Manual for DGSA (Distance based global sensitivity analysis) Matlab toolbox

Author: Jihoon Park (jhpark3@stanford.edu)

Lastly updated: May 31, 2016

1. Overview

This manual is to 1) help users apply DGSA on their own data to obtain sensitivities, 2) reproduce results from the paper written by Park et al. (2016): *DGSA: a matlab toolbox for distance-based generalized sensitivity analysis for geoscientific computer experiments*. This manual focuses on the implementation of DGSA so it is recommended to consult with the paper for related theories. All the code is uploaded at https://github.com/SCRFpublic/DGSA.

2. Structure of the program

The program has 4 main scripts – 1) *main_DGSA_analytic.m*, 2) *main_DGSA_Reservoir_Sensitivity.m*, 3) *main_DGSA_ParameterUncertaintyReduction.m* and 4) *main_compareDGSA_Sobol.m* .

The first script, *main_DGSA_analytic.m* has an analytic example originally written by Céline Scheidt. The forward model is simply a bivariate function that helps users understand and utilize DGSA. The rest of the scripts reproduce results from Park et al. (2016). The second script, *main_DGSA_Reservoir_Sensitivity.m* has an example of applying DGSA to reservoir responses. Both main and conditional effects are computed and visualized. In addition, responses are plotted according to the classes they belong to.

In the third script, m*ain_DGSA_ParameterUncertaintyReduction.m*, uncertainty of parameters is reduced by decreasing uncertainty of insensitive parameters based on net conditional effects. The third script uses the variables from the second script – they are separated for convenience. Thus, if a user wants to perform reduction of parameter uncertainty, the second script *main_DGSA_Reservoir_Sensitivity* should be run first and all the results should be saved.

The fourth script, *main_compareDGSA_Sobol.m* validates the results of DGSA by comparing to Sobol's indices. If a user only needs to apply DGSA, he or she does not need to run this script.

3. Subdirectories

The program has 7 subdirectories containing functions and data files.

DGSA_computations: Scripts regarding computations of DGSA.
Utilities: Scripts needed for data processing.
InputData: Input data for the example.

Visualizations: Scripts to visualize the results.

Sobol_DGSA_comparision: Scripts regarding computations of Sobol's indices.

VariablesSaved: Saved variables.

KPCASOM: Scripts to compute ranks of spatial models. This folder contains the example of computing ranks of spatial models from gradual deformation as discussed in Park et al. (2016).

4. Run DGSA

4.1 Summary of variables.

All information related to DGSA is given and saved at the data structure named *DGSA* in the script *main_DGSA_Reservoir_Sensitivity.m* and *main_DGSA_ParameterUncertaintyReduction.m*. Section 4.1.1 and 4.1.2 provide the summary of variables with brief descriptions.

4.1.1 Inputs specified by a user

DGSA

.ParametersNames: (cell, length= # of parameters) Names of parameters loaded from *PriorParameters.dat* or *PriorParameters_Spatial.dat* in InputData directory.

.ParametersValues: (matrix, # of models x # of parameters) Values of each parameter from Monte Carlo sampling. This is also loaded from 'PriorParametres.dat' or 'PriorParameters_Spatial.dat' in InputData directory. Missing data should be filled with NaN.

.Nbcluster: (scalar) # of clusters for k-medoid clustering.

.D: (matrix, # of models x # of models or vector, length =# of models x (# of models -1)): Distance matrix or vector of distances between the model.

DGSA.MainEffects.Display

.ParetoPlotbyCluster: (logical, length=1) If it is true, a Pareto plot to display main effects by each cluster and parameter is displayed.

.StandardizedSensitivity: (string) The method to visualize standardized main effects can be chosen with the variable. This can have three string values - 'Pareto (Pareto plot)','CI(Pareto plot with confidence interval)','None'.

DGSA.ConditionalEffects

.NbBins: (vector, length=# of parameters) Specifies each number of bins of parameters to compute conditional effects.

PriorResponses: (cell, # of wells x # of models) Each cell has a matrix of reservoir responses.

Responses_AfterPUR: (cell, # of wells x # of models) Each cell has a matrix of reservoir responses after reduction of parameter uncertainty

SigLevel_CI: (vector, length=3) Specifies the quantiles for confidence interval. This variable is needed only when a confidence interval is displayed.

Parameter2Reduce: (string, length=1) Specifies the name of parameter whose uncertainty will be reduced.

DGSA. ConditionalEffects.Display

. *SensitivityByClusterAndBins*: (logical) If true, a Pareto plot is displayed to visualize conditional effects from each bin of parameter and each class.

.*StandardizedSensitivity*:(logical) Specifies the method to visualize standardized conditional effects. It can have 'Pareto(default)', 'Hplot', and 'Bubble'.

4.1.2 Outputs

DGSA

.*N:* (scalar) # of models (sample size)

DGSA.Clustering

.*T*: (vector, length=# of models) Class which each model belongs to.

.*medoids*: (vector, length=# of clusters) Indices of medoids.

. *weights*: (vector, length=# of clusters) # of models that belong to each class.

If users have their own data for clustering, the results should be re-organized and saved at *DGSA.Clustering* as described above.

DGSA.MainEffects.

. *SensitivitybyCluster*: (matrix, # of parameters x # of clusters) Main effects by parameters (rows) and clusters (columns)

.*SensitivityStandardized*: (vector, length=# of parameters) Main effects averaged over clusters.

. *H0accMain*: (logical, length=# of parameters) Result of bootstrap hypothesis test. If true, a parameter is sensitive.

DGSA. ConditionalEffects.

.*Names_ConditioanlEffects*: (cell, length=(# of parameters) x (# of parameters-1)) Names of conditional effects.

.SensitivityByClusterandBins: (4D array, (# of parameters) x (# of parameters-1) x (# of clusters) x (maximum value of bins)): Element (i, j, c, l) corresponds to conditional effect of i-th parameter conditioned to l-th bin of j-th parameter from class c. This corresponds to $\hat{d}^{S}_{c,i|j,l}$ in Eq. (7) from Park et al. (2016).

