PriorModelGeneration

May 15, 2016

1 Prior Model Generation

Author: Lewis Li **Date**: May 15th 2016

1.1 Introduction

In this notebook, we will demonstrate how we generated the prior models which are used as the input for PFA to build the statistical relationship between the data and forecast. The case study in question is based on a onshore reservoir in Libya, consisting of 5 compartments. Refer to this repository for details regarding the construction of the structural and depositional model. In our case study, we will consider 12 uncertain parameters with prior distributions shown below:

Symbol	Description	Prior Distribution
S_{wir}	Irreducible water saturation	$N(0.2, 0.05)$
S_{wor}	Irreducible oil saturation	$N(0.2, 0.05)$
k_{rw}^{end}	End point water relative permeability	$N(0.3, 0.1)$
k_{ro}^{end}	End point oil relative permeability	$N(0.7, 0.1)$
n_o	Oil Corey exponent	$N(2.5, 0.2)$
n_w	Water Corey exponent	$N(2.0, 0.2)$
F_1	Fault 1 transmissibility multiplier	$U[0.20.8]$
F_2	Fault 2 transmissibility multiplier	$U[0.20.8]$
F_3	Fault 3 transmissibility multiplier	$U[0.20.8]$
F_4	Fault 4 transmissibility multiplier	$U[0.20.8]$
ν_o	Oil viscosity	$N(4, 0.2)$
OWC	Oil water contact depth	$U[1061, 1076]$

In [1]:
```
% ParameterRanges is a struct that contains
Normal = 0;
Uniform = 1;

PriorParameterDistribution = struct();

% Irreducible water saturation
PriorParameterDistribution.('Swir') = [0.2 0.05 Normal];

% Irreducible oil saturation
PriorParameterDistribution.('Swor') = [0.2 0.05 Normal];

% End point water rel perm
PriorParameterDistribution.('krw_end') = [0.3 0.1 Normal];

% End point oil rel perm
```

```
PriorParameterDistribution.('kro_end') = [0.7 0.1 Normal];

% Oil Corey exponent
PriorParameterDistribution.('no') = [2.5 0.2 Normal];

% Water Corey exponent
PriorParameterDistribution.('nw') = [2 0.2 Normal];

% Fault 1 trans multiplier
PriorParameterDistribution.('FaultMulti1') = [0.2 0.8 Uniform];

% Fault 2 trans multiplier
PriorParameterDistribution.('FaultMulti2') = [0.2 0.8 Uniform];

% Fault 3 trans multiplier
PriorParameterDistribution.('FaultMulti3') = [0.2 0.8 Uniform];

% Fault 4 trans multiplier
PriorParameterDistribution.('FaultMulti4') = [0.2 0.8 Uniform];

% Oil viscosity
PriorParameterDistribution.('Viscosity') = [4 0.2 Normal];

% Oil water contact
PriorParameterDistribution.('OWC') = [1061 1076 Uniform];
```

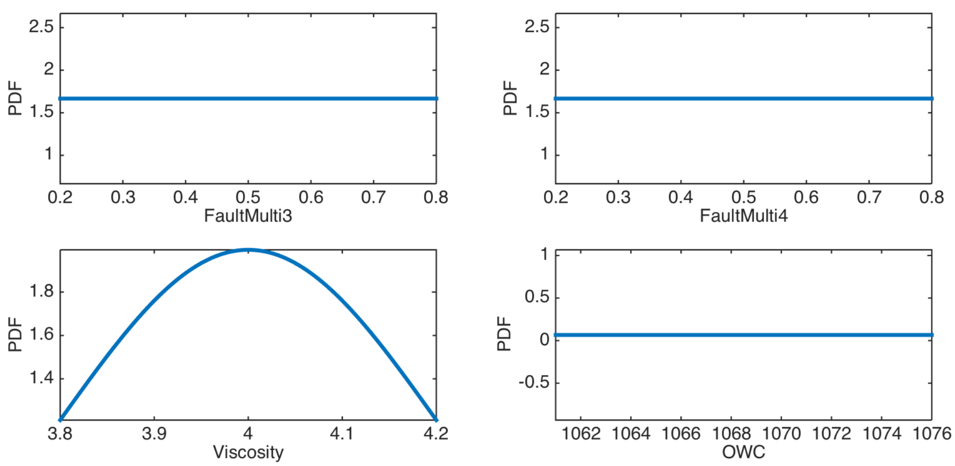
We will generate 500 prior models for the 12 realizations.

