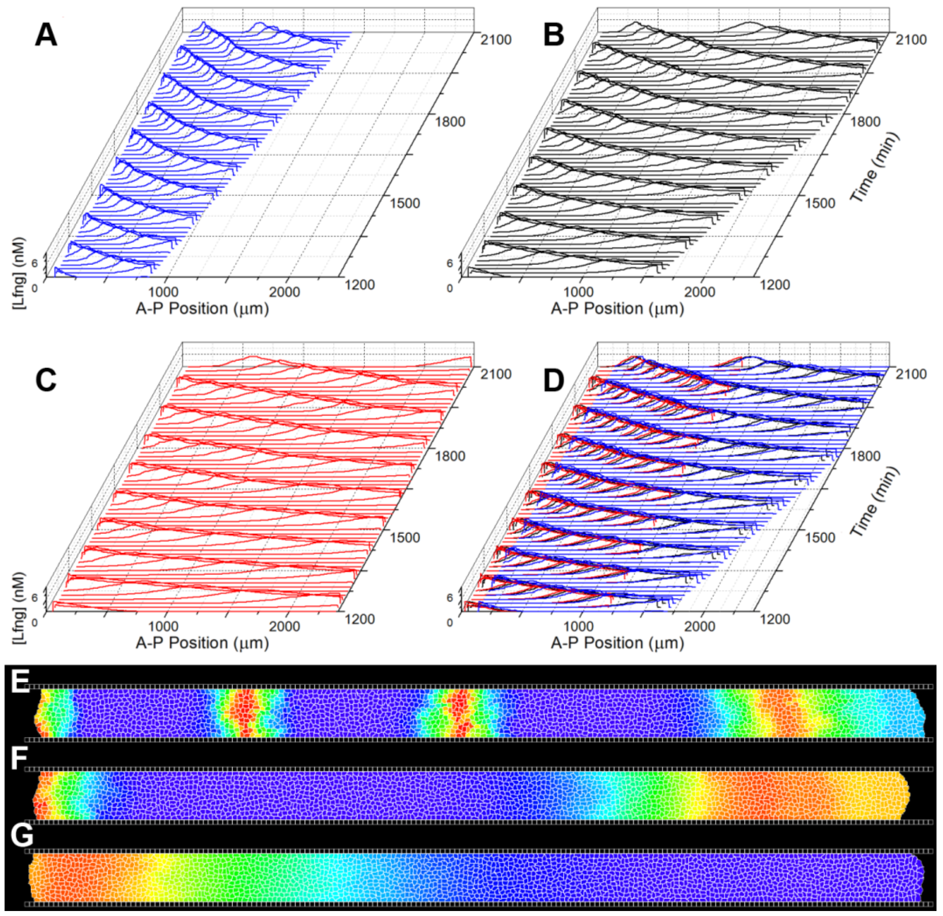

Figure 4.11: *In silico* Lfng expression for different PSM growth rates. (**A-D**) The number of *in silico* Lfng stripes in the **PSM** is independent of the **PSM** growth rate for fixed segmentation-clock period and minimum (anterior) concentration of FGF8. (**A**) **PSM** growth rate = 0.82 μm/min. (**B**) Reference simulation (**PSM** growth rate = 1.63 μm/min). (**C**) **PSM** growth rate = 3.27 μm/min. (**D**) Rescaling the length of the **PSM** to match the reference simulation demonstrates that the three cases are equivalent after accounting for the expansion or compression of the Wnt3a gradient. (**E-G**) The number of *in silico* Lfng concentration stripes in the **PSM** depends on the **PSM** growth rate for a fixed segmentation-clock period and **PSM** length. (**E**) **PSM** growth rate = 0.82 μm/min. (**F**) Reference simulation (**PSM** growth rate = 1.63 μm/min). (**G**) **PSM** growth rate = 3.27 μm/min. Anterior to the left. Scale bar 80 μm. The color scale is the same as that in **Figure 3.6**. Parameters, when not otherwise noted, are equal to those in the reference simulation (**Figure 4.1**).

Gomez *et al.* estimated that the number of high Lfng concentration stripes should be proportional to the ratio of the length of the PSM to the somite size, given a conserved relationship between the segmentation-clock period and the AP position as a fraction of the total PSM length. In our model, in which the **PSM** increases its AP length exclusively through **cell** addition at the posterior end rather than through uniform expansion of the PSM due to cell proliferation throughout the tissue, the relationship between the fractional AP position and the segmentation-clock period is not conserved when the **PSM** length is constant and the **PSM** growth rate is varied. However, even without changing the parameters governing FGF8 and Wnt3a production, diffusion or decay (*i.e.*, without changing the relationship between the absolute AP position and the segmentation-clock period), our model predicts that the number of high Lfng concentration stripes in the PSM will increase with the ratio of PSM length to somite size.

These observations, summarized in **Table 4.2**, have implications regarding how the differentiation front is positioned *in vivo*. If the FGF8 concentration defines the differentiation front independently of other factors, then the length of the PSM depends only on the PSM growth rate, and the number of Lfng stripes will be independent of the growth rate. Additional FGF8 antagonists or differentiation promoters, *e.g.*, RA, however, might allow both slow and fast PSM growth to produce a PSM of the same length, in which case a more slowly growing PSM will have more simultaneous Lfng stripes (see **Figure 4.11**).

Simulations in which the **PSM** length is fixed are more consistent with *in vivo* observations that Lfng stripe number depends on the PSM growth rate (Gomez, Ozbudak et al. 2008) than simulations in which differentiation is cell-autonomous or nearly cell-autonomous, suggesting that at least one additional signal besides FGF8, probably RA, positions the differentiation front and maintains a constant PSM length.

Table 4.2: Dependence of the number of Lfng stripes in the modeled PSM on the PSM growth rate and segmentation-clock period.

PSM length	PSM growth rate	Minimum (anterior) FGF8 and Wnt3a	Segmentation-clock frequency	Number of simultaneous Lfng stripes
Reference	Reference	Reference	High	Increased
Reference	Reference	Reference	Low	Decreased
Variable	High	Reference	Reference	Reference
Variable	Low	Reference	Reference	Reference
Reference	High	Variable	Reference	Decreased
Reference	Low	Variable	Reference	Increased

"Reference" indicates that the value is the same as in the reference simulation; "Variable" indicates that the value is free to change in response to changes in other factors; "High" and "Low" indicate imposed changes; "Increased" and "Decreased" indicate results for imposed changes. All are relative to the values in the reference simulation.

4.5 Somites form *in silico* in the absence of dynamic gene expression stripes

Traveling stripes of gene expression are observed in the PSM of many species, including chick, mouse, zebrafish and corn snake (Palmeirim, Henrique et al. 1997; Forsberg, Crozet et al. 1998; Dequeant, Glynn et al. 2006; Gomez, Ozbudak et al. 2008). Gibb *et al.* (Gibb, Zagorska *et al.* 2009) proposed that a Wnt-gradient-based segmentation-clock period gradient and the resulting traveling stripes of high protein concentration are conserved across species, and suggested that traveling stripes may play an important role in somitogenesis. *In vivo*, stripes arrest and stabilize at the position of the next presumptive somite, suggesting that they may be involved in cell-type specification and/or differentiation prior to somite formation. Such a mechanism would be a significant extension of a pure clock-and-wavefront model, which does not require a non-uniform segmentation-clock phase profile in the PSM. Indeed, the original clock-and-wavefront model (Cooke and Zeeman 1976) included intracellular segmentation clocks but neither required nor predicted anteriorly-traveling stripes of high protein concentration.

Simulations of our model with a uniform Wnt3a concentration corresponding to the level of the determination front in our reference simulation did not produce traveling stripes of high Lfng concentration (see **Figure 4.10** (**D**)) but formed normal **somites** (**Figure 4.12**). The constant Wnt3a concentration actually improves synchronization between segmentation clocks in adjacent **cells**, reducing anterior-posterior cell misdifferentiation and increasing **somite** accuracy and regularity. These results suggest that traveling stripes and AP variation in segmentation-clock period are not essential for somitogenesis. However, the stripes may play a role in aspects of somitogenesis or subsequent development that are not accounted for in our model.

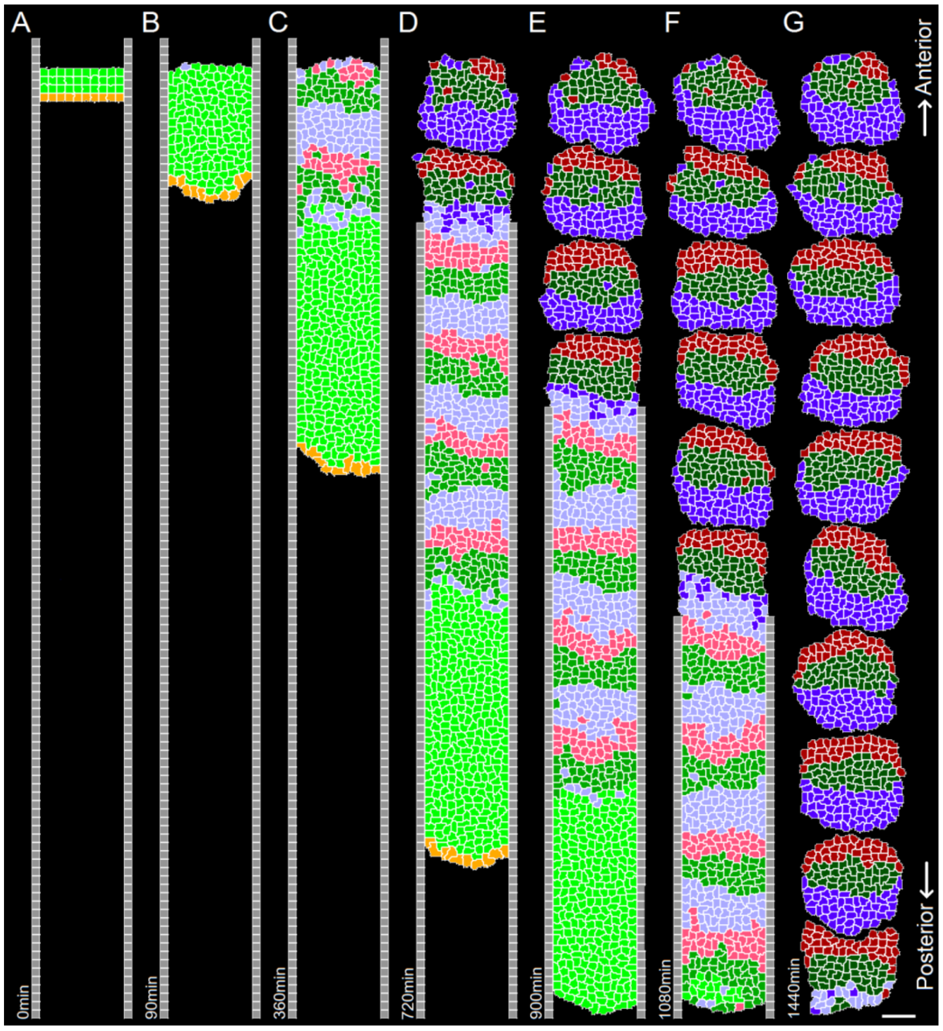

Figure 4.12: ***In silico*** **somitogenesis with a uniform Wnt3a concentration.** When [Wnt3a] is uniform throughout the **PSM**, traveling Lfng stripes do not form, but segmentation is normal, demonstrating that traveling stripes of high protein concentration are not necessary for somitogenesis in our model. The constant Wnt3a concentration actually improves synchronization between segmentation clocks in adjacent **cells**, reducing anterior-posterior cell misdifferentiation and increasing **somite** accuracy and regularity. Times: (**A**) 0 min, (**B**) 90 min, (**C**) 360 min, (**D**) 720 min, (**E**) 900 min, (**F**) 1080 min and (**G**) 1440 min. [Wnt3a] = 0.5 nM. All other parameters are equal to those in the reference simulation (**Figure 4.1**). Anterior at the top. Scale bar 40 μm. **Cell** colors are the same as in **Figure 3.6**.

4.6 The sensitivity of *in silico* somite formation depends on the intervening changes in cell-cell adhesion

4.6.1 Somites form *in silico* for a wide range of determination-differentiation delays when determined cell types have intermediate adhesion

We chose an interval of four segmentation-clock periods as the reference delay between **cell** fate determination and **cell** differentiation based on Dubrulle and colleagues' experimental observation that in chick, cell-type determination occurs approximately four somite lengths in advance of the newest somite border (Dubrulle, McGrew *et al.* 2001). The duration of the delay varies among organisms (Dubrulle, McGrew et al. 2001; Giudicelli, Ozbudak et al. 2007; Gomez, Ozbudak et al. 2008), but neither the need for a delay nor the reasons for its duration are apparent. For the clock-and-wavefront mechanism to function in multiple species, it must function over a wide range of delay times. We simulated **somite** formation for determination-to-differentiation delays ranging from zero to 8 segmentation-clock periods. Clean **somite** borders formed for all delays simulated (**Figure 4.13**). To present a more challenging case, we ran simulations with long (eight segmentation-clock periods) determination-to-differentiation delays and determined **cell** types with adhesive properties closer to those of undetermined **PSM cells**. These simulations showed more mixing between **cells** with distinct determined **cell** types than in the reference simulation, but, after differentiation, the **cells** still sorted into distinct populations with well-defined borders, forming **somites** indistinguishable from those in the reference simulation (**Figure 4.14**).

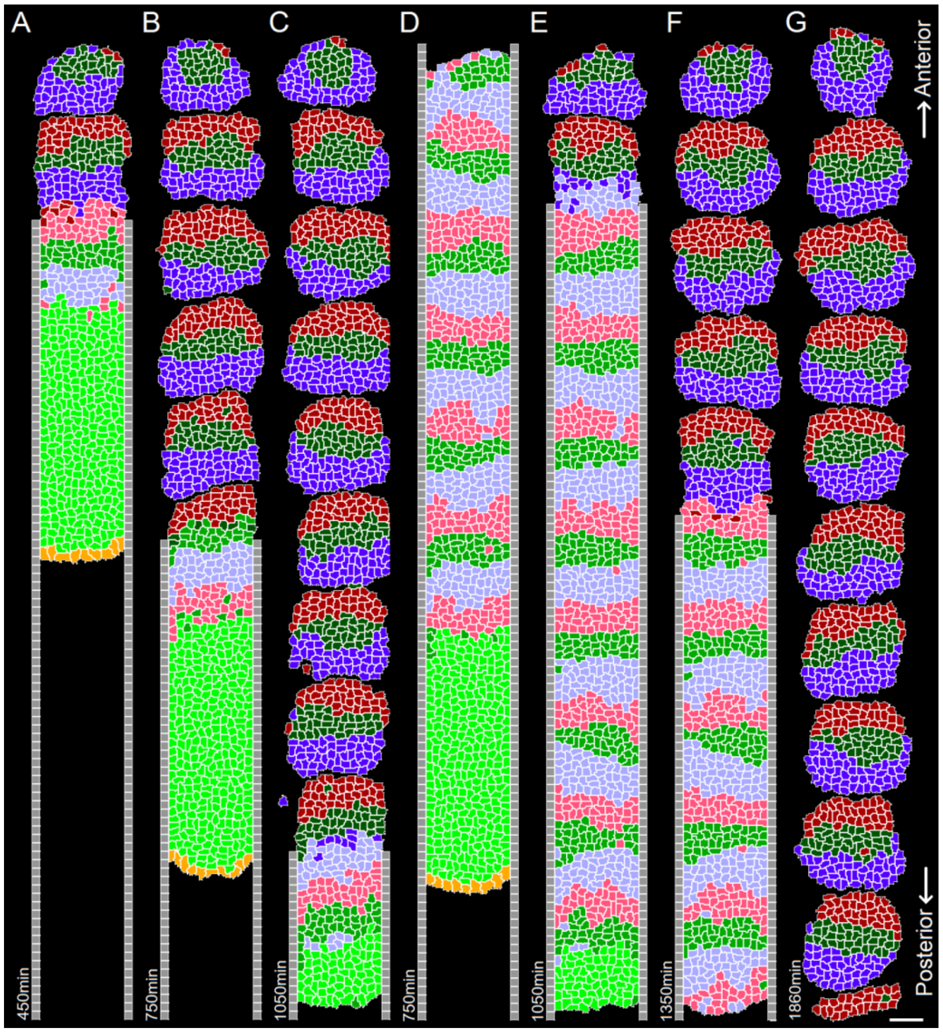

Figure 4.13: *In silico* somite formation dependence on the time interval between determination and differentiation. Somites form independently of the determination-differentiation delay for reference adhesion values. (**A-C**) Snapshots for a shorter than normal determination-differentiation delay of one segmentation-clock period (90 min), taken at: (**A**) 450 min, (**B**) 750 min and (**C**) 1050 min. (**D-G**) Snapshots for a longer than usual determination-differentiation delay of eight segmentation-clock periods (720 min), taken at: (**D**) 750 min, (**E**) 1050 min, (**F**) 1350 min and (**G**) 1860 min. The determination-differentiation delay in the reference simulation is four clock periods (360 min). Parameters, when not otherwise noted, are equal to those in the reference simulation (**Figure 4.1**). Anterior at top. Scale bar 40 μm. **Cell** colors are the same as in **Figure 3.6**.

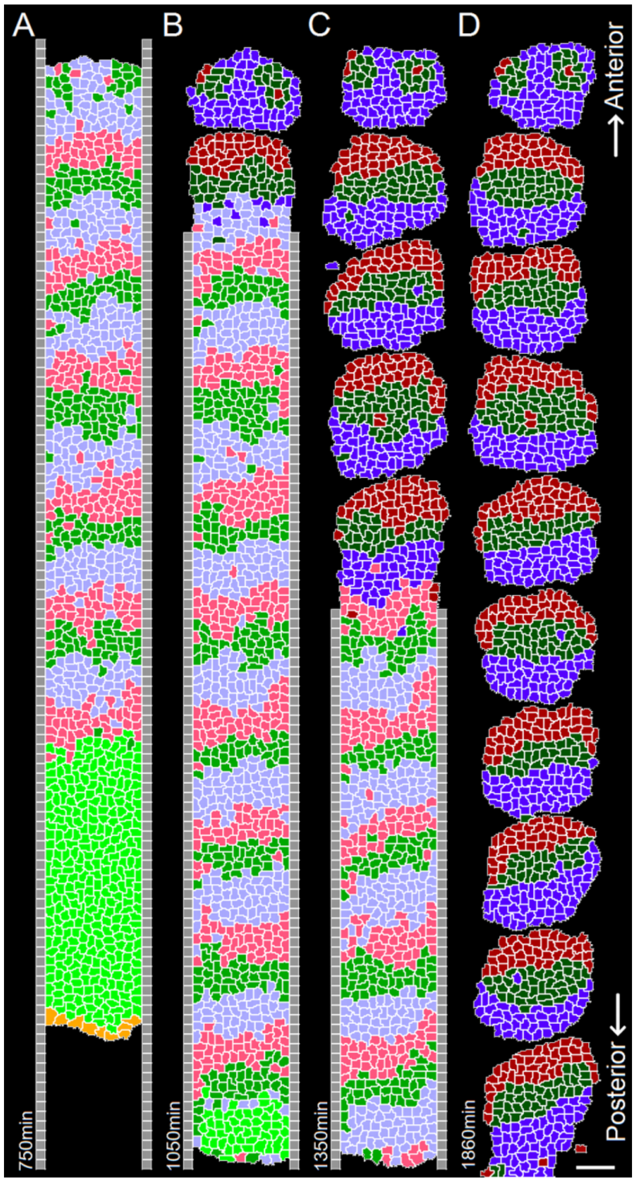

Figure 4.14: Somite formation with increased post-determination cell mixing. Assigning a larger determination-differentiation delay (8 cell cycles) and adhesion parameters for **determined cell types** closer to those of **PSM cells** than in the reference simulation increases **cell** mixing among distinct determined **cell types** prior to differentiation. However, **cell** sorting after differentiation corrects the moderate amount of mixing across presumptive **somite** borders and leads to clean **somite** boundaries. Times: (**A**) 750 min, (**B**) 1050 min, (**C**) 1350 min and (**D**) 1860 min. Anterior at top. Scale bar 40 µm. Contact energies: $J_{\text{pre\_EphA4,pre\_EphA4}}$ = -22; $J_{\text{pre\_ephrinB2,pre\_ephrinB2}}$ = -22; $J_{\text{pre\_Core,pre\_Core}}$ = -25; $J_{\text{pre\_EphA4,EphA4}}$ = -22; $J_{\text{pre\_ephrinB2,ephrinB2}}$ = -22; other contact energies are unchanged from **Table 3**. Parameters, when not otherwise noted, are equal to those in the reference simulation (**Figure 4.1**). **Cell** colors are the same as in **Figure 3.6**.

4.6.2 Somite formation *in silico* is sensitive to long delays between determination and changes in cell-cell adhesion

Together, the above results suggest that somitogenesis is relatively insensitive to the length of the determination-differentiation delay and may not require a determination-to-differentiation interval. One simplification included in our model, however, is that immediately after determination **cells** change their adhesion properties slightly, whereas, *in vivo*, determined cells take some time to express even low concentrations of adhesion molecules at their membranes. In order to more closely model the situation *in vivo*, we introduced an interval between **cell** fate specification and any changes to their adhesion properties.

In one set of simulations, **cells** underwent differentiation immediately after the interval, with no period of intermediate adhesion. For intervals of fewer than two segmentation-clock periods, these simulations formed **somites** that were joined by a greater-than-normal number of **Core cells** stranded in the intersomitic gaps. For intervals of two or more segmentation-clock periods, these simulations formed **somites** that were predominantly fused or joined by large numbers of **Core cells** stranded in the intersomitic gaps. The severity of the *in silico* phenotype increased for increasing intervals (**Figure 4.15**).

In a second set of simulations, **cells** had **determined-cell-type** adhesions for the remainder of the standard four-segmentation-clock-period determination-differentiation delay. In these simulations, **cell** sorting during the period of intermediate adhesion partially corrected the *in silico* phenotype, preventing fused **somites** in simulations with intervals of two segmentation-clock periods, decreasing the number of fused **somites** in simulations with intervals of greater than two segmentation-clock periods, and decreasing both the number of stranded **Core cells** and the number of **somites** joined by stranded **Core cells** for all tested intervals (see **Figure 4.15**).

