Coordinate systems

Mauro Mongiardo[1]

[1] Department of Engineering, University of Perugia, Perugia, Italy.

Table of contents

Coordinate systems

Coordinate systems

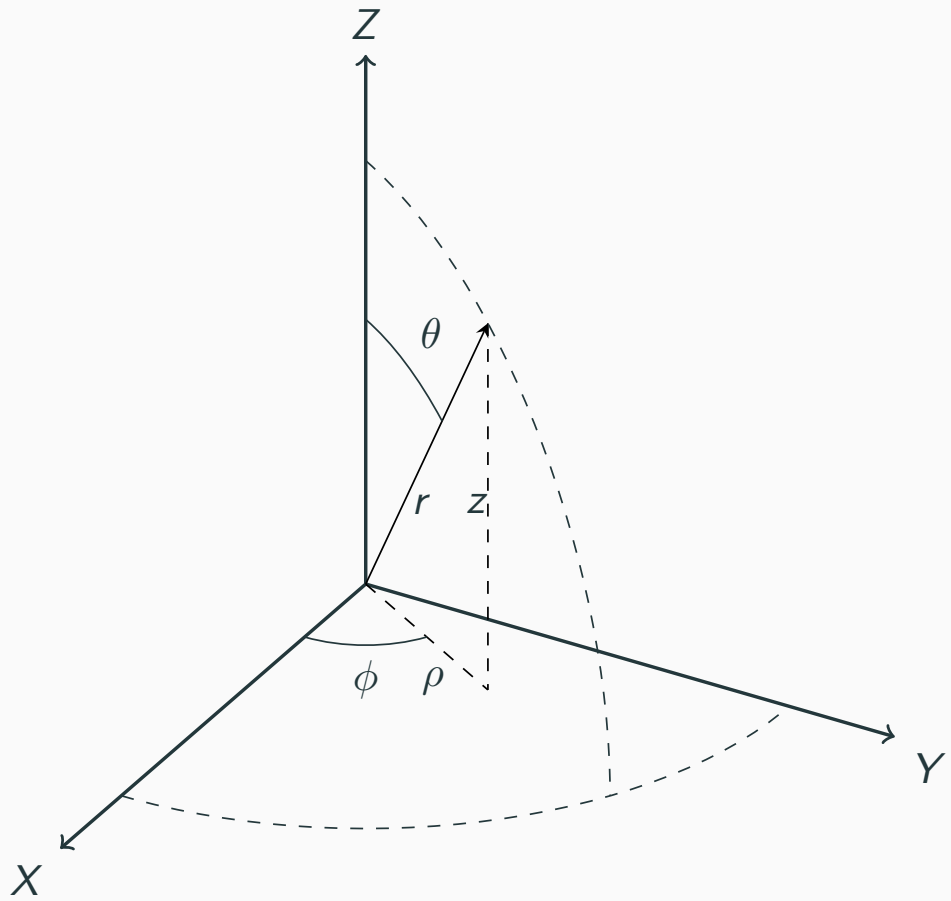

Figure 1: Convention for coordinate orientation.

We will use cartesian, circular cylindrical and spherical coordinates systems.

Coordinates transformations

Cylindrical Coordinates

When passing from rectangular to cylindrical coordinates the following transformations apply:

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2} = r \sin\theta \\ \phi &= \tan^{-1}\left(\frac{y}{x}\right) \\ z &= r \cos\theta \end{aligned} \tag{1}$$

From cylindrical to rectangular:

$$\begin{aligned} x &= \rho \cos\phi \\ y &= \rho \sin\phi \\ z &= z \end{aligned} \tag{2}$$

The volume element is

$$dV = \rho \, d\rho \, d\phi \, dz \tag{3}$$

Spherical Coordinates

Rectangular to spherical

$$\begin{aligned} r &= \sqrt{x^2+y^2+z^2} = \sqrt{\rho^2+z^2} \\ \theta &= \tan^{-1}\left(\frac{\sqrt{x^2+y^2}}{z}\right) = \tan^{-1}\left(\frac{\rho}{z}\right) \\ \phi &= \tan^{-1}\left(\frac{y}{x}\right). \end{aligned} \tag{4}$$

and for the spherical to rectangular

$$\begin{aligned} x &= r\,\sin\theta\,\cos\phi \\ y &= r\,\sin\theta\,\sin\phi \\ z &= r\cos\theta\,. \end{aligned} \tag{5}$$

The volume element is

$$dV = r^2\sin\theta\,dr\,d\theta\,d\phi \tag{6}$$

Examples

Before proceeding further complete examples 3.4, 3.5, 3.6

Unit vectors

Unit vectors: transformations from cylindrical/spherical to rectangular

The coordinate unit vectors are defined as vectors of unit length pointing along coordinate lines in the direction of increasing coordinate variables.