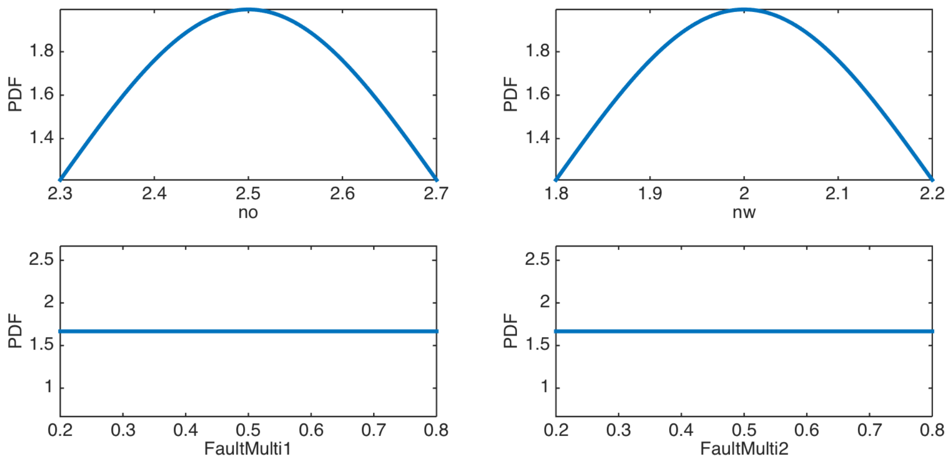
```
In [2]: % Case Name
        CaseName = 'Prior';

        % Number of simulations
        NbSimu = 500;

        % Number of parameters we will vary
        NbParams = 12;

        % Set random seed
        rng('shuffle');
```

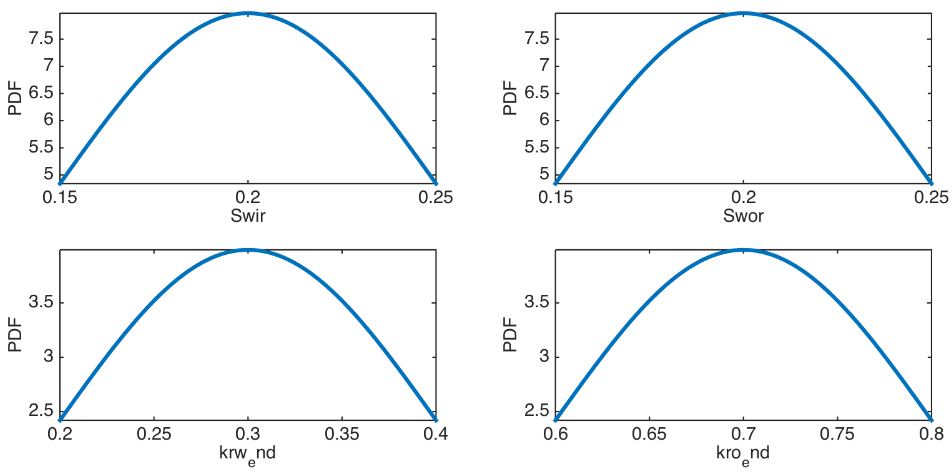
We can plot the prior distributions for each parameter below:

```
In [5]: %plot inline -s 1800,800
        ParameterNames = fieldnames(PriorParameterDistribution);

        rng('shuffle');

        % Iterate over each uncertain reservoir parameter
        for i = 1:numel(ParameterNames)
            ParameterName = ParameterNames{i};
            ParameterRange = PriorParameterDistribution.(ParameterName);

            % Plot prior values
            figureid = i-floor((i-1)/4)*4;
            if (mod(i-1,4) == 0)
```

```
        figure(floor(i/4)+1);
        figurepic=floor(i/4)+1;
    end

    subplot(2,2,figureid);

    if (ParameterRange(3) == Normal)
        xx = linspace(ParameterRange(1)-ParameterRange(2),...
            ParameterRange(1)+ParameterRange(2));
        yy = normpdf(xx,ParameterRange(1),ParameterRange(2));
    elseif (ParameterRange(3) == Uniform)
        xx = linspace(ParameterRange(1),ParameterRange(2));
        yy = ones(size(xx))/(ParameterRange(2)-ParameterRange(1));
    end

    FontSize = 14;
    plot(xx,yy,'LineWidth',3);
    set(gcf,'color','w');
    set(gca,'FontSize',FontSize);
    xlabel(ParameterName,'FontSize',FontSize);
    ylabel('PDF','FontSize',FontSize);
    axis tight;
end
```





