# SIMULATION OF FLUID MIXING IN THE SMALL INTESTINE

---

A Thesis
Submitted to
the Temple University Graduate Board

---

In Partial Fulfillment
of the Requirements for the Degree
MASTER OF SCIENCE OF MECHANICAL ENGINEERING

---

by
Perry S. Orthey
July 2015

Examining Committee Members:

Dr. Dmitri Vainchtein, Thesis Advisor, Mechanical Engineering
Dr. Henry Parkman, Temple University Hospital
Dr. Shriram Pillapakkam, Mechanical Engineering

# ABSTRACT

There are many gastrointestinal diseases, such as Crohn's disease, which can be treated effectively with topical, localized medicine delivered to the intestinal wall through the gastrointestinal tract by the use of a targeted drug delivery capsule. The success of such a delivery method is contingent on the properties of the fluid flow near the delivery site; specifically, how well-mixed the medicine will be in the chyme so that it can act on the intestinal wall. Pursuant to understanding the mixed state of the chyme, several fluid simulations were performed with ANSYS Fluent, simulating different types of muscular contractions. Fluid particles – which were originally segregated into three sections of the simulated small intestine – were tracked, and simulation results were compared based on how well-distributed the tracked particles were at the end state, using the second moment of distribution. The results of this comparison have revealed that there is little difference between the mixing produced by segmentation in a 3.0 cm diameter small intestine and that produced in a 2.0 cm diameter small intestine. Results have also shown little difference between mixing produced when the segmentation contractions vary qualitatively in any of several ways. There is, however, some difference between distribution produced by segmentation contractions and peristaltic, or propulsive, contractions. This work could be further pursued with more simulations; of different types of contractions, of contraction patterns with different properties, and with simulations with more comprehensive particle tracking. It would also be straightforward to incorporate analysis of the large intestine into the study.

# TABLE OF CONTENTS

Page

# LIST OF TABLES

# LIST OF FIGURES

CHAPTER 1

# INTRODUCTION

After food has been processed in the stomach, it moves into the small intestine. When the food first enters the small intestine, it is usually broken down and dissolved into a fairly homogeneous liquid called chyme. As chyme is propelled through the small intestine by peristaltic impulses, the nutrients in the chyme will be transferred to the bloodstream through the intestinal wall. There are also frequent segmentation contractions which serve to mix the fluid in the small intestine to promote diffusivity at the intestinal wall.

The small intestine in a living adult is approximately seven meters long. Transit time through the small intestine varies widely among individuals. One study concluded that traces of barium moved through the small intestine in less than two hours for over 80% of cases, but took up to five hours in some cases (Kim, 1968). The purpose of the small intestine is to finish breaking down the food that is passed to it from the stomach through the pyloric sphincter, and to absorb the nutrients from the broken down food through the intestinal walls. This means that chyme close to the intestinal walls quickly becomes empty of nutrients as they are absorbed. It is therefore desirable to know, for the purposes of understanding healthy digestion, how well-mixed small intestine contents become due to natural muscular behavior of the small intestine.

Another reason we would like to study the state of fluid mixing in the small intestine is to investigate the effectiveness that targeted delivery medicine has in traversing from the point of release to the intestinal wall. Previous studies have already

found that certain targeted drug delivery methods are imprecise. One such study, published by Alimentary Pharmacology and Therapeutics, reports: "If the delay is chosen to be, for instance, 1 h followed by an immediate release, then the target will often be missed. With a shorter delay followed by an extended release, the chance to hit the target area is higher (at the expense of a higher loss outside the target area)." (Wikberg, Ulmius, & Ragnarsson, 1997). While it is difficult to precisely predict what kinds of flows will be in the intestine at the time of release, especially when motility patterns between individuals are so diverse, understanding the typical mixing accomplished by regular contractions could give some insight into how best to schedule the release of these targeted-delivery drugs.

The main concern of the research is with mixing in the transversal direction. Specifically, the interest is to determine if particles which are originally concentrated near the center of the lumen end up well-distributed throughout the lumen. Therefore, the most defining characteristic of any particle's position is its perpendicular distance from the centerline of the intestine. We performed simulations of small intestine contractions with different parameters, and with different types of contractions. For each simulation, we tracked particles that started in one of three concentric, equivolumetric sections. The mixed state of the tracked particles was quantified using the second moment of distribution. Simulations were performed using ANSYS Fluent, and particle positions were exported and tabulated with CFD-Post, to be analyzed using MATLAB.

## Current State of Research

In order for the simulations to achieve maximum usefulness in the medical field, they should be constructed using conditions as close as possible to the conditions in a real

human small intestine. However, after investigating the current literature in the medical fields of gastroenterology and gastrointestinal motility, it looks as though these parameters are not known. One review claims, “No single method or preparation informs comprehensively on the movements of the gastro-intestinal tract.” (Shulze, 2014). Most of the experiments performed before now have been based on pressure readings and electrical sensory input to the intestinal muscles (Liu, Huang, Huang, Liu, & Chen, 2013). Very little research is being pursued in this field to obtain visual imagery of contractions of a healthy small intestine. The main reason for this is the difficulty of obtaining such imagery, especially in humans (Shulze, 2014). This visual imagery is essential to the furtherance of these numerical simulations, as accurate simulations are impossible without knowing the typical dimensions of intestinal contractions. It is also possible that further work on these numerical simulations will lead to new developments to obtain the real-life images.

Simulations of fluid flow in the small intestine were discussed in several publications. Much of the previous work has been done mathematically. For example, Lew, Fung, and Lowenstein found solutions to a simplified Navier-Stokes equation using boundary conditions similar to those in the small intestine. However, they did not have access to actual values of some parameters. Instead of finding what the actual fluid behavior is, they found an array of solutions using different combinations of unknown parameters (Lew, Fung, & Lowenstein, 1971). More recent studies have also used this array-based approach in absence of real-life values for some parameters (Roy, Rios, & Riahi, 2011). This approach is useful in determining the effects that some parameters have on the fluid flow, but it is insufficient for studying the actual fluid flow. The present

study has adopted a version of this parametric analysis, to examine the effects that some parameters have on the second moment of distribution.

## Segmentation and Peristalsis

The purpose of segmentation in the small intestine is to ensure that the contents of the chyme are fairly homogeneous, so that when nutrients are slowly absorbed at the intestinal walls, the chyme near the center, which has not yet been tapped for nutrients, can access the intestinal wall. Segmental contractions cause the chyme to move back and forth in the small intestine, causing no net forward motion, so that the nutritious chyme stays in the small intestine for a long enough time to be completely digested.

Peristalsis is another type of contraction occurring in the small intestine. Peristalsis predominates in the small intestine during phase III of the migrating motor complex (Liu, Huang, Huang, Liu, & Chen, 2013), and is responsible for most intestinal noises associated with being hungry. Instead of repeating in a cycle along a length of intestine, however, peristaltic contractions appear at one point and propagate along a length of intestine.

CHAPTER 2

# SIMULATIONS

All simulations were performed in ANSYS Fluent, by using user-defined functions, or UDFs, to move the intestinal wall according to our specifications. UDFs used are given in Appendix A.

## Segmentation

We performed several different simulations of segmentation in the small intestine to study the effects that intestinal diameter and contraction patterns have on the mixing. For one case, the intestinal inner diameter was taken as 3.0 cm, for the other, 2.0 cm. This is a commonly accepted range of small intestine lumen diameters in healthy individuals. We also investigated the effects of certain contraction patterns; by switching the nodes of contraction with the antinodes after two full contractions, by decreasing the wavelength of contractions, and by lowering the frequency. This was to see if different sorts of contractile behavior have significant effects on the mixing of particles.