.StandardizedSensitivity: (vector, length=(# of parameters) x (# of parameters -1)) Conditional effects averaged over bins and classes.

.H0accConditional: (logical vector, length=(# of parameters) x (# of parameters -1)) If it is true, corresponding conditional effect is sensitive.

NetConditionalEffects: (matrix, (# of bins of specified parameter) x (# of parameters-1)) Net conditional effects of specified parameter.

Bin_MaxNetConditional: (scalar) Index of bin of maximum net conditional effects.

4.2 Load input data

Inputs for DGSA can be loaded as:

```
%% 2. Load & process input data
load('PriorResponses'); % load responses
load('DistanceMatrix'); % load distance matrix. Distance matrix can be
computed from pdist() function in Matlab.
                        % For details,refer to main_DGSA_analytic.m and the
manual included.
PriorResponses=WWPTbyWell;
DGSA=ProcessInputParameters('PriorParameters.dat'); % Process Input
parameters
%DGSA=ProcessInputParameters('PriorParameters_Spatial.dat'); % This data
includes two more parameters - permeability and porosity.


DGSA.D=D;
```

```
clear WWPTbyWell;clear D;
```

Here, Monte Carlo samples from prior distributions of parameters (*PriorParameters.dat*) and corresponding flow responses (*PriorResponses.mat*) are loaded. There are two files containing values of parameters: *PriorParameters.dat* and *PriorParameters_Spatial.dat*. They are identical except that the latter one has two more uncertain parameters: *perm* (permeability) and *poro* (porosity). The two parameters take ranks as values when applying DGSA. Those ranks are from applying SOM (self-organizing map) and KPCA (Kernel principle component analysis) to permeability and porosity.

Distance matrix was previously computed and is also loaded. This is because in Park et al.(2016)'s case, *dtmax*, which is the maximum time step at which a streamline simulator (3DSL) generates streamline, is also one of the uncertain parameters. Since time steps varied by models, distance matrix was computed separately by including all time steps available.

If time steps are the same for all the models, distance matrix can be simply computed from *pdist* command in Matlab (See *main_DGSA_analytic.m).* For example, if the response *r* is the matrix (# of models x (# of time steps X # of wells)), then you can simply compute distance as :

```
D=pdist(r)
```

In the program, data structure named *DGSA* is used to specify inputs and save results from DGSA.

4.3 Compute and visualize main effects

In order to run DGSA, clustering should be performed first. In the example, k-medoid clustering is directly applied to the distance matrix. The number of the cluster is set to 3. This can be changed by the variable *DGSA.Nbcluster.*

Main effects can be computed and visualized by the script *ComputeMainEffects*.m . The script takes the data structure *DGSA* as input and users can specify how they visualize main effects. If *DGSA.MainEffects.Display.ParetoPlotbyCluster*=1, a Pareto plot showing main effects by clusters is displayed as shown in Fig. 1.

Main effects averaged over clusters (standardized main effect) can be visualized in two methods. The first one is to use a Pareto plot (Fig. 2) and the second is to add a confidence interval and color the bars according to its relative location to the interval (Fig. 3). This is specified by assigning the string variable *DGSA.MainEffects.Display.StandardizedSensitivity* to 'Pareto', 'CI', or 'None', respectively.

If a confidence interval is added to the plot, the quantile of the distribution from bootstrap sampling should be assigned to the variable *SigLevel_CI*. When 'Pareto' or 'None' is entered, this variable from the inputs of *ComputeMainEffects*.m should be removed.

```
%% 4. Compute & display main effects
```

```
% 4.1 Specify variables
DGSA.Nbcluster=3;
DGSA.MainEffects.Display.ParetoPlotbyCluster=1; % if true, main effects over
cluster will be displayed with Pareto plot.
DGSA.MainEffects.Display.StandardizedSensitivity='CI';

% if 'CI', the confidence interval will be overlapped on the Pareto plot
% if 'Pareto', Pareto plots will be displayed only
% if 'None' No plot will be generated for standard main effects
% (default)

% 4.2 Compute main effects from DGSA.
SigLevel_CI=[.95,.9,1]; % This is needed only when you want to display
confidence intervals
                        % If you do not need to display confidence
intervals,set DGSA.MainEffects.Display.StandardizedSensitivity='Preto' or
'None'

DGSA=ComputeMainEffects(DGSA,SigLevel_CI);
```
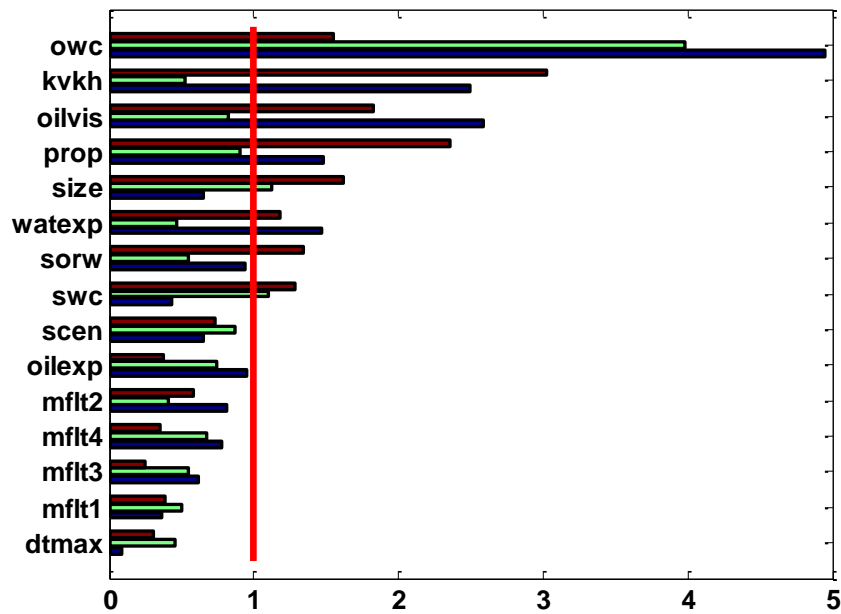

