```
%Windkessel_Main

%this function uses previously saved impedance data and initial conditions
%from an excel sheet to run the middle cycle for three-element Windkessel
%model and four-element Windkessel models


clear
close all
clc

global P_data t_total R0 t_span P0 Q

tic
format long

%% Data Loading

PathName = uigetdir(pwd, 'Select a folder');
files = dir(fullfile(PathName, '*.xlsx'));
files2=struct2table(files);
files3=table2cell(files2);
cd(PathName)
file_names=files3(:,1);

for u=1:length(file_names)

    FileName=strjoin(file_names(u)); %selects rat currently up for analysis
in loop
    newfilename=FileName(1:end-5); %forms rat name for use in naming files
later
    newfilename2=newfilename(1:end-1); %removes last underscore
    newfilename3=newfilename2; %make a copy of newfilename2

    count=0;
    for i=1:length(newfilename2) %this loop inserts a forward slash so that
when used in the title of a graph still shows underscore

        if newfilename2(i)== '_'

            newfilename3(count+i)='\';
            newfilename3=[newfilename3(1:count+i) newfilename2(i:end)];
            count=count+1;

        end

    end

    DP_Q = xlsread(FileName); %loads individual rat waveform data
    initial_values=xlsread(FileName,2); %loads initial value for Rd
    initial_values2 =xlsread(FileName,1);  %loads impedance spectrum

    flow_data=DP_Q(:,4); %loads flow waveform
    pressure_data=DP_Q(:,1); %loads pressure waveform

    %% Select a cycle
```

```
    %Selecting middle cycle based on minimum pressures
    DataInv = 1.01*max(pressure_data) - pressure_data; %invert PA pressure
data to then find peaks (minimums)
    [~,MinIdx] = findpeaks(DataInv,'MinPeakDistance',150);
    k=1;

    if MinIdx(k)<100
        k=k+1;
    end

    %Defining the middle cycle for calculations
    flow=flow_data(MinIdx(k):MinIdx(k+1));
    P_data=pressure_data(MinIdx(k):MinIdx(k+1));
    t_total=0.001:0.001:length(flow)*0.001;

    %% Initial parameters

    impedance = initial_values2(:,6);

    R= mean(impedance(4:11)); %3-10 harmonics for initial R

    Rd= initial_values(1)-R; %initial Rd

    R0=initial_values(1); %%for 2 element Windkessel R0=Z0

    % Compliance 2-element model

    %define parameters ode45 in opt2 function (custom function)
    P0=P_data(1);
    t_span=[0 length(t_total)*0.001];
    Q=flow;

    %Initial guess of 0.5 for C0
    beta2=0.5;
    f1=fminsearchbnd(@opt2,beta2,0,[]); %uses fminsearchbnd function to
optimize C0 only

    C0=f1;



    %% Call each Windkessel Model

     cd('C:\Users\gerri\Dropbox\Windkessel\') %set path to save values

    %Use custom functions for each model to run the data
    % % 4 Element

[Rd_S4,R_S4,C_S4,L_S4,error_S4,R2_S4,exitflag_S4,ranking_S4,AIC_S4]=Four_Elem
ent_Model_Series_NEW(Rd, R, C0, flow, newfilename);

    % % 3 Elements

[Rd_S3,R_S3,C_S3,L_S3,error_S3,R2_S3,exitflag_S3,ranking_S3,AIC_S3]=Three_Ele
ment_Series(Rd, R, C0, flow, newfilename);

    %save results in one vector to print to excel
```

```
    WS4_results=[Rd_S4 R_S4 C_S4 L_S4 error_S4 R2_S4 exitflag_S4 AIC_S4];
    WS3_results=[Rd_S3,R_S3,C_S3,L_S3,error_S3,R2_S3,exitflag_S3 AIC_S3];

    %save all results in one excel sheet for stats
    rat_name=file_names(u);
    xlswrite('compiled_results.xlsx',rat_name,1,['A' num2str(u)])
    xlswrite('compiled_results.xlsx',WS4_results,1,['B' num2str(u) ':J'
num2str(u)])
    xlswrite('compiled_results.xlsx',WS3_results,1,['AF' num2str(u) ':AN'
num2str(u)])

    xlswrite('sensitivity_results.xlsx',rat_name,1,['A' num2str(u)])
    xlswrite('sensitivity_results.xlsx',ranking_S4,1,['B' num2str(u) ':E'
num2str(u)])
    xlswrite('sensitivity_results_mct.xlsx',ranking_S3,1,['Q' num2str(u) ':T'
num2str(u)])

    xlswrite('intial_compliance.xlsx',rat_name,1,['A' num2str(u)])
