```
% Master script for running the target analysis with Markov chain Monte
% Carlo sampler as proposed in Ashner and Tisdale, JPC 2018.  The script
% calls a model evaluator function and MCMC sampler as subroutines.
% Effective use of the script requires modifying a few lines for different
% models, as explained below.  The lines that should be checked when the
% model is changed are: 24, 25, 31, 70
% -Matthew Ashner, 2018

warning('off','MATLAB:rankDeficientMatrix');

% Uncomment to supress warnings in the workers if using the
% parallelization. Parallelization is enabled by setting 'Parallel' to true
% as another name-value pair in the TAMCMC calls.
% Genrally, the parallelization seems to not be useful for the target analysis problem.
% I suspect that the paralellization fails to improve performance because
% the most expensive step is the large matrix decomposition used to perform
% the linear least squares and determine the component spectrum, and matlab
% already has built-in efficient multithreading for the mrdivide function.
%
% spmd
%   warning('off','MATLAB:rankDeficientMatrix')
% end

lb=[-2.5,-6,1e-5]'; ub=[-1,-2.5,0.01]'; %Set lower and upper bounds for parameters
islog=logical([1 1 0]);  %Define which parameters are in log space (1 for log, 0 for linear)

%Define prior and log likelihood function handles.  Change the second function
%handle definition to reflect the name and inputs of your kinetic model
%function, and replace the inputs with the relevant variable names
%containing those inputs in your workspace.
logPfuns={@(params)all(params>=lb & params<=ub), @(params) BXkinetics(params,data,sigmatrunc,timetrunc,powerstrunc)};

%Initialize walkers randomly in the space defined by the bounds and execute
%tempered MCMC to find the global likelihood maximum.
nparams=length(lb);
randinit=rand(nparams,100).*repmat(ub-lb,1,100)+repmat(lb,1,100);
[walkers1, logP1]= TAmcmc(randinit,logPfuns,200000,'StepSize',1.7,'Temper',true);

%Sort the walkers by likelihood, use a threshold of the difference between
%succesive walkers to find the cut-off for the set of walkers grouped in
%the highest likelihood area.
[probs,inds]=sort(mean(logP1(2,:,150:end),3));
bound=find(probs-circshift(probs,1)>=10,1,'last');

%If all of the walkers are close together, the bound returns empty, this
%block detects that and keeps all the walkers in that case.
if isempty(bound)
```

```
    postsub=walkers1(:,:,150:end);
else
    postsub=walkers1(:,inds(bound:end),150:end);
end

%Estimate the most likely values for each parameter from the tempered MCMC
clear init
for i=1:nparams
[f,xi]=ksdensity(postsub(i,:));
[~,ind]=max(f);
init(i)=xi(ind);
end

%Initialize second MCMC in a ball around the area found in the first MCMC
nlogs=length(find(islog));
randinit2=nan(nparams,100);

randinit2(~islog,:)=init(~islog)'.*10.^(randn(nparams-nlogs,100).*0.002);
randinit2(islog,:)=init(islog)'+0.002.*randn(nlogs,100);

%Excecute second/main MCMC run, adjust step size for a 50-75% rejection
%rate as displayed in the progress bar.
[walkers2, logP2]= TAmcmc(randinit2,logPfuns,500000,'StepSize',2.2);

post2=walkers2(:,:,50:end); post2=post2(:,:); %Trim off burn-in

%Conver logged rate constants to time constants.
paramsconv=post2; paramsconv(islog,:)=10.^-post2(islog,:);

%Pull 100 random samples from the markov chain for plotting
samps=ceil(45100*rand(100,1));
for i=1:100
    [~,specsamps(:,:,i),datasamps(:,:,:,i)]=logPfuns{2}(post2(:,samps(i)));
end
```