```
% Take Home 6 part c - Anita Mezzetti
clear all
warning off;

rng(1)
s = rng;

% define set size
training_size = 2500;
testing_size = 5000;

% create sets X, y, X*
[Heston_train, Heston_test] = create_X_sets(training_size,
 testing_size);
Heston_price_train = create_y_set(training_size, Heston_train);

X = Heston_train;
y = Heston_price_train;
X_star = Heston_test;

% define input for fitGPR:
%K_type = 'squaredexponential';
theta0_se = [0.5 ,0.5];
bound_theta_se = [0.01, 0; 5, 10];
%
% %K_type = 'linear';
% theta0_l = [0.5, 0.5, 0.5];
% bound_theta_l = [0.01, 0.01, 0; 5, 5, 10];
%
% %K_type = 'periodic';
% theta0_p = [0.5 ,0.3, 0.5];
% bound_theta_p = [0.01, 0, 0; 5, 10, 10];
```

# squaredexponential

```
[max_lik_se, theta_opt_se, m_star_se, K_line_se] =...
    fitGPR(X, y, 'squaredexponential', theta0_se, bound_theta_se,
 X_star);
%
% %%% linear
% [max_lik_l, theta_opt_l, m_star_l, K_line_l] =...
%     fitGPR(X, y, 'linearkernel', theta0_l, bound_theta_l, X_star);
% fprintf("\n\nLinear Kernel:")
% fprintf("\nMaxima of marginal likelihood: %.3f ", max_lik_l)
% fprintf("\nOptimal hyper-parameters: sigma0=%.3f, sigma1=%.3f, c=
%.3f \n",...
%     theta_opt_l(1), theta_opt_l(2), theta_opt_l(3))
%
% %%% periodic
% [max_lik_p, theta_opt_p, m_star_p, K_line_p] =...
%     fitGPR(X, y, 'periodickernel', theta0_p, bound_theta_p, X_star);
% fprintf("\n\nPeriodic Kernel:")
```

```
% fprintf("\nMaxima of marginal likelihood: %.3f ", max_lik_p)
% fprintf("\nOptimal hyper-parameters: sigma0=%.3f, l=%.3f, p=%.3f
 \n",...
%     theta_opt_p(1), theta_opt_p(2), theta_opt_p(3))
%


% define the best kernel (max logp)
% if max_lik_p>max_lik_l && max_lik_p>max_lik_se
%     fprintf("\n\n periodickernel is the best kernel \n")
%     y_gpr_test = m_star_p;
% elseif max_lik_se>max_lik_l && max_lik_se>max_lik_p
%     fprintf("\n\n sqrdexp is the best kernel \n")
%     y_gpr_test = m_star_se;
% else
%     fprintf("\n\n linearkernel is the best kernel \n")
%     y_gpr_test = m_star_l;
% end

y_gpr_test = m_star_se;

gprMdl1 = fitrgp(Heston_train,
 Heston_price_train, 'KernelFunction', 'squaredexponential');
y_heston_test = predict(gprMdl1,Heston_test);

% errors
mae_error = max(abs(y_gpr_test - y_heston_test));
aae_error = mean(abs(y_gpr_test - y_heston_test));

% print errors
fprintf("\nMAE = %.5f \n", mae_error)
fprintf("AAE = %.5f", aae_error)


function [Heston_train, Heston_test] = create_X_sets(training_size,
 testing_size)

Heston_train = zeros(training_size,8);
Heston_test = zeros(testing_size,8);

% Training Set
strike= 0.4 + (1.6-0.4).*rand(training_size,1);
T= 11/12 + (1-11/12).*rand(training_size,1);
r= 0.015 + (0.025-0.015).*rand(training_size,1);
kappa= 1.4 + (2.6-1.4).*rand(training_size,1);
theta= 0.45 + (0.75-0.45).*rand(training_size,1);
rho= -0.75 + (-0.45+0.75).*rand(training_size,1);
sigma= 0.01 + (0.1-0.01).*rand(training_size,1);
v0= 0.01 + (0.1-0.01).*rand(training_size,1);

Heston_train(:,1)=strike(randperm(training_size));
Heston_train(:,2)=T(randperm(training_size));
Heston_train(:,3)=r(randperm(training_size));
Heston_train(:,4)=kappa(randperm(training_size));
```

```
Heston_train(:,5)=theta(randperm(training_size));
Heston_train(:,6)=rho(randperm(training_size));
Heston_train(:,7)=sigma(randperm(training_size));
Heston_train(:,8)=v0(randperm(training_size));

% Test Set
strike= 0.5 + (1.5-0.5).*rand(testing_size,1);
T= 11/12 + (1-11/12).*rand(testing_size,1);
r= 0.015 + (0.025-0.015).*rand(testing_size,1);
kappa= 1.5 + (2.5-1.5).*rand(testing_size,1);
theta= 0.5 + (0.7-0.5).*rand(testing_size,1);
rho= -0.7 + (-0.5+0.7).*rand(testing_size,1);
sigma= 0.02 + (0.1-0.02).*rand(testing_size,1);
v0= 0.02 + (0.1-0.02).*rand(testing_size,1);

Heston_test(:,1)=strike(randperm(testing_size));
Heston_test(:,2)=T(randperm(testing_size));
Heston_test(:,3)=r(randperm(testing_size));
Heston_test(:,4)=kappa(randperm(testing_size));
Heston_test(:,5)=theta(randperm(testing_size));
Heston_test(:,6)=rho(randperm(testing_size));
Heston_test(:,7)=sigma(randperm(testing_size));
Heston_test(:,8)=v0(randperm(testing_size));

end

function Heston_price_train = create_y_set(training_size,
 Heston_train)
% finding the prices in training set
Heston_price_train=zeros(training_size,1);

for i = 1:training_size
    strike=Heston_train(i,1);
    T=Heston_train(i,2);
    r=Heston_train(i,3);
    kappa=Heston_train(i,4);
    theta=Heston_train(i,5);
    rho=Heston_train(i,6);
    sigma=Heston_train(i,7);
    v0=Heston_train(i,8);
    S=1;
    alpha=2;

 Heston_price_train(i,1)=FFT_Heston(kappa,theta,sigma,rho,r ,v0,S,strike,T,alpha);
end

end
```

```
                                          First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0       1    7.987638e+05    0.000e+00    1.298e+16
    1      25    7.987266e+05    0.000e+00    1.298e+16    1.835e-07

Local minimum possible. Constraints satisfied.
```

```
fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.


MAE = 0.53746
AAE = 0.26384
```

*Published with MATLAB® R2020b*