# Model Intestinal Microflora In Computer Simulation (MIMICS) Technical Report:

## MIMICS Cellular Automaton Program Design and Performance Testing

Michael H.F. Wilkinson
Centre for High Performance Computing
University of Groningen
Landleven 1, 9747 AD Groningen, The Netherlands
Phone: +31-50-363-3374
Fax: +31-50-363-3406
E-mail: michael@rc.service.rug.nl

## MIMICS Technical Reports

The MIMICS project of the Centre for High Performance Computing of the University of Groningen is a project initiated by the International Study Group for New Antimicrobial Strategies (ISGNAS). Its aim is to explore computer simulation methods for the study of the intestinal microflora and its interactions with the host. MIMICS technical reports are intended to explain various technical issues involved in this modelling. As such, the main readership are persons involved in the MIMICS project, other ISGNAS projects, and those intending to implement similar models. Parts of the contents may be reproduced in articles at a later date.

# TABLE OF CONTENTS

**1. INTRODUCTION** **1**

**2. THE CONCEPTUAL MODEL** **1**

**2.1 Modelling Bacterial Metabolisms** **2**

**2.2 Spatial and temporal discretization** **8**

**2.3 Transport Equations** **9**

**2.4 Boundary conditions** **11**

**2.5 Variable geometry extensions** **13**

**3. PROGRAM DESIGN** **16**

**3.1 Overall program structure** **16**

**3.2 Constants, variables and data structures needed** **17**

**3.3 Subroutine EXPINI** **21**

**3.4 Subroutines METAB and MDIFEQ** **22**

**3.5 Subroutine LFLOW** **24**

**3.6 Subroutines DIFFUS and BOUNDS** **24**

**3.7 Subroutine REPPROG** **27**

**3.8 Subroutines STDATA and FINREP** **27**

**4. THE TESTING STAGES** **27**

**4.1 Laminar flow** **27**

**4.2 Diffusion** **29**

**4.3 The metabolism** **29**

**4.4 Integration of all components.** **33**

**5. DISCUSSION** **34**

**6. REFERENCES AND FURTHER READING** **35**

## 1. Introduction

Model Intestinal Microflora In Computer Simulation (MIMICS) is a project aimed at developing computer simulation tools for modelling the intestinal microflora and its interactions with the host. The International Study  Group for New Antimicrobial Strategies (ISGNAS) started the project to try to keep advancement of theory in line with the huge increase of the resolving power of observational tools available to microbiologists. As a first start of the project a pilot study was performed to explore aerobe-anaerobe interactions in the gut from a theoretical point of view, and to see whether computer simulation could actually mimic reality in this field, where observations are relatively plentiful. The results of this study have been presented elsewhere (Wilkinson, 1997).

This paper describes the development of the main tool: a large scale cellular automaton which can simulate both metabolic and transport processes in the human intestine. The program runs on the Cray J932 supercomputer of the Centre for High Performance Computing. The paper is divided into three main sections after the introduction. The first develops the conceptual model of the intestine in mathematical terms, the second describes the program design, and the third the testing phases.

## 2. The Conceptual Model

The modelling of the intestinal microflora and its environment can be broken up into 5 somewhat interrelated parts:

1. the bacterial metabolisms
2. the chemistry of the environment (food, oxygen supply, etc.)
3. the geometry of the environment
4. the mechanics of transport
5. the interaction with the immune system

The bacterial metabolisms must include food and oxygen uptake, and may include production of metabolites, toxins, etc., and damage by toxic substances (e.g. oxygen) either by inhibition of the metabolism, or direct damage leading to cell lysis. No metabolite or toxin production is modelled explicitly in this pilot study. All other effects will be.

The chemistry of the environment will be kept simple. Bacteria only see two substances: food and oxygen. Any competition is indirect, through depletion of the food supply. The only expected symbiosis is the depletion of oxygen by the aerobes to produce an anoxic environment for anaerobes.

For computing time reasons, the model intestine's geometry will be an axially symmetric, unflexible straight tube of varying diameter. This can be conveniently modelled in a rectangular array. The model will be constructed in such a way as to allow flexibility of the tube at a later stage, for modelling of peristalsis.

The transport equations will be kept to diffusion and laminar flow of an incompressible fluid. This should work well enough, except for the final part of the large intestine, where water is extracted from the lumen. Evidently, the rectum expands and contracts too dramatically to be modelled by a tube of fixed dimensions. Given the limited amount of time for this pilot study, it would seem best simply to model the intestine, up to, but not including the rectum.

At this stage, the immune system will be left out of the model completely. This may appear to be extremely arbitrary, as the immune system must interact with the intestinal flora. Nonetheless there are two good reasons for eliminating the immune system from the model: (i) the majority of bacteria in a healthy intestine do not seem to evoke an immune response, thus modelling them without an interaction may well be realistic, and (ii) the aim of this model is to see whether the distribution of aerobic and anaerobic bacteria in the intestine *can be* the result of the symbiotic relationship described above; the immune system should not influence this.

It is likely that this model will apply quite well to the lumen of the intestine, and less to the mucosa, with its complicated surface structure. Later, more complicated models of mucosa, immune system, bacteriocin production, etc., could be added to the simpler "luminal" model developed here.

## 2.1 Modelling Bacterial Metabolisms

Bacterial metabolism come in six more-or-less distinct types. Four grow in completely oxygen free conditions:

| | |
|---|---|
| - strict anaerobe | Even very low concentrations of oxygen kill them |
| - moderate anaerobe | Can survive low concentrations of oxygen |
| - tolerant anaerobe | Could not give a damn about oxygen |
| - facultative (an)aerobe | Grow better with oxygen, but fairly well in absence of $O_2$ |

Two other types require oxygen for survival

|  |  |
|---|---|
| - microaerophile | needs low concentrations of $O_2$ to survive; perishes at moderate concentrations. |
| - strict aerobe | requires oxygen to grow; no toxic effects at normal levels |

Ideally, we would like to model the growth of each type of bacterial metabolism with equations of the same general form. To achieve this, let us model growth as dominated by four basic processes:

1. Food uptake by aerobic metabolism
2. Food uptake by anaerobic metabolism
3. Destruction of cells by oxygen (or other toxins)
4. Cell maintenance

At least one of the first two processes must take place, since the bacteria must acquire food in some way. The third process (destruction) may or may not take place. The final process is always present, unless the cells may enter into a dormant state (e.g. spore formation). Spore formation will not be included in the model.
Food uptake by any metabolism may be modelled by Michaelis-Menten type kinetics [e.g. Gerritse *et al.* 1990]:

$$\mu = \mu_{max} \frac{[F]}{K_F + [F]} \tag{2.1.1}$$

In which $\mu$ is the growth rate per unit of bacterial biomass [$s^{-1}$], $\mu_{max}$ is the maximum growth rate (of order $2 \times 10^{-4}$/s=0.7/h), $[F]$ the concentration of food substrate [kg/l], and $K_F$ the half saturation food concentration [kg/l]. In reality, multiple substrates and multiple $K_F$ values may apply to the metabolism of a single bacterium, but at this stage only a single food pool and a single value for $K_F$ will be used in the model.
A purely anaerobic metabolism is modelled by letting $\mu_{max}$ be dependent of oxygen concentration through Michaelis-Menten kinetics:

$$\mu_{max} = \mu_{O_2} \frac{[O_2]}{K_{R,O_2} + [O_2]} \qquad (2.1.2)$$

with $\mu_{O_2}$ maximum growth rate by aerobic metabolism, $[O_2]$ the oxygen concentration [mol/l] and $K_{R,O_2}$ the half saturation respiration rate oxygen concentration [mol/l]. Gerritse *et al.* [1990] use two different half saturation concentrations and two different $\mu_{O_2}$ to model the presence of a high and a low affinity oxygen uptake system in a single bacterium. Later models may include such a refinement.

A purely anaerobic metabolism, including inhibition of the metabolism by oxygen can be modelled by:

$$\mu_{max} = \mu_{an} - \mu_{O_2} \frac{[O_2]}{K_{R,O_2} + [O_2]} \qquad (2.1.3)$$

where $\mu_{an}$ is the maximum growth rate by anaerobic metabolism. Again, Gerritse *et al.* suggest the use of a low and a high affinity oxygen uptake system. Instead of this, we include a model for toxicity of oxygen independent of food uptake, again via Michaelis-Menten kinetics: bacteria are destroyed by the presence of oxygen (irreversible as opposed to reversible damage). The basal metabolic requirement per cell for maintenance purposes, which may often be omitted from models of chemostats with high dilution rates, is modelled as a constant.

Putting all these effects together, we have:

$$\mu = \left\{ \mu_{O_2} \frac{[O_2]}{K_{R,O_2} + [O_2]} + \mu_{an} \right\} \frac{[F]}{K_F + [F]} - \kappa_{O_2} \frac{[O_2]}{K_{T,O_2} + [O_2]} - \mu_{basal} \qquad (2.1.4)$$

in which:

| | |
|---|---|
| $\mu_{basal}$ | basal metabolism (minimum metabolic requirement) |
| $\kappa_{O_2}$ | maximum oxygen kill rate |
| $K_{T,O_2}$ | half saturation kill rate oxygen concentration |

Strict aerobes will have a zero maximum anaerobic metabolism, and zero maximum oxygen kill rate. Conversely, strict anaerobes will have a positive $\mu_{an}$, and negative $\mu_{O_2}$ and/or positive $\kappa_{O_2}$. Tolerant anaerobes will show positive $\mu_{an}$, and zero $\mu_{O_2}$ and $\kappa_{O_2}$. Facultatives have positive $\mu_{an}$, and $\mu_{O_2}$ combined with zero $\kappa_{O_2}$, and, finally, microaerophiles have zero $\mu_{an}$, and positive $\mu_{O_2}$ and $\kappa_{O_2}$.

