ALEMO: PROJECT

Bone Modeling

BY:-

SACHIN S SHETTY

Table of Contents

1.Objective 3

2. Tasks to Complete…….……..3

3. Appendix…….……...……13

1. Objective

Bone remodeling results from the competition between two cell populations in the bone: osteoclasts and osteoblasts. A mathematical model for bone remodeling can be formulated as a set of two ordinary differential equations:

$$\dot{y_1} = \alpha_1 y_1^{g_{11}} y_2^{g_{21}} - \beta_1 y_1$$
$$\dot{y_2} = \alpha_2 y_1^{g_{12}} y_2^{g_{22}} - \beta_2 y_2$$

where y_1 and y_2 are the normalized number of osteoclasts and the normalized number of osteoblasts; α_i and β_i are coefficients expressing the activities of cell production and removal; and parameters g_{ij} represent the net effectiveness of osteoclast- or osteoblast-derived autocrine or paracrine factors.

The goal of the project is to create a solver for the Komarova's model and use it to simulate random bone remodeling.

2. Tasks to Complete

T1: Complete Matlab's function KomarovaModel.m

The completed equations of Matlab function KomarovaModel.m are:

- $y_1 = y(1);$
- $y_2 = y(2);$
- $\dot{y}_1 = (a_1 * y_1{}^{g_{11}} * \dot{} y_2{}^{g_{21}}) - (b_1 * y_1);$
- $\dot{y}_2 = (a_2 * y_1{}^{g_{12}} * y_2{}^{g_{22}}) - (b_2 * y_2);$

T2: Complete Matlab's function KomarovaModel_Jac.m

The completed equations of Matlab function KomarovaModel_Jac.m are:

- $y_1 = y(1);$
- $y_2 = y(2);$
- $J_{11} = \left(a_1 * (y_2{}^{g_{21}}) * \left(g_{11} * \left(y_1{}^{(g_{11}-1)}\right)\right)\right) - b_1;$
- $J_{12} = \left(a_1 * (y_1{}^{g_{11}}) * \left(g_{21} * \left(y_2{}^{(g_{21}-1)}\right)\right)\right);$

- $J_{21} = \left(a_2 * (y_2{}^{g_{22}}) * \left(g_{12} * \left(y_1{}^{(g_{12}-1)}\right)\right)\right);$
- $J_{22} = \left(a_2 * (y_1{}^{g_{12}}) * \left(g_{22} * \left(y_2{}^{(g_{22}-1)}\right)\right)\right) - b_2;$

- $J = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}$

T3: Write backward Euler's formula for Komarova's equations in scalar form.

Backward Euler Scheme – Vector Form

$$\frac{\boldsymbol{y_{i+1} - y_i}}{\boldsymbol{\Delta t_i}} = \boldsymbol{f_{i+1}}$$

Then,

- $\frac{\boldsymbol{y_{1,i+1} - y_{1,i}}}{\boldsymbol{\Delta t_i}} = \boldsymbol{f_{1,i+1}}$
- $\frac{\boldsymbol{y_{2,i+1} - y_{2,i}}}{\boldsymbol{\Delta t_i}} = \boldsymbol{f_{2,i+1}}$

- $\frac{\boldsymbol{y_{1,i+1} - y_{1,i}}}{\boldsymbol{\Delta t_i}} = \left(\boldsymbol{a_1 y_{1,i+1}^{g_{11}} y_{2,i+1}^{g_{21}} - b_1 y_{1,i+1}}\right)$
- $\frac{\boldsymbol{y_{2,i+1} - y_{2,i}}}{\boldsymbol{\Delta t_i}} = \left(\boldsymbol{a_2 y_{1,i+1}^{g_{12}} y_{2,i+1}^{g_{22}} - b_1 y_{2,i+1}}\right)$

- $\boldsymbol{y_{1,i+1} - y_{1,i} = \Delta t_i\left(a_1 y_{1,i+1}^{g_{11}} y_{2,i+1}^{g_{21}} - b_1 y_{1,i+1}\right)}$
- $\boldsymbol{y_{2,i+1} - y_{2,i} = \Delta t_i\left(a_2 y_{1,i+1}^{g_{12}} y_{2,i+1}^{g_{22}} - b_1 y_{2,i+1}\right)}$

Therefore, Scalar form is,

- $\boldsymbol{y_{1,i+1} - y_{1,i} - \Delta t_i\left(a_1 y_{1,i+1}^{g_{11}} y_{2,i+1}^{g_{21}} - b_1 y_{1,i+1}\right) = 0}$
- $\boldsymbol{y_{2,i+1} - y_{2,i} - \Delta t_i\left(a_2 y_{1,i+1}^{g_{12}} y_{2,i+1}^{g_{22}} - b_1 y_{2,i+1}\right) = 0}$

T4: Show that y_{i+1} is the solution of g (z) = 0.