    xlswrite('intial_compliance.xlsx',f1,1,['B' num2str(u)])



    cd(PathName) %return to original path to load remaining rat data

close all
end
```

```
%% Four_Element_Model_Series
% Code that fits predicts pressure using 4 parameter electrical
model in series
% Using acquired flow data
% ------------------------------------------------------------
%  Code written by JG
%  Last modified by JG
---------------


function
[Rd_opt,R_opt,C_opt,L_opt,error,R2,exitflag,ranking,AIC]=Four_El
ement_Model_Series_NEW(Rd_imp, R_imp, C_pre, flow, newfilename)
global P_data t_span P0 Q dQ_dt d2Q_dt2 t_total
format long

%% Initial Values

%Load initial conditions from main code (except L)
R=R_imp; % mmHg*min/ml
Rd= Rd_imp;
C= C_pre;
L=  0.002; %initial guess for L

%% Fitting flow
omega=(1/(length(t_total)*0.001))*2*pi; %determine frequency of
waveform

n1=20; %number of sine and cosine terms used for fitting
[Q,dQ_dt,x]=cossin_fit(t_total,flow,omega,n1); %define dq/dt for
4-element calculations


%% Differentiate Flow again
d2Q_dt2=0;
i=1;

%define d2q/dt2 for 4-element calculations
for n=1:n1
    d2Q_dt2=d2Q_dt2-
x(i)*(n^2)*(omega^2)*cos(n*omega.*t_total)+x(i+1)*(n^2)*(omega^2
)*sin(n*omega.*t_total);
    i=i+2;
end

%% Inductance in series
```

```
%defining values for ode45 calculations
t_span=[0 length(t_total)*0.001];
beta=[R;Rd;C;L]; %load parameters to be optimized
P0=P_data(1);

[f2, ~, exitflag] =fminsearchbnd(@opt_S,beta, [0;0;0;0], []);
%optimize using opt_S function (custom) and fminsearchbnd

options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6); %tolerance
[T,P_calc2]=ode45(@(T,P_calc2)
W_S(T,P_calc2,f2),t_span,P0,options); %final solution

P_calc2=interp1(T,P_calc2(:,1),t_total)'; %interpolate to normal
time steps

%% plot using Windkessel plot function

f2 = f2';
%calculate error
error=sqrt(sum((P_calc2-P_data).^2)/sum(P_data.^2));

%calculate R2 value
y_mean = mean((P_data-P_calc2).^2);
SS_tot = sum((P_data-y_mean).^2);
SS_res = sum((P_data-P_calc2).^2);

R2 = 1-(SS_res/SS_tot);

%AIC and Sensitivity
J = sum(abs((P_data-P_calc2)).^2);
n = length(P_data);
np = 4;

AIC = n*log10(error/n)+2*np;
newfilename2=[char(newfilename) '_WS4_NEW2']; %naming scheme for
individual rat files
m=4;
ranking=Sensitivity_Series_NEW(f2, newfilename2,m); %calculate
sensitivity values using Sensitivity_Series_NEW (custom
function)

%% Save Data
%Saving Data for individual rats
xlswrite([char(newfilename) '_WS4_NEW2.xlsx'],P_calc2,1,'A1')

xlswrite([char(newfilename) '_WS4_NEW2.xlsx'],f2(2),2,'A1')
xlswrite([char(newfilename) '_WS4_NEW2.xlsx'],f2(1),2,'B1')
```

```
xlswrite([char(newfilename) '_WS4_NEW2.xlsx'],f2(3:4),2,'C1:D1')

xlswrite([char(newfilename) '_WS4_NEW2.xlsx'],error,2,'E1')
xlswrite([char(newfilename) '_WS4_NEW2.xlsx'],R2,2,'F1')
xlswrite([char(newfilename) '_WS4_NEW2.xlsx'],AIC,2,'G1')
xlswrite([char(newfilename) '_WS4_NEW2.xlsx'],exitflag,2,'H1')

%parameters to be loaded in main Windkessel code
Rd_opt=f2(2);
R_opt=f2(1);
C_opt=f2(3);
L_opt=f2(4);


end
```

```
close all; clc; clear all;

%% Code that fits PAP pressure and flow in order to calculate
impedance
%
%  Code written by JG 06/10/2015
%  Last modified by JG 10/11/2016

% ------------------------------------------------------------------
---------------
format long

%% Data Loading

addpath('C:\Users\gerri\Dropbox\Windkessel\MATLABcode\Functions\
');

[FileName,PathName] = uigetfile('*.txt','Select the MATLAB code
file');
cd(PathName)
dual_pressure = load(FileName);
newfilename=FileName(1:end-4); %forms rat name for use in naming
files later
newfilename2=newfilename(1:end-1); %removes last underscore
newfilename3=newfilename2; %make a copy of newfilename2 that
becomes one with forward slashes
newfilename2=newfilename;
count=0;
for i=1:length(newfilename2) %this loop inserts a forward slash
