GEOINFORMATICS PROJECT

Implementation Document

Version 1.0

Project name: Drone Surveying Planning (DSP)

Prepared by: AbdelGafar Omar, Rossi Marta

Contents

Revision History

Name	Date	Reason for Changes	Version

List of Figures

Introduction

The purpose of the Algorithm document is to illustrate the different algorithms used in implementing DSP plugin using three algorithms which give the user the possibility to choose the proper one for his/her flight planning.

The three algorithms are:

- Normal case algorithm.
- Least square method using simulation Data.
- Least square method using DEM.

1. Normal case algorithm.

//Inputs:

Sensor focal length ‘c’ ← user input
Sensor width ‘fw’ ← user input
Sensor height ‘fh’ ← user input
No. of pixels of the sensor in x ‘npx’ ← user input
No. of pixels of the sensor in y ‘npy’ ← user input
Distance of surveying area in X ‘X’ ← extracted from Area shape file
Distance of surveying area in Y ‘Y’ ← extracted from Area shape file
Flight height ‘h’ ← user input
Along track overlapping ‘Rl’ ← user input
Cross track overlapping ‘Rt’ ← user input
Collimation error ‘scoll’ ← user input and its default value is 1 in Pix
Simulation points density ‘delta’ ← user input via combo box (Low, Medium, High)

//outputs:

Propagate s2zy and s2zx

Footprint width ‘W’ ← (fw*h)/c
Footprint height ‘H’ ← (fh*h)/c

Ground Sample Distance using W ‘GSDw’ ← W/npx
Ground Sample Distance using H ‘GSDh’ ← H/npx

Basline ‘b’ ← (1-Rl)*H
interaxie ‘interaxie’ ← (1-Rt)*W

No. of strips in y ‘nstrip_y’ ← ceil (Y/b)
No. of strips in x ‘nstrip_x’ ← ceil (X/i)

Real baseline ‘b_real’← Y/nstrip_y
Real interaxie ‘i_real’← X/nstrip_x

Real Along track overlapping ‘Rl_real’ ←1- b_real /H
Real cross track overlapping ‘Rt_real’ ←1- i_real /W

The x coordinate of the camera position in GCS ‘xo’← range from i_real/2 to X with step of i_real
The y coordinate of the camera position in GCS ‘yo’← range from b_real/2 to Y with step of b_real

mesh grid xo and yo

Assign the z coordinate of the camera h to Zo

Create a flight path matrix ‘flpath’ with dimension of [nstrip_x* nstrip_y , 2]

for i from 0 to the # of strips in x

fill the first column of flpath from first element to nstrip_y with xo(i)

if yo is odd:

fill the second column with the range of nstrip_y with flipping y0

else:

fill the second column with the range of to nstrip_y with y0

end for

Create simulation X coordinates of a grid 'xGrid'← range from 0+delta/2 to X-delta/2 with a step of delta

Create simulation Y coordinates of a grid 'yGrid'← range from 0+delta/2 to Y-delta/2 with a step of delta

Mesh grid xGrid and yGrids

Create a matrix of zeros with the same dimension of yGrid ← over_y

for i from 0 to the no of strips in y

if ((yGrid greater than or equal yo(i)-H/2 and yGrid less than or equal yo(i)+H/2 and))

increment over_y matrix with 1

end for

Create a matrix of zeros with the same dimension of xGrid ← over_x

for i from 0 to the no of strips in x

if ((xGrid greater than or equal yo(i)-H/2 and xGrid less than or equal yo(i)+H/2 and))

increment over_x matrix with 1

end for

Mesh grid over_x and over_y

Propagation of the collimation 's2px'← 2* (scoll*fw/npx)^2

Sigma ^2 of z in longitudinal direction 's2zy'← (h^2 / (c*0.001*i_real))^2 * s2px
Sigma ^2 of z in transversal direction 's2zx'← (h^2 / (c*0.001*b_real))^2 * s2px
Propagate s2zy and s2zx← sqrt((s2zy*(Over_y-1)+s2zx*(Over_x - 1)))/((Over_y-1)+(Over_x-1))^2

2. Least square using Data Simulation Algorithm

//Inputs:

Sensor focal length 'c' ← user input
Sensor width 'fw' ← user input
Sensor height 'fh' ← user input
No. of pixels of the sensor in x 'npx' ← user input
No. of pixels of the sensor in y 'npy' ← user input
Distance of surveying area in X 'X' ← extracted from Area shape file
Distance of surveying area in Y 'Y' ← extracted from Area shape file
Flight height 'h' ← user input
Along track overlapping 'Rl' ← user input
Cross track overlapping 'Rt' ← user input
Collimation error 'scoll' ← user input and its default value is 1 in Pix
Simulation points density 'delta' ← user input via combo box (Low, Medium, High)

//outputs:

Empirical error matrix in x direction
Empirical error matrix in y direction
Empirical error matrix in z direction
Theoretical error matrix in x direction
Theoretical error matrix in y direction
Theoretical error matrix in z direction

Footprint width 'W' ← (fw*h)/c
Footprint height 'H' ← (fh*h)/c

Ground Sample Distance using W 'GSDw' ← W/npx
Ground Sample Distance using H 'GSDh' ← H/npx

Basline 'b' ← (1-Rl)*H
interaxie 'interaxie' ← (1-Rt)*W

No. of strips in y 'nstrip_y' ← ceil (Y/b)
No. of strips in x 'nstrip_x' ← ceil (X/interaxie)

Real baseline 'b_real'← Y/nstrip_y
Real interaxie 'i_real'← X/nstrip_x

Real Along track overlapping 'Rl_real' ←1- b_real /H
Real cross track overlapping 'Rt_real' ←1- i_real /W

The x coordinate of the camera position in GCS 'xo'← range from i_real/2 to X with step of i_real
The y coordinate of the camera position in GCS 'yo'← range from b_real/2 to Y with step of b_real
mesh grid xo and yo
Assign the z coordinate of the camera h to Zo
Numbering grid point 'ptName'← matrix of the same dimensions as XGrid its value with the values from 0: the length of Xgrid

Create a flight path matrix ‘flpath’ with dimension of [nstrip_x* nstrip_y , 2]

for i from 0 to the no of strips in x
fill the first column of flpath from first element to nstrip_y with x0(i)
if yo is odd:
fill the second column with the range of nstrip_y with flipping y0
else:
fill the second column with the range of to nstrip_y with y0

end for

Create simulation X coordinates of a grid ‘xGrid’← range from 0+delta/2 to X-delta/2 with a step of delta
Create simulation Y coordinates of a grid ‘yGrid’← range from 0+delta/2 to Y-delta/2 with a step of delta
Mesh grid xGrid and yGrids

for i in range from 0 to nstrip_y*nstrip_x

xsiGrid← (XGrid – Xo(i))/h*c
etaGrid← (YGrid – yo(i))/h*c

if the absolute value of xsiGrid less than or equal fw/2 and etaGrid less than or equal fh/2
mask← 1
else
mask←0
Vector of no of observed point ‘pt_obs’ ← vector of [pt_obs; pt_name of where all mask values =1]
Vector of no of image containing the point ‘im_obs’ ← vector of [im_obs; i*mask of where all mask values =1]

observed xsi ‘xsi_obs’ ← vector of [xsi_obs; xsiGrid where all mask values =1]
observed eta ‘eta_obs’ ← vector of [eta_obs; etaGrid where all mask values =1]
end for

xsi_obs← xsi_obs+scoll *(fw/npx)*random matrix has the same size of xsi_obs
eta_obs← eta_obs +scoll *(fw/npx)*random matrix has the same size of eta_obs

Create sparse matrix A of size 2* the length of pt_obs ,3*no of grid points
The values in the couples (1: the length of pt_obs , pt_obs)← c
Update as well the couples (1: the length of pt_obs , pt_obs+2* no of grid points)←xsi_obs
Update as well the couples ((the length of pt_obs+1 to 2* the length of pt_obs , pt_obs+ no of grid points) ← c
Update as well the couples ((the length of pt_obs+1 to 2* the length of pt_obs , pt_obs+ 2*no of grid points) ← eta_obs