We have,

$$g(z) = z - y_i - \Delta t_i f(z)$$

Where, $z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$

And, we substitute $z = y_{i+1}$

That is,

$$g(y_{i+1}) = y_{i+1} - y_i - \Delta t_i f(y_{i+1})$$

We know that,

$$\frac{\boldsymbol{y_{i+1} - y_i}}{\boldsymbol{\Delta t_i}} = \boldsymbol{f(y_{i+1})}$$

Substituting in $g(y_{i+1})$,

We get

$$g(y_{i+1}) = y_{i+1} - y_i - \Delta t_i(\frac{y_{i+1} - y_i}{\Delta t_i})$$

On solving,

$$g(y_{i+1}) = 0$$

Therefore,

y_{i+1} is the root of $g(z) = 0$ and it is a solution of g(z)=0.

T5: Derive a mathematical expression for the gradient of g(z).

We have,

$$g(z) = z - y_i - \Delta t_i f(z)$$

And,

$$L(z) = \frac{\partial g}{\partial z}$$

Then,

$$\frac{\partial g}{\partial z} = I - 0 - \Delta t \frac{\partial f(z)}{\partial y}$$

$$\frac{\partial g}{\partial z} = I - \Delta t \frac{\partial f(z)}{\partial y}$$

We know that,

$$J(y) = \frac{\partial f}{\partial y} = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{bmatrix}$$

