```
function [ gamma, betax, kappa, minssq, smeanmod, xmeanmod, rmeanmod] =
    haiti_calc(gamma, betax, kappa, phip, susceptible_data,
    infectious_data, removed_data )

%HAITI_CALC
%   This file is used to calculate the matrices for the compartments s, x
%   and r receiving the initial values from the Initialisation file.




%Amount of days calculated
t=92;

%Amount of districts Haiti is divided into
d=10;

%Setting the parameters for the differential equations to the right
%dimensions
gamma=ones(10,1)*gamma;

betax=ones(10,1)*betax;

kappa=kappa;






%Prepare all the needed blank matrices
s=ones(d,t);

x=ones(d,t);

r=ones(d,t);




%------------------------------------------------
%THIS IS THE INITIAL CONTIDIONS FOR THE MODEL
%Setting the infectious compartment of Grand Anse to the same as Nord
    Ouest
s(:,1)=susceptible_data(:,1);

x(:,1)=infectious_data(:,1);

r(:,1)=removed_data(:,1);

x(3,1)=infectious_data(6,1);




%Start the loop for time iteration calculation and handing data over to
```

```
    the
%euler function
for p=2:t


    lambda = betax .* x(:,p-1) + (kappa*phip)*x(:,p-1);







    [s(:,p),x(:,p), r(:,p)]=euler(s(:,p-1),x(:,p-1),r(:,p-1), lambda,
        gamma);


end




%Initialisation of solver
%Calculating the total sum of squares.
%NOTE!!: Because the infectious compartment is higly nonlinear and
%impossible to model with such simple equations we ignore that
    compartment
%in our solver
ssqr=sum(sum(((r-removed_data)).^2));

ssqs=sum(sum(((s-susceptible_data)).^2));

totalssq=ssqs+ssqr;

%Preparing the coefficients matrix and starting the cycle count of the
%solver
coefficients=ones(3,500);
solvercount=2;

%Setting the first two elements of the SSQ vector to ensure
ssqvector(solvercount-1)=totalssq+1;
ssqvector(solvercount)=totalssq;










%Initialising the solver loop, setting the criteria to break if
    difference
%is >= 0.0001%
while abs(ssqvector(solvercount-1)-ssqvector(solvercount))  >=  0.000001*
    ssqvector(solvercount-1)
```

```
%Changing of parameters for solver
solver_percent=0.02;



%Preparing all the scenarios for the solver
%Sequence of 4 for kappa

kappasolve=ones(1,8)*kappa(1,1);

kappasolve(:,[1:4])=kappasolve(:,[1:4])+kappasolve(:,[1:4]).*
    solver_percent;

kappasolve(:,[5:8])=kappasolve(:,[5:8])-kappasolve(:,[5:8]).*
    solver_percent;




%Sequence of 2 for betax

betaxsolve=ones(10,8)*betax(1,1);

betaxsolve(:,[1,2,5,6])=betaxsolve(:,[1,2,5,6])+betaxsolve(:,[1,2,5,6]).*
    solver_percent;

betaxsolve(:,[3,4,7,8])=betaxsolve(:,[3,4,7,8])-betaxsolve(:,[3,4,7,8]).*
    solver_percent;





%Sequence of 1 for gamma
gammasolve=ones(10,8)*gamma(1,1);

gammasolve(:,[1:2:7])=gammasolve(:,[1:2:7])+gammasolve(:,[1:2:7]).*
    solver_percent;

gammasolve(:,[2:2:8])=gammasolve(:,[2:2:8])-gammasolve(:,[2:2:8]).*
    solver_percent;


%Prepare all the needed matrices

s=ones(d,t);

x=ones(d,t);

r=ones(d,t);

%---------------------------------------------
%THIS IS THE INITIAL CONTIDIONS FOR THE MODEL
%Setting the infectious compartment of Grand Anse to the same as Nord
    Ouest
```

```
s(:,1)=susceptible_data(:,1);

x(:,1)=infectious_data(:,1);

r(:,1)=removed_data(:,1);

x(3,1)=infectious_data(6,1);


%Preparing the SSQ vector for all 8 scenarios
ssqsolver=ones(1,8)*10;

    for z=1:8

        gammasolveloop=gammasolve(:,z);

        betaxsolveloop=betaxsolve(:,z);

        kappasolveloop=kappasolve(z);



        %Start the loop for time iteration calculation

        for p=2:t


            lambdasolveloop = betaxsolveloop .* x(:,p-1) +
                (kappasolveloop*phip)*x(:,p-1);





            [s(:,p),x(:,p), r(:,p)]=eulersolve(s(:,p-1),x(:,p-1),r(:,p-1)
                , lambdasolveloop, gammasolveloop);


        end


        ssqxsolve=sum(sum(((x-infectious_data)).^2));

        ssqrsolve=sum(sum(((r-removed_data)).^2));

        ssqssolve=sum(sum(((s-susceptible_data)).^2));
```

```
    totalssqsolve=ssqssolve+ssqrsolve;% +finalssqxsolve;

    ssqsolver(z)=totalssqsolve;




end




%Choosing the scenario with the smallest SSQ and setting the
%coefficients to the new values
[C,I]=min(ssqsolver);

gamma=gammasolve(:,I);

betax=betaxsolve(:,I);

kappa=kappasolve(:,I);






totalssq=ssqsolver(I);

%Placing all the coefficients within a matrix
coefficients(1,solvercount)=gamma(1,1);

coefficients(2,solvercount)=betax(1,1);

coefficients(3,solvercount)=kappa(1,1);

%Increasing the cycle count of the solver
solvercount=solvercount+1;

%Placing the new SSQ withing the solver SSQ vector to check for break
    criteria
ssqvector(solvercount)=totalssq;



%If 1000 cycles were calculated this function will stop the loop.
%Mainly to exclude the possibility for infinite loops
if solvercount>=1000
        break
end
```

```
    end





    %Recalculate with the best fit parameters
    [minssq,vectorpoint]=min(ssqvector);

    gamma=coefficients(1,vectorpoint);

    betax=coefficients(2,vectorpoint);

    kappa=coefficients(3,vectorpoint);

    for p=2:t


        lambda = betax .* x(:,p-1) + (kappa*phip) * x(:,p-1);





        [s(:,p),x(:,p), r(:,p)]=euler(s(:,p-1),x(:,p-1),r(:,p-1), lambda,
            gamma);


    end


    %Calculating the means to hand back to the initialiser file
    smeanmod=mean(s);
    xmeanmod=mean(x);
    rmeanmod=mean(r);









    end
```