In an ecosystem with $P$ species of bacteria, the differential equation for the concentration of the $k^{th}$ species of bacteria $[X_k]$ associated with growth rate $\mu_k$ is simply:

$$[\dot{X}_k] = \mu_k [X_k] \tag{2.1.5}$$

The equation must be combined with equations for food and oxygen use:

$$[\dot{F}] = \sum_{k=1}^{P} \left\{ \alpha_{\kappa,k} \kappa_{O_2,k} \frac{[O_2]}{K_{T,O_2,k} + [O_2]} - \left\{ \frac{\mu_{O_2,k}}{\alpha_{O_2,k}} \frac{[O_2]}{K_{R,O_2,k} + [O_2]} + \frac{\mu_{an,k}}{\alpha_{an,k}} \right\} \frac{[F]}{K_{F,k} + [F]} \right\} [X_k] \tag{2.1.6}$$

with:

| | |
|---|---|
| $\alpha_{O_2,k}$ | efficiency factor of aerobic metabolism $(0 < \alpha_{O_2,k} \leq 1)$ |
| $\alpha_{an,k}$ | efficiency factor of anaerobic metabolism $(0 < \alpha_{an,k} \leq 1)$ |
| $\alpha_{\kappa,k}$ | fraction of oxygen killed bacteria returned as food $(0 \leq \alpha_{\kappa,k} \leq 1)$ |

Thus the change in food concentration has two components: (i) a negative component in the uptake by bacteria, and (ii) a positive component associated with the kill rate. If we define the amount of biomass as the sum of food and bacteria concentrations, the system loses biomass, due to (i) imperfect efficiencies of the metabolisms, (ii) the basal metabolism of bacteria, and (iii) the fraction of killed bacteria which does not return to the food pool. If a finite amount of food is added to a closed system, the number of bacteria is automatically limited by that supply. The amount of food will not go negative, since the only negative component in the food consumption rate becomes zero when the food supply reaches zero.

The oxygen usage is:

$$[\dot{O_2}] = -\sum_{k=1}^{P} \left\{ \beta_{\mu,k} \frac{[O_2]}{K_{R,O_2,k} + [O_2]} \frac{[F]}{K_{F,k} + [F]} + \beta_{\kappa,k} \frac{[O_2]}{K_{T,O_2,k} + [O_2]} \right\} [X_k] \tag{2.1.7}$$

with:

| | |
|---|---|
| $\beta_{\mu,k}$ | maximum oxygen uptake rate due to aerobic metabolism |
| $\beta_{\kappa,k}$ | maximum oxygen uptake rate due to toxic effect on anaerobes or microaerophiles |

Thus the metabolism of each bacterial species may be characterized by 12 parameters, which are listed in Table 1.

### *2.1.1 A note on steric hindrance.*

At high concentrations of bacteria, the presence of other bacteria in the vicinity of each bacterium may produce a form of steric hindrance of the uptake of food and oxygen. The uptake of any substance (food or toxin) by a bacterium is proportional to its freely accessible surface area. Any part of the surface touching (or close too) other bacteria will not be freely accessible. For each bacterium, the probability that another bacterium is within a certain range of its surface area is proportional to the volume density of all species present. To a good approximation, the uptake rate of a substance *S* will be proportional to:

$$[\dot{S}] \propto 1 - \frac{\sum_{k=1}^{P} [X_k]}{X_{max}} \tag{2.1.8}$$

in which $X_{max}$ is the maximum possible density all bacteria put together may reach, i.e. when they are packed so tighly no free surface area is left to them. To correct for this, a term similar to a logistic term could be incorporated into the differential equations. Simply multiplying the righthand sides of (2.1.6) and (2.1.7) with the righthand side of (2.1.8) is sufficient for food and oxygen uptake. Equation (2.1.4) must be modified as follows:

**Table 1. Parameters describing a bacterial metabolism**

| Symbol | Meaning | Typical values | Units |
|---|---|---|---|
| $\mu_{O_2}$ | maximum growth rate by aerobic metabolism | 0.5-4.0 * $10^{-4}$ | /s |
| $\mu_{an}$ | maximum growth rate by anaerobic metabolism | 0.1-1.0 * $10^{-4}$ | /s |
| $\mu_{basal}$ | basal metabolism (minimum metabolic requirement) | ? | /s |
| $K_F$ | half saturation uptake rate food concentration | 0.1 | μmol/l |
| $K_{R,O_2}$ | half saturation respiration rate oxygen concentration | 0.5-400 | μmol/l |
| $K_{T,O_2}$ | half saturation kill rate oxygen concentration | 10 | μmol/l |
| $\kappa_{O_2}$ | maximum oxygen kill rate | ? | /s |
| $\alpha_{O_2}$ | efficiency factor of aerobic metabolism | 0-1 | |
| $\alpha_{an}$ | efficiency factor of anaerobic metabolism | 0-1 | |
| $\alpha_\kappa$ | fraction of oxygen killed bacteria returned as food | 0-1 | |
| $\beta_\mu$ | maximum oxygen uptake rate due to aerobic metabolism | ? | /s |
| $\beta_\kappa$ | maximum oxygen uptake rate due to toxic effect on anaerobes or microaerophiles | ? | /s |

$$\mu = \frac{X_{max} - \sum_{k=1}^{P}[X_k]}{X_{max}} \left( \left\{ \mu_{O_2} \frac{[O_2]}{K_{R,O_2} + [O_2]} + \mu_{an} \right\} \frac{[F]}{K_F + [F]} - \kappa_{O_2} \frac{[O_2]}{K_{T,O_2} + [O_2]} \right) - \mu_{basal} \qquad (2.1.9)$$

Note that the basal metabolism does not scale with the righthand side of (2.1.8). Equation (2.1.5) remains unchanged. Only if very high concentrations of bacteria are to be modelled will the corrections discussed here be used. Later versions of the model may include these modifications.

## 2.2 Spatial and temporal discretization

In reality the intestine has a highly complicated geometry, showing many twists and turns, and a highly rugged inner surface with detailed corrugations from the centimetre downto the sub-micron scale. The strategic model which will be developed here must necessarily omit such detail. Whatever the complexity of the geometry, the topology is relatively simple: that of a single, unbranching tube. In the initial modelling a simple axisymetric geometry will be imposed on the model, chiefly for ease of computation. It will be discussed how detail may be added at a later stage. Because of this, it is likely that the model will perform rather better for the luminal than the mucosal flora. As a physical scale, we will assume the length of the intestine to be 10 m, and the diameter some 5 cm (fixed at first, may become variable later). The time scales of interest are in the order of days for the flora as a whole, and in the order of 20 minutes for the bacteria (fastest doubling time).

The model intestine is assumed to be a cylindrical tube of length $L$ and radius $R$, which is subdivided into $M$ sections of length $l$ in the axial direction and $N$ equivolume concentric cylinders in the radial direction. Indices $i \in \{1, 2, \dots, N\}$ and $j \in \{1, 2, \dots, M\}$ are used to denote volume elements in radial and axial directions respectively. The volume of each element is of course:

$$V_{i,j} = \frac{\pi R^2 l}{N} \tag{2.2.1}$$

The $i^{th}$ radial division thus has an outer diameter of:

$$R_i = \sqrt{\frac{i}{N}} R \qquad i \in \{1, 2, \dots, N\} \tag{2.2.2}$$

Allowing the radius to vary over the length of the tube should be explored at a later stage in the model (section 2.5).

The time steps will all be held equal. Three time scales are of interest in determining the time steps: (i) the metabolic time scale (tens of minutes), (ii) the diffusion time scale (??), and (iii) the bulk transport time scale (tens of minutes). The shortest time scale determines the step taken, since our model will compute the three processes consecutively, rather than simultaneously. It may even be sensible to use extremely short time steps, so the differential equations may be replaced by difference equations for simplicity.

