```
function [logprob , varargout]=BXkinetics(params,data,sigma,t,powers)
% Function to evaluate kinetic model for biexciton and exciton states with
% poisson statistics:
%
% Full fluence series, no hot carrier thermalization, BX and X states, -
% k1,k2,XC
% BX-->k1-->X-->k2
% XC=cross section parameter (N/uW)
% Evaluates component dynamics
%
% Function evaluated the log likelihood function. The component
% spectra and fitted TA data are returned as optional outputs so that the
% same function can be used to evaluate the fit and produce visualizations
% after running the MCMC itself.
%
% Inputs:
%  params = kinetic model parameters - rate constants and cross section
%       parameters
%  data = Fluence dependent TA data stored as a 3D array, wavelength x time x
%       fluence
%  sigma = standard deviations of the individual points in data, typically
%       calcualted as the sample standard deviation from the set of scans
%  t = Time delays in ps, stored as a row vector
%  Powers = Vector of pump powers from each measurement, in uW
%
% Outputs:
%  logprob = log likelihood calculated from the sum of squared error
%  spec = Component spectra (optional)
%  datafit = Fitted TA spectra (optional)
%
%       -Matthew Ashner, 2018

params(1:2)=10.^params(1:2);  %Convert log rate constants to linear space
[specpts,tpts,numtrace]=size(data);  %Determine data size and preallocate arrays
dyn=NaN(2,tpts,numtrace);
datafit=NaN(specpts,tpts,numtrace);

%Solve kintetic model by eigenvalue decomposition for each fluence

%Define kinetic model as a matrix, edit this line to change kinetic model
K=[-params(1),0;params(1),-params(2)];
[U,D]=eig(K);  %Perform eigenvalue decomposition

numt=length(t);
numk=length(diag(D));

krep=repmat(diag(D),1,numt);
trep=repmat(t,numk,1);
eigdyn=exp(krep.*trep); %Dynamics in eigenbasis

```

```
%Evaluate kinetics with fluence-dependent initial conditions, edit this
%section to change model for component initialization
for i=1:numtrace
    %BX and X source terms from poisson distribution
    BXyield=1-poisscdf(1,params(3)*powers(i));
    Xyield=poisspdf(1,params(3)*powers(i));

    %Generate source term in component basis, generate matrix transform
    %that transforms source term to eigenbasis, evaluates dynamics, and
    %transforms back to component basis
    source=[BXyield;Xyield];
    A2=U*diag(U\source);
    dyn(:,:,i)=A2*eigdyn;  %Dynamics evaulated at input time points

end

%Concatenate fluence along time dimension to convert from 3D to 2D for
%linear least squares
data2=reshape(data,specpts,tpts*numtrace);
dyn2=reshape(dyn,2,tpts*numtrace);

spec=data2/dyn2; %Perform linear least squares

%Evaluate fitted TA data, back in 3D form
for i=1:numtrace
    datafit(:,:,i)=spec*dyn(:,:,i);
end

%Evaluate log probability
logprob=-1*sum((datafit-data).^2./sigma.^2,'all');

%Output component spectra and fitted TA data if requested
varargout{1}=spec;
varargout{2}=datafit;
end
```