so that when used in the title of a graph still shows underscore

    if newfilename2(i)== '_'

        newfilename3(count+i)='\';
        newfilename3=[newfilename3(1:count+i)
newfilename2(i:end)];
        count=count+1;

    end

end

%loading original data
time = dual_pressure(:,1); %time
RVP = dual_pressure(:,3); %RV pressure
PAP = dual_pressure(:,2); %PA pressure
flow=dual_pressure(:,4); %flow
```

```
hr=dual_pressure(:,5); %heart rate

flow_org=flow;
RVP=sgolayfilt(RVP,3,11);
flow=sgolayfilt(flow,3,15);
PAP=sgolayfilt(PAP,3,15);
data_length=length(flow);

%% Selecting Range of flow and pressure

DataInv = 1.01*max(PAP) - PAP; %invert PA pressure data to then
find peaks (minimums)
[~,MinIdx] = findpeaks(DataInv,'MinPeakDistance',150);


figure(1)
plot(PAP)
hold on
for i=1:length(MinIdx)

    scatter(MinIdx(i),PAP(MinIdx(i))) %plot minimums on PAP

end

for i=2:length(MinIdx)

        L(i-1)=MinIdx(i)-MinIdx(i-1); %find lengths of good
cycles

end
title('pressure')
hold off

figure(13)
plot(flow)
title('flow')
L(:,~any(L,1))=[];
avg_length=round(mean(L)); %find average length
i=4;
avg_hr=mean(mean(hr));
prompt='Input Stroke Volume: ';
sv=input(prompt);
expect_CO=(avg_hr*sv)/1000;
disp_exp_co=num2str(expect_CO);
legend(['\bf expected_co=' char(disp_exp_co)])
num_seg=round(length(MinIdx)/3); %preallocate space for flow,
pressures, heart rate
```

```
flow2=zeros(num_seg,avg_length*3);%preallocation to avoid issues
with cycles with different lengths
PAP2=zeros(num_seg,avg_length*3);
hr2=zeros(num_seg,avg_length*3);
RVP2=zeros(num_seg,avg_length*3);
count=1;
flow_max=750; %limits for flow peaks, may vary from rat to rat
flow_min=50;

while i<length(MinIdx)

    %Makes sure that the 3 cycles selected are close to the
average length
    %times 3
    if MinIdx(i)-MinIdx(i-3)<(avg_length*3)+40 && MinIdx(i)-
MinIdx(i-3)>(avg_length*3)-40

        %estimates the length of each cycle of the 3 for
checking maximum values
        dist=round((MinIdx(i)-MinIdx(i-3))/3)-2;

        %next three if statements find if the max pressure of
each selected
        %cycle is within range.  If so it loads those 3 cycles
        if max(flow(MinIdx(i-3):MinIdx(i-3)+dist))>flow_min &&
max(flow(MinIdx(i-3):MinIdx(i-3)+dist))<flow_max

            if max(flow(MinIdx(i-3)+dist:MinIdx(i-
3)+dist*2))>flow_min && max(flow(MinIdx(i-3)+dist:MinIdx(i-
3)+dist*2))<flow_max

                if max(flow(MinIdx(i-3)+dist*2:MinIdx(i-
3)+dist*3))>flow_min && max(flow(MinIdx(i-3)+dist*2:MinIdx(i-
3)+dist*3))<flow_max

                    seg_length=MinIdx(i)-MinIdx(i-3);
                    flow2(count,1:seg_length+1)=flow(MinIdx(i-
3):MinIdx(i));
                    PAP2(count,1:seg_length+1)=PAP(MinIdx(i-
3):MinIdx(i));
                    hr2(count,1:seg_length+1)=hr(MinIdx(i-
3):MinIdx(i));
                    RVP2(count,1:seg_length+1)=RVP(MinIdx(i-
3):MinIdx(i));
                    sys_p(count)=max(PAP(MinIdx(i-2:i-1)));
                    dia_p(count)=PAP(MinIdx(i-1));
                    count=count+1;
```

```
                end

            end

        end

    end

    i=i+1;

end

flow2( ~any(flow2,2), : ) = []; %removes any rows of all 0's
PAP2( ~any(PAP2,2), : ) = [];
hr2( ~any(hr2,2), : ) = [];
RVP2( ~any(RVP2,2), : ) = [];


PAP2(:,~any(PAP2,1))=[]; %removes any columns of all 0's since
pap never reaches 0
PAP2=PAP2(:,1:end-58); %remove the end to remove any extra 0's
(due to preallocating this happens)
[~,X2]=size(PAP2); %truncate all other loads matrices to this
length
flow2=flow2(:,1:X2);
hr2=hr2(:,1:X2);
RVP2=RVP2(:,1:X2);
t=0.001:0.001:X2*0.001; %form time vector in milliseconds for
new size

% figure(2)
% plot(PAP2(1,:))
% figure(3)
% plot(flow2(1,:))

%% Fitting the PA pressure

[num_seg2,~]=size(flow2); %finds the number of 3 cycles segments