Most contractions were assumed to be periodic with a period of two seconds. This means that it takes two seconds for any point on the intestinal wall to complete a full cycle back to its original position with its original velocity. The density of the fluid was set as that of water, while the viscosity was set to ten times that of water, similar to thick soup, which is a common estimation. Finally, the contraction distance was two-thirds of the diameter of the intestine, so that two-thirds of the diameter of the lumen was covered by the contraction at the peak of the contraction wave.

The first set of simulations for small intestine segmentation was performed assuming that the contractions occur in exactly the same positions for long periods of time. The second set of simulations was performed assuming that the nodes and antinodes of the contractions switch after two full cycles of contractions. This was done to see the effects on the mixing of switching the nodes and antinodes. Results are presented in Figure 1.

The mixing results for these two simulations were qualitatively similar. The quantitative description for mixing used was also similar for the two cases; the values of the second moment of distribution between the two cases were not very different. These values are given later.

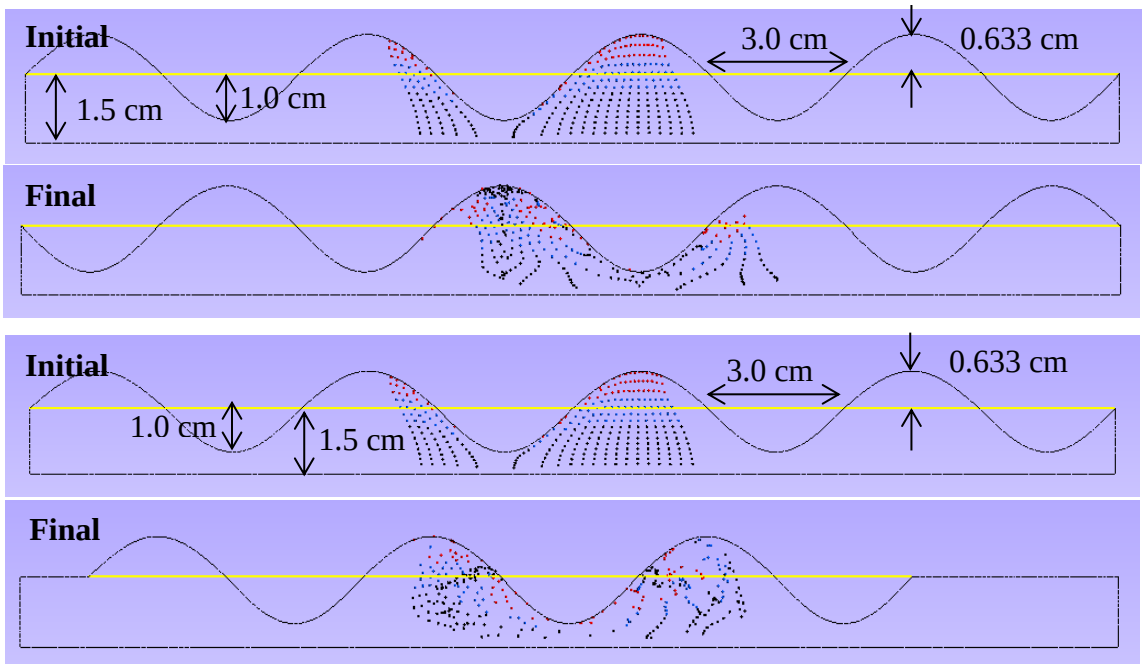


**Figure 1: Segmentation waves in the 3 cm small intestine. Two images are shown from each simulation to demonstrate the mixing accomplished by the simulation. The top two images are taken from one set of simulations, the bottom two images from another. Both simulate 24.0 cm of small intestine undergoing periodic segmental contractions. The contractions of the simulation in the top two images are completely stationary, while those in the bottom two images move once by a quarter of the wavelength halfway through the simulation. As it is an axisymmetric setup, only half of the diameter of the small intestine is shown. The horizontal black line in each image represents the axis of the intestine, around which there is assumed to be symmetry. The yellow line is the position of the intestinal wall when there are no contractions. The particles are colored according to their initial transversal position: black particles started closest to the axis, blue particles in the mid region, and red particles started closest to the intestinal wall. To ensure conservation of volume, the intestine relaxes a shorter distance than it contracts. These still images were taken some time after the start of their respective simulations. In both sets of images, the top image was taken after 0.5 seconds, and the bottom after 7.5 seconds.**

The next images in Figure 2 are from simulations of segmentation in the small intestine assuming a lumen diameter of 2.0 cm. Other than the diameter, they have exactly the same setup as the first and second simulations.

Although these four simulations indicate that mixing may be established, we currently do not have the information we need to make the simulations completely accurate. We have assumed values of many parameters – such as time period, length of affected intestine, contraction distance, and others – which are critical to the flow, and we would like to use actual values of these parameters. However, the quantitative description

of mixing we used gives a rudimentary comparison between the four simulations' results, seen in Table 1.

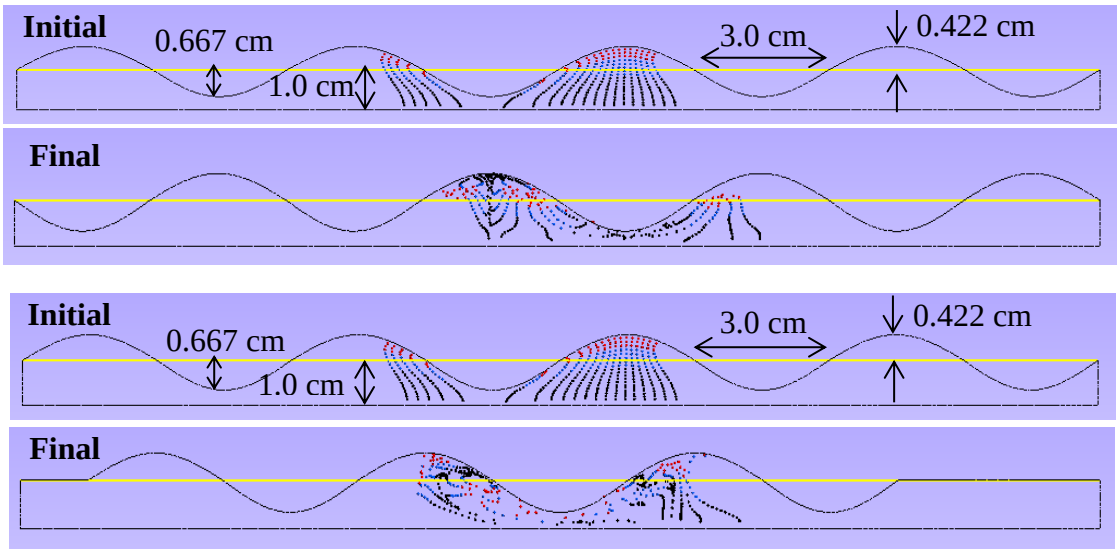


**Figure 2: Segmentation waves in the 2 cm small intestine. Two images are shown from each simulation to demonstrate the mixing accomplished by the simulation. The top two images are taken from one set of simulations, the bottom two images from another. Both simulate 24.0 cm of small intestine undergoing periodic segmental contractions. The contractions of the simulation in the top two images are completely stationary, while those in the bottom two images move once by a quarter of the wavelength halfway through the simulation. As it is an axisymmetric setup, only half of the diameter of the small intestine is shown. The horizontal black line in each image represents the axis of the intestine, around which there is assumed to be symmetry. The yellow line is the position of the intestinal wall when there are no contractions. The particles are colored according to their initial transversal position: black particles started closest to the axis, blue particles in the mid region, and red particles started closest to the intestinal wall. To ensure conservation of volume, the intestine relaxes a shorter distance than it contracts. These still images were taken some time after the start of their respective simulations. In both sets of images, the top image was taken after 0.5 seconds, and the bottom after 7.5 seconds.**

Other parameters varied were wavelength of contractions, and frequency of contractions. Wavelength was changed by decreasing the distance that a single contraction covers, so that there are more total contractions on the 24 cm length of intestine simulated.