The results presented in **Figure 4.13** indicate that the differences between the outcomes of these two sets of simulations are not due exclusively to the difference in the duration of the

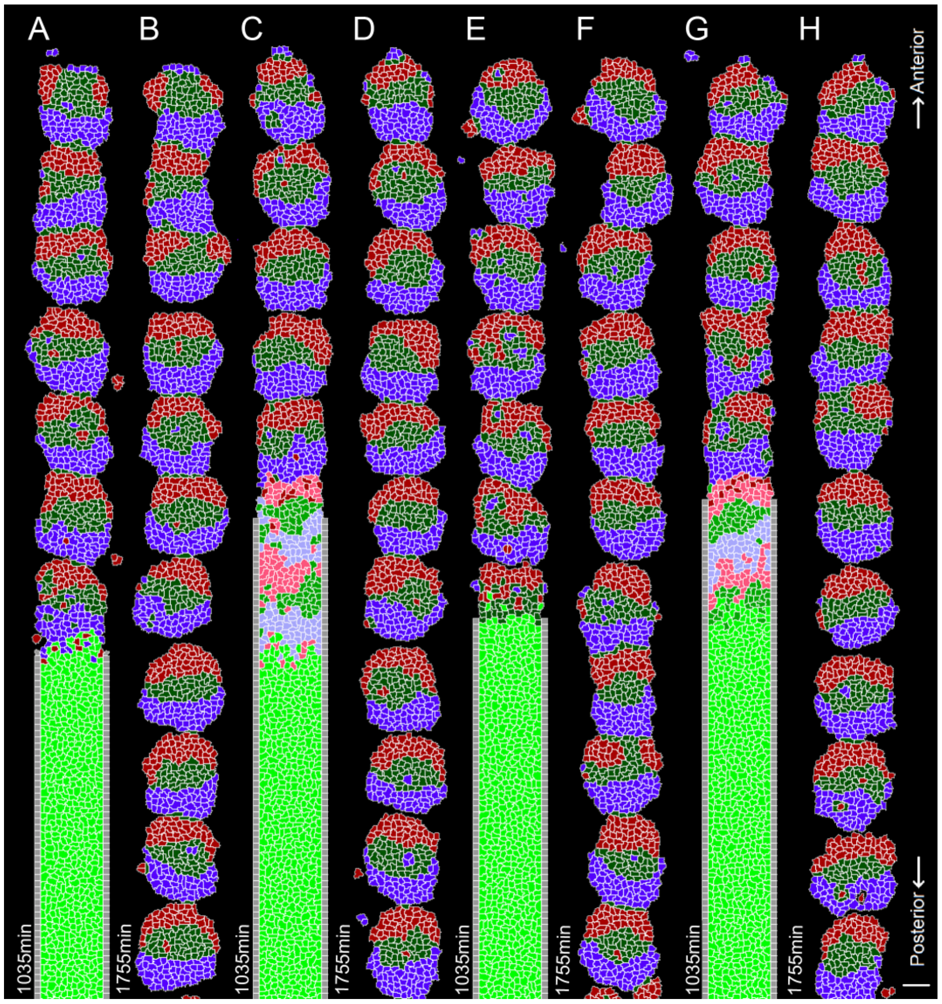

Figure 4.15: Effect of intermediate adhesion levels between determination and differentiation on segmentation quality. Without a period of intermediate adhesion, the excessive mixing of **determined cell types** across their original borders leads to fused **somites** and a high occurrence of intersomitic stranded **Core cells**. When **cells** have a period of **determined-cell-type** adhesions during the determination-differentiation interval, **cell** over-mixing is partially corrected. (**A-H**) t_1 = determination-differentiation interval; t_2 = period of intermediate adhesion (within interval). (**A-B**) t_1 = 180 min, t_2 = 0 min at: (**A**) 1035 min and (**B**) 1755 min; (**E-F**) t_1 = 225 min, t_2 = 0 min at: (**E**) 1035 min and (**F**) 1755min. (**C-D**) t_1 = 360 min, t_2 = 180 min at: (**C**) 1035 min and (**D**) 1755 min; (**G-H**) t_1 = 360 min, t_2 = 135 min at: (**G**) 1035 min and (**H**) 1755 min. Except where otherwise stated, parameters are equal to those in the reference simulation (**Figure 4.1**). **Cell** colors are the same as in **Figure 3.6**.

period of intermediate adhesion: in the absence of an interval between **cell**-fate specification and changes to **cells'** adhesion properties, **somites** form normally even for very short periods of intermediate adhesion (**Figure 4.13 (A-C)**).

These results suggest that cell sorting during the determination-differentiation delay is functional and enables a required error correction mechanism depending on gradually changing adhesion properties, as approximated in our first set of simulations.

Chapter 5

DISCUSSION

Our integrated model of somitogenesis combines submodels operating at different scales: an extended and corrected Goldbeter-Pourquié intracellular segmentation-clock network (Goldbeter and Pourquie 2008), Lewis-like Delta/Notch cell-cell segmentation-clock synchronization (Lewis 2003), signaling gradients produced by FGF8 mRNA and Wnt3a protein decay (Aulehla, Wehrle et al. 2003; Dubrulle and Pourquie 2004), and a simplified version of Glazier *et al.*'s 2D model of differential cell-adhesion- and motility-driven intersomitic gap formation and intrasomitic compartment maintenance (Glazier, Zhang *et al.* 2008), as well as novel models of cell-type determination and differentiation.

5.1 Features and limitations of individual submodels revealed by building the composite model and implications of coordination between mechanisms across scales

Integrating submodels revealed previously unappreciated features and limitations of those submodels. While the Goldbeter-Pourquié intracellular segmentation-clock network and Lewis-style intercellular Delta/Notch coupling seem reasonable when considered separately, combining them showed a significant limitation of the Goldbeter-Pourquié model: even with the addition of intercellular coupling through the Delta/Notch loop, the model is unable to entrain Wnt and FGF oscillations among **cells**. We resolved this inconsistency in our extended submodel by adding experimentally-supported feedback from the Notch pathway to the FGF pathway (Niwa, Masamizu *et al.* 2007) (additional juxtacrine signaling between **cells'** FGF and/or Wnt oscillators could also solve the problem but is experimentally unsupported, though not ruled out). Our extended segmentation-clock submodel is able to synchronize neighboring **cells** and maintain synchronization for realistic levels of **cell** motion and neighbor-exchange in the absence of significant additional perturbation, but is limited in its ability to synchronize neighboring **cells** when we introduce additional perturbations like a random initial distribution of clock phases (**somite** formation is robust for variations of about ± 5% in initial clock phases, but fails for greater variation).

At the same time, the Goldbeter-Pourquié segmentation-clock model has the previously unreported feature that its oscillation period responds to the degree of Wnt signaling. This property is responsible for the spontaneous emergence in our model of “pseudo-waves” of Lfng expression that resemble the traveling-stripe patterns observed *in vivo.* The extended clock submodel also produces a consistent phase relationship among oscillating components, providing a plausible mechanism for translating internal **cell** states at determination into mechanical properties that distinguish differentiated **cell** types.

While we implemented a particular submodel for **cell** determination based on a set of provocative observations (Dubrulle, McGrew *et al.* 2001), the results of our integrated model do not depend in detail on this particular determination submodel. The network in **Figure 2.3** (**A**) is only a speculative mechanism connecting the segmentation clock, determination front and the mechanical properties that characterize subpopulations of cells. We currently lack the experimental evidence necessary to construct a realistic model network capable of explaining PSM cell determination. The results that I have presented do, however, demonstrate that fate determination according to a cell’s internal state (*i.e.*, the phase of its segmentation clock) at the time that it encounters a determination front is a plausible mechanism for patterning a dynamic PSM.

In our submodel of the morphogen gradients, we neglected possible non-diffusive mechanisms of molecular transport such as endocytotic transport (discussed in (Dierick and Bejsovec 1999; Pfeiffer and Vincent 1999; Christian 2000; Scholpp and Brand 2004)). We also neglected potential feedback between FGF8 and Wnt3a signaling (aside from their interaction within the segmentation-clock network). Signaling in the integrated model is thus close to cell-autonomous for FGF8 and cell-autonomous for Wnt, so that the magnitude of signaling depends strongly on the amount of time that a **cell** has been in the **PSM** rather than on its AP position. Because time spent in the **PSM** and AP position correlate closely, the impact of this simplification is relatively slight, but not completely negligible, particularly in cases of high **cell** motility. Groups of **cells** that enter the simulation at approximately the same time determine and differentiate roughly simultaneously regardless of their spatial separation.

Non-**cell**-autonomous mechanisms play a significant role in producing the model's results. This is seen in simulations in which **cells** retain their **PSM**-like adhesion properties for some time after receiving their somitic fates. Allowing **cells** to mix for some time after receiving their fates, but before they change their adhesion properties, disrupts **somite** formation. Were **cells** acting completely autonomously, the cases with and without this post-determination mixing would be identical, since **cells** would have had determined fates from the moment they entered the simulation.

Our model strengthens the claim made in (Glazier, Zhang *et al.* 2008) that differential adhesion is capable of translating patterns of protein expression into tissue morphology in the PSM. While the previous work periodically imposed patterns of the relevant adhesion molecules, induced determination and differentiation in conjunction with **cell** motion results in a more stochastic pattern of adhesion molecules; nevertheless **cell** rearrangements due to differential adhesion still produce dynamic morphologies reminiscent of those *in vivo*.

5.2 Testable predictions of the composite model and opportunities for future experiments

The emergent behaviors that arise within the integrated model and its response to certain perturbations lead to a series of experimentally-testable predictions. In our model, the spontaneous transient increase in **cell** motion during **somite** formation results from sorting due to differential **cell**-**cell** adhesion, predicting that experimentally interfering with the mechanics of cell adhesion and effective repulsion will decrease measured cell motility and hamper cell rearrangement along forming somite borders. Although experiments have shown that boundary formation in zebrafish requires Eph/ephrin signaling (Durbin, Brennan *et al.* 1998; Watanabe, Sato *et al.* 2009), a detailed study of the effects of EphA4 and ephrinB2 knock-outs on cell-cell adhesion and boundary dynamics is still lacking, particularly in chicken and mouse.

The adhesion-driven cell-sorting submodel predicts the persistence of some misplaced cell types well into somitogenesis; so in experiments which label cells based on anterior and posterior somitic markers, a few misplaced cells (according to their genetic AP identities) should persist in the most recently-formed somites.

Simulating organisms with more cells per somite only required changing the period of the segmentation clock or the **PSM** growth rate. Simulating organisms with fewer cells per somite additionally required decreased **cell** motility due to the narrow width of the stripes of EphA4-/ephrinB2-expressing **cells**. This result suggests that smaller embryos (those with fewer cells per somite) require stricter control of spatial stochasticity, predicting that evolutionary pressures will lead to noisier (more error-prone) differentiation in organisms with big somites than in organisms with smaller somites. Since regulation of **cell** motility (as modeled in this work), the strength of the segmentation-clock synchronization mechanism and the rapidity of expression and accumulation of adhesion molecules at the membrane following **cell**-**type** determination all affect noise levels, we predict that measurement of these quantities in a variety of species will show stricter control in smaller embryos.

Many mathematical/computational models of somitogenesis have assumed that some external factor such as FGF8 or Wnt3a signaling modulates the period of the segmentation clock (Tiedemann, Schneltzer *et al.* 2007; Santillán and Mackey 2008; Gibb, Zagorska *et al.* 2009; Morelli, Ares *et al.* 2009; Uriu, Morishita *et al.* 2009). While we did not impose such an assumption, Lfng travelling stripes arose spontaneously in our simulations via an emergent version of this mechanism. Because the period-modulating Wnt3a signal in our model is **cell**-autonomous, the traveling stripes of Lfng expression in our simulations are nearly **cell**-autonomous pseudo-waves. Our model predicts, therefore, that excising segments of the PSM will not disrupt traveling Lfng stripes. Similarly, the model predicts that tissue inversion experiments should result in reversed-direction traveling stripes of mRNA/protein expression in the inverted tissue that closely resemble normal traveling stripes apart from their direction. The model also predicts that imposing uniform Wnt3a signaling across the PSM by knocking out endogenous Wnt3a signaling and applying a uniform concentration of Wnt3a to the tissue will eradicate travelling-stripe expression of segmentation-clock genes, but not their oscillations or the formation of normal somites (see **Figure 4.10** (**D**), **Figure 4.12**).

Even if different mechanisms were responsible for the traveling stripes of gene expression in the PSM, as long as the mechanisms are cell-autonomous, *e.g.*, if the segmentation-clock oscillations gradually slow as a function of the cells' residence time in the PSM (as occurs in our model due to the decay of Wnt3a) or if the stripes result from conserved clock phase differences between cells as they enter the posterior PSM, then our predictions for surgical experiments would not distinguish between such mechanisms and those in our model, but our predictions for the effect of Wnt3a manipulation would change. If the expression stripes are propagating waves arising from the physical boundary conditions and segmentation-clock coupling between cells, altering Wnt3a expression will not have the effect our model predicts, and surgically manipulating the PSM would disrupt the waves.

5.3 Limitations of the composite model

Because our model does not include mechanisms for border correction beyond adhesion-mediated cell sorting, it is sensitive to over-mixing of **cell** types that can arise either as a consequence of initial mis-determination (which can be due to local segmentation-clock desynchronization or to significantly different levels of FGF8 among neighboring **cells** at the determination front) or from extensive **cell** diffusion in the interval between determination and differentiation. The impact of over-mixing is particularly pronounced in a 2D model, in which **cell** motion is constrained to a single plane. In 2D a **cell** is more likely to become surrounded on all sides by **cells** of an inappropriate type, trapping the mislocated **cell** in a local adhesion-energy minimum and impeding correction of the mistake by adhesion-driven sorting.

Experiments suggest that Eph-ephrin signaling, epithelialization and cell-ECM interactions all play significant roles in forming the intersomitic gap (Saga and Takeda 2001; Sato, Yasuda *et al.*

2002; Martins, Rifes *et al.* 2009; Watanabe, Sato *et al.* 2009). Our current model omits detailed submodels of these mechanisms, relying solely on differential adhesion between EphA4-, ephrinB2- and N-cadherin-expressing **cells** to form and maintain the gap. These mechanisms likely operate in concert; for instance, localization of EphA4-expressing cells in the anterior somite compartment and anterior tip of the PSM and ephrinB2-expressing cells in the posterior somite compartment arises from mechanisms included in our model and facilitates Eph-ephrin signaling across the presumptive intersomitic gap that is omitted from our model. The occasional failure of our model to produce perfectly separated **somites** in situations with increased noise due to high cell motility or perturbations in the segmentation clock is almost certainly due to the lack of these additional mechanisms. That our model does reproduce the major events in somitogenesis in a realistic fashion suggests that the mechanisms included in the model are the most significant ones in tissue patterning *in vivo*.

5.4 Future extensions of the composite model

The present work primarily tests the ability of a composite model composed of existing submodels of somitogenesis at different scales to reproduce key dynamical and morphological features of *in vivo* somitogenesis, and investigates the necessary interplay between those submodels. It does not explore a wide range of alternate mechanisms or submodels at each scale. The groundwork laid here and the modular nature of the composite model will allow us to extend the model to perform more focused explorations of particular mechanisms important to somitogenesis.

We excluded a detailed submodel of RA signaling and RA-FGF8 interaction from the current composite model, choosing instead to model the simplest-case scenario involving morphogen-segmentation-clock interaction and threshold-positioned developmental fronts. In addition to other functions, RA probably refines determination and/or differentiation front positioning. Mutually antagonistic RA and FGF8 signaling may lead to a “bistability window” in which cells can abruptly switch from states of high FGF8 and low RA signaling to states of high RA and low FGF8 signaling, possibly leading cohorts of cells to simultaneously undergo determination or differentiation (Goldbeter, Gonze *et al.* 2007; Santillán and Mackey 2008). The next generation of our composite model will include submodels for RA production and distribution similar in their levels of detail to those for FGF8 in the current integrated model, as well as submodels of the interactions between modeled morphogens, so that we may address such questions as how having groups of cells determine simultaneously affects the necessity of striped gene expression in the anterior PSM and whether RA increases the robustness of determination and differentiation positioning to perturbations due to cell motion and noise in the segmentation clock.

Other extensions of the composite model will include submodels of FGF signaling-dependent cell motility, as described by Delfini *et al.* (Delfini, Dubrulle *et al.* 2005); gradual changes in cell adhesion after determination, which could change the reorganization dynamics of cells after differentiation; and the addition of somite epithelialization, which may improve the realism of simulated somite border dynamics and result in more realistic somite morphologies.

Finally, the most limiting assumption of the current integrated model is that a 2D AP-by-ML slice along the DV midline serves as a proxy for 3D somitogenesis. Extending the model into three dimensions will allow us to address the impact of dimensionality on our key results (*e.g.*, by assessing the relative sensitivity of 2D and 3D models dynamics to changes in **cell** motility and adhesion parameters, and the ability of segmentation clocks in neighboring **cells** to synchronize). A 3D model will also allow us to address DV asymmetries both in boundary conditions and the emergent structure of somites.

5.5 Conclusion

Multi-scale modeling of highly conserved developmental processes such as somitogenesis is a necessary first step in developing predictive models of how potential toxins or therapies affect development. Even if the interactions of a chemical agent are predictable on the molecular scale, we need multi-scale models to predict how these molecular perturbations will affect tissue-, organ- and organism-level dynamics and morphologies, *e.g.*, somitogenesis and later segmental development. Ultimately, understanding how perturbations at a single scale propagate to other scales will be essential to evaluating whether the perturbations are likely to have therapeutic or dangerous effects. While such powerful predictive tools are still some way in the future, they will only be possible with the aid of flexible, well-crafted, well-understood multi-scale models.

REFERENCES

Alber, M., N. Chen, et al. (2006). "Multiscale dynamics of biological cells with chemotactic interactions: from a discrete stochastic model to a continuous description." Phys Rev E Stat Nonlin Soft Matter Phys **73**(5 Pt 1): 051901.

Armstrong, N., K. Painter, et al. (2009). "Adding Adhesion to a Chemical Signaling Model for Somite Formation." Bulletin of Mathematical Biology **71**(1): 1-24.

Aulehla, A., C. Wehrle, et al. (2003). "Wnt3a plays a major role in the segmentation clock controlling somitogenesis." Dev Cell **4**(3): 395-406.

Baker, R. E., S. Schnell, et al. (2006). "A clock and wavefront mechanism for somite formation." Dev Biol **293**(1): 116-126.

Baker, R. E., S. Schnell, et al. (2006). "A clock and wavefront mechanism for somite formation." Developmental Biology **293**(1): 116-126.

Baker, R. E., S. Schnell, et al. (2006). "A mathematical investigation of a Clock and Wavefront model for somitogenesis." Journal of Mathematical Biology **52**(4): 458-482.

Baker, R. E., S. Schnell, et al. (2007). "A mathematical basis for the clock and wavefront model for somitogenesis." Developmental Biology **306**(1): 400-400.

Baker, R. E., S. Schnell, et al. (2008). "Mathematical models for somite formation." Multiscale Modeling of Developmental Systems **81**: 183-203.

Baker, R. K. and P. B. Antin (2003). "Ephs and ephrins during early stages of chick embryogenesis." Dev Dyn **228**(1): 128-142.

Barrantes, I. B., A. J. Elia, et al. (1999). "Interaction between Notch signalling and Lunatic fringe during somite boundary formation in the mouse." Curr Biol **9**(9): 470-480.

Barrios, A., R. J. Poole, et al. (2003). "Eph/Ephrin signaling regulates the mesenchymal-to-epithelial transition of the paraxial mesoderm during somite morphogenesis." Curr Biol **13**(18): 1571-1582.

Benazeraf, B., P. Francois, et al. (2010). "A random cell motility gradient downstream of FGF controls elongation of an amniote embryo." Nature **466**(7303): 248-252.

Blentic, A., E. Gale, et al. (2003). "Retinoic acid signalling centres in the avian embryo identified by sites of expression of synthesising and catabolising enzymes." Dev Dyn **227**(1): 114-127.

Burgess, R., P. Cserjesi, et al. (1995). "Paraxis: a basic helix-loop-helix protein expressed in paraxial mesoderm and developing somites." Dev Biol **168**(2): 296-306.

Burgess, R., A. Rawls, et al. (1996). "Requirement of the paraxis gene for somite formation and musculoskeletal patterning." Nature **384**(6609): 570-573.

Christian, J. L. (2000). "BMP, Wnt and Hedgehog signals: how far can they go?" Current Opinion in Cell Biology **12**(2): 244-249.

Conlon, R. A., A. G. Reaume, et al. (1995). "Notch1 is required for the coordinate segmentation of somites." Development **121**(5): 1533-1545.

Cooke, J. and E. C. Zeeman (1976). "A clock and wavefront model for control of the number of repeated structures during animal morphogenesis." J Theor Biol **58**(2): 455-476.

Czirok, A., E. A. Zamir, et al. (2006). "Extracellular matrix macroassembly dynamics in early vertebrate embryos." Curr Top Dev Biol **73**: 237-258.

Delfini, M. C., J. Dubrulle, et al. (2005). "Control of the segmentation process by graded MAPK/ERK activation in the chick embryo." Proc Natl Acad Sci U S A **102**(32): 11343-11348.