## 2.3 Transport Equations

### *2.3.1 Diffusion*

Transport through diffusion applies to all volume elements. The process must be slow enough (or the time step must be short enough) to ensure that only diffusion between immediate neighbours need be considered. If this is the case, the diffusion of a substance *s* between two volume elements α and β is simply:

$$[\dot{s}]_\alpha = \frac{\delta_s}{V_\alpha} A_{\alpha\beta} \frac{[s]_\beta - [s]_\alpha}{\Delta z} \tag{2.3.1}$$

with:

| | |
|---|---|
| $[s]_\alpha$ | concentration of *x* in volume element α |
| $[s]_\beta$ | concentration of *x* in volume element β |
| $\delta_s$ | diffusion coefficient of *x* |
| $A_{\alpha\beta}$ | surface area of contact between volume elements α and β |
| $V_\alpha$ | volume of compartment α |
| $\Delta z$ | distance between centres of gravity of volume elements |

The change in concentration is thus proportional to an estimate of the concentration gradient of *s*. In the axial direction the distance between centroids is simply the element size *l*. In the axial direction it may be approximated as:

$$\Delta r = \frac{V_{i,j}}{A_{i,j,rad}} = \frac{R}{2\sqrt{N}\sqrt{i}} \tag{2.3.2}$$

In which $A_{i,j,rad}$ is the surface area of contact between elements ($i,j$) and ($i$+1,$j$). The diffusion into the $i^{th}$ compartment in the radial direction and the $j^{th}$ compartment in the axial direction is then:

$$\begin{aligned}[\dot{s}]_{i,j} &= \frac{\delta_s}{l^2}\left([s]_{i,j-1}+[s]_{i,j+1}-2[s]_{i,j}\right)\\ &\quad+\frac{4N\delta_s}{R^2}\left\{(i-1)\left([s]_{i-1,j}-[s]_{i,j}\right)+i\left([s]_{i+1,j}-[s]_{i,j}\right)\right\}\end{aligned} \tag{2.3.3}$$

for all $i \in \{1,2,\ldots,N\}$ and $j \in \{1,2,\ldots,M\}$. Therefore the model intestine will be represented as a 2-D array, with indices running from 0 to $N$+1 and 0 to $M$+1 in radial and axial directions respectively. The elements at $j$=0 and $j$=$M$+1 contain the boundary conditions at the stomach and anus respectively. The elements at $i$=$N$+1 hold the boundary conditions at the intestinal epithelium, and the elements at $i$=0 are padding for the diffusion equation, to prevent bounds check errors. Note that this latter boundary condition is automatically satisfied: diffusion from segment (0,$j$) into (1,$j$) is zero, because the contact surface area is 0. The other boundary conditions require more thought. These will be dealt with in section 2.4.

### *2.3.2 Laminar flow*

The next stage concerns laminar flow through the tube. In laminar flow the flow velocity as a function of radial position $v(r)$ is:

$$v(r) = v_{max}\left(1-\frac{r^2}{R^2}\right) \quad \Leftrightarrow \quad v(i) = v_{max}\left(1-\frac{i-1}{N-1}\right) \tag{2.3.4}$$

to a good approximation. It may be advisable to let the $N^{th}$ radial compartment have a residual velocity:

$$v(i) = \left(v_{max}-v_{min}\right)\left(1-\frac{i-1}{N-1}\right)+v_{min} \tag{2.3.5}$$

If we assume that no mixing takes place during flow, i.e. the diffusion time is long compared to flow time scale, a simple difference equation may be used to simmulate laminar flow:

$$[s]_{i,j,t+1} = [s]_{i,j-U_i,t} + \left( U_i - \frac{v(i)\Delta t}{l} \right) \left\{ [s]_{i,j-U_i-1,t} - [s]_{i,j-U_i,t} \right\} \tag{2.3.6}$$

with:

$$U_i = \mathrm{TRUNC}\left( \frac{v(i)\Delta t}{l} \right) \tag{2.3.7a}$$

If the time step is short enough to have $U_i$=0, we can simplify (2.3.6) to:

$$[s]_{i,j,t+1} = [s]_{i,j,t} + \frac{v(i)\Delta t}{l} \left\{ [s]_{i,j-1,t} - [s]_{i,j,t} \right\} \tag{2.3.8}$$

This has the distinct advantage that we do not have to introduce more boundary conditions than for diffusion (no negative values of *j* are required).

## 2.4 Boundary conditions

One of the trickier parts of the model design is formulating adequate boundary conditions. Let us first consider the boundary at *i*=*N*+1. Each of the substances in the model (food, oxygen and different types of bacteria) have a subtly different behaviour at the epithelium. Oxygen is simplest: it can diffuse through the epithelium, and is always present in the epithelium, roughly at the level of oxygenated blood or tissue. Hence there will be a constant supply of oxygen from the intestinal wall. Food behaves in a similar, but slightly more complex manner. If the concentration of food in the intestine, uptake takes place, removing food from the intestine, on the other hand, mucus containing nutrients is produced at an approximately constant rate. If the uptake is not passive diffusion, but has Michaelis-Menten type kinetics, we have a flow between element (*N*,*j*) and (*N*+1,*j*) of:

$$[\dot{F}]_{N,j} = \frac{2N}{R} \left( \mu_{muc} - \lambda_F \frac{[F]_{N,j}}{K_{uptake} + [F]_{N,j}} \right) \tag{2.4.1}$$

with:

| | |
|---|---|
| $\mu_{muc}$ | rate of mucus production per unit of surface area |
| $K_{uptake}$ | half saturation food uptake concentration |
| $\lambda_F$ | maximum active food uptake rate |

We can equate (2.4.1) to the term in (2.3.2) concerning diffusion between (*N*,*j*) and (*N*+1,*j*), if we set the boundary condition:

$$[F]_{N+1,j} = [F]_{N,j} + \frac{\mu_{muc}}{\delta_F} - \frac{\lambda_F}{\delta_F}\frac{[F]_{N,j}}{K_{uptake} + [F]_{N,j}} \tag{2.4.2}$$

in which $\delta_F$ is the diffusion rate constant for food. Thus, before each diffusion cycle, the elements at *i*=*N*+1 must be updated according to 2.4.2.

For bacteria we choose to ignore translocation in this model, and prohibit them from diffusing through the epithelium. This can be achieved by simply setting:

$$[X_k]_{N+1,j} = [X_k]_{N,j} \quad \text{for all } k \tag{2.4.3}$$

The boundary conditions at *j*=0 are set somewhat differently. For all time steps the influx of food, oxygen, and bacteria are set by user specified functions or arrays, and at each time step these values are used for all elements at *j*=0. For *j*=*M*+1 the simplest plausible boundary condition is probably hermetic sealing (for diffusion):

$$\begin{aligned}
&[F]_{i,M+1} = [F]_{i,M} \\
&[O_2]_{i,M+1} = [O_2]_{i,M} \\
&[X_k]_{i,M+1} = [X_k]_{i,M} \qquad \text{for all } k
\end{aligned} \tag{2.4.4}$$

If desired some influx of oxygen in the same way as through the intestinal epithelium may be introduced.

## 2.5 Variable geometry extensions

An important extension to the model is the inclusion of variable geomtry in the form of a varying radius of the tube, i.e. replacing $R$ by $R_j$, or even by $R_j(t)$. This extra degree of freedom in the model allows increased realism, e.g. letting the large intestine be larger in diameter than the small intestine, and even modelling of true peristalsis. In the current project we will allow the diameter to vary in the axial dimension $z$ but not in time. A full modelling of peristalsis is beyond the scope of this paper. The program structure should allow the inclusion of time dependence, so efforts by others in the area of peristalsis modelling may be added later.

Let us assume that the we have an array of $R_j$ for $j$=0,1,...$M$, and let each $R_j$ pertain to the right edge of each axial section (see Fig. 2.5.1). Time dependence may be introduced at any time by altering the contents of the array at every time step. The basic spatial discretization still holds, but the volumes and surface areas of each element become a function of $j$. The volume of element ($i$,$j$) becomes:

$$V_{i,j} = \frac{\pi l\left(R_{j-1}^2 + R_{j-1}R_j + R_j^2\right)}{3N} \tag{2.5.1}$$

which is still independent of $i$. The radial surface area of contact between element ($i$,$j$) and ($i$+1,$j$) becomes:

$$A_{i,j,rad} = \pi l\left(R_{j-1} + R_j\right)\sqrt{\frac{i}{N}} \tag{2.5.2}$$

which depends on both $i$ and $j$. The axial surface area of contact between element ($i$,$j$) and ($i$,$j$+1) is:

$$A_{i,j,axial} = \frac{\pi R_j^2}{N} \tag{2.5.3}$$

independent of $i$. If we do not change the geometry in time, it is probably best to precompute the values of the contact surface areas and volumes of the elements, and store them in arrays.

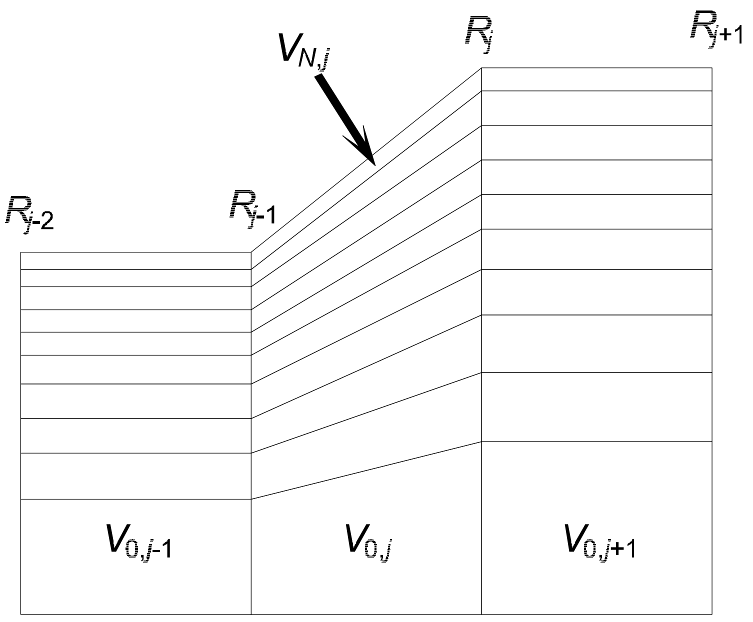


**Figure 2.5.1.** A simple variable geometry: each axial section *j* has a trapezoidal shape determined by $R_{j\text{-}1}$ and $R_j$.

