This script reproduces Figure 7 of the revised manuscript entitled *Gaussian Process Emulation for High-Dimensional Coupled Systems,* which has the associated ID TCH-23-051.

In Figure 7, we apply the PPLE and the PPCE to a high dimensionoal, multiphysics model -- the two-permeability coupled fluid flow and mechanical deformation Terzaghi consolidation problem.

```
clear all;
close all;
```

We've added two extra figures to demonstrate that the results in the paper are robust, and do not depend on a particular traning/testing set. To include this check, set Robust_check to 1.

```
Robust_check=1;
```

The folder "functions" contains the RobustGaSP-in-Matlab package which can be found here:

https://github.com/MengyangGu/RobustGaSP-in-Matlab

Depending on your operating system, you may need to compile C++ files embedded in this package. Instructions to do so are here:

https://github.com/MengyangGu/RobustGaSP-in-Matlab/blob/master/README.md

```
addpath('functions');
T = 100; % Number of design runs
```

Data for experiment correpondsing to Figure 7: gall, fall, xxall, yyall

```
load Fig7data.mat
gall_t = gall(1:T,2:end);  % Pressure values
fall_t = fall(1:T,2:end);  % Porosity Values

% Design input
xxall_t = xxall(1:T,:);   % Input to the inside emulator (f)
yyall_t = yyall(1:T,:);   % Input to the outside emulator (g)
```

The code repeats the experiment twice, once using PCA for dimension reduciton on porosity, and once usinig gKDR.

```
for kk=1:2
    if kk==1
        dim_reduction = 'pca';
    elseif kk==2
        dim_reduction = 'gKDR';
    end

clear z;
```

gKDR script is from here: https://www.ism.ac.jp/~fukumizu/software.html

```
switch dim_reduction
```

```
    case 'pca'
        [z, num_modes,total_var_z] = pca_calculation(fall_t,0);
    case 'gKDR'
        num_modes = 4; %set to miniminum # of mode that produce low RMSE. 3-5 all work
        B =  gKDR(fall_t,gall_t,num_modes);
        z = fall_t*B;
end
```

This function fits a composite emulator (a PPE) to $\boldsymbol{\eta}(x, z)$directly.

```
[model] = composite__(xxall_t, gall_t);
```

This function fits a PPLE to $\boldsymbol{\eta}(x, z) = \mathbf{g}(z, \mathbf{f}(x))$. The expressions from Section 3 are implementted in intgasp()

```
[model_f,model_g] = intgasp(xxall_t,z,yyall_t,gall_t);
```

```
%preallocating for prediction experiment
error_le_2=zeros(T,101); true_le_2=error_le_2; predg_le_2=error_le_2;  stdg_le_2=error_
error_ce_2=error_le_2; true_ce_2=error_ce_2; predg_ce_2=error_ce_2;  stdg_ce_2=error_c
```