n1=90; %number of terms for fitting
n2=10; %number of harmonics we will analyze

for gg=1:num_seg2
```

```
    avg_hr(gg)=mean(hr2(gg,:)); %finds average heart rate over
span and finds angular frequency for pressure and flow fittings
    omega(gg)=avg_hr(gg)*2*pi/60;
    %fits the pressure using a*cos(nwt)+b*sin(nwt)+c
    %also acquires coefficients for determining amplitude and
phase of
    %pressure

[PAP_fit(gg,:),~,PAP_co(gg,:)]=cossin_fit(t,PAP2(gg,:)',omega(gg
),n1);
    p_co=PAP_co(gg,:); %load coefficients for segment

    i=1;

    for n=1:n2


        phi_pressure(gg,n)=atan(p_co(i+1)/p_co(i)); %phase angle
determined from coefficients

        Pn(gg,n)=sqrt(p_co(i)^2+p_co(i+1)^2); %amplitude
determined from coefficients

        i=i+2;

    end

    %% Fitting the flow

    %fits flow the same as pressure

[flow_fit(gg,:),~,flow_co(gg,:)]=cossin_fit(t,flow2(gg,:)',omega
(gg),n1);
    f_co=flow_co(gg,:); %load flow coefficients

    i=1;

    for n=1:n2



        phi_flow(gg,n)=atan(f_co(i+1)/f_co(i)); %phase angle for
flow

        Qn(gg,n)=sqrt(f_co(i)^2+f_co(i+1)^2); %amplitude for
flow
```

```
        i=i+2;

    end

    %% Impedance Calculation

    Z(gg,:)=(Pn(gg,:)./Qn(gg,:)); %calculates impedance at each
harmonic frequency

    % calculates mean pressure and flow to see how it matches
with the last term in fitting
    mean_flow(gg)=trapz(t,flow2(gg,:))/(t(end)-t(1));
    mean_pressure(gg)=trapz(t,PAP2(gg,:))/(t(end)-t(1));

end

%% Plotting data

[aa,~]=size(flow_co);

z2=[PAP_co(:,end)./flow_co(:,end) Z]; %add in 0th order harmonic
(total resistance)
z3=z2';
Zc=mean(z3(8:11,:));

z0=PAP_co(:,end)./flow_co(:,end); %store 0th order impedance to
analyze trends
z_values=[z0 Zc'];
mean_values=[PAP_co(:,end) flow_co(:,end)];

hfig1=figure(3);
hfig2=figure(4);
search=0;
i=1;
[num_avg_seg,~]=size(z2);
num_prompt=num2str(num_avg_seg);
str='y';
N='n';
Y='y';
K='k';
while search==0 && i<=num_avg_seg

    if i~=1

        %asks user to plot next segment or ask for specific
number
```

```
        prompt='Would you like to plot another? y/n \n For a
specific impedance type k:  ';
        str=input(prompt,'s');

    end

    tf1=strcmp(str,N); %these variables compare user input to
no, yes, and chossing number respectively
    tf2=strcmp(str,Y);
    tf3=strcmp(str,K);

    if tf2==1 && tf1==0 && tf3==0 %if user enter y for yes


        clc
%         close(hfig1)
%         close(hfig2)
        num2=num2str(i);
        %plots the orginal pressure data with fitted data
        figure(3)
        plot(PAP_fit(i,:),'r','LineWidth',2)
        hold on
        plot(PAP2(i,:),'LineWidth',2)
        plot(RVP2(i,:),'k','LineWidth',2)
        set(gca,'FontSize',14);
        xlabel('\bf Time (msec)')
        ylabel('\bf Pressure (mmHg)')
        title(['\bf Fitted Pressure Curve and Pressure for '
char(newfilename3) '(' char(num2) ')'])
        legend('Fitted curve','Pressure')
        hold off
        movegui('west')

        %plots the original flow data with fitted data
        figure(4)
        plot(flow_fit(i,:),'r','LineWidth',2)
        hold on
        plot(flow2(i,:),'LineWidth',2)
        set(gca,'FontSize',14);
        xlabel('\bf Time (msec)')
        ylabel('\bf Flow (ml/min)')
        title(['\bf Fitted flow Curve and Flow for '
char(newfilename3) '(' char(num2) ')'])
        disp_mean=num2str(mean_values(i,2));
        legend('Fitted curve',['Flow (mean=' char(disp_mean)
')'])
        hold off
```

```
        movegui('east')

        harm=0:n2;
        phase=(phi_pressure(i,:)-phi_flow(i,:));
        phase=[0 phase];
        figure(5)
        %plots phase
        plot(harm,phase,'.-r','LineWidth',3,'MarkerSize',30)
        set(gca,'FontSize',14);
        set(gca,'XTick',harm);