Dequeant, M. L., E. Glynn, et al. (2006). "A complex oscillating network of signaling genes underlies the mouse segmentation clock." Science **314**(5805): 1595-1598.
Dierick, H. and A. Bejsovec (1999). "Cellular mechanisms of wingless/Wnt signal transduction." Current Topics in Developmental Biology, Vol 43 **43**: 153-190.
Diez del Corral, R., I. Olivera-Martinez, et al. (2003). "Opposing FGF and retinoid pathways control ventral neural pattern, neuronal differentiation, and segmentation during body axis extension." Neuron **40**(1): 65-79.
Diez del Corral, R. and K. G. Storey (2004). "Opposing FGF and retinoid pathways: a signalling switch that controls differentiation and patterning onset in the extending vertebrate body axis." Bioessays **26**(8): 857-869.
Duband, J. L., S. Dufour, et al. (1987). "Adhesion molecules during somitogenesis in the avian embryo." J Cell Biol **104**(5): 1361-1374.
Dubrulle, J., M. J. McGrew, et al. (2001). "FGF signaling controls somite boundary position and regulates segmentation clock control of spatiotemporal Hox gene activation." Cell **106**(2): 219-232.
Dubrulle, J. and O. Pourquie (2004). "fgf8 mRNA decay establishes a gradient that couples axial elongation to patterning in the vertebrate embryo." Nature **427**(6973): 419-422.
Durbin, L., C. Brennan, et al. (1998). "Eph signaling is required for segmentation and differentiation of the somites." Genes Dev **12**(19): 3096-3109.
Espinosa, L., J. Ingles-Esteve, et al. (2003). "Phosphorylation by glycogen synthase kinase-3 beta down-regulates Notch activity, a link for Notch and Wnt pathways." J Biol Chem **278**(34): 32227-32235.
Evrard, Y. A., Y. Lun, et al. (1998). "lunatic fringe is an essential mediator of somite segmentation and patterning." Nature **394**(6691): 377-381.
Forsberg, H., F. Crozet, et al. (1998). "Waves of mouse Lunatic fringe expression, in four-hour cycles at two-hour intervals, precede somite boundary formation." Curr Biol **8**(18): 1027-1030.
Gibb, S., A. Zagorska, et al. (2009). "Interfering with Wnt signalling alters the periodicity of the segmentation clock." Dev Biol **330**(1): 21-31.
Giudicelli, F., E. M. Ozbudak, et al. (2007). "Setting the tempo in development: an investigation of the zebrafish somite clock mechanism." PLoS Biol **5**(6): e150.
Glazier, J. A. and F. Graner (1993). "Simulation of the differential adhesion driven rearrangement of biological cells." Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics **47**(3): 2128-2154.
Glazier, J. A., Y. Zhang, et al. (2008). "Coordinated action of N-CAM, N-cadherin, EphA4, and ephrinB2 translates genetic prepatterns into structure during somitogenesis in chick." Curr Top Dev Biol **81**: 205-247.
Goldbeter, A., D. Gonze, et al. (2007). "Sharp developmental thresholds defined through bistability by antagonistic gradients of retinoic acid and FGF signaling." Dev Dyn **236**(6): 1495-1508.
Goldbeter, A. and O. Pourquie (2008). "Modeling the segmentation clock as a network of coupled oscillations in the Notch, Wnt and FGF signaling pathways." J Theor Biol **252**(3): 574-585.
Gomez, C., E. M. Ozbudak, et al. (2008). "Control of segment number in vertebrate embryos." Nature **454**(7202): 335-339.
Gossler, A. and M. Hrabe de Angelis (1998). "Somitogenesis." Curr Top Dev Biol **38**: 225-287.
Grima, R. and S. Schnell (2008). "Modelling reaction kinetics inside cells." Essays in Biochemistry: Systems Biology, Vol 45 **45**: 41-56.

Harbott, L. K. and C. D. Nobes (2005). "A key role for Abl family kinases in EphA receptor-mediated growth cone collapse." Mol Cell Neurosci **30**(1): 1-11.

Horikawa, K., K. Ishimatsu, et al. (2006). "Noise-resistant and synchronized oscillation of the segmentation clock." Nature **441**(7094): 719-723.

Horikawa, K., G. Radice, et al. (1999). "Adhesive subdivisions intrinsic to the epithelial somites." Dev Biol **215**(2): 182-189.

Hrabe de Angelis, M., J. McIntyre, 2nd, et al. (1997). "Maintenance of somite borders in mice requires the Delta homologue DII1." Nature **386**(6626): 717-721.

Jensen, P. B., L. Pedersen, et al. (2010). "A Wnt Oscillator Model for Somitogenesis." Biophysical Journal **98**(6): 943-950.

Jiang, Y. J., B. L. Aerne, et al. (2000). "Notch signalling and the synchronization of the somite segmentation clock." Nature **408**(6811): 475-479.

Johnson, J., J. Rhee, et al. (2001). "The anterior/posterior polarity of somites is disrupted in paraxis-deficient mice." Dev Biol **229**(1): 176-187.

Knezevic, V., R. De Santo, et al. (1998). "Continuing organizer function during chick tail development." Development **125**(10): 1791-1801.

Kulesa, P. M. and S. E. Fraser (2002). "Cell dynamics during somite boundary formation revealed by time-lapse analysis." Science **298**(5595): 991-995.

Lewis, J. (2003). "Autoinhibition with transcriptional delay: a simple mechanism for the zebrafish somitogenesis oscillator." Curr Biol **13**(16): 1398-1408.

Linker, C., C. Lesbros, et al. (2005). "beta-Catenin-dependent Wnt signalling controls the epithelial organisation of somites through the activation of paraxis." Development **132**(17): 3895-3905.

Mara, A., J. Schroeder, et al. (2007). "Priming, initiation and synchronization of the segmentation clock by deltaD and deltaC." Nat Cell Biol **9**(5): 523-530.

Martins, G. G., P. Rifes, et al. (2009). "Dynamic 3D cell rearrangements guided by a fibronectin matrix underlie somitogenesis." PLoS One **4**(10): e7429.

McGrew, M. J., J. K. Dale, et al. (1998). "The lunatic fringe gene is a target of the molecular clock linked to somite segmentation in avian embryos." Curr Biol **8**(17): 979-982.

Mellitzer, G., Q. Xu, et al. (1999). "Eph receptors and ephrins restrict cell intermingling and communication." Nature **400**(6739): 77-81.

Mombach, J. C., J. A. Glazier, et al. (1995). "Quantitative comparison between differential adhesion models and cell sorting in the presence and absence of fluctuations." Phys Rev Lett **75**(11): 2244-2247.

Morelli, L. G., S. Ares, et al. (2009). "Delayed coupling theory of vertebrate segmentation." HFSP J **3**(1): 55-66.

Morimoto, M., Y. Takahashi, et al. (2005). "The Mesp2 transcription factor establishes segmental borders by suppressing Notch activity." Nature **435**(7040): 354-359.

Nakajima, Y., M. Morimoto, et al. (2006). "Identification of Epha4 enhancer required for segmental expression and the regulation by Mesp2." Development **133**(13): 2517-2525.

Nelson, W. J. and R. Nusse (2004). "Convergence of Wnt, beta-catenin, and cadherin pathways." Science **303**(5663): 1483-1487.

Niwa, Y., Y. Masamizu, et al. (2007). "The initiation and propagation of Hes7 oscillation are cooperatively regulated by Fgf and notch signaling in the somite segmentation clock." Dev Cell **13**(2): 298-304.

Oginuma, M., Y. Niwa, et al. (2008). "Mesp2 and Tbx6 cooperatively create periodic patterns coupled with the clock machinery during mouse somitogenesis." Development **135**(15): 2555-2562.

Ozawa, M., H. Baribault, et al. (1989). "The cytoplasmic domain of the cell adhesion molecule uvomorulin associates with three independent proteins structurally related in different species." EMBO J **8**(6): 1711-1717.

Ozbudak, E. M. and J. Lewis (2008). "Notch signalling synchronizes the zebrafish segmentation clock but is not needed to create somite boundaries." PLoS Genet **4**(2): e15.

Palmeirim, I., J. Dubrulle, et al. (1998). "Uncoupling segmentation and somitogenesis in the chick presomitic mesoderm." Dev Genet **23**(1): 77-85.

Palmeirim, I., D. Henrique, et al. (1997). "Avian hairy gene expression identifies a molecular clock linked to vertebrate segmentation and somitogenesis." Cell **91**(5): 639-648.

Pfeiffer, S. and J. P. Vincent (1999). "Signalling at a distance: Transport of Wingless in the embryonic epidermis of Drosophila." Seminars in Cell & Developmental Biology **10**(3): 303-309.

Popławski, N. J., A. Shirinifard, et al. (2008). "Simulation of single-species bacterial-biofilm growth using the Glazier-Graner-Hogeweg model and the CompuCell3D modeling environment." Math Biosci Eng **5**(2): 355-388.

Saga, Y. (2007). "Segmental border is defined by the key transcription factor Mesp2, by means of the suppression of Notch activity." Dev Dyn **236**(6): 1450-1455.

Saga, Y. and H. Takeda (2001). "The making of the somite: molecular events in vertebrate segmentation." Nat Rev Genet **2**(11): 835-845.

Santillán, M. and M. C. Mackey (2008). "A Proposed Mechanism for the Interaction of the Segmentation Clock and the Determination Front in Somitogenesis." PLoS One **3**(2): e1561.

Sato, Y., K. Yasuda, et al. (2002). "Morphological boundary forms by a novel inductive event mediated by Lunatic fringe and Notch during somitic segmentation." Development **129**(15): 3633-3644.

Scholpp, S. and M. Brand (2004). "Endocytosis controls spreading and effective signaling range of Fgf8 protein." Current Biology **14**(20): 1834-1841.

Shirinifard, A., J. S. Gens, et al. (2009). "3D multi-cell simulation of tumor growth and angiogenesis." PLoS One **4**(10): e7190.

Stern, C. D. (1992). "Vertebrate gastrulation." Curr Opin Genet Dev **2**(4): 556-561.

Stern, C. D., J. Charite, et al. (2006). "Head-tail patterning of the vertebrate embryo: one, two or many unresolved problems?" International Journal of Developmental Biology **50**(1): 3-15.

Stern, C. D., S. E. Fraser, et al. (1988). "A cell lineage analysis of segmentation in the chick embryo." Development **104 Suppl**: 231-244.

Swat, M. H., S. D. Hester, et al. (2009). "Multicell simulations of development and disease using the CompuCell3D simulation environment." Methods Mol Biol **500**: 361-428.

Takahashi, Y., T. Inoue, et al. (2003). "Feedback loops comprising Dll1, Dll3 and Mesp2, and differential involvement of Psen1 are essential for rostrocaudal patterning of somites." Development **130**(18): 4259-4268.

Takahashi, Y., K. Koizumi, et al. (2000). "Mesp2 initiates somite segmentation through the Notch signalling pathway." Nat Genet **25**(4): 390-396.

Tiedemann, H. B., E. Schneltzer, et al. (2007). "Cell-based simulation of dynamic expression patterns in the presomitic mesoderm." J Theor Biol **248**(1): 120-129.

Uriu, K., Y. Morishita, et al. (2009). "Traveling wave formation in vertebrate segmentation." Journal of Theoretical Biology **257**(3): 385-396.

Uriu, K., Y. Morishita, et al. (2010). "Synchronized oscillation of the segmentation clock gene in vertebrate development." Journal of Mathematical Biology **61**(2): 207-229.

Wahl, M. B., C. Deng, et al. (2007). "FGF signaling acts upstream of the NOTCH and WNT signaling pathways to control segmentation clock oscillations in mouse somitogenesis." Development **134**(22): 4033-4041.

Watanabe, T., Y. Sato, et al. (2009). "EphrinB2 coordinates the formation of a morphological boundary and cell epithelialization during somite segmentation." Proc Natl Acad Sci U S A **106**(18): 7467-7472.

Yasuhiko, Y., S. Haraguchi, et al. (2006). "Tbx6-mediated Notch signaling controls somite-specific Mesp2 expression." Proc Natl Acad Sci U S A **103**(10): 3651-3656.

Appendix A

SEGMENTATION CLOCK NETWORK SUBMODEL EQUATIONS

The ordinary differential equations we used to model the segmentation clock are based on those used by Goldbeter and Pourquié, with a few additions, mainly to the DeltaNotch pathway, to account for cell-cell synchronization and increase connection between the internal pathways.

Notch loop:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = \varepsilon\left(v_{\mathrm{sN}} - v_{\mathrm{dN}} \cdot \frac{N}{K_{\mathrm{dN}} + N} - N_{\mathrm{Sig}} \right) \tag{A1}$$

$$\frac{\mathrm{d}N_{\mathrm{a}}}{\mathrm{d}t} = \varepsilon\left(N_{\mathrm{Sig}} - v_{\mathrm{dNa}} \cdot \frac{N_{\mathrm{a}}}{K_{\mathrm{dNa}} + N_{\mathrm{a}}} - V_{\mathrm{tr}} \right) \tag{A2}$$

$$\frac{\mathrm{d}N_{\mathrm{an}}}{\mathrm{d}t} = \varepsilon\left(V_{\mathrm{tr}} - v_{\mathrm{dNan}} \cdot \frac{N_{\mathrm{an}}}{K_{\mathrm{dNan}} + N_{\mathrm{an}}} \right) \tag{A3}$$

$$\frac{\mathrm{d}MF}{\mathrm{d}t} = \varepsilon\left(v_{\mathrm{sF}} \cdot \frac{(N_{\mathrm{an}} - N_{\mathrm{ap}})^2}{K_{\mathrm{A}}{}^2 + (N_{\mathrm{an}} - N_{\mathrm{ap}})^2} - v_{\mathrm{mF}} \cdot \frac{MF}{K_{dmF} + MF} \right) \tag{A4}$$

$$\frac{\mathrm{d}F}{\mathrm{d}t} = \varepsilon\left(k_{\mathrm{sF}} \cdot MF - v_{\mathrm{dF}} \cdot \frac{F}{K_{\mathrm{dF}} + F} \right) \tag{A5}$$

$$\frac{\mathrm{d}MDMF}{\mathrm{d}t} = \varepsilon\left(v_{\mathrm{s0MDMF}} \cdot \frac{(N_{\mathrm{an}} - N_{\mathrm{ap}})^2}{K_{\mathrm{aNDMF}}^2 + (N_{\mathrm{an}} - N_{\mathrm{ap}})^2} - v_{\mathrm{dmDMF}} \cdot \frac{MDMF}{K_{\mathrm{dmDMF}} + MDMF} \right) \tag{A6}$$

$$\frac{\mathrm{d}DMF}{\mathrm{d}t} = \varepsilon\left(k_{\mathrm{sDMF}} \cdot MDMF - v_{\mathrm{dDMF}} \cdot \frac{DMF}{K_{\mathrm{dDMF}} + DMF} \right) \tag{A7}$$

$$\frac{\mathrm{d}DL_{\mathrm{c}}}{\mathrm{d}t} = \varepsilon\left(k_{\mathrm{sDL}} \cdot MDL \cdot \frac{(N_{\mathrm{an}} - N_{\mathrm{ap}})}{K_{\mathrm{Nan}} + (N_{\mathrm{an}} - N_{\mathrm{ap}})} - k_{\mathrm{tDL}} \cdot DL_{\mathrm{c}} - v_{\mathrm{dDLc}} \cdot \frac{DL_{\mathrm{c}}}{K_{\mathrm{dDLc}} + DL_{\mathrm{c}}} \right) \tag{A8}$$

$$\frac{\mathrm{d}DL_{\mathrm{m}}}{\mathrm{d}t} = \varepsilon\left(k_{\mathrm{tDL}} \cdot DL_{\mathrm{c}} - v_{\mathrm{dDLm}} \cdot \frac{DL_{\mathrm{m}}}{K_{\mathrm{dDLm}} + DL_{\mathrm{m}}}\right) \tag{A9}$$

$$N_{\mathrm{Sig}} = k_{\mathrm{c}} \cdot N \cdot \frac{K_{\mathrm{IF}}^2}{K_{\mathrm{IF}}^2 + F^2} \cdot \frac{DL_{\mathrm{mSig}}}{K_{\mathrm{aDL}} + DL_{\mathrm{mSig}}} \tag{A10}$$

$$DL_{\mathrm{mSig}} = \frac{DL_{\mathrm{mExt}}}{K_{\mathrm{DL}} + DL_{\mathrm{mExt}}} \tag{A11}$$

$$V_{\mathrm{tr}} = k_{\mathrm{t1}} \cdot N_{\mathrm{a}} - k_{\mathrm{t2}} \cdot N_{\mathrm{an}} \tag{A12}$$

$$N_{\mathrm{ap}} = v_{\mathrm{Nap}} \cdot N_{\mathrm{an}} \cdot \frac{K}{K_{\mathrm{pN}} + K} \tag{A13}$$

Wnt loop:

$$\frac{\mathrm{d}K}{\mathrm{d}t} = \theta \cdot V_1 \tag{A14}$$

$$\frac{\mathrm{d}B}{\mathrm{d}t} = \theta\left(v_{\mathrm{sB}} - V_{\mathrm{K}} \cdot \frac{AK}{K_{\mathrm{t}}} + V_{\mathrm{P}} + V_2 - k_{\mathrm{d1}} \cdot B\right) \tag{A15}$$

$$\frac{\mathrm{d}B_{\mathrm{p}}}{\mathrm{d}t} = \theta\left(V_{\mathrm{K}} \cdot \frac{AK}{K_{\mathrm{t}}} - V_{\mathrm{P}} - k_{\mathrm{d2}} \cdot B_{\mathrm{p}}\right) \tag{A16}$$

$$\frac{\mathrm{d}B_{\mathrm{N}}}{\mathrm{d}t} = -\theta \cdot V_2 \tag{A17}$$

$$\frac{\mathrm{d}MAx}{\mathrm{d}t} = \theta\left(v_0 + v_{\mathrm{MB}} \cdot \frac{B_{\mathrm{N}}^2}{K_{\mathrm{aB}}^2 + B_{\mathrm{N}}^2} + v_{\mathrm{MXa}} \cdot \frac{X_{\mathrm{a}}^2}{K_{\mathrm{aXa}}^2 + X_{\mathrm{a}}^2} - v_{\mathrm{md}} \cdot \frac{MAx}{K_{\mathrm{md}} + MAx}\right) \tag{A18}$$

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \theta\left(k_{\mathrm{sAx}} \cdot MAx - v_{\mathrm{dAx}} \cdot \frac{A}{K_{\mathrm{dAx}} + A} + V_1\right) \tag{A19}$$

$$V_1 = d_1 \cdot AK - a_1 \cdot A \cdot K \tag{A20}$$

$$AK = K_{\mathrm{t}} - K \tag{A21}$$

$$V_{\mathrm{K}} = V_{\mathrm{MK}} \cdot \frac{K_{\mathrm{ID}}}{K_{\mathrm{ID}} + D} \cdot \frac{B}{K_1 + B} \tag{A22}$$

$$V_{\mathrm{P}} = V_{\mathrm{MP}} \cdot \frac{B_{\mathrm{p}}}{K_2 + B_{\mathrm{p}}} \tag{A23}$$

$$V_2 = k_{\mathrm{t4}} \cdot B_{\mathrm{N}} - k_{\mathrm{t3}} \cdot B \tag{A24}$$

FGF loop:

$$\frac{\mathrm{d}Ras_{\mathrm{a}}}{\mathrm{d}t} = \eta\left(V_{\mathrm{MaRas}} \cdot \frac{Fgf^2}{K_{\mathrm{aFgf}}^2 + Fgf^2} \cdot \frac{Ras_{\mathrm{i}}}{K_{\mathrm{aRas}} + Ras_{\mathrm{i}}} - V_{\mathrm{MdRas}} \cdot \frac{Ras_{\mathrm{a}}}{K_{\mathrm{dRas}} + Ras_{\mathrm{a}}}\right) \tag{A25}$$

$$\frac{\mathrm{d}ERK_{\mathrm{a}}}{\mathrm{d}t} = \eta\left(V_{\mathrm{MaErk}} \cdot \frac{Ras_{\mathrm{a}}}{Ras_{\mathrm{t}}} \cdot \frac{ERK_{\mathrm{i}}}{K_{\mathrm{aErk}} + ERK_{\mathrm{i}}} - k_{\mathrm{cDusp}} \cdot Dusp \cdot \frac{ERK_{\mathrm{a}}}{K_{\mathrm{dErk}} + ERK_{\mathrm{a}}}\right) \tag{A26}$$

$$\frac{\mathrm{d}X_{\mathrm{a}}}{\mathrm{d}t} = \eta\left(V_{\mathrm{MaX}} \cdot \frac{ERK_{\mathrm{a}}}{ERK_{\mathrm{t}}} \cdot \frac{X_{\mathrm{i}}}{K_{\mathrm{aX}} + X_{\mathrm{i}}} - V_{\mathrm{MdX}} \cdot \frac{X_{\mathrm{a}}}{K_{\mathrm{dX}} + X_{\mathrm{a}}}\right) \tag{A27}$$

$$\frac{\mathrm{d}MDusp}{\mathrm{d}t} = \eta\left(V_{\mathrm{MDusp}} - V_{\mathrm{dMDusp}}\right) \tag{A28}$$

$$\frac{\mathrm{d}Dusp}{\mathrm{d}t} = \eta\left(k_{\mathrm{sDusp}} \cdot MDusp - V_{\mathrm{dDusp}} \cdot \frac{Dusp}{K_{\mathrm{dDusp}} + Dusp}\right) \tag{A29}$$

$$Ras_{\mathrm{i}} = Ras_{\mathrm{t}} - Ras_{\mathrm{a}} \tag{A30}$$

$$ERK_{\mathrm{i}} = ERK_{\mathrm{t}} - ERK_{\mathrm{a}} \tag{A31}$$

$$X_{\mathrm{i}} = X_{\mathrm{t}} - X_{\mathrm{a}} \tag{A32}$$

$$V_{\mathrm{dMDusp}} = V_{\mathrm{MdMDusp}} \cdot \frac{MDusp}{K_{\mathrm{dMDusp}} + MDusp} \tag{A33}$$

$$V_{\mathrm{MDusp}} = V_{\mathrm{MsMDusp}} \cdot \frac{X_a^2}{K_{\mathrm{aMDusp}}^2 + X_a^2}\left(v_{\mathrm{DuspDMF}} \cdot \frac{K_{\mathrm{IMDusp}}}{K_{\mathrm{IMDusp}} + DMF} + v_{\mathrm{DuspX}}\right) \tag{A34}$$

Appendix B

SEGMENTATION CLOCK NETWORK SUBMODEL PARAMETERS AND INITIAL CONDITIONS

Parameters and variables listed in **bold** are our additions to the Goldbeter-Pourquié model.