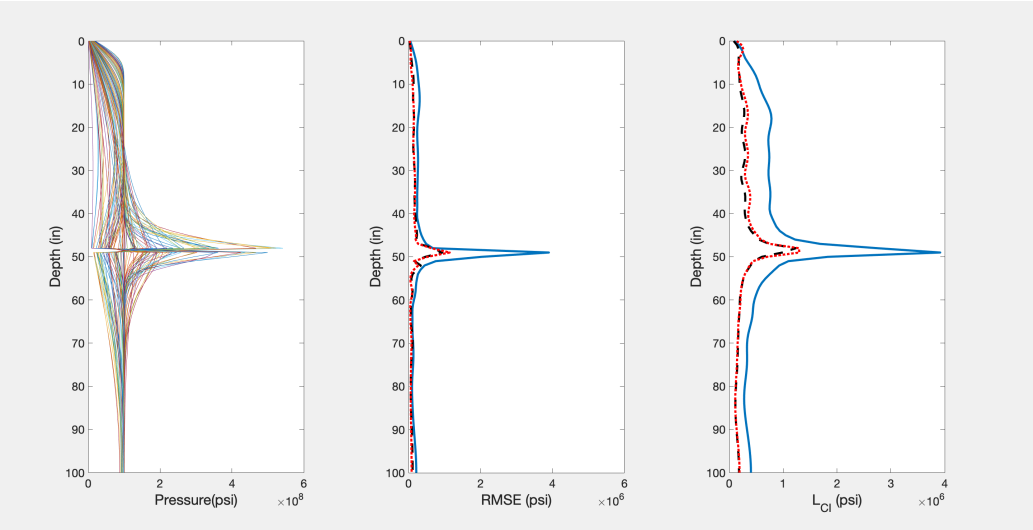
Predicting each of the pressure curves from the testing inputs.

```
for p = 1:(length(gall)-T)
    true_pres = gall(T+p,2:end); %True pressure values
    newu = xxall(T+p,:); %testing data for inside emulator
    newy = yyall(T+p,:); %testing data for the outside emulator

    % Make predictions using PPLE
    [pred_model] = predict_intgasp(model_f,model_g,newu,newy);
    % Make predictions using PPCE
    [mean_comp,stdg_comp] = pred_composite(xxall_t, newu, gall_t, model);

    % Extract prediction standard deviation and mean from linked
    % emulator
    stdg = sqrt(pred_model.var);
    predg = pred_model.mean;

    % Save error, true, predicted, and std deviation data
    [error_le_2(p,:),true_le_2(p,:),predg_le_2(p,:),stdg_le_2(p,:)] = save_data(predg,
    [error_ce_2(p,:),true_ce_2(p,:),predg_ce_2(p,:),stdg_ce_2(p,:)] = save_data(mean_c
end
 if kk==1
     error_le_PCA=error_le_2;
     stdg_le_PCA=stdg_le_2;
 elseif kk==2
     error_le_gKDR=error_le_2;
     stdg_le_gKDR=stdg_le_2;
```

```
 end
end
```

Function for plotting.

```
plot_figure7(true_le_2,error_le_PCA,error_le_gKDR,error_ce_2, stdg_le_PCA, stdg_le_gKDR
```



If Robust_check is off, the code stops here. Otherwise we repeat the experiment above (for PCA only as gKDR is a bit slower) 25 times, each time dividing up the training and testing data randomly.

```
if Robust_check==0
    return
end

Nsamp=25;
err_le_save1=zeros(Nsamp,101);
err_ce_save1=err_le_save1;
std_le_save1=err_le_save1;
std_ce_save1=err_ce_save1;

err_le_save2=zeros(Nsamp,101);
err_ce_save2=err_le_save1;
std_le_save2=err_le_save1;
std_ce_save2=err_ce_save1;

for kk=1:Nsamp
    %kk
    load Fig7data.mat
    newinds=randsample(200,200,false);
    %randomly mix up testing and training data
    gall=gall(newinds,:);
    xxall=xxall(newinds,:);
```