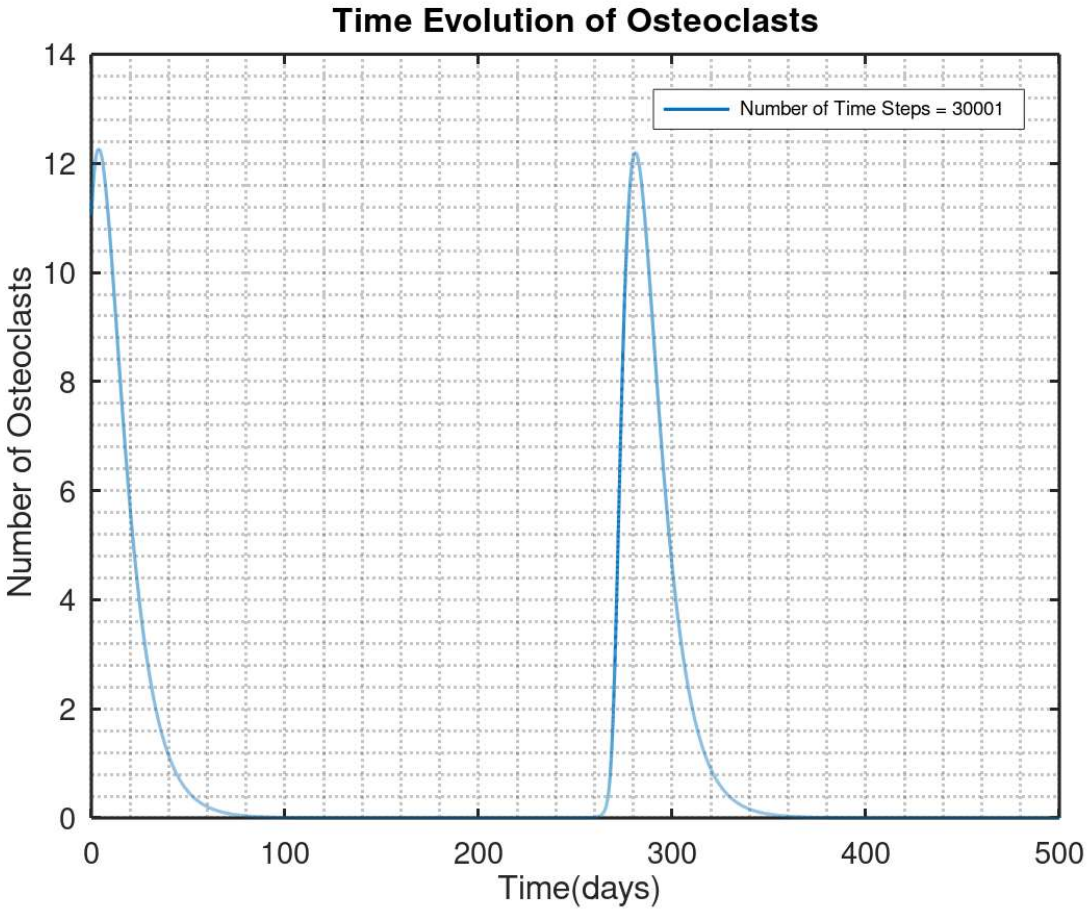
Then,

Substituting, the values of $\frac{\partial g}{\partial z}$ in $L(z)$,we get

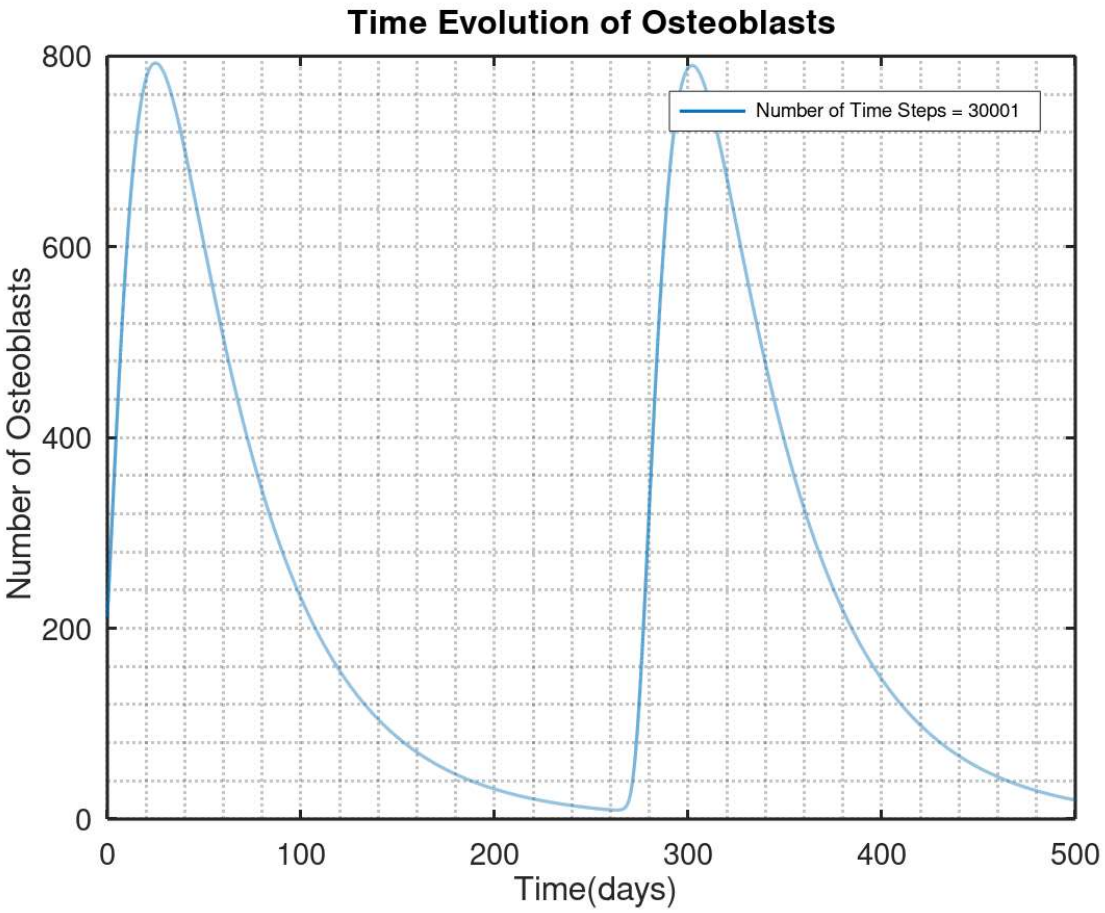
$$L(z) = I - \Delta t J(z)$$

T8: Test your code using the script testKomarova.m. Export some figures showing the time evolution of osteoclasts and osteoblasts in random remodeling.

These are the plots obtained when number of time steps (Nt) = 30001 showing the time evolution of Osteoclasts and Osteoblasts.