Parameters:

Notch loop

Parameter	Description	Equation	Value
v_{sN}	Maximum rate of Notch synthesis	A1	0.23 nM/min
v_{dN}	Maximum rate of Notch degradation	A1	2.82 nM/min
K_{dN}	Michaelis constant for Notch degradation	A1	1.4 nM
k_c	Rate constant for Notch cleavage	A10	3.45 min^{-1}
v_{dNa}	Maximum rate of cytoplasmic NICD degradation	A2	0.01 nM/min
K_{dNa}	Michaelis constant for cytoplasmic NICD degradation	A2	0.001 nM
v_{dNan}	Maximum rate of nuclear NICD degradation	A3	0.1 nM/min
K_{dNan}	Michaelis constant for nuclear NICD degradation	A3	0.001 nM
K_{IF}	Threshold constant for inhibition of Notch cleavage by Lfng	A10	0.45 nM
K_{aDL}	Delta signaling threshold constant for Notch cleavage	A10	0.035 nM
k_{t1}	Rate constant for NICD entry into nucleus	A12	0.1 min^{-1}
k_{t2}	Rate constant for NICD exit from the nucleus	A12	0.1 min^{-1}
$\boldsymbol{v_{Nap}}$	**Maximum fraction of NICD phosphorylated**	**A13**	**1.0**
$\boldsymbol{K_{pN}}$	**Threshold constant for Gsk3β-mediated NICD phosphorylation**	**A13**	**2.5 nM**
v_{sF}	Maximum rate of Lfng transcription	A4	3.24 nM/min
K_A	Threshold constant for activation of Lfng gene by NICD	A4	0.05 nM
v_{mF}	Maximum rate of Lfng mRNA degradation	A4	1.92 nM/min
K_{dmF}	Michaelis constant for Lfng mRNA degradation	A4	0.768 nM
k_{sF}	Rate constant for Lfng protein synthesis	A5	0.3 min^{-1}
v_{dF}	Maximum rate of Lfng protein degradation	A5	0.39 nM/min

K_{dF}	Michaelis constant for Lfng protein degradation	A5	0.37 nM
$\boldsymbol{v}_{\mathbf{s0MDMF}}$	**Maximum rate of DMF transcription**	**A6**	**0.497 nM/min**
$\boldsymbol{K}_{\mathbf{aNDMF}}$	**Michaelis constant for Notch activation of DMF**	**A6**	**0.05 nM**
$\boldsymbol{v}_{\mathbf{dmDMF}}$	**Maximum rate of DMF mRNA degradation**	**A6**	**0.314 nM/min**
$\boldsymbol{K}_{\mathbf{dmDMF}}$	**Michaelis constant for DMF mRNA degradation**	**A6**	**0.4 nM**
$\boldsymbol{k}_{\mathbf{sDMF}}$	**Rate constant for translation of DMF**	**A7**	**0.1047 min^{-1}**
$\boldsymbol{v}_{\mathbf{dDMF}}$	**Maximum rate of DMF protein degradation**	**A7**	**0.209 nM/min**
$\boldsymbol{K}_{\mathbf{dDMF}}$	**Michaelis constant for DMF protein degradation**	**A7**	**0.5 nM**
$\boldsymbol{k}_{\mathbf{sDL}}$	**Rate constant for Delta protein synthesis**	**A8**	**0.75 min^{-1}**
$\boldsymbol{MDL}$	**Concentration of Delta mRNA**	**A8**	**0.5 nM**
$\boldsymbol{K}_{\mathbf{Nan}}$	**Michaelis constant for NICD stimulation of Delta translation**	**A8**	**0.04 nM**
$\boldsymbol{k}_{\mathbf{tDL}}$	**Rate constant for Delta protein transport to membrane**	**A8**	**0.5 min^{-1}**
$\boldsymbol{v}_{\mathbf{dDLc}}$	**Maximum rate of Delta protein degradation in cytoplasm**	**A8**	**0.5 nM/min**
$\boldsymbol{K}_{\mathbf{dDLc}}$	**Michaelis constant for Delta protein degradation in cytoplasm**	**A8**	**0.5 nM**
$\boldsymbol{v}_{\mathbf{dDLm}}$	**Maximum rate of Delta protein degradation on membrane**	**A9**	**0.5 nM/min**
$\boldsymbol{K}_{\mathbf{dDLm}}$	**Michaelis constant for Delta protein degradation on membrane**	**A9**	**0.5 nM**
$\boldsymbol{K}_{\mathbf{DL}}$	**Threshold constant for Delta signaling**	**A11**	**0.08 nM**
ε	Scaling factor for Notch loop	A1-9	0.43

Wnt loop

Parameter	Description	Equation	Value
a_1	Rate constant for Gsk3β-Axin2 binding	A20	1.8 nM^{-1} min^{-1}
d_1	Rate constant for Gsk3β-Axin2 dissociation	A20	0.1 min^{-1}
v_{sB}	Maximum rate of β-catenin synthesis	A15	0.087 nM/min
k_{t3}	Rate constant for β-catenin entry into the nucleus	A24	0.7 min^{-1}
k_{t4}	Rate constant for β-catenin exit from the nucleus	A24	1.5 min^{-1}
V_{MK}	Maximum rate of Gsk3β-mediated β-catenin phosphorylation	A22	4.5 nM/min

K_t	Total Gsk3β concentration	A15-16, A21	3.0 nM
K_{ID}	Threshold constant for Dsh inhibition of β-catenin phosphorylation	A22	0.5 nM
K_1	Michaelis constant for β-catenin phosphorylation	A22	0.28 nM
V_{MP}	Maximum rate of β-catenin dephosphorylation	A23	1.0 nM/min
K_2	Michaelis constant for β-catenin dephosphorylation	A23	0.03 nM
k_{d1}	Rate constant for unphosphorylated β-catenin degradation	A15	0.0 min^{-1}
k_{d2}	Rate constant for phosphorylated β-catenin degradation	A16	7.062 min^{-1}
v_0	Basal rate of Axin2 transcription	A18	0.06 nM/min
v_{MB}	Maximum rate of β-catenin activated Axin2 transcription	A18	1.64 nM/min
K_{aB}	Threshold constant for β-catenin activation of Axin2	A18	0.7 nM
v_{md}	Maximum rate of Axin2 mRNA degradation	A18	0.8 nM/min
K_{md}	Michaelis constant for Axin2 mRNA degradation	A18	0.48 nM
v_{MXa}	Maximum rate of Xa activated Axin2 transcription	A18	0.5 nM/min
K_{aXa}	Threshold constant for Xa activation of Axin2	A18	0.05 nM
k_{sAx}	Rate constant for Axin2 translation	A19	0.02 min^{-1}
v_{dAx}	Maximum rate of Axin2 protein degradation	A19	0.6 nM/min
K_{dAx}	Michaelis constant for Axin2 protein degradation	A19	0.63 nM
θ	Scaling factor for Wnt loop	A14-19	1.12

Fgf loop

Parameter	Description	Equations	Value
Ras_t	Total concentration of Ras protein	A30	2.0 nM
V_{MaRas}	Maximum rate of Ras activation	A25	4.968 nM/min
K_{aFgf}	FGF8 threshold constant for activation of Ras	A25	0.5 nM
K_{aRas}	Inactive Ras threshold constant for activation of Ras	A25	0.103 nM
V_{MdRas}	Maximum rate of Ras activation	A25	0.41 nM/min
K_{dRas}	Michaelis constant for Ras inactivation	A25	0.1 nM

ERK_t	Total concentration of ERK protein kinase	A31	2.0 nM
V_{MaErk}	Maximum rate of Ras-mediated ERK activation	A26	3.3 nM/min
K_{aErk}	Inactive ERK threshold constant for ERK activation	A26	0.05 nM
k_{cDusp}	Rate constant for inactivation of ERK	A26	1.35 min^{-1}
K_{dErk}	Michaelis constant for inactivation of ERK	A26	0.05 nM
X_t	Total concentration of factor X	A32	2.0 nM
V_{MaX}	Maximum rate of ERK-mediated X activation	A27	1.6 nM/min
K_{aX}	Threshold constant for ERK-mediated X activation	A27	0.05 nM
V_{MdX}	Maximum rate of X inactivation	A27	0.5 nM/min
K_{dX}	Michaelis constant for X inactivation	A27	0.05 nM
$V_{MsMDusp}$	Maximum rate of Dusp6 transcription	A34	0.9 nM/min
K_{aMDusp}	Threshold constant for X-activated Dusp6 transcription	A34	0.5 nM
$V_{MdMDusp}$	Maximum rate of Dusp6 mRNA degradation	A33	0.5 nM/min
K_{dMDusp}	Michaelis constant for Dusp6 mRNA degradation	A33	0.5 nM
k_{sDusp}	Rate constant for Dusp6 translation	A29	0.5 min^{-1}
V_{dDusp}	Maximum rate of Dusp6 protein degradation	A29	2.0 nM/min
K_{dDusp}	Michaelis constant for Dusp6 protein degradation	A29	0.5 nM
$\boldsymbol{v_{DuspX}}$	**Fraction of Dusp6 transcription solely regulated by X**	**A34**	**0.2**
$\boldsymbol{v_{DuspDMF}}$	**Fraction of Dusp6 transcription under regulation by DMF**	**A34**	**0.8**
$\boldsymbol{K_{IMDusp}}$	**Threshold constant for DMF-mediated inhibition of Dusp6**	**A34**	**0.3 nM**
η	Scaling factor for FGF8 loop	A25-29	0.328

Initial variable values

Notch loop

Variable	**Description**	**Equations**	**Value**
N	Notch in the membrane	A1, A10	0.0552 nM
N_a	Activated Notch	A2	0.8095 nM

N_{an}	Activated Notch in the nucleus	A3-4, A6, A8, A12-13	0.0039 nM
$\boldsymbol{N_{\mathrm{ap}}}$	**Phosphorylated Notch**	**A4, A6, A8, A13**	**0.0013 nM**
MF	Lunatic Fringe mRNA	A4-5	0.003 nM
F	Lunatic Fringe protein	A5, A10	0.0006 nM
$\boldsymbol{MDMF}$	**Dusp Modification Factor mRNA**	**A6-7**	**0.0011 nM**
$\boldsymbol{DMF}$	**Dusp Modification Factor protein**	**A7, A34**	**0.0003 nM**
$\boldsymbol{DL_{\mathrm{c}}}$	**Delta protein in the cytoplasm**	**A8-9**	**0.0138 nM**
$\boldsymbol{DL_{\mathrm{m}}}$	**Delta protein on the membrane**	**A9**	**0.0058 nM**

Wnt loop

Variable	Description	Equations	Value
K	Free Gsk3 protein	A13-14, A20-21	1.2252 nM
B	β-catenin	A15, A22, A24	0.9118 nM
B_{p}	Phosphorylated β-catenin	A16, A23	0.0171 nM
B_{N}	Nuclear β-catenin	A17-18, A24	0.4324 nM
MAx	Axin2 mRNA	A18-19	7.7560 nM
A	Axin2 protein	A19-20	0.1076 nM
AK	Axin2-Gsk3 complex	A15-16, A20-21	1.7747 nM

Fgf loop

Variable	Description	Equations	Value
Ras_{a}	Activated Ras protein	A25-26, A30	1.9912 nM
ERK_{a}	Activated ERK protein	A26-27, A31	1.9780 nM
X_{a}	Activated transcription factor X	A27, A32, A34	1.9777 nM
$MDusp$	Dusp mRNA	A28-29, A33	2.9054 nM
$Dusp$	Dusp protein	A26, A29	0.7692 nM

Appendix C

SOURCE CODE

Instructions on how to dowload and run the source code, and how to alter key simulation parameters are given in **Appendix D: RUNNING THE SIMULATIONS.**

Segmenation-clock network

The files SegClock.h and SegClock.cpp contain the C++ class **Oscillator**, described in Chapter 3, METHODS, in Section 3.2.4, Segmentation-clock network. The **Oscillator** class stores the current segmentation-clock chemical concentrations as variables that are updated when the segmentation-clock equations are iterated. The **Iterate** function uses the current variable values, the values of the local FGF8, Wnt3a and Delta signaling environment (taken as inputs to the function), and the segmentation-clock equations to update the values of the variables with a fourth-order Runge-Kutta method. The "GET" functions (*e.g.*, `getLfng()`) return values of the variables for use within the CC3D simulation Python script. The segmentation-clock equations implemented in the **Oscillator** class are listed in Appendix A, SEGMENTATION CLOCK NETWORK SUBMODEL EQUATIONS; parameters and initial conditions are listed in Appendix B, SEGMENTATION CLOCK NETWORK SUBMODEL PARAMETERS AND INITIAL CONDITIONS.

For implementation in CC3D, SegClock.h and SegClock.cpp are wrapped in Python. We performed this using *Simplified Wrapper and Interface Generator*, or *SWIG* (http://www.swig.org). The instructions for running the simulations (**Appendix D: RUNNING THE SIMULATIONS**) include steps to Python-wrap the segmentation-clock files.

File 1 (header): SegClock.h

```
/*****      SegClock.h        *************
*                                         *
*     sdh BIOC @ IU 03 Nov 2009           *
*                                         *
*******************************************/

class Oscillator
{
public:
      Oscillator(double FGF, double WNT, double KADL, int COUPLING,
double DT);
      ~Oscillator();
//Copy constructor
      Oscillator(const Oscillator * _osc);
//Public functions
      void Iterate(double FGF, double WNT, double KADL, double DLMSIG);
      void PhaseShift(double FGF, double WNT, double KADL,
      double ShiftTime);
//Public GET functions
      double getLfng();
      double getAxin();
      double getDusp();
      double getDLm();
      double getBeta();
      double getN();
      double getNa();
      double getNan();
      double getNap();
      double getMF();
      double getMDMF();
      double getDMF();
      double getDLc();
      double getK();
      double getBp();
      double getBN();
      double getMAx();
      double getAK();
      double getRasa();
      double getERKa();
      double getXa();
      double getMDusp();

private:
//Parameters
      double Fgf; double D; double Kadl; int coupling; double dt;
//Notch
      double vsN; double vdN; double KdN; double kc; double VdNa;
      double KdNa; double VdNan; double KdNan; double KIF; double kt1;
      double kt2; double vNap; double KpN; double vsF; double KA;
      double vmF; double KdmF; double ksF; double vdF; double KdF;
      double MDL; double ksDL; double ktDL; double vdDL; double KdDL;
      double KNan; double eps; double KDL; double phi; double vs0MDMF;
      double KaNDMF; double vdmDMF; double KdmDMF; double ksDMF;
      double vdDMF; double KdDMF; double DLmSig;
//Wnt
      double a1; double d1; double vsB; double kt3; double kt4;
      double VMK; double Kt; double KID; double K1; double VMP;
      double K2; double kd1; double kd2; double v0; double vMB;
      double KaB; double vmd; double Kmd; double vMXa; double KaXa;
      double ksAx; double vdAx; double KdAx; double theta;
//Fgf
      double Rast; double VMaRas; double KaFgf; double KaRas;
      double VMdRas; double KdRas; double ERKt; double VMaErk;
      double KaErk; double kcDusp; double KdErk; double Xt;
      double VMaX; double KaX; double VMdX; double KdX;
      double VMsMDusp; double KaMDusp; double VMdMDusp; double KdMDusp;
      double ksDusp; double VdDusp; double KdDusp; double vDuspX;
      double vDuspDMF; double KIMDusp; double eta;
//Variables
//Notch
      double N; double Na; double Nan; double MF; double F;
      double MDMF; double DMF; double DLc; double DLm;
//Wnt
      double K; double B; double Bp; double BN; double MAx; double A;

//Fgf
      double Rasa; double ERKa; double Xa; double MDusp; double Dusp;
//Integration functions and parameters
      double H;
      void RungeKutta();
      void RK_k1();
      void RK_k2();
      void RK_k3();
      void RK_k4();
///Notch
      double N_; double Na_; double Nan_; double MF_; double F_;
      double MDMF_; double DMF_; double DLc_; double DLm_;
      double k1N; double k1Na; double k1Nan; double k1MF; double k1F;
      double k1MDMF; double k1DMF; double k1DLc; double k1DLm;
      double k2N; double k2Na; double k2Nan; double k2MF; double k2F;
      double k2MDMF; double k2DMF; double k2DLc; double k2DLm;
      double k3N; double k3Na; double k3Nan; double k3MF; double k3F;
      double k3MDMF; double k3DMF; double k3DLc; double k3DLm;
      double k4N; double k4Na; double k4Nan; double k4MF; double k4F;
      double k4MDMF; double k4DMF; double k4DLc; double k4DLm;
///Wnt
      double K_; double B_; double Bp_; double BN_; double MAx_;
      double A_;
      double k1K; double k1B; double k1Bp; double k1BN; double k1MAx;
      double k1A; double k2K; double k2B; double k2Bp; double k2BN;
      double k2MAx; double k2A; double k3K; double k3B; double k3Bp;
      double k3BN; double k3MAx; double k3A; double k4K; double k4B;
      double k4Bp; double k4BN; double k4MAx; double k4A;
///FGF
      double Rasa_; double ERKa_; double Xa_; double MDusp_;
      double Dusp_; double k1Rasa; double k1ERKa; double k1Xa;
      double k1MDusp; double k1Dusp; double k2Rasa; double k2ERKa;
      double k2Xa; double k2MDusp; double k2Dusp; double k3Rasa;
      double k3ERKa; double k3Xa; double k3MDusp; double k3Dusp;
      double k4Rasa; double k4ERKa; double k4Xa; double k4MDusp;
      double k4Dusp;
//ODEs
//Notch Path
      double dN_dt(); double dNa_dt(); double dNan_dt(); double Nap();
      double dMF_dt(); double dF_dt(); double dMDMF_dt();
      double dDMF_dt(); double dDLc_dt(); double dDLm_dt();
      void setDLmSig(double DLMSIG); double NSig(); double Vtr();
      double VMDMF(); double Kt_DL();
//Wnt Path
      double dK_dt(); double dB_dt(); double dBp_dt(); double dBN_dt();
      double dMAx_dt(); double dA_dt(); double AK(); double V1();
      double V2(); double VK(); double VP(); double KMAx();
//Fgf Path
      double dRasa_dt(); double dERKa_dt(); double dXa_dt();
      double dMDusp_dt(); double dDusp_dt(); double Rasi();
      double VaRas(); double VdRas(); double ERKi(); double VaERK();
      double VdERK(); double Xi(); double VaX(); double VdX();
      double VsMDusp(); double VdMDusp();
//Etc.
      int i; int ShiftSteps;
};
```