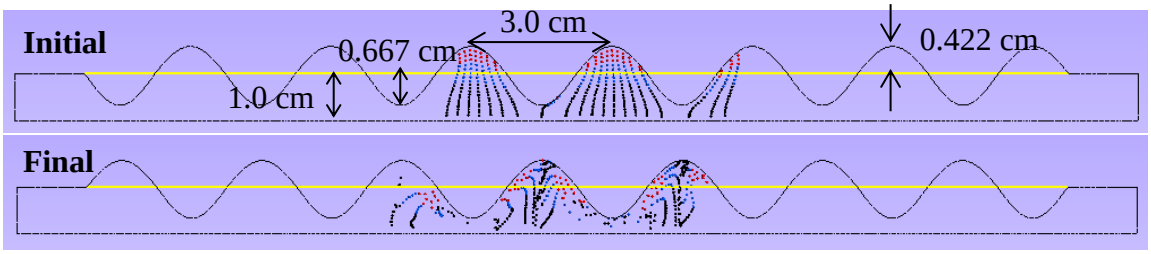


**Figure 3: Shorter wavelength segmentation waves. Two images are shown to demonstrate the mixing accomplished by the simulation. All spatial parameters are identical to the previous 2.0 cm diameter cases except for the wavelength. Where the peak-to-peak distance of the previous simulations was 6.0 cm, it is now 3.0 cm. The straight lines at the far left and right intestinal wall are there to simplify the mesh generation; they have very little effect on the tracked particles, which all inhabit the middle portion of the segment. These still images were taken some time after the start of their respective simulations. In both sets of images, the top image was taken after 0.5 seconds, and the bottom after 7.5 seconds.**

Another simulation we performed varies the frequency of contractions, but all other parameters remain the same. This simulation was run for twice as long, so that it would include the same number of total contractions as the others; two values of distribution are included in Table 1, one which results after eight seconds, the same time as the other contractions; and another after sixteen seconds, so that the same number of contractions are simulated. The results for the lower frequency and shorter wavelength can be seen in Table 1.

## Particle Trajectory Mapping

One of the ways we analyzed the fluid motion in the process of segmentation was to see whether it is possible to construct a particle trajectory map. Such a map would predict the trajectory of a particle which started in a given location, after the fluid motion had reached steady state. The general concept of a map is illustrated in Figure 4.

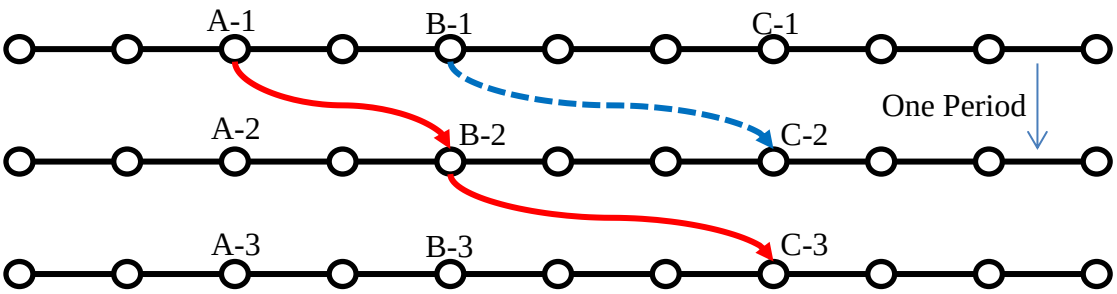


**Figure 4: Schematic of the periodic map. One particle moves from its initial position A-1 to B-2 after one period, then from B-2 to C-3 after another period. If the system in which these particles reside is able to be mapped, then a particle starting from B-1 will move to C-2 after one period.**

We investigated the case of flow induced by segmentation, after two full cycles of contractions, to see if it would be possible to construct a map. Our method was to release particles in known locations after two full cycles of contractions, then to release another set of particles in those same locations after one cycle of contractions, then release one more set of particles after another cycle of contractions. We then analyzed the paths of all of these particles for some time after their release, to see if they followed similar paths. Some results are documented here. In both figures, the mesh used by the respective simulation is shown.

All of the particles from the study presented in Figure 5 had similar results; the spacing between particle end positions is considerable, but it is approximately the same as the mesh cell size. This indicates that a finer mesh is needed to investigate the case.

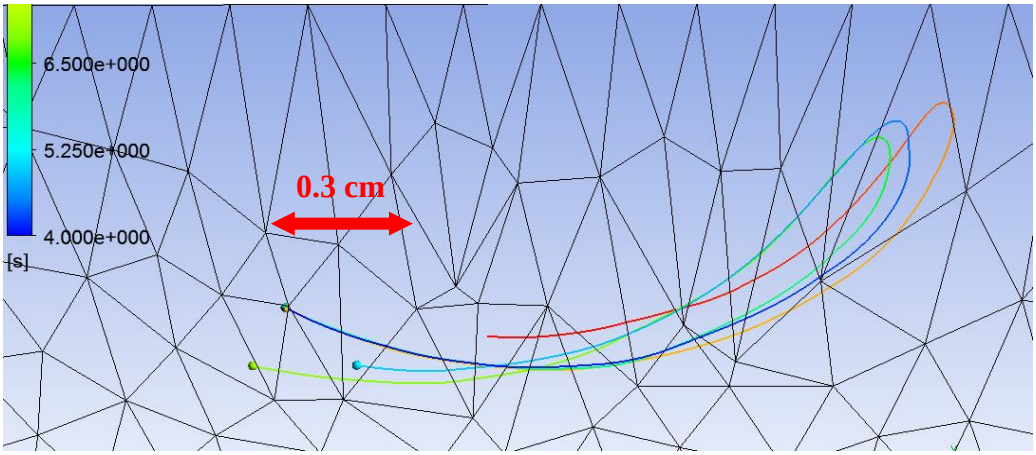


**Figure 5: Trajectory map 1, coarse mesh. The particles all started from the dark blue point, and their paths for one second are shown; the particle with the light blue path was released at 4.0 seconds, the particle with the green path was released at 6.0 seconds, and the particle with the red path was released at 8.0 seconds. The accuracy of these positions is suspect, as the mesh size is large compared to the differences in the end positions.**