        % xlim([0.5 5])
        title(['\bf Phase Shift for ' char(newfilename3) '('
char(num2) ')'])
        xlabel('\bf Harmonic (n)')
        ylabel('\bf Phase (radians)')
        movegui('south')

        %plots the impedance
        figure(6)
        plot(harm,z2(i,:),'.-','LineWidth',3,'MarkerSize',30)
        set(gca,'FontSize',14);
        set(gca,'XTick',harm);
        % axis([-0.5 5 0 12])
        title(['\bf Impedance for ' char(newfilename3) '('
char(num2) ')'])
        xlabel('\bf Harmonic (n)')
        ylabel('\bf Impedance (mmHg/mL/min)')

    else if tf1==1 && tf2==0 && tf3==0 %if user enters n for no

            search=1;

    else if tf3==1 && tf1==0 && tf2==0 %if user enters k for
inputing specific number

            %user inputs number of segment they desire to
analyze
            prompt=['Enter average cycle number you would like
to plot (1-' char(num_prompt) '):  '];
            num=input(prompt);
            num3=num2str(num);

            clc
%             close(hfig1)
%             close(hfig2)
            %plots the orginal pressure data with fitted data
            figure(3)
```

```
            plot(PAP_fit(num,:),'r','LineWidth',2)
            hold on
            plot(PAP2(num,:),'LineWidth',2)
            plot(RVP2(num,:),'k','LineWidth',2)
            set(gca,'FontSize',14);
            xlabel('\bf Time (msec)')
            ylabel('\bf Pressure (mmHg)')
            title(['\bf Fitted Pressure Curve and Pressure for '
char(newfilename3) '(' char(num3) ')'])
            legend('Fitted curve','Pressure')
            hold off
            saveas(gcf,[char(newfilename) 'fitted_pressure ' '('
char(num3) ').jpg'])

            %plots the original flow data with fitted data
            figure(4)
            plot(flow_fit(num,:),'r','LineWidth',2)
            hold on
            plot(flow2(num,:),'LineWidth',2)
            set(gca,'FontSize',14);
            xlabel('\bf Time (msec)')
            ylabel('\bf Flow (ml/min)')
            title(['\bf Fitted flow Curve and Flow for '
char(newfilename3) '(' char(num3) ')'])
            disp_mean=num2str(mean_values(n,2));
            legend('Fitted curve',['Flow (mean=' char(disp_mean)
')'])
            hold off
            saveas(gcf,[char(newfilename) 'fitted_flow ' '('
char(num3) ').jpg'])

            harm=0:n2;
            phase=(phi_pressure(num,:)-phi_flow(num,:));
            phase=[0 phase];
            figure(5)
            plot(harm,phase,'.-r','LineWidth',3,'MarkerSize',30)
            set(gca,'FontSize',14);
            set(gca,'XTick',harm);
            % xlim([0.5 5])
            title(['\bf Phase Shift for ' char(newfilename3) '('
char(num3) ')'])
            xlabel('\bf Harmonic (n)')
            ylabel('\bf Phase (radians)')
            saveas(gcf,[char(newfilename) 'phase ' '('
char(num3) ').jpg'])

            %plots the impedance from
```

```
            figure(6)
            plot(harm,z2(num,:),'b.-
','LineWidth',3,'MarkerSize',25)
            set(gca,'FontSize',14);
            set(gca,'XTick',harm);
%             set(gca,'XTickLabel',{'0','5','10'})
%             set(gca,'FontWeight','bold','Xtick',[0 5 10])
%             axis([0 10 0 1.3])
            title(['\bf Impedance for ' char(newfilename3) '('
char(num3) ')'])
            xlabel('\bf n')
%             ylabel({'\bf Impedance'; '\bf
[mmHg/mL/min]'},'FontSize',42)
            ylabel('\bf Z(n)')
            saveas(gcf,[char(newfilename) 'imped ' '('
char(num3) ').jpg'])
            z_to_save=z2(num,:);
%             save(['Impedance_' char(newfilename2) '('
char(num3) ')' '.txt'],'z_to_save','-ascii') %saves impedance
vector as a text file


            pap_stuff=PAP2(num,:)';
            rvp_stuff=RVP2(num,:)';
            pap_fit=PAP_fit(num,:)';
            flow_stuff=flow2(num,:)';