Changing the geometry does not change the metabolic rates of the bacteria in any way; it is the transport equations which change. By applying equations 2.5.1 through 2.5.3 to the general form for diffusion, we find:

$$\begin{aligned}\dot{[s]}_{i,j} &= \frac{\delta_s}{lV_{i,j}}\left\{A_{i,j,axial}\left([s]_{i,j+1}-[s]_{i,j}\right)+A_{i,j-1,axial}\left([s]_{i,j-1}-[s]_{i,j}\right)\right\}\\ &+\frac{\delta_s}{V_{i,j}^2}\left\{A_{i,j,rad}^2\left([s]_{i+1,j}-[s]_{i,j}\right)+A_{i-1,j,rad}^2\left([s]_{i-1,j}-[s]_{i,j}\right)\right\}\end{aligned} \tag{2.5.4}$$

or:

$$\begin{aligned}\dot{[s]}_{i,j} &= \frac{\delta_s}{l^2R_V^2}\left\{R_j^2\left([s]_{i,j+1}-[s]_{i,j}\right)+R_{j-1}^2\left([s]_{i,j-1}-[s]_{i,j}\right)\right\}\\ &+\frac{4NR_S^2\delta_s}{R_V^4}\left\{(i-1)\left([s]_{i-1,j}-[s]_{i,j}\right)+i\left([s]_{i+1,j}-[s]_{i,j}\right)\right\}\end{aligned} \tag{2.5.5}$$

with:

$$R_V^2=\frac{R_{j-1}^2+R_{j-1}R_j+R_j^2}{3} \qquad \text{and} \qquad R_S=\frac{R_{j-1}+R_j}{2}$$

The problem of laminar flow also requires modifications. Assume that the fluid flowing throught the tube is incompressible, and each element's volume does not change in time. This means that the net volume must flow into each element is zero. Equation 2.3.6 for laminar flow needs first to be modified by using volume flow and element volume, instead of flow velocity and linear length:

$$[\dot{s}]_{i,j} = \frac{I(i)}{V_{i,j}}\left\{[s]_{i,j-1,t} - [s]_{i,j,t}\right\} \tag{2.5.6}$$

with $I(i)$ the volume flow rate:

$$I(i) \equiv A_{i,j,axial} \cdot v_j(i) \tag{2.5.6a}$$

Obviously, under laminar flow conditions this volume flow rate is a function of $i$ in the same way as the velocity $v(i)$:

$$I(i) = \left(I_{max} - I_{min}\right)\left(1 - \frac{i-1}{N-1}\right) + I_{min} \tag{2.5.7}$$

If we need to use flow volumes larger than the volume elements themselves, (2.5.6) is inadequate. Quite generally the change in the **amount** of a substance $s$ in a volume element is:

$$V_{i,j}\Delta[s] = \iiint\limits_{V_{in,i,j}=I_{in,i,j}\Delta t} [s]dV - \iiint\limits_{V_{out,i,j}=I_{out,i,j}\Delta t} [s]dV \tag{2.5.8}$$

The integrals of the concentration of $s$ over the inflowing volume must be approximated by sums:

$$S_{in,i,j} = \iiint\limits_{V_{in,i,j}=I_{in,i,j}\Delta t} [s]dV \approx \sum_{h=jstart}^{j-1} V_{i,h}[s]_{i,h} - \left\{\sum_{h=jstart}^{j-1} V_{i,h} - I_{in,i,j}\Delta t\right\}[s]_{i,jstart} \tag{2.5.9}$$

with *jstart* chosen so that:

$$\sum_{h=jstart}^{j-1} V_{i,h} \geq I_{in,i,j}\Delta t \qquad \text{and} \qquad \sum_{h=jstart+1}^{j-1} V_{i,h} < I_{in,i,j}\Delta t \tag{2.5.9a}$$

The outflow $S_{out,i,j}$ is simply the same as the inflow $S_{in,i,j+1}$ of element (*i*,*j*+1).
The boundary conditions need no modifications, since the diffusion equations (which they pertain to) have already been altered. Consider the most complicated boundary condition, that of food. Using the altered geometry, the flow becomes:

$$[\dot{F}]_{N,j} = \frac{2NR_S}{R_V^2}\left(\mu_{muc} - \lambda_F \frac{[F]_{N,j}}{K_{uptake} + [F]_{N,j}}\right) \tag{2.5.10}$$

It can easily be seen that equation 2.5.8 relates to 2.5.5 in exactly the same way as equation 2.4.1 relates to 2.3.2, and therefore we get the same boundary condition.

## 3. Program design

### 3.1 Overall program structure

The program structure will be very basic. After initialization and input of experimental conditions (food and oxygen supply, bacterial metabolisms, etc.), the progam will loop for a specified number of time steps through five subroutines which simulate (i) bacterial metabolism, (ii) laminar flow, (iii) diffusion, and (iv) store data and (v) report progress. subroutines (i) through (iv) themselves loop through all elements of the *i* and *j* indices of the grid representing the intestine. After the loop ends, a final report is given and all files are closed. Thus, the program can be built from the following subunits:

| | |
|---|---|
| 1. Data structure declaration initialization | |
| 2. Subroutine EXPINI | Read experimental conditions from file |
| 3. Subroutine METAB | Compute metabolism |
| 4. Subroutine DIFFUS | Diffusion on results of metabolism |
| 5. Subroutine LFLOW | Laminar flow through tube |
| 6. Subroutine STDATA | Store data |

|  |  |  |
|---|---|---|
| 7. | Subroutine REPROG | Report progress |
| 8. | Subroutine FINREP | Final report |

The program will be written in FORTRAN 77 with extensions for parallel processing, Hence the short names and the use of the term subroutine. For the sake of clarity, all implicit typing will be disabled.

The main structure of the program is (in C-like pseudo-code):

```
main()
{  Data structure declaration and initialization;
   EXPINI();
   for {t=0; t<=tmax; t++}
      {  METAB();
         LFLOW();
         DIFFUS();
         STDATA();
         REPROG();
      }
   FINREP();
}
```

The program will be developed and tested in three stages:

|  |  |  |
|---|---|---|
| 1. | LFLOW be | only laminar flow: METAB and DIFFUS will dummy routines |
| 2. | LFLOW+DIFFUS | diffusion added, METAB still dummy |
| 3. | LFLOW+DIFFUS+METAB | full program active |

In the first stage all data initialization, experiment setup, data storage and reporting routines must be in place (they may be of a somewhat limited form at first). Runs of all three stages will be presented as video animation for demonstration purposes.

## 3.2 Constants, variables and data structures needed

The central data structure will be a 3-D array (X) of floating point values with indices I, J and K for the radial, axial, and "chemical" (bacteria and food) dimensions respectively. Valid indices run from 0..N+1, 0..M+1 and 1..P+2 for I, J and K respectively (P=number of bacterial species). The array is indexed as X(K,I,J), which ensures that the chemistry of each volume element is stored in adjacent array elements, which facilitates passing these data to the routine which solves the differential equations.

We will introduce a convention that values of K running from 1 to P inclusive index speices of bacteria, KFOOD=P+1 pertains to food, and KO2=P+2 pertains to oxygen.

Besides the main data structure (the array X) a number of other arrays and variables and constants (parameters in Fortran parlance) are necessary. To begin with, following constants will be used:

| | | |
|---|---|---|
| 1. | NMAX=20 | Maximum number of divisions in radial direction |
| 2. | MMAX=1000 | Maximum number of divisions in radial direction |
| 3. | PMAX =6 | Maximum number of species in population |
| 4. | ITMAX=10000 | Maximum number of time steps to iterate |

These define the static sizes of the arrays used, to allow passing of data in common blocks without to much trouble. Dynamically, smaller values define the actual bounds of the array used in the program. These are integer values stored in the variables:

| | | |
|---|---|---|
| 1. | N | Number of divisions in radial direction |
| 2. | M | Number of divisions in radial direction |
| 3. | P | Number of species in population |
| 4. | ITNUM | Number of time steps to iterate |

Arrays of these lengths will always be indiced by variables I, J, K and ITER respectively. KFOOD and KO2 are initialized to the values defined above.

Besides these integers, floating point variables must store the physical length of the intestine, the time and duration of a time step, and the tolerance of the computations:

| | | |
|---|---|---|
| 1. | T | Physical time (s) |
| 2. | TSTEP | Physical duration of time step (s) |
| 3. | TOL | Tolerance |
| 4. | LENGTH | Total length of intestine (m) |

The other variables in the program can be divided into three categories: (i) environmental variables defining the shape and wall structure of the intestine, (ii) metabolic variables, defining the different species of bacteria, and (iii) transport variables, relating to diffusion and laminar flow. These will be described in the following subsections. For all arrays the *valid* index range is given. To find the maximum (static) index range substitute N, M, P and ITNUM for NMAX, MMAX, PMAX and ITMAX respectively.

