Mathematical Models of the Delipidation Cascade of Low Density Lipoproteins (LDL)

S. Das Gupta[*], R. Santos[†], S. Speakman[‡], D. Tello[§], C. Xue[¶]

Industrial Mentor: David Ross[||]

Institute for Mathematics and its Applications (IMA)
University of Minnesota
400 Lind Hall
207 Church Street S.E.
Minneapolis, MN 55455-0436
Tel: (612) 624-6066
FAX: (612) 626-7370
http://www.ima.umn.edu

August 10, 2005

Abstract

Small, dense LDL has been linked to coronary artery disease (CAD). Many studies have evaluated what measure to use: LDL itself, ApoB⁄Apo-A1 Ratio, NMR, C-Reactive protein... and how prescribed drugs a play part in the process. We devised a mathematical model of LDL size distribution that is characterized by its components and processes.

[*]Mississippi State University, Starkville, MS *sd181@msstate.edu*
[†]University of Nevada, Las Vegas, NV *santosr8@unlv.nevada.edu*
[‡]University of Kentucky, Lexington, KY *skylerspeakman@gmail.com*
[§]Arizona State University, Tempe, AZ *dtello@mathpost.asu.edu*
[¶]University of Minnesota, Minneapolis, MN cxue@math.umn.edu
[||]Kraiser Permanente, University of Virginia, Charlottesville, VA *david.s.ross@kp.org*

1 Introduction

Many health problems have been linked to low-density lipoproteins (LDL) such as high cholesterol (hypercholesterolaemia), high triglyceride (hypertriglycerolaemia), coronary artery disease (CAD), coronary heart disease (CHD) and others. Statistics have shown that "20% of US population have high blood cholesterol" [2]. Those whose total cholesterol level is above 240mg/dL are at high risk for having a heart attack (myocardial infarction-MI). The methods for early detection of cardiovascular disease measures are LDL itself, Apo-B/Apo-A1 ratio, NMR, C-reactive protein and others.

Our goal in this study is to devise a mathematical model of LDL size distribution that characterizes its component and processes so that we can eventually analyze data and given some parameters, predict results.

To find out what components and processes to include in our model, we read several papers of which we used Packard's 2003 diagram showing all the elements involved and decided on a list of key players: LDL containing TG, CE, and Apo-B; HDL containing TG, CE, and Apo-A; HL hepatic lipase, CETP Cholesterol Ester Transfer Protein and the liver which is our source and sink.

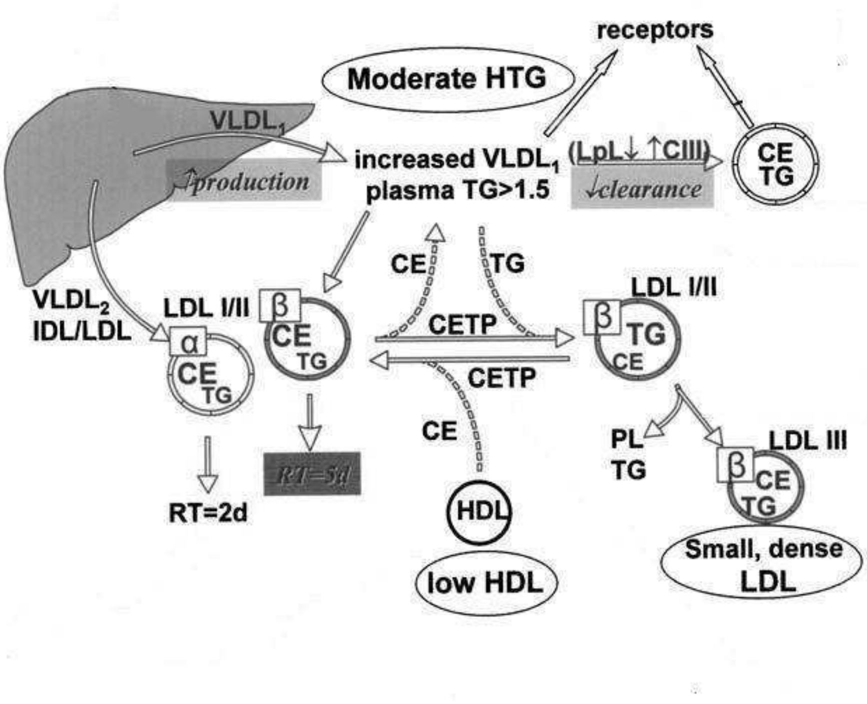
Next we had to find out how the key players interacted with one another and through much discussion and study found that CETP and HL were the transporters of TG and CE. The main difference between HDL and LDL were the apolipoproteins Apo-A1 and Apo-B100 respectively which evolved into our terms $A_{i,j}$ and $B_{i,j}$. Our first attempt, the Toy model, only had one subscript A_j which adds a TG making a new component A_{j+1} and using the law of mass action we got a system of the differential equations and its equilibrium states.

Our next task was to develop a two dimensional model that reflects the reactions of CETP and HL. We originally had CETP going in both directions and HL only going in one direction. We incorporated α, β and γ to represent our CETP and HL processes and created a partial differential equation with initial and boundary conditions but we had little direction and could not analyze it's stability and therefore set it for future study.

We found out later that HDL also transports triglycerides so now we had to adjust our chemical reactions to include such a transfer. Finally, we ended with five reactions for $A_{i,j}$ and 5 reactions for $B_{i,j}$ and ended up with a monstrous system of differential equations that we solved numerically using MATLAB's Runga Kutta Method RK45.

2 The Model

The formation of our model was based on Packards 2003 Diagram. The list of chemical players involved in our model are LDL, HDL, CETP, HL and the liver.



2.1 The Toy Model

Suppose that in some well-mixed solution there is a fixed number of molecules of species A (think Apo-B). Any number of molecules of species T (there is a fixed number of T molecules too, for the moment), can glomb on to one of these A molecules. However, the propensity for T molecules to bind to an A depends on how many T molecules are already bound to that A.

Let A_j be the number of A molecules with molecules of species T bound to them. Let f_j and r_j be the second-order forward binding rate and the first-order dissociation rates, respectively. Let be the concentration of species T in solution. Then a simple model is

$$A_{j-1} + T \underset{r_{i,j}}{\overset{f_{i-1,j}}{\rightleftarrows}} A_j$$

$$\begin{aligned}
\frac{dA}{dt} &= -r_{j-1}A_j + f_{j-1}TA_{j-1} \\
\frac{dA_{j-1}}{dt} &= r_{j-1}A_j - (r_{j-2} + f_{j-1}T)A_{j-1} + f_{j-2}TA_{j-2} \qquad (1) \\
\frac{dA_0}{dt} &= r_0A_1 - f_0TA_0 \\
\frac{dT}{dt} &= \sum_{i=0}^{j-1} r_iA_{i+1} - T\sum_{i=0}^{j-1} f_iA_i
\end{aligned}$$