Fig. 1 Main effects of each parameter by cluster
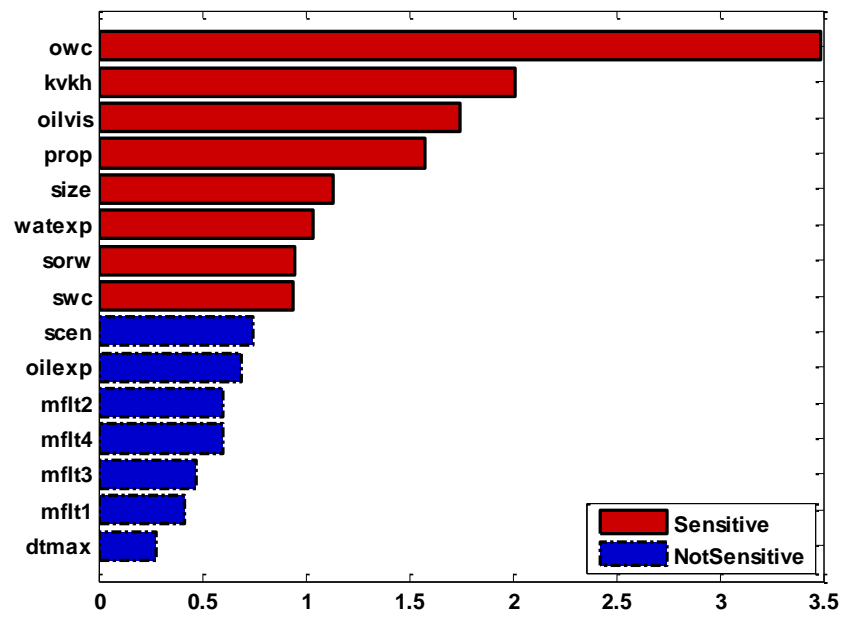
Fig. 2 Pareto plot of standardized main effects
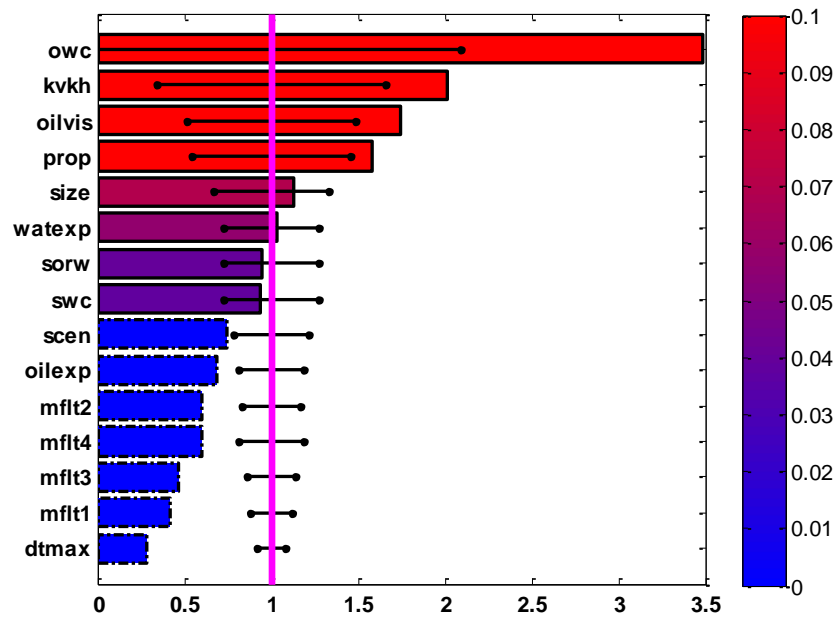
Fig. 3 Pareto plot with confidence intervals

CDFs that are used to compute main effects can be displayed by *cdf_MainFactor.m*. You can specify the name of parameters in the function. For example, the following example displays CDFs of parameters *owc* (oil-water contact) and *oilvis* (viscosity of oil), see Fig. 4.

```
%% Display cdfs

cdf_MainFactor(DGSA.ParametersValues, DGSA.Clustering,
DGSA.ParametersNames,{'owc','oilvis'});
```
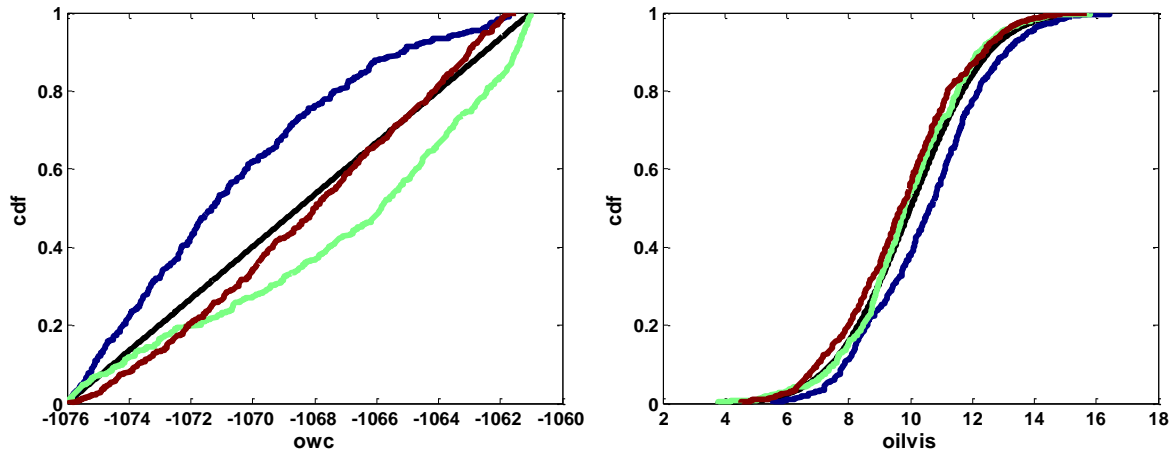