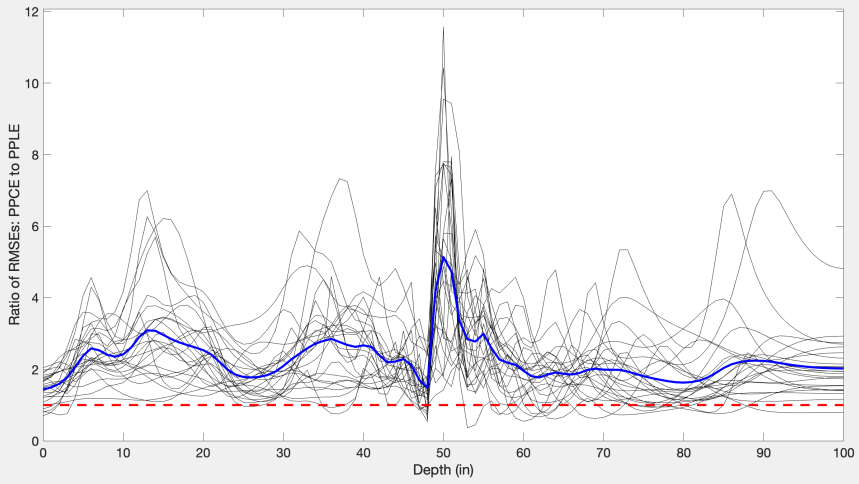
```
    yyall=yyall(newinds,:);
    fall=fall(newinds,:);

    % Extract initial T data points for design output
    gall_t = gall(1:T,2:end);  % Pressure values
    fall_t = fall(1:T,2:end);  % Porosity Values

    % Design input
    xxall_t = xxall(1:T,:);   % Input to the inside emulator (f)
    yyall_t = yyall(1:T,:);   % Input to the outside emulator (g)
    % return
    clear z;
    % Perform dimension reduction for porosity
    [z, num_modes,total_var_z] = pca_calculation(fall_t,0);

    % Construct the composite emulator
    [model] = composite__(xxall_t, gall_t );
    % Construct the linked emulator
    [model_f,model_g] = intgasp(xxall_t,z,yyall_t,gall_t);

    % Begin prediction loop
    for p = 1:(length(gall)-T)
        p;
        true_pres = gall(T+p,2:end); %True pressure values
        newu = xxall(T+p,:); %testing data for inside emulator
        newy = yyall(T+p,:); %testing data for the outside emulator

        % Make predictions using PPLE
        [pred_model] = predict_intgasp(model_f,model_g,newu,newy);
        % Make predictions using PPCE
        [mean_comp,stdg_comp] = pred_composite(xxall_t, newu, gall_t, model);

        % Extract prediction standard deviation and mean from linked
        % emulator
        stdg = sqrt(pred_model.var);
        predg = pred_model.mean;

        %the varying permeability case
        [error_le_2(p,:),true_le_2(p,:),predg_le_2(p,:),stdg_le_2(p,:)] = save_data(pr
        [error_ce_2(p,:),true_ce_2(p,:),predg_ce_2(p,:),stdg_ce_2(p,:)] = save_data(mea
    end

err_le_save2(kk,:)=sqrt(mean(error_le_2.^2));
err_ce_save2(kk,:)=sqrt(mean(error_ce_2.^2));
std_le_save2(kk,:)=mean(real(stdg_le_2));
std_ce_save2(kk,:)=mean(stdg_ce_2);
end
```

Below are two additional figures where we randomly mix up the training and testing data to demonstrate efficiency and robustness of the proposed PPLE method.

For each training/testing set, we plot the ratio of the RMSE of the composite emulator to the linked emulator (Figure 20). We also plot the ratio of the lengths of the 95% credible intervals, again composite to linked (Figure 30).

In each figure, for reference, we plot the constant one as a red dashed line (ratios larger than one indicating improvement of the PPLE over the PPCE), and the average of the ratios over the 25 samples of training/testing data in blue.

```
figure(20)
set(gcf, 'Position', get(0, 'Screensize'), 'Visible', 'on');
 plot(0:1:100,err_ce_save2'./err_le_save2','k')
 hold on
 plot(0:1:100,mean(err_ce_save2./err_le_save2),'b','linewidth',3)
 plot(0:1:100,ones(101,1), 'r--','linewidth', 3)
 axis([0 100 0 max(max(err_ce_save2'./err_le_save2'))+0.5])
 xlabel('Depth (in)')
ylabel('Ratio of RMSEs: PPCE to PPLE')
 ah=gca;
 set(ah,'fontsize',18)
```




```
 figure(30)
 set(gcf, 'Position', get(0, 'Screensize'), 'Visible', 'on');
 plot(0:1:100,std_ce_save2'./std_le_save2','k')
 hold on
 plot(0:1:100,mean(std_ce_save2./std_le_save2),'b','linewidth',3)
 plot(0:1:100,ones(101,1), 'r--','linewidth', 3)
 axis([0 100 0 max(max(std_ce_save2'./std_le_save2'))+0.5])
 xlabel('Depth (in)')
ylabel('Ratio of length of CIs: PPCE to PPLE')
 ah=gca;
```

```
set(ah,'fontsize',18)
```