If $\overline{A}$ is the fixed number of A molecules and $\overline{T}$ the fixed number of T molecules, then well have

$$\overline{T} = T + \sum_{i=0}^{j} iA_i$$

$$\overline{A} = \sum_{i=0}^{j} A_i$$

Were only interested in equilibria. The equilibrium equations are

$$\begin{aligned}
A_j &= \frac{f_{j-1}T}{r_{j-1}}A_{j-1} \\
A_{j-1} &= \frac{r_{j-1}A_j + f_{j-2}TA_{j-2}}{r_{j-2} + f_{j-1}T} \\
A_1 &= \frac{f_0T}{r_0}A_0 \\
T &= \frac{\sum_{i=0}^{j-1} r_iA_{i+1}}{\sum_{i=0}^{j-1} f_iA_i}
\end{aligned}$$

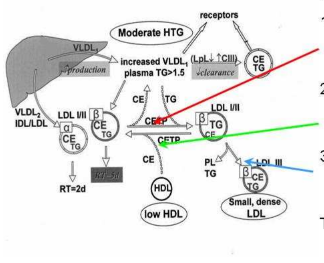
This is, emphatically, a toy model, it omits stuff that we simply cant omit, but it has the structure that Dr. Ross thinks were looking for. You can solve the equilibrium equations in the case in which all of the f_j are the same and all of the r_j are the same, and obtain the following Jacobian.

$$\begin{pmatrix}
-r & fT & 0 & 0 & 0 & 0 & 0 & fA_{j-1} \\
r & -(r+fT) & fT & 0 & 0 & 0 & 0 & f(A_{j-2} - A_{j-1}) \\
0 & r & -(r+fT) & fT & 0 & 0 & 0 & f(A_{j-3} - A_{j-2}) \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & -(r+fT) & fT & 0 & f(A_1 - A_2) \\
0 & 0 & 0 & \dots & r & -(r+fT) & fT & f(A_0 - A_1) \\
0 & 0 & 0 & \dots & 0 & r & -fT & -fA_0 \\
r & r - fT & r - fT & \dots & r - fT & r - fT & -fT & -f\sum_{i=0}^{j-1} A_i
\end{pmatrix}$$

2.2 The Two Dimensional Toy Model

Our second model used Packard's 2003 Diagram to incorporate the correct mechanisms into our toy model. We had several processes that we needed to reflect: 1) Exchange of CE and TG by CETP. 2) Removal of CE from HDL by CETP. 3) Removal of TG from LDL by HL. These three reactions are directly responsible for LDL delipidation.

Since the toy model is the one dimensional case of gaining or losing TG, we naturally went to the two dimensional case where you gain, lose or exchange a CE and TG, denoted by the indices i and j respectively.



Chemical reactions incorporated:

1. CETP facilitates the exchange of CE for TG from VLDL to LDL
2. CETP also facilitates the process of HDL taking CE from LDL to VLDL.
3. HL removes TG from LDL transforming it from LDL/IDL to small, dense LDL

These three reactions are directly responsible for LDL delipidation.

$$B_{i,j} + T \underset{r}{\overset{f}{\rightleftharpoons}} B_{i,j+1}$$

$$B_{i,j} + C \underset{\rho}{\overset{\phi}{\rightleftharpoons}} B_{i+1,j}$$

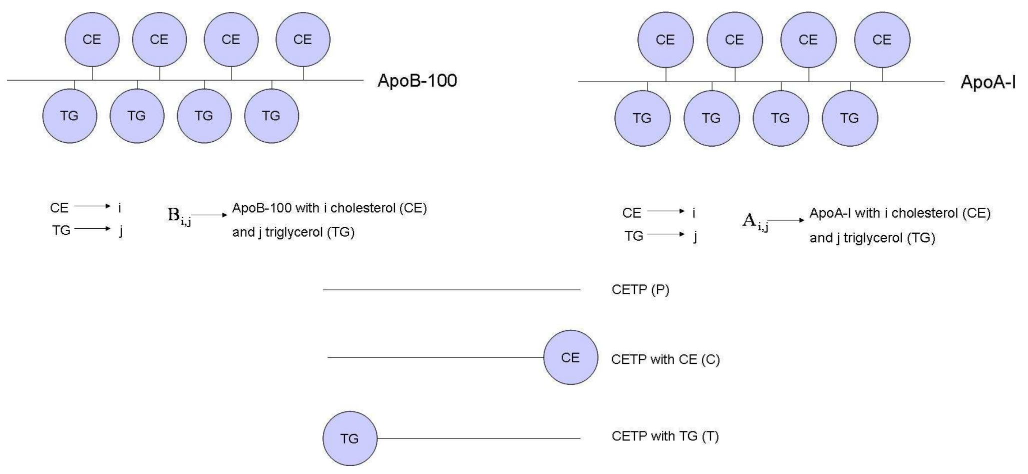
Parameter	Description
$B_{i,j}$	Number of Apo-B (in LDL) associated with i (CE) and j (TG)
$\overline{B}$	Total amount of LDL (Apo-B)
C	Number of free CE in plasma
$\overline{C}$	Total amount of CE
T	Number of free TG in plasma
$\overline{T}$	Total amount of TG
f	Rate of the reaction in which $B_{i,j}$ gets a TG
r	Rate of the reaction in which $B_{i,j}$ loses a TG
ϕ	Rate of the reaction in which $B_{i,j}$ gets a CE
ρ	Rate of the reaction in which $B_{i,j}$ loses a TG

Table 1: Definitions of Parameters of 2^{nd} Model

$$
\begin{aligned}
\frac{dB_{i,j}}{dt} &= fT(B_{i,j-1} - B_{i,j}) + r(B_{i,j+1} - B_{i,j}) + \phi C(B_{i-1,j} - B_{i,j}) + \rho(B_{i+1,j} - B_{i,j}) \\
\frac{dT}{dt} &= \sum_{i=0}^{n-1}(r_i \sum_{j=1} B_{i,j} - fT \sum_{j=0} B_{i,j}) \\
\frac{dC}{dt} &= \sum_{j=0}(\rho \sum_{i=1} B_{i,j} - \phi C \sum_{i=0} B_{i,j}) \qquad (2) \\
\overline{T} &= T + \sum_{i=0}\sum_{j=0} jB_{i,j} \\
\overline{C} &= C + \sum_{i=0}\sum_{j=0} iB_{i,j} \\
\overline{B} &= \sum_{i=0}\sum_{j=0} B_{i,j}
\end{aligned}
$$

2.3 The Double Two Dimensional Toy Model

The diagrams below show that apolipoproteins serve as backbones to TG and CE of LDL and HDL. We adopted the model that has a pool of CETP meaning the CETP can have nothing on it ($B_{0,0}$), have one CE on it ($B_{1,0}$) or have one TG on its backbone ($B_{0,1}$).