            fit_flow=flow_fit(num,:)';
            imped=z2(num,:);
            mpap_flow=mean_values(num,:);
            zs=z_values(num,:);

            [r1,~]=size(pap_stuff);
            r2=num2str(r1);

            xlswrite([char(newfilename) ' (' char(num3)
').xlsx'],pap_stuff,1,['A1:A' char(r2)]) %saves the vector pap
            xlswrite([char(newfilename) ' (' char(num3)
').xlsx'],rvp_stuff,1,['B1:B' char(r2)])%saves the vector rvp
            xlswrite([char(newfilename) ' (' char(num3)
').xlsx'],pap_fit,1,['C1:C' char(r2)])%saves the vector fitted
pap
            xlswrite([char(newfilename) ' (' char(num3)
').xlsx'],flow_stuff,1,['D1:D' char(r2)])%saves the vector flow
            xlswrite([char(newfilename) ' (' char(num3)
').xlsx'],fit_flow,1,['E1:E' char(r2)])%saves the vector fitted
flow
```

```
            xlswrite([char(newfilename) ' (' char(num3)
').xlsx'],imped',1,'F1:F11')%saves the vector impedance
            xlswrite([char(newfilename) ' (' char(num3)
').xlsx'],phase',1,'G1:G11')%saves the vector phase
            xlswrite([char(newfilename) ' (' char(num3)
').xlsx'],mpap_flow,2,'C1:D1')%saves mean pressure and flow
values in mpa to sheet 2
            xlswrite([char(newfilename) ' (' char(num3)
').xlsx'],zs,2,'A1:B1')%saves z0 and zc to sheet 2


        end

        end

    end

    i=i+1;

end


mean(mean(hr2)) %displays heart rate over file
```

```
function error=opt2(theta)

global t_span P0 P_data t_total %global variables to use in
ode45 to iteratively solve for C0

%ode45 accessing W2 function (custom) that has definition of 2-
element
%windkessel model
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6);
[T1,P_calc2]=ode45(@(T1,P_calc2)W2(T1,P_calc2,theta),t_span,P0,o
ptions);

%calculated result (interpolated to be compared to data)
P_calc2=interp1(T1,P_calc2,t_total)';

%error function for optimization
error=sqrt(sum((P_calc2-P_data).^2)/sum(P_data.^2));

end
```

```
function error=opt_3element(beta)

global t_span P0 P_data t_total %load parameters for ode45
calculations

options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6);%tolerance
[T1,P_calc2]=ode45(@(T1,P_calc2)W_3E(T1,P_calc2,beta),t_span,P0,
options); %ode45 solution using W_3E function (custom)

P_calc2=interp1(T1,P_calc2,t_total)'; %interpolation of waveform
to compare to data

% error calculations
error=sqrt(sum((P_calc2-P_data).^2)/sum(P_data.^2));

end
```

```
function error=opt_S(beta)

global t_span P0 P_data t_total %load parameters for ode45
calculations

options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6); %tolerance
[T1,P_calc2]=ode45(@(T1,P_calc2)W_S(T1,P_calc2,beta),t_span,P0,o
ptions); %ode45 solution using W_S function (custom)

P_calc2=interp1(T1,P_calc2(:,1),t_total)'; %interpolation of
waveform to compare to data

% error calculations
error=sqrt(sum((P_calc2-P_data).^2)/sum(P_data.^2));


end
```

```
%Sensitivity Calcluations for series models

function n=Sensitivity_Series_NEW(f2, FileName,m)
global P_data Q dQ_dt d2Q_dt2 %load applicable values from main
code


    R = f2(1);
    Rd = f2(2);
    C = f2(3);
    L = f2(4);

t = 0.001:0.001:length(P_data)*0.001;

%differentiate in terms of parameters
Exp_pos = exp(t./(Rd*C));
Exp_neg = exp(-t./(Rd*C));
Series_EQ = (L*d2Q_dt2+ (R+(L/(C*Rd)))*dQ_dt+
((1/C)+(R/(C*Rd)))*Q);

%check
for i = 1:length(t)
    int_Series = Exp_pos(1:i).*Series_EQ(1:i);
    Trap_S(i) = .001*trapz(int_Series);

    P(i) = Exp_neg(i)*Trap_S(i)+P_data(1)*Exp_neg(i);
end

plotting_windkessel(t, P_data, P)

%in terms of R
for i = 1:length(t)
    int_R = (dQ_dt(1:i)+ Q(1:i)./(C*Rd)).*Exp_pos(1:i);
    Trap_R(i) = .001*trapz(int_R);

    dP_dR(i) = Exp_neg(i).* Trap_R(i);
end

%in terms of Rd
for i = 1:length(t)
    int_Rd = (-
t(1:i)/((Rd^2)*C)).*Exp_pos(1:i).*Series_EQ(1:i)...