### *3.2.1 Environmental variables*

The environment of the intestine is defined by its wall properties:

| | | | |
|---|---|---|---|
| 1. | R(0:M) | $=R_j$ | local radius of intestine (m) |
| 2. | MUMUC(1:M) | $=\mu_{muc}$ | mucus production rate |
| 3. | KUPTK(1:M) | $=K_{uptake}$ | half saturation food uptake concentration |
| 4. | LAMDAF(1:M) | $=\lambda_F$ | maximum active food uptake rate |
| 5. | O2WALL(1:M) | $=[O_2]_{N+1,j}$ | oxygen concentration in wall |
| 6. | V(1:M) | $=V_{i,j}$ | volume of element (I,J) (independent of I) |
| 7. | AAXIAL(0:M) | $=\mathrm{A}_{i,j,axial}$ | axial contact area between elements (I,J) and (I,J+1) (independent of I) |
| 8. | ARAD(0:N,1:M) | $=\mathrm{A}_{i,j,rad}$ | radial contact area between elements (I,J) and (I+1,J) |

Strictly speaking the O2WALL need not be defined separately, as it is stored in X(KO2,N+1,J) anyway. For clarity it may however be useful. The contents of these arrays are read once at initialization and never changed. The last three variables are precomputed from R(1:M) in the current implementation. In a later version incorporating peristalsis the arrays will be filled by the flow subroutine to be used by the diffusion subroutine.

### *3.2.2 Metabolic variables*

The metabolisms of bacteria is defined by the twelve parameters in Table 1. Accordingly there are twelve arrays of 1..P to store these data for each of the P species in the model:

| | | |
|---|---|---|
| 1. | MUO2(1:P) | maximum growth rate by aerobic metabolism |
| 2. | MUAN(1:P) | maximum growth rate by anaerobic metabolism |
| 3. | MBASAL(1:P) | basal metabolism (minimum metabolic requirement) |

| | |
|---|---|
| 4. KF(1:P) | half saturation uptake rate food concentration |
| 5. KRO2(1:P) | half saturation respiration rate oxygen concentration |
| 6. KTO2(1:P) | half saturation kill rate oxygen concentration |
| 7. KILLO2(1:P) | maximum oxygen kill rate |
| 8. AO2(1:P) | food uptake by aerobic metabolism$\left(=\mu_{O_2,k}/\alpha_{O_2,k}\right)$ |
| 9. AAN(1:P) | food optuke by anaerobic metabolism$\left(=\mu_{an,k}/\alpha_{an,k}\right)$ |
| 10. AKAPPA(1:P) | food production rate by dying cells $\left(=\alpha_{\kappa,k}\kappa_{O_2,k}\right)$ |
| 11. BETAMU(1:P) | maximum oxygen uptake rate due to aerobic metabolism |
| 12. BETAKL(1:P) | maximum oxygen uptake rate due to toxic effect on anaerobes and microaerophiles |

Parameters 8, 9 and 10 are slightly modified forms of those in Table 1, to improve numerical efficicency (see section 3.4). As with the environmental variables, metabolic variables are read from file at the start of the program and remain unchanged.

### *3.2.3 Transport variables*

These may be split into diffusion and laminar flow variables. The parameters needed to compute the diffusion (once boundary conditions have been set) are just the diffusion rate constants:

| | |
|---|---|
| 1. DELTA(1:P+2) | diffusion rates $\delta$ |

DELTA(KFOOD) pertains to food; DELTA(KO2) to oxygen; the rest to species of bacteria. Laminar flow needs:

| | | |
|---|---|---|
| 2. IMIN(1:ITNUM) | $I_{min}$ | minimum flow rate ($m^3/s$) |
| 3. IMAX(1:ITNUM) | $I_{max}$ | maximum flow rate ($m^3/s$) |
| 4. FLCONC(1:ITNUM,1:P+2) | concentration of bacteria, food and oxygen at J=0 for each time step | |

Again, the data are read at initialization, and not changed afterwards.

## 3.3 Subroutine EXPINI

Subroutine EXPINI reads a set of data files and initializes all variables listed in the previous section. The data may be separated into four data files:

1. Discretization and environment data
2. Metabolic variables
3. Transport variables
4. Initial conditions of the intestine

The files are ASCII files which are read in the order shown.

The first file contains N, M, P and ITNUM, setting up the array sizes, and T, TSTEP, LENGTH and TOL, setting up the physical scales and precision. All these are on a single line. This is followed by a line of the form:

```
0   R(0)
```

followed by M lines of data of the form:

```
J   R(J)   MUMUC(J)   KUPTK(J)   LAMDAF(J)   O2WALL(J)
```

After reading these data EXPINI initializes all other environmental data. Strictly speaking the J in each data line is superfluous (and is ignored by the program!), but very useful for editing purposes. The data should be read in free format, to reduce the possibility of errors.

The second data file contains the metabolic data for each bacterium, in the order listed in Table 1. Each collumn in the file pertains to a bacterium, each row to a parameter.

The first lines of the transport variables file contain the DELTA values for bacteria, food and oxygen: a total of P+2 lines of the form:

```
K     DELTA(K)
```

Again, the K is superfluous but handy. Again it is ignored by the program. The file also contains ITNUM lines of the form:

```
ITER  IMIN(ITER)  IMAX(ITER)  FLCONC(ITER,1)  ...  FLCONC(ITER,P+2)
```

Finally the initialization file contains M*N lines of the form:

```
I   J   X(1,I,J)   X(2,I,J)   ...   X(P+2,I,J)
```

Initializing all the values of array X within the intestine. Boundary conditions are set by each call to DIFFUS. Note again the I and J are read but ignored by the program. The order **must** be:

```
1   1   X(1,1,1)   X(2,1,1)   ...   X(P+2,1,1)
2   1   X(1,2,1)   X(2,2,1)   ...   X(P+2,2,1)
.   .      .          .       ...        .
.   .      .          .       ...        .
.   .      .          .       ...        .
N   1   X(1,N,1)   X(2,N,1)   ...   X(P+2,N,1)
1   2   X(1,1,2)   X(2,1,2)   ...   X(P+2,1,2)
.   .      .          .       ...        .
.   .      .          .       ...        .
.   .      .          .       ...        .
.   .      .          .       ...        .
N   M   X(1,N,M)   X(2,N,M)   ...   X(P+2,N,M)
```

EXPINI also initializes the output file to which STDATA writes. It opens it and writes a header, containing such information as N, M, P, ITNUM, and TSTEP.

### 3.4 Subroutines METAB and MDIFEQ

This subroutine accepts, TOL, TSTEP and T as input, and loops through the I and J indices from 1 to N inclusive and from 1 to M inclusive respectively. Within each loop the "chemistry" and population of every grid point is passed to a subroutine which can solve the initial value problem posed by the differential equations in section 2.1, using the current contents as initial values, T as start time and TSTEP as interval length.

At this point in time it is proposed to use the D02BAE (on Cray, D02BAF elsewhere) subroutine from the NAG Fortran Library Mark 14. This subroutine in turn needs a subroutine which specifies the differential equation itself as one of its parameters. This subroutine (MDIFEQ) must have the form:

```
SUBROUTINE MDIFEQ (T,Y,F)
REAL T, Y(*),F(*)
```

In which T is the time, Y is an array of input values and F an array containing the derivatives with respect to T (valid entries from 1 to P+2 inclusive). All MDIFEQ needs to do is to compute the time derivatives of all concentrations from the input concentrations and the metabolisms specified by the metabolic variables (in a common block).

The efficiency of the MDIFEQ can be improved over direct implementation of equations (2.1.4) to (2.1.7), by introducing three temporary variables:

$$R_{an,k} = \frac{[F]}{K_{F,k} + [F]}[X_k] \tag{3.3.1a}$$

$$R_{O_2,k} = \frac{[O_2]}{K_{R,O_2,k} + [O_2]} R_{an,k} \tag{3.3.1b}$$

$$R_{T,k} = \frac{[O_2]}{K_{T,O_2,k} + [O_2]}[X_k] \tag{3.3.1c}$$

When using these interim values, the derivatives reduce to:

$$[\dot{X}_k] = \mu_{O_2,k} R_{O_2,k} + \mu_{an,k} R_{an,k} - \kappa_{O_2,k} R_{T,k} - \mu_{basal,k}[X_k] \tag{3.3.2a}$$

$$[\dot{F}] = \sum_{k=1}^{P} \left\{ A_{T,k} R_{T,k} - A_{O_2,k} R_{O_2,k} - A_{an,k} R_{an,k} \right\} \tag{3.3.2b}$$

$$[\dot{O}] = -\sum_{k=1}^{P} \left\{ \beta_{\kappa,k} R_{T,k} + \beta_{\mu,k} R_{O_2,k} \right\} \tag{3.3.2c}$$

with:

$$A_{T,k} \equiv \alpha_{\kappa,k} \kappa_{O_2,k}, \quad A_{O_2,k} \equiv \frac{\mu_{O_2,k}}{\alpha_{O_2,k}} \quad \text{and} \quad A_{an,k} \equiv \frac{\mu_{an,k}}{\alpha_{an,k}}$$

Thus we precompute the ratios of the metabolic rates ($\mu$) and efficiencies ($\alpha$) and the products of the kill rates ($\kappa$) and the fractions returned to the food pool. This saves in the order of 3*N*M*P*ITNUM computations. Per call to MDIFEQ, direct implementations requires 14P multiplies, 11P divides, 31P array references and 17P adds/subtracts. Using the set of equations above this is reduced to 12P multiplies, 3P divides, 17P array references and 11P adds/subtracts.

