ALGORITHIMICS FOR ENGINEERING MODELLING (ALEMO)

Numerical Algorithm for Bone Remodeling using MATLAB/Octave

By

Yusuf Akande SALAUDEEN

M1 – Computational Mechanics

December 2020

1.0 Project Overview

Bone remodeling is a term that involves the removal of bone tissues in a skeleton to enable the formation of new bones. It involves two main processes of bone resorption and bone formation. Bone resorption begins when a specific portion of the bone is removed by osteoclasts, which is to be replaced later by osteoblasts activity. For bone formation, osteoblasts provide necessary collagen and mineral deposits over the area of the bone previously removed by osteoclasts. These two activities usually occur at random places in the cell, called random or general remodeling. They can as well target a specific area due to necessity, called target remodeling. This project was carried out using a journal titled "Mathematical model predicts a critical role for osteoclast autocrine regulation in the control of bone remodeling," referred to as the Komorova model with the name of one of its co-authors.

2.0 Objectives

The objectives of the project include, but not limited to:

i. to understand the use of a numerical process for practical problems
ii. to create a solver for the Komorova model and use it to simulate random bone remodeling
iii. to understand the flow of an algorithmic process

3.0 Model Formulation (Governing Equations)

The project was modeled using the following mathematical relations:

$$\dot{y}_1 = \alpha_1 y_1^{g_{11}} y_2^{g_{21}} - \beta_1 y_1$$
$$\dot{y}_2 = \alpha_2 y_1^{g_{12}} y_2^{g_{22}} - \beta_2 y_2$$

where

y_1 and y_2 are the normalized numbers of osteoclasts and the normalized number of osteoblasts;

α_i and β_i represents the coefficients expressing the activities of cell production and removal;

g_{ij} describe the net effectiveness of osteoclast or osteoblast derived autocrine or paracrine factors.

4.0 Project Requirements

1. The time step integration required a numerical scheme such as the Backward Euler method for random bone remodeling.
2. It involved solving a non-linear system of two equations in the two unknowns y_1 and y_2 for each new time step using the Newton-Raphson approach.
3. The evolution of y_1 and y_2 was then to be interpolated on a finer mesh, using *interp1* for piecewise linear interpolation to perform convergence study.
4. Finally, the bone density was computed from the following integral formula involving both variables y_1 and y_2, which had to be applied at all time steps to obtain a result with a time series describing the time evolution of the bone density.

$$M(t) = \int_0^t k_1 x_1(\tau) + k_2 x_2(\tau) d\tau$$

In this integral formula, the trapezoidal rule was used for t in the time-stepping vector. x_1 and x_2 are defined as:

$$x_i(\tau) = max(y_i - y_{i_{ss}}, 0) \quad i = 1, 2$$

5.0 Defining the Komorova Model In MATLAB

The model formulation was expressed in a vector form for ease of manipulation in MATLAB (R2019a) as follows:

$$\mathbf{f}(\mathbf{y}) = \begin{bmatrix} a_1 y_1^{g_{11}} y_2^{g_{21}} - b_1 y_1 \\ a_2 y_1^{g_{12}} y_2^{g_{22}} - b_2 y_2 \end{bmatrix}$$

in shorthand notation, $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$

and the normalized osteoblasts and osteoclasts expressed as a column vector as given below:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

A function of f(y) was completed in MATLAB script named KomorovaModel.m as follows:

```
ydot1 = a1*y1^g11*y2^g21 - b1*y1;
ydot2 = a2*y1^g12*y2^g22 - b2*y2;
```

The values of y1 and y2 have been defined, as initial conditions, in another vector named y_1 in testKomorova.m as a function of yss.

```
%% INITIAL CONDITIONS
yss = [1.060660171777250 ; 2.121320343560527e+02];
y1 = yss + [10;0];
```

With initial conditions, y1 and y2, nicely subsequently defined in the KomorovaModel.m as:

```
y1 = y(1);
y2 = y(2);
```

5.1 The Jacobian of the model

Since the algorithm involves using the Newton-Raphson approach to solve two non-linear equations, it was necessary to calculate the Jacobian of the model, which is expressed as the gradient of the original model vector equation, f(y). Simply put, the Jacobian is the derivative of f(y) with respect to components y_1 and y_2.

