

---

# mgen Documentation

*Release heads/master*

**NOhs**

**Apr 21, 2023**



---

## Contents:

---

<b>1</b>	<b>Simple usage examples</b>	<b>3</b>
<b>2</b>	<b>Module documentation</b>	<b>5</b>
2.1	2D rotation matrix functions . . . . .	5
2.2	3D rotation matrix functions . . . . .	5
2.3	nD rotation matrix functions . . . . .	6
	<b>Index</b>	<b>9</b>



The mgen package offers functions to generate plain rotation matrices using either of the following options:

- (2D) Give an angle
- (3D) Give three Euler / Tait-Bryan angles
- (3D) Give a rotation axis and an angle
- (3D) Give an angle and a cartesian coordinate axis around which to rotate (x, y, z)
- (nD) Give an angle and a pair of orthogonal vectors that span a plane



# CHAPTER 1

---

## Simple usage examples

---

The following code snippet show how to use the 2D function:

```
>>> import numpy as np
>>> np.set_printoptions(suppress=True)
>>> from mgen import rotation_from_angle

>>> matrix = rotation_from_angle(np.pi/2)
>>> matrix.dot([1, 0])
# array([0., 1.]
```

The following code snippet shows how to use the three different 3D options:

```
>>> import numpy as np
>>> np.set_printoptions(suppress=True)

>>> from mgen import rotation_around_axis
>>> from mgen import rotation_from_angles
>>> from mgen import rotation_around_x

>>> matrix = rotation_from_angles([np.pi/2, 0, 0], 'XYX')
>>> matrix.dot([0, 1, 0])
array([0., 0., 1.])

>>> matrix = rotation_around_axis([1, 0, 0], np.pi/2)
>>> matrix.dot([0, 1, 0])
array([0., 0., 1.])

>>> matrix = rotation_around_x(np.pi/2)
>>> matrix.dot([0, 1, 0])
array([0., 0., 1.]
```

Here is an example of how to create an n-dimensional rotation:

```
>>> import numpy as np
>>> np.set_printoptions(suppress=True)

>>> from mgen import rotation_from_angle_and_plane

>>> matrix = rotation_from_angle_and_plane(np.pi/2, (0, 1, 0, 0), (0, 0, 1, 0))
>>> matrix.dot([0, 1, 0, 0])
# array([0., 0., 1., 0.]
```

Finally an example of a  $n \times n$  random matrix  $\in O(n)$  which can be used to rotate a vector with uniform distribution on the unit  $n$ -sphere:

```
>>> from mgen import random_matrix

>>> n = 6
>>> matrices = np.array([random_matrix(n) for _ in range(100)])
>>> vector = np.zeros(n)
>>> vector[0] = 1.0

# vectors of length one should keep length one
>>> rotated = np.matmul(matrices, vector)
>>> lengths = np.sum(rotated*rotated, axis=1)
>>> print("Minimum length: %.5f; Maximum length: %.5f"%(np.amin(lengths), np.
↪amax(lengths)))
```



## 2.1 2D rotation matrix functions

`mgen.rotation_from_angle` (*angle*)

generates a 2x2 rotation matrix from a given angle following the definition here: [https://en.wikipedia.org/wiki/Rotation\\_matrix](https://en.wikipedia.org/wiki/Rotation_matrix).

**Parameters** `angle` (*float*) – the angle by which to rotate

**Returns** the rotation matrix

**Return type** a 2x2 `numpy.ndarray`

## 2.2 3D rotation matrix functions

`mgen.rotation_from_angles` (*angles, rotation\_sequence*)

Generate a 3x3 rotation matrix using proper Euler angles or Tait-Bryan angles as defined here: [https://en.wikipedia.org/wiki/Euler\\_angles#Rotation\\_matrix](https://en.wikipedia.org/wiki/Euler_angles#Rotation_matrix). The angles given correspond to rotations as given in `rotation_sequence`, e.g.:

```
rotation_from_angles([np.pi/2, 2/3*np.pi, np.pi/3], 'ZYX')
```

would create a rotation matrix equal to the product of a rotation around Z by 90° times a rotation around Y by 60° times a rotation around X by 30°. For all the details please have a look at the linked wikipedia article.

**Parameters**

- **angles** (*array like*) – the three angles in radians that define the rotation as a vector of length 3
- **rotation\_sequence** (*str*) – the sequence of rotations that make up the total rotation. Example: XYZ yields the rotation matrix  $R = XYZ$ , i.e. the product of the three matrices X, Y and Z.