File 2: SegClock.cpp

```
/*****      SegClock.cpp      *******************
*                                               *
*     Hester, Phys Dept and BIOC @ IU           *
*     03 Nov 2009                               *
*                                               *
*************************************************/

#ifdef COMPILE_ON_MSVC
using namespace std;
#endif

#include "config.h"
#include "SegClock.h"
#include <stdio.h>
#include <math.h>
#include <cstdlib>
#include <iostream>

Oscillator::Oscillator(double FGF, double WNT, double KADL, int
COUPLING, double DT)
{
//Set Parameters
      Fgf = FGF; D = WNT; Kadl = KADL; coupling = COUPLING; dt = DT;
      H=dt/6;
//Notch parameters from Appendix B, SEGMENTATION CLOCK NETWORK SUBMODEL
//PARAMETERS AND INITIAL CONDITIONS
      vsN=0.23; vdN=2.82; KdN=1.4; kc=3.45; VdNa=0.01; KdNa=0.001;
      VdNan=0.1;
      KdNan=0.001; KIF=0.45; kt1=0.1; kt2=0.1; vNap=1.0; KpN=2.5;
      vsF=3.24;
      KA=0.05; vmF=1.92; KdmF=0.768; ksF=0.3; vdF=0.39; KdF=0.37;
            MDL=0.5; ksDL=0.75; ktDL=0.5; vdDL=0.5; KdDL=0.5;
      KNan=0.04; eps=0.43;
      KDL=0.08; phi=0.9; vs0MDMF=0.2375; KaNDMF=0.05; vdmDMF=0.15;
      KdmDMF=0.4; ksDMF=0.05; vdDMF=0.1; KdDMF=0.5;
//Wnt parameters from Appendix B, SEGMENTATION CLOCK NETWORK SUBMODEL
//PARAMETERS AND INITIAL CONDITIONS
      a1=1.8; d1=0.1; vsB=0.087; kt3=0.7; kt4=1.5; VMK=4.5; Kt=3.0;
      KID=0.5; K1=0.28; VMP=1.0; K2=0.03; kd1=0.0; kd2=7.062; v0=0.06;
      vMB=1.64; KaB=0.7; vmd=0.8; Kmd=0.48; vMXa=0.5; KaXa=0.05;
      ksAx=0.02; vdAx=0.6; KdAx=0.63; theta=1.12;
//Fgf parameters from Appendix B, SEGMENTATION CLOCK NETWORK SUBMODEL
//PARAMETERS AND INITIAL CONDITIONS
      Rast=2; VMaRas=4.968; KaFgf=0.5; KaRas=0.103; VMdRas=0.41;
      dRas=0.1; ERKt=2; VMaErk=3.3; KaErk=0.05; kcDusp=1.35;
      KdErk=0.05; Xt=2; VMaX=1.6; KaX=0.05; VMdX=0.5; KdX=0.05;
      VMsMDusp=0.9; KaMDusp=0.5; VMdMDusp=0.5; KdMDusp=0.5; ksDusp=0.5;
      VdDusp=2; KdDusp=0.5; vDuspX=0.2; vDuspDMF=0.8; KIMDusp=0.3;
      eta=0.328;

//Set Initial Values
//Notch initial conditions from Appendix B, SEGMENTATION CLOCK NETWORK
//SUBMODEL PARAMETERS AND INITIAL CONDITIONS
      N=0.1144; Na=0.7745; Nan=0.0243; MF=1.2464; F=2.3795; DMF=0.5525;
      DMF=0.2090; DLc=0.1255; DLm=0.0906;

//Wnt initial conditions from Appendix B, SEGMENTATION CLOCK NETWORK
//SUBMODEL PARAMETERS AND INITIAL CONDITIONS
      K=1.4428; B=0.4705; Bp=0.0090; BN=0.2146; MAx=0.1523; A=0.0461;
//Fgf initial conditions from Appendix B, SEGMENTATION CLOCK NETWORK
//SUBMODEL PARAMETERS AND INITIAL CONDITIONS
      Rasa=1.9888; ERKa=0.01930; Xa=0.0015; MDusp=2.1651; Dusp=8.5078;
}

Oscillator::Oscillator(const Oscillator * _osc)
//Constructor creates a new instance of the class Oscillator, using the
//parameter and current variable values from an existing Oscillator
//class; this allows model cells to divide and create daughter cells
//that inherit their oscillator states.
{
      Fgf= _osc->Fgf; D= _osc->D; Kadl= _osc->Kadl;
      coupling= _osc->coupling; dt= _osc->dt; H= _osc->H;

//Parameters
//Notch
      vsN= _osc-> vsN; vdN= _osc-> vdN; KdN= _osc-> KdN; kc= _osc-> kc;
      VdNa= _osc-> VdNa; KdNa= _osc-> KdNa; VdNan= _osc-> VdNan;
      KdNan= _osc-> KdNan; KIF= _osc-> KIF; kt1= _osc-> kt1;
      kt2= _osc-> kt2; vNap= _osc-> vNap; KpN= _osc-> KpN;
      vsF= _osc-> vsF; KA= _osc-> KA; vmF= _osc-> vmF;
      KdmF= _osc-> KdmF; ksF= _osc-> ksF; vdF= _osc-> vdF;
      KdF= _osc-> KdF;MDL= _osc-> MDL; ksDL= _osc-> ksDL;
      ktDL= _osc-> ktDL; vdDL= _osc-> vdDL; KdDL= _osc-> KdDL;
      KNan= _osc-> KNan; eps= _osc-> eps; KDL= _osc-> KDL;
      phi= _osc-> phi; vs0MDMF= _osc-> vs0MDMF; KaNDMF= _osc-> KaNDMF;
      vdmDMF= _osc-> vdmDMF; KdmDMF= _osc-> KdmDMF;
      ksDMF= _osc-> ksDMF; vdDMF= _osc-> vdDMF; KdDMF= _osc-> KdDMF;
```

```
//Wnt
      a1= _osc-> a1; d1= _osc-> d1; vsB= _osc-> vsB; kt3= _osc-> kt3;
      kt4= _osc-> kt4; VMK= _osc-> VMK; Kt= _osc-> Kt; KID= _osc-> KID;
      K1= _osc-> K1; VMP= _osc-> VMP; K2= _osc-> K2; kd1= _osc-> kd1;
      kd2= _osc-> kd2; v0= _osc-> v0; vMB= _osc-> vMB; KaB= _osc-> KaB;
      vmd= _osc-> vmd; Kmd= _osc-> Kmd; vMXa= _osc-> vMXa;
      KaXa= _osc-> KaXa; ksAx= _osc-> ksAx; vdAx= _osc-> vdAx;
      KdAx= _osc-> KdAx; theta= _osc-> theta;
//Fgf
      Rast= _osc-> Rast; VMaRas= _osc-> VMaRas; KaFgf= _osc-> KaFgf;
      KaRas= _osc-> KaRas; VMdRas= _osc-> VMdRas; KdRas= _osc-> KdRas;
      ERKt= _osc-> ERKt; VMaErk= _osc-> VMaErk; KaErk= _osc-> KaErk;
      kcDusp= _osc-> kcDusp; KdErk= _osc-> KdErk; Xt= _osc-> Xt;
      VMaX= _osc-> VMaX; KaX= _osc-> KaX; VMdX= _osc-> VMdX;
      KdX= _osc-> KdX; VMsMDusp= _osc-> VMsMDusp;
      KaMDusp= _osc-> KaMDusp; VMdMDusp= _osc-> VMdMDusp;
      KdMDusp= _osc-> KdMDusp; ksDusp= _osc-> ksDusp;
      VdDusp= _osc-> VdDusp; KdDusp= _osc-> KdDusp;
      vDuspX= _osc-> vDuspX; vDuspDMF= _osc-> vDuspDMF;
      KIMDusp= _osc-> KIMDusp; eta= _osc-> eta;

//Initial Values
//Notch
      N= _osc-> N; Na= _osc-> Na; Nan= _osc-> Nan; MF= _osc-> MF;
      F= _osc-> F; MDMF= _osc-> MDMF; DMF= _osc-> DMF; DLc= _osc-> DLc;
      DLm= _osc-> DLm;
//Wnt
      K= _osc-> K; B= _osc-> B; Bp= _osc-> Bp; BN= _osc-> BN;
      MAx= _osc-> MAx; A= _osc-> A;
//Fgf
      Rasa= _osc-> Rasa; ERKa= _osc-> ERKa; Xa= _osc-> Xa;
      MDusp= _osc-> MDusp; Dusp= _osc-> Dusp;
}

Oscillator::~Oscillator()
{}

//GET functions
//GET functions allow access to the current chemical values of
//segmentation-clock components from within the CC3D simulation Python
//script.
double Oscillator::getLfng()
{     return F;   }
double Oscillator::getAxin()
{     return A;   }
double Oscillator::getDusp()
{     return Dusp;      }
double Oscillator::getDLm()
{     return DLm; }
double Oscillator::getBeta()
{     return B;   }
double Oscillator::getN()
{     return N;   }
double Oscillator::getNa()
{     return Na;  }
double Oscillator::getNan()
{     return Nan; }
```

```
double Oscillator::getNap()
{      return vNap*Nan*K/(KpN+K);     }
double Oscillator::getMF()
{      return MF;   }
double Oscillator::getMDMF()
{      return MDMF;       }
double Oscillator::getDMF()
{      return DMF;  }
double Oscillator::getDLc()
{      return DLc;  }
double Oscillator::getK()
{      return K;    }
double Oscillator::getBp()
{      return Bp;   }
double Oscillator::getBN()
{      return BN;   }
double Oscillator::getMAx()
{      return MAx;  }
double Oscillator::getAK()
{      return Kt - K;     }
double Oscillator::getRasa()
{      return Rasa;       }
double Oscillator::getERKa()
{      return ERKa;       }
double Oscillator::getXa()
{      return Xa;   }
double Oscillator::getMDusp()
{      return MDusp;      }

void Oscillator::Iterate(double FGF, double WNT, double KADL, double
DLMSIG)
/*The Iterate function takes the current values of the variables and
the inputs (the local FGF8, Wnt3a and Delta signals) and updates the
values of the variables for a time step using a fourth-order Runge-
Kutta integration solver.*/
{
      Fgf=FGF; D=WNT; Kadl=KADL; setDLmSig(DLMSIG);
      RungeKutta();
}

void Oscillator::PhaseShift(double FGF, double WNT, double KADL, double
ShiftTime)
//The PhaseShift function advances the segmenation clock by a given
//time shift.
{
      Fgf=FGF; D=WNT; Kadl=KADL; ShiftSteps=int(ShiftTime/dt);
      for (i=0; i<ShiftSteps; i++)
      {      setDLmSig(DLm);
             RungeKutta();      }
}

//Runge-Kutta Integrator
void Oscillator::RungeKutta()
{      RK_k1();
       RK_k2();
       RK_k3();
       RK_k4();
///Notch
      N+=H*(k1N+2*(k2N+k3N)+k4N);
      Na+=H*(k1Na+2*(k2Na+k3Na)+k4Na);
      Nan+=H*(k1Nan+2*(k2Nan+k3Nan)+k4Nan);
      MF+=H*(k1MF+2*(k2MF+k3MF)+k4MF);
      F+=H*(k1F+2*(k2F+k3F)+k4F);
      MDMF+=H*(k1MDMF+2*(k2MDMF+k3MDMF)+k4MDMF);
      DMF+=H*(k1DMF+2*(k2DMF+k3DMF)+k4DMF);
      DLc+=H*(k1DLc+2*(k2DLc+k3DLc)+k4DLc);
      DLm+=H*(k1DLm+2*(k2DLm+k3DLm)+k4DLm);

///Wnt
      K+=H*(k1K+2*(k2K+k3K)+k4K);
      B+=H*(k1B+2*(k2B+k3B)+k4B);
      Bp+=H*(k1Bp+2*(k2Bp+k3Bp)+k4Bp);
      BN+=H*(k1BN+2*(k2BN+k3BN)+k4BN);
      MAx+=H*(k1MAx+2*(k2MAx+k3MAx)+k4MAx);
      A+=H*(k1A+2*(k2A+k3A)+k4A);

///FGF
      Rasa+=H*(k1Rasa+2*(k2Rasa+k3Rasa)+k4Rasa);
      ERKa+=H*(k1ERKa+2*(k2ERKa+k3ERKa)+k4ERKa);
      Xa+=H*(k1Xa+2*(k2Xa+k3Xa)+k4Xa);
      MDusp+=H*(k1MDusp+2*(k2MDusp+k3MDusp)+k4MDusp);
      Dusp+=H*(k1Dusp+2*(k2Dusp+k3Dusp)+k4Dusp);
}
// Calculate k1 for all DEQs
void Oscillator::RK_k1()
{     N_=N; Na_=Na; Nan_=Nan; MF_=MF; F_=F; MDMF_=MDMF; DMF_=DMF;
      DLc_=DLc; DLm_=DLm;
      K_=K; B_=B; Bp_=Bp; BN_=BN; MAx_=MAx; A_=A;
      Rasa_=Rasa; ERKa_=ERKa; Xa_=Xa; MDusp_=MDusp; Dusp_=Dusp;
///Notch
      k1N=dN_dt();
      k1Na=dNa_dt();
      k1Nan=dNan_dt();
      k1MF=dMF_dt();
      k1F=dF_dt();
      k1MDMF=dMDMF_dt();
      k1DMF=dDMF_dt();
      k1DLc=dDLc_dt();
      k1DLm=dDLm_dt();
///Wnt
      k1K=dK_dt();
      k1B=dB_dt();
      k1Bp=dBp_dt();
      k1BN=dBN_dt();
      k1MAx=dMAx_dt();
      k1A=dA_dt();
///FGF
      k1Rasa=dRasa_dt();
      k1ERKa=dERKa_dt();
      k1Xa=dXa_dt();
      k1MDusp=dMDusp_dt();
      k1Dusp=dDusp_dt();                  }

// Calculate k2 for all DEQs
```

```
void Oscillator::RK_k2()
{      N_=N+0.5*dt*k1N; Na_=Na+0.5*dt*k1Na; Nan_=Nan+0.5*dt*k1Nan;
       MF_=MF+0.5*dt*k1MF; F_=F+0.5*dt*k1F; MDMF_=MDMF+0.5*dt*k1MDMF;
       DMF_=DMF+0.5*dt*k1DMF; DLc_=DLc+0.5*dt*k1DLc;
       DLm_=DLm+0.5*dt*k1DLm; k2N=dN_dt(); K_=K+0.5*dt*k1K;
       B_=B+0.5*dt*k1B; Bp_=Bp+0.5*dt*k1Bp; BN_=BN+0.5*dt*k1BN;
       MAx_=MAx+0.5*dt*k1MAx; A_=A+0.5*dt*k1; Rasa_=Rasa+0.5*dt*k1Rasa;
       ERKa_=ERKa+0.5*dt*k1ERKa; Xa_=Xa+0.5*dt*k1Xa;
       MDusp_=MDusp+0.5*dt*k1MDusp; Dusp_=Dusp+0.5*dt*k1Dusp;
///Notch
       k2Na=dNa_dt();
       k2Nan=dNan_dt();
       k2MF=dMF_dt();
       k2F=dF_dt();
       k2MDMF=dMDMF_dt();
       k2DMF=dDMF_dt();
       k2DLc=dDLc_dt();
       k2DLm=dDLm_dt();
///Wnt
       k2K=dK_dt();
       k2B=dB_dt();
       k2Bp=dBp_dt();
       k2BN=dBN_dt();
       k2MAx=dMAx_dt();
       k2A=dA_dt();

///FGF
       k2Rasa=dRasa_dt();
       k2ERKa=dERKa_dt();
       k2Xa=dXa_dt();
       k2MDusp=dMDusp_dt();
       k2Dusp=dDusp_dt();                  }

//Calculate k3 for all DEQs
void Oscillator::RK_k3()
{      N_=N+0.5*dt*k2N; Na_=Na+0.5*dt*k2Na; Nan_=Nan+0.5*dt*k2Nan;
       MF_=MF+0.5*dt*k2MF; F_=F+0.5*dt*k2F; MDMF_=MDMF+0.5*dt*k2MDMF;
       DMF_=DMF+0.5*dt*k2DMF; DLc_=DLc+0.5*dt*k2DLc;
       DLm_=DLm+0.5*dt*k2DLm; K_=K+0.5*dt*k2K; B_=B+0.5*dt*k2B;
       Bp_=Bp+0.5*dt*k2Bp; BN_=BN+0.5*dt*k2BN; MAx_=MAx+0.5*dt*k2MAx;
       A_=A+0.5*dt*k2A; Rasa_=Rasa+0.5*dt*k2Rasa;
       ERKa_=ERKa+0.5*dt*k2ERKa; Xa_=Xa+0.5*dt*k2Xa;
       MDusp_=MDusp+0.5*dt*k2MDusp; Dusp_=Dusp+0.5*dt*k2Dusp;
///Notch
       k3N=dN_dt();
       k3Na=dNa_dt();
       k3Nan=dNan_dt();
       k3MF=dMF_dt();
       k3F=dF_dt();
       k3MDMF=dMDMF_dt();
       k3DMF=dDMF_dt();
       k3DLc=dDLc_dt();
       k3DLm=dDLm_dt();
///Wnt
       k3K=dK_dt();
       k3B=dB_dt();
       k3Bp=dBp_dt();
```

```
        k3BN=dBN_dt();
        k3MAx=dMAx_dt();
        k3A=dA_dt();
///FGF
        k3Rasa=dRasa_dt();
        k3ERKa=dERKa_dt();
        k3Xa=dXa_dt();
        k3MDusp=dMDusp_dt();
        k3Dusp=dDusp_dt();                  }

//Calculate k4 for all DEQs
void Oscillator::RK_k4()
{       N_=N+dt*k3N; Na_=Na+dt*k3Na; Nan_=Nan+dt*k3Nan; MF_=MF+dt*k3MF;
        F_=F+dt*k3F; MDMF_=MDMF+dt*k3MDMF; DMF_=DMF+dt*k3DMF;
        DLc_=DLc+dt*k3DLc; DLm_=DLm+dt*k3DLm;
        K_=K+dt*k3K; B_=B+dt*k3B; Bp_=Bp+dt*k3Bp; BN_=BN+dt*k3BN;
        MAx_=MAx+dt*k3MAx; A_=A+dt*k3A; Rasa_=Rasa+dt*k3Rasa;
        ERKa_=ERKa+dt*k3ERKa; Xa_=Xa+dt*k3Xa;
        MDusp_=MDusp+dt*k3MDusp; Dusp_=Dusp+dt*k3Dusp;
///Notch
        k4N=dN_dt();
        k4Na=dNa_dt();
        k4Nan=dNan_dt();
        k4MF=dMF_dt();
        k4F=dF_dt();
        k4MDMF=dMDMF_dt();
        k4DMF=dDMF_dt();
        k4DLc=dDLc_dt();
        k4DLm=dDLm_dt();
///Wnt
        k4K=dK_dt();
        k4B=dB_dt();
        k4Bp=dBp_dt();
        k4BN=dBN_dt();
        k4MAx=dMAx_dt();
        k4A=dA_dt();
///FGF
        k4Rasa=dRasa_dt();
        k4ERKa=dERKa_dt();
        k4Xa=dXa_dt();
        k4MDusp=dMDusp_dt();
        k4Dusp=dDusp_dt();
                              }


//Delta/Notch ODEs
/*The segmenation-clock equations; a listing of the equations can be
found in Appendix A, SEGMENTATION CLOCK NETWORK SUBMODEL EQUATIONS.
Each function returns the value of d_/dt for a chemical component of
the segmentation clock.*/
double Oscillator::dN_dt()
{       return eps*(vsN-vdN*N_/(KdN+N_)-NSig());   }
double Oscillator::dNa_dt()
{       return eps*(NSig()-VdNa*Na_/(KdNa+Na_)-Vtr());   }
double Oscillator::dNan_dt()
{       return eps*(Vtr()-VdNan*Nan_/(KdNan+Nan_));      }
double Oscillator::Nap()
```

```
{	return vNap*Nan_*K_/(KpN+K_); }
double Oscillator::dMF_dt()
{	return eps*(vsF*pow((Nan_-Nap()),2)/
	(pow(KA,2)+pow((Nan_-Nap()),2))-vmF*MF_/(KdmF+MF_));	}
double Oscillator::dF_dt()
{	return eps*(ksF*MF_-vdF*F_/(KdF+F_));	}
double Oscillator::dMDMF_dt()
{	return phi*(VMDMF()-vdmDMF*(MDMF_/(KdmDMF+MDMF_)));	}
double Oscillator::dDMF_dt()
{	return phi*(ksDMF*MDMF_-vdDMF*DMF_/(KdDMF+DMF_));	}
double Oscillator::dDLc_dt()
{	return eps*(ksDL*MDL*(Nan_-Nap())/
	(KNan+(Nan_-Nap()))-Kt_DL()-vdDL*DLc_/(KdDL+DLc_));	}
double Oscillator::dDLm_dt()
{	return eps*(Kt_DL()-vdDL*DLm_/(KdDL+DLm_));	}
void Oscillator::setDLmSig(double DLMSIG)
{	if (coupling==2) DLmSig=DLMSIG/(KDL+DLm);
	else DLmSig=DLm/(KDL+DLm);	}
double Oscillator::NSig()
{	return
	kc*N_*pow(KIF,2)/(pow(KIF,2)+pow(F_,2))*DLmSig/(Kadl+DLmSig);
		}
double Oscillator::Vtr()
{	return kt1*Na_-kt2*Nan_;	}


double Oscillator::VMDMF()
{	return vs0MDMF*(pow((Nan_-Nap()),2)/
	(pow(KaNDMF,2)+pow((Nan_-Nap()),2)));	}
double Oscillator::Kt_DL()
{	return ktDL*DLc_; }
//Wnt ODEs
double Oscillator::dK_dt()
{	return theta*V1();	}
double Oscillator::dB_dt()
{	return theta*(vsB - VK()*AK()/Kt + VP() + V2() - kd1*B_);	}
double Oscillator::dBp_dt()
{	return theta*(VK()*AK()/Kt - VP() - kd2*Bp_);	}
double Oscillator::dBN_dt()
{	return (-1)*theta*V2(); }
double Oscillator::dMAx_dt()
{	return theta*(KMAx() - vmd*MAx_/(Kmd+MAx_));	}
double Oscillator::dA_dt()
{	return theta*(ksAx*MAx_ - vdAx*A_/(KdAx+A_) + V1());	}
double Oscillator::AK()
{	return Kt - K_;	}
double Oscillator::V1()
{	return d1*AK() - a1*A_*K_;	}
double Oscillator::V2()
{	return kt4*BN_ - kt3*B_;	}
double Oscillator::VK()
{	return VMK*KID/(KID+D)*B_/(K1+B_);	}
double Oscillator::VP()
{	return VMP*Bp_/(K2+Bp_);	}
double Oscillator::KMAx()
{	return v0 + vMB*pow(BN_,2)/(pow(KaB,2)+pow(BN_,2)) +
	vMXa*pow(Xa_,2)/(pow(KaXa,2)+pow(Xa_,2)); }
```

```
//Fgf ODEs
double Oscillator::dRasa_dt()
{      return eta*(VaRas()-VdRas()); }
double Oscillator::dERKa_dt()
{      return eta*(VaERK()-VdERK()); }
double Oscillator::dXa_dt()
{      return eta*(VaX()-VdX());      }
double Oscillator::dMDusp_dt()
{      return eta*(VsMDusp()-VdMDusp());    }
double Oscillator::dDusp_dt()
{      return eta*(ksDusp*MDusp_-VdDusp*Dusp_/(KdDusp+Dusp_));      }
double Oscillator::Rasi()
{      return Rast-Rasa_;       }
double Oscillator::VaRas()
{      return
       VMaRas*pow(Fgf,2)/(pow(KaFgf,2)+pow(Fgf,2))*(Rasi()/(KaRas+Rasi()
       ));    }
double Oscillator::VdRas()
{      return VMdRas*Rasa_/(KdRas+Rasa_);   }
double Oscillator::ERKi()
{      return ERKt-ERKa_;       }
double Oscillator::VaERK()
{      return VMaErk*(Rasa_/Rast)*(ERKi()/(KaErk+ERKi()));    }
double Oscillator::VdERK()
{      return kcDusp*Dusp_*ERKa_/(KdErk+ERKa_);   }

double Oscillator::Xi()
{      return Xt-Xa_;     }
double Oscillator::VaX()
{      return VMaX*(ERKa_/ERKt)*(Xi()/(KaX+Xi()));      }
double Oscillator::VdX()
{      return VMdX*Xa_/(KdX+Xa_);     }
double Oscillator::VsMDusp()
{      return
       VMsMDusp*pow(Xa_,2)/(pow(KaMDusp,2)+pow(Xa_,2))*(vDuspDMF*KIMDusp
       /(KIMDusp+DMF_)+vDuspX);       }
double Oscillator::VdMDusp()
{      return VMdMDusp*MDusp_/(KdMDusp+MDusp_);   }
```

CompuCell3D Python simulation files

The files Somites.py and Somites_Step.py are the CompuCell3D (CC3D) simulation files. Somites_Step.py contains the CC3D steppables for the simulation; steppables contain simulation modules that perform a task or set of related tasks within the simulation. In Somites.py, we set the values of many of the simulation variables, define the **cell** types, specify the **cell**-**cell** adhesion matrix (see Table 3.3 in Chapter 3, METHODS, Section 3.2.1, Model GGH cell types), register the simulation Steppables from Somites_Step.py with CC3D, set the initial configuration of the **cells** (see **Figure 3.6** in Chapter 3, METHODS, Section 3.2.7, Initial conditions) and initialize the simulation. For more information on running CC3D simulations, see (Swat, Hester et al. 2009).