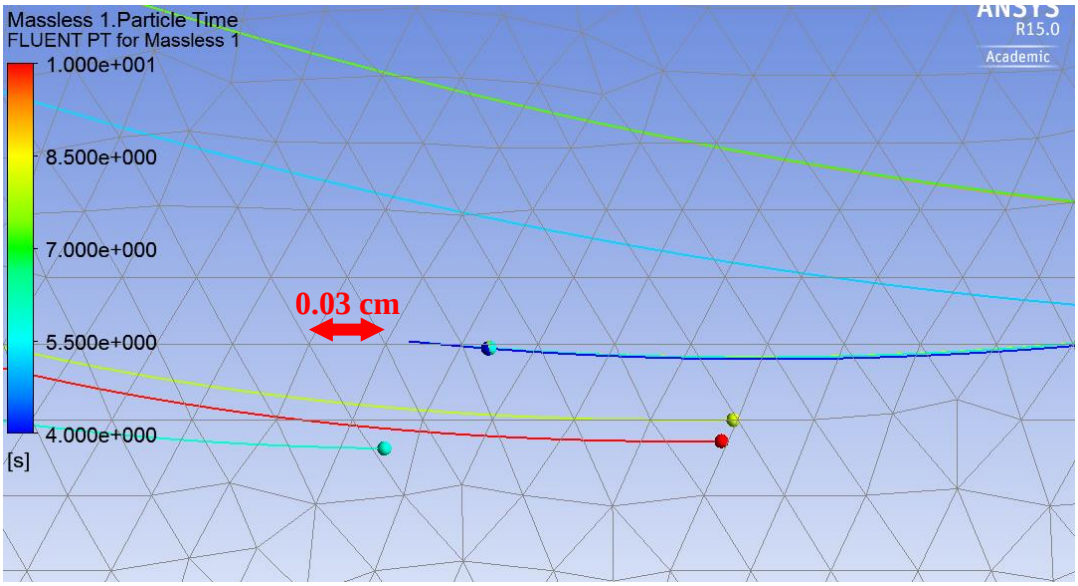


**Figure 6: Trajectory map 2, fine mesh. The particles all started from the dark blue point, and their paths for two seconds are shown; the particle with the light blue path was released at 4.0 seconds, the particle with the yellow path was released at 6.0 seconds, and the particle with the red path was released at 8.0 seconds. The accuracy of these positions is suspect, as the mesh size is large compared to the differences in the end positions.**

The results from the finer mesh in Figure 6 were more conclusive. First of all, the first particle released has several cell-lengths between its end-point and those of the other particles. This indicates that it does truly have a different end-point. This suggests that when this particle is released, there are still some latent transient properties present in the flow. However, the other two particles, both released later, end up very close to each

other; this indicates that the distance between them might be purely due to error. Although this does not conclude that a mapping is possible, it is evidence which, when further pursued, might lead to a periodic map.

## Peristalsis

We performed a set of simulations of peristaltic contractions using the same setup for particle tracking as in the simulations of segmental contractions. The intestinal diameter for this simulation was taken to be 2.0 cm, which is the same as in the standard segmentation contraction simulations. There are many parameters whose exact proportions are not known, so this simulation requires further validation. The simulation is illustrated in Figure 7.

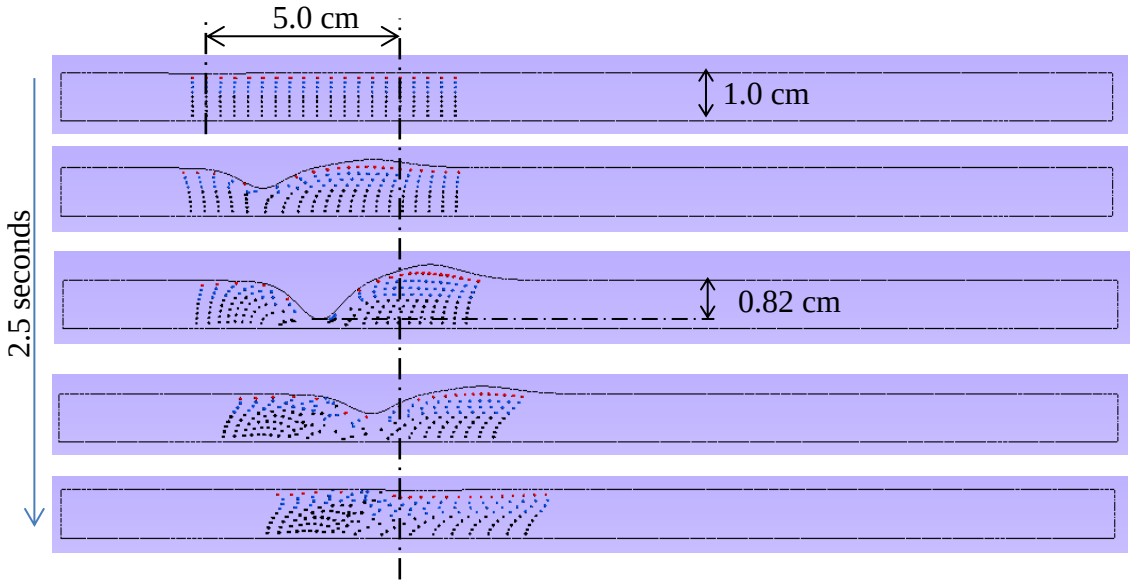


**Figure 7: Peristalsis in the Small Intestine. The first of these images was taken very close to the beginning of the simulation, and each of the following four images was taken about 0.625 seconds after the previous image. The images are arranged to illustrate how a peristaltic wave was simulated. As in Figures 1 and 2, the black particles are the ones which started closest to the axis, blue particles started in the mid region, and red particles started closest to the intestinal wall. The black line at the bottom of each image represents the intestinal axis.**

The simulation shows that there is some mixing in the small intestine after just one peristaltic contraction. It is conceivable that after several peristaltic contractions, the particles will be further mixed. However, the particles are certainly not distributed as well by the peristaltic simulation as by the segmentation simulations; nearly all of the black particles are still in the same region they started in, and the red particles don't seem to have moved at all. These findings are quantified by the same process as the findings from the simulations of segmentation contractions, and are compared with those in Table 1.

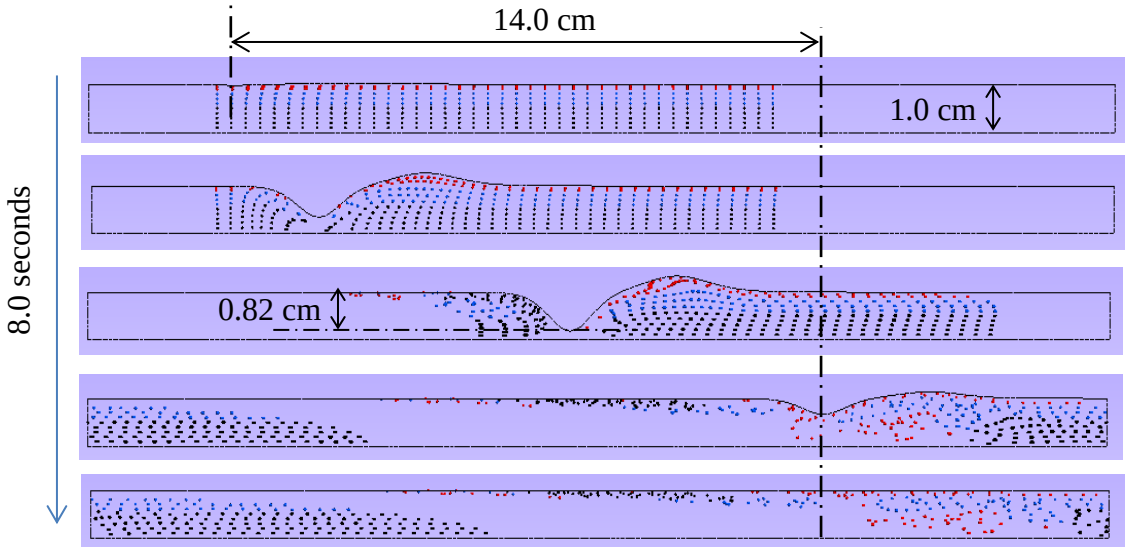


**Figure 8: Long duration peristalsis. The first of these images was taken very close to the beginning of the simulation. The second was taken after 1.0 seconds, the third after 4.0 seconds. The fourth was taken at 7.0 seconds, and the last was taken after 8.0 seconds. The contraction grows slowly, then stays at a constant depth for 6.25 seconds, then relaxes slowly. The black particles are the ones which started closest to the intestinal axis, the blue particles started in between, and the red particles started closest to the intestinal wall. The black line at the bottom of each image represents the intestinal axis. The tracked particles which come from the left in the fourth and fifth images are those which were pushed through the right boundary due to the flow.**

We performed additional, longer simulations of peristaltic contractions, shown in Figure 8. They grow the same way as the previous simulation, stay at the same contraction depth as they propagate for some length of intestine, then relax the same way as the contraction in the previous simulation. This contraction profile is more realistic than the one

simulated in Figure 7. All dimensions and propagation velocity are the same as in the previous case; just the simulation time and the distance traveled are greater.