It is left as an exercise to draw them in the three coordinate systems. The following rules are applicable to transformations among unit coordinate vectors.

Transformations from cylindrical/spherical to rectangular:

$$\begin{aligned}
\mathbf{u}_x &= \mathbf{u}_\rho \cos\phi - \mathbf{u}_\phi \sin\phi \\
&= \mathbf{u}_r \sin\theta \cos\phi + \mathbf{u}_\theta \cos\theta \cos\phi - \mathbf{u}_\phi \sin\phi \\
\mathbf{u}_y &= \mathbf{u}_\rho \sin\phi + \mathbf{u}_\phi \cos\phi \\
&= \mathbf{u}_r \sin\theta \sin\phi + \mathbf{u}_\theta \cos\theta \sin\phi + \mathbf{u}_\phi \cos\phi \\
\mathbf{u}_z &= \mathbf{u}_r \cos\theta - \mathbf{u}_\theta \sin\theta
\end{aligned} \tag{7}$$

Transformations from rectangular/spherical to cylindrical

Transformations from rectangular/spherical to cylindrical

$$\begin{aligned} \mathbf{u}_\rho &= \mathbf{u}_x \cos\phi + \mathbf{u}_y \sin\phi = \mathbf{u}_r \sin\theta + \mathbf{u}_\theta \cos\theta \\ \mathbf{u}_\phi &= -\mathbf{u}_x \sin\phi + \mathbf{u}_y \cos\phi \\ \mathbf{u}_z &= \mathbf{u}_r \cos\theta - \mathbf{u}_\theta \sin\theta \end{aligned} \tag{8}$$

Transformations from rectangular/cylindrical to spherical

Transformations from rectangular/cylindrical to spherical:

$$
\begin{aligned}
\mathbf{u}_r &= \mathbf{u}_x \sin\theta \cos\phi + \mathbf{u}_y \sin\theta \sin\phi + \mathbf{u}_z \cos\theta \\
&= \mathbf{u}_\rho \sin\theta + \mathbf{u}_z \cos\theta \\
\mathbf{u}_\theta &= \mathbf{u}_x \cos\theta \cos\phi + \mathbf{u}_y \cos\theta \sin\phi - \mathbf{u}_z \sin\theta \\
&= \mathbf{u}_\rho \cos\theta - \mathbf{u}_z \sin\theta \\
\mathbf{u}_\phi &= -\mathbf{u}_x \sin\phi + \mathbf{u}_y \cos\phi
\end{aligned} \tag{9}
$$

Coordinate transformations for vector components

Coordinate transformations for vector components

Rectangular to cylindrical components

$$\begin{aligned} F_\rho &= F_x \cos\phi + F_y \sin\phi \\ F_\phi &= -F_x \sin\phi + F_y \cos\phi \\ F_z &= F_z \end{aligned} \tag{10}$$

Rectangular to spherical components

$$\begin{aligned} F_r &= F_x \sin\theta \cos\phi + F_y \sin\theta \sin\phi + F_z \cos\theta \\ F_\theta &= F_x \cos\theta \cos\phi + F_y \cos\theta \sin\phi - F_z \sin\theta \\ F_\phi &= -F_x \sin\phi + F_y \cos\phi \end{aligned} \tag{11}$$

Cylindrical to rectangular components

$$\begin{aligned} F_x &= F_\rho \cos\phi - F_\phi \sin\phi \\ F_y &= F_\rho \sin\phi + F_\phi \cos\phi \\ F_z &= F_z \end{aligned} \tag{12}$$

Cylindrical to spherical components

$$\begin{aligned} F_r &= F_\rho \sin\theta + F_z \cos\theta \\ F_\theta &= F_\rho \cos\theta - F_z \sin\theta \\ F_\phi &= F_\phi \end{aligned} \tag{13}$$