In later versions a correction for steric hindrance, simply by multiplying all the righthand sides in (3.3.1a,b,c) by the righthand side of (2.1.8).

## 3.5 Subroutine LFLOW

Computes the laminar flow through the intestine using equations (2.5.6) and (2.5.7). The differential equation is may be replaced by a difference equation using TSTEP as time step. In pseudo-code:

```
LFLOW (ITER,TSTEP)
{  for (I=1;I<=N;I++)
      {   Initialize X(K,I,0) to FLCONC(ITER,K) for all K
          Compute volume flow speed at I
          set V(I,0) to flow volume
          for (J=M;J>0;J--)
             {  Vol=V(I,J)
               for (K=1;K<=P+2;K++)
                 compute new value for X(K,I,J)
             }
      }
}
```

Note the order of the scan through J! Since the new value of X(K,I,J) depends solely on itself and X(K,I,J-1) this approach ensures that unchanged values of concentrations are used. When flow may also be backwards, other measures will become necessary.

The current version uses the volumes precomputed by EXPINI, later versions will compute the volumes and surface areas from the peristalsis.

## 3.6 Subroutines DIFFUS and BOUNDS

Computes diffusion, after first setting the boundary conditions set out in section 2.4 by a call to BOUNDS. BOUNDS copies the contents of X(K,N,J) to X(K,N+1,J) for all bacteria (K<KFOOD), and of X(K,I,M) to X(K,I,M+1) for all K. Besides it computes the values of

all X(KFOOD,N+1,J) from MUMUC(J), KUPTK(J), LAMDA(J) and DELTA(KFOOD). The values of X(KO2,N+1,J) are set by EXPINI.

Once the boundary conditions are set up, the diffusion equation (2.5.4) is used (in the form of a difference equation) to compute new values for all X(K,I,J). For efficiency reasons, the form of (2.5.4) is altered somewhat:

$$\begin{aligned}\Delta\left[X_{k,i,j}\right] = {} & w_{j-1}\left[X_{k,i,j-1}\right] + w_{j+1}\left[X_{k,i,j+1}\right] \\ & + w_{i-1}\left[X_{k,i-1,j}\right] + w_{i+1}\left[X_{k,i+1,j}\right] - w_0\left[X_{k,i,j}\right]\end{aligned} \tag{3.6.1}$$

in which:

$$w_{j-1,k} = \frac{A_{i,j-1,axial}}{lV_{i,j}}\delta_k\Delta t \ , \quad w_{j+1,k} = \frac{A_{i,j,axial}}{lV_{i,j}}\delta_k\Delta t \ ,$$

$$w_{i-1,k} = \left(\frac{A_{i-1,j,rad}}{V_{i,j}}\right)^2\delta_k\Delta t \ , \quad w_{i+1,k} = \left(\frac{A_{i,j,rad}}{V_{i,j}}\right)^2\delta_k\Delta t$$

and

$$w_{0,k} = w_{j-1,k} + w_{j+1,k} + w_{i-1.k} + w_{i+1.k}$$

Direct implementation of the *w*-values as described above is basically an Euler method solution to the full differential equation. Though easy to implement, this has the distinct disadvantage that the "solution" found this way may start wild oscillations and result in negative values of the concentrations, especially at slightly longer time steps. A better method is to divide each volume element into four subdivisions, each bordering on a single subdivision of one of its neighbours. Then use the analytical solution of the differential equation (2.3.2):

$$[s]_\alpha(t+\Delta t) = [s]_\alpha(t) + V_\beta\frac{1-e^{-\frac{(V_\alpha+V_\beta)\delta_s A_{\alpha\beta}}{V_\alpha V_\beta \Delta z[s]_\alpha(\infty)}\Delta t}}{V_\alpha+V_\beta}\left\{[s]_\beta(t)-[s]_\alpha(t)\right\} \tag{3.6.2}$$

The mean concentration is just the mean of the solutions in each of four subdivisions. Since each of the subdivisions has just one quarter of the volume of the entire element, the correct value of *w* become:

$$w_{j-1,k} = V_{i,j-1}\frac{1-e^{-\frac{4\left(V_{i,j}+V_{i,j-1}\right)\delta_k A_{i,j-1,axial}}{lV_{i,j}V_{i,j-1}}\Delta t}}{4\left(V_{i,j}+V_{i,j-1}\right)}$$

$$w_{j+1,k} = V_{i,j+1}\frac{1-e^{-\frac{4\left(V_{i,j}+V_{i,j+1}\right)\delta_k A_{i,j,axial}}{lV_{i,j}V_{i,j+1}}\Delta t}}{4\left(V_{i,j}+V_{i,j+1}\right)}$$

$$w_{i-1,k} = \frac{1-e^{-8\left(\frac{A_{i-1,j,rad}}{V_{i,j}}\right)^2\delta_k\Delta t}}{8}\quad,\qquad w_{i+1,k} = \frac{1-e^{-8\left(\frac{A_{i,j,rad}}{V_{i,j}}\right)^2\delta_k\Delta t}}{8}$$

and $w_{0,k} = w_{j-1,k} + w_{j+1,k} + w_{i-1,k} + w_{i+1,k}$

The routine loops through J from 1 to M. Within the subroutine, a temporary array (XTEMP) containing the *new* values of elements X(K,I,**J-1**) for K=1,2,...,P+2 and I=1,2,...N is used to ensure the implementation of (3.6.1) uses the *old* values of X(K,I,J-1) during computation of ΔX at (K,I,J). Each time a ΔX is computed, X(K,I,J-1) is update, and XTEMP(K,I) is assigned the sum of X(K,I,J) and ΔX. In pseudo-code:

```
DIFFUS()
{  BOUNDS()
   initialize XTEMP(K,I) for all K and I
   for (J=1;J<=M;J++)
      {  for (I=1;I<=N;I++)
            { for (K=1;K<=P+2;K++)
                 {  Compute w0, w(i-1),w(j-1) etc.
                    compute DX
                    X(K,I,J-1)=XTEMP(K,I)
                    XTEMP(K,I)=X(K,I,J)+DX;
                 }
            }
      }
for (I=1;I<=N;I++)
   for (K=1;K<=P+2;K++)
      X(K,I,M)=XTEMP(K,I)
}
```

(Note the final loop to fill the elements X(K,I,M)!)

### 3.7 Subroutine REPPROG

Reports the progress of the program. At its minimum it should report ITER and TIME at each time step. The current settings of IMIN and IMAX, input concentrations and e.g. concentrations at output (J=M) could be given. Later versions might include statistics of the system (e.g. total biomass contained in bacteria). The reports are sent to the default output.

### 3.8 Subroutines STDATA and FINREP

STDATA stores the elements in array X for I=1,2,...,N, J=1,2,...,M and K=1,2,...,P, along with ITER, TIME, IMIN and IMAX. At each call, a header containing the latter data is written. After this, N*M lines of the form:

```
I   J   X(1,I,J)   X(2,I,J)   ...   X(P+2,I,J)
```

are written. As the last output of bith REPPROG and STDATA are available, FINREP just produces a single file (lastout.dat), containing just the last N*M lines of the output, which contain the final state of the model. This allows easy use of the final output of a run as initial condition for the next run.  FINREP closes all output files. Later versions of FINREP may produce more detailed reports.

# 4. The Testing Stages

### 4.1 Laminar flow

The program was run with diffusion and metabolism commented out. Laminar flow tests involved several runs of the program at different flow speeds and different longitudinal resolutions. An empty intestine was chosen as initial condition. A single block pulse of unit concentration and varying duration was given as input at t=0.

#### *4.1.1 Results*

At low spatial resolution, some "implicit diffusion" is obvious. This implicit diffusion is caused by the fact that the flow is not guaranteed to be an integer number of volume elements. Therefore, at each step each element is assigned a weighted average of the concentrations of two or more volume elements. This immediately implies some mixing which ought of course not to take place during laminar flow. Quantitatively, consider a row of volume elements (labelled 0,1,...,$M$) of equal volume $V$. Initially we have a concentration of 1.0 in element zero, zero in all others. If we have an incompressible flow, with a magnitude of just a fraction $p$ of a volume element per time step, after $T$ (<$M$) time steps, the distribution over the $M$ bins is given by:

$$[X]_j = \begin{cases} \frac{T!}{j!(T-j)!} p^j (1-p)^{T-j} & j \in \{0,1,\ldots,T\} \\ 0 & \text{elsewhere} \end{cases} \tag{4.1.1}$$

Which is just a binomial distribution. A slightly more general expression should include an integer part of the flow speed (flowspeed $I$=($n$+$p$)$V$/Δ$t$). In this case (4.1.1) becomes:

$$[X]_{nT+j} = \begin{cases} \frac{T!}{j!(T-j)!} p^j (1-p)^{T-j} & j \in \{0,1,\ldots,T\} \wedge (nT+j) \in \{0,1,\ldots,M\} \\ 0 & \text{elsewhere} \end{cases} \tag{4.1.3}$$

The standard deviation (in volume elements) of this distribution is just:

$$\sigma_{[X]} = \sqrt{Tp(1-p)} \leq \frac{\sqrt{T}}{2} \tag{4.1.2}$$

In keeping with real diffusion, the spread caused by implicit diffusion rises with the square root of the time elapsed. However, unlike real diffusion, standard deviation increases with the number of time *steps*, rather than the *total* time. A few large time steps will be more accurate than many small ones. Furthermore, the standard deviation expressed in units of physical length is proportional to the linear size of each volume element. Therefore, by increasing spatial resolution implicit diffusion may be reduced arbitrarily, and certainly to the extent that real diffusion, combined with the smearing out of substances by the velocity difference between tube centre and wall, are an order of magnitude larger.