Fig. 4 CDFs of owc and oilvis

4.4 Compute and visualize conditional effects

In order to compute conditional effects, the number of bins for each parameter should be determined. In this example, it is set to 3 for every parameter as:

```
DGSA.ConditionalEffects.NbBins=3*ones(1,length(DGSA.ParametersNames));
```

Conditional effects can be computed by *ComputeConditionalEffects.m* and displayed by *DisplayConditionalEffects.m*. The program offers three ways to display standardized conditional effects and this can be selected by the variable *DGSA.ConditionalEffects.Display.StandardizedSensitivity*. If it is set to 'Pareto', a Pareto plot will be displayed. However, if the number of parameter increases, using a Pareto plot is not practical, because if there are k parameters, k(k-1) conditional effects exist. If the variable is set to be 'Bubble' or 'Hplot', a bubble plot or a h-plot will be displayed.

For both bubble plot and h-plot, the radius of each bubble is proportional to the main effects and it can be scaled by some positive value. In order to intuitively visualize the relationship between parameters, trying several values of scales are needed. To make this easier, input dialog pops up for bubble plot and h-plot, see Fig. 5. The results of h-plot and bubble plot are shown in Fig. 6 and 8, respectively. Fig. 7 shows the ratio of cumulative sum of eigenvalues to the sum of all eigenvalues in order to show the proportion of the variance explained in h-plot.

8

```matlab
%% 5. Compute & Display conditional effects

% 5.1 Specify additional variables to estimate conditional effects.

DGSA.ConditionalEffects.NbBins=3*ones(1,length(DGSA.ParametersNames));

% 5.2 Compute conditional effects

rng('default');
DGSA=ComputeConditionalEffects(DGSA);

% 5.3 Display conditional effcts

% Speicify the method to display standardized conditional effects.
DGSA.ConditionalEffects.Display.SensitivityByClusterAndBins=1; % if true,
display pareto plots to visualize sensitivities by bins/clusters.
DGSA.ConditionalEffects.Display.StandardizedSensitivity='Hplot'; % If
omitted Pareto plot will be used. However, this is not recommended when
there are lots of parameters

% Visualize conditional effects
DisplayConditionalEffects(DGSA,DGSA.ConditionalEffects.Display.StandardizedS
ensitivity)

% 5.4 Display class condtional CDFs
cdf_ConditionalEffects('owc','kvkh',DGSA,1)
```


Fig. 5 Input dialog to enter scale of bubble and font size
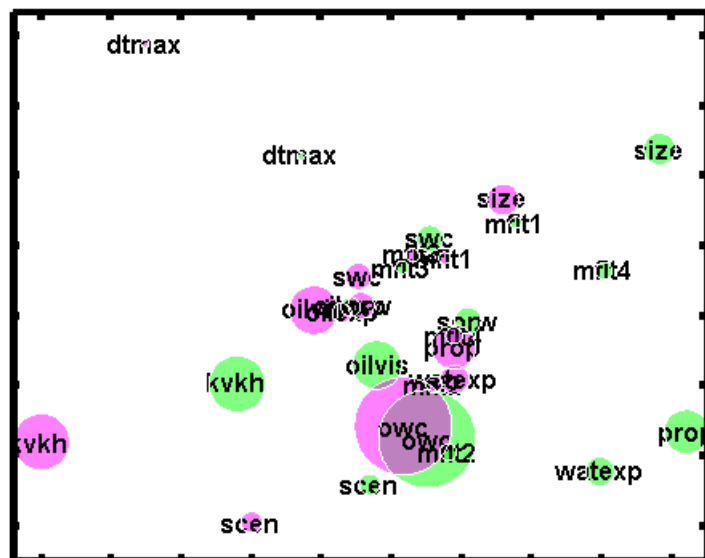
Fig. 6 h-plot, magenta: Conditioning parameter, green: Conditioned parameter
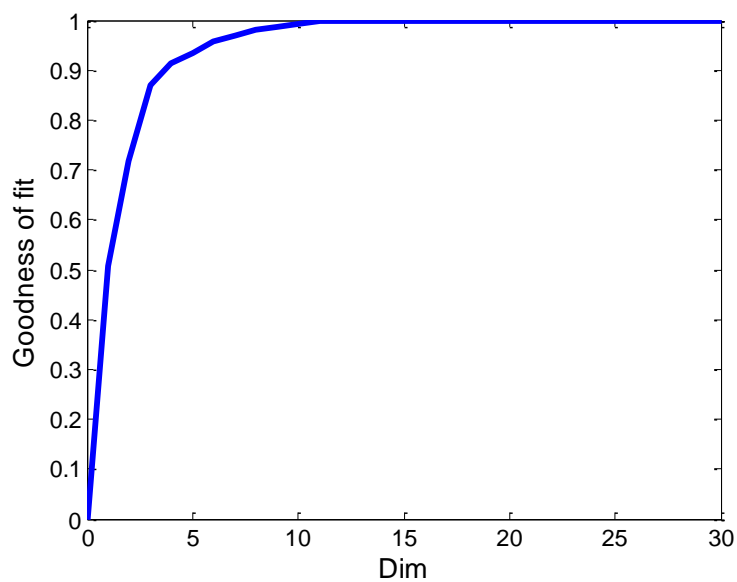