Observation vector ‘yo’ ← [xsi_obs *Zo+c*Xofor each im_obs; eta_obs *Zo+c*Yofor each im_obs]

Normal matrix ‘N’ ← transpose of A matrix * the A matrix

Perform the LU decomposition inverse for Normal matrix ‘N’

Normal known term nt ← transpose of A *yo

Extract the estimated values of X
Extract the estimated values of Y
Extract the estimated value of Z
Calculate the residuals vector.
Calculate the posterior variance.
Calculate Empirical standard division vector ‘s2’.
Calculate theoretical standard division vector ‘s3’.
Empirical error matrix in x direction
Empirical error matrix in y direction
Empirical error matrix in z direction
Theoretical error matrix in x direction
Theoretical error matrix in y direction
Theoretical error matrix in z direction

3. Least square using Data Simulation Algorithm

3.1 DEM Data Extraction

//Inputs:

rasin ← user input of Raster representing the DEM
fn← user input of Polygon Shapefile of the surveying area

//outputs:

Matrix of Elevations from DEM
Matrix of mesh gridding values of X and Y coordinates of GCP

Read the survey area shapefile as Geodata frame and save it in variable gdf
cr ← The coordinate reference system of the data frame.
Function newpath so you can generate new path for new layers
pass In : source path , name of the layer with its extension
path ← read the source path
path_len← the length of the source path without the last element
n_path← concatenate the source path without last element + the name of the new layer
pass Out: n_path
End newpath
Initialize x as an array.
Initialize y as an array.
i ← 0.
for index to the list of exterior coordinates of geometry in the Geodata frame do:
xy coordinate = list of pt points.
x← x coordinate from xy coordinate
y← y coordinate from xy coordinate
end for
max_x ← the max of x
max_y the min of x
min_y ←← the max of y
min_x ← the min of y
p1 ← [min_x, min_y]
p2 ← [max_x, min_y]
p3 ← [max_x, max_y]
p4 ← [min_x, max_y]
d_xy ← Calculate the Euclidean distances in x and y
X← the min distance in d_xy
Y← the max distance in d_xy
Create polygon from point list as [p1,p2,p3,p4,p1] save it in variable polygon
data ← feature ‘id’= 1 and ‘geometry’ column = polygon
Create Geodata frame contains the polygon with (data and (cr as string))
shpin_name← user input of the name of the new rectangular Polygon Shapefile
shpin ← use the new path function to generate the path of the new rectangle shapefile
Write poly_gdf into ESRI Shape file

rasout_name← user input of the name of the new clipped raster
rasout ← use the new path function to generate the path of the new clipped raster
rect_DEM← Clipping the rasin raster by shpin shapefile and pass it → rasout ---by using gdal Wrapper
ds← open rasout raster
band← read raster band of ds
Set the no data values in the band as nan
Elev ← read the band values as an array
Assign -32767 or-99999 in the Elev array as nan
Zo← the mean of Elev array + The flight hight
r_n← Get the number of rows from Elev array
r_n← Get the number of rows from Elev array
x_0← Get the x coordinate of the raster origin
Dem_res_x← Get the pixel size in x (the resolution in x)
y_0← Get the y coordinate of the raster origin
Dem_res_y← Get the pixel size in y (the resolution in y)
X_f ← x_0 + Dem_res_x × r_n
Y_f ← y_0 + Dem_res_y × c_n
xGrid← Create an array of values in x axis from x_0 to X_f by steps of Dem_res_x
yGrid← Create an array of values in y axis from x_0 to Y_f by steps of Dem_res_y
if the absolute value of Dem_res_x and the absolute value of Dem_res_y <15
Take the values of xGrid with index step of 2 and assign it to Xgrid
Take the values of yGrid with index step of 2 and assign it to YGrid
Take the values of Elev with index step of 2
else:

Xgrid ← xGrid
Ygrid ← yGrid
Elev ← Elev

Mesh griding Xgrid and Ygrid

3.2 Least square using DEM Algorithm.

//Inputs:

Sensor focal length 'c' ← user input
Sensor width 'fw' ← user input
Sensor height 'fh' ← user input
No. of pixels of the sensor in x 'npx' ← user input
No. of pixels of the sensor in y 'npy' ← user input
Distance of surveying area in X 'X' ← extracted from Area shape file
Distance of surveying area in Y 'Y' ← extracted from Area shape file
Flight height 'h' ← user input
Along track overlapping 'Rl' ← user input
Cross track overlapping 'Rt' ← user input
Collimation error 'scoll' ← user input and its default value is 1 in Pix
Simulation points density 'delta' ← user input and its default value is 1 in m

//outputs:

Empirical error matrix in x direction
Empirical error matrix in y direction
Empirical error matrix in z direction
Theoretical error matrix in x direction
Theoretical error matrix in y direction
Theoretical error matrix in z direction

Footprint width 'W' ← (fw*h)/c
Footprint height 'H' ← (fh*h)/c

Ground Sample Distance using W 'GSDw' ← W/npx
Ground Sample Distance using H 'GSDh' ← H/npx

Basline 'b' ← (1-Rl)*H
interaxie 'i' ← (1-Rt)*W

No. of strips in y 'nstrip_y' ← ceil (Y/b)
No. of strips in x 'nstrip_x' ← ceil (X/i)

Real baseline 'b_real'← Y/nstrip_y
Real interaxie 'i_real'← X/nstrip_x

Real Along track overlapping 'Rl_real' ←1- b_real /H
Real cross track overlapping 'Rt_real' ←1- i_real /W

The x coordinate of the camera position in GCS 'xo'← range from min of x + i_real/2 to max of x with step of i_real

The y coordinate of the camera position in GCS 'yo'← range from min of y + b_real/2 to max of y with step of b_real

mesh grid xo and yo

Numbering grid point 'ptName'← matrix of the same dimensions as XGrid its value with the values from 0: the length of Xgrid

Create a flight path double array matrix 'flpath' with dimension of [nstrip_x* nstrip_y , 2]
for i from 0 to the no of strips in x
fill the first column of flpath from first element to nstrip_y with x0(i)
if yo is odd:
fill the second column with the range of nstrip_y with flipping y0
else:
fill the second column with the range of to nstrip_y with y0

end for

Create simulation X coordinates of a grid 'xGrid'← range from 0+delta/2 to X-delta/2 with a step of delta
Create simulation Y coordinates of a grid 'yGrid'← range from 0+delta/2 to Y-delta/2 with a step of delta
Mesh grid xGrid and yGrids
Create a matrix of zeros with the same dimension of yGrid ← over_y

for i in range from 0 to nstrip_y*nstrip_x

xsiGrid← (XGrid – Xo(i))/h*c
etaGrid← (YGrid – yo(i))/h*c

if the absolute value of xsiGrid less than or equal fw/2 and etaGrid less than or equal fh/2
mask← 1
else
mask←0
Vector of no of observed point 'pt_obs' ← vector of [pt_obs; pt_name of where all mask values =1]
Vector of no of image containing the point 'im_obs' ← vector of [im_obs; i*mask of where all mask values =1]
observed xsi 'xsi_obs' ← vector of [xsi_obs; xsiGrid where all mask values =1]
observed eta 'eta_obs' ← vector of [eta_obs; etaGrid where all mask values =1]

end for

xsi_obs← xsi_obs+scoll *(fw/npx)*random matrix has the same size of xsi_obs
eta_obs← eta_obs +scoll *(fw/npx)*random matrix has the same size of eta_obs

Create sparse matrix A of size 2* the length of pt_obs ,3*no of grid points

The values in the couples (0: the length of pt_obs , pt_obs)← c

Update as well the couples (0: the length of pt_obs , pt_obs+2* no of grid points)←xsi_obs
Update as well the couples ((the length of pt_obs+1 to 2* the length of pt_obs , pt_obs+ no of grid points) ← c
Update as well the couples ((the length of pt_obs+1 to 2* the length of pt_obs , pt_obs+ 2*no of grid points) ← eta_obs
Observation vector 'yo' ← [xsi_obs *Zo+c*Xofor each im_obs; eta_obs *Zo+c*Yofor each im_obs]
Normal matrix 'N' ← transpose of A matrix * the A matrix
Perform the LU decomposition inverse for Normal matrix 'N'
Normal term tn ← transpose of A *yo
Extract the estimated values of X.
Extract the estimated values of Y.
Extract the estimated value of Z.
Calculate the residuals vector.
Calculate the posterior variance.