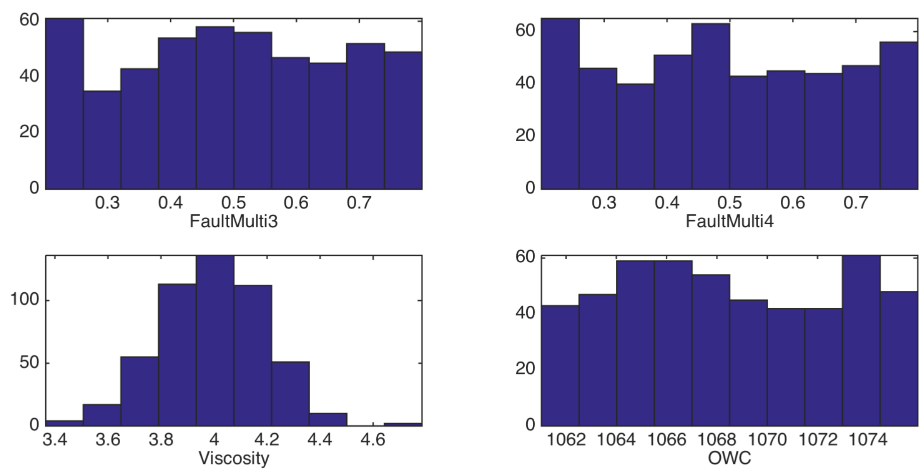


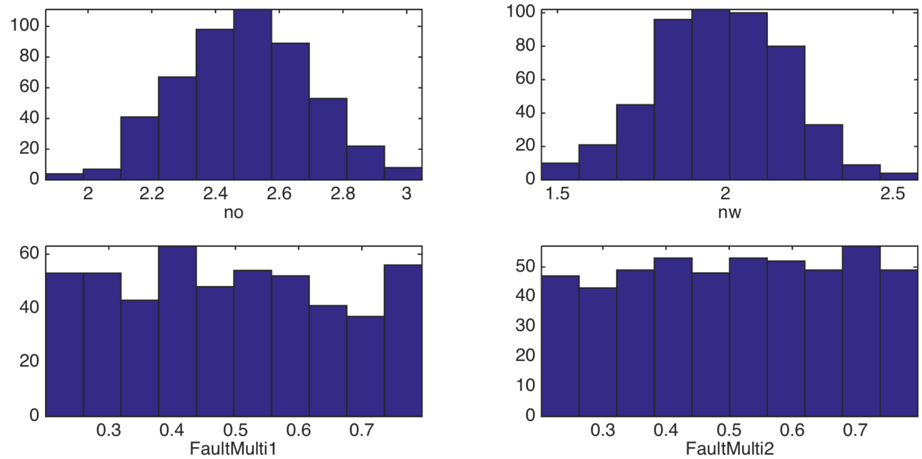
1.2 Sampling the Prior Distributions

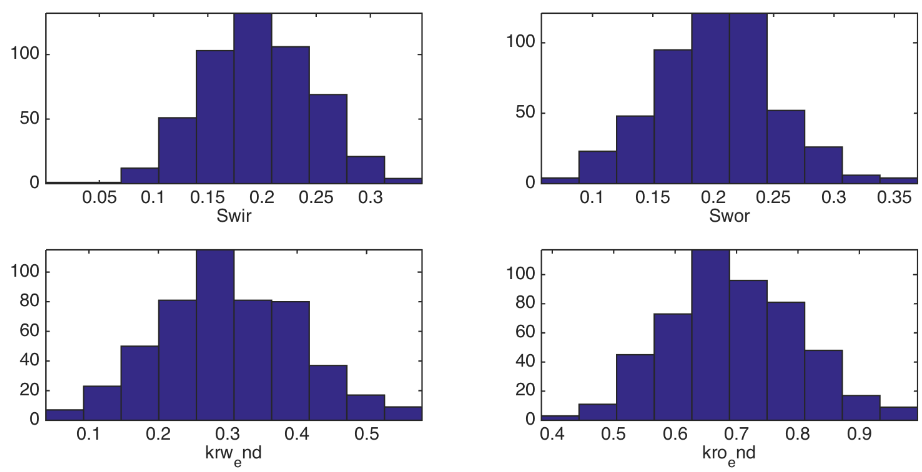
We can then sample from the prior distributions to generate the prior models