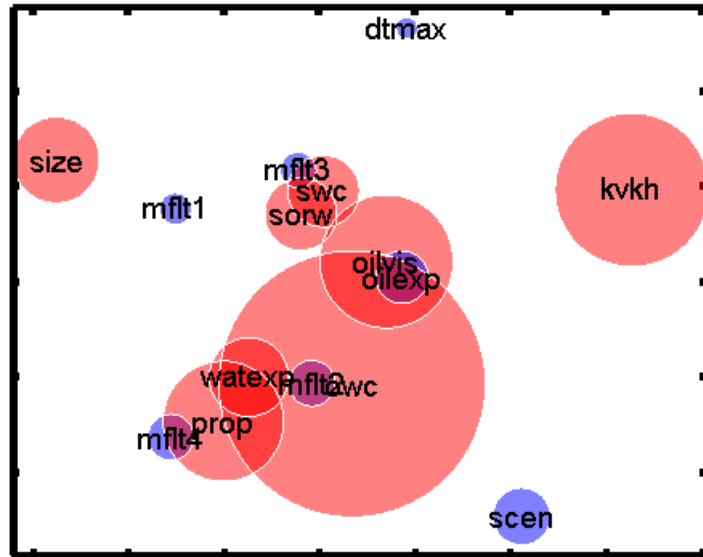*Fig. 7 Variance explained over # of eigenvectors*

Fig. 8 Bubble plot to show symmetric conditional effects

Class conditional CDFs can be displayed by *cdf_ConditionalEffects.m*. The following example shows conditional effect of *owc* given *kvkh*, s(*owc*|*kvkh*) from the first cluster, see Fig. 9.

```
% 5.4 Display class condtional CDFs
cdf_ConditionalEffects('owc','kvkh',DGSA,1)
```
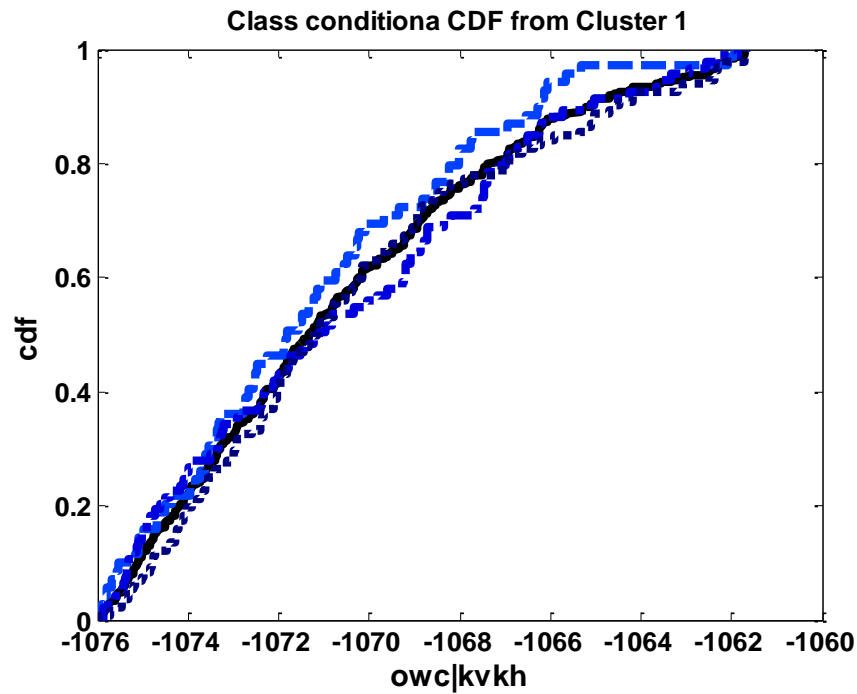
Class conditiona CDF from Cluster 1

Fig. 9 CDF of owc conditioned to kvkh from 1ˢᵗ cluster

4.5 Visualize responses by cluster

Results of clustering can be verified and visualized by MDS (Multidimensional scaling) plot (*DisplayMDSplot.m*). The responses from each well can be colored according to the clusters they belong to by *dynamicbycluster.m*, see Fig. 10 and 11.

```
%% 6. Check the response

ClusterColor=DisplayMDSplot(DGSA.D, DGSA.Clustering);

% Specify texts for plots
PlotLabels.xlabtxt='Time (day)'; PlotLabels.ylabtxt='WWPT (stb)';
PlotLabels.xlimits=[0, 7300]; PlotLabels.ylimits=[0 7E7];

% Visualize dynamic response by clusters
dynamicbycluster(DGSA.Clustering,PriorResponses, ClusterColor,PlotLabels);
```
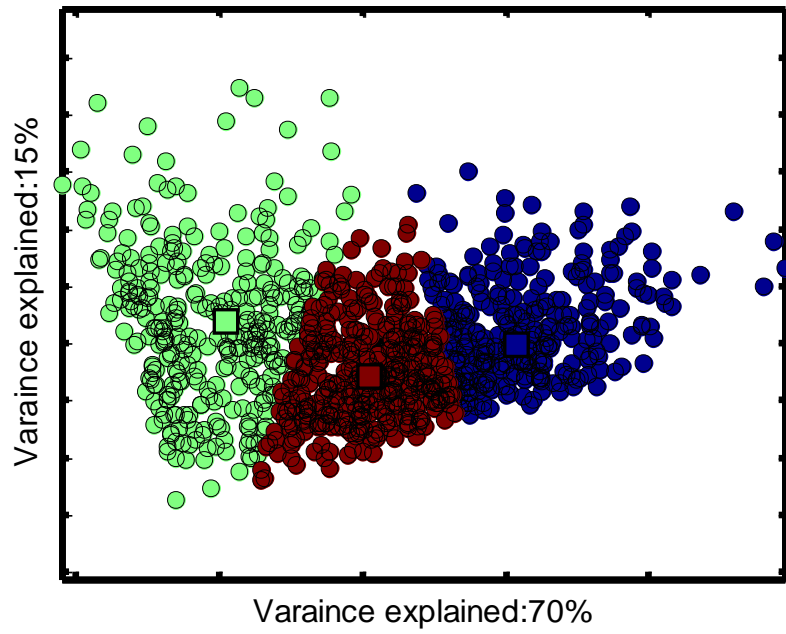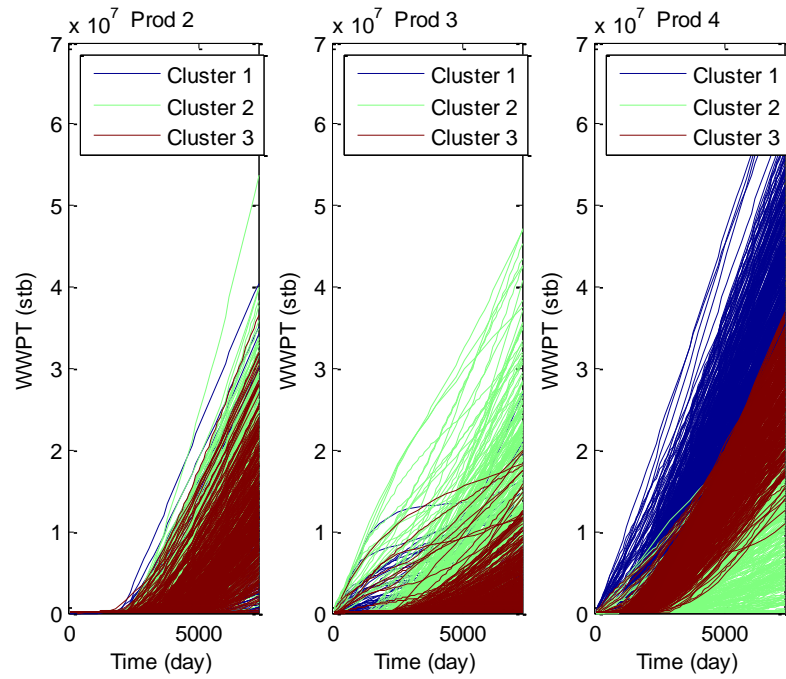
Fig. 10 MDS plot colored by clusters

Fig. 11 Responses from each well colored by clusters

5. Reduction of parameter uncertainty

The purpose of the script *main_DGSA_ParameterUncertaintyReduction.m* is to reduce the uncertainty of parameters using net conditional effects from DGSA. The script uses the results of the preceding script, *main_DGSA_Reservoir_Sensitivity.m*. The results are saved as *DGSA_Completed.mat* (or *DGSA_Completed_Spatial.dat*) in the subdirectory *VariablesSaved*.

5.1 Load inputs

In addition to the previous script, the flow response from reduced parameters are loaded as:

```
clear all; close all; fclose('all');
addpath(genpath(pwd))
load('DGSA_Completed'); % Load all variables from the previous script.
%%
load('Responses_PUR');
Responses_AfterPUR=WWPT_PUR; clear WWPT_PUR;
```

Responses_AfterPUR is a cell variable of which dimension is (# of wells x # of models). Each cell has a matrix containing responses.

5.2 Net conditional effects

Two functions are offered to compute and display net conditional effects. *ComputeNetConditionalEffects.m* shows the net conditional effects by bins and parameters. The parameter can be specified by the variable *Parameter2Reduce*. For example, in Fig. 12, the plot shows the net conditional effect of *oilexp* (Corey exponent of oil) and each part of bars shows the conditional effect with other parameters at each bin. This can be carried out for every parameter by *DisplayNetConditionalEffects_all.m*, see Fig. 13

```
%% Net Conditional effects
Parameter2Reduce='oilexp';

% Net conditional effects for a single parameter
[NetConditionalEffects,Bin_MaxNetConditional]=ComputeNetConditionalEffects...
.
    (Parameter2Reduce, DGSA);

% Net conditional effects for every parameter
DisplayNetConditionalEffects_all(DGSA);
```
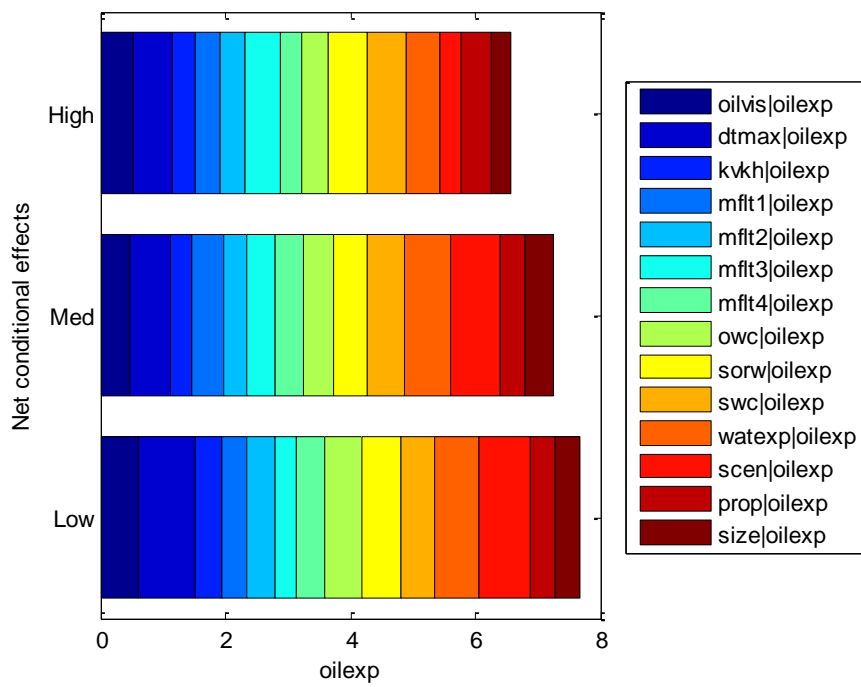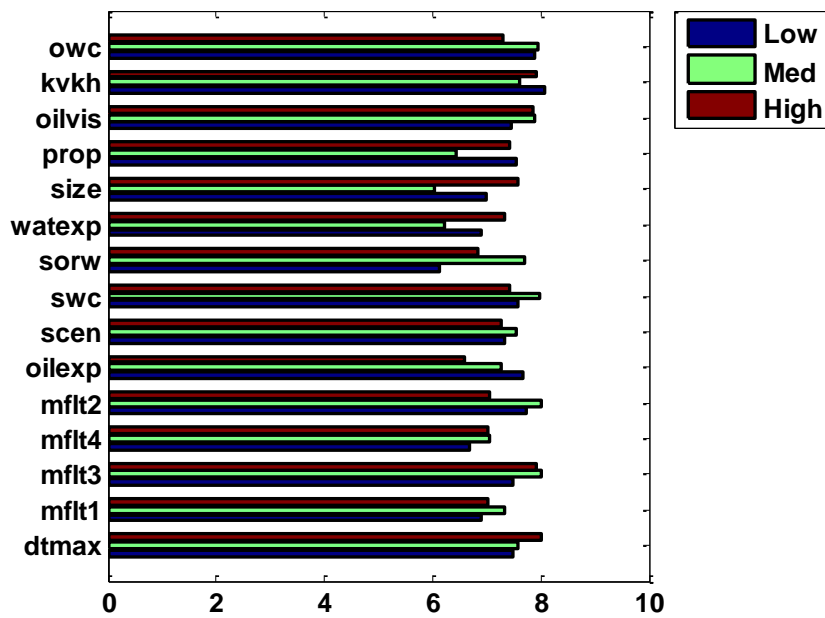
Fig. 12 Net conditional effects of oilexp

Fig. 13 Net conditional effect of every parameter

15

Responses from prior and reduced parameters are compared, see Fig. 14. Two kinds of responses have similar uncertainty. For plotting, time grid (*ResponsesCompared.tgrid*), index of response (*ResponsesCompared.idxResponse*, it is set to 2 in the example because the target responses are at the second column) and quantiles (*ResponsesCompared.QuantileLevel*) should be given as shown in the following code.

```
ResponsesCompared.tgrid=365:365:7300; ResponsesCompared.idxResponse=2;
ResponsesCompared.QuantileLevel=[.1,.5,.9];
ResponsesCompared.xlab='Time(day)';ResponsesCompared.ylab='WWPT (stb)';
CompareResponses(PriorResponses,Responses_AfterPUR,ResponsesCompared);
```
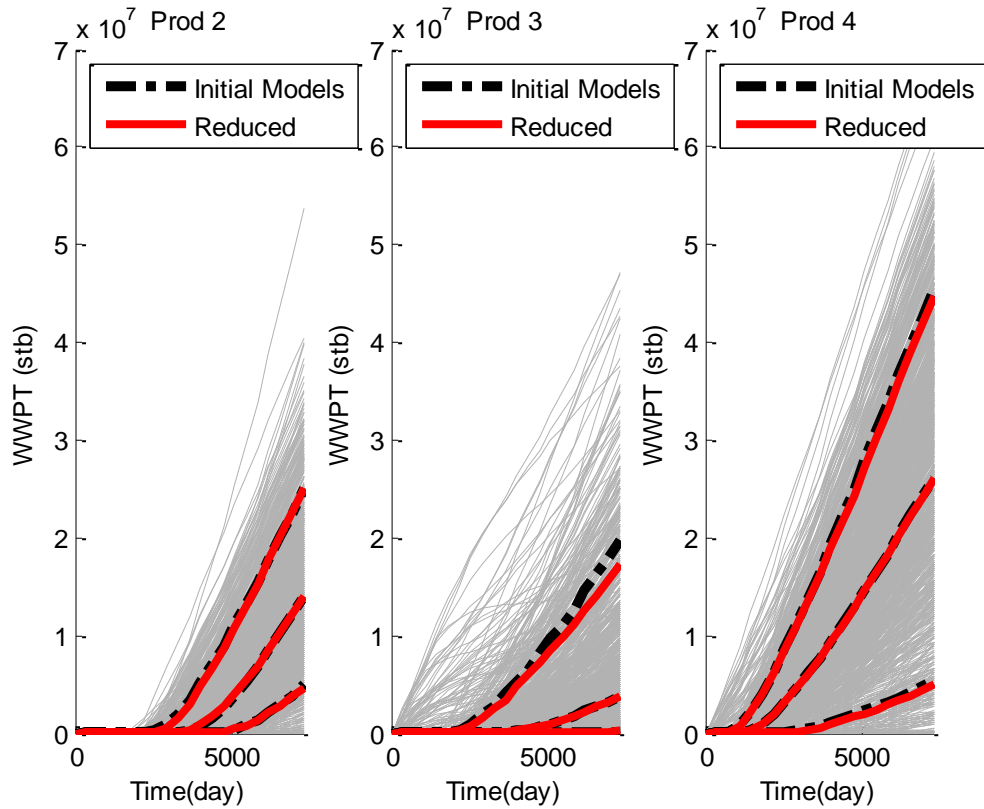

Fig. 14 Comparison of response uncertainty, (black): Samples from prior distributions, (red) Samples from model parameters of which uncertainty is reduced by DGSA

6. Comparison between DGSA/Sobols' indicies

main_compareDGSA_Sobol.m is to validate the results of DGSA by comparing sensitivities to Sobols' indices: first order effect and total effect.

6.1 Load inputs & specify additional variables

Sensitivities from Sobol's method requires the response to be univariate, so the target response is changed to FWPT (cumulative production of water). The responses and list of values of parameters need to be loaded as:

```
%%
timegrid=0:365:7300; N_Models=1000;nbcluster=3;
%% Load responses from reservoir simulator

load('FWPT.mat'); Sobol_responses=FWPT; clear FWPT;
DGSA_responses=Sobol_responses(1:N_Models,:);
DGSA=ProcessInputParameters('SobolParameters.dat');
```

The values of parameters to compute Sobol's indicies are offered at *SobolParameters.dat*. *Sobol_responses* is a matrix of responses to compute Sobol's indicies (n(k+2) x # of timesteps, n=sample size, k=# of parameters).

Also, DGSA is performed at every time step specified by variable *timegrid*, see Fig. 15.

```
%% Run DGSA over time
DGSA.Nbcluster=nbcluster;
MainEffects_over_time=DGSA_over_time(DGSA,DGSA_responses,timegrid);
```

Sobol's first order and total effects are computed *by SobolSensitivity.m*. Fig. 16 and 17 show first order and total effects at each time, respectively.
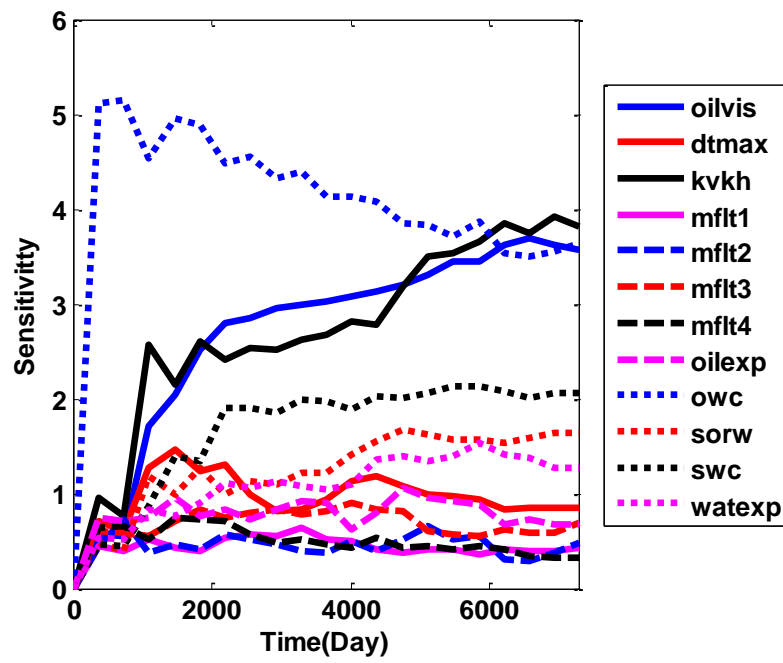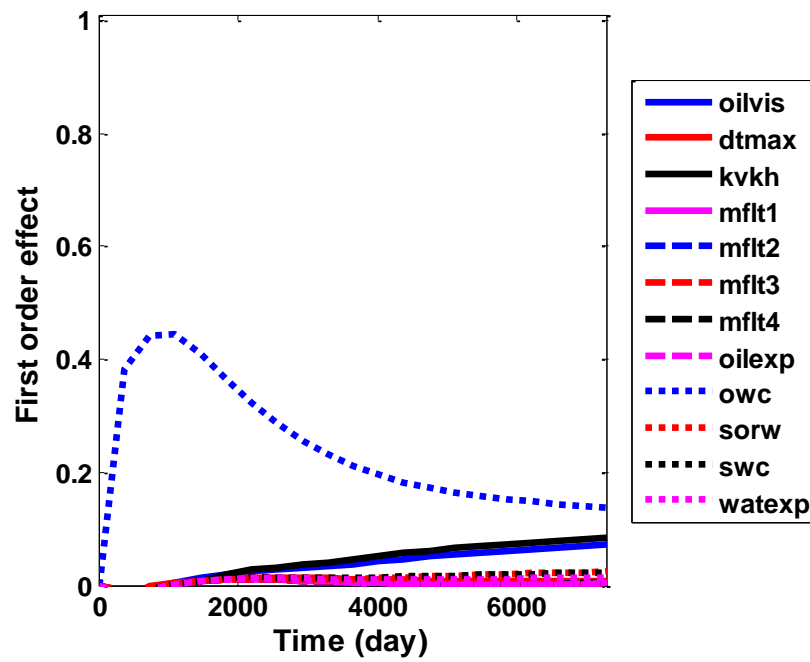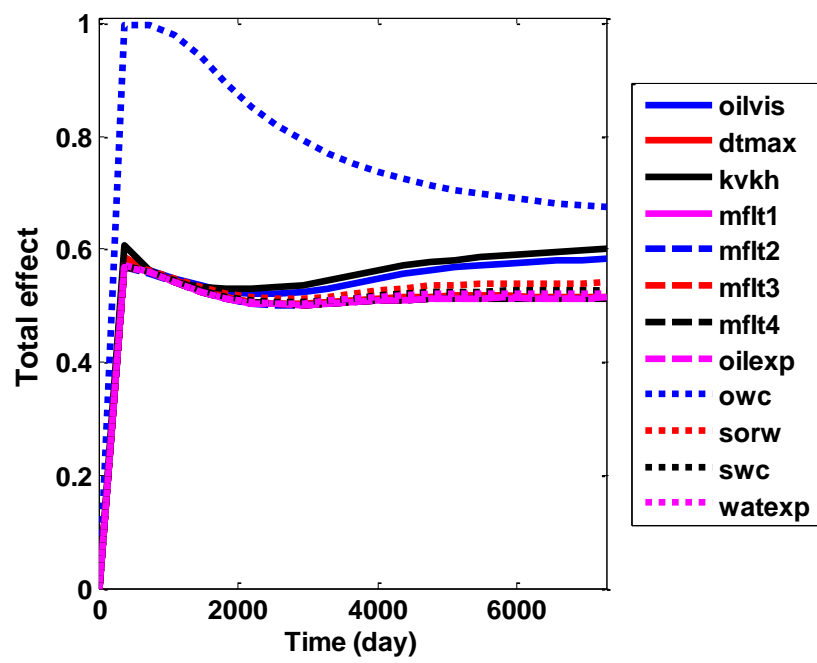
Fig. 15 DGSA over time

Fig. 16 First order effect over time

Fig. 17 Total effect over time