4. Appendix

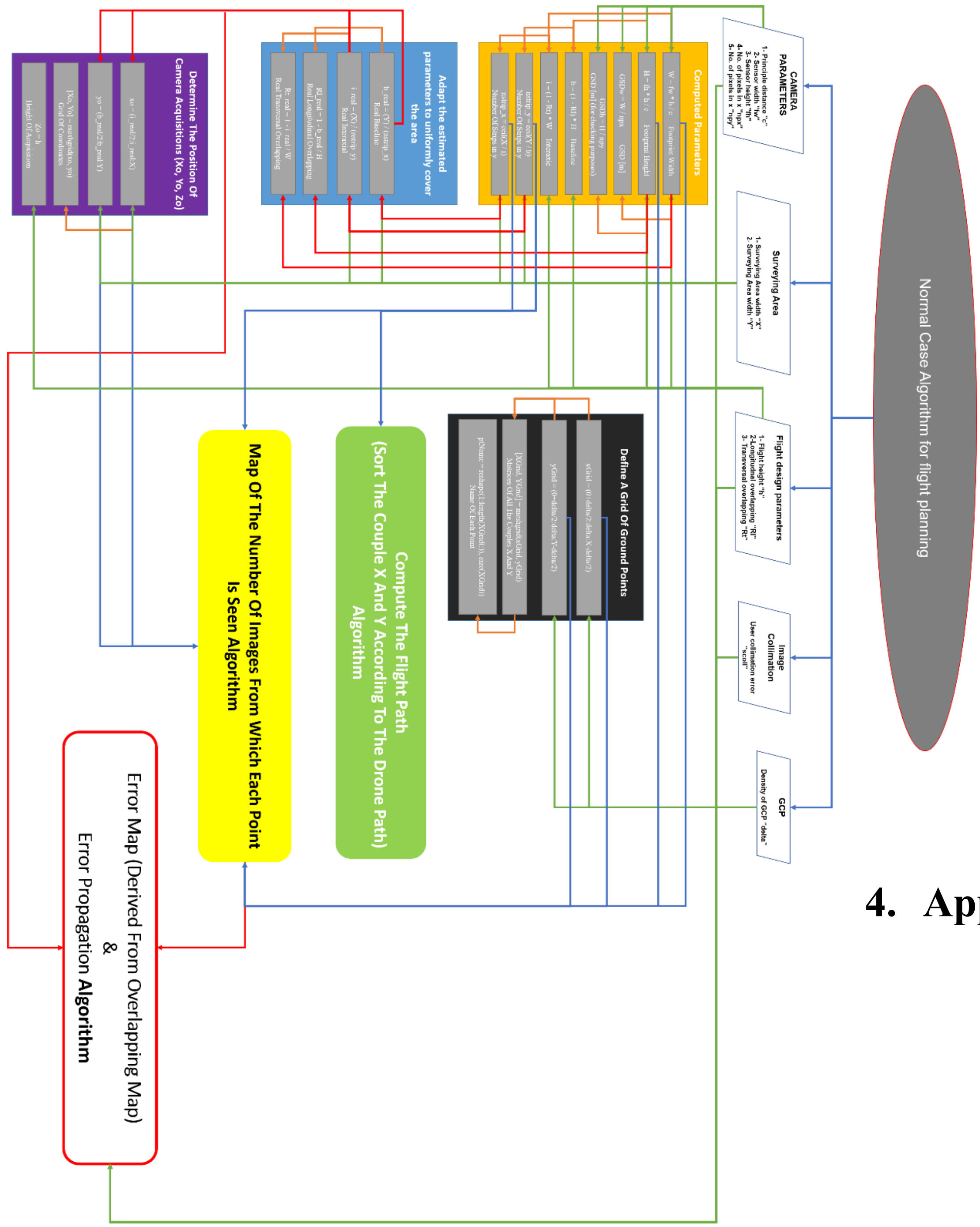

Figure 1Normal Case Algorithm for flight planning

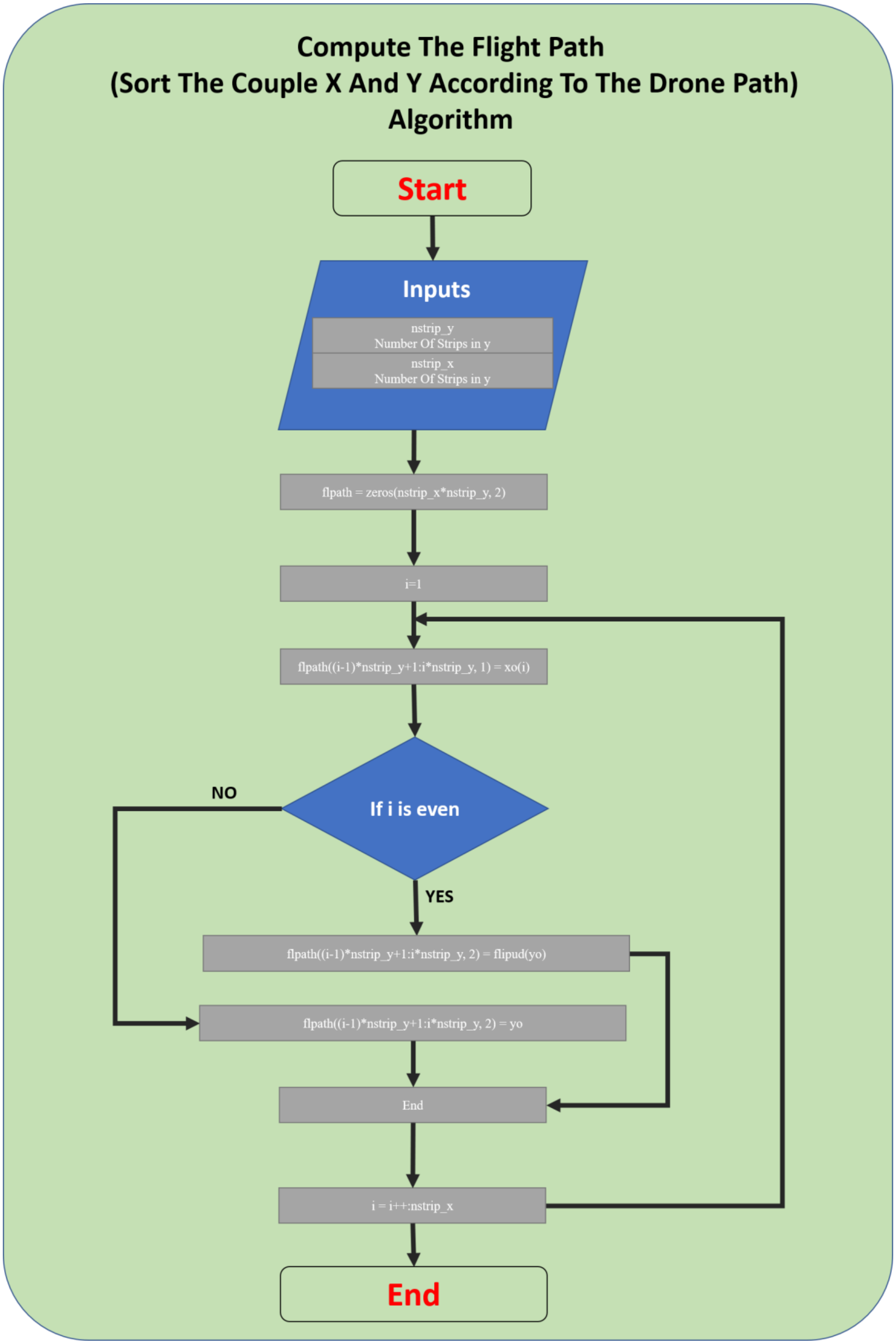

Figure 2Flight Path computation

