generateSensorSquareArrayGrid

Generates a position grid of sensors in x, y and z dimension. So the function returns a grid in shape of a square in which all sensors have even distances to each and another in x and y direction z is constant due to that all sensor are in the same distance to the magnet.

The size of the sensor array is described by its edge length a

$$A = a^2$$

and the distance d of each coordinate to the next point in x and y direction

$$d = \frac{a}{N-1}$$

The coordinates of the array are scale from center of the square. So for the upper left corner position is described by

$$x_{1,1} = -\frac{a}{2} \quad y_{1,1} = \frac{a}{2} \quad z = const.$$

The coodinates of each dimension are placed in matrices of size N x N related to the number of sensors at one edge of the square Array. So position pattern in x dimension are returned as

$$X_0 = \begin{bmatrix} x_{1,1} & \cdots & x_{1,N} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,N} \end{bmatrix}$$

$$x_{i,j} = x_{1,1} + j \cdot d - d$$

same wise for y dimension but transposed

$$Y_0 = \begin{bmatrix} y_{1,1} & \cdots & y_{1,N} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,N} \end{bmatrix}$$

$$y_{i,j} = y_{1,1} - i \cdot d + d$$

$$Y_0 = -X_0^T$$

and z dimension

$$Z_0 = \begin{bmatrix} z_{1,1} & \cdots & z_{1,N} \\ \vdots & \ddots & \vdots \\ z_{N,1} & \cdots & z_{N,N} \end{bmatrix}$$

$$z_{i,j} = 0$$

for

$$i = 1, 2, \ldots, N \quad j = 1, 2, \ldots, N$$

A relative position shift can be performed by pass a position vector p with relativ position to center

$$\vec{p} = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$$

So that a left shift in x direction relative to the magnet in the center of the coordinate system is done by negative values for p(1) and an up shift in y direction is performed by positive values for p(2). To gain distance in z from center point so the magnet is above the z layer of the sensor array increase the z positive. In addition to the z shift an offset r sphere can be set. The offset represents the radius of a sphere magnet in which center the dipole is placed. The dipole is placed in the center of the coordinate system and sensor array position is relative to the dipole or center. So shifts are described by

$$X = X_0 + x_p \quad Y = Y_0 + y_p \quad Z = Z_0 - (z_p + r_{sp})$$

Syntax

```
[X, Y, Z] = generateSensorArrayGrid(N, a, p, r)
```

Description

[X, Y, Z] = generateSensorArrayGrid(N, a, p, r) returns a sensor array grid of size N x N with grid position matrices for x, y and z positions of each sensor in the array.

Examples

```
% generate a grid of 8 x 8 sensors with no shift in x or y direction
and a static position of 4mm under the center in z dimension with a
z offset of 2mm so (2 + 2)mm
N = 8;
p = [0, 0, 2]
r = 2;
[X, Y, Z] = generateSensorArrayGrid(N, a, p, r);

% same layer but left shift by 2mm and down shift in y by 1mm
p = [-2, 1, 2]
r = 2;
[X, Y, Z] = generateSensorArrayGrid(N, a, p, r);
```

Input Argurments

N positive integer scalar number of sensors at one edge of the square grid. So the resulting grid has dimensions N x N.

a positive real scalar value of sensor array edge length.

p relative position vector, relative sensor array postion to center of the array. Place the array in 3D coorodinate system relative to the center of system.

r positive real scalar is offset in z dimension and represents the sphere radius in which center the magnetic dipole is placed.

Output Argurments

X x coordinates for each sensor in N x N matrix where each point has the same orientation as in y and z dimension.

Y y coordinates for each sensor in N x N matrix where each point has the same orientation as in x and z dimension.

Z z coordinates for each sensor in N x N matrix where each point has the same orientation as in x and y dimension.

Requirements

- Other m-files required: None
- Subfunctions: meshgrid
- MAT-files required: None

See Also

- meshgrid


```
function [X, Y, Z] = generateSensorArraySquareGrid(N, a, p, r)
    arguments
        % validate N as positive integer
        N (1,1) double {mustBePositive, mustBeInteger}
        % validate array edge length as positive scalar
        a (1,1) double {mustBeReal, mustBePositive}
        % validate p as column vector of real scalars
        p (3,1) double {mustBeReal, mustBeVector}
        % validate r as real scalar
        r (1,1) double {mustBeReal}
    end

    % half edge length for square corners
    aHalf = a / 2;

    % distance in x and y direction of each coordinate to next point
    d = a / (N - 1);

    % grid vector for x and y coordinates z is constant layer with shifts
    x = (-aHalf:d:aHalf) + p(1);
    y = (aHalf:-d:-aHalf) + p(2);
    z = -(p(3) + r);

    % scale grid in x, y dimension with constant z dimension
    [X, Y, Z] = meshgrid(x, y, z);
end
```

Published with MATLAB® R2020b