Main file: Somites.py

```
################### Somites.py #######################
#                                                    #
#     Belmonte, Hester, Phys Dept and BIOC @ IU      #
#     2011                                           #
#                                                    #
######################################################

import sys,time
from os import environ
from os import getcwd
import string
sys.path.append(environ["PYTHON_MODULE_PATH"])
sys.path.append(environ["SWIG_LIB_INSTALL_DIR"])

global cd; global Lz; global Lx; global margin; global T
global LamV; global LamS; global tV; global tS
global FgfWnt_ext; global fgfOn; global wntOn; global WntMax
global WntMin; global initT; global CellDif; global DiffTh
global WallOff; global Walls; global TailBudGrow; global Grow
global ImagF; global ImagTC; global Delay; global winDiff
global OscTime; global FgfDiff; global MediumDiff
global MediumWallDiff; global D_fgf; global k_fgf; global s_fgf;
global mFgfDecay; global k_mfgf; global mfgf0; global nOrder; global w
global nLayers; global fgfFreq; global p7; global p7_r

#PARAMETERS:
cd=7            #typical cell diameter (PSM)
w=7             #wall diameter
Lz=100*cd       #vertical height  --  #120*cd
Lx=10*cd        #horizontal width
```

```
margin=35       #margin
nLayers=3*cd    #numer of initial psm layers
T=120           #Temperature
nOrder=4        #Distance of interaction (1; 1.43; 1.75; 2; 2.3; 2.86)
#
#OTHER PARAMETERS (flags):
CellDif=1         #Cell Differentiation from Clock & Wavefront
DiffTh=13.9       #Cell Determination FGF Threshold
DiffTh2=0#0.35    #Cell Differentiation FGF Threshold (0 -> ticker delay
                   method)
Delay=24000       #delay between pre-differentiation and segmentation
Walls=1           #(1=walls On; 0=walls off)
WallOff=1         #Internal wall disapearance
TailBudGrow=1     #TailBud grow routine
Grow=0.120        #Tailbud cell's grow rate
ImagF=200         #Frequency of the images (mcs)
#
#VOLUME/SURFACE PARAMETERS:
LamV=15.          #Lambda Volume
LamS=15.          #Lambda Surface
tV=cd*cd          #standard target Volume
tS=28             #standard target Surface
#
#GRADIENT/DIFFUSION PARAMETERS:
fgfFreq=8                  #Frequency of diffusion solver calls
FgfDiff=1                  #FgfDiffusion (0=Static; 1=Diffusion On)
MediumWallDiff=1           #Diffusion through Med & walls (0=Off; 1=On)
D_fgf=0.00441*fgfFreq      #diff constant
k_fgf=0.00300*fgfFreq      #decay constant
s_fgf=0.02750*fgfFreq      #secretion constant
mFgfDecay=2                #mRNA type of decay (1=Linear, 2=Exponential)
k_mfgf=0.000075*fgfFreq    #mRNA decay (Lin=0.00038 / Exp=0.00015)
fgf0=45                    #FGF8 max level
mfgf0=5                    #mRNA max level
#wnt gradient
WntMax=1.60                #Maximum Wnt value
WntMin=0.00                #Minimum Wnt value
#
#OSCILLATOR PARAMETERS:
stopDetClocks=1         #turns off oscillators in determined cell types
oscFreq=1               #Frequency of oscillator calls (per MCS)
Coupling=2              #0=> off; 1=>self; 2=>on
initShift=0             #Initial time shift (in MCS)
KADL=0.035              #Value of KADL parameter in the clock (0.035)
tStep=0.015             #size of the integration step
tPerMCS=tStep*oscFreq   #time per MCS
#
#PERTURBATION SIMULATIONS:
pt=20000                #Time of perturbation
p6=0                    #Random Phase - Initial Condition
p6_r=0                  #Random Phase - Initial Condition | amount of
                         randomness
p6_in=0                 #Random Phase - Initial Condition | inheritance
                         (0%: same phase, 100%: uncorrelated)
p7=0                    #Flag: Self-coupled (no cell-cell Delta/Notch)
p7_r=0                  #If p7 is ON, PSM cells added with a pseudo-
                         random clock advance between 0 and p7_r minutes
```

```
detWait=0*Delay/4      #Postpone assignment of determination types
                        after assigning cell types
def configureSimulation(sim):
   import CompuCellSetup
   from XMLUtils import ElementCC3D
   cc3d=ElementCC3D("CompuCell3D")
   potts=cc3d.ElementCC3D("Potts")
   potts.ElementCC3D("Dimensions",{"x":Lx+2*margin,"y":1,"z":Lz})
   potts.ElementCC3D("Steps",{},100000)
   potts.ElementCC3D("Anneal",{},1)
   potts.ElementCC3D("Temperature",{},int(T))
   potts.ElementCC3D("NeighborOrder",{},2)

   cellType=cc3d.ElementCC3D("Plugin",{"Name":"CellType"})
   cellType.ElementCC3D("CellType", {"TypeName":"Medium","TypeId":"0"})
   cellType.ElementCC3D("CellType", {"TypeName":"Psm","TypeId":"1"})
   if (TailBudGrow==1):
      cellType.ElementCC3D("CellType", {"TypeName":"S","TypeId":"2"})
   elif (TailBudGrow==2):
      cellType.ElementCC3D\
("CellType", {"TypeName":"S","TypeId":"2","Freeze":True})
   cellType.ElementCC3D\
("CellType", {"TypeName":"p_Eph","TypeId":"3"})
   cellType.ElementCC3D\
("CellType", {"TypeName":"p_ephrin","TypeId":"4"})
   cellType.ElementCC3D\
("CellType", {"TypeName":"p_In","TypeId":"5"})
   cellType.ElementCC3D\
("CellType", {"TypeName":"Eph","TypeId":"6"})
   cellType.ElementCC3D\
("CellType", {"TypeName":"ephrin","TypeId":"7"})
   cellType.ElementCC3D\
("CellType", {"TypeName":"In","TypeId":"8"})
   cellType.ElementCC3D\
("CellType", {"TypeName":"Wall" ,"TypeId":"9","Freeze":True})
   cellType.ElementCC3D\
("CellType", {"TypeName":"Wall-Off","TypeId":"10"})

   contact=cc3d.ElementCC3D("Plugin",{"Name":"Contact"})
   #MEDIUM
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "Medium"},0)
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "Psm"},15)
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "S"},15)
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "p_Eph"},15)
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "p_ephrin"},15)
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "p_In"},15)
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "Eph"},5)
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "ephrin"},5)
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "In"},15)
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "Wall"},0)
   contact.ElementCC3D\
("Energy", {"Type1":"Medium", "Type2": "Wall-Off"},100)
   #PSM
   contact.ElementCC3D("Energy", {"Type1":"Psm", "Type2": "Psm"},-20)
   contact.ElementCC3D("Energy", {"Type1":"Psm", "Type2": "S"},-20)
   contact.ElementCC3D("Energy", {"Type1":"Psm", "Type2": "p_Eph"},-20)
   contact.ElementCC3D\
("Energy", {"Type1":"Psm", "Type2": "p_ephrin"},-20)
   contact.ElementCC3D("Energy", {"Type1":"Psm", "Type2": "p_In"},-20)
   contact.ElementCC3D("Energy", {"Type1":"Psm", "Type2": "Eph"},-20)
   contact.ElementCC3D\
("Energy", {"Type1":"Psm", "Type2": "ephrin"},-20)
   contact.ElementCC3D("Energy", {"Type1":"Psm", "Type2": "In"},-20)
   contact.ElementCC3D("Energy", {"Type1":"Psm", "Type2": "Wall"},30)
   contact.ElementCC3D\
("Energy", {"Type1":"Psm", "Type2": "Wall-Off"},100)
   #S
   contact.ElementCC3D("Energy", {"Type1":"S", "Type2": "S"},-20)
   contact.ElementCC3D("Energy", {"Type1":"S", "Type2": "p_Eph"},-20)
   contact.ElementCC3D\
("Energy", {"Type1":"S", "Type2": "p_ephrin"},-20)
   contact.ElementCC3D("Energy", {"Type1":"S", "Type2": "Eph"},-20)
   contact.ElementCC3D("Energy", {"Type1":"S", "Type2": "ephrin"},-20)
   contact.ElementCC3D("Energy", {"Type1":"S", "Type2": "In"},-20)
   contact.ElementCC3D("Energy", {"Type1":"S", "Type2": "Wall"},30)
   contact.ElementCC3D\
("Energy", {"Type1":"S", "Type2": "Wall-Off"},100)
   #p_EPH
   contact.ElementCC3D\
("Energy", {"Type1":"p_Eph", "Type2": "p_Eph"},-25) #strong=25 weak=22
   contact.ElementCC3D("Energy", {"Type1":"p_Eph", "Type2": "Eph"},-25)
#strong=25 weak=22
   contact.ElementCC3D\
("Energy", {"Type1":"p_Eph", "Type2": "p_ephrin"},-20)
   contact.ElementCC3D\
("Energy", {"Type1":"p_Eph", "Type2": "p_In"},-20)
   contact.ElementCC3D\
("Energy", {"Type1":"p_Eph", "Type2": "ephrin"},-20)
   contact.ElementCC3D("Energy", {"Type1":"p_Eph", "Type2": "In"},-20)
   contact.ElementCC3D("Energy", {"Type1":"p_Eph", "Type2": "Wall"},30)
   contact.ElementCC3D\
("Energy", {"Type1":"p_Eph", "Type2": "Wall-Off"},100)
   #p_EPHRIN
   contact.ElementCC3D\
("Energy", {"Type1":"p_ephrin", "Type2": "p_ephrin"},-25)
#strong=25 weak=22
   contact.ElementCC3D\
("Energy", {"Type1":"p_ephrin", "Type2": "ephrin"},-25)
#strong=25 weak=22
   contact.ElementCC3D\
("Energy", {"Type1":"p_ephrin", "Type2": "p_In"},-20)
   contact.ElementCC3D\
("Energy", {"Type1":"p_ephrin", "Type2": "Eph"},-20)
   contact.ElementCC3D\
("Energy", {"Type1":"p_ephrin", "Type2": "In"},-20)
   contact.ElementCC3D\
("Energy", {"Type1":"p_ephrin", "Type2": "Wall"},30)
   contact.ElementCC3D("Energy", {"Type1":"p_ephrin", "Type2": "Wall-
Off"},100)
   #p_IN
   contact.ElementCC3D\
("Energy", {"Type1":"p_In", "Type2": "p_In"},-35)   #strong=35 weak=25
   contact.ElementCC3D("Energy", {"Type1":"p_In", "Type2": "Eph"},-25)
#strong=25 weak=22
   contact.ElementCC3D\
("Energy", {"Type1":"p_In", "Type2": "ephrin"},-25) #strong=25 weak=22
   contact.ElementCC3D("Energy", {"Type1":"p_In", "Type2": "In"},-35)
#strong=35 weak=25
   contact.ElementCC3D("Energy", {"Type1":"p_In", "Type2": "Wall"},30)
   contact.ElementCC3D\
("Energy", {"Type1":"p_In", "Type2": "Wall-Off"},100)
   #EPH
   contact.ElementCC3D("Energy", {"Type1":"Eph", "Type2": "Eph"},-25)
   contact.ElementCC3D("Energy", {"Type1":"Eph", "Type2": "In"},-25)
   contact.ElementCC3D("Energy", {"Type1":"Eph", "Type2": "ephrin"},80)
   contact.ElementCC3D("Energy", {"Type1":"Eph", "Type2": "Wall"},30)
   contact.ElementCC3D\
("Energy", {"Type1":"Eph", "Type2": "Wall-Off"},100)
   #EPHRIN
   contact.ElementCC3D\
("Energy", {"Type1":"ephrin", "Type2": "ephrin"},-25)
   contact.ElementCC3D("Energy", {"Type1":"ephrin", "Type2": "In"},-25)
   contact.ElementCC3D\
("Energy", {"Type1":"ephrin", "Type2": "Wall"},30)
   contact.ElementCC3D\
("Energy", {"Type1":"ephrin", "Type2": "Wall-Off"},100)
   #IN
   contact.ElementCC3D("Energy", {"Type1":"In", "Type2": "In"},-40)
   contact.ElementCC3D("Energy", {"Type1":"In", "Type2": "Wall"},30)
   contact.ElementCC3D\
("Energy", {"Type1":"In", "Type2": "Wall-Off"},100)
   #WALLS
   contact.ElementCC3D("Energy", {"Type1":"Wall", "Type2": "Wall"},0)
   contact.ElementCC3D\
("Energy", {"Type1":"Wall", "Type2": "Wall-Off"},100)
   contact.ElementCC3D\
("Energy", {"Type1":"Wall-Off", "Type2": "Wall-Off"},100)
   #-neighbor order
   contact.ElementCC3D("NeighborOrder",{},nOrder)

   ptrpd = cc3d.ElementCC3D("Plugin",{"Name":"PixelTracker"})
   vlfpd = cc3d.ElementCC3D("Plugin",{"Name":"VolumeLocalFlex"})
   slfpd = cc3d.ElementCC3D("Plugin",{"Name":"SurfaceLocalFlex"})
   ntpd = cc3d.ElementCC3D("Plugin",{"Name":"NeighborTracker"})
   comtpd = cc3d.ElementCC3D("Plugin",{"Name":"CenterOfMass"})

   uipd = cc3d.ElementCC3D("Steppable",{"Type":"UniformInitializer"})
   #CELLS INITIAL CONFIGURATION:
   if (Walls):
      #Left Wall
      region = uipd.ElementCC3D("Region") #Left Wall2
      region.ElementCC3D("BoxMin",{"x":margin-w,  "y":0,  "z":15})
      region.ElementCC3D("BoxMax",{"x":margin,  "y":1,  "z":Lz-8})
      region.ElementCC3D("Types",{}, "Wall" )
      region.ElementCC3D("Width", {}, w)
      #Right Wall
      region = uipd.ElementCC3D("Region")  #Right Wall2
      region.ElementCC3D("BoxMin",{"x":Lx+margin,  "y":0,  "z":15})
      region.ElementCC3D("BoxMax",{"x":Lx+margin+w,  "y":1,  "z":Lz-8})
      region.ElementCC3D("Types",{}, "Wall" )
      region.ElementCC3D("Width", {}, w)
   #PSM CELLS
   region = uipd.ElementCC3D("Region") #PSM CELLS
   region.ElementCC3D("BoxMin",{"x":margin,  "y":0,  "z":50})
   region.ElementCC3D\
("BoxMax",{"x":margin+Lx,  "y":1,  "z":50+nLayers})
   region.ElementCC3D("Types",{}, "Psm")
   region.ElementCC3D("Width", {}, cd)
   #SOURCE CELLS
   region = uipd.ElementCC3D("Region") #SOURCE CELLS
   region.ElementCC3D("BoxMin",{"x":margin,  "y":0,  "z":50+nLayers})
   region.ElementCC3D\
("BoxMax",{"x":margin+Lx,  "y":1,  "z":50+nLayers+cd})
   region.ElementCC3D("Types",{}, "S")
   region.ElementCC3D("Width", {}, cd)

   bwpd=cc3d.ElementCC3D("Steppable",{"Type":"BoxWatcher"})
   bwpd.ElementCC3D("XMargin",  {},  2)
   bwpd.ElementCC3D("YMargin",  {},  1)
   bwpd.ElementCC3D("ZMargin",  {},  2)

   flexDiffSolver=cc3d.ElementCC3D\
("Steppable",{"Type":"FlexibleDiffusionSolverFE","Frequency":fgfFreq})
   if (FgfDiff):
      diffusionField=flexDiffSolver.ElementCC3D("DiffusionField")
      diffusionData=diffusionField.ElementCC3D("DiffusionData")
      diffusionData.ElementCC3D("FieldName",{},"06_Fgf")  #need to have
the same name of the field
      diffusionData.ElementCC3D("DiffusionConstant",{},D_fgf)
      diffusionData.ElementCC3D("DecayConstant",{},k_fgf)
      if not MediumWallDiff:
        diffusionData.ElementCC3D("DoNotDiffuseTo",{},"Wall")
        diffusionData.ElementCC3D("DoNotDiffuseTo",{},"Wall-Off")
        diffusionData.ElementCC3D("DoNotDiffuseTo",{},"Medium")

   CompuCellSetup.setSimulationXMLDescription(cc3d)


import CompuCellSetup
sim,simthread = CompuCellSetup.getCoreSimulationObjects()
configureSimulation(sim)

import CompuCell
CompuCellSetup.initializeSimulationObjects(sim,simthread)
pyAttributeAdder,dictAdder=CompuCellSetup.attachDictionaryToCells(sim)

#Add Python steppables here
steppableRegistry=CompuCellSetup.getSteppableRegistry()
```

```
changeWatcherRegistry=CompuCellSetup.getChangeWatcherRegistry(sim)
stepperRegistry=CompuCellSetup.getStepperRegistry(sim)


from Somites_Step import InitVolSur
initVolSur=InitVolSur(_simulator=sim,_frequency=1,_LamV=LamV,_LamS=LamS
,_tV=tV,_tS=tS,_fgf0=fgf0,_DiffTh=DiffTh)
steppableRegistry.registerSteppable(initVolSur)

from Somites_Step import mFgfWntPatternWatch
mfgfWntPatternWatch=mFgfWntPatternWatch(_simulator=sim,_frequency=ImagF
*15)
steppableRegistry.registerSteppable(mfgfWntPatternWatch)

from Somites_Step import Oscillators
oscillators=Oscillators(_simulator=sim,_frequency=oscFreq,_tPerMCS=tPer
MCS,_tStep=tStep,_initShift=initShift,_Coupling=Coupling,_KADL=KADL,_Wn
tMax=WntMax,_fgf0=fgf0,_pt=pt,_p6=p6,_p7=p7,_p7_r=p7_r)
steppableRegistry.registerSteppable(oscillators)

if (CellDif):
   from Somites_Step import DoBoundary

doBoundary=DoBoundary(_simulator=sim,_frequency=1,_WallOff=WallOff,_Dif
fTh=DiffTh,_DiffTh2=DiffTh2,_Delay=Delay,_stopDetClocks=stopDetClocks,
_detWait=detWait)
steppableRegistry.registerSteppable(doBoundary)

if (TailBudGrow==1):
   from Somites_Step import TailBud

tailBud=TailBud(_simulator=sim,_frequency=1,_tV=tV,_tS=tS,_LamV=LamV,_L
amS=LamS,_Grow=Grow,_p6=p6,_p6_in=p6_in,_KADL=KADL,_WntMax=WntMax,_fgf0
=fgf0,_p7=p7,_p7_r=p7_r)
steppableRegistry.registerSteppable(tailBud)

#Create extra player fields here or add attributes
dim=sim.getPotts().getCellFieldG().getDim()
AxinField=simthread.createFloatFieldPy(dim,"01_Axin2")
LfngField=simthread.createFloatFieldPy(dim,"02_Lfng")
DuspField=simthread.createFloatFieldPy(dim,"03_Dusp")
LfngAxinField=simthread.createFloatFieldPy(dim,"04_LfngAxin")
mFgfField=simthread.createFloatFieldPy(dim,"05_mFgf")
#5-------------------------------------------06_Fgf)
FgfCellsField=simthread.createFloatFieldPy(dim,"07_FgfCells")
WntField=simthread.createFloatFieldPy(dim,"08_Wnt")

#1
from Somites_Step import ExtraFieldAxin
extraFieldAxin=ExtraFieldAxin(_simulator=sim,_frequency=ImagF,_WntMax=W
ntMax)
extraFieldAxin.setScalarField(AxinField)
steppableRegistry.registerSteppable(extraFieldAxin)

#2
from Somites_Step import ExtraFieldLfng
```

```
extraFieldLfng=ExtraFieldLfng(_simulator=sim,_frequency=ImagF,_WntMax=W
ntMax)
extraFieldLfng.setScalarField(LfngField)
steppableRegistry.registerSteppable(extraFieldLfng)

#3
from Somites_Step import ExtraFieldDusp
extraFieldDusp=ExtraFieldDusp(_simulator=sim,_frequency=ImagF)
extraFieldDusp.setScalarField(DuspField)
steppableRegistry.registerSteppable(extraFieldDusp)

#4
from Somites_Step import ExtraFieldLfngAxin
extraFieldLfngAxin=ExtraFieldLfngAxin(_simulator=sim,_frequency=ImagF,_
WntMax=WntMax)
extraFieldLfngAxin.setScalarField(LfngAxinField)
steppableRegistry.registerSteppable(extraFieldLfngAxin)

#5
from Somites_Step import ExtraFieldmFgf
extraFieldmFgf=ExtraFieldmFgf(_simulator=sim,_frequency=ImagF,_mfgf0=mf
gf0)
extraFieldmFgf.setScalarField(mFgfField)
steppableRegistry.registerSteppable(extraFieldmFgf)

#6 - Perturbations (Bead Insertion and Tissue Inversion)
if (FgfDiff):
   from Somites_Step import ExtraFieldFgfDiff
   extraFieldFgfDiff=ExtraFieldFgfDiff(_simulator=sim,_frequency=fgfFre
   q,_nLayers=nLayers,_cd=cd,_Freq=ImagF,_s_fgf=s_fgf,_mfgf0=mfgf0,_fgf
   0=fgf0,_k_mfgf=k_mfgf,_k_fgf=k_fgf,_WntMax=WntMax,_WntMin=WntMin,_mF
   gfDecay=mFgfDecay,_margin=margin)
   extraFieldFgfDiff.setScalarField("06_Fgf")  #need to have the same
name of the diffusion filed above
   steppableRegistry.registerSteppable(extraFieldFgfDiff)

#7
from Somites_Step import ExtraFieldFgfCells
extraFieldFgfCells=ExtraFieldFgfCells(_simulator=sim,_frequency=ImagF,_
fgf0=fgf0)
extraFieldFgfCells.setScalarField(FgfCellsField)
steppableRegistry.registerSteppable(extraFieldFgfCells)

#8
from Somites_Step import ExtraFieldWnt
extraFieldWnt=ExtraFieldWnt(_simulator=sim,_frequency=ImagF,_WntMax=Wnt
Max)
extraFieldWnt.setScalarField(WntField)
steppableRegistry.registerSteppable(extraFieldWnt)

CompuCellSetup.mainLoop(sim,simthread,steppableRegistry)
##sys.exit()
```

In the file Somites_Step.py, we code the steppable routines. The roles of the steppables in the simulation are briefly described below:

In **`InitVolSur`**, we set the initial state for each **cell**, including the volume and surface-area constraints and the contents of the **cell's** dictionary, which are initialized to zero. Dictionary contents include the segmentation clock and several "flags" that will trigger events in the **cell's** "life" in the simulation.

The **`Oscillators`** steppable initializes the **Oscillator** class for each **cell**, calls on each **cell's** **Oscillator** class to iterate the segmentation-clock equations, and stores key values generated by the segmentation clock for each **cell**.

The **`DoBoundary`** steppable is where we carry out **cell** differentiation. **Cells** are reassigned new **cell types** according to the criteria described in sections **3.2.5, Clock-wavefront readout** and **3.2.6, Differentiation**.

The **`TailBud`** steppable is responsible for **PSM** growth. It increases **Source cells'** target volumes at each MCS and initiates **Source cell** division in **cells** that reach a critical size. Once a **Source cell** divides, the daughter **cell** that is in contact with Medium at the posterior of the simulation remains a **Source cell** and the daughter **cell** that does not maintain contact with Medium at the posterior of the simulation becomes a **PSM cell**. This maintains the **Source cell** population without allowing stray **Source cells** to "wander" into the **PSM**.

Each of the **`ExtraField`** steppables creates a scalar field for a certain value of interest, *e.g.*, **cells'** Axin2 concentrations, for visualization in the CC3D player as the simulation runs.