In a separate test, the initial condition contained a block shaped lump of food, and its progress through the intestine was monitored for 200 time steps. The total amount of food present in the intestine was monitored. Until the pulse reached the end of the intestine, no change in this total amount was detected. This test showed that the laminar flow model does not act as either a source or sink.

## 4.2 Diffusion

As with laminar flow, diffusion was tested with the other factors in the model commented out. Five conditions to be met by the diffusion model were tested: (i) diffusion must conserve matter, (ii) diffusion of a delta function spike must yield an approximate Gaussian distribution with a standard deviation of $(\delta_F t)^{½}$, with $t$ the time elapsed, (iii) diffusion speed must be largely independent of temporal discretization and (iv) spatial discretization, and (v) the model must be numerically stable (no negative values of oscillations).

To verify the first condition, the boundary conditions in unit bounds were set to "hermetic sealing" for all boundaries. Total amount of all substances before and after runs were reported by routines EXPINI and FINREP respectively. No losses were detected for any geometry, even after runs of 100000 time steps.

The second condition was verified by computing the standard deviation of the distribution of food after introduction of a delta function spike at $t$=0 halfway up the intestine. The geometry was kept as a simple cylinder of unvarying diameter. The ratio of the spread and the square root of $t$ was computed and reported by REPROG. This ratio remained constant to within 0.1‰, throughout runs of up to 30000 time steps. As can be seen in figures 4.2.1 and 4.2.2, the distribution is indeed very nearly Gaussian.

Conditions (iii) and (iv) were tested by varying temporal and spatial discretization. The results of 4 runs with time steps running from 20 s to 2000 s are shown in figure 4.2.1. Only at the largest time steps is the diffusion underestimated noticeably. Even so, the result seems quite acceptable even for the largest time step.

Figure 4.2.2 shows the result of using different spatial resolutions (M=20, 50, 200) for time steps of 600 seconds. Again, the results are very similar for all resolutions.

The presence of oscillations and negative values was checked in all runs. Using the exponential forms of $w$-values derived from (3.6.2) no oscillations or negative values occurred, even at time steps of 3600s. The simple (Euler) formulations resulted in wild oscillations even at short time steps (60 s).

## 4.3 The metabolism

The metabolism was tested using a simplified program (chemostat.f) which lacked all facilities for transport. It is designed to simulate batch and continuous cultures in well mixed chemostats. The program serves as a test-bed for differential equation solvers for the type of equation specified in section 2.1. Apart from the NAG library (Numerical Algorithms Group

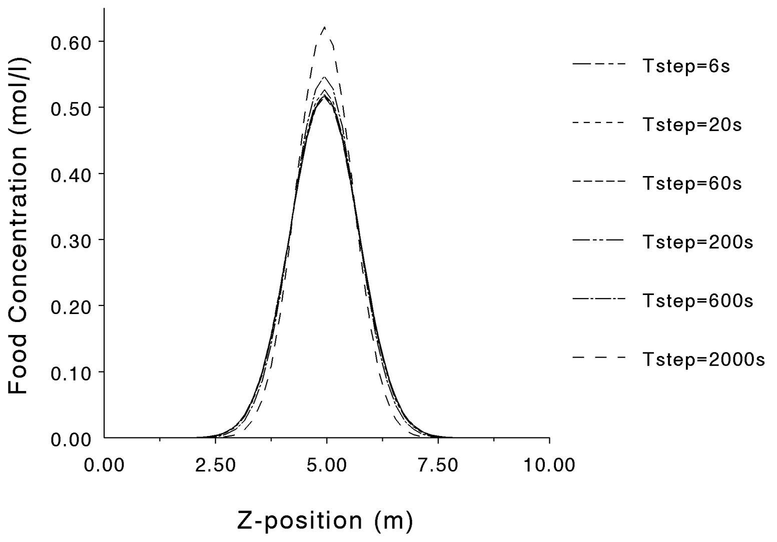


**Figure 4.2.1**. Distribution of food after $6x10^5$s for different temporal resolutions (M=100, $\delta_F$=1.0x$10^{-6}$). The solid line represents the theoretical Gaussian distribution.

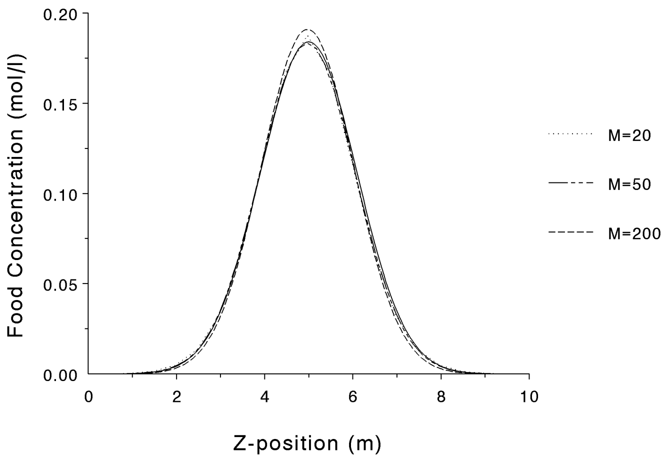


**Figure 4.2.2**. Distribution of food after $6x10^5$s using time steps of 60s and $\delta_F$=1.0x$10^{-6}$, for different longitudinal resolutions (M). The solid line represents the theoretical Gaussian distribution.

Ltd, Oxford, UK) D02BAE ninth order adaptive step Runge-Kutta-Merson method, a simpler, fourth order, single step Runge-Kutta method was implemented [Press *et al.* 1986]. Using the metabolisms for a strict aerobe and a moderate anaerobe (uptake inhibited by $O_2$) a batch run was set up using 0.01 mol/l of food, and 100 µmol/l of $O_2$, and cellular C for both species. Both equation solvers had problems in that they tended to show large negative values of food and oxygen, and bacteria growing happily on these negative food and oxygen supplies. The reason for this behaviour lies in the Michaelis-Menten terms in the equations:

$$[\dot{s}] \propto -\frac{[s]}{K_S + [s]} \tag{4.3.1}$$

If $K_S$ is small, then the change in [$s$] is independent of [$s$] except in a very small region, close to zero. Thus at high *absolute* concentrations [$s$] will be an approximately linear function of time. Only in when [$s$]≈$K_S$ does the exponential behaviour set in. To circumvent these problems, two methods were tested: (i) use D02EAE, a stiff equation solver from the NAG library, and (ii) modify the one-step fourth order equation solver in such a way that it avoids zero crossings, i.e, let it step carefully when concentrations get low.
The modification is simple. For each substance test:

$$[s] + [\dot{s}]\Delta t < \varepsilon[s] \tag{4.3.2}$$

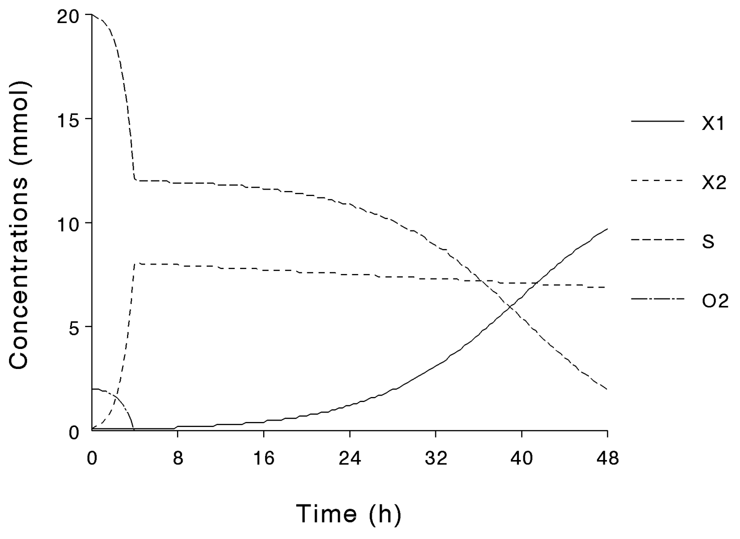
If this is the case compute a new time step:

$$\Delta t = \frac{(\varepsilon - 1)[s]}{[\dot{s}]} \tag{4.3.3}$$

Execute the single step Runge-Kutta using the shortest time step found for all substances. Repeat this procedure until the required time step has been reached. For an exponentially decaying function arround zero, and with 0≤ε<1, the desired step size should always be reached within a finite number of steps:

$$\Delta t = \frac{(\varepsilon - 1)Ae^{-Bt}}{-BAe^{-Bt}} = \frac{(1-\varepsilon)}{B} \tag{4.3.4}$$

Thus, the adaptive step size becomes a positive constant, which is indepent of $A$. The latter is important since it means that an error in the estimate of the next step, which means $A$ has