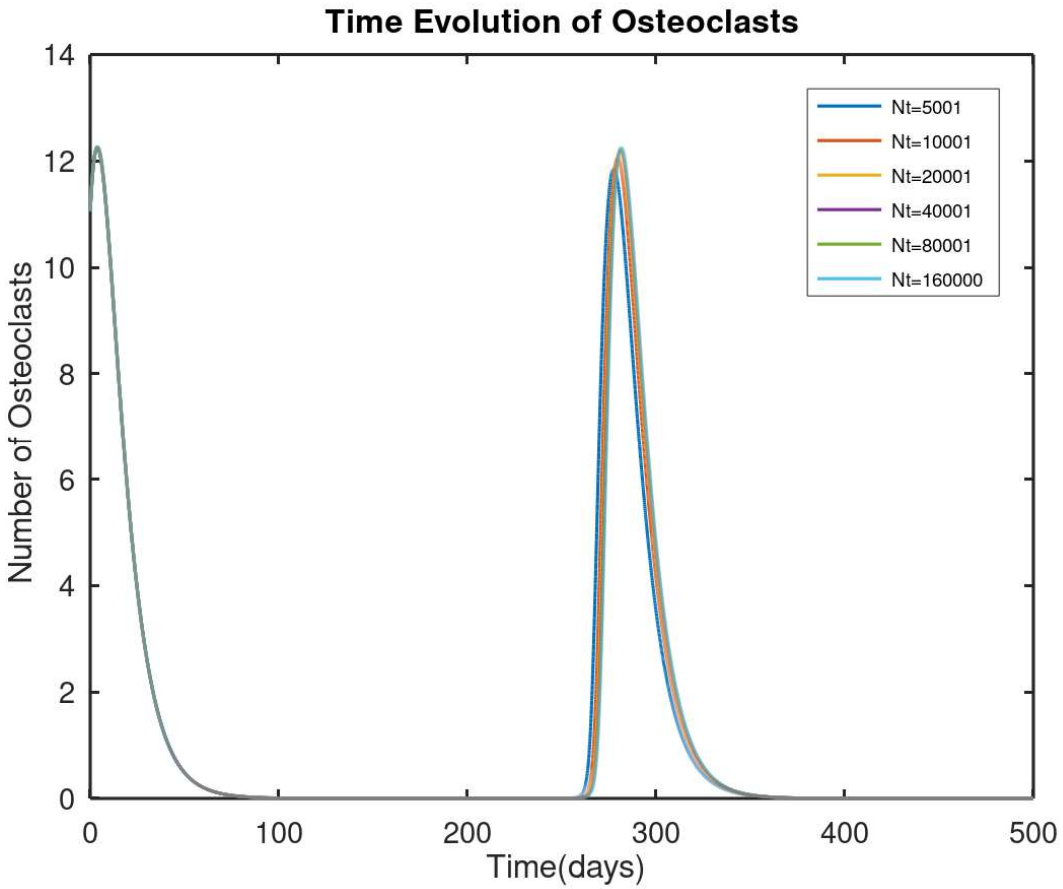


The above plot represents number of Osteoclasts versus Time (days) when the number of time steps Nt =30001.The rise to the maximum number of Osteoclasts is sudden and after reaching the maximum number of Osteoclasts the decline is also rapid.



The above plot represents number of Osteoblasts versus Time (days) when the number of time steps Nt =30001. The rise to the maximum number of Osteoblasts is sudden and after reaching the maximum number of Osteoclasts the decline is gradual.

T9: Repeat the numerical solution while increasing the number of time steps. Plot the solutions on the same graphs. Describe qualitatively what happens as the time step size decreases.



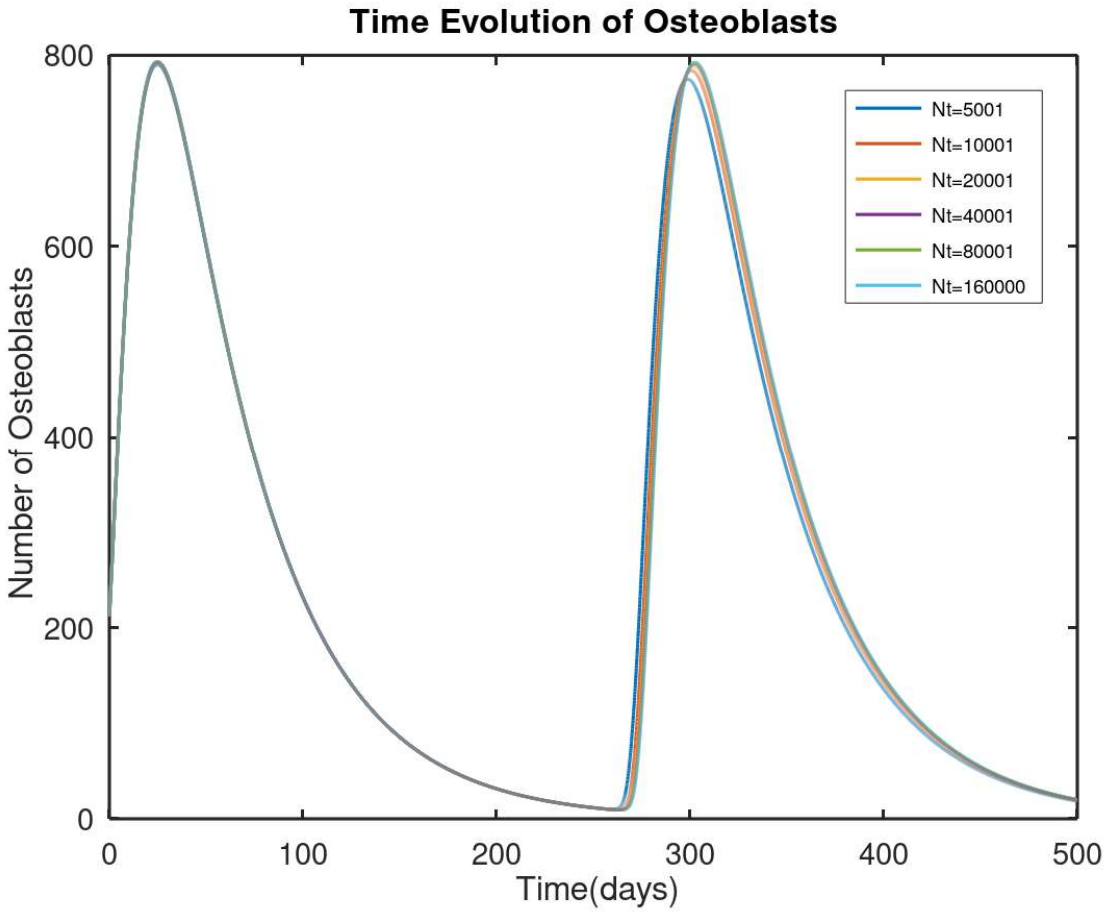
The above graph represents the time evolution of Osteoclasts versus time. The evolution of Osteoclasts is done for different number of time steps such as