Coordinate systems

Spherical to rectangular components

$$\begin{aligned} F_x &= F_r \sin\theta \cos\phi + F_\theta \cos\theta \cos\phi - F_\phi \sin\phi \\ F_y &= F_r \sin\theta \sin\phi + F_\theta \cos\theta \sin\phi + F_\phi \cos\phi \\ F_z &= F_r \cos\theta - F_\theta \sin\phi \end{aligned} \tag{14}$$

Spherical to cylindrical components

$$\begin{aligned} F_\rho &= F_r \sin\theta + F_\theta \cos\theta \\ F_\phi &= F_\phi \\ F_z &= F_r \cos\theta - F_\theta \sin\theta \end{aligned} \tag{15}$$

Summary of coordinate transformations:r2c

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$

/* numerical
fpprintprec:5$
ratprint : false$
x :5.0 ;
y :2.0 ;
z :4.6 ;
*/

/* rectangular to cylindrical */
%rho : sqrt(x^2+y^2)$
phi : atan2(y,x)$
z :z$
print("rectangular to cylindrical")$
print('%rho, " = ",%rho)$
print('phi, " = ",phi)$
print('z, " = ",z)$

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of coordinate transformations:c2r

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* c2r.wxm */

/* numerical
fpprintprec:5$
ratprint : false$
%rho :5.0 ;
phi : 3.14/4.0 ;
z :4.6 ;
 */

/* cylindrical to rectangular */
x : %rho * cos(phi)$
y : %rho * sin(phi)$
z : z$

print("cylindrical to rectangular")$
print('x, " = ",x)$
print('y, " = ",y)$
print('z, " = ",z)$

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of coordinate transformations: r2s

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* r2s.wxm */

/* numerical
fpprintprec:5$
ratprint : false$
x :5.0 ;
y :2.0 ;
z :4.6 ;
*/

r : sqrt(x^2+y^2+z^2)$
theta : atan2(sqrt(x^2 +y^2),z)$
phi : atan2(y,x)$

print("rectangular to spherical")$
print('r, " = ",r)$
print('theta, " = ",theta)$
print('phi, " = ",phi)$

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of coordinate transformations: s2r

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* s2r.wxm */

/* numerical
fpprintprec:5$
ratprint : false$
r :5.0 ;
theta :float(%pi)/4 ;
phi :4.6 ;
*/

x : r * sin(theta) * cos(phi)$
y : r * sin(theta) * sin(phi)$
z : r *cos(theta)$

print("spherical to rectangular")$
print('x, " = ",x)$
print('y, " = ",y)$
print('z, " = ",z)$

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of coordinate transformations: c2s

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* c2s.wxm */

/* numerical
fpprintprec:5$
ratprint : false$
%rho :5.0 ;
phi :4.6 ;
z :2.0;
*/

/* cylindrical to spherical */
r : sqrt(%rho^2 + z^2)$
theta  :atan2(%rho,z) $
phi : phi$

print("cylindrical to spherical")$
print('r, " = ",r)$
print('theta, " = ",theta)$
print('phi, " = ",phi)$

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of coordinate transformations: s2c

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* s2c.wxm */

/* numerical */
fpprintprec:5$
ratprint : false$
r :5.0 ;
theta :float(%pi)/4 ;
phi :0.6 ;


%rho : r * sin(theta) $
phi : phi$
z : r *cos(theta)$

print("spherical to cylindrical")$
print('%rho, " = ",%rho)$
print('phi, " = ",phi)$
print('z, " = ",z)$

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of vector transformations: v r2c

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* V_r2c.wxm */

print(" from the cartesian expression of a vector pass to the cylindrical representation")$
/* numerical
fpprintprec:5$
ratprint : false$
A[x] :5.0 ;
A[y] : 3.14/4.0 ;
A[z] :4.6 ;
phi : atan2(A[y],A[x]);
*/

/* for unit vectors the following holds:
A[x] : x[0];
A[y] : y[0];
A[z] : z[0];
*/

print("Transformation matrix r2cmat")$
r2cmat :matrix([cos(phi),sin(phi),0],[-sin(phi),cos(phi),0],[0,0,1]);

print("vector A in rectangular coordinates")$
Avect : [A[x],A[y],A[z]];

print("rectangular to cylindrical")$
Aspher : r2cmat . Avect ;

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of vector transformations: v c2r