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

This was completed in the MATLAB function file named KomorovaModel_Jac.m :

```
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; %%<---
```

6.0 The Numerical Approach

6.1 Backward Euler Scheme

The Backward Euler approach was adopted as one of the numerical methods to evaluate the model. This approach is an implicit method; it provides a solution to an ordinary differential equation at the current time step and a later one by using the result of the previous step. Thus, if a function is a non-linear function of its arguments, each time step would require an iterative procedure.

Using this scheme on the model vectors, we have:

$$\frac{\mathbf{y}_{i+1} - \mathbf{y}_i}{\Delta t_i} = \mathbf{f}_{i+1}$$

and in scalar form:

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

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

where

$\Delta t_i = t_{i+1} - t_i$ defines the time step size for N_t time steps.

For N_t number of time steps, the time-stepping vector is N_{t+1}

$$t = [t_1, t_2, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_{N_t}, t_{N_t+1}]$$

Note that this approach means iterating the result at t_i to get the result of t_{i+1}; it is also equivalent to iterating the result at t_{i-1} to obtain the result at time step t_i. The second approach was adopted in this exercise with the code snippet given in Appendix A.

6.2 Proof that the solution of g(z) as y_{i+1}

Let's denote vector **z** as the value of the solution of the model at time step t_{i+1}, which is y_{i+1}; the Backward Euler scheme above can be written as a function of z as **g(z)**:

$$\mathbf{g}(\mathbf{z}) = \mathbf{z} - \mathbf{y}_i - \Delta t_i \mathbf{f}(\mathbf{z})$$

when g(z) = 0, the equation simplifies to:

$$\mathbf{y}_{i+1} - \mathbf{y}_i - \Delta t_i \, \mathbf{f}_{i+1} = 0$$

By identification of the two equations, the root of the equation is: $\mathbf{y}_{i+1}$

6.3 Newton – Raphson approach

I adopted the Newton-Raphson procedure to compute the values of the osteoblasts and osteoclasts at different time steps. This method is efficient for finding the roots of non-linear functions, as in the Komorova Model. This approach's main idea is to iterate the solution at point k to obtain the result at point k+1 by approximating the function f(x) by its tangent. This concept is presented in solveNR.m, as given in Appendix B.

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$$

By expressing the scheme as a function of vector **z** to denote the solution at time step t_{i+1}, Newton's iteration becomes:

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} - \mathbf{L}^{-1}(\mathbf{z}^{(k)})\mathbf{g}(\mathbf{z}^{(k)})$$

with an initial guess: $\mathbf{z}^{(1)} = \mathbf{y}_i$

Note that vector **L** is the gradient of **g(z),** which can also be expressed as

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

6.4 Proof of the relationship and L and the Jacobian

From the expression of g(z) as

$$\mathbf{g}(\mathbf{z}) = \mathbf{z} - \mathbf{y}_i - \Delta t_i \mathbf{f}(\mathbf{z})$$

$$z_1 - y_{1,i} - \Delta t_i \left(a_1 y_{1,i+1}^{g_{11}} y_{2,i+1}^{g_{21}} - b_1 y_{1,i+1} \right) = 0$$

$$z_2 - y_{2,i} - \Delta t_i \left(a_2 y_{1,i+1}^{g_{12}} y_{2,i+1}^{g_{22}} - b_1 y_{2,i+1} \right) = 0$$

Differentiating the above expression partially with respect to **z,** which implies, y_{i+1} gives:

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

with **I** being an identity matrix.

7.0 Testing the Algorithm

The code was tested against the model parameters given in testKmorova.m, presented in Appendix C, to depict the evolution of osteoclasts and osteoblasts with time in random modeling. The first test was done with a hundred (100) number of time steps and then compared with an increased five hundred steps (500).