CE
TG
ApoB-100
ApoA-I
CE → i
TG → j
$B_{i,j}$ → ApoB-100 with i cholesterol (CE) and j triglycerol (TG)
$A_{i,j}$ → ApoA-I with i cholesterol (CE) and j triglycerol (TG)
CETP (P)
CETP with CE (C)
CETP with TG (T)

Chemical Reactions	**Reaction Rates**	**Description**
$A_{i,j} + P \longrightarrow A_{i,j-1} + T$	$\tau^A_{i,j}$	$A_{i,j}$ loses a TG at rate $\tau^A_{i,j}$
$A_{i,j} + P \longrightarrow A_{i-1,j} + C$	$\gamma^A_{i,j}$	$A_{i,j}$ loses a CE at rate $\gamma^A_{i,j}$
$A_{i,j} + T \longrightarrow A_{i,j+1} + P$	$\theta^A_{i,j}$	$A_{i,j}$ gets a TG at rate $\theta^A_{i,j}$
$A_{i,j} + C \longrightarrow A_{i+1,j} + P$	$\mu^A_{i,j}$	$A_{i,j}$ gets a CE at rate $\mu^A_{i,j}$
$A_{i,j} + H \longrightarrow A_{i,j-1}$	$\alpha^A_{i,j}$	$A_{i,j}$ undergoes hydrolysis at rate $\alpha^A_{i,j}$
$B_{i,j} + P \longrightarrow B_{i,j-1} + T$	$\tau^B_{i,j}$	$B_{i,j}$ loses a TG at rate $\tau^B_{i,j}$
$B_{i,j} + P \longrightarrow B_{i-1,j} + C$	$\gamma^B_{i,j}$	$B_{i,j}$ loses a CE at rate $\gamma^B_{i,j}$
$B_{i,j} + T \longrightarrow B_{i,j+1} + P$	$\theta^B_{i,j}$	$B_{i,j}$ gets a TG at rate $\theta^B_{i,j}$
$B_{i,j} + C \longrightarrow B_{i+1,j} + P$	$\mu^B_{i,j}$	$B_{i,j}$ gets a CE at rate $\mu^B_{i,j}$
$B_{i,j} + H \longrightarrow B_{i,j-1}$	$\alpha^B_{i,j}$	$B_{i,j}$ undergoes hydrolysis at rate $\alpha^B_{i,j}$

Table 2: Reactions and Rates

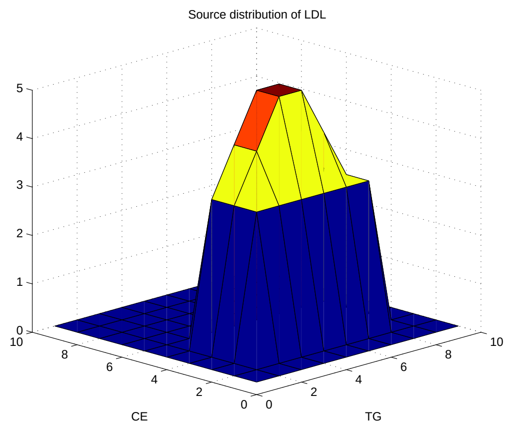
$$
\begin{aligned}
\frac{dB_{i,j}}{dt} &= -[P(\gamma^B_{i,j} + \tau^B_{i,j}) + C\mu^B_{i,j} + T\theta^B_{i,j} + H\alpha^B_{i,j}]B_{i,j} + C\mu^B_{i-1,j}B_{i-1,j} + T\theta^B_{i,j-1}B_{i,j-1} \\
&\quad + P\gamma_{i+1,j}B_{i+1,j} + (P\tau^B_{i,j+1} + H\alpha^B_{i,j+1})B_{i,j+1} + \hat{B}_{i,j} \\
\frac{dA_{i,j}}{dt} &= -[P(\gamma^A_{i,j} + \tau^A_{i,j}) + C\mu^A_{i,j} + T\phi^A_{i,j} + H\alpha^A_{i,j}]A_{i,j} + C\mu^A_{i-1,j}A_{i-1,j} + T\theta^A_{i,j-1}A_{i,j-1} \\
&\quad + P\gamma_{i+1,j}A_{i+1,j} + (P\tau^A_{i,j+1} + H\alpha^A_{i,j+1})A_{i,j+1} + \hat{A}_{i,j} \\
\frac{dC}{dt} &= P\sum_{i,j=1}(\gamma^B_{i,j}B_{i,j} + \gamma^A_{i,j}A_{i,j}) - C\sum_{i,j=1}(\mu^B_{i-1,j}B_{i-1,j} + \mu^A_{i-1,j}A_{i-1,j}) \\
\frac{dT}{dt} &= P\sum_{i,j=1}(\tau^B_{i,j}B_{i,j} + \tau^A_{i,j}A_{i,j}) - T\sum_{i,j=1}(\theta^B_{i,j-1}B_{i,j-1} + \theta^A_{i,j-1}A_{i,j-1}) \\
\frac{dP}{dt} &= -P\sum_{i,j=1}(\gamma^B_{i,j} + \tau^B_{i,j})B_{i,j} + (\gamma^A_{i,j} + \tau^A_{i,j})A_{i,j} + T\sum_{i,j=1}(\theta^B_{i,j-1}B_{i,j-1} + \theta^A_{i,j-1}A_{i,j-1}) \\
&\quad + C\sum_{i,j=1}(\mu^B_{i-1,j}B_{i-1,j} + \mu^A_{i-1,j}A_{i-1,j}) \\
\frac{dH}{dt} &= -H\sum_{i,j=1}(\alpha^B_{i,j}B_{i,j} + \alpha^A_{i,j}A_{i,j}) + \hat{H} \\
\overline{B} &= \sum_{i,j=0} A_{i,j} \\
\overline{A} &= \sum_{i,j=0} B_{i,j} \\
\overline{C} &= C + \sum_{i,j=0} i(A_{i,j} + B_{i,j}) \\
\overline{T} &= T + \sum_{i,j=0} j(A_{i,j} + B_{i,j}) \\
\overline{P} &= P + T + C
\end{aligned}
$$

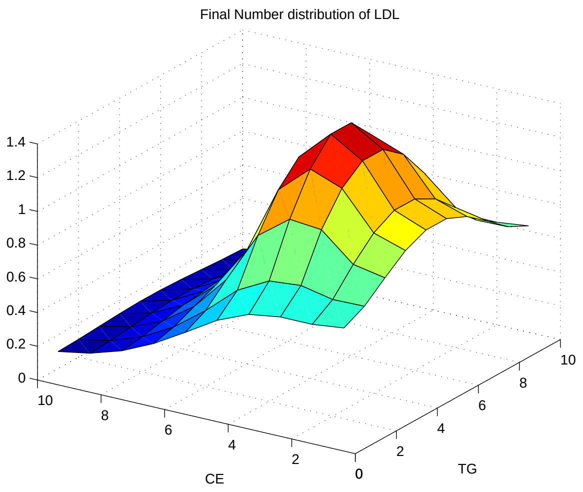
Parameter	Description
$A_{i,j}$	Amount of Apo-A (in HDL) that associated with i (CE) and j (TG)
$\hat{\text{A}}$	Source/sink of $A_{i,j}$
$\overline{A}$	Total amount of HDL (Apo-A)
$B_{i,j}$	Number of Apo-B (in LDL) that associated with i (CE) and j (TG)
$\hat{\text{B}}$	Source/sink of $B_{i,j}$
$\overline{B}$	Total amount of LDL (Apo-B)
C	Amount of CETP grabbing CE (CETP-CE)
$\overline{C}$	Total amount of CE
T	Amount of CETP grabbing TG (CETP-TG)
$\overline{T}$	Total amount of TG
P	Amount of free CETP in plasma
$\overline{P}$	Total amount of CETP
$\hat{\text{P}}$	Source/sink of P
H	Amount of free HL in plasma
$\hat{\text{H}}$	Source/sink of H