In [6]:
```
rng('shuffle');
PriorModelParameters = struct();
% Iterate over each uncertain reservoir parameter
for i = 1:numel(ParameterNames)
    ParameterName = ParameterNames{i};
    ParameterRange = PriorParameterDistribution.(ParameterName);

    % Sample from uniform/normal distributions
    if (ParameterRange(3) == Normal)
        Value = ParameterRange(1) + ParameterRange(2)*randn(NbSimu,1);
    elseif(ParameterRange(3) == Uniform)
        Value = ParameterRange(1) + rand(NbSimu,1)*...
```

```
            (ParameterRange(2) - ParameterRange(1));
    end

    % Store sampled value
    PriorModelParameters.(ParameterName) = Value;

    % Plot prior values
    figureid = i-floor((i-1)/4)*4;
    if (mod(i-1,4) == 0)
         figure(floor(i/4)+1);
         figurepic=floor(i/4)+1;
    end

    subplot(2,2,figureid);

    FontSize = 14;
    hist(Value);
    set(gcf,'color','w');
    set(gca,'FontSize',FontSize);
    xlabel(ParameterName,'FontSize',FontSize);
    axis tight;

end
```





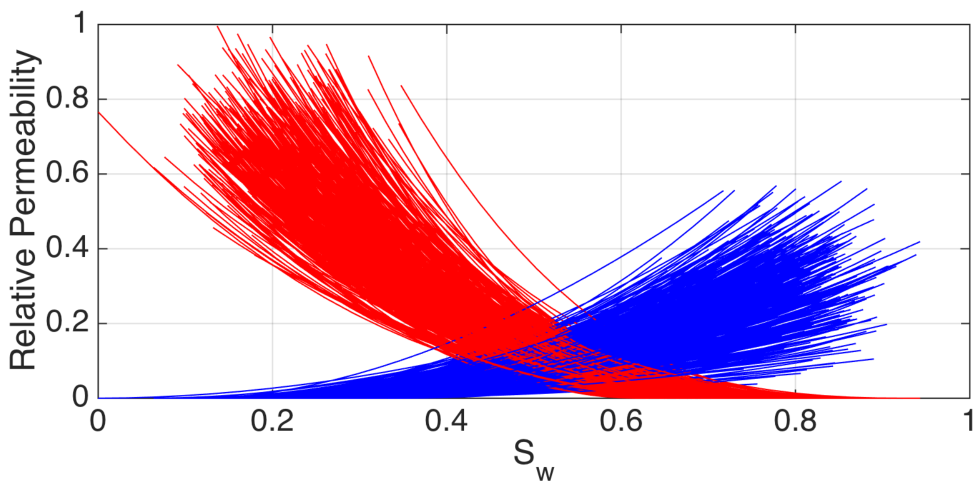


1.3 Corey Curve Construction

For each of the sampled models, we will construct the relative permeability using the Corey expressions.

In [7]:
```
%% Construct relative permeability curves using Corey parameters
clear SW_corey_m;
warning('off','all')
RelPermEntries = 25;
SW_corey_m = zeros(NbSimu,RelPermEntries);
krw_model = zeros(NbSimu,RelPermEntries);
kro_model = zeros(NbSimu,RelPermEntries);

for i=1:NbSimu
    SW_corey_m(i,:) = linspace(PriorModelParameters.('Swir')(i), ...
        1 - PriorModelParameters.('Swor')(i),RelPermEntries);
end
```

```
hCorey = figure;
for i=1:NbSimu
    krw_model(i,:) = PriorModelParameters.('krw_end')(i) .* ...
        ((SW_corey_m(i,:)-PriorModelParameters.('Swir')(i))./...
        (1-PriorModelParameters.('Swir')(i)-...
        PriorModelParameters.('Swor')(i))).^PriorModelParameters.('nw')(i);
    kro_model(i,:) = PriorModelParameters.('kro_end')(i) .* ...
        ((1 - SW_corey_m(i,:) - PriorModelParameters.('Swor')(i))./...
        (1 - PriorModelParameters.('Swir')(i) - PriorModelParameters.('Swor')(i))).^...
        PriorModelParameters.('no')(i);

    plot(SW_corey_m(i,:), krw_model(i,:),'b-', SW_corey_m(i,:), ...
        kro_model(i,:), 'r-');
    xlabel('S_w');
    ylabel('Relative Permeability');
    xlim([0 1]); ylim([0 1]); grid on; hold on;
end
set(gcf,'color','w');
set(gca,'FontSize',24);
```