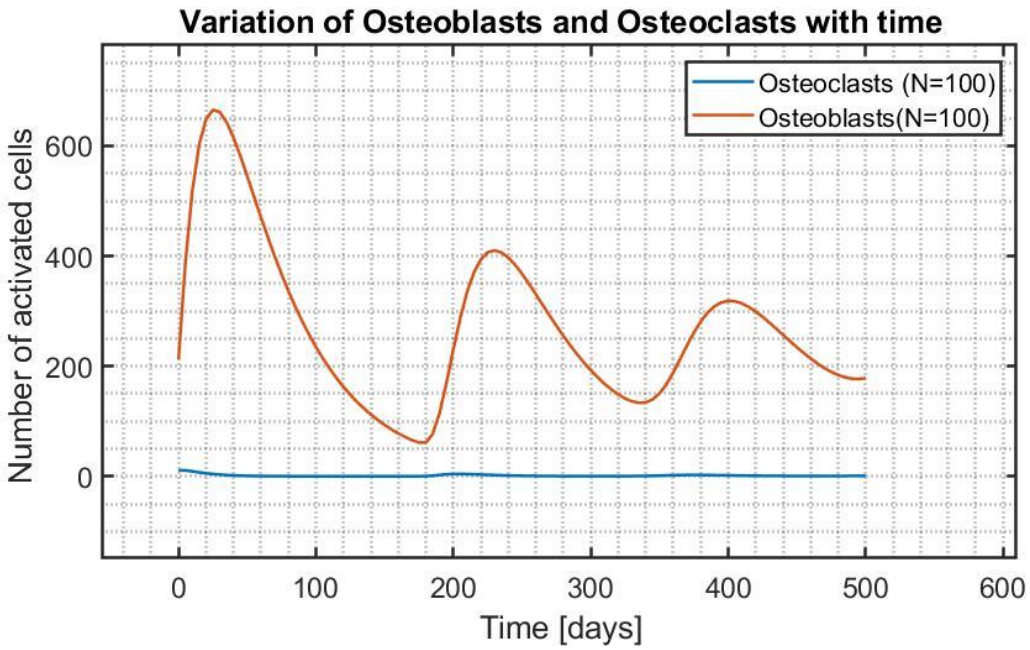

Fig 1: Evolution of osteoclasts and osteoblast with 100 number of time steps

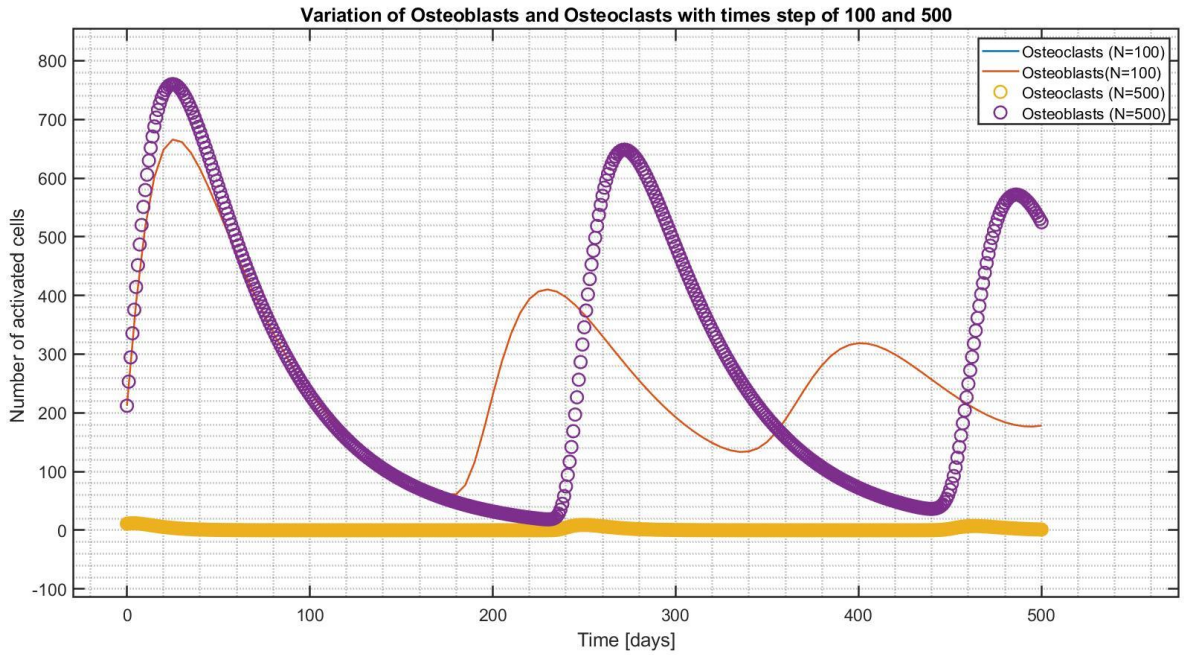

Fig 2: Comparison between evolution of osteoclasts and osteoblast with 100 and 500 number of time steps

From the models, it is observed that the Backward Euler scheme is an **unconditionally stable approach**, as small changes in the initial data only result in small changes in the solution of the differential equations.