Table 3: Definitions of New Parameters

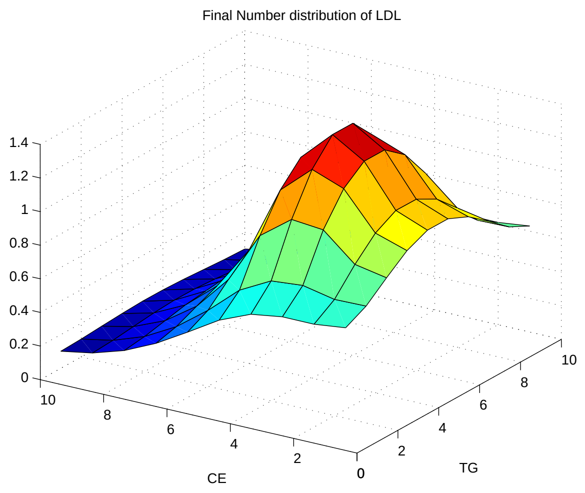
2.4 Analysis

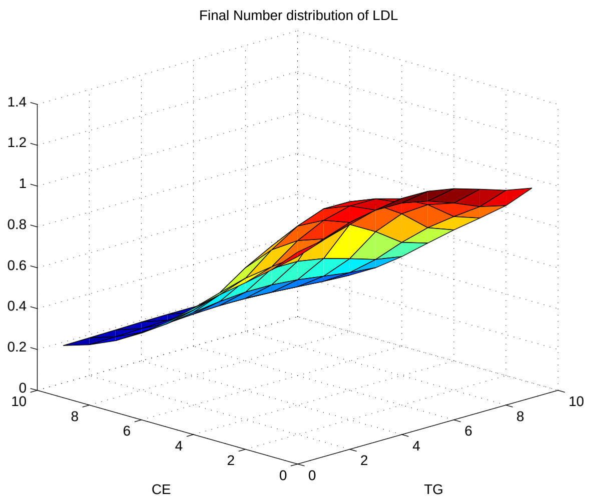
In order to gain a better understanding of the underlying intricacies of our model, we coded the system of differential equations into Matlab and solved it using Runge-Kutta 4-5. With this new tool we were able to develop a series of cause and effect relationships that were not obvious from the system of ODEs alone. On the whole, our virtual system correlates with the biological ones described in the literature. Below are some of these examples.



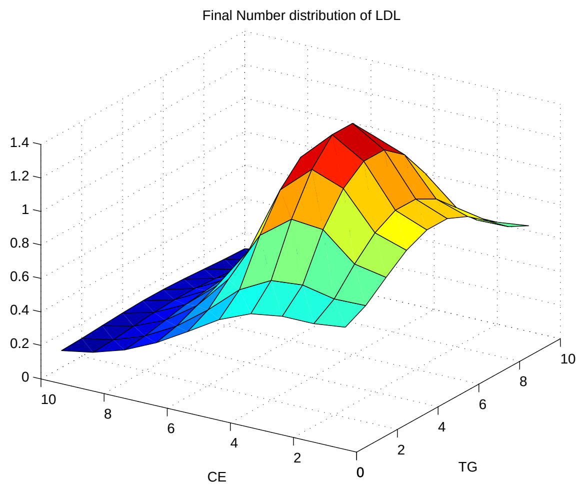


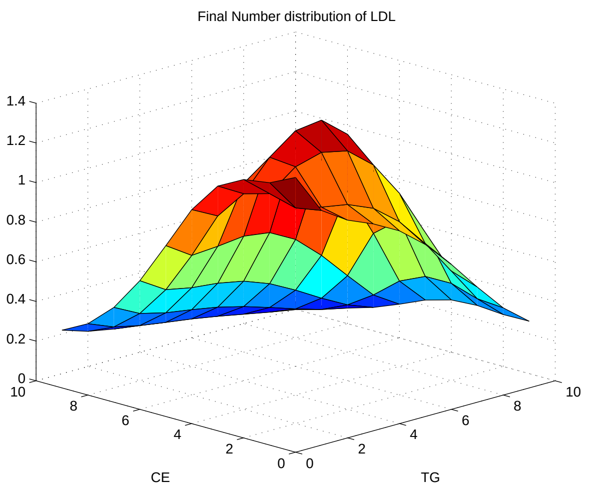
These graphs are examples of output from the code. The graph on the left represents a source distribution. We have coded the source to create a mixture of LDLs. These LDLs have a CE count between 3-5 and a TG count between 2-8. Biologicaly this source represents the liver. The graph on the right is the resulting distribution from this source. Nominal parameters were used in this simulation. This chart will be the standard to which we will compare the rest of the LDL distributions.



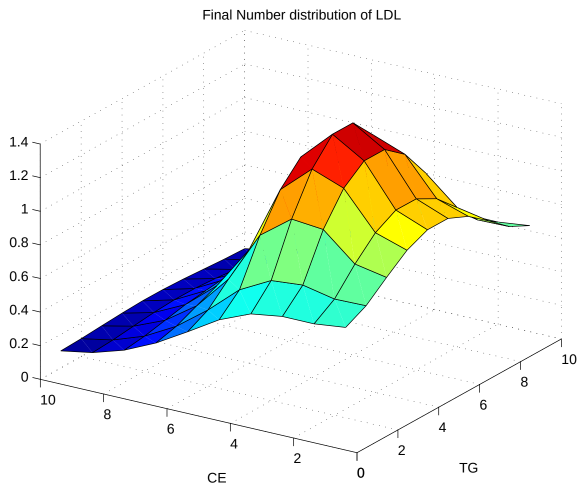


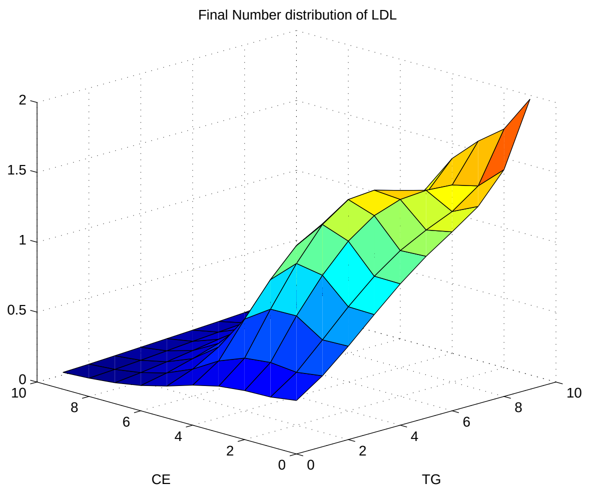
The distribution on the right is a result of increasing the CETP count in the system. Because there are more CETPs, interactions between the various LDLs / IDLs / VLDLs are more common. This leads to a more even distribution than we have in our standard. As a result there is a small increase in SDLDL.