        + Exp_pos(1:i).*((-L/(C*(Rd^2)))*dQ_dt(1:i)...
        +((-R*Q(1:i))./(C*(Rd^2))));
    Trap_Rd(i) = .001*trapz(int_Rd);

    int_Series_RD = Exp_pos(1:i).*Series_EQ(1:i);
```

```
    Trap_SRd(i) = .001*trapz(int_Series_RD);

    dP_dRd(i) = (t(i)/((Rd^2)*C)).*Exp_neg(i).*Trap_SRd(i)+
Exp_neg(i).*Trap_Rd(i)...
    +((t(i)*P_data(1))/(C*(Rd^2))).*Exp_neg(i);

end

%in terms of C
for i = 1:length(t)
    int_C = (-
t(1:i)/((C^2)*Rd)).*Exp_pos(1:i).*Series_EQ(1:i)...
        + Exp_pos(1:i).*((-L/(Rd*(C^2)))*dQ_dt(1:i)...
        -Q(1:i)./(C^2)+ ((-R*Q(1:i))./(Rd*(C^2))));
    Trap_C(i) = .001*trapz(int_C);

    int_Series_C = Exp_pos(1:i).*Series_EQ(1:i);
    Trap_SC(i) = .001*trapz(int_Series_C);

    dP_dC(i) = (t(i)/((C^2)*Rd)).*Exp_neg(i).*Trap_SC(i)+
Exp_neg(i).*Trap_C(i)...
    +((t(i)*P_data(1))/((C^2)*Rd)).*Exp_neg(i);

end

%in terms of L (for 4-element)
if m==4
    for i = 1:length(t)
        int_L = (d2Q_dt2(1:i)+
(1/C*Rd)*dQ_dt(1:i)).*Exp_pos(1:i);
        Trap_L(i) = .001*trapz(int_L);

        dP_dL = Exp_neg(1:i).*Trap_L;
    end


    %Jacobian
    J = [dP_dR; dP_dRd; dP_dC; dP_dL];

    %Normalized
    Par_Pres = [R./P_data, Rd./P_data, C./P_data, L./P_data];

    for i = 1:4
   S(i,:) =  Par_Pres(:,i).*J(i,:)';
    end

    %Ranking
```

```
    for i = 1:4
        n(i) = norm(S(i,:));
    end

else %3-element conditions

     %Jacobian
    J = [dP_dR; dP_dRd; dP_dC];

    %Normalized
    Par_Pres = [R./P_data, Rd./P_data, C./P_data];

    for i = 1:3
        S(i,:) =  Par_Pres(:,i).*J(i,:)';
    end

    %Ranking
    for i = 1:3
        n(i) = norm(S(i,:));
    end

end


%Correction for saving values
if m==4
n = [n(2), n(1), n(3), n(4)];

else

n = [n(2), n(1), n(3), 0];
S(4,:)=0;

end

%saving rankings for each parameter for each rat
xlswrite(['Sensitivity_' char(FileName) '.xlsx'],S(2,:)',1,'A1')
xlswrite(['Sensitivity_' char(FileName) '.xlsx'],S(1,:)',1,'B1')
xlswrite(['Sensitivity_' char(FileName) '.xlsx'],S(3,:)',1,'C1')
xlswrite(['Sensitivity_' char(FileName) '.xlsx'],S(4,:)',1,'D1')
xlswrite(['Sensitivity_' char(FileName) '.xlsx'],n,2,'A1:D1')



end
```

```
%% Code that fits predicts pressure using 3 parameter electrical
model
% Using known flow data
%  Code written by JG 08/29/2015
%  Last modified by JG 05/31/2017
% ----------------------------------------------------------------
---------------

%% Data Loading

function
[Rd_opt,R_opt,C_opt,L_opt,error,R2,exitflag,ranking,AIC]=Three_E
lement_Series(Rd_imp, R_imp, C_Pre, flow, newfilename)
global P_data t_span P0 Q dQ_dt t_total
format long

%% Fitting flow

omega=(1/(length(t_total)*0.001))*2*pi; %determine frequency of
waveform

n1=20; %number of sine and cosine terms used for fitting
[Q,dQ_dt]=cossin_fit(t_total,flow,omega,n1); %define dq/dt for
3-element calculations

%% 3 Element Model

%Initial parameters
R=R_imp;
Rd=Rd_imp;
C=C_Pre;

%parameters for ode45
P0=P_data(1);
t_span=[0 length(t_total)*0.001];

%load values that will be optimized
beta=[R; Rd; C];

