In[1]:= `Needs["SpinDynamica`"]`

```
SpinDynamica version 3.0.1 loaded
```

ModifyBuiltIn: The following built-in routines have been modified in SpinDynamica:
{Chop, Dot, Duration, Exp, Expand, ExpandAll, NumericQ, Plus, Power, Simplify, Times, WignerD}.
Evaluate ??*symbol* to generate the additional definitions for *symbol*.

In[2]:= `?AxesToEuler`

AxesToEuler[system2,system1,opts] gives the Euler angles Ω21={α21,β21,γ21} by which each axis of system2 may be rotated around the system1 axes in order to be brought into coincide with the corresponding axis of system1:

eμ1=Rz1(α21)Ry1(β21)Rz1(γ21).eμ2, where *μ*∈{x,y,z}.

The same result is generated by rotating around the system2 axes:

eμ1=Rz2(α21)Ry2(β21)Rz2(γ21).eμ2, where *μ*∈{x,y,z}.

The same result is also generated by using "moving axes", where the axes follow the Euler rotations, but in this case the angles are applied in reverse order:

eμ1=Rz2'(γ21)Ry2'(β21)Rz2(α21).eμ2,

where the primed axes of system1 follow the rotation.

In a typical application, system2 is the principal axis system P of an anisotropic interaction, and system1 is a reference axis system such as a molecular frame M, or the laboratory frame L. The routine AxesToEuler[P,M] then generates the Euler angles ΩPM which would be used in spherical tensor realisations of the spin Hamiltonian.

Each of system1 and system2 may have the form {Z} or {X,Z} or {X,Y,Z}. Here Z,X and Y are 3-vectors. If only Z is specified, the two remaining axes are constructed. If X and Z is specified, the z-axis of the constructed system is equal to Z and the x-axis is in the XZ plane. If all three axes are specified, they must be a right-handed orthogonal set.
If system1 is missing, the system {ex,ey,ez} is assumed.

AxesToEuler also accepts axis definitions containing symbolic angles; in this case the symbolic result assumes that all angles are in the first quadrant (i.e. greater than 0 and less than *π*/2).

generate a randomly oriented Axis system (specifying the z-axis)

In[3]:= `vecz = RandomReal[{-1, 1}, {3}]`

Out[3]= {0.394581, -0.159527, -0.412323}

In[4]:= `axes = AxisSystem[vecz]`

Out[4]= {{0.74607, 0.240267, 0.621009},
{0., -0.932631, 0.360833}, {0.665868, -0.269206, -0.695807}}

derive the Euler angles relating the axis system to the original frame

In[5]:= `{α, β, γ} = AxesToEuler[axes]`

Out[5]= {0.526364, 2.34034, -2.75739}

In[6]:= `{α, β, γ} / °`

Out[6]= {30.1584, 134.092, -157.987}

rotate the axis system back to the original position

In[7]:=
```
Chop@RotateEuler[axes, {α, β, γ}]
```
Out[7]=
```
{{1., 0, 0}, {0, 1., 0}, {0, 0, 1.}}
```

generate a randomly oriented Axis system (specifying two axes)

In[8]:=
```
vecx = RandomReal[{-1, 1}, {3}]
vecz = RandomReal[{-1, 1}, {3}]
```
Out[8]=
```
{0.00705992, -0.289924, -0.926157}
```
Out[9]=
```
{-0.641769, 0.0301905, -0.0273466}
```

In[10]:=
```
axes = AxisSystem[vecx, vecz]
```
Out[10]=
```
{{0.0265384, -0.299698, -0.953665},
 {-0.0575177, -0.952878, 0.29785}, {-0.997992, 0.0469481, -0.0425258}}
```

derive the Euler angles relating the axis system to the original frame

In[11]:=
```
{α, β, γ} = AxesToEuler[axes]
```
Out[11]=
```
{2.83887, 1.61333, 0.0470079}
```

In[12]:=
```
{α, β, γ} / °
```
Out[12]=
```
{162.655, 92.4373, 2.69336}
```

rotate the axis system back to the original position

In[13]:=
```
Chop@RotateEuler[axes, {α, β, γ}]
```
Out[13]=
```
{{1., 0, 0}, {0, 1., 0}, {0, 0, 1.}}
```