From the figures, it can be seen that the longer contraction produces more mixing than the shorter one. However, there are more particles tracked in the case of the contraction presented in Figure 8, to ensure that the effects of the propagating contraction were captured by tracked particles. Some of these particles were pushed outside of the right boundary, and were returned to the simulation space due to the periodic conditions applied to the boundaries. These particles were not transversally mixed, but are still included in the calculation of second moment of distribution. The results of this simulation are in Table 1.

## Quantification of Mixing

We used the second moment of distribution to describe how well-mixed the particles in a simulation become after a certain number of contractions. The axisymmetric tube of the small intestine is split into three concentric, equivolumetric sections for the purpose of measuring the distribution. In this document, “R1 particles” are those that are initially positioned in the center section, “R2 particles” are initially positioned in the middle section, and “R3 particles” are those which are initially closest to the intestinal wall. See Figure 9.

It is expected that after undergoing perfect mixing, it does not matter where a group of particles started, any group will end up evenly distributed among the three sections outlined in Figure 9. Such a distribution earns a distribution score of zero, while a group of particles which all end up in the same section earns a score of one. The second

moment of distribution for each set of particles is calculated by first calculating the average particle density among the three sections, which is 0.333 for each set of particles. Then, the square of the difference between the actual particle density in each section and the average are added together, and scaled by a constant factor to ensure a range between zero and one.

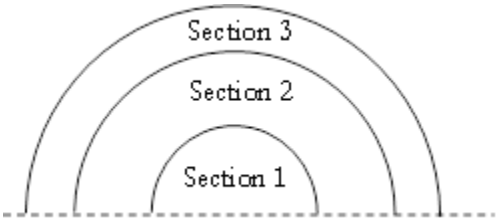


**Figure 9: Sections of the axisymmetric small intestine. R1 particles begin the simulation in section 1, R2 particles in section 2, and R3 particles in section 3. Drawing not to scale; in simulation, the three sections have equal volume.**

## Discussion

The results in Table 1 seem accurate, but they could be improved with more simulations against which to compare results. Based on the distribution scores, it seems as though segmentation produces about the same amount of mixing overall. Peristalsis produces worse mixing than segmentation all things considered, however for each peristaltic simulation only one contraction was simulated. Predictably, there is overall better mixing in the low-frequency case after more contractions.

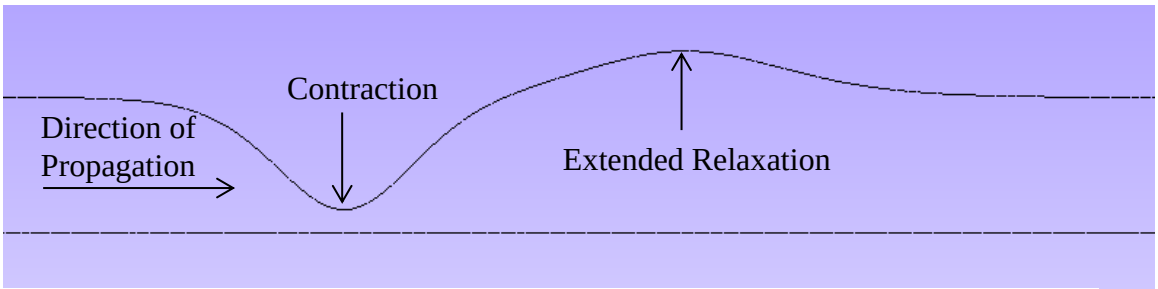


**Figure 10: Components of peristaltic contraction.**

**Table 1: Second moments of distribution of particles during segmentation and peristalsis. "R1" are particles which started closest to the center of the lumen, "R2" are the particles in the middle section, and "R3" are the particles which started closest to the intestinal wall. Tabulated are measurements of how well distributed the particles are. A score of zero means they are perfectly distributed, while a score of one indicates that all the particles have remained together.**

| | | | Distribution of R1: | Distribution of R2: | Distribution of R3: |
|---|---|---|---|---|---|
| Segmentation | | 3.0 cm diameter | 0.18393 | 0.08188 | 0.09083 |
| | | Switch-node: 3.0 cm diameter | 0.16654 | 0.06457 | 0.33250 |
| | | 2.0 cm diameter | 0.11744 | 0.07000 | 0.28583 |
| | | Switch-node: 2.0 cm diameter | 0.11592 | 0.04120 | 0.06750 |
| | | Short Wavelength: 2.0 cm diameter | 0.14592 | 0.03730 | 0.12583 |
| | Low Frequency | 2.0 cm diameter, same time | 0.28868 | 0.08920 | 0.28000 |
| | | 2.0 cm diameter, same number of contractions | 0.10688 | 0.10690 | 0.27083 |
| Peristalsis | | 2.0 cm diameter, short duration | 0.62841 | 0.16083 | 1.00000 |
| | | 2.0 cm diameter, long duration | 0.36876 | 0.50896 | 0.29359 |

For both peristalsis and segmentation contractions, there are some parameters whose real-life values we would like to know. For example, it is unclear how far the intestinal wall contracts towards the center of the intestine in real life. We also do not know how far the peristaltic contraction propagates along the intestine before relaxing back to its initial state. Additionally, there are several unanswered qualitative questions. It is unclear what the profile of the contraction looks like. We do not know if it travels some distance while it is fully contracted or if it contracts then relaxes immediately as in the simulation. The extended relaxation in front of the contraction is also unclear; we do not know how far this relaxation extends beyond the contraction, or how much it adds to the radius of the intestine. For these simulations we assumed that it extends about 3 cm in front of the contraction, and that it adds only about 0.3 cm to the radius.

CHAPTER 3

# FUTURE WORK

The primary goal of this research is to numerically determine the extent to which mixing occurs in the small intestine. At the current time, there is a lot more work to be done towards this goal; there are many improvements to be made to the simulations in order for them to more accurately portray real-life fluid flow in the intestines. Furthermore, many areas of study have not yet been pursued. Further research in this field will bring us closer to an accurate representation of the real-life small and large intestines so that the simulations will grow more relevant.

## Improvement of Simulations

First of all, we should perform more simulations with more parameters varied; all of the current simulations, for example, feature sinusoidal intestinal contractions. While we don't know what shape the intestinal walls have, we should try to determine whether the shape of the contractions has any bearing on the effectiveness of the mixing. Other parameters to vary include length of segmentation contractions, time period of contractions, duration of peristaltic contractions, and many, many more.

Another way to improve the simulations is to track more particles in the simulation. So far there are not more than three hundred particles tracked in a single simulation, which leaves less than one hundred particles for the diametrically smaller sections of the lumen. While this will require more intensive simulations, the simulations will have results based on a larger data pool, less subject to error.