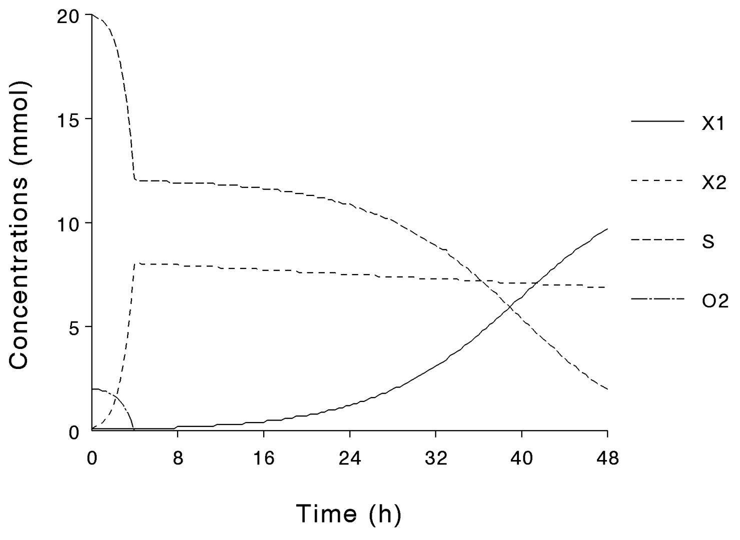


**Figure 4.3.1**. A comparison of two differential equation solving algorithms for a simulation of a batch culture of strict anaerobes (X1), strict aerobes (X2), consuming initial concentrations of substrate (S) and oxygen (O2): A) "one step" Runge-Kutta method; B) NAG library routine D02EAE for stiff differential equations. No significant differences could be determined.

been modified, does not enter into (4.2.4). The algorithm should not enter into an infinite loop.

Both methods were used in a re-run of the batch test and the results are shown in figure 4.3.1. No significant differences in behaviour can be seen in these experiments. Occasionally, the stiff equation solver may still cause zero crossings (not evident in the figure). More significantly, the NAG routine is far slower than the single step Runge-Kutta method.

## 4.4 Integration of all components.

The final testing of the complete program was done with the following parameters: radial subdivisions N=10, axial subdivisions M=100, intestinal length L=6 m with varying diamter; the first 4.98 m are the small intestine, with a radius of 1 cm; the next 18 cm are the "caecum" (radius 5 cm), followed by a "colon" of 84 cm long and 3 cm radius. All concentrations are given in mol/l: food and all bacteria in moles of organic carbon, oxygen simply in moles of molecular oxygen ($O_2$). To convert to numbers of bacteria, it was assumed that the volume of a single bacterium was $10^{-15}$ l (i.e. a maximum of $10^{12}$/g), and that they contained roughly 10% w/w of organic C. This yields a conversion factor from mol/l to bacteria/g of about $1.2 \cdot 10^{11}$.

Using the above model, experiments were done to simulate colonization in a sterile intestine. One or two species of bacteria, selected from three available types (strict aerobe, facultative anaerobe and strict anaerobe), were introduced into a sterile intestine, in which the oxygen concentration of the lumen was in equilibrium with the walls (0.1 mmol/l). The input of food, oxygen, and bacteria was in block waves with a 40% duty cycle. Food concentration at maximum was 7 mol/l, oxygen concentration 0.1 mmol/l, and in most experiments the food inflow contained a maximum of $1.2 \cdot 10^3$ bacteria/g of each species. Though this may be a bit high, runs with only 12 bacteria/g showed virtually identical results, so evidently this parameter is relatively unimportant in the initial colonization phase.

Figure 4.4.1 shows the resulting colonization of the intestine in this experiment. When strict anaerobes were introduced simultaneously with either facultative anaerobes or aerobes, the latter colonized within 1 day, reaching a maximum at day 4. After this, they were replaced by the anaerobes, which only appeared in any numbers at day 3. After 5 to 6 days a stable equilibrium was reached with strict anaerobes outnumbering facultatives or aerobes by 2.4-2.7 ${}^{10}$log steps. Small oscillations caused by the periodic input of food remained visible. Once stabilized, the population did not change if the influx of bacteria from the "stomach" reduced to zero, thus they had colonized the lumen. A further discussion of these results is given in Wilkinson (1997).

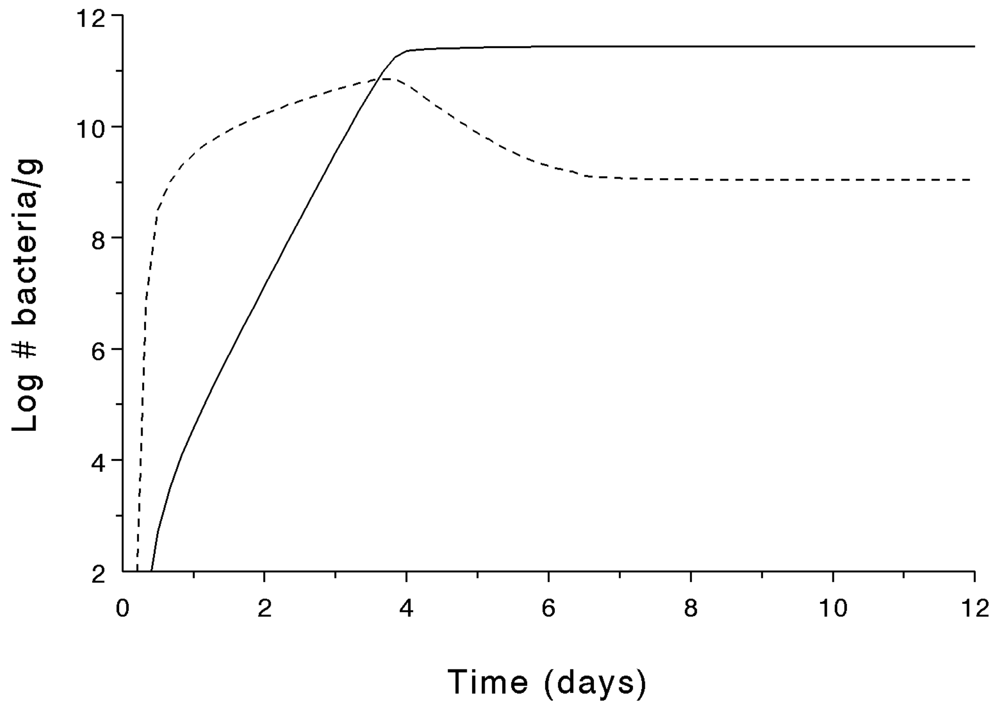


**Figure 4.4.1.** Colonization process in a di-associated sterile intestine modelled by computer simulation. Equal numbers of two species of bacteria (one strict (solid line) and one facultative anaerobe (dashed)) are fed into the sterile intestine, which contains an initial oxygen concentration of 0.1 mmol/l. Initially, the facultatives colonize, later, as oxygen levels drop, the strict anaerobes outcompete the facultatives.

## 5. Discussion

This report shows that it is possible to model the intestinal microflora and transport processes on the Cray J932 supercomputer in a comparatively simple program. However, at this point in time, the program does not run very well in parallel, i.e., only a single processor of the multiprocessor machine is used. In spite of this, all of the full scale tests could be run within a medium batch job, requiring less than 5 CPU hours of computing time. Use of parallel processors should be straightforward in the metabolic phase, and will be implemented in future versions. Furthermore, the "monolithic" program structure leaves something to be desired; breaking up the program into more-or-less independent modules must be done in future.

Further extensions should include: more species of bacteria and substrates, a proper mucosa, true peristalsis, and an immune system. Given a modular design, "plugging in" such components should not pose too many problems.

The MIMICS V.5 cellular automaton described in this paper is now being used for extended experiments in modelling the effect of selective decontamination.

## Acknowledgements

This study was made possible by generous funding from the Institute for Microbiology and Biochemistry, Herborn-Dill, Germany, and the International Study Group for New Antimicrobial Strategies. I would like to thank Drs R. de Bruin and J. Kraak of the Centre for High Performance Computing, University of Groningen, for many useful suggestions and discussions, and assistance with the visualization of the simulations, and Dr F. Schut of Microscreen B.V., for giving me much useful literature on bacterial metabolism.

## 6. References and Further Reading

Angelis, D.L. (1992) "Dynamics of Nutrient Cycling and Food Webs", Chapman and Hall, London, UK.

Gerritse, J., Schut, F., Gottschal, J.C. (1992) Modelling of mixed chemostat cultures of an aerobic bacterium *Comamonas testosteroni*, and an anaerobic bacterium *Veillonella alcalescens*: comparison with experimental data. *Appl. Environm. Microbiol.*, **58**, 1466-1476.

Numerical Algorithms Group Ltd (1990) "NAG Fortran Library Manual Mark 14", Vol 2, NAG Ltd, Oxford, UK.

Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T. (1986), “Numerical Recipes”, Cambridge University Press, Cambridge.

Wilkinson, M.H.F. (1997) Nonlinear dynamics, chaos-theory, and the "sciences of complexity": their relevance to the study of the interaction between host and microflora. In: "Old Herborn University Monograph Vol. 10: New Antimicrobial Strategies" (P. Heidt, V. Rusch, and D. van der Waaij, Eds.), Herborn Litterae, Herborn-Dill, Germany, pp. 111-130.