The distribution on the right is a result of increasing hepatic lipase in the system. Hepatic lipase interacts with the LDLs and removes a TG from them. It follows that increasing this rate would shift the distribution away from VLDLs. Our model follows accordingly.





The distribution on the right is a result of increasing HDL in the system. HDL, or good cholesterol removes CE from the system. Mathematically, we reperesent this by giving HDL a higher sink rate. The CETPs interact between the LDL and the now prevalent HDL. However, the HDL are removed from the system more rapidly, taking CE with them. As a result, there are less CE in remaining LDLs. The model represents this by leaving a heavily skewed distribution in favor of VLDLs. This large amount of HDL virtually eradicates any SDLDL in the system.

3 Concluding Remarks and Future Work

We believe that we have accomplished our goal of creating a mathematical model for LDL Delipidation by creating a system of ODE's that reflects the mechanisms occurring in the system. Although an analytical solution may be possible but cannot be attained in the timeframe given, we solved the system numerically with MATLAB RK45. The system does numerically have steady states.

For future work, we would like to incorporate real source terms and reaction rates to improve our ODE model, and add in prescibed drugs like statins and fibrates to test the model. We need to continue the steady-state analysis of our ODE model and add more features to the MATLAB code.

If we allow the amount of CE and TG in HDL take continous value and suppose that the distribution of $\rho(x, y)$ of LDL flows at rate $\vec{G}(x, y)$ with source term of $f(x, y)$. We obtain the following PDE model:

$$\frac{\partial \rho(x,y)}{\partial t} \quad + \quad \nabla \cdot (\vec{G}(x,y)\rho(x,y)) = f(x,y)$$

and what we need to do is find a way on how to interpret $\vec{G}(x, y)$ in terms of the chemical reactions rates.

4 Acknowledgements

We would like to thank Dr. David Ross, Dr. Richard J. Braun, Dr.Santosa, Dr. Duprez, Dr. Arnold, The Institute for Mathematics and Its Applications(IMA), and the University of Minnesota for their hospitality and the opportunity to participate in the program.

David Tello would like to thank Reynaldo Castro-Estrada, Edgar Diaz, Arlene Evangelista, Cristi Guevara, and Alicia Urdapilleta; graduate students at Arizona State University for the help provided with the manuscript. Chuan Xue would like to thank Prof. Hans Othmer for his recommendations that help substantially understand the elaboration and development of the models. Rina Santos would like to thank her husband Raymond Santos for taking care of our four children for ten long days and also thanks to Dr. Richard Braun for his encouragement and help throughout our project. Skyler Speakman would like to thank John Sabino from Rice University and Valjean Eleander from UNLV for providing valuable input which lead to improving the matlab code.

The authors are solely responsible for the views and opinions expressed in this research; it does not necessarily reflect the ideas and/or opinions of the IMA or the University of Minnesota.

5 Appendix

5.1 Abbreviations

Apo	apolipoprotein
CAD	coronary artery disease
CE	cholesteryl ester
CETP	cholesteryl ester transfer protein
CHD	coronary heart disease
HDL	high density lipoprotein
HL	hepatic lipase
IDL	intermediate density lipoprotein
LDL	low density lipoprotein
LPL	lipoprotein lipase
MI	myocardial infraction
NMR	nuclear magnetic resonance
PL	phospholipid
TG	triglyceride or triacylglycerol
TRL	triglyceride-rich lipoprotein
VLDL	very low density lipoprotein