One way to bring the simulations closer to the realistic setup is to implement corrugated intestinal walls in the simulations. In a real small intestine or large intestine, there are folds and layers that make the inside of the intestinal wall ridged and furrowed. All simulations we have performed so far have featured smooth walls. We would like to know if changing the intestinal parameters to include more realistic, corrugated intestinal walls makes a significant difference in the mixing patterns of the small intestine and large intestine. This change would be simple to implement, as we could approximate the effects of random ridges with a sinusoidal wave pattern on the intestinal wall.

Finally, from the point of view of dynamical systems it would be very helpful to construct a map which tells us, after the simulation has reached steady state, where a particle in a certain location will end up after a complete contraction cycle. The proof of concept of this map has been accomplished, albeit imprecisely. The challenges for this goal are mostly computational. To improve the map, an automated method of checking the positions of particles after a certain number of contraction cycles should be developed.

In order to address the issues of unknown parameters, we will pursue new avenues of research in collaboration with Temple University School of Medicine, as personnel there have more direct experience in the field we are investigating. They also are more aware of the current state of research in the field, and such a perspective would certainly help us guide our own research.

## Large Intestine

Muscular contractions in the large intestine, or colon, can also be grouped into segmentation and peristalsis (Emmanuel & Raeburn, 2011). However, the colon diameter is larger than the small intestine, and the digested food takes a much longer time going through the colon than through the small intestine, despite the much longer length of the small intestine.

Additionally, there are more distinct types of peristalsis in the large intestine. The large intestine has two major types of propagating contractions: low-amplitude propagated contractions, and high-amplitude propagated contractions, abbreviated as LAPCs and HAPCs, respectively. Low-amplitude propagated contractions cause less pressure and happen more often – about 100 times in a day. High-amplitude propagated contractions cause more pressure and only occur about 5 times in a day (Emmanuel & Raeburn, 2011).

Although the work we have done so far has been primarily focused on the small intestine, results of the study for the large intestine would be just as relevant to medical purposes. Therefore, our future work will feature a much broader scope to include simulation analysis of large intestine as well as small intestine. The requirements for implementing large intestine simulations to this research are straightforward. First of all, we would have to find more extensive literature on large intestine fluid properties, like density and viscosity. Most of the work is simply in doing more simulations, with large intestinal parameters and types of contractions. Since the user-defined functions for small intestine contractions are already written, most of this work is already done.

# BIBLIOGRAPHY


Emmanuel, A., & Raeburn, A. (2011). Small intestine and colon motility. *Medicine*, 218-223.

Helander, H. F., & Fandricks, L. (2014). Surface area of the digestive tract - revisited. *Scandinavian journal of gastroenterology*, 49.

Husebye, E. (1999). The patterns of small bowel motility: Physiology and implications in organic disease and functional disorders. *Neurogastroenterology and Motility, 11*, 141-161.

Kim, S. K. (1968). Small intestine transit time in the normal small bowel study. *American journal of roentgenology, radium therapy, and nuclear medicine*, 522-524.

Lew, H. S., Fung, Y. C., & Lowenstein, C. B. (1971). Peristaltic carrying and mixing of chyme in the small intestine (An analysis of a mathematical model of peristalsis of the small intestine). *Journal of Biomechanics*, 297-315.

Liu, C. J., Huang, S. C., Huang, Y. C., Liu, C. Y., & Chen, H. I. (2013). Sonographic demonstration of human small intestinal migrating motor complex phase III. *Neurogastroenterology & Motility*, 198-202.

Malcolm, A., & Kellow, J. E. (2000). Small intestine motility. *Current Opinion in Gastroenterology*, 140-146.

Nadeem, S., Ashiq, S., & Ali, M. (2012). Williamson fluid model for the peristaltic flow of chyme in small intestine. *Mathematical Problems in Engineering*, 18 pages.

Roy, R., Rios, F., & Riahi, D. N. (2011). Mathematical models for flow of chyme during gastrointestinal endoscopy. *Applied Mathematics*, 600-607.

Shulze, K. S. (2014). The imaging and modelling of the physical processes involved in digestion and absorption. *Acta Physiologica*, 394-405.

Stevenson, G. W., Collins, S. M., & Somers, S. (1988). Radiological appearance of migrating motor complex of the small intestine. *Gastrointestinal Radiology*, 215-218.

Wikberg, M., Ulmius, J., & Ragnarsson, G. (1997). Review article: targeted drug delivery in treatment of intestinal diseases. *Alimentary Pharmacology & Therapeutics*, 109-115.

APPENDIX A

# USER DEFINED FUNCTIONS

This user defined function was used for the 3.0 cm segmentation case when the nodes and antinodes stayed in the same locations for the entire simulation.

```
#include "udf.h"


DEFINE_GRID_MOTION(sine_wave, domain, dt, time, dtime)
{

        Thread *tf = DT_THREAD (dt);
        face_t f;
        Node *node_p;
        real x, pi, t, tmove;
        int n;

        /* Set/activate the deforming flag on adjacent cell zone, which      */
        /* means that the cells adjacent to the deforming wall will also be  */
        /* deformed, in order to avoid skewness.                             */

        SET_DEFORMING_THREAD_FLAG (THREAD_T0 (tf));

        pi = 3.14159265358979323846264338;
        t = CURRENT_TIME;
        tmove = 0.5;

        begin_f_loop (f, tf)
        {

                f_node_loop (f, tf, n)
                {

                        node_p = F_NODE (f, tf, n);

                        /* Update the current node only if it has not been   */
                        /* previously visited:                               */

                        if (NODE_POS_NEED_UPDATE (node_p))
                        {

                        /* Set flag to indicate that the current node's  */
                        /* position has been updated, so that it will not be  */
```

```
                        /* updated during a future pass through the loop:      */

                        NODE_POS_UPDATED (node_p);

                        x = NODE_X (node_p);
                                if (sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x) > 0)
        {
        NODE_Y (node_p) = 0.015 + 0.00633*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x);
        }
        else
        {
        NODE_Y (node_p) = 0.015 + 0.01*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x);
        }
                        }
                }
        }
end_f_loop (f, tf);
}
```

This user defined function was used for the 3.0 cm segmentation case when the nodes and antinodes switched halfway through the simulation.

```
#include "udf.h"


DEFINE_GRID_MOTION(sine_wave_switchnodes, domain, dt, time, dtime)
{
        Thread *tf = DT_THREAD (dt);
        face_t f;
        Node *node_p;
        real x, xn, pi, t, tmove;
        int n;

        /* Set/activate the deforming flag on adjacent cell zone, which       */
        /* means that the cells adjacent to the deforming wall will also be   */
        /* deformed, in order to avoid skewness.                              */

        SET_DEFORMING_THREAD_FLAG (THREAD_T0 (tf));

        pi = 3.14159265358979323846264338;
        t = CURRENT_TIME;
        tmove = 0.5;

        begin_f_loop (f, tf)
        {
                f_node_loop (f, tf, n)
                {
                        node_p = F_NODE (f, tf, n);

                        /* Update the current node only if it has not been    */
                        /* previously visited:                                */

                        if (NODE_POS_NEED_UPDATE (node_p))
                        {
                        /* Set flag to indicate that the current node's  */
                        /* position has been updated, so that it will not be  */
                        /* updated during a future pass through the loop:     */

                        NODE_POS_UPDATED (node_p);

                        x = NODE_X (node_p);
                                if (t < 4)
                                {
                                        if (sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x) > 0)
{
NODE_Y (node_p) = 0.015 + 0.00633*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x);
```

```
        }
        else
        {
        NODE_Y (node_p) = 0.015 + 0.01*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x);
        }
                                }
                                else
                                {
                                xn = x+0.015;
                                        if (x < 0.015)
                                        {
                                        NODE_Y (node_p) = 0.015;
                                        }
                                        else
                                        {
                                                if (x > 0.24-0.045)
                                                {
                                                NODE_Y (node_p) = 0.015;
                                                }
                                                else
                                                {
                        if (sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*xn) > 0)
                        {
        NODE_Y (node_p) = 0.015 + 0.00633*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*xn);
                        }
                        else
                        {
        NODE_Y (node_p) = 0.015 + 0.01*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*xn);
                        }
                                                }
                                        }
                                }
                        }
                }
        }
end_f_loop (f, tf);
}
```

