100/100

Zarek McGee
MTH410- Optim 2
Final Project

Estimate of Distributed Source Parameters for a Nonlinear Dynamical System

Task 1 (with Code Listing):

```
%%%% Est. of Distributed Source Parameters for a Nonlin. Dynamical System
%%%%

% This Constrained Optimization problem can be reduced to an unconstrained
% problem by expressing the state vector u(m) in terms of alpha.

% load prdata1.m & prdata2.m before running
% INPUT: finalProj_zm(49,100,0.02,1,prdata1,prdata2)

function [a,lam,lam_pr2,f1,f2] = finalProj_zm(n,m,h,u0,prdata1,prdata2)

    x = linspace(h,1-h,n);

    %Choose initial condition
    if u0 == 1
        u0 = sin(2*pi*x);
    elseif u0 == 2
        u0 = sin(pi*x);
    end
    u0 = u0(:);

    % Create matrix A
    k = 0.5*h^2;
    s = k/(h^2); % s = 0.5
    A = full(gallery('tridiag',n,s,1-2*s,s));

    a0 = zeros(n,1);

    % Minimize fModel to obtain optimal alpha parameter
    [a] = fminunc(@(a) fModel(n,m,k,a,u0,A,prdata1), a0);

    % Calculate state values using optimal a* for u(:,m) at time m=100 and
m=200
    [u] = uMatrix(n,m,k,a,u0,A);
    [u_pr2] = uMatrix(n,200,k,a,u0,A);

    % Calculate Lagrange Multipliers
    [lam] = adjointModel(n,m,k,A,u,prdata1);
    [lam_pr2] = adjointModel(n,m,k,A,u,prdata2);
```

```
    % Calculate gradient of reduced function F w.r.t. alpha
    [agradF] = gradF(m,k,lam);
    [agradF_pr2] = gradF(m,k,lam_pr2);

    %fcost
    f1 = fcost(m,u,prdata1);
    f2 = fcost(200,u_pr2,prdata2);

    % Plot results
    figure(1)
    plot(a); xlabel("Model State"); ylabel("alpha");
    title("Spatial Distr. of alpha at time t(m)");

    figure(2);
    plot(prdata1,'o'); xlabel("Model State");
    hold on;
    plot(u(:,m)); legend("prdata1","u(m)");
    title("prdata1 vs. u(m) at time t(m)");

    figure(3);
    plot(prdata2,'o'); xlabel("Model State");
    hold on;
    plot(u_pr2(:,2*m)); legend("prdata2","u(2m)");
    title("prdata2 vs. u(m) at time t(2m)");

end


% Matrix u containing our state vectors at time 1:m
function [u] = uMatrix(n,m,k,a,u0,A)

    u = zeros(n,m);
    u(:,1) = (A * u0) - (k * u0.^3) + (k*a);
    for j=2:m
        u(:,j) = (A * u(:,j-1)) - (k * u(:,j-1).^3) + (k*a);
    end

end

% Backwards integration to obtain Lagrange Multipliers
function [lam] = adjointModel(n,m,k,A,u,prdata)

    lam = zeros(n,m);
    lam(:,m) = u(:,m) - prdata;
    for j=m-1:-1:1
        jcbnF = diag(3 * u(:,j).^2);
        lam(:,j) = [A - k * jcbnF] * lam(:,j+1);
    end

end

% Gradient of reduced function F w.r.t. alpha
function agradF = gradF(m,k,lam)
```