5.2 Matlab Code

```
function team5
%------------------------Team5---------------------
%  Skyler Speakman, University of Kentucky
%  This is farm from optimized but it seems to get the job done.
%  This code sets up and runs a built in Matlab ODE solver rk45.
%  Team5 is the overhead involved and system gives the ODE's
%  To run it, change your directory to the one containing the team5.m
%  and type team5 in matlab.  Or you can use the 'run' button above
%-------------------------------------------------
clear all;

global M Phat Hhat Ahat Bhat alphaA alphaB gammaA gammaB muA muB ...
       tauA tauB thetaA thetaB lambda;


%----------------------gives the 'size' of our distributions-----
M=10;                      %"capacity" of an ldl.  In reality this
                         %means a lipid can hold M-1
                         % TG's or CE's
%---------------------------------------------------------------


%--------------------how long to run the simulation---------------
tspan = [0;1];          % time span...I really have no idea
```

```
%-----------------------------------------------------------------


%------------INITIAL CONDITIONS-------------------------
%special attention on A0.  HDL cannot have any TG.
A0= zeros(M);
%A0(:,1) = 1;
%temp = eye(M)*1;
%A0 = repmat(temp(:,1),1,M)'
lambda = .25; B0= zeros(M);
%B0(3,8) = 2;
%B0(2,5) = 17;
%B0(2,2) = 65;
%B0(2,1) = 500;

P0 = 30;             % initial number of empty cept
T0 = 0;              % initial number of cetp with TG
C0 = 0;              % initial number of cetp with CE
H0 = 0;             % inital number of Hepatic lipase

%---------------------------------------------------------


%----------------CONSTANTS------------------------------

%----------sink / source constants ---
Phat = 0;  %just made up
Hhat = 0; %just made up
temp = eye(M)*1; Ahat = repmat(temp(:,1),1,M)*2'; Ahat = zeros(M)
Bhat = zeros(M);
%Bhat(3,8) =2;
%Bhat(2,5) = 17;
%Bhat(2,2) = 65;
%Bhat(2,1) = 500
%Bhat(5,5) = 20;
Bhat(3:5,3:8) = 3; Bhat(4:5,4:7) = 4; Bhat(4:5,5:6) = 5;


%---------reaction rate constants---
alphaB = ones(M)/6; gammaA = ones(M); gammaB = ones(M); muA    =
ones(M)*40; muB    = ones(M)*40; tauB   = ones(M); thetaB =
ones(M)*40;

tauB(:,1)=0; thetaB(:,M)=0; gammaA(1,:) = 0; gammaB(1,:) = 0;
muA(M,:) = 0; muB(M,:) = 0;

%----below stay 0 to keep TG from interacting with HDL
alphaA = ones(M); tauA = ones(M); thetaA = ones(M);
```

```
%--------------------------------------------------------

%---------Performing some overhead and calling ODE45---
A0vec = reshape(A0,M*M,1);           %putting the init dist in vector form
B0vec = reshape(B0,M*M,1);           %don't touch
z0 = [ P0; T0; C0; H0; A0vec; B0vec]; %forming init vector
[t,z] =ode45(@system, tspan, z0);     % calling ode solver
%-------------------------------------------------------

%-----------------OUTPUT---------------------

%---------displaying on screen
disp('solution:'); disp('P cetp:'); disp(z(end,1)); disp('T cetp:');
disp(z(end,2)'); disp('C cetp:'); disp(z(end,3)'); disp('H Hepatic
lipase:'); disp(z(end,4)'); A = reshape(z(end,5:M*M+4),M,M); Ahat; B
= reshape(z(end,M*M+5:end),M,M);
%-------------------------------------


%-------------plots----------------
figure(1) subplot(2,2,1) plot(t,z(:,1)); grid on xlabel('time')
ylabel('P') subplot(2,2,2) plot(t,z(:,2)); grid on xlabel('time')
ylabel('T') subplot(2,2,3) plot(t,z(:,3)); grid on xlabel('time')
ylabel('C') subplot(2,2,4) plot(t,z(:,4)); grid on xlabel('time')
ylabel('H')
%------------------

figure(2)
%subplot(2,1,1)
%surf(A0);
%plot(1:M,Ahat(:,1));
%view(-40,30);
%xlabel('TG');
%xlabel('CE');
%ylabel('Number distribution of HDL');
%title('Source distribution for HDL');
%subplot(2,1,2)
%plot(1:M,A(:,1))
%xlabel('CE')
%ylabel('Number distribution of HDL');
%surf(A);
%view(90,0);
%xlabel('TG');
%ylabel('CE');
%zlabel('Number distribution of HDL');
%title('Final Number distribution of HDL');
subplot(1,2,1) surf(Bhat); view(-45,20); xlabel('TG'); ylabel('CE');
%zlabel('Number distribution of LDL');
```

```
title('Source distribution of LDL'); subplot(1,2,2) surf(B);
view(-45,20); xlabel('TG'); ylabel('CE');
%zlabel('Number distribution of LDL');
title('Final Number distribution of LDL'); figure(5) surf(B);
view(-45,20); xlabel('TG'); ylabel('CE');
%zlabel('Source Distribution of LDL');
title('Final Number distribution of LDL');
%----------------------------------


%---------------------------------------------

function dz = system(t,z)
%----------------------SYSTEM-------------------
% Skyler Speakman, University of Kentucky
% The following code is called by team5.  It is passed t, z
% and updates dz accordingly.  dz is then passed back to the ode solver
% called in team5.
%---------------------------------------------
global M Phat Hhat Ahat Bhat alphaA alphaB gammaA gammaB muA muB ...
       tauA tauB thetaA thetaB lambda;


%P=z(1); T=z(2); C=z(3); H=z(4);  % these can be uncommented and all
                                  % the z(1)...z(4)'s below can be
                                  % replaced with P, C, T, H.

%-------------------------------reshape-----------
%the built in ode requires that z and dz be passed back and forth
%as vectors, but forming and displaying them are easier in array notation
%reshape is the step in between these 2.
A = reshape(z(5:M*M+4),M,M); B = reshape(z(M*M+5:end),M,M);

%-------------------------------------------------

% dz= [ dpdt, dTdt , dCdt, dHdt, ...dAdt..., ...dBdt...]
% Above gives the general form of dz, the first 4 are scalars
% and after that we use reshape on the arrays dAdt, dBdt to
% put them in vector form


%------------------------forming the first 4 elements of dz---------
dPdt = Phat + sum(sum( - gammaB.*B*z(1) - tauB.*B*z(1) + muB.*B*z(3)
...
                       + thetaB.*B*z(2) - gammaA.*A*z(1) ...
                       - tauA.*A*z(1) + muA.*A*z(3) + thetaA.*A*z(2) ));

dTdt = sum(sum( tauB.*B*z(1) - thetaB.*B*z(2) + tauA.*A*z(1)...
                             - thetaA.*A*z(2)));

dCdt = sum(sum( gammaB.*B*z(1) - muB.*B*z(3) + gammaA.*A*z(1) ...
                             - muA.*A*z(3)));

dHdt = Hhat +  sum(sum( -alphaB.*B*z(4) - alphaA.*A*z(4)));
%-------------------------------------------------------------------

%-------------forms an M*M array representing dA/dt(i,j)--------------
%I'm pretty sure there's a faster way of doing this but this works for now
%I take into account all the possible array access errors
% muA(0,1) doesn't exist.  Same thing happens on the other side
% muA(M+1,0) doesnt work either.  Below is the if/else logic
% for the 4 sides and 4 corners.
for i = 1:M
    for j = 1:M
        if (i>1) && (j>1) && (i<M) && (j<M) %in the middle everything possible
            dAdt(i,j) =-gammaA(i,j)*A(i,j)*z(1) ...
                       - tauA(i,j)*A(i,j)*z(1) ...
                       - muA(i,j)*A(i,j)*z(3) ...
                       - thetaA(i,j)*A(i,j)*z(2) ...
                       - alphaA(i,j)*A(i,j)*z(4) ...
                       + muA(i-1,j)*A(i-1,j)*z(3) ...
                       + thetaA(i,j-1)*A(i,j-1)*z(2) ...
                       + gammaA(i+1,j)*A(i+1,j)*z(1) ...
                       + tauA(i,j+1)*A(i,j+1)*z(1) ...
                       + alphaA(i,j+1)*A(i,j+1)*z(4) ...
                       + Ahat(i,j) - 5*lambda*A(i,j);

        elseif (i==1) && (1<j) && (j<M) %on the top, No CE's to give up
            dAdt(i,j) = - tauA(i,j)*A(i,j)*z(1) ...
                       - muA(i,j)*A(i,j)*z(3) ...
                       - thetaA(i,j)*A(i,j)*z(2) ...
                       - alphaA(i,j)*A(i,j)*z(4) ...
                       + thetaA(i,j-1)*A(i,j-1)*z(2) ...
                       + gammaA(i+1,j)*A(i+1,j)*z(1) ...
                       + tauA(i,j+1)*A(i,j+1)*z(1) ...
                       + alphaA(i,j+1)*A(i,j+1)*z(4) ...
                       + Ahat(i,j)- lambda*A(i,j);
                 % + muA(i-1,j)*A(i-1,j)*z(3) ...
                 % -gammaA(i,j)*A(i,j)*z(1) ...
        elseif (1<i) &&(i<M)  && (j==1)% on the left, No TE's to give up
                                                % can't gain a TE
            dAdt(i,j) =-gammaA(i,j)*A(i,j)*z(1) ...
                       - muA(i,j)*A(i,j)*z(3) ...
                       - thetaA(i,j)*A(i,j)*z(2) ...
                       + muA(i-1,j)*A(i-1,j)*z(3) ...
                       + gammaA(i+1,j)*A(i+1,j)*z(1) ...
```

```
               + tauA(i,j+1)*A(i,j+1)*z(1) ...
               + alphaA(i,j+1)*A(i,j+1)*z(4)...
               + Ahat(i,j) - lambda*A(i,j);
         % + thetaA(i,j-1)*A(i,j-1)*z(2) ...
         %- tauA(i,j)*A(i,j)*z(1) ...
         %- alphaA(i,j)*A(i,j)*z(4) ...
elseif (i==M) && (1<j) && (j<M)% on the bottom side Full CE
    dAdt(i,j) = -gammaA(i,j)*A(i,j)*z(1) ...
               - tauA(i,j)*A(i,j)*z(1) ...
               - thetaA(i,j)*A(i,j)*z(2) ...
               - alphaA(i,j)*A(i,j)*z(4) ...
               + muA(i-1,j)*A(i-1,j)*z(3) ...
               + thetaA(i,j-1)*A(i,j-1)*z(2) ...
               + tauA(i,j+1)*A(i,j+1)*z(1) ...
               + alphaA(i,j+1)*A(i,j+1)*z(4) ...