Steppables file:

```
########### Somites_Step.py ###########################
#                                                     #
#     Hester, Belmonte, Phys Dept and BIOC @ IU       #
#     2011                                            #
#                                                     #
#######################################################


from PySteppables import *
from SegClock import Oscillator
import CompuCell
import sys,time
import random
import math
from math import *
from random import uniform
from CompuCell import getPyAttrib
from CompuCell import MitosisSimplePlugin
from PlayerPython import *
from copy import deepcopy
import time

global pi; global Amp; global osc_ephrin; global osc_EPH; global fgf_Te
global fgf_Td; global ClockDelay; global Coupling; global initalShift
global timePerMCS; global timeStep; global InteratePerMCS
global FgfExt; global WntExt; global numIntegrationSteps; global tOsc
global Time; global year; global month; global day; global hours
global minutes; global seconds; global s

pi=math.pi
#DIRECTORY NAME:
Time=time.localtime()
year=str(Time[0]); month=str(Time[1]); day=str(Time[2])
hours=str(Time[3]); minutes=str(Time[4]); seconds=str(Time[5])
if (Time[3]<10): hours="0"+hours
if (Time[4]<10): minutes="0"+minutes
if (Time[5]<10): seconds="0"+seconds
#s="Somites_py_"+month+"_"+day+"_"+year+"___"+hours+"_"+minutes+"_"+sec
onds+"/"
s=""


class InitVolSur(SteppablePy):
# The InitVolSur Steppable initializes the cells in the simulation with
#their surface-area and volume constraints, as well as their attached
#cell attributes
   def
__init__(self,_simulator,_frequency,_LamV,_LamS,_tV,_tS,_fgf0,_DiffTh):
      SteppablePy.__init__(self,_frequency)
      self.simulator=_simulator
      self.inventory=self.simulator.getPotts().getCellInventory()
      self.cellList=CellList(self.inventory)
      self.LamV=_LamV; self.LamS=_LamS; self.tV=_tV; self.tS=_tS
      self.fgf0=_fgf0; self.DiffTh=_DiffTh
   def start(self):
      for cell in self.cellList:
         if cell:
            cellDict=CompuCell.getPyAttrib(cell)
            cellDict["mfgf"]=0;      cellDict["fgf"]=0
            cellDict["wnt"]=0;       cellDict["Osc"]=0
            cellDict["oldDelta"]=0;  cellDict["Lfng"]=0
            cellDict["Axin"]=0;      cellDict["Dusp"]=0
            cellDict["Beta"]=0;      cellDict["OnOff"]=1
            cellDict["Bound"]=0;     cellDict["ticker"]=0
            cellDict["ephrinEph"]=0; cellDict["PostDetTime"]=0
            cellDict["DiffLength"]=0
            if (cell.type<=8): #cells
               cell.targetVolume=self.tV;   cell.targetSurface=self.tS
               cell.lambdaVolume=self.LamV
               cell.lambdaSurface=self.LamS
            elif (cell.type>=9): #walls
               cell.targetVolume=0; cell.targetSurface=0
               cell.lambdaVolume=self.LamV*10
               cell.lambdaSurface=self.LamS*10



class mFgfWntPatternWatch(SteppablePy): #optional
# The mFgfWntPatternWatch Steppable outputs files with the values of
#mFgf and Wnt at different positions along the AP axis, averaged over
#the ML dimension. See Figure 3.1 A for an example of the output data.

   def __init__(self,_simulator,_frequency):
      SteppablePy.__init__(self,_frequency)
      self.simulator=_simulator
      self.inventory=self.simulator.getPotts().getCellInventory()
      self.potts=self.simulator.getPotts()
      self.cellField=self.potts.getCellFieldG()
      self.cellFieldG=self.simulator.getPotts().getCellFieldG()
      self.dim=self.cellFieldG.getDim()

   def start(self):
      self.FileNameMF=s+'mFgfPatternXtime.dat'
      self.FileNameF=s+'FgfCellsPatternXtime.dat'
      self.FileNameW=s+'WntPatternXtime.dat'

   def step(self,mcs):
      FileMF=open(self.FileNameMF,'a')
      FileF=open(self.FileNameF,'a')
      FileW=open(self.FileNameW,'a')
      FileMF.write("%s" % (str(mcs)+", "))
      FileF.write("%s" % (str(mcs)+", "))
      FileW.write("%s" % (str(mcs)+", "))
      for z in range(self.dim.z):
         avgMF=0.0; avgF=0.0; avgW=0.0; n=0
         for x in range(self.dim.x):
            point=CompuCell.Point3D(x,0,z)
            cell=self.cellField.get(point)
            if (cell):
               if (cell.type<=7):#12):
                  n+=1; cellDict=CompuCell.getPyAttrib(cell)
                  avgMF+=cellDict["mfgf"]
                  avgF+=cellDict["fgf"]
                  avgW+=cellDict["wnt"]
         if (avgMF!=0):
            FileMF.write("%s" % (str(avgMF/n)+","))
         else:
            FileMF.write("%s" % (str(avgMF)+","))
         if (avgF!=0):
            FileF.write("%s" % (str(avgF/n)+","))
         else:
            FileF.write("%s" % (str(avgF)+","))
         if (avgW!=0):
            FileW.write("%s" % (str(avgW/n)+","))
         else:
            FileW.write("%s" % (str(avgW)+","))
      FileMF.write("%f\n" % (0))
      FileF.write("%f\n" % (0))
      FileW.write("%f\n" % (0))
      FileMF.close(); FileF.close(); FileW.close()



class Oscillators(SteppablePy):
# The Oscillators steppable
   def
__init__(self,_simulator,_frequency,_tPerMCS,_tStep,_initShift,_Couplin
g,_KADL,_WntMax,_fgf0,_pt,_p6,_p7,_p7_r):
      SteppablePy.__init__(self,_frequency)
      self.simulator=_simulator
      self.potts=self.simulator.getPotts()
      self.cellField=self.potts.getCellFieldG()
      self.cellFieldG=self.simulator.getPotts().getCellFieldG()
      self.nTrackerPlugin=CompuCell.getNeighborTrackerPlugin()
      self.inventory=self.simulator.getPotts().getCellInventory()
      self.cellList=CellList(self.inventory)
      self.dim=self.cellFieldG.getDim()
      self.tPerMCS=_tPerMCS; self.tStep=_tStep;
self.initShift=_initShift; self.Coupling=_Coupling; self.KADL=_KADL
      self.WntMax=_WntMax; self.fgf0=_fgf0
      self.pt=_pt; self.p6=_p6; self.p7=_p7; self.p7_r=_p7_r
      self.tOsc=0
      self.FileNameCell=[]; self.counter=0

   def start(self):
      n=0
      for cell in self.cellList:
         if (cell.type<=2):
            n+=1; m=n
      for cell in self.cellList:
         if (cell.type<=2):
            cellDict=CompuCell.getPyAttrib(cell)
            n-=1; cellDict["OnOff"]=1
            print "Initializing cell oscillator:", cell.id, "  cells to
go:", n
            if (self.p6==0):
              if (n==(m-1)):
```

```
cellDict["Osc"]=Oscillator(self.fgf0,self.WntMax,self.KADL,self.Couplin
g,self.tStep)

cellDict["Osc"].PhaseShift(self.fgf0,self.WntMax,self.KADL,self.initShi
ft)
                 OldOsc=cellDict["Osc"]
              else:
                 cellDict["Osc"]=Oscillator(OldOsc)
            else:
              r=random.random(); RandPhaseShift=int(r*self.initShift)

cellDict["Osc"]=Oscillator(self.fgf0,self.WntMax,self.KADL,self.Couplin
g,self.tStep)

cellDict["Osc"].PhaseShift(self.fgf0,self.WntMax,self.KADL,RandPhaseShi
ft)

   def step(self,mcs):

nTrackerAccessor=self.nTrackerPlugin.getNeighborTrackerAccessorPtr()
      IteratePerMCS=int(self.tPerMCS/self.tStep)
      Wnt_ext=self.WntMax; Fgf_ext=self.fgf0
      #UPDATING THE CLOCK:
      for i in range(0,IteratePerMCS):
#range(int(timePerMCS/timeStep)):
         #updating delta expressions
         for cell in self.cellList:
            if (cell.type<6):
               cellDict=CompuCell.getPyAttrib(cell)
               if (cellDict["OnOff"]==1):
                  cellDict["oldDelta"]=cellDict["Osc"].getDLm()
         #going over one iteration
         if (self.Coupling==2 and self.p7==0):
            for cell in self.cellList:
               if (cell.type<6):
                  cellDict=CompuCell.getPyAttrib(cell)
                  if (cellDict["OnOff"]==1):
                     DLext=0; NumNeighbors=0

NeighborList=CellNeighborListAuto(self.nTrackerPlugin,cell)
                     #Coupling with the neighbors
                     for Neighbor in NeighborList:
                        if (Neighbor.neighborAddress and
Neighbor.neighborAddress.type<=5):

neighborDict=CompuCell.getPyAttrib(Neighbor.neighborAddress)
                           if (neighborDict["OnOff"]==1):
                              DLext+=neighborDict["oldDelta"];
NumNeighbors+=1
                     if (NumNeighbors!=0):
                        DLext=DLext/float(NumNeighbors)
                     #updating clock

cellDict["Osc"].Iterate(cellDict["fgf"],cellDict["wnt"],self.KADL,DLext
)
         elif (self.Coupling==2 and self.p7!=0):
            for cell in self.cellList:
               if (cell.type==2):
                  cellDict=CompuCell.getPyAttrib(cell)
                  if (cellDict["OnOff"]==1):
                     DLext=cellDict["oldDelta"]
                  #updating clock

cellDict["Osc"].Iterate(cellDict["fgf"],cellDict["wnt"],self.KADL,DLext
)
               elif (cell.type<6):
                  cellDict=CompuCell.getPyAttrib(cell)
                  if (cellDict["OnOff"]==1):
                     DLext=0; NumNeighbors=0

NeighborList=CellNeighborListAuto(self.nTrackerPlugin,cell)
                     #Coupling with the neighbors
                     for Neighbor in NeighborList:
                        if (Neighbor.neighborAddress and
Neighbor.neighborAddress.type!=2 and Neighbor.neighborAddress.type<=5):

neighborDict=CompuCell.getPyAttrib(Neighbor.neighborAddress)
                           if (neighborDict["OnOff"]==1):
                              DLext+=neighborDict["oldDelta"];
NumNeighbors+=1
                     if (NumNeighbors!=0):
                        DLext=DLext/float(NumNeighbors)
                     #updating clock

cellDict["Osc"].Iterate(cellDict["fgf"],cellDict["wnt"],self.KADL,DLext
)
         #Self-Coupling
         elif (self.Coupling==1):
            for cell in self.cellList:
               if (cell.type<6):
                  cellDict=CompuCell.getPyAttrib(cell)
                  if (cellDict["OnOff"]==1):
                     DLext=cellDict["oldDelta"]
                  #updating clock

cellDict["Osc"].Iterate(cellDict["fgf"],cellDict["wnt"],self.KADL,DLext
)
      #UPDATING PROTEIN EXPRESSIONS:
      for cell in self.cellList:
         if (cell.type<6):
            cellDict=CompuCell.getPyAttrib(cell)
            cellDict["Lfng"]=cellDict["Osc"].getLfng()
            cellDict["Axin"]=cellDict["Osc"].getAxin()
            cellDict["Dusp"]=cellDict["Osc"].getDusp()
            cellDict["Beta"]=cellDict["Osc"].getBeta()



class DoBoundary(SteppablePy):
   def
__init__(self,_simulator,_frequency,_WallOff,_DiffTh,_DiffTh2,_Delay,_s
topDetClocks,_detWait):
      SteppablePy.__init__(self,_frequency)
```

```
        self.simulator=_simulator
        self.inventory=self.simulator.getPotts().getCellInventory()
        self.cellList=CellList(self.inventory)
        self.nTrackerPlugin=CompuCell.getNeighborTrackerPlugin()
        self.nTrackerAccessor =
self.nTrackerPlugin.getNeighborTrackerAccessorPtr()

self.lengthConstraintFlexPlugin=CompuCell.getLengthConstraintLocalFlexP
lugin()
        self.WallOff=_WallOff; self.DiffTh=_DiffTh; self.DiffTh2=_DiffTh2
        self.Delay=_Delay; self.detWait=_detWait
        self.stopDetClocks=_stopDetClocks

    def start(self):
        self.FileNameB=s+'BoundaryWatch.dat'
        self.FileNameS=s+'SomBoundaryWatch.dat'

    def step(self,mcs):
        wall_off=10

nTrackerAccessor=self.nTrackerPlugin.getNeighborTrackerAccessorPtr()
        for cell in self.cellList:
            #DETERMINATION:
            if (cell.type==1): #PSM cell
                cellDict=CompuCell.getPyAttrib(cell)
                if (cellDict["fgf"]<=self.DiffTh and
cellDict["PostDetTime"]==0):
                    cellDict["PostDetTime"]=1
                    if self.stopDetClocks:
                        cellDict["OnOff"]=0
                    Ax=cellDict["Axin"]; Bcat=cellDict["Beta"];
F=cellDict["Lfng"]; fgf=cellDict["fgf"]; wnt=cellDict["wnt"]
                    FileB=open(self.FileNameB,'a')
                    FileB.write("%s\n" % (str(mcs)+", "+str(Ax)+",
"+str(Bcat)+", "+str(F)+", "+str(fgf)+", "+str(wnt)))
                    FileB.close()
                    cellDict["ticker"]=self.Delay
                    Ax=Ax/0.047; Bcat=Bcat/0.58;  F=F/1.0 #1.8  #0.127,
0.87, 2.6
                    #Assigning fates:
                    if (Ax>F):  # #### A
                        cellDict["ephrinEph"]=1
                        if (Bcat>0.7):  # #### B (0.7)
                          cellDict["ephrinEph"]=0
                    else:
                        cellDict["ephrinEph"]=2
                        if (Bcat>0.7):  # #### B (0.7)
                          cellDict["ephrinEph"]=0
                elif (cellDict["PostDetTime"]>=1):
                    if (cellDict["PostDetTime"]>=self.detWait):
                        cellDict["OnOff"]=0
                        #pre-differentiation
                        if cellDict["ephrinEph"]==0:
                            cell.type=5 #pre_In
                        if cellDict["ephrinEph"]==1:
                            cell.type=4 #pre_ephrin
                        if cellDict["ephrinEph"]==2:
```

```
                    cell.type=3 #pre_Eph
                cellDict["PostDetTime"]+=1
            #DIFFERENTIATION:
            if (cell.type>2 and cell.type<=5):
                ok=0; cellDict=CompuCell.getPyAttrib(cell)
                #cheking differentiation method (2nd signal vs delay)
                if (self.DiffTh2!=0):
                    #cheking if fgf is below 2nd threshold
                    if (cellDict["fgf"]<=self.DiffTh2):
                        ok=1
                elif (cellDict["ticker"]>0):
                    #Counting the time
                    cellDict["ticker"]=cellDict["ticker"]-1
                    if (cellDict["ticker"]<1):
                        ok=1
                #intiate differentiation if condition is met
                if (ok==1):
                    cellDict["OnOff"]=0
                    fgf=cellDict["fgf"]; wnt=cellDict["wnt"]
                    FileS=open(self.FileNameS,'a')
                    FileS.write("%s\n" % (str(mcs)+", "+str(fgf)+",
"+str(wnt)))
                    FileS.close()
                    #differentiating
                    if cellDict["ephrinEph"]==1: #potential ephrin cell
                        cell.type=7  #pre_ephrin --> ephrin
                    if cellDict["ephrinEph"]==2: #potential Eph cell
                        cell.type=6  #pre_Eph --> Eph
                    if cellDict["ephrinEph"]==0: #potential Core cell
                        cell.type=8  #pre_In --> In
                    wall_off=cell.zCM/float(cell.volume)
        #WALL DISAPPEARANCE:
        if (self.WallOff):
            for cell in self.cellList:
                if (cell.type==9): #Wall
                    if ((cell.zCM/float(cell.volume))<(wall_off-5)):
                        cell.type=10


class TailBud(SteppablePy):
    def
__init__(self,_simulator,_frequency,_tV,_tS,_LamV,_LamS,_Grow,_p6,_p6_i
n,_KADL,_WntMax,_fgf0,_p7,_p7_r):
        SteppablePy.__init__(self,_frequency)
        self.simulator=_simulator
        self.potts=self.simulator.getPotts()
        self.nTrackerPlugin=CompuCell.getNeighborTrackerPlugin()
        self.inventory=self.potts.getCellInventory()
        self.cellList=CellList(self.inventory)
        self.CellList=CellList(self.inventory)
        self.mitosisPlugin=MitosisSimplePlugin()
        self.mitosisPlugin.setPotts(self.potts)
        self.cellField=self.potts.getCellFieldG()
        self.doublingVolume=0
        self.mitosisPlugin.setDoublingVolume(self.doublingVolume)
        self.mitosisPlugin.turnOn()
```

```
      self.mitosisPlugin.init(self.simulator)
      self.counter=0
      self.mitosisFlag=0
      self.dim=self.potts.getCellFieldG().getDim()
      self.nTrackerAccessor =
self.nTrackerPlugin.getNeighborTrackerAccessorPtr()
      self.tV=_tV; self.tS=_tS; self.LamV=_LamV; self.LamS=_LamS;
self.Grow=_Grow
      self.p6=_p6; self.p6_in=_p6_in; self.KADL=_KADL;
self.WntMax=_WntMax; self.fgf0=_fgf0
      self.firstdiv=99999
      self.p7=_p7; self.p7_r=_p7_r

   def split_cell(self, cell, child_cell_type): #, zCM):
      if (cell):
         split_point = self.get_split_point(cell)
         self.mitosisPlugin.field3DChange(split_point, cell, cell)
         self.mitosisPlugin.setDoublingVolume(cell.volume)
         didMitosis=self.mitosisPlugin.doMitosis()
         if (didMitosis):
            childCell=self.mitosisPlugin.getChildCell()
            parentCell=self.mitosisPlugin.getParentCell()
            if (parentCell and childCell and didMitosis):
               parentCellDict=CompuCell.getPyAttrib(parentCell)
               childCellDict=CompuCell.getPyAttrib(childCell)
               parentCellDict["OnOff"]=1
               for key,item in parentCellDict.items():
                  if (key == "Osc"):
                     parentOsc=parentCellDict["Osc"]
                     childCellDict["Osc"] = Oscillator(parentOsc)
                  else:
                     childCellDict[key] = deepcopy(parentCellDict[key])
            if (parentCell):
               parentCell.targetVolume=self.tV;
parentCell.lambdaVolume=self.LamV
               parentCell.targetSurface=self.tS;
parentCell.lambdaSurface=self.LamS
            if (childCell):
               childCell.type=parentCell.type
               childCell.targetVolume=self.tV;
childCell.lambdaVolume=self.LamV
               childCell.targetSurface=self.tS;
childCell.lambdaSurface=self.LamS
         return 0,0

   def get_split_point(self, cell):  #Customized split point
      volume=float(cell.volume); ok=0
      xCM=cell.xCM/volume; yCM=cell.yCM/volume; zCM=cell.zCM/volume
      split_point =
CompuCell.Point3D(int(xCM+.5),int(yCM+.5),int(zCM+.5))
      test_cell = self.cellField.get(split_point)
      i=0.5; j=0.5; k=0.5
      while (test_cell):
         k+=1
         split_point =
CompuCell.Point3D(int(xCM+i),int(yCM+j),int(zCM+k))
         test_cell = self.cellField.get(split_point)
         if test_cell:
            if (test_cell.id!=cell.id): break
      split_point =
CompuCell.Point3D(int(xCM+.5),int(yCM+.5),int(zCM+k-1))
      test_cell = self.cellField.get(split_point)
      return split_point

   def step(self,mcs):

nTrackerAccessor=self.nTrackerPlugin.getNeighborTrackerAccessorPtr()
      for cell in self.cellList:
         #TAILBUD TYPES:
         if (cell.type==1): #PSM Cell
            med=0; numSource=0;
cellNeighborList=CellNeighborList(nTrackerAccessor,cell)
            for neighbor in cellNeighborList:
               if neighbor:
                  if (neighbor.type==2): #Source cell
                     numSource+=1
               else: #medium
                  med=1
            if (med*numSource>1):   #PSM cells --> Source cells
               if (mcs>2*self.firstdiv):
                  cell.type=2
         elif (cell.type==2): #Source Cell
            med=0;
cellNeighborList=CellNeighborList(nTrackerAccessor,cell)
            for neighbor in cellNeighborList:
               if neighbor:
                  pass
               else: #medium
                  med=1
            if (med==0):  #Source cells --> PSM cells
               if (mcs>self.firstdiv):
                  cell.type=1
                  if (self.p7): #adding a small random phase
                     r=random.random()
                     RandPhaseShift= int(r*self.p7_r)
#int(r*90*self.p6_in/100)
                     cellDict=CompuCell.getPyAttrib(cell)

cellDict["Osc"].PhaseShift(self.fgf0,self.WntMax,self.KADL,RandPhaseShi
ft)
                  if (self.p6):  #adding random phase
                     r=random.random()
                     RandPhaseShift=int(r*90*self.p6_in/100)
                     cellDict=CompuCell.getPyAttrib(cell)

cellDict["Osc"].PhaseShift(self.fgf0,self.WntMax,self.KADL,RandPhaseShi
ft)
            #turning tailbud off when it reaches the bottom
            else:
               zCM=cell.zCM/float(cell.volume)
               if (zCM>(self.dim.z-20)):
                  cell.type=1
         #GROWTH AND DIVISION:
               #growth
```

```
                    if (cell.targetVolume<3*self.tV):
                       cell.targetVolume+=self.Grow
                       cell.targetSurface=5*(cell.targetVolume)**(1/2.)
                    #division
                    if (cell.volume>=2*self.tV):
                       print "mitosis at ", mcs, " cell id =", cell.id
                       self.split_cell(cell,1) #,zCM)
                       if (mcs<self.firstdiv):
                          self.firstdiv=mcs


#1
class ExtraFieldAxin(SteppablePy):
   def __init__(self,_simulator,_frequency,_WntMax):
      SteppablePy.__init__(self,_frequency)
      self.simulator=_simulator
      self.potts=self.simulator.getPotts()
      self.cellField=self.potts.getCellFieldG()
      self.cellFieldG=self.simulator.getPotts().getCellFieldG()
      self.inventory=self.simulator.getPotts().getCellInventory()
      self.cellList=CellList(_inventory=self.inventory)
      self.pixelTrackerPlugin=CompuCell.getPixelTrackerPlugin()
      self.dim=self.cellFieldG.getDim()
      self.WntMax=_WntMax
      self.AxMax=0

   def setScalarField(self,_field):
      self.scalarField=_field

   def start(self):pass

   def step(self,mcs):
      for x in range(self.dim.x):
         for z in range(self.dim.z):
            fillScalarValue(self.scalarField,x,0,z,0)
      for cell in self.cellList:
         if (cell.type<=12):
            cellDict=CompuCell.getPyAttrib(cell)
            pixelList=CellPixelList(self.pixelTrackerPlugin,cell)
            Axin2=cellDict["Axin"]
            if (Axin2>self.AxMax):
               self.AxMax=Axin2; color=Axin2
            NormAxin=1#0.025756*math.exp(1.41138*Wnt)
            for pixelData in pixelList:
               pt=pixelData.pixel

fillScalarValue(self.scalarField,pt.x,pt.y,pt.z,Axin2/NormAxin)


#2
class ExtraFieldLfng(SteppablePy):
   def __init__(self,_simulator,_frequency,_WntMax):
      SteppablePy.__init__(self,_frequency)
      self.simulator=_simulator
      self.potts=self.simulator.getPotts()
      self.cellField=self.potts.getCellFieldG()
      self.cellFieldG=self.simulator.getPotts().getCellFieldG()
      self.inventory=self.simulator.getPotts().getCellInventory()
      self.cellList=CellList(_inventory=self.inventory)
      self.pixelTrackerPlugin=CompuCell.getPixelTrackerPlugin()
      self.dim=self.cellFieldG.getDim()
      self.WntMax=_WntMax
      self.LfngMax=0

   def setScalarField(self,_field):
      self.scalarField=_field

   def start(self):pass

   def step(self,mcs):
      for x in range(self.dim.x):
         for z in range(self.dim.z):
            fillScalarValue(self.scalarField,x,0,z,0)
      for cell in self.cellList:
         if (cell.type<=12):
            cellDict=CompuCell.getPyAttrib(cell)
            pixelList=CellPixelList(self.pixelTrackerPlugin,cell)
            Lfng=cellDict["Lfng"]
            if (Lfng>self.LfngMax):
               self.LfngMax=Lfng; color=Lfng
            #LfMax=0.32 + 0.0588*self.WntMax
            NormLf=1#2.2067+0.2369*Wnt
            for pixelData in pixelList:
               pt=pixelData.pixel

fillScalarValue(self.scalarField,pt.x,pt.y,pt.z,Lfng/NormLf)


#3
class ExtraFieldDusp(SteppablePy):
   def __init__(self,_simulator,_frequency):
      SteppablePy.__init__(self,_frequency)
      self.simulator=_simulator
      self.potts=self.simulator.getPotts()
      self.cellField=self.potts.getCellFieldG()
      self.cellFieldG=self.simulator.getPotts().getCellFieldG()
      self.inventory=self.simulator.getPotts().getCellInventory()
      self.cellList=CellList(_inventory=self.inventory)
      self.pixelTrackerPlugin=CompuCell.getPixelTrackerPlugin()
      self.dim=self.cellFieldG.getDim()

   def setScalarField(self,_field):
      self.scalarField=_field

   def start(self):pass

   def step(self,mcs):
      color=10.1
      for x in range(self.dim.x):
         for z in range(self.dim.z):
            fillScalarValue(self.scalarField,x,0,z,0)
      for cell in self.cellList:
         if (cell.type<=12):
            cellDict=CompuCell.getPyAttrib(cell)
```

```
                pixelList=CellPixelList(self.pixelTrackerPlugin,cell)
                Dusp=cellDict["Dusp"]
                for pixelData in pixelList:
                   pt=pixelData.pixel
                   fillScalarValue(self.scalarField,pt.x,pt.y,pt.z,Dusp)


#4
class ExtraFieldLfngAxin(SteppablePy):
   def __init__(self,_simulator,_frequency,_WntMax):
      SteppablePy.__init__(self,_frequency)
      self.simulator=_simulator
      self.potts=self.simulator.getPotts()
      self.cellField=self.potts.getCellFieldG()
      self.cellFieldG=self.simulator.getPotts().getCellFieldG()
      self.inventory=self.simulator.getPotts().getCellInventory()
      self.cellList=CellList(_inventory=self.inventory)
      self.pixelTrackerPlugin=CompuCell.getPixelTrackerPlugin()
      self.dim=self.cellFieldG.getDim()
      self.WntMax=_WntMax

   def setScalarField(self,_field):
      self.scalarField=_field

   def start(self):pass

   def step(self,mcs):
      AxMax=0.025756*math.exp(1.41138*self.WntMax) #-0.08 +
0.03*math.exp(self.WntMax/0.49)
      LfMax=2.2067+0.2369*self.WntMax #2.6
      for x in range(self.dim.x):
         for z in range(self.dim.z):
            fillScalarValue(self.scalarField,x,0,z,0)
      for cell in self.cellList:
         if (cell.type<=12):
            cellDict=CompuCell.getPyAttrib(cell)
            pixelList=CellPixelList(self.pixelTrackerPlugin,cell)
            Lfng=cellDict["Lfng"]
            Axin=cellDict["Axin"]
            for pixelData in pixelList:
               pt=pixelData.pixel
               if (pt.x<(self.dim.x/2-1)):

fillScalarValue(self.scalarField,pt.x,pt.y,pt.z,Lfng/LfMax)
               if (pt.x>(self.dim.x/2+1)):

fillScalarValue(self.scalarField,pt.x,pt.y,pt.z,Axin/AxMax)


#5
class ExtraFieldmFgf(SteppablePy):
   def __init__(self,_simulator,_frequency,_mfgf0):
      SteppablePy.__init__(self,_frequency)
      self.simulator=_simulator
      self.potts=self.simulator.getPotts()
      self.cellField=self.potts.getCellFieldG()
      self.cellFieldG=self.simulator.getPotts().getCellFieldG()
      self.inventory=self.simulator.getPotts().getCellInventory()
      self.cellList=CellList(_inventory=self.inventory)
      self.pixelTrackerPlugin=CompuCell.getPixelTrackerPlugin()
      self.dim=self.cellFieldG.getDim()
      self.mfgf0=_mfgf0

   def setScalarField(self,_field):
      self.scalarField=_field

   def start(self):pass

   def step(self,mcs):
      color=self.mfgf0
      for x in range(self.dim.x):
         for z in range(self.dim.z):
            fillScalarValue(self.scalarField,x,0,z,0)
      for cell in self.cellList:
         if (cell.type<=12):
            cellDict=CompuCell.getPyAttrib(cell)
            pixelList=CellPixelList(self.pixelTrackerPlugin,cell)
            mFgf=cellDict["mfgf"]
            for pixelData in pixelList:
               pt=pixelData.pixel
               fillScalarValue(self.scalarField,pt.x,pt.y,pt.z,mFgf)


#6 - Diffusive Fgf/Perturbations
class ExtraFieldFgfDiff(SteppablePy):
   def
__init__(self,_simulator,_frequency,_nLayers,_cd,_Freq,_s_fgf,_mfgf0,_f
gf0,_k_mfgf,_k_fgf,_WntMax,_WntMin,_mFgfDecay,_margin):
      SteppablePy.__init__(self,_frequency)
      self.simulator=_simulator
      self.potts=self.simulator.getPotts()
      self.cellField=self.potts.getCellFieldG()
      self.cellFieldG=self.simulator.getPotts().getCellFieldG()
      self.inventory=self.simulator.getPotts().getCellInventory()
      self.cellList=CellList(_inventory=self.inventory)
      self.dim=self.cellFieldG.getDim()
      self.nTrackerPlugin=CompuCell.getNeighborTrackerPlugin()
      self.nTrackerAccessor =
self.nTrackerPlugin.getNeighborTrackerAccessorPtr()
      self.pixelTrackerPlugin=CompuCell.getPixelTrackerPlugin()
      self.nLayers=_nLayers; self.cd=_cd; self.Freq=_Freq
      self.s_fgf=_s_fgf; self.mfgf0=_mfgf0; self.fgf0=_fgf0;
self.k_mfgf=_k_mfgf; self.k_fgf=_k_fgf
      self.WntMax=_WntMax; self.WntMin=_WntMin;
self.mFgfDecay=_mFgfDecay; self.margin=_margin

   def setScalarField(self,_field):
      self.scalarField=_field

   def start(self):

field=CompuCell.getConcentrationField(self.simulator,self.scalarField)
      #Calculating the initial wnt and mfgf gradients on the cells
      for cell in self.cellList:
         if (cell.type<=2):
            cellDict=CompuCell.getPyAttrib(cell)
            cellDict["mfgf"]=self.mfgf0
            cellDict["wnt"]=self.WntMax
            value=self.fgf0
            #calculating the initial fgf signal gradient
            z=int(cell.zCM/float(cell.volume))
            x=int(cell.xCM/float(cell.volume))
            if (cell.type==1):
               mfgf=self.mfgf0-0.2*(((50+self.nLayers+self.cd/2)-
z)/self.cd)
               cellDict["mfgf"]=mfgf
               ratio=mfgf/self.mfgf0 # WntN = {0,1}
               cellDict["wnt"]=self.WntMin + (self.WntMax-
self.WntMin)*ratio
               value=self.fgf0*ratio
            pixelList=CellPixelList(self.pixelTrackerPlugin,cell)
            for pixelData in pixelList:
               point=pixelData.pixel
               field.set(point,value)
               #Updating cells fgf dictionary:
               if (point.x==x and point.z==z):
                  cellDict["fgf"]=value
      #recording initial fgf levels
      n = int((self.dim.x-2*self.margin)/(2*self.cd))
      self.FileNameF=[]; self.FileNameMF=[]
      self.FileNameAx=[]; self.FileNameLf=[]
      for i in range(1,n+1):
         self.FileNameF.append(s+'FgfPatternXtime--'+str(i)+'.dat')
         self.FileNameMF.append(s+'mFgfPatternXtime--'+str(i)+'.dat')
         self.FileNameAx.append(s+'AxinPatternXtime--'+str(i)+'.dat')
         self.FileNameLf.append(s+'LfngPatternXtime--'+str(i)+'.dat')

   def step(self,mcs):

field=CompuCell.getConcentrationField(self.simulator,self.scalarField)

nTrackerAccessor=self.nTrackerPlugin.getNeighborTrackerAccessorPtr()
      for cell in self.cellList:
         if (cell.type!=2):
            if (cell.type<=5):
               cellDict=CompuCell.getPyAttrib(cell)
               inc=self.s_fgf*cellDict["mfgf"]
               z=int(cell.zCM/float(cell.volume))
               x=int(cell.xCM/float(cell.volume))
               pixelList=CellPixelList(self.pixelTrackerPlugin,cell)
               for pixelData in pixelList:
                  point=pixelData.pixel
                  value=field.get(point)
                  #updating field
                  field.set(point,value+inc)
                  if (point.x==x and point.z==z):
                     #Updating cells fgf dictionary:
                     cellDict["fgf"]=value
                     #Updating Wnt dictionary
                     WntN=cellDict["mfgf"]/self.mfgf0 # WntN = {0,1}
                     #WntN=value/self.fgf0 # WntN = {0,1}
```

```
                    cellDict["wnt"]=self.WntMin + (self.WntMax-
self.WntMin)*WntN
                #Updating cells mfgf dictionary
                if self.mFgfDecay==1: #Linear decay
                   cellDict["mfgf"]-=self.k_mfgf
                   if (cellDict["mfgf"]<0):
                      cellDict["mfgf"]=0
                elif (self.mFgfDecay==2): #Exponential decay
                   cellDict["mfgf"]-=self.k_mfgf*cellDict["mfgf"]
          #Updating remaining cells mfgf and wnt dictionary
             elif (cell.type<=8):
                cellDict=CompuCell.getPyAttrib(cell)
                WntN=cellDict["mfgf"]/self.mfgf0 # WntN = {0,1}
                cellDict["wnt"]=self.WntMin + (self.WntMax-
self.WntMin)*WntN
                if self.mFgfDecay==1: #Linear decay
                   cellDict["mfgf"]-=self.k_mfgf
                   if (cellDict["mfgf"]<0):
                      cellDict["mfgf"]=0
                elif (self.mFgfDecay==2): #Exponential decay
                   cellDict["mfgf"]-=self.k_mfgf*cellDict["mfgf"]
          #mantaining high levels of fgf in Source cells
          else:
             pixelList=CellPixelList(self.pixelTrackerPlugin,cell)
             for pixelData in pixelList:
                point=pixelData.pixel
                field.set(point,self.fgf0)
       #Recording fgf levels
       if (mcs%(self.Freq*15)==0):
          n=int((self.dim.x-2*self.margin)/(2*self.cd))
          for i in range(0,n):
             FileF=open(self.FileNameF[i],'a')
             FileF.write("%s" % (str(mcs)+", "))
             FileMF=open(self.FileNameMF[i],'a')
             FileMF.write("%s" % (str(mcs)+", "))
             FileAx=open(self.FileNameAx[i],'a')
             FileAx.write("%s" % (str(mcs)+", "))
             FileLf=open(self.FileNameLf[i],'a')
             FileLf.write("%s" % (str(mcs)+", "))
             for z in range(self.dim.z):
                point=CompuCell.Point3D(self.margin+i*5,0,z)
                value=str(field.get(point))
                FileF.write("%s" % (value+","))
                cell=self.cellField.get(point)
                if (cell and cell.type<=7):
                   #FileF.write("%s" % (value+","))
                   cellDict=CompuCell.getPyAttrib(cell)
                   value=str(cellDict["mfgf"])
                   FileMF.write("%s" % (value+","))
                   value=str(cellDict["Axin"])
                   FileAx.write("%s" % (value+","))
                   value=str(cellDict["Lfng"])
                   FileLf.write("%s" % (value+","))
                else:
                   #FileF.write("%s" % (str(0)+","))
                   FileMF.write("%s" % (str(0)+","))
                   FileAx.write("%s" % (str(0)+","))
                   FileLf.write("%s" % (str(0)+","))
             FileF.write("%f\n" % (0)); FileF.close()
             FileMF.write("%f\n" % (0)); FileMF.close()
             FileAx.write("%f\n" % (0)); FileAx.close()
             FileLf.write("%f\n" % (0)); FileLf.close()


 #7
 class ExtraFieldFgfCells(SteppablePy):
    def __init__(self,_simulator,_frequency,_fgf0):
       SteppablePy.__init__(self,_frequency)
       self.simulator=_simulator
       self.potts=self.simulator.getPotts()
       self.cellField=self.potts.getCellFieldG()
       self.cellFieldG=self.simulator.getPotts().getCellFieldG()
       self.inventory=self.simulator.getPotts().getCellInventory()
       self.cellList=CellList(_inventory=self.inventory)
       self.pixelTrackerPlugin=CompuCell.getPixelTrackerPlugin()
       self.dim=self.cellFieldG.getDim()
       self.fgf0=_fgf0

    def setScalarField(self,_field):
       self.scalarField=_field

    def start(self):pass

    def step(self,mcs):
       color=self.fgf0
       for x in range(self.dim.x):
          for z in range(self.dim.z):
             fillScalarValue(self.scalarField,x,0,z,0)
       for cell in self.cellList:
          if (cell.type<=12):
             cellDict=CompuCell.getPyAttrib(cell)
             pixelList=CellPixelList(self.pixelTrackerPlugin,cell)
             fgf=cellDict["fgf"]
             for pixelData in pixelList:
                pt=pixelData.pixel
                fillScalarValue(self.scalarField,pt.x,pt.y,pt.z,fgf)


 #8
 class ExtraFieldWnt(SteppablePy):
    def __init__(self,_simulator,_frequency,_WntMax):
       SteppablePy.__init__(self,_frequency)
       self.simulator=_simulator
       self.potts=self.simulator.getPotts()
       self.cellField=self.potts.getCellFieldG()
       self.cellFieldG=self.simulator.getPotts().getCellFieldG()
       self.inventory=self.simulator.getPotts().getCellInventory()
       self.cellList=CellList(_inventory=self.inventory)
       self.pixelTrackerPlugin=CompuCell.getPixelTrackerPlugin()
       self.dim=self.cellFieldG.getDim()
       self.WntMax=_WntMax

    def setScalarField(self,_field):
       self.scalarField=_field
```

```
def start(self):pass

def step(self,mcs):
   color=self.WntMax
   for x in range(self.dim.x):
      for z in range(self.dim.z):
         fillScalarValue(self.scalarField,x,0,z,0)
   for cell in self.cellList:
      if (cell.type<=12):
         cellDict=CompuCell.getPyAttrib(cell)
         pixelList=CellPixelList(self.pixelTrackerPlugin,cell)
         wnt=cellDict["wnt"]
         for pixelData in pixelList:
            pt=pixelData.pixel
            fillScalarValue(self.scalarField,pt.x,pt.y,pt.z,wnt)
```