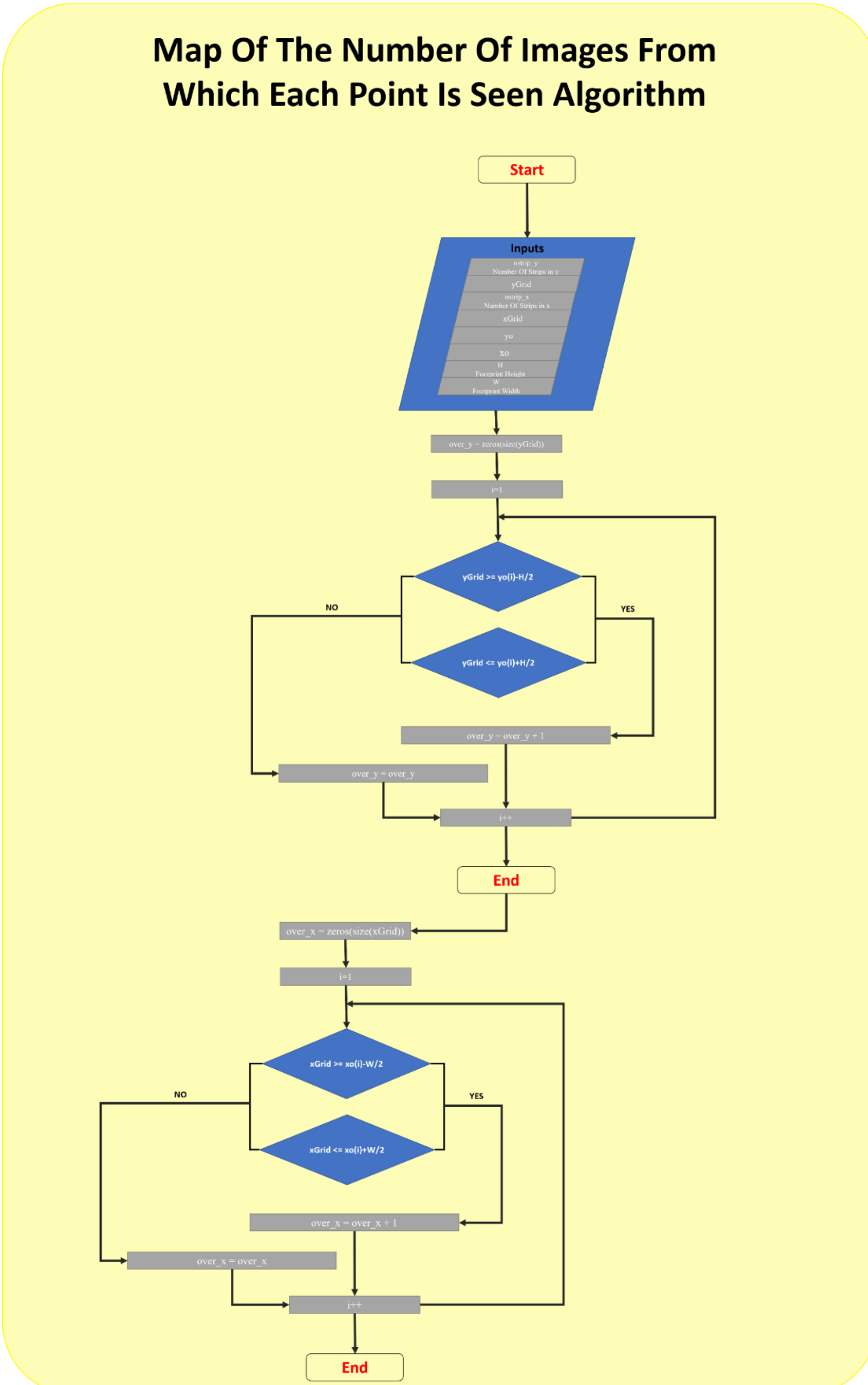

Figure 3Map of number of images containing each Grid point.

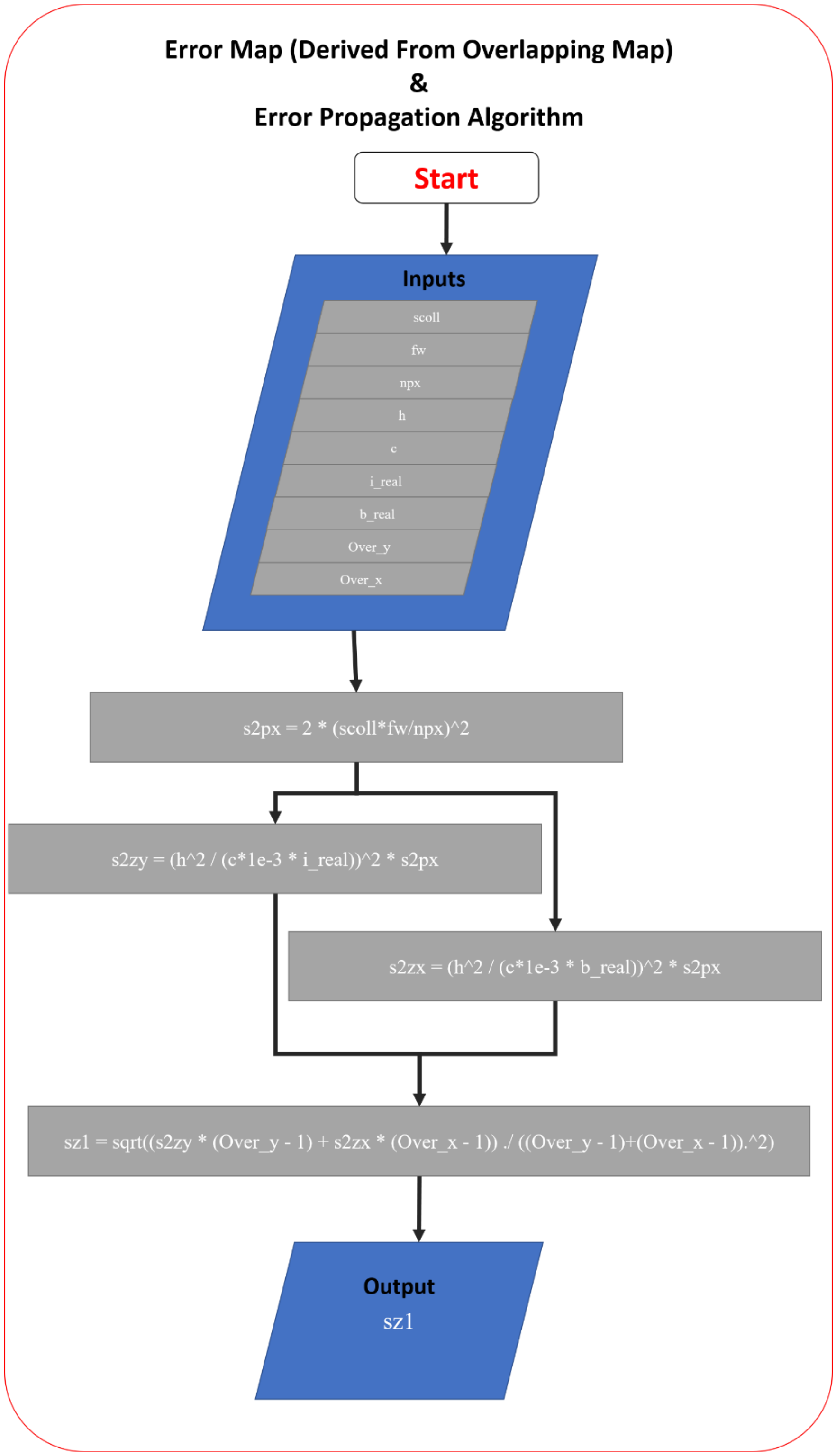

Figure 4Error map calculation for the Normal Case Algorithm.