**Returns** the rotation matrix

**Return type** a 3x3 `numpy.array`

`mgen.rotation_around_axis` (*axis*, *angle*)

Generates a 3x3 rotation matrix using the Euler-Rodrigues formula following the definition here: [https://en.wikipedia.org/wiki/Euler%E2%80%93Rodrigues\\_formula](https://en.wikipedia.org/wiki/Euler%E2%80%93Rodrigues_formula).

**Parameters**

- **axis** (*array like*) – the axis around which to rotate as a vector of length 3 (no normalisation required)
- **angle** (*float*) – the angle in radians to rotate

**Returns** the rotation matrix

**Return type** a 3x3 `numpy.ndarray`

`mgen.rotation_around_x` (*angle*)

Generates a 3x3 rotation matrix that performs a rotation around the x-axis following the definitions here: [https://en.wikipedia.org/wiki/Rotation\\_matrix#Basic\\_rotations](https://en.wikipedia.org/wiki/Rotation_matrix#Basic_rotations)

**Parameters** **angle** (*float*) – the angle by which to rotate around the x-axis

**Returns** the rotation matrix

**Return type** a 3x3 `numpy.ndarray`

`mgen.rotation_around_y` (*angle*)

Generates a 3x3 rotation matrix that performs a rotation around the y-axis following the definitions here: [https://en.wikipedia.org/wiki/Rotation\\_matrix#Basic\\_rotations](https://en.wikipedia.org/wiki/Rotation_matrix#Basic_rotations)

**Parameters** **angle** (*float*) – the angle by which to rotate around the y-axis

**Returns** the rotation matrix

**Return type** a 3x3 `numpy.ndarray`

`mgen.rotation_around_z` (*angle*)

Generates a 3x3 rotation matrix that performs a rotation around the z-axis following the definitions here: [https://en.wikipedia.org/wiki/Rotation\\_matrix#Basic\\_rotations](https://en.wikipedia.org/wiki/Rotation_matrix#Basic_rotations)

**Parameters** **angle** (*float*) – the angle by which to rotate around the z-axis

**Returns** the rotation matrix

**Return type** a 3x3 `numpy.ndarray`

## 2.3 nD rotation matrix functions

`mgen.rotation_from_angle_and_plane` (*angle*, *vector1*, *vector2*, *abs\_tolerance=1e-10*)

generates an nxn rotation matrix from a given angle and a plane spanned by two given vectors of length n: [https://de.wikipedia.org/wiki/Drehmatrix#Drehmatrizen\\_des\\_Raumes\\_%27%22%60UNIQ-postMath-0000003B-QINU%60%22'%27](https://de.wikipedia.org/wiki/Drehmatrix#Drehmatrizen_des_Raumes_%27%22%60UNIQ-postMath-0000003B-QINU%60%22'%27)

The formula used is

$$M = +(\cos \alpha - 1) \cdot (v_1 \otimes v_1 + v_2 \otimes v_2) - \sin \alpha \cdot (v_1 \otimes v_2 - v_2 \otimes v_1)$$

with  $M$  being the returned matrix,  $v_1$  and  $v_2$  being the two given vectors and  $\alpha$  being the given angle. It differs from the formula on wikipedia in that it is the transposed matrix to yield results that are consistent with the 2D and 3D cases.

**Parameters**

- **angle** (*float*) – the angle by which to rotate
- **vector1** (*array like*) – one of the two vectors that span the plane in which to rotate (no normalisation required)
- **vector2** (*array like*) – the other of the two vectors that span the plane in which to rotate (no normalisation required)
- **abs\_tolerance** (*float*) – The absolute tolerance to use when checking if vectors have length 0 or are parallel.

**Returns** the rotation matrix

**Return type** an nxn `numpy.ndarray`

`mgen.random_matrix(n)`

Generate a nxn random matrix. The distribution of rotations is uniform on the n-sphere. The random matrix is from the  $O(n)$  group (not  $SO(n)$ ) and uses the gram-schmidt algorithm to orthogonalize the random matrix, see <http://www.ams.org/notices/200511/what-is.pdf> If a rotation from  $SO(n)$  is needed, look for: “A statistical model for random rotations” doi:10.1016/j.jmva.2005.03.009

**Parameters** **n** (*int*) – dimension of space in which the rotation operates

**Returns** rotation matrix

**Return type** a nxn `numpy.ndarray`



## R

`random_matrix()` (*in module mgen*), 7  
`rotation_around_axis()` (*in module mgen*), 6  
`rotation_around_x()` (*in module mgen*), 6  
`rotation_around_y()` (*in module mgen*), 6  
`rotation_around_z()` (*in module mgen*), 6  
`rotation_from_angle()` (*in module mgen*), 5  
`rotation_from_angle_and_plane()` (*in module mgen*), 6  
`rotation_from_angles()` (*in module mgen*), 5