               + Ahat(i,j) - lambda*A(i,j);
               %+ gammaA(i+1,j)*A(i+1,j)*z(1) ...
               %- muA(i,j)*A(i,j)*z(3) ...


elseif (1<i) && (i<M) && (j==M) % on the right full of TG
    dAdt(i,j) = -gammaA(i,j)*A(i,j)*z(1) ...
               - tauA(i,j)*A(i,j)*z(1) ...
               - muA(i,j)*A(i,j)*z(3) ...
              - alphaA(i,j)*A(i,j)*z(4) ...
               + muA(i-1,j)*A(i-1,j)*z(3) ...
               + thetaA(i,j-1)*A(i,j-1)*z(2) ...
               + gammaA(i+1,j)*A(i+1,j)*z(1) ...
               + Ahat(i,j) - lambda*A(i,j);
             %  + tauA(i,j+1)*A(i,j+1)*z(1) ...
             %  + alphaA(i,j+1)*A(i,j+1)*z(4) ...
             % - thetaA(i,j)*A(i,j)*z(2) ...


elseif (i==1) && (j==1)%upper left
    dAdt(i,j) = - muA(i,j)*A(i,j)*z(3) ...
               - thetaA(i,j)*A(i,j)*z(2) ...
               + gammaA(i+1,j)*A(i+1,j)*z(1) ...
               + tauA(i,j+1)*A(i,j+1)*z(1) ...
               + alphaA(i,j+1)*A(i,j+1)*z(4) ...
               + Ahat(i,j)- lambda*A(i,j);
                 % + muA(i-1,j)*A(i-1,j)*z(3) ...
              % + thetaA(i,j-1)*A(i,j-1)*z(2) ...
              ... %-gammaA(i,j)*A(i,j)*z(1) ...
                  ...%- tauA(i,j)*A(i,j)*z(1) ...
                  %- alphaA(i,j)*A(i,j)*z(4) ...

elseif (i==1) && (j==M) % upper right
```

```
            dAdt(i,j) = ...
                      - tauA(i,j)*A(i,j)*z(1) ...
                      - muA(i,j)*A(i,j)*z(3) ...
                      - alphaA(i,j)*A(i,j)*z(4) ...
                      + thetaA(i,j-1)*A(i,j-1)*z(2) ...
                      + gammaA(i+1,j)*A(i+1,j)*z(1) ...
                      + Ahat(i,j) - lambda*A(i,j);
                    %  + muA(i-1,j)*A(i-1,j)*z(3) ...
                      %   + tauA(i,j+1)*A(i,j+1)*z(1) ...
                   %   + alphaA(i,j+1)*A(i,j+1)*z(4) ...
                   %-gammaA(i,j)*A(i,j)*z(1) ...
                   %- thetaA(i,j)*A(i,j)*z(2) ...

        elseif (i==M) && (j==M) % lower right
            dAdt(i,j) = -gammaA(i,j)*A(i,j)*z(1) ...
                      - tauA(i,j)*A(i,j)*z(1) ...
                      - alphaA(i,j)*A(i,j)*z(4) ...
                      + muA(i-1,j)*A(i-1,j)*z(3) ...
                      + thetaA(i,j-1)*A(i,j-1)*z(2) ...
                      + Ahat(i,j) - lambda*A(i,j);
                       %  + gammaA(i+1,j)*A(i+1,j)*z(1) ...
                    %  + tauA(i,j+1)*A(i,j+1)*z(1) ...
                    %  + alphaA(i,j+1)*A(i,j+1)*z(4) ...
                     %- muA(i,j)*A(i,j)*z(3) ...
                        %- thetaA(i,j)*A(i,j)*z(2) ...

        elseif (i==M) && (j==1) % lower left
            dAdt(i,j) = -gammaA(i,j)*A(i,j)*z(1) ...
                      - thetaA(i,j)*A(i,j)*z(2) ...
                     + muA(i-1,j)*A(i-1,j)*z(3) ...
                      + tauA(i,j+1)*A(i,j+1)*z(1) ...
                      + alphaA(i,j+1)*A(i,j+1)*z(4) ...
                      + Ahat(i,j) - lambda*A(i,j)  ;
                     % + thetaA(i,j-1)*A(i,j-1)*z(2) ...
                     % + gammaA(i+1,j)*A(i+1,j)*z(1) ...
                     %- tauA(i,j)*A(i,j)*z(1) ...
                      %- muA(i,j)*A(i,j)*z(3) ...
                         %- alphaA(i,j)*A(i,j)*z(4) ...
         end
    end
end
%----------------------------------------------------------

%---------forms an M*M array representing dB/dt(i,j)---------
% Pretty much exact same as above but with B's instead of A's
for i=1:M
    for j= 1:M
        if (i>1) && (j>1) && (i<M) && (j<M) %in the middle everything possible
    dBdt(i,j) =-gammaB(i,j)*B(i,j)*z(1) ...
               - tauB(i,j)*B(i,j)*z(1) ...
               - muB(i,j)*B(i,j)*z(3) ...
               - thetaB(i,j)*B(i,j)*z(2) ...
               - alphaB(i,j)*B(i,j)*z(4) ...
               + muB(i-1,j)*B(i-1,j)*z(3) ...
               + thetaB(i,j-1)*B(i,j-1)*z(2) ...
               + gammaB(i+1,j)*B(i+1,j)*z(1) ...
               + tauB(i,j+1)*B(i,j+1)*z(1) ...
               + alphaB(i,j+1)*B(i,j+1)*z(4) ...
               + Bhat(i,j) -lambda*B(i,j);

elseif (i==1) && (1<j) && (j<M) %on the top side, No CE's to give up
    dBdt(i,j) = - tauB(i,j)*B(i,j)*z(1) ...
               - muB(i,j)*B(i,j)*z(3) ...
               - thetaB(i,j)*B(i,j)*z(2) ...
               - alphaB(i,j)*B(i,j)*z(4) ...
               -gammaB(i,j)*B(i,j)*z(1) ...
               + thetaB(i,j-1)*B(i,j-1)*z(2) ...
               + gammaB(i+1,j)*B(i+1,j)*z(1) ...
               + tauB(i,j+1)*B(i,j+1)*z(1) ...
               + alphaB(i,j+1)*B(i,j+1)*z(4) ...
               + Bhat(i,j) -lambda*B(i,j);
           % + muB(i-1,j)*B(i-1,j)*z(3) ...



elseif (1<i) &&(i<M)  && (j==1)% on left, No TE's to give up
    dBdt(i,j) =-gammaB(i,j)*B(i,j)*z(1) ...
               - muB(i,j)*B(i,j)*z(3) ...
               - thetaB(i,j)*B(i,j)*z(2) ...
               - tauB(i,j)*B(i,j)*z(1) ...
               + muB(i-1,j)*B(i-1,j)*z(3) ...
               + gammaB(i+1,j)*B(i+1,j)*z(1) ...
               + tauB(i,j+1)*B(i,j+1)*z(1) ...
               + alphaB(i,j+1)*B(i,j+1)*z(4)...
               + Bhat(i,j) -lambda*B(i,j);
           % + thetaB(i,j-1)*B(i,j-1)*z(2) ...

           %- alphaB(i,j)*B(i,j)*z(4) ...

elseif (i==M) && (1<j) && (j<M)% on bottom Full CE
    dBdt(i,j) = -gammaB(i,j)*B(i,j)*z(1) ...
               - tauB(i,j)*B(i,j)*z(1) ...
               - thetaB(i,j)*B(i,j)*z(2) ...
               - alphaB(i,j)*B(i,j)*z(4) ...
               + muB(i-1,j)*B(i-1,j)*z(3) ...
               + thetaB(i,j-1)*B(i,j-1)*z(2) ...
               + tauB(i,j+1)*B(i,j+1)*z(1) ...
               + alphaB(i,j+1)*B(i,j+1)*z(4) ...
               + Bhat(i,j) -lambda*B(i,j);
            %+ gammaB(i+1,j)*B(i+1,j)*z(1) ...
            %- muB(i,j)*B(i,j)*z(3) ...

elseif (1<i) && (i<M) && (j==M) % on right full of TG
    dBdt(i,j) = -gammaB(i,j)*B(i,j)*z(1) ...
               - tauB(i,j)*B(i,j)*z(1) ...
               - muB(i,j)*B(i,j)*z(3) ...
               - alphaB(i,j)*B(i,j)*z(4) ...
               + muB(i-1,j)*B(i-1,j)*z(3) ...
               + thetaB(i,j-1)*B(i,j-1)*z(2) ...
               + gammaB(i+1,j)*B(i+1,j)*z(1) ...
               + Bhat(i,j) -lambda*B(i,j);
               %  + tauB(i,j+1)*B(i,j+1)*z(1) ...
             %  + alphaB(i,j+1)*B(i,j+1)*z(4) ...
             % - thetaB(i,j)*B(i,j)*z(2) ...


elseif (i==1) && (j==1) %upper left
    dBdt(i,j) = - muB(i,j)*B(i,j)*z(3) ...
               - thetaB(i,j)*B(i,j)*z(2) ...
               + gammaB(i+1,j)*B(i+1,j)*z(1) ...
               + tauB(i,j+1)*B(i,j+1)*z(1) ...
               + alphaB(i,j+1)*B(i,j+1)*z(4) ...
               + Bhat(i,j) -lambda*B(i,j);
            % + muB(i-1,j)*B(i-1,j)*z(3) ...
              % + thetaB(i,j-1)*B(i,j-1)*z(2) ...
                %- alphaB(i,j)*B(i,j)*z(4) ...
                %- tauB(i,j)*B(i,j)*z(1) ...
                %-gammaB(i,j)*B(i,j)*z(1) ...

elseif (i==1) && (j==M) % upper right
    dBdt(i,j) = - tauB(i,j)*B(i,j)*z(1) ...
               - muB(i,j)*B(i,j)*z(3) ...
              - alphaB(i,j)*B(i,j)*z(4) ...
               + thetaB(i,j-1)*B(i,j-1)*z(2) ...
               + gammaB(i+1,j)*B(i+1,j)*z(1) ...
            + Bhat(i,j) -lambda*B(i,j);
            %  + muB(i-1,j)*B(i-1,j)*z(3) ...
            %   + tauB(i,j+1)*B(i,j+1)*z(1) ...
            %   + alphaB(i,j+1)*B(i,j+1)*z(4) ...
             %- thetaB(i,j)*B(i,j)*z(2) ...
             %-gammaB(i,j)*B(i,j)*z(1) ...

elseif (i==M) && (j==M) % lower right
    dBdt(i,j) = -gammaB(i,j)*B(i,j)*z(1) ...
```

```
                        - tauB(i,j)*B(i,j)*z(1) ...
                        - alphaB(i,j)*B(i,j)*z(4) ...
                        + muB(i-1,j)*B(i-1,j)*z(3) ...
                        + thetaB(i,j-1)*B(i,j-1)*z(2) ...
                        + Bhat(i,j) -lambda*B(i,j);
                        %  + gammaB(i+1,j)*B(i+1,j)*z(1) ...
                      %  + tauB(i,j+1)*B(i,j+1)*z(1) ...
                      %  + alphaB(i,j+1)*B(i,j+1)*z(4) ...
                         %- muB(i,j)*B(i,j)*z(3) ...
                        %- thetaB(i,j)*B(i,j)*z(2) ...

        elseif (i==M) && (j==1) % lower left
            dBdt(i,j) = -gammaB(i,j)*B(i,j)*z(1) ...
                       - thetaB(i,j)*B(i,j)*z(2) ...
                       + muB(i-1,j)*B(i-1,j)*z(3) ...
                        + tauB(i,j+1)*B(i,j+1)*z(1) ...
                        + alphaB(i,j+1)*B(i,j+1)*z(4) ...
                        + Bhat(i,j) -lambda*B(i,j);
                    % + thetaB(i,j-1)*B(i,j-1)*z(2) ...
                       % + gammaB(i+1,j)*B(i+1,j)*z(1) ...
                       %- alphaB(i,j)*B(i,j)*z(4) ...
                        %- muB(i,j)*B(i,j)*z(3) ...
                         %- tauB(i,j)*B(i,j)*z(1) ...
        end
    end
end
%------------------------------------------------------------


%-------------------forming dz-------------------
%after the creation of the first 4, and making 2 arrays
%I cram it all into one parameter, dz which is passed back
%to the odesolver.
dz(1) = dPdt; dz(2) = dTdt; dz(3) = dCdt; dz(4) = dHdt; dz(5:M^2+4)
= reshape(dAdt,M*M,1); dz((M)*(M)+5:2*M^2+4) = reshape(dBdt,M*M,1);
dz = dz'; %makes sure its a column vec
%---------------------------------------------------
```