Nt = [5000, 10000, 20000, 40000, 80000, 160000]

As seen in the graph, the solution approaches towards a more accurate value as the number of time steps increases.

When the number of days is less, the number of osteoclasts looks to be same for different time steps.

As you can see from the plot the difference between plots becomes less as Number of time steps reaches above 80001.



The above graph represents the time evolution of Osteoblasts versus time. The evolution of Osteoblasts is done for different number of time steps such as

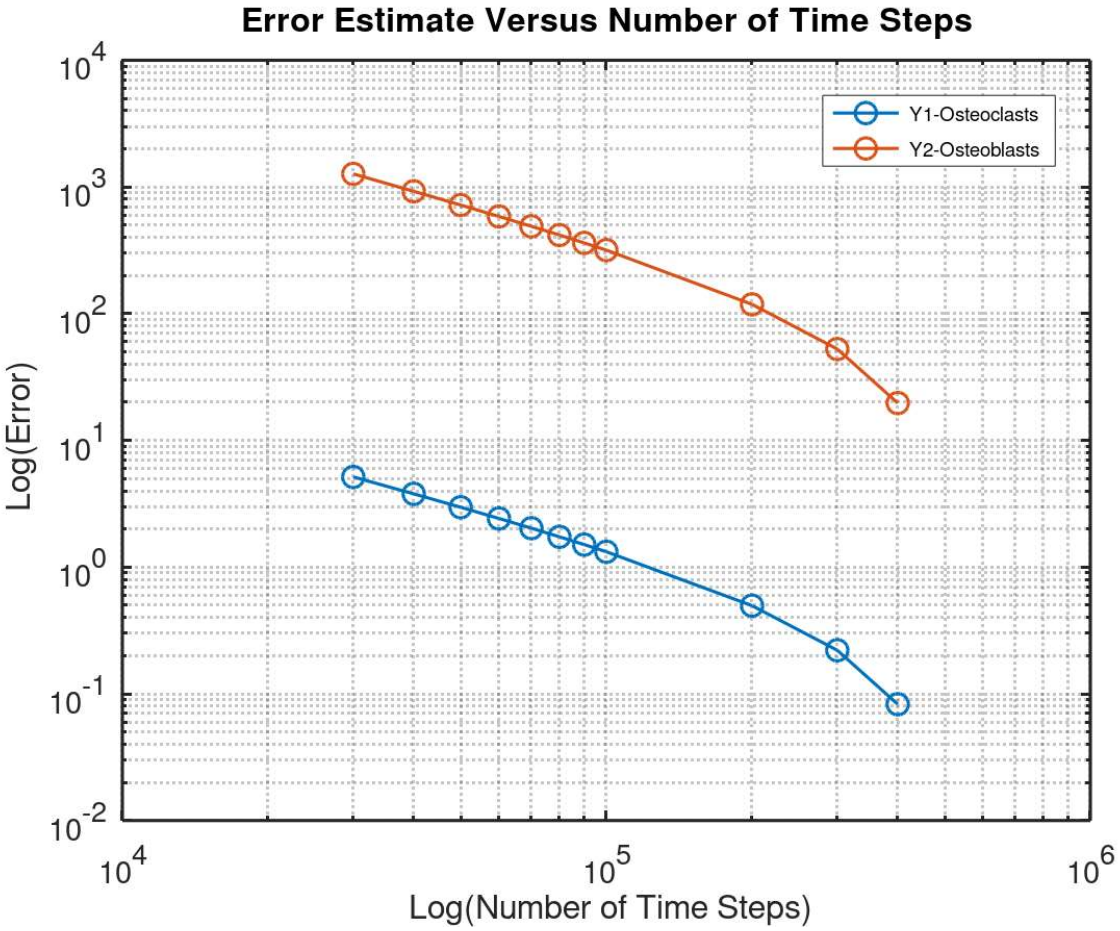
Nt = [5000, 10000, 20000, 40000, 80000, 160000]

As seen in the graph, the solution approaches towards a more accurate value as the number of time steps increases.