Appendix D

RUNNING THE SIMULATIONS

Required Software

1) Python (available for download at http://www.python.org)
2) CompuCell3D (available for download at http://www.compucell3d.org)
3) CMake (required for Linux or Mac OS X only, available for download at http://cmake.org/cmake/resources/software.html)

Simulation Files

The simulation source code is available in a zip file (*BelmonteHesterSomite.zip*) at http://www.compucell3d.org/BelmonteHesterSomite. It includes the following files:

1) *Somites.py* (The main CC3D/Python file with the model parameters)
2) *Somites_Step.py* (The CC3D/Python file were all calculations are implemented)
3) *SegClock.cpp* (C++ code implementing the segmentation-clock)
4) *SegClock.h* (C++ header file)
5) *somite_demo_setup.exe* (Windows setup program)
6) *CMakeLists.txt* (Setup file for Linux/OS X users)

Preparing the simulation to run

For **Windows** users:

1) Decompress the ZIP file in a convenient location. Double-click on *somite_demo_setup.exe* .

2) When prompted for an installation destination, select the folder CompuCell3D was installed in (typically C:\Program Files (x86)\CompuCell3D\).

For **Linux** or **Mac** users:

1) Decompress the ZIP file in a convenient location.
2) Start the CMake GUI, and click on Browse Source. Select the folder where the ZIP file was decompressed.
3) Click on Browse Build and select the folder where the ZIP file was decompressed.
4) Click on Configure. When prompted, select “Unix Makefiles,” and “Use default native compilers,” and click Done.
5) Click on “COMPUCELL3D_PATH” and enter the location where CompuCell3D was installed, and click Configure.
6) Click on Generate.
7) Exit CMake.
8) From a Terminal window, browse to the folder where the ZIP file was decompressed.
9) Type “make” and press return. When it has completed type “make install” and press return.

Running the simulation in CC3D

1) Start CompuCell3D.
2) From the file menu, select Open Simulation File.
3) Browse to the CompuCell3D Demos folder.
4) Inside the folder BelmonteHesterSomite, select *Somites.py*, and click Open.
5) Click the Play button in the upper left hand side of the CompuCell3D window.

Changing simulation parameters

The downloadable files are configured with the reference simulation parameters. A user can modify the parameter settings within the main Python file (*Somites.py*). Examples of key parameter changes described in this thesis follow.

Altering the segmentation-clock period

The parameters controlling the behavior of the segmentation-clock network can be found under the heading `#OSCILLATOR PARAMETERS`. The segmentation-clock period can be changed by altering the relationship between three parameters: `oscFreq`, `tStep` and `tPerMCS`. Reference values are given below:

```
oscFreq=1              #Frequency of oscillator calls (per MCS)
tStep=0.015            #size of the integration step
tPerMCS=tStep*oscFreq  #time per MCS
```

`oscFreq` sets the number of times that the segmentation-clock equations are iterated during a Monte Carlo step, while `tStep` sets the integration time step for the segmentation-clock equations. By default, the time per MCS (`tPerMCS`) is set to the number of segmentation-clock iterations performed per MCS multiplied by the integration step. By altering the relationship between “segmentation-clock time” and “MCS time,” a user can change the effective period of the segmentation clock. For example, the following parameter settings would double the segmentation-clock period compared to the reference simulation:

```
oscFreq=1              #Frequency of oscillator calls (per MCS)
tStep=0.015            #size of the integration step
tPerMCS=2*tStep*oscFreq  #time per MCS
```

Altering the PSM growth rate

The parameter controlling the PSM growth rate is found under the commented header `#OTHER PARAMETERS (flags)`. The parameter `Grow` sets the number of square pixels that will be added to the **Source cells'** target volume per MCS. Increasing `Grow` results in faster PSM growth and decreasing `Grow` results in slower PSM growth.

Altering cell motility

The degree of cell motility in the simulation can be adjusted by changing the parameter `LamS`, under the commented heading `#VOLUME/SURFACE PARAMETERS`. The default value of `LamS` is 15. Increasing `LamS` results in "stiffer" cells, while decreasing `LamS` results in more motile cells. The connection between the `LamS` parameter and cell motility is described in section **3.1.2.2, Cell motility**.

Coupling and uncoupling neighboring cells' segmentation clocks

The parameter controlling segmentation-clock coupling among neighbors is under the commented heading `#OSCILLATOR PARAMETERS`. When the parameter `Coupling` is set to 2, neighboring clocks are coupled through Delta/Notch signaling; when `Coupling` is set to 1, individual **cells** receive Delta signaling equivalent to their own outgoing Delta signal; when `Coupling` is set to 0, the **cells** receive a constant level of Delta signaling.

Susan D Hester

Molecular and Cellular Biology Department, University of Arizona
Life Sciences South 244
1007 E. Lowell Street
Tucson, Arizona 85721
(812) 345-7174
sdhester@email.arizona.edu

September 2011

EDUCATION

PhD Indiana University Bloomington
Biophysics, April 2012

Thesis: *Multi-scale cell-based computational models of vertebrate segmentation and somitogenesis illuminate coordination of developmental mechanisms across scales*
Prof James Glazier (advisor)

MS Indiana University Bloomington
Biophysics, December 2008

BS Arizona State University
Physics, May 2006 *Summa cum laude*

POSTDOCTORAL POSITIONS

August 2011 – Present: University of Arizona Molecular and Cellular Biology
Science Education Research: Integrating quantitative thinking into an introductory university biology curriculum

PROFESSIONAL INTERESTS

Science Education
Critical-thinking-based approaches to teaching the sciences
Teaching Biology as an integrated science
Defining student success in the sciences

Developmental Biology
Emergent pattern formation in development
Robustness and evolutionary flexibility of developmental algorithms
Integrating experimental and computational approaches

PUBLICATIONS

Refereed

Hester SD, Belmont JM, Gens JS, Clendenon SG, Glazier JA. A multi-cell, multi-scale model of vertebrate segmentation and somite formation. (under review by *PLoS Comp Biol*)

Textbook chapters

Swat MH, Hester SD, Balter AI, Heiland RW, Zaitlen BL, et al. (2009) Multicell simulations of development and disease using the CompuCell3D simulation environment. *Methods Mol Biol* 500: 361–428.

ORAL CONFERENCE PRESENTATIONS

Hester, SD, Belmonte, JM, Gens, JS, Clendenon, SG, Glazier, JA. A novel multi-scale model explores our current understanding of vertebrate segmentation and somite formation and provides a tool for performing experiments in silico. *Experimental Biology 2011*.

Hester, SD, Belmonte, JM, Glazier, JA. Avian Somitogenesis as a Model for Building Multi-Scale Cell-Based Simulations. *Biocomplexity Workshop X (2009): Quantitative Tissue Biology and Virtual Tissues.*

TEACHING EXPERIENCE

Introductory Physics I – *Introduction to mechanics (algebra-based)*
Associate Instructor
Recitation/Discussion: Fall 2006, Spring 2007, Fall 2007
Laboratory: Fall 2006, Summer 2007

General Physics II – *Introduction to electricity, magnetism and modern physics (calculus-based)*
Associate Instructor
Recitation/Discussion: Spring 2008

Physical Science through Inquiry – *Introduction to physics and hands-on physics instruction for elementary education majors*
Associate Instructor
Laboratory: Spring 2008

TEACHING AWARDS

Joseph and Sophia Konopinski Award 2007
Departmental award for excellent physics instruction

OUTREACH

Indiana University Physics and Astronomy Open House Volunteer
Indiana University Bloomington, Fall 2006-2010

Indiana Science Olympiad State Tournament Competitive Event Co-Coordinator
Indiana University Bloomington, March 2010

Indiana University Physics Forum Volunteer Tutor
Fall 2006-Spring 2008