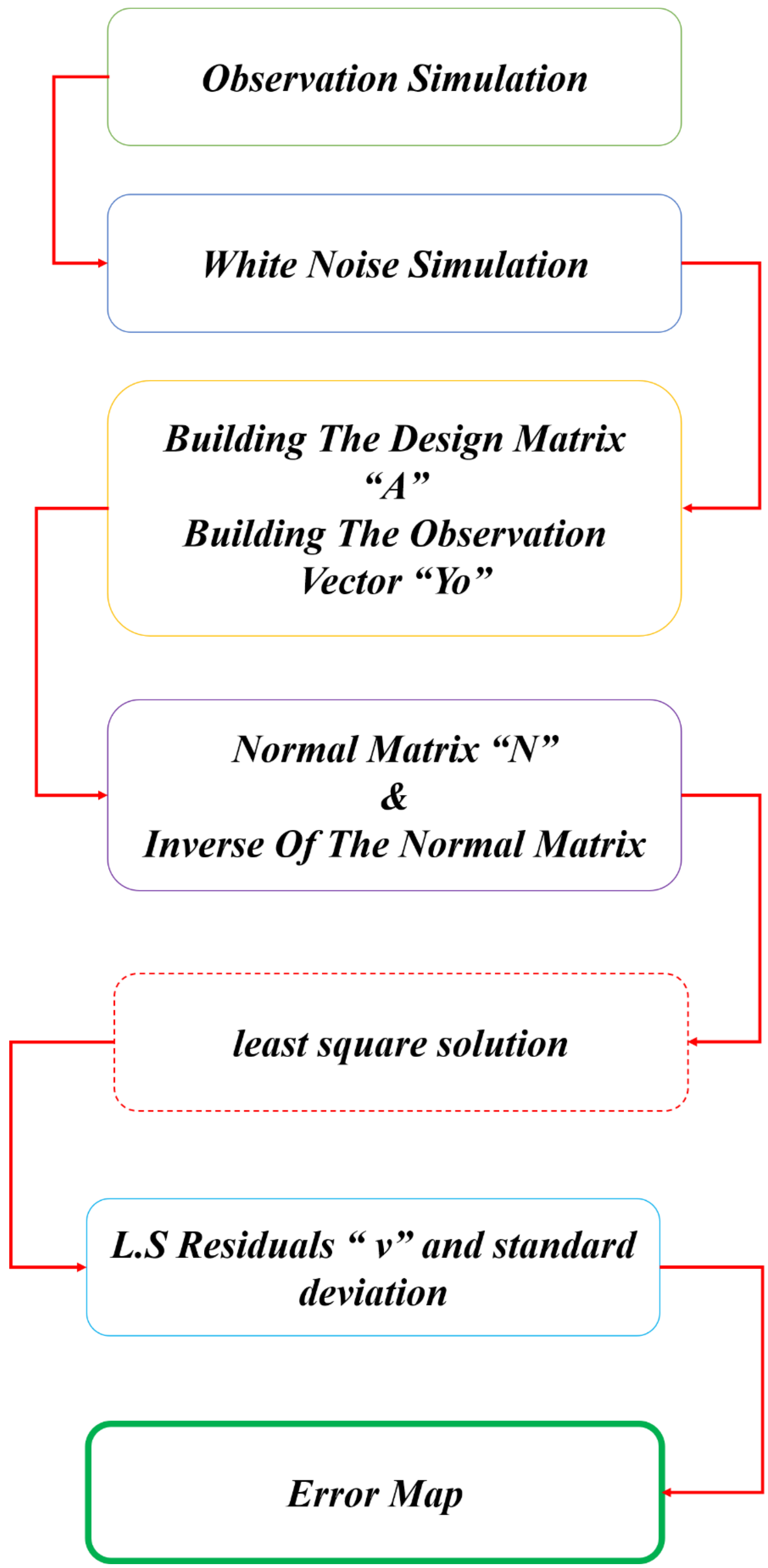

Figure 5 Flowchart for Least square using simulation data Algorithm.

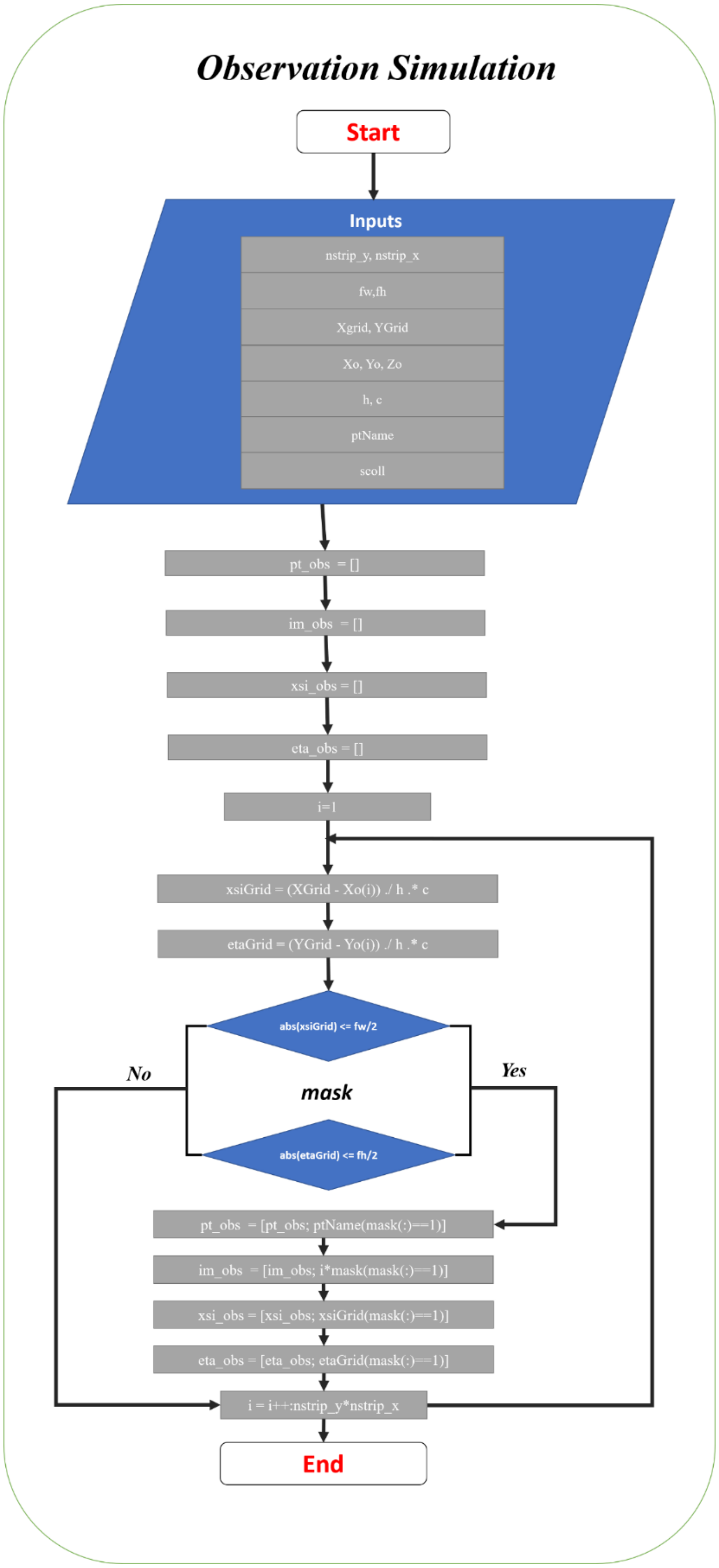

Figure 6 Simulation of the observation flowchart.