Similar to Osteoclasts,When the number of days is less, the number of osteoblasts looks to be same for different time steps.

As you can see from the plot the difference between plots becomes less as Number of time steps reaches above 80001.

T10: Plot the error estimate for y1 and y2 versus the number of time steps in double logarithmic scale. Use Matlab's function loglog.

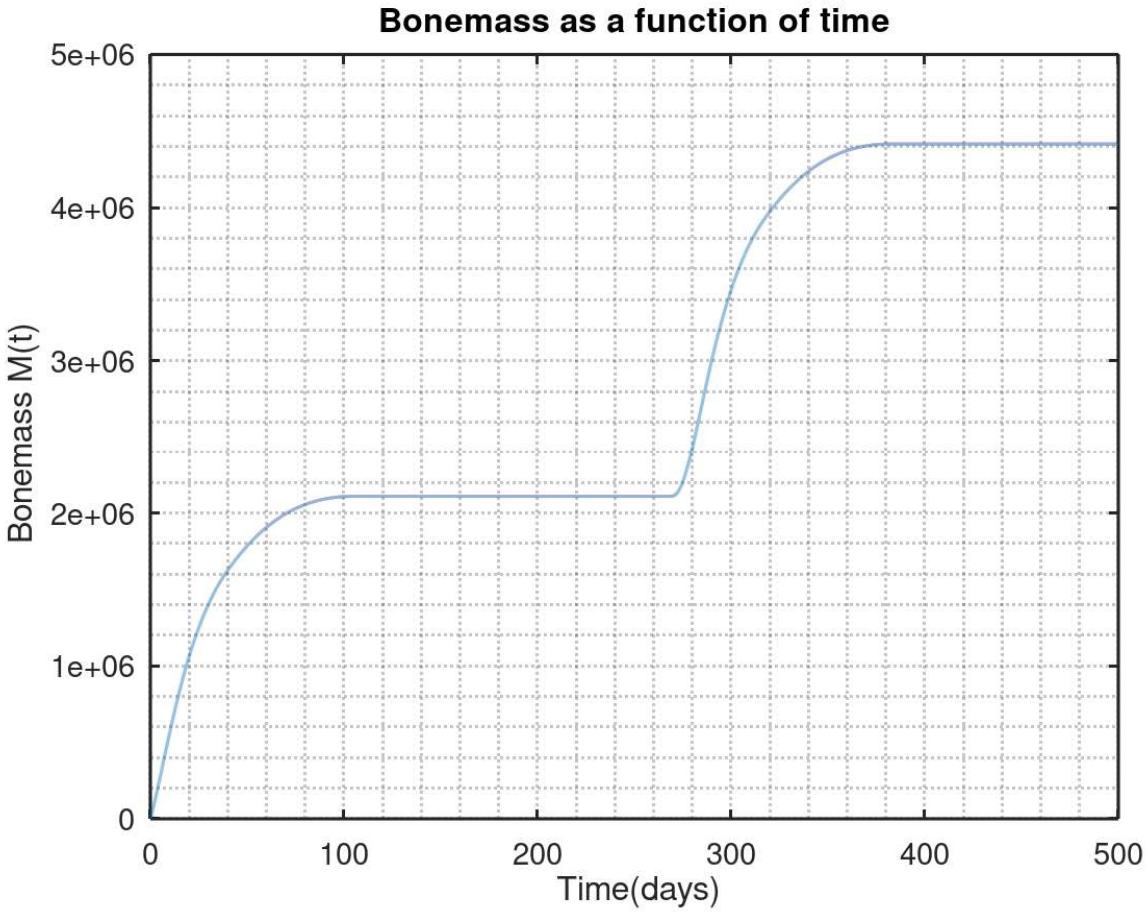


The above graph represents the loglog graph of Error versus Number of time steps. Here the exact solution of the system is calculated for a Time step of 0.5M.
The different time step choosen are:

Nt=[30000,40000,50000,60000,70000,80000,90000,100000,200000,300000,400000,500000]

As seen from the plot, as the number of time steps increases the error of decreases, for both Osteoclasts and Osteoblasts.It reaches to zero as it reaches to 0.5M time steps.

T11: Plot the bone mass M (t) as a function of time.



The number of time steps choosen is 80000 as the difference between plots above this value is less.The above plot represent Bone mass as a function of time, Where y-axis is represented by Bonemass M(t) and the x-axis is represented by the number of days. The bone mass increases in the above shown way as the number of days increase.