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* V_r2c.wxm */

print(" from the cylindrical expression of a vector pass to the cartesian representation")$
/* numerical  */
fpprintprec:5$
ratprint : false$
Arho :1.0 ;
Aphi : 3.14/4.0 ;
Az : 1.0 ;
print("in order to evaluate phi we need to know where the vector is to be evaluated!!!")$
x :1.0; y :1.0 ; phi : float(atan2(y,x));


/* for unit vectors the following holds:
A%rho : %rho[0];
Aphi : phi[0];
Az : z[0];
*/

print("Transformation matrix r2cmat")$
r2cmat :matrix([cos(phi),sin(phi),0],[-sin(phi),cos(phi),0],[0,0,1])$
c2rmat:trigsimp(invert(r2cmat));
c2rmat:float(c2rmat);

print("vector A in cylindrical coordinates")$
Acyl : [Arho,Aphi,Az];

print("cylindrical to rectangular")$
Arect : c2rmat . Acyl ;

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of vector transformations: v r2s

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* V_c2s.wxm */

print(" from the cartesian expression of a vector express the spherical representation")$
/* numerical
fpprintprec:5$
ratprint : false$
Ax :1.0 ;
Ay : 0.0 ;
Az : 0.0 ;
print("coordinates where the vector is evaluated")$
Px: 0.0;
Py: 1.0;
Pz: 0.0;
phi : float(atan2(Py,Px));
theta : float(atan2(sqrt(Px^2+Py^2),Pz));
*/

/* for unit vectors the following holds:
Ax : x0;
Ay : y0;
Az : z0;
*/

print("Transformation matrix s2rmat")$
r2smat :matrix([sin(theta)* cos(phi),sin(theta)*sin(phi),cos(theta)],
               [cos(theta)*cos(phi),cos(theta)*sin(phi),-sin(theta)],
               [-sin(phi),cos(phi),0]);

print("vector A in rectangular coordinates")$
Arect : [Ax,Ay,Az];

print("rectangular to spherical")$
Aspher : r2smat . Arect ;

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of vector transformations: v s2r

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* V_s2r.wxm */

print("From the spherical expression of a vector express the cartesian representation")$
/* numerical
fpprintprec:5$
ratprint : false$
Ar :1.0 ;
Atheta : 0.0 ;
Aphi : 0.0 ;
print("coordinates where the vector is evaluated")$
Px: 0.0;
Py: 1.0;
Pz: 0.0;
phi : float(atan2(Py,Px));
theta : float(atan2(sqrt(Px^2+Py^2),Pz));
*/


print("Transformation matrix s2rmat")$
r2smat :matrix([sin(theta)* cos(phi),sin(theta)*sin(phi),cos(theta)],
               [cos(theta)*cos(phi),cos(theta)*sin(phi),-sin(theta)],
               [-sin(phi),cos(phi),0])$
s2rmat : trigsimp(invert(r2smat));


print("vector A in spherical coordinates")$
Aspher : [Ar,Atheta,Aphi];

print("spherical to rectangular")$
Arect : s2rmat . Aspher ;

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of vector transformations: v c2s

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* V_c2s.wxm */

print("From the cylindrical representation of a vector express the spherical one")$
/* numerical
fpprintprec:5$
ratprint : false$
Arho :0.0 ;
Aphi : 0.0 ;
Az : 1.0 ;
print("coordinates where the vector is evaluated")$
Px: 0.0;
Py: 1.0;
Pz: 0.0;
phi : float(atan2(Py,Px));
theta : float(atan2(sqrt(Px^2+Py^2),Pz));
*/

print("Transformation matrix c2smat")$
c2smat :matrix([sin(theta),0,cos(theta)],
               [cos(theta),0,-sin(theta)],
               [0,1,0]);

print("vector A in cyl coordinates")$
Acyl : [Arho,Aphi,Az];

print("cylindrical to spherical")$
Aspher : c2smat . Acyl ;

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Summary of vector transformations: v s2c

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* V_c2s.wxm */

print("From the spherical representation of a vector express the cylindrical one")$
/* numerical */
fpprintprec:5$
ratprint : false$
Ar :0.0 ;
Atheta : -1.0 ;
Aphi : 1.0 ;
print("coordinates where the vector is evaluated")$
Px: 0.0;
Py: 1.0;
Pz: 0.0;
phi : float(atan2(Py,Px));
theta : float(atan2(sqrt(Px^2+Py^2),Pz));


print("Transformation matrix c2smat")$
c2smat :matrix([sin(theta),0,cos(theta)],
               [cos(theta),0,-sin(theta)],
               [0,1,0])$
s2cmat : trigsimp(invert(c2smat));


print("vector A in spher coordinates")$
Aspher : [Ar,Atheta,Aphi];


print("spherical to cylindrical")$
Acyl : s2cmat . Aspher ;
/* */
print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Pauli matrices in Cylindrical co-ordinates