As expected, the accuracy of the scheme decreases with a decrease in the number of time steps. For instance, the following represents the evolution of osteoclasts and osteoblasts for twenty (20) nu**mber of times steps:**

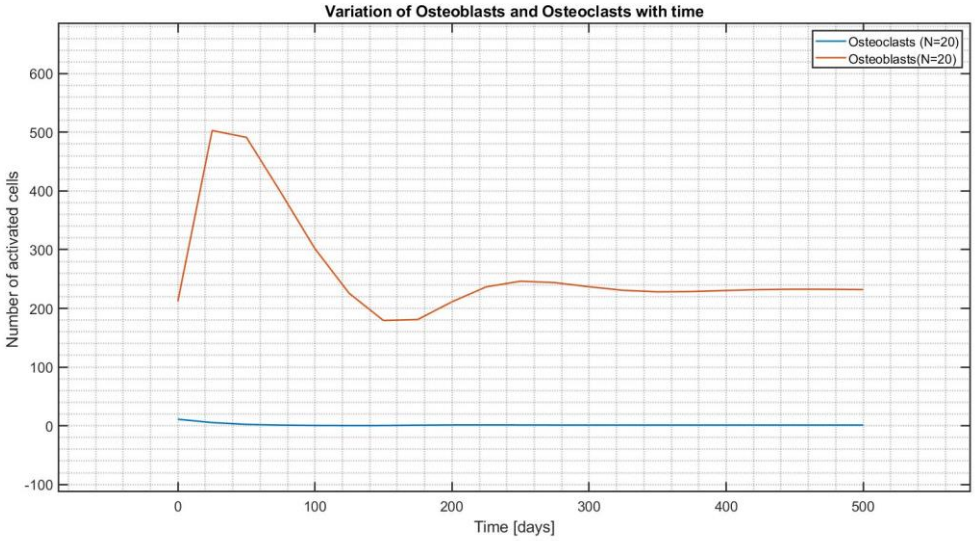

Fig 3: Evolution of osteoclasts and osteoblast with 20 number of time steps

8.0 Convergence Study

A convergence study is usually performed to determine the proximity of the approximated solution to the exact solution. The convergence criterion depends on the nature of the experiment, the degree of accuracy expected, and the choice of the algorithm, among others. In that wise, the approximation error between the reference solution and any other solution is to be determined by using the norm of the difference between these solutions.

For this model, the number of time steps was increased to 0.5 million points in order to understand the nature of the model. This half-a-million time steps is one that can be presumed to present the 'ground truth,' close to the exact evolution. This was done firstly by maintaining the time step size as 5000, and then as 500 000 to observe the 'true' evolution of both osteoblasts and osteoclasts.