This user defined function was used for the 2.0 cm segmentation case when the nodes and antinodes stayed in the same locations for the entire simulation.

```
#include "udf.h"


DEFINE_GRID_MOTION(sine_wave, domain, dt, time, dtime)
{

        Thread *tf = DT_THREAD (dt);
        face_t f;
        Node *node_p;
        real x, pi, t, tmove;
        int n;

        /* Set/activate the deforming flag on adjacent cell zone, which      */
        /* means that the cells adjacent to the deforming wall will also be  */
        /* deformed, in order to avoid skewness.                             */

        SET_DEFORMING_THREAD_FLAG (THREAD_T0 (tf));

        pi = 3.14159265358979323846264338;
        t = CURRENT_TIME;
        tmove = 0.5;

        begin_f_loop (f, tf)
        {

                f_node_loop (f, tf, n)
                {

                        node_p = F_NODE (f, tf, n);

                        /* Update the current node only if it has not been   */
                        /* previously visited:                               */

                        if (NODE_POS_NEED_UPDATE (node_p))
                        {

                        /* Set flag to indicate that the current node's  */
                        /* position has been updated, so that it will not be  */
                        /* updated during a future pass through the loop:     */

                        NODE_POS_UPDATED (node_p);

                        x = NODE_X (node_p);
                                if (sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x) > 0)
```

```
{
NODE_Y (node_p) = 0.01 + 0.00667*0.633*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x);
}
else
{
NODE_Y (node_p) = 0.01 + 0.00667*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x);
}
                                }
                        }
                }
end_f_loop (f, tf);
}
```

This user defined function was used for the 2.0 cm segmentation case when the nodes and antinodes switched halfway through the simulation.

```
#include "udf.h"

DEFINE_GRID_MOTION(sine_small_switchnodes, domain, dt, time, dtime)
{
        Thread *tf = DT_THREAD (dt);
        face_t f;
        Node *node_p;
        real x, xn, pi, t, tmove;
        int n;

        /* Set/activate the deforming flag on adjacent cell zone, which      */
        /* means that the cells adjacent to the deforming wall will also be  */
        /* deformed, in order to avoid skewness.                             */

        SET_DEFORMING_THREAD_FLAG (THREAD_T0 (tf));

        pi = 3.14159265358979323846264338;
        t = CURRENT_TIME;
        tmove = 0.5;

        begin_f_loop (f, tf)
        {
                f_node_loop (f, tf, n)
                {
                        node_p = F_NODE (f, tf, n);

                        /* Update the current node only if it has not been   */
                        /* previously visited:                               */

                        if (NODE_POS_NEED_UPDATE (node_p))
                        {
                        /* Set flag to indicate that the current node's  */
                        /* position has been updated, so that it will not be */
                        /* updated during a future pass through the loop:    */

                        NODE_POS_UPDATED (node_p);

                        x = NODE_X (node_p);
                                if (t < 4)
                                {
                                        if (sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x) > 0)
                                        {
NODE_Y (node_p) = 0.01 + 0.667*0.00633*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x);
```

```
                                        }
                                        else
                                        {
NODE_Y (node_p) = 0.01 + 0.667*0.01*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*x);
                                        }
                                }
                                else
                                {
                                xn = x+0.015;
                                        if (x < 0.015)
                                        {
                                        NODE_Y (node_p) = 0.010;
                                        }
                                        else
                                        {
                                                if (x > 0.24-0.045)
                                                {
                                                NODE_Y (node_p) = 0.010;
                                                }
                                                else
                                                {
                                        if (sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*xn) > 0)
                                                        {
NODE_Y (node_p) = 0.01 + 0.667*0.00633*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*xn);
                                                        }
                                                        else
                                                        {
NODE_Y (node_p) = 0.01 + 0.667*0.01*sin(0.5*2*pi*t)*sin((1/0.06)*2*pi*xn);
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }
end_f_loop (f, tf);
}
```

This user defined function was used for the short-duration peristalsis simulation shown.

```
#include "udf.h"


DEFINE_GRID_MOTION(peristalsis_smalldia_short, domain, dt, time, dtime)
{

        Thread *tf = DT_THREAD (dt);
        face_t f;
        Node *node_p;
        real x, pi, t, t0, t1;
        int n;

        /* Set/activate the deforming flag on adjacent cell zone, which      */
        /* means that the cells adjacent to the deforming wall will also be  */
        /* deformed, in order to avoid skewness.                             */

        SET_DEFORMING_THREAD_FLAG (THREAD_T0 (tf));

        pi = 3.14159265358979323846264338;
        t = CURRENT_TIME;
        t0 = 1.25;
        t1 = 2.5;
        begin_f_loop (f, tf)
        {

                f_node_loop (f, tf, n)
                {

                        node_p = F_NODE (f, tf, n);

                        /* Update the current node only if it has not been   */
                        /* previously visited:                               */

                        if (NODE_POS_NEED_UPDATE (node_p))
                        {

                        /* Set flag to indicate that the current node's  */
                        /* position has been updated, so that it will not be */
                        /* updated during a future pass through the loop:    */

                        NODE_POS_UPDATED (node_p);

                        x = NODE_X (node_p);
```

```
                                    if (t<t1)
                                    {
                                        if (t<t0)
                                        {
                                    NODE_Y (node_p) = 0.01 - 0.667*t/t0*0.0125*1/
                                                ((cosh(100*((x-0.2/6)-0.02*t)))*
                                                 (cosh(100*((x-0.2/6)-0.02*t)))*
                                                 (cosh(100*((x-0.2/6)-0.02*t)))*
                                                 (cosh(100*((x-0.2/6)-0.02*t))))
                                                                + 0.667*0.0125*.41*t/t0*1/
                                                ((cosh(79*((x-0.35/6)-0.02*t)))*
                                                 (cosh(79*((x-0.35/6)-0.02*t)))*
                                                (cosh(79*((x-0.35/6)-0.02*t))));
                                        }
                                        else
                                        {
                                    NODE_Y (node_p) = 0.01-0.667*(t1-t)/t0*0.0125*1/
                                                ((cosh(100*((x-0.2/6)-0.02*t)))*
                                                (cosh(100*((x-0.2/6)-0.02*t)))*
                                                (cosh(100*((x-0.2/6)-0.02*t)))*
                                                (cosh(100*((x-0.2/6)-0.02*t))))
                                                        + 0.667*0.0125*.41*(t1-t)/t0*1/
                                                ((cosh(79*((x-0.35/6)-0.02*t)))*
                                                (cosh(79*((x-0.35/6)-0.02*t)))*
                                                (cosh(79*((x-0.35/6)-0.02*t))));
                                        }
                                    }
                                    else
                                    {
                                    NODE_Y (node_p) = 0.01;
                                    }
                                }
                            }
                        }
end_f_loop (f, tf);
}
```