Pauli matrices in Cylindrical coordinates

We have already introduced the Pauli matrices for the rectangular coordinate system σ_i, with $i = 0, \ldots, 3$. When expressing a vector can identify σ_1 with $\mathbf{u}_x$, σ_2 with $\mathbf{u}_y$ and σ_3 with $\mathbf{u}_z$. By using (8) we can write:

$$\begin{aligned} \mathbf{u}_\rho &= \mathbf{u}_x \cos\phi + \mathbf{u}_y \sin\phi \\ \sigma_\rho &= \cos\phi\sigma_1 + \sin\phi\sigma_2 \\ &= \begin{pmatrix} 0 & e^{-i\phi} \\ e^{i\phi} & 0 \end{pmatrix}. \end{aligned} \tag{16}$$

Similarly, for the unit vector along ϕ we can write

$$\begin{aligned} \mathbf{u}_\phi &= -\mathbf{u}_x \sin\phi + \mathbf{u}_y \cos\phi \\ \sigma_\phi &= -\sin\phi\sigma_1 + \cos\phi\sigma_2 \\ &= \begin{pmatrix} 0 & -i\,e^{-i\phi} \\ i\,e^{i\phi} & 0 \end{pmatrix}. \end{aligned} \tag{17}$$

Pauli matrices in Cylindrical coordinates

Naturally, for the z component we have that $\sigma_z = \sigma_3$. Summarizing, we have the following three matrices in cylindrical coordinates

$$\begin{aligned} \sigma_\rho &= \begin{pmatrix} 0 & e^{-i\,\phi} \\ e^{i\,\phi} & 0 \end{pmatrix} \\ \sigma_\phi &= \begin{pmatrix} 0 & -i\,e^{-i\,\phi} \\ i\,e^{i\,\phi} & 0 \end{pmatrix} \\ \sigma_z &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \end{aligned} \tag{18}$$

Pauli matrices in Cylindrical coordinates

It is readily proved that the matrices $\sigma_\rho, \sigma_\phi, \sigma_z$ when multiplied by themselves give the identity matrix σ_0, their trace is null, their determinant is always -1 and their dot product is zero if they are not the same. In addition we have

$$\begin{aligned} i\,\sigma_z &= \sigma_\rho \sigma_\phi \\ i\,\sigma_\rho &= \sigma_\phi \sigma_z \\ i\,\sigma_\phi &= \sigma_z \sigma_\rho\,. \end{aligned} \tag{19}$$

Therefore the generic vector **A** can be expressed in cylindrical coordinates in terms of Pauli matrices as

$$\begin{aligned} \tilde{A} &= A_\rho \sigma_\rho + A_\phi \sigma_\phi + A_z \sigma_z \\ &= \begin{pmatrix} A_z & e^{-i\,\phi}\,(A_\rho - i\,A_\phi) \\ e^{i\,\phi}\,(A_\rho + i\,A_\phi) & -A_z \end{pmatrix}. \end{aligned} \tag{20}$$

Pauli matrices in Cylindrical coordinates

It is noted that since the vector is the same when expressed in rectangular or cylindrical components we have the identity

$$A_\rho \sigma_\rho + A_\phi \sigma_\phi + A_z \sigma_z = A_x \sigma_1 + A_y \sigma_2 + A_z \sigma_3 \,. \tag{21}$$

By performing the dot product of the above expression with a selected sigma matrix we can recover the desired field component in terms of the other basis.

Pauli matrices in Cylindrical coordinates

As an example let us consider to retrieve the expression of A_ρ in terms of the rectangular components. It is sufficient to dot multiply both sides of (27) times σ_ρ to retrieve

$$\begin{aligned} A_\rho &= A_x \sigma_1 \cdot \sigma_\rho + A_y \sigma_2 \cdot \sigma_\rho + A_z \sigma_3 \cdot \sigma_\rho \\ &= A_x \cos\phi + A_y \sin\phi \end{aligned} \tag{22}$$