[f1,~,exitflag]=fminsearchbnd(@opt_3element,beta,[0;0;0],[]);
%optimize using opt_3element function (custom) and fminsearchbnd

options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6); %set tolerance
```

```
[T,P_calc]=ode45(@(T,P_calc)
W_3E(T,P_calc,f1),t_span,P0,options); %final calculation with
optimized parameters

P_calc=interp1(T,P_calc,t_total)'; %interpolate to normal time
steps

error=sqrt(sum((P_calc-P_data).^2)/sum(P_data.^2)); %calculate
error

parameters=[f1(2) f1(1) f1(3) 0 error]; %loads parameters and
error (0 included to standardize saving process with 4-element)

%added 0 so sensitivity function (custom) can load both 3 and 4
element
%models
f1 = [f1;0]';

%calculate R2 value
y_mean = mean((P_data-P_calc).^2);
SS_tot = sum((P_data-y_mean).^2);
SS_res = sum((P_data-P_calc).^2);

R2 = 1-(SS_res/SS_tot);

%AIC and Sensitivity calculations
J = sum(abs((P_data-P_calc)).^2);
n = length(P_data);
np = 3;

AIC = n*log10(error/n)+2*np;
newfilename2=[char(newfilename) '_W3S_NEW2']; %naming scheme for
individual rat files
m=3;
ranking=Sensitivity_Series_NEW(f1, newfilename2,m); %calculate
sensitivity values using Sensitivity_Series_NEW (custom
function)

%Saving Data for individual rats
xlswrite([char(newfilename) '_W3S_NEW2.xlsx'],P_calc,1,'A1')
xlswrite([char(newfilename) '_W3S_NEW2.xlsx'],P_data,1,'B1')
xlswrite([char(newfilename) '_W3S_NEW2.xlsx'],flow,1,'C1')
xlswrite([char(newfilename) '_W3S_NEW2.xlsx'],parameters,2,'A1')
xlswrite([char(newfilename) '_W3S_NEW2.xlsx'],R2,2,'F1')
xlswrite([char(newfilename) '_W3S_NEW2.xlsx'],AIC,2,'H1')
xlswrite([char(newfilename) '_W3S_NEW2.xlsx'],exitflag,2,'I1')
```

```
%parameters to be loaded in main Windkessel code
Rd_opt=f1(2);
R_opt=f1(1);
C_opt=f1(3);
L_opt=0;


end
```

```
%This function defines the 2-element Windkessel model
function dpdt=W2(T,P,theta)

global Q t_total R0 %defines constants

% loads C0 for calculation/optimization
C=theta(1);

%interpolation for flow waveform for solving
Qi=interp1(t_total,Q,T,'spline');

dpdt=(Qi/C)-(P/(R0*C));
```

```
%differential equation for pressure in 3 element Windkessel
function dpdt = W_3E(T,P,beta)

    global Q dQ_dt t_total %loading constants

%interpolation for flow waveforms for calcluations
    Qi=interp1(t_total,Q,T,'spline');
    dQ_dti=interp1(t_total,dQ_dt,T,'spline');

%loading parameters
    R=beta(1);
    Rd=beta(2);
    C=beta(3);

    %equation for 3-element model
    dpdt=R*dQ_dti+(Qi/C)+((Qi*R)/(Rd*C))-P/(Rd*C);


end
```

```
%differential equation for pressure in inductor in parallel 4
element Windkessel
function dpdt = W_S(T,P,beta)

global Q dQ_dt d2Q_dt2 t_total %loading constants

%interpolation for flow waveforms for calculations
Qi=interp1(t_total,Q,T,'spline');
dQ_dti=interp1(t_total,dQ_dt,T,'spline');
d2Q_dt2i=interp1(t_total,d2Q_dt2,T,'spline');

%loading parameters
    R=beta(1);
    Rd=beta(2);
    C=beta(3);
    L=beta(4);

%first derivative
dpdt= L*d2Q_dt2i+ (R+(L/(C*Rd)))*dQ_dti+ ((1/C)+(R/(C*Rd)))*Qi-
(P(1)/(Rd*C));


end
```