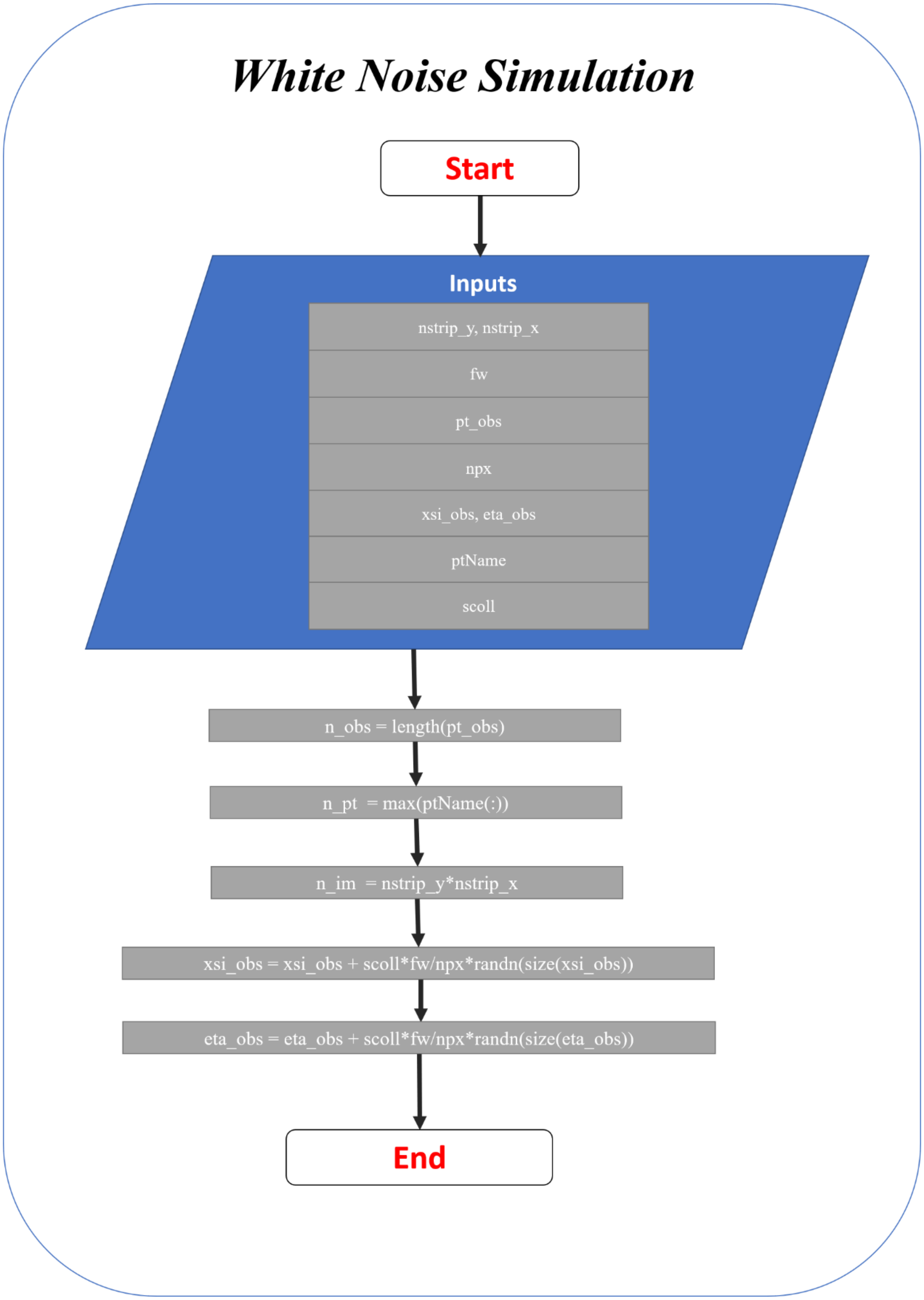

Figure 7 Adding white noise for simulation points.