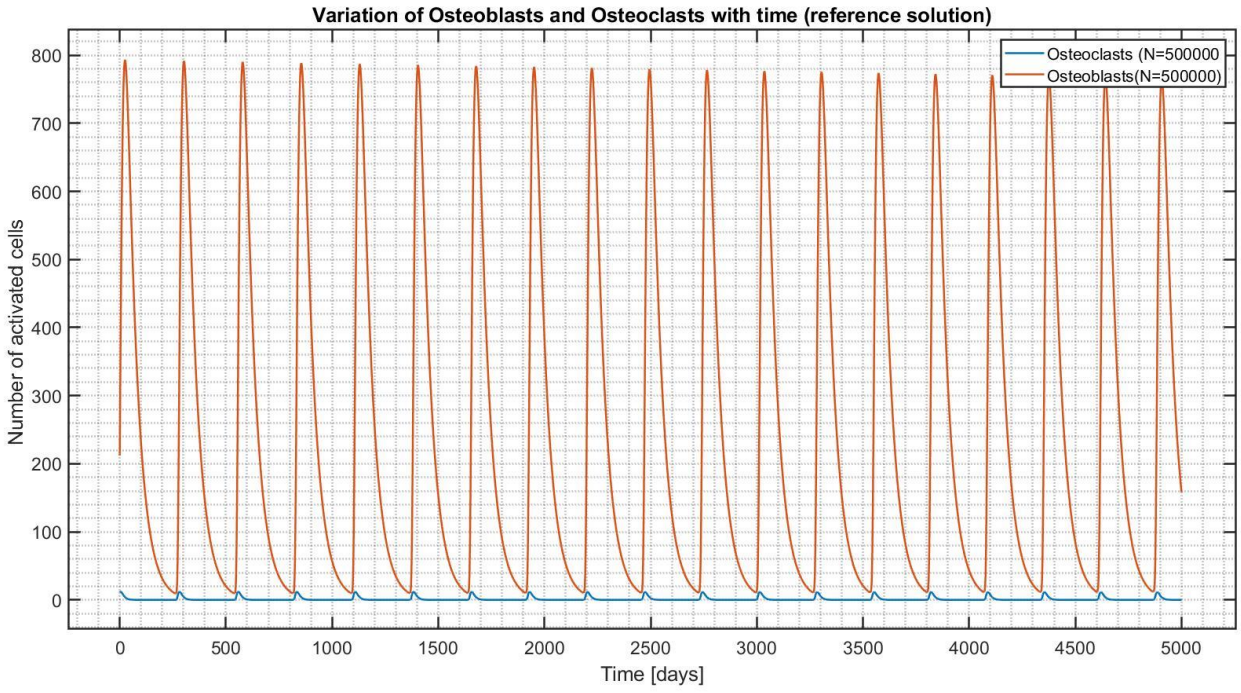

Fig 4: Evolution at 500000 times steps and 5000-time step size

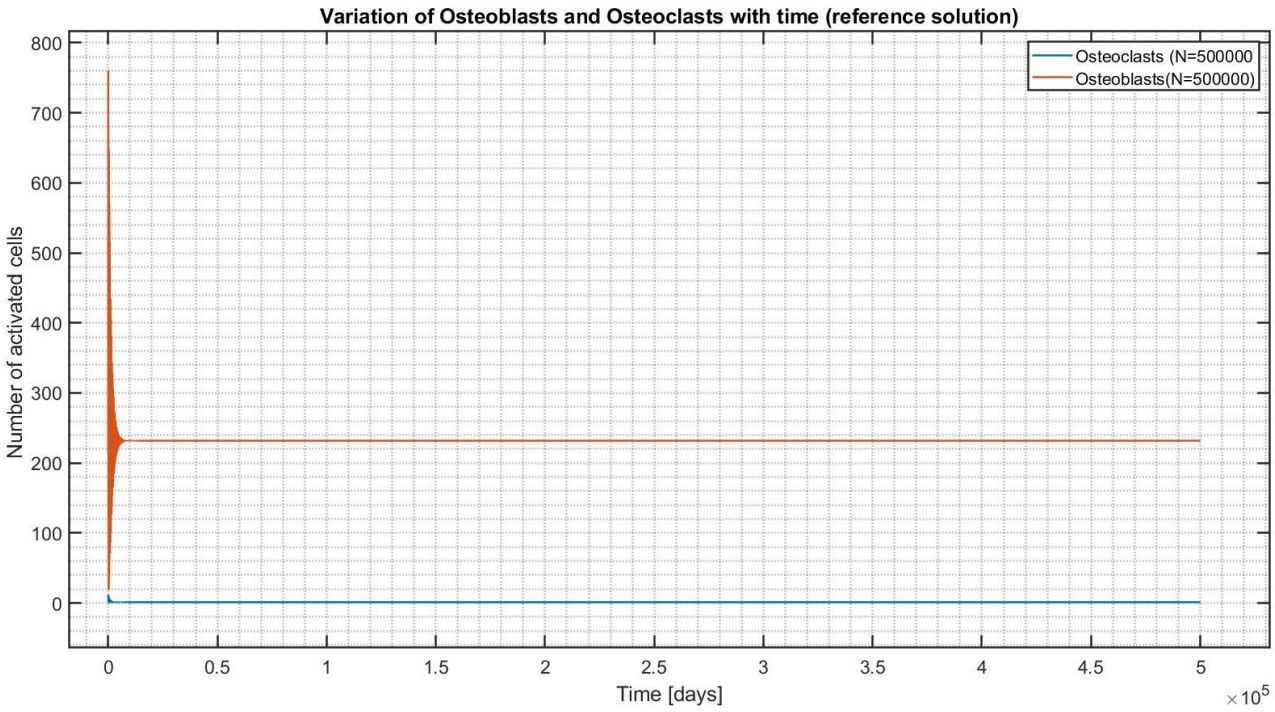

Fig 5: "Reference solution"