3. Appendix

- **Code: KomarovaModel**

```
function ydot = KomarovaModel(y,a1,a2,b1,b2,g11,g12,g21,g22)
%% DO NOT MODIFY THIS PART--------------------------------
%check size of the input vector
if numel(y) ~= 2
    error('The input vector Y must have two elements')
end
%%-------------------------------------------------------
y1 = y(1);
y2 = y(2);

ydot1 = (a1*(y1^g11)*(y2^g21))-(b1*y1);

ydot2 = (a2*(y1^g12)*(y2^g22))-(b2*y2);

%% DO NOT MODIFY THIS PART--------------------------------
%make sure that output argument ydot has the same
ydot = reshape([ydot1,ydot2],size(y));
%%-------------------------------------------------------
end
```

- **Code: KomarovaModel_Jac**

```
function J = KomarovaModel_Jac(y,a1,a2,b1,b2,g11,g12,g21,g22)
%% DO NOT MODIFY THIS PART--------------------------------
%check size of the input vector
if numel(y) ~= 2
    error('The input vector Y must have two elements')
end
%%-------------------------------------------------------
y1 = y(1);
y2 = y(2);

% Jij is the derivative of ydot_i w.r.t. y_j

J11 = (a1*(y2^g21)*(g11*(y1^(g11-1))))-b1;
J12 = (a1*(y1^g11)*(g21*(y2^(g21-1))));
J21 = (a2*(y2^g22)*(g12*(y1^(g12-1))));
J22 = (a2*(y1^g12)*(g22*(y2^(g22-1))))-b2;

% J is a 2x2 matrix containing the J_(ij)'s
J = [J11 J12;J21 J22];

end
```

- **Code: BwdEuler**

```
function y = BwdEuler(time,ffun,Jfun,y1)
% SOLVE ydot = f(t,y)
% t   is the time stepping vector
% ffun is the function handle @(t,y) for the rhs of the ODE f(t,y)
% Jfun is the function handle @(t,y) for the gradient of f with respect to
% y
% y1 is the initial condition
%% DO NOT MODIFY THIS PART--------------------------------
if size(y1,2)~=1
    error('The initial condition must be a COLUMN vector')
end
if min(size(time))~=1
    error('The time step vector must be either a COLUMN or a ROW')
end
%%-------------------------------------------------------
% Allocate space for solution
y = zeros(size(y1,1),numel(time));
% Set initial condition
y(:,1)=y1;

% Start time stepping
for i = 2:((numel(time)))
% compute Delta t
dt = time(i)-time(i-1);
% set g fun
gfun = @(z)(z-y(:,i-1)-dt*ffun(time(i),z));
% Set L
Lfun = @(z)(eye(2)-dt*Jfun(time(i),z));
% solve nonlinear problem using the solution at previous time step as
% initial guess
y(:,i) = solveNR(gfun,Lfun,y(:,i-1));
%Display message
fprintf('Solved time step %d of %d\n',i,numel(time));
end
```

- **Code: SolveNR**

```
function znew = solveNR(gfun,Lfun,z1)

err = 1;
k = 0;
znew = z1;

while (err>1e-10)&&(k<=10000)
%STORE the old solution
zold = znew;
%Increment iteration index k
k = k+1;
%COMPUTE THE FUNCTION
g = gfun(zold);
%COMPUTE THE GRADIENT
L = Lfun(zold);
%UPDATE Z
znew =(zold)-(inv(L)*g);

%ERROR ESTIMATION
err = max(norm(g),norm(zold-znew));
end

if err>1e-10
    error('Newton Raphson did not converge! Try increasing the error tolerance or the number of iterations!')
end

end
```

- **Code: testKomarova**

```
function [y,time]=testKomarova(Nt)
#clear all
#close all
#clc
%% MODEL PARAMETERS
a1= 3;
a2= 4;
b1= 0.2;
b2= 0.02;
g11= 1.1;
g12= 1;
g21=-0.5;
g22= 0;
k1= 0.18;
k2= 0.0014;
BMi = 92.123644277624891; % initial bone mass
%% INITIAL CONDITIONS
yss = [1.060660171777250 ; 2.121320343560527e+02];
y1 = yss + [10;0];

%% NUMERICAL PARAMETERS
#Nt = 1600;
time = linspace(0,500,Nt+1);

%% HANDLE FUNCTIONS DEFINITION
ffun = @(t,y)KomarovaModel(y,a1,a2,b1,b2,g11,g12,g21,g22);
Jfun = @(t,y)KomarovaModel_Jac(y,a1,a2,b1,b2,g11,g12,g21,g22);
% since all the parameters are fixed fun only depends on t and
%% SOLUTION
tic
y = BwdEuler(time,ffun,Jfun,y1);
toc