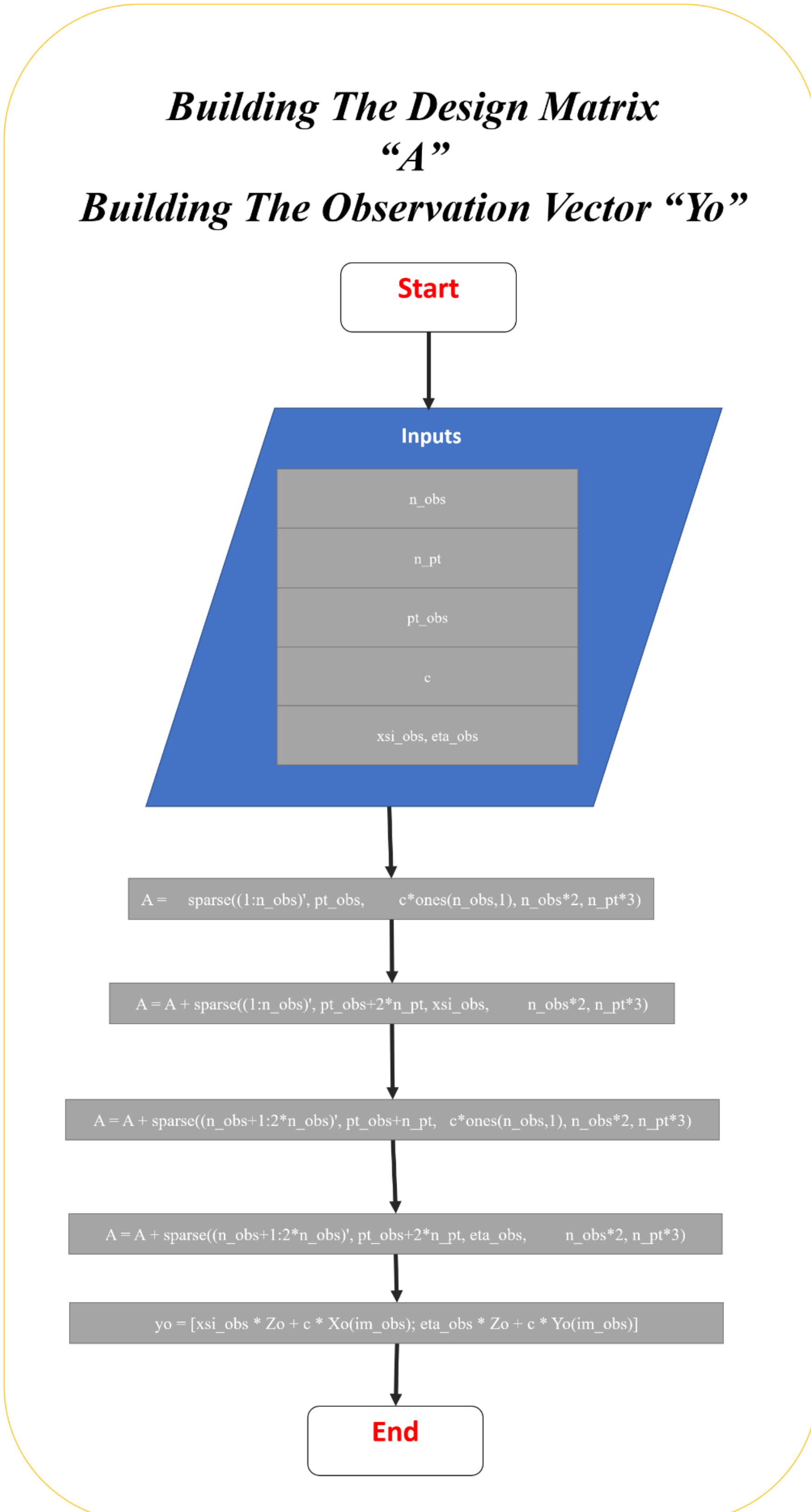

Figure 8 A matrix and Observation vector Y0 creation

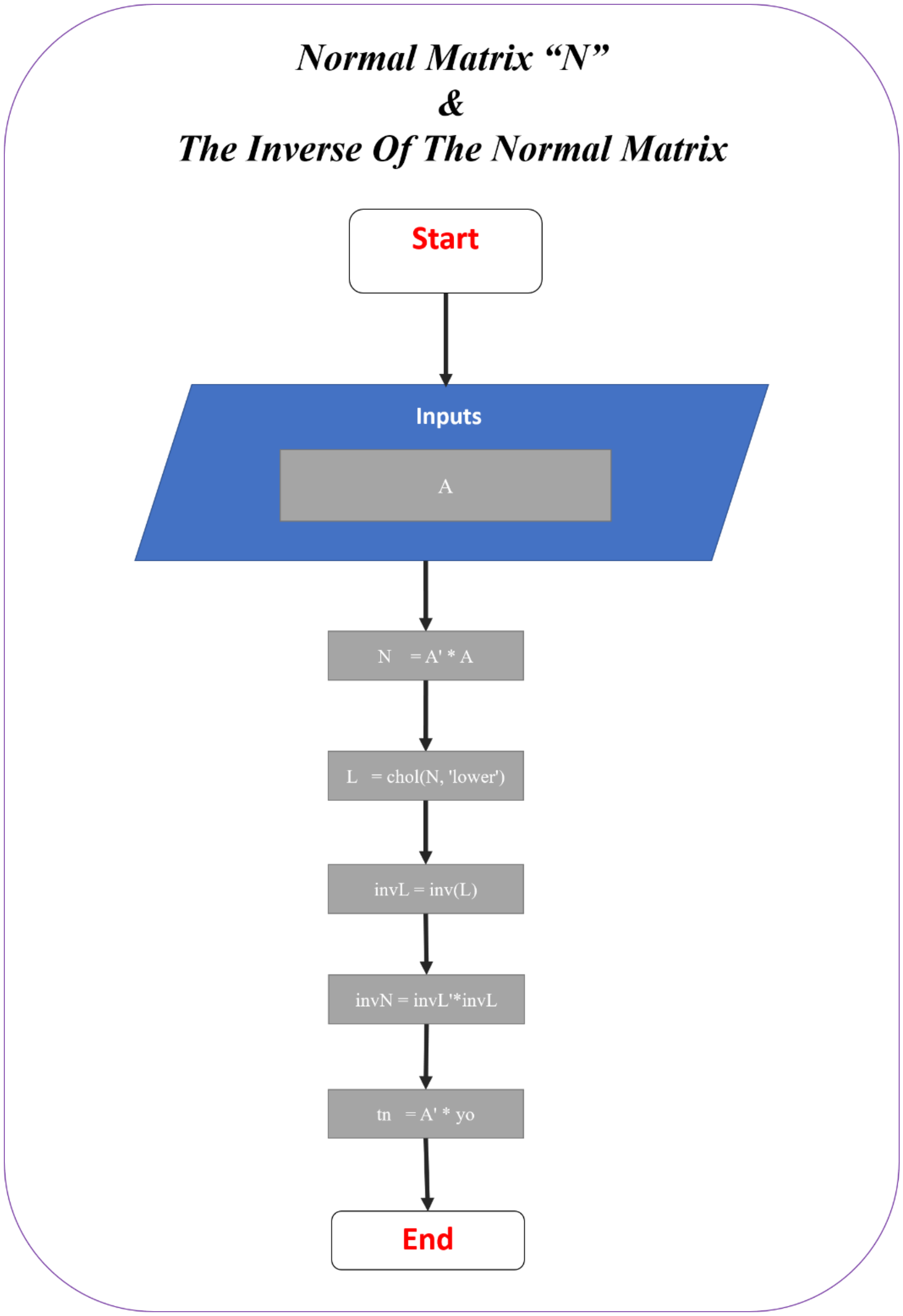

Figure 9 Normal matrix calculation and inversion.

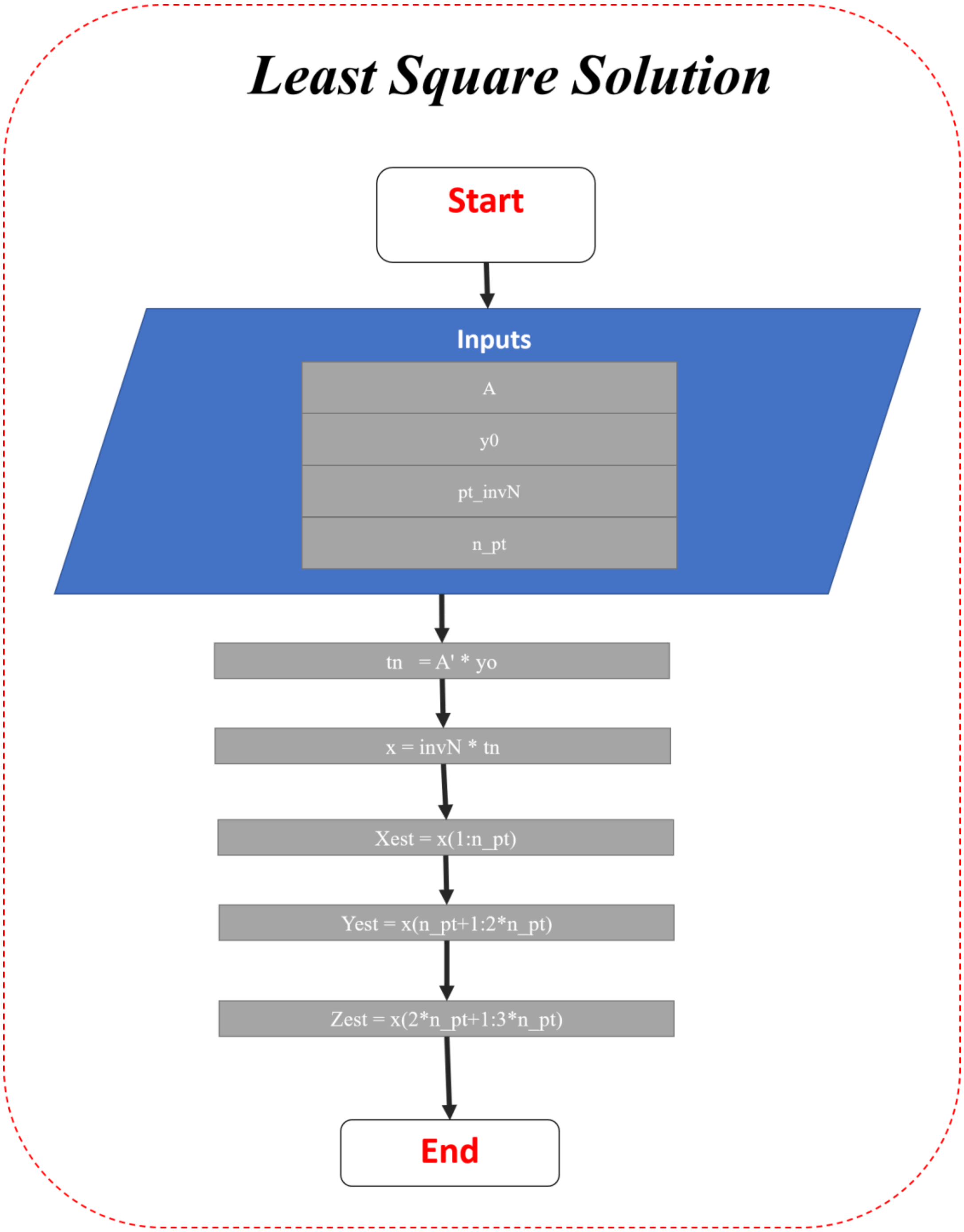

Figure 10 Least square solution.