References

[1] Anderson, C. Low density lipoprotein (LDL) heterogeneity. Ph. D. Thesis. Karolinska University Press. Stockholm, Sweden 2003.

[2] American Heart Association. http://www.americanheart.org/present.jhtml?identifier=512

[3] Demant, T.; Packard, C.; Demmelmair, H.; Stewart, P.; Bedynek, A.; Bedford, D.; Seidel, D.; Shepherd, J. Sensitive methods to study human apolipoprotein B metabolism using stable isotope-labeled amino acids. *The American Physiological Society (1996).*

[4] Packard, C. Triacylglycerol-rich lipoproteins and the generation of small, dense low density lipoprotein. *Biochemical Society Transactions*, Vol. 31 (2003), part 5.

[5] Packard, C.; Demant, T.; Stewart, J.; Bedford, D.; Caslake, M.; Schwertfeger, G.; Bedynek, A.; Shepherd, J.; Seidel, D. Apolipoprotein B metabolism and the distribution of VLDL and LDL subfractions. *Journal of Lipid Research*, Vol. 41 (2000), pp 305-317.

[6] Walldius, G.; Jungner, I. Rationale for using apolipoprotein B and apolipoprotein A-1 as indicators of cardiac risk and as targets for lipid-lowering therapy. *European Heart Journal*, Vol. 26 (2005), pp 210-212.