```

- **Code: Random Remodeling (Nt=30000)**

```
#To program the Time evolution of Osteoclasts and Osteoblasts in Ranc

Nt=[30000];

for i=1:(numel(Nt))

#Function call for testKomarova.m
[y,time]=testKomarova(Nt(i));
y1=y(1:1,:);
y2=y(2:2,:);

#To plot Osteoclasts
figure
plot(time,y1,'-','MarkerSize',10,'LineWidth',1.5);
grid minor
xlabel(' Time(days)')
ylabel(' Number of Osteoclasts')
title('Time Evolution of Osteoclasts')
legend('Number of Time Steps = 30001')
set(gca,'FontSize',14)
set(gca,'LineWidth',1.5)

#To plot Osteoblasts
figure
plot(time,y2,'-','MarkerSize',10,'LineWidth',1.5);
grid minor
xlabel(' Time(days)')
ylabel(' Number of Osteoblasts')
```

- **Code: Time evolution of Osteoclasts and Osteoblasts for Different Time Steps**

```
#To program the Time evolution of Osteoclasts and Osteoblasts for Different Time Steps

Nt=[5000,10000,20000,40000,80000,160000];
for i=1:(numel(Nt))

#Function call for testKomarova.m
[y,time]=testKomarova(Nt(i));
y1=y(1:1,:);
y2=y(2:2,:);

#To plot Osteoclasts

plot(time,y1,'-','MarkerSize',10,'LineWidth',1.5);
hold on
grid minor
xlabel(' Time(days)')
ylabel(' Number of Osteoclasts')
title('Time Evolution of Osteoclasts')
legend('Nt=5001','Nt=10001','Nt=20001','Nt=40001','Nt=80001','Nt=160000')
set(gca,'FontSize',14)
set(gca,'LineWidth',1.5)

#To plot Osteoblasts

plot(time,y2,'-','MarkerSize',10,'LineWidth',1.5);
hold on
grid minor
xlabel(' Time(days)')
ylabel(' Number of Osteoblasts')
title('Time Evolution of Osteoblasts')
legend('Nt=5001','Nt=10001','Nt=20001','Nt=40001','Nt=80001','Nt=160000')
set(gca,'FontSize',14)
set(gca,'LineWidth',1.5)

end
```

- **Code: Plot the error estimate for y1 and y2 versus the number of time steps in double logarithmic scale. Use Matlab's function loglog.**

```
#Plot the error estimate for y1 and y2 versus the number of time steps
load Y5L_t5L
Exact_y1=y(1:1,:);
Exact_y2=y(2:2,:);
time_Exact=linspace(0,500,500001);
Nt=[30000,40000,50000,60000,70000,80000,90000,100000,200000,300000,400000,500000]
Err_y=zeros(1,numel(Nt))
Err_y1=zeros(1,numel(Nt))
Err_y2=zeros(1,numel(Nt))
time=zeros(1,numel(Nt))
for i=1:(numel(Nt))
[Aprox_y,Aprox_time]=testKomarova(Nt(i));
Aprox_y1=Aprox_y(1:1,:);
Aprox_y2=Aprox_y(2:2,:);
y1_intrp=interp1(Aprox_time,Aprox_y1,time_Exact);
norm(Exact_y1)
norm(y1_intrp)
y2_intrp=interp1(Aprox_time,Aprox_y2,time_Exact);
norm(Exact_y2)
norm(y2_intrp)
y_intrp=[y1_intrp;y2_intrp];
Err_y(i)=norm(y_intrp)-norm(y)
Err_y1(i)=norm(Exact_y1)-norm(y1_intrp)
Err_y2(i)=norm(y2_intrp)-norm(Exact_y2)
end
figure()
loglog(Nt,Err_y1,'-o','MarkerSize',10,'LineWidth',1.5)
hold on
loglog(Nt,Err_y2,'-o','MarkerSize',10,'LineWidth',1.5)
grid minor
xlabel(' Log(Number of Time Steps)')
ylabel(' Log(Error)')
title('Error Estimate Versus Number of Time Steps')
legend('Y1-Osteoclasts','Y2-Osteoblasts')
set(gca,'FontSize',14)
```

- **Code: Plot the bone mass M (t) as a function of time.**

```
#Program to Plot Bonemass as a function of time
Nt=[80000];
k1= 0.18;k2= 0.0014;

yss = [1.060660171777250 ; 2.121320343560527e+02];
yss1=yss(1:1,:);yss2=yss(2:2,:);

[Final_y,Final_time]=testKomarova(Nt);
y1=Final_y(1:1,:);
y2=Final_y(2:2,:);

x1=max(y1-yss1,0);
x2=max(y2-yss2,0);

dt=Nt/500;

m(1)=92.123644277624891;

for i = 2:((numel(Final_time)))

   m(i)=m(i-1)+(0.5*dt*((k1*x1(:,i))+(k2*x2(:,i))+ (k1*x1(:,i-1))+(k2*x2(:,i-1))));

end
plot(Final_time,m,'-','MarkerSize',10,'LineWidth',1.5);
grid minor
xlabel(' Time(days)')
ylabel(' Bonemass M(t)')
title('Bonemass as a function of time')
set(gca,'FontSize',14)
set(gca,'LineWidth',1.5)
```