1.4 Generating the Simulation Decks

We will finally generate a simulation file for each prior model by using a baseline simulation deck as the starting point.

```
In [ ]: %% Load baseline simulation deck for 3DSL
        % Path to baseline case
        BaseCaseDatPath = '../../data/3DSLFiles/CompressibleBaseCase.dat';

        % Allocate a cell array that we will use to store the baseline
        s=cell(GetNumberOfLines(BaseCaseDatPath),1);

        fid = fopen(BaseCaseDatPath);
        lineCt = 1;
```

```
tline = fgetl(fid);

while ischar(tline)
    s{lineCt} = (tline);
    lineCt = lineCt + 1;
    tline = fgetl(fid);
end

% Directory to store output simulation decks
OutputDirectory = ['/media/Scratch2/Data/3DSLRuns/Compressible/' ...
    TrialName '/'];

% Generate a seperate deck for each
for k=1:NbSimu

    FolderNameIteration = [OutputDirectory 'Run', num2str(k)];

    %creating an new folder for this iteration
    %checking if there is already a folder with that name
    if exist(FolderNameIteration,'dir') ~= 7
        mkdir(FolderNameIteration);
    end

    % Create new file
    file_name = [FolderNameIteration '/Run', num2str(k), '.dat'];
    fileID = fopen(file_name,'w+');

    % Loading everything before the Faultmultiplier
    MULTFLTIndex = SearchCellArray('MULTFLT',s);
    for j=1:MULTFLTIndex
        fprintf(fileID,'%c',s{j});
        fprintf(fileID,'\n');
    end


    % Fault Multiplier
    F1Mult=['fault_1' blanks(1) ...
        num2str(PriorModelParameters.('FaultMulti1')(k)) blanks(1) '/'];
    F2Mult=['fault_2' blanks(1) ...
        num2str(PriorModelParameters.('FaultMulti2')(k)) blanks(1) '/'];
    F3Mult=['fault_3' blanks(1) ...
        num2str(PriorModelParameters.('FaultMulti3')(k)) blanks(1) '/'];
    F4Mult=['fault_4' blanks(1) ...
        num2str(PriorModelParameters.('FaultMulti4')(k)) blanks(1) '/'];

    fprintf(fileID,'%s',F1Mult);
    fprintf(fileID,'\n');
    fprintf(fileID,'%s',F2Mult);
    fprintf(fileID,'\n');
    fprintf(fileID,'%s',F3Mult);
    fprintf(fileID,'\n');
    fprintf(fileID,'%s',F4Mult);
    fprintf(fileID,'\n');
    fprintf(fileID,'/ \n');
```

```
PVMULTIndex = SearchCellArray('PVMULT',s);
CVISCOSITIESIndex = SearchCellArray('CVISCOSITIES',s);

% Writing everything before Viscosity
for j=PVMULTIndex:CVISCOSITIESIndex
    fprintf(fileID,'%c',s{j});
    fprintf(fileID,'\n');
end

% Printing the PVTs out - in this case only viscosity is changed
Visc=[num2str(PriorModelParameters.('Viscosity')(k)) blanks(1) '0.1 0.4 /'];
fprintf(fileID,'%s',Visc);

KRWOIndex = SearchCellArray('KRWO',s);


% Writing everything before Viscosity
for j=CVISCOSITIESIndex+1:KRWOIndex
    fprintf(fileID,'%c',s{j});
    fprintf(fileID,'\n');
end

% Writing out the Relperms

formatSpecRelPerm = '%4.4f %4.8f %4.8f %s\n';
fprintf(fileID,'%s\n', '--    Sw        krw       kro      Pc');
for j=1:RelPermEntries
    fprintf(fileID,formatSpecRelPerm,SW_corey_m(k,j),...
        krw_model(k,j),kro_model(k,j));
    fprintf(fileID,'\n');
end
fprintf(fileID, '/\n');
fprintf(fileID,'%s\n', 'END RELPERMS');
%check if the slash works in the simulation

INITIALCOND = SearchCellArray('INITIALCOND',s);
OWC = SearchCellArray('OWC',s);

% Writing everything before OWC
for j=INITIALCOND:OWC
    fprintf(fileID,'%c',s{j});
    fprintf(fileID,'\n');
end

% Writing out the OWC
OWCValue=['-', num2str(PriorModelParameters.('OWC')(k))];
fprintf(fileID,'%s',OWCValue);
fprintf(fileID,'\n');
fprintf(fileID, '/\n');

EndInitialCondition=SearchCellArray('END INITIALCOND',s);
```

```
        % Writing out the rest
        for j=EndInitialCondition:GetNumberOfLines(BaseCaseDatPath)
            fprintf(fileID,'%c',s{j});
            fprintf(fileID,'\n');
        end

        fclose(fileID);
    end
```

```
In [ ]:
```