This user defined function was used for the short wavelength case of segementation.

```
#include "udf.h"


DEFINE_GRID_MOTION(sine_wave_swl, domain, dt, time, dtime)
{

        Thread *tf = DT_THREAD (dt);
        face_t f;
        Node *node_p;
        real x, pi, t, tmove;
        int n;

        /* Set/activate the deforming flag on adjacent cell zone, which      */
        /* means that the cells adjacent to the deforming wall will also be  */
        /* deformed, in order to avoid skewness.                             */

        SET_DEFORMING_THREAD_FLAG (THREAD_T0 (tf));

        pi = 3.14159265358979323846264 34;
        t = CURRENT_TIME;
        tmove = 0.5;

        begin_f_loop (f, tf)
        {
                f_node_loop (f, tf, n)
                {
                        node_p = F_NODE (f, tf, n);

                        /* Update the current node only if it has not been   */
                        /* previously visited:                               */

                        if (NODE_POS_NEED_UPDATE (node_p))
                        {

                        /* Set flag to indicate that the current node's  */
                        /* position has been updated, so that it will not be */
                        /* updated during a future pass through the loop:    */

                        NODE_POS_UPDATED (node_p);

                        x = NODE_X (node_p);
                                if (x < 0.015)
                                {
                                NODE_Y (node_p) = 0.01;
                                }
```

```
                                else
                                {
                                        if (x > 0.24-0.015)
                                        {
                                        NODE_Y (node_p) = 0.01;
                                        }
                                        else
                                        {
                                                if   (sin(0.5*2*pi*t)*sin((1/0.03)*2*pi*x)   >
0)
                                                {
NODE_Y (node_p) = 0.01 + 0.00667*0.633*sin(0.5*2*pi*t)*sin((1/0.03)*2*pi*x);
                                                }
                                                else
                                                {
NODE_Y (node_p) = 0.01 + 0.00667*sin(0.5*2*pi*t)*sin((1/0.03)*2*pi*x);
                                                }
                                        }
                                }
                        }
                }
        }
end_f_loop (f, tf);
}
```

This user defined function was used for the lower frequency case of segmentation

```
#include "udf.h"


DEFINE_GRID_MOTION(sine_wave_lowfq, domain, dt, time, dtime)
{

        Thread *tf = DT_THREAD (dt);
        face_t f;
        Node *node_p;
        real x, pi, t, tmove;
        int n;

        /* Set/activate the deforming flag on adjacent cell zone, which      */
        /* means that the cells adjacent to the deforming wall will also be  */
        /* deformed, in order to avoid skewness.                             */

        SET_DEFORMING_THREAD_FLAG (THREAD_T0 (tf));

        pi = 3.14159265358979323846264 34;
        t = CURRENT_TIME;
        tmove = 0.5;

        begin_f_loop (f, tf)
        {
                f_node_loop (f, tf, n)
                {
                        node_p = F_NODE (f, tf, n);

                        /* Update the current node only if it has not been   */
                        /* previously visited:                               */

                        if (NODE_POS_NEED_UPDATE (node_p))
                        {

                        /* Set flag to indicate that the current node's  */
                        /* position has been updated, so that it will not be */
                        /* updated during a future pass through the loop:    */

                        NODE_POS_UPDATED (node_p);

                        x = NODE_X (node_p);
                                if (sin(0.25*2*pi*t)*sin((1/0.06)*2*pi*x) > 0)
                                {
NODE_Y (node_p) = 0.01 + 0.00667*0.633*sin(0.25*2*pi*t)*sin((1/0.06)*2*pi*x);
                                }
```

```
                                    else
                                    {
NODE_Y (node_p) = 0.01 + 0.00667*sin(0.25*2*pi*t)*sin((1/0.06)*2*pi*x);
                                    }
                                }
                            }
                        }
end_f_loop (f, tf);
}
```

This user defined function was used for the long-duration peristalsis

```
#include "udf.h"


DEFINE_GRID_MOTION(peristalsis_long_smalldia, domain, dt, time, dtime)
{
        Thread *tf = DT_THREAD (dt);
        face_t f;
        Node *node_p;
        real x, pi, t, t0, t1, t2;
        int n;

        /* Set/activate the deforming flag on adjacent cell zone, which      */
        /* means that the cells adjacent to the deforming wall will also be  */
        /* deformed, in order to avoid skewness.                             */

        SET_DEFORMING_THREAD_FLAG (THREAD_T0 (tf));

        pi = 3.14159265358979323846264338;
        t = CURRENT_TIME;
        t0 = 1.25;
        t1 = 6.25;
        t2 = 7.5;
        begin_f_loop (f, tf)
        {
                f_node_loop (f, tf, n)
                {
                        node_p = F_NODE (f, tf, n);

                        /* Update the current node only if it has not been   */
                        /* previously visited:                               */
                        if (NODE_POS_NEED_UPDATE (node_p))
                        {

                        /* Set flag to indicate that the current node's  */
                        /* position has been updated, so that it will not be */
                        /* updated during a future pass through the loop:    */

                        NODE_POS_UPDATED (node_p);

                        x = NODE_X (node_p);
                                if (t<t2)
                                {
                                        if (t<t1)
                                        {
                                                if (t<t0)
```

```
                                        {
                                    NODE_Y (node_p) = 0.01 - 0.667*t/t0*0.0125*1/
                                                          ((cosh(100*((x-0.2/6)-0.02*t)))*
                                                           (cosh(100*((x-0.2/6)-0.02*t)))*
                                                           (cosh(100*((x-0.2/6)-0.02*t)))*
                                                           (cosh(100*((x-0.2/6)-0.02*t))))
                                                       + 0.667*0.0125*.41*t/t0*1/
                                                          ((cosh(79*((x-0.35/6)-0.02*t)))*
                                                           (cosh(79*((x-0.35/6)-0.02*t)))*
                                                           (cosh(79*((x-0.35/6)-0.02*t))));
                                        }
                                        else
                                        {
                                    NODE_Y (node_p) = 0.01 - 0.667*1*0.0125*1/
                                                          ((cosh(100*((x-0.2/6)-0.02*t)))*
                                                           (cosh(100*((x-0.2/6)-0.02*t)))*
                                                           (cosh(100*((x-0.2/6)-0.02*t)))*
                                                           (cosh(100*((x-0.2/6)-0.02*t))))
                                                          + 0.667*0.0125*.41*1*1/
                                                          ((cosh(79*((x-0.35/6)-0.02*t)))*
                                                           (cosh(79*((x-0.35/6)-0.02*t)))*
                                                           (cosh(79*((x-0.35/6)-0.02*t))));
                                        }
                                    }
                                    else
                                    {
                                NODE_Y (node_p) = 0.01 - 0.667*(t2-t)/t0*0.0125*1/
                                                          ((cosh(100*((x-0.2/6)-0.02*t)))*
                                                           (cosh(100*((x-0.2/6)-0.02*t)))*
                                                           (cosh(100*((x-0.2/6)-0.02*t)))*
                                                           (cosh(100*((x-0.2/6)-0.02*t))))
                                                          + 0.667*0.0125*.41*(t2-t)/t0*1/
                                                          ((cosh(79*((x-0.35/6)-0.02*t)))*
                                                           (cosh(79*((x-0.35/6)-0.02*t)))*
                                                           (cosh(79*((x-0.35/6)-0.02*t))));
                                    }
                                }
                                else
                                {
                                NODE_Y (node_p) = 0.01;
                                }
                            }
                        }
                    }
end_f_loop (f, tf);
}
```