Note that when the number of time step size was sufficiently increased, the time step value tends to zero, and the model was solved at almost every instance of time. This dramatically reduced the approximation error compared to when the time step size was larger, corresponding to less number of time steps.

To perform this convergence study, another set of solutions has to be performed for fewer times steps. If the reference solution is denoted y1 and the other solution as y2, we find the interpolation of these two solutions at a series of points based on our interest. Interpolation can be done in several ways, either using user-defined functions or using Matlab inbuilt functions. In this case, `interp1`, which is a Matlab inbuilt function, was used.

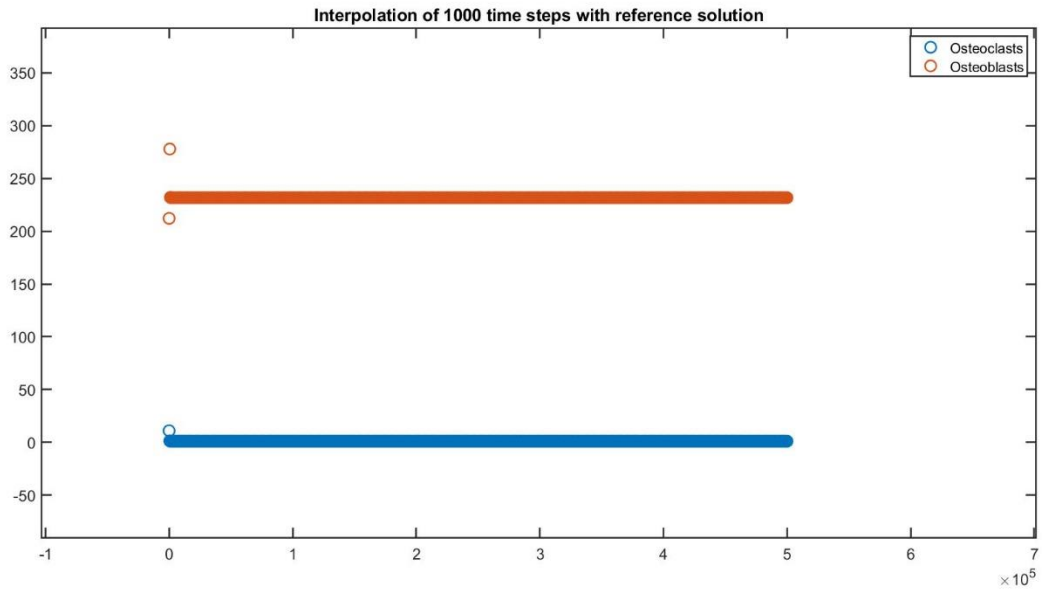

Fig 6: Interpolation Curve

8.1 Error calculation

As previously highlighted, the norm of the difference between the 'almost' exact solution (the reference solution) and any other two solutions gives the error – represented by a log-log plot, as presented in Appendix D. For this, the time steps of one of the solutions doubled that of the other. The constant variation of the error further indicates that the Backward Euler scheme is an unconditionally scheme.