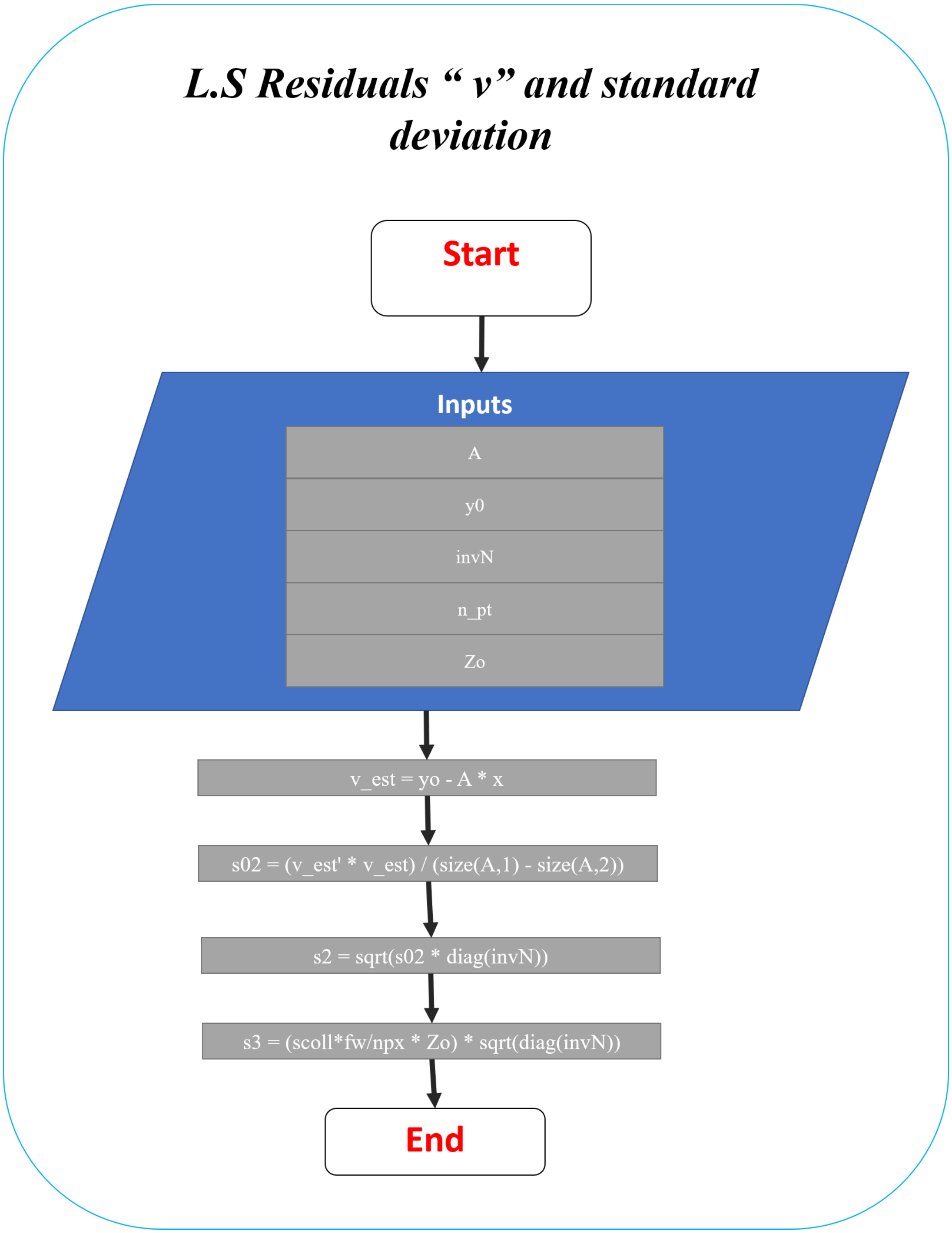

Figure 11 Residuals and standard division calculation

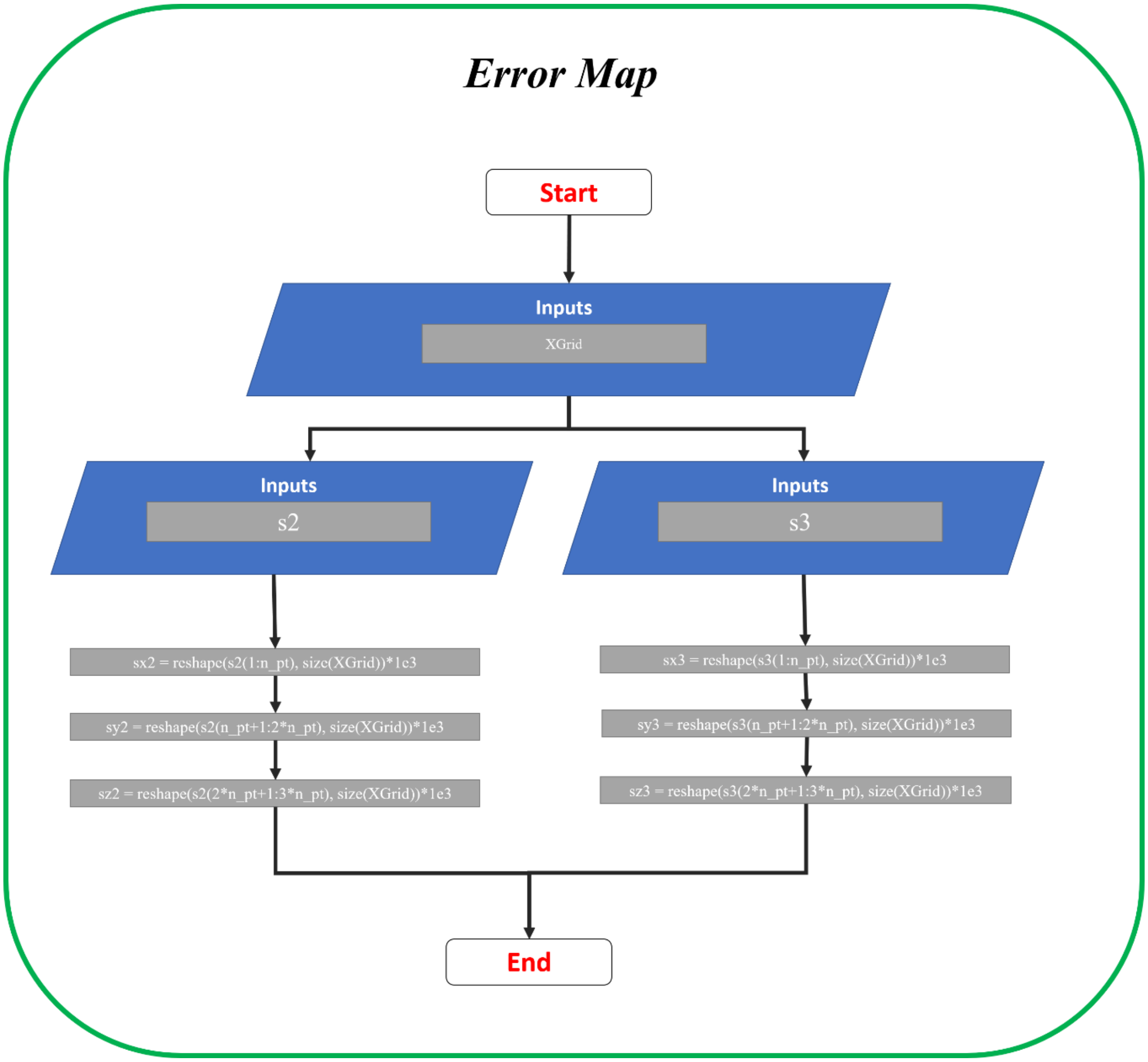

Figure 12 Error map Generation