Morris Method Guide of Usage

Required Files

Our Morris Method model requires **total 4 files** to run.

1. loop_function.m
2. morrisMethod.m
3. ODEs.m
4. SAfinal.m

All the files can be found in ‘MorrisMethod_Final.zip’ file.

Files to be Modified

2 out of 4 files are the subject to changes.

1. morrisMethod.m
2. ODEs.m

[morrisMethod.m]

Editor - /Users/jacobkwak/Documents/MATLAB/iGEM/Morris Method/morrisMethod.m

morrisMethod.m

```
function self_use_ans = morrisMethod()
%% Program to run
%236154 615324 365142
%312456978
%924178
%10
%10
%7,8
% This algorithm is an adaptation of the method of Sensitivity Analysis
% called the Morris method.
%
% Sensitivity analysis is used to estimate the influence of uncertainty
% factors on the output of a function.
% The Morris method is sometimes referenced to as a qualitative method : it
% gives rough estimations with a limited number of calculations.
% The Morris method can be used to simplify a function, as a first step. It
% can identify the factors with a low influence which can be fixed.
% For further information :
% Saltelli, A., Tarantola, S., Campolongo, F., and Ratto, M. (2004).
% Sensitivity Analysis in Practice - A Guide to Assessing Scientific Models. Wiley.
%
% This algorithm reduces the risk to underestimate and fix non-negligible factors.
% It is presented in:
% Henri Sohier, Helene Piet-Lahanier, Jean-Loup Farges, Analysis and optimization of an air-launch-to-orbit
% (http://www.sciencedirect.com/science/article/pii/S0094576514004974)
%
% This program is divided in 6 parts:
% 1) Clearing the memory
% 2) Parameters : Please fill in
% 3) Initialization of the variables
% 4) Loop
% 5) Output text
% 6) Figure
%
% Please fill in the second part to apply the algorithm to your function.
% Do not change the parameters to see the results with the "modified Sobol
% test function".
%
% This program outputs a figure as well as a short summary in the console.
% Consider fixing the factors which appear as negligible on the left of the
% figure (not necessary all the factors under the limit).
%% 1) Clearing the memory
close all; % Closes the figures
clear all; % Clears the memory
clc; % Clears the command window

%% 2) Parameters : Please fill in
% Number of factors of uncertainty of the function studied :
nfac=10;

% Maximum number of simulation runs :
% Large number = better estimation of the influence of the factors
% Recommended value : (number of factors + 1) * 10
% The algorithm will maybe exceed this value if it is considered necessary
nsim_max = 1000;

% Function studied :
% Replace test_function by the name of your function. It must be a
% function with one multidimensional input x. x must represent the values
% of the uncertainty factors in the quantiles hyperspace (the i-th
% coordinate of x is not the actual value of the i-th factor, but the
% corresponding value of the cumulative distribution function of the i-th
% factor). To adapt your function, first calculate the actual values of
% the factors by applying the inverse of their cumulative distribution
% function to each coordinate of x; Matlab includes such inverses:
% mathworks.com/help/stats/icdf.html ).
studied_function = @(x)ODEs(x);

%% 3) Initialization of the variables
table_outputs = []; % All the outputs of the simulations runs. One line = results around one point of the f
table_ee = []; % All the elementary effects. One line = elementary effects at one point of the factors hype
factors_over = []; % Indexes of the factors over the limit (important factors). The elementary effects of t
table_factors_over = []; % Factors over the limit at the different steps. n-th line = factors with no eleme
points=[]; % Sampled points of the factors hyperspace where the elementary effects are calculated.
n=1; % Current step.
nsim = nfac+1; % Number of simulation runs after the next step.
initialization = 0; % Boolean, the calculations will foccus on the factors under the limit when it will equ
```

Simulation Parameters

nfac: number of parameters to consider

nsim_max: number of maximum simulations

Set ODEs as Studied Function

[ODEs.m]

Editor – /Users/jacobkwak/Documents/MATLAB/iGEM/Morris Method/ODEs.m

ODEs.m

```
1   function answer = ODEs(param)
2   % reaction_12(30000,[0,0,0,0,0,0])
3   t = 30000;
4   % initial concentrations
5   a0=0; %Diamine
6   s0=0; %H2O2
7   d0=0; %OxyRn
8   f0=0; %OxyRa
9   g0=0; %BFP
10  h0=0; %RFP
11
12  tf = t.*2;
13
14  %rate constants, 1/sec 1/mM
15  %k1 = 10.^(-4);
16  k1 = param(1);
17  %k2 = 4.20*10^(-4)*60;
18  k2 = param(2);
19  %kd1 = (2.65)^2.603;
20  kd1 = param(3);
21  %kd2 = 11.73^4.245;
22  kd2 = param(4);
23  %ka = 2*10^(-7);
24  ka = param(5);
25  %kb = 2*10^(-7);
26  kb = param(6);
27
28  %r1 = (10^(-4))*60; % Basal expression of OxyRn
29  r1 = param(7);
30  %r2 = 10 * 10^(-6)*60;
31  r2 = param(8);
32  %d2 = (log(2)/54)/60;
33  d2 = param(9);
34  %d4 = ((log(2))/20)/60;
35  d4 = param(10);
36
37  n = 2.603;
38  m = 4.245;
39
40  %Cad = 0;
41  %Cad = 5*10.^(-6);
42  %Cad = 10*10.^(-6);
43  % Cad = 15*10.^(-6);
44
45  % initial concentration
46  N0 = [a0;s0;d0;f0;g0;h0];
47
48  %opt=odeset('Events',@ReachSS);
49  %option2 = odeset('NonNegative',1);
50  [tt,NN] = ode15s(@f,[0,tf],N0);
51
52  AA = NN(:,1);
53  SS = NN(:,2);
54  DD = NN(:,3);
55  FF = NN(:,4);
56  GG = NN(:,5);
57  HH = NN(:,6);
58  answer = AA(end);
59
60      function dYdt = f(t,Y)
61
62          Dia=Y(1);
63          H2O2=Y(2);
64          OxyRn=Y(3);
65          OxyRa=Y(4);
66          BFP=Y(5);
67          RFP=Y(6);
68
69          dDiadt = -k1.*H2O2;
70          dH2O2dt = k1.*Dia-k2.*OxyRn.*H2O2 +r2;
71          dOxyRndt = r1 - k2.*OxyRn.*H2O2;
72          dOxyRadt = k2.*OxyRn.*H2O2;
73          dBFPdt = ka*((OxyRa^n)/(kd1+OxyRa^n))-d2.*BFP;
74          dRFPdt = kb*((OxyRa^m)/(kd2+OxyRa^m))-d4.*RFP;
75
76          dYdt=[dDiadt;dH2O2dt;dOxyRndt;dOxyRadt;dBFPdt;dRFPdt];
77
78      end
79
```

Initial Concentration Variables

Result Variables

In-ODE Variables

Initial Concentrations Variables

Initial points of resultants are stored in variables.

Rate Constants

Rate constants are assigned from the input parameter of ODEs function. (Not hardcoded)

Refer to 'All Constants_Updated.docx'

Solving the ODEs

Pre-declared initial concentration variables are fed into the ODE solver.

Results from the ODE solver will be saved in the 'NN' array.

* Results for each resultant will be saved in separate arrays. In this case, (Diamine:AA), (H2O2:SS), and so on.

** Set the 'answer' variable as the resultant of interest. Sensitivity index of parameters with respect to the resultant of interest will be calibrated.

*** Please pay attention to the order of initial parameter variables, result variables, and in-ODE variables. They MUST be matching.