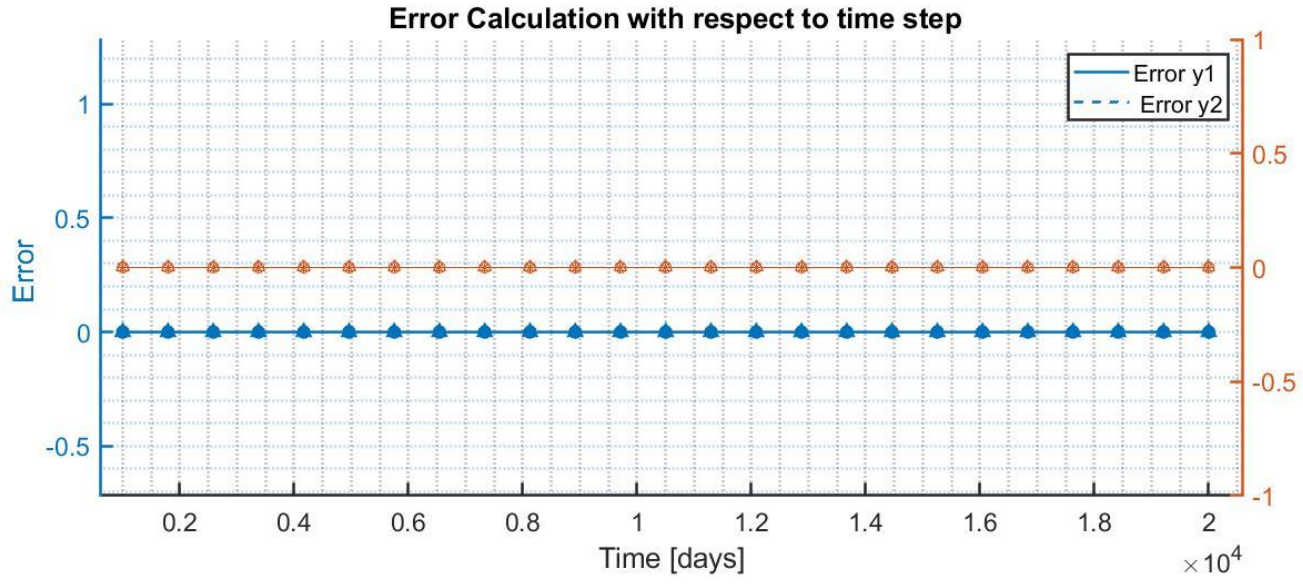

Fig 7: Error Plot (LogLog Graph)

9.0 Bone Mass Density

Based on the error analysis above, the bone mass density for the modeled can be calculated by using the trapezoidal rule as follows:

$$M(t) = \int_0^t k_1 x_1(\tau) + k_2 x_2(\tau) d\tau$$

The values of x_i are defined as:

$$x_i(\tau) = max(y_i - y_{i_{ss}}, 0) \quad i = 1, 2$$

The initial value of the bone mass was given as M = 92.123644277624891 and other values specified in the testKomorva.m .

APPENDIX

APPENDIX A: Backward Euler Scheme

```
% Allocate space for solution
y = zeros(size(y1,1),numel(t));

% Set initial condition
y(:,1)= y1(:);%%<----------------------

% Start time stepping
for i = 2:numel(t)
% compute Delta t
% trying to find the solution at ith step step, using the solution of
% previous time step i - 1
dt = t(i)-t(i-1); %%<----------------------
% set g fun

gfun = @(z) (z - y(:, i-1) - dt*ffun(t(i),z)); %%<----------------------

% Set L
Lfun = @(z) (eye(2) - dt*Jfun(t(i), z)); %%<----------------------
```

APPENDIX B: Newton-Raphson Approach

```
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-(L\g); %%<----------------------;

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

if err>1e-10
    error('Newton Raphson did not converge! Tr
end

end
```

APPENDIX C: Model Parameters testKomorova.m

```
%% 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
```

APPENDIX D: Convergence study (Error calculation)

```
m=50;
error1 = zeros(m);
error2= zeros(m);
we = zeros();
Nt = 0;
for i = 1:m
    Nt = Nt+1000;
    time = linspace(0,500,Nt+1);
    t = linspace(0,500, Nt+1);
    ya= BwdEuler(time,ffun,Jfun,y1);
    ya1= ya(1,:);
    ya2= ya(2,:);
    yinterp1 = interp1(time,ya1,t);
    yinterp2 = interp1(time,ya2,t);
    error1(i)= norm(a-yinterp1);
    error2(i) = norm(b-yinterp2); %
    we(i) = i;
end
time21 = linspace(1000,20000,m);
figure()
title("error")
hold on
yyaxis right
loglog(time21, error1);
yyaxis left
loglog(time21,error2,'LineWidth',1.5)
grid minor
```