```
function [output] = morris_screening(settings)
% function [output] = morris_screening(settings)
%
% Description: Run morris screening to determine the most sensitive
 parameters
% in a model defined by the user.
%
% Author(s) Paul R. Miles | August 8, 2018
%           Graham T. Pash
%
% Input: Structure
% - model: Anonymous model function
% - parameters: Cell array with names, nominal values
% - sample_points: Integer, number of sample points
%
% Output: Structure
% - mu_i_star
% - sigma_i
%
% References:
% [1] Smith, R. C. (2013). Uncertainty quantification: theory,
 implementation,
%     and applications (Vol. 12). SIAM.
% [2] Morris, M. D. (1991). Factorial sampling plans for preliminary
%     computational experiments. Technometrics, 33(2), 161-174.

% TODO: create default inputs
% unpack input settings
model = settings.model;
parameters = settings.parameters;
sample_points = settings.sample_points;
% sample_type = settings.sample_type;
delta = settings.delta;
use_method = settings.use_method;

if isfield(settings,'observation_error')
    observation_error = settings.observation_error;
else
    observation_error = false;
end

parameters = setup_parameters(parameters);

% determine number of parameters
p = length(parameters);
I = eye(p); % [p x p] identity matrix
d = zeros(p, 1);
G = zeros(sample_points, p);
mod_threshold = round(sample_points/10);
for jj = 1:sample_points
    if mod(jj,mod_threshold) == 0
        fprintf('Status: %i of %i\n', jj, sample_points);
```

```
    end
    % Construct D
    D = datasample([-1,1], p) * delta;
    Dphys = map_sampling_to_physical(D, parameters);
    % Generate set
    x0 = zeros(p,1);
    for pp = 1:p
        if strcmpi(parameters(pp).dist, 'uniform')
            x0(pp) = rand(1, 1);
%             fprintf('pp = %i, uniform\n', pp)
        else
            x0(pp) = randn(1, 1);
%             fprintf('pp = %i, normal\n', pp)
        end
    end
    % map x to physical coordinates
    theta0 = map_sampling_to_physical(x0, parameters);
    theta0 = theta0(:); % Enforce column vector
    % evaluate model at physical coordinates
    y0 = model(theta0);
    % calculate elementary effect
    for ii = 1:p
        if strcmpi(use_method, 'complex')
            y = model(theta0+1i*delta*I(:,ii));
            d(ii) = imag(y)/delta;
        else
            theta = map_sampling_to_physical(x0 + D(ii)*I(:,ii),
 parameters);
            theta = theta(:);
            y = model(theta);
            d(ii) = (y - y0)/D(ii); % sampling vs. physical mismatch?
        end
        if observation_error ~= false
            if strcmpi(parameters(ii).dist, 'normal')
                d(ii) = d(ii)*sqrt(parameters(ii).distpar(2)/
observation_error);
            end
        end
    end
    G(jj,:) = d(:)';
end

% Construct the Morris indices.
mu = (1/sample_points)*sum(G, 1);
mu_i_star = (1/sample_points)*sum(abs(G), 1);
sigma_i = (1/(sample_points-1))*sum(bsxfun(@minus, G, mu).^2, 1);

% append features to output
output.mu_i_star = mu_i_star;
output.sigma_i = sigma_i;
output.d = d;
output.G = G;
output.parameters = parameters;
```

Published with MATLAB® R2016a