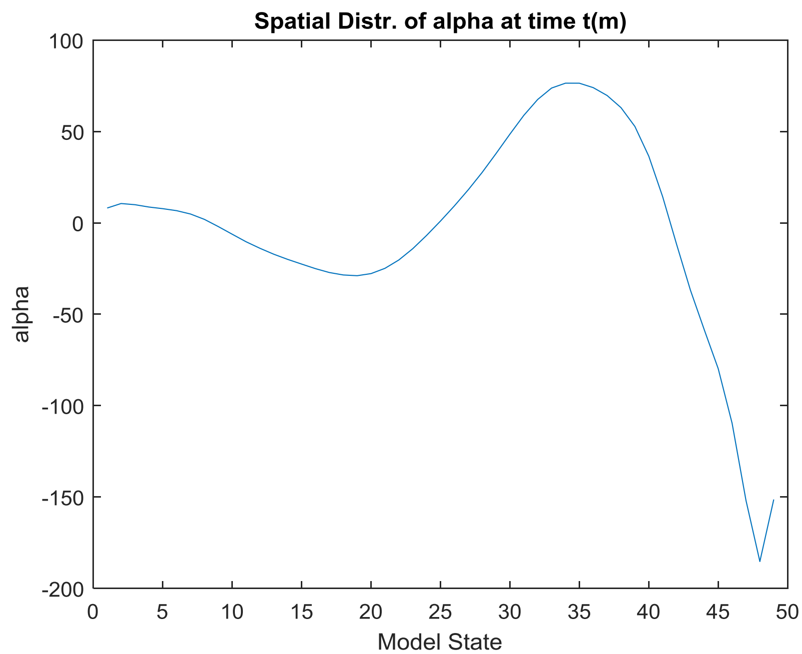
```
    sumlam = lam(:,1);
    for j=2:m
        sumlam = sumlam + lam(:,j);
    end

    agradF = k*sumlam;

end

% Model to minimize via fminunc
function [f,u] = fModel(n,m,k,a,u0,A,prdata)

    u = uMatrix(n,m,k,a,u0,A);
    f = 0.5 * norm(u(:,m) - prdata,2)^2;

end

% Cost function
function f = fcost(m,u,prdata)

    f = 0.5 * norm(u(:,m) - prdata,2)^2;

end
```

Task 2 (u_0 = sin(2*pi*x)):

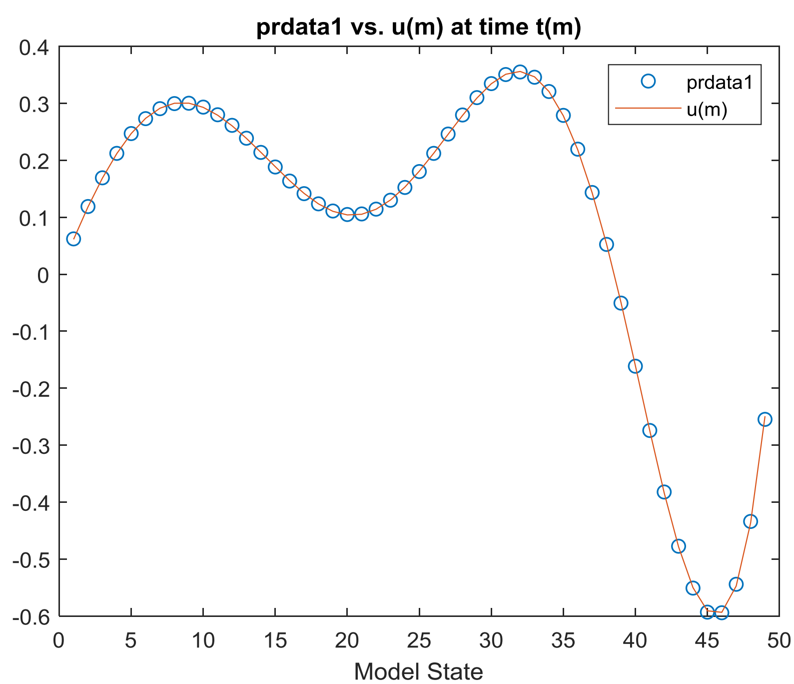
At time m=100, f(alpha*) = 3.6027e-05.

At time 2m, f(alpha*) = 3.6208e-05.

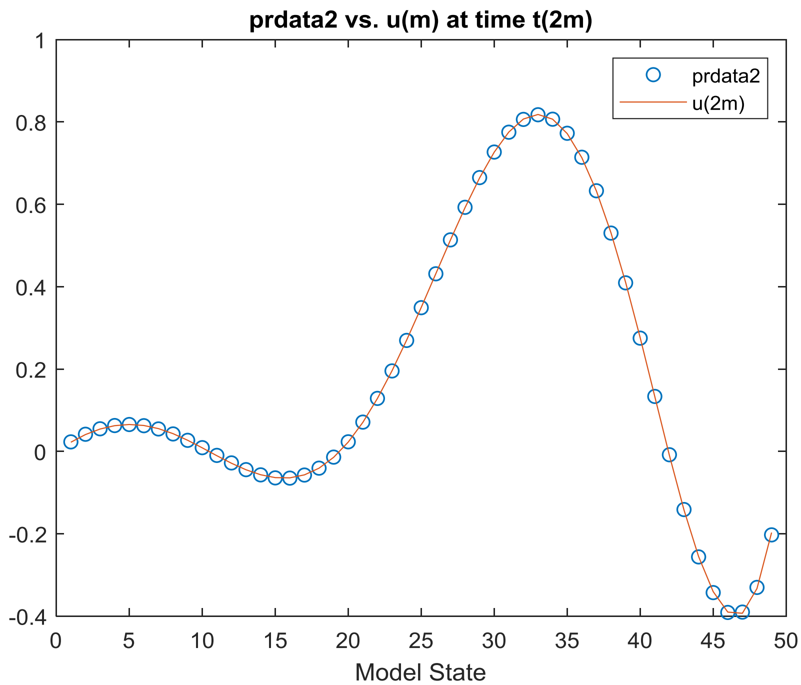
Spatial Distr. of optimal alpha parameter:



State Values at time m=100 vs. prdata1:



Validation State Values at time m=200 vs. prdata2:

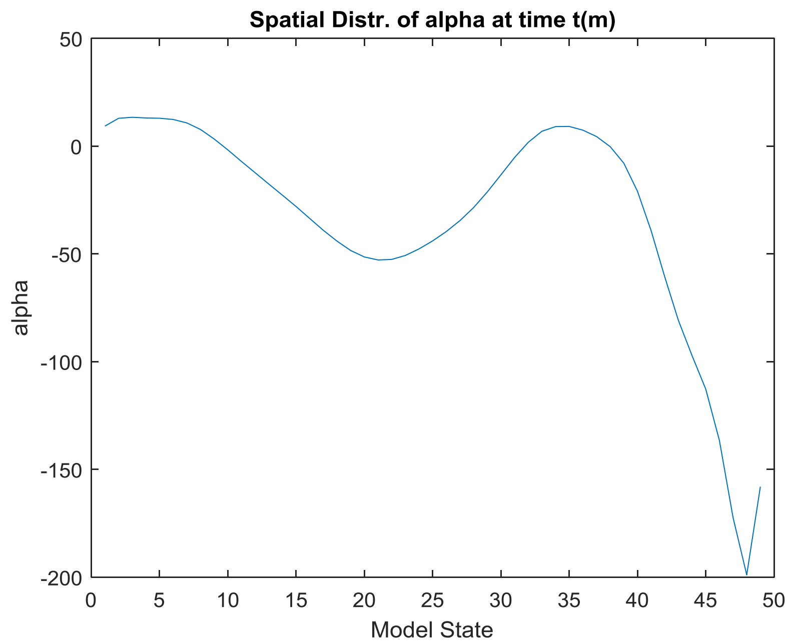


Task 3 ($u_0 = \sin(pi*x)$):

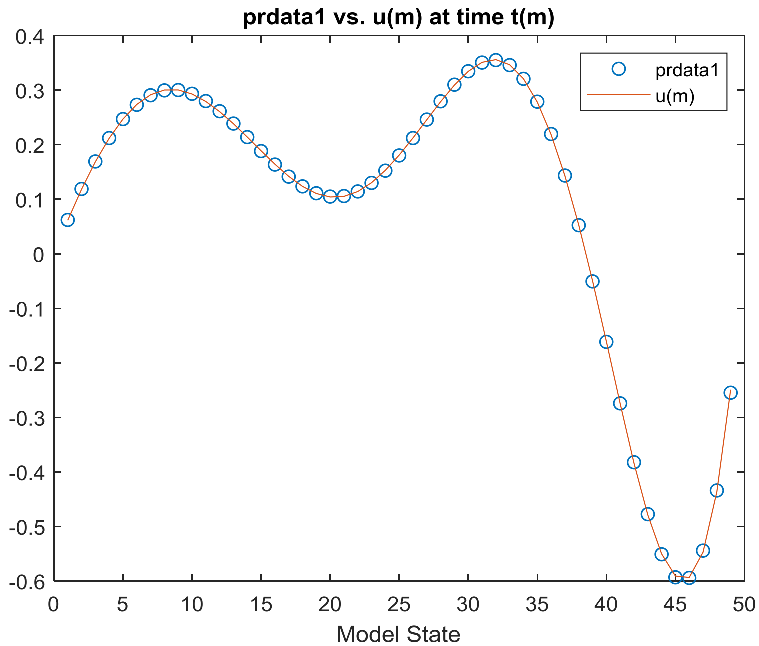
At time m=100, f(alpha*) = 3.6488e-05.

At time 2m, f(alpha*) = 10.8138. The initial condition $u_0(x) = \sin(pi*x)$ leads to a very poor result for the validation procedure at time 2m.

Spatial Distr. of optimal alpha parameter:



State Values at time m=100 vs. prdata1:



Validation State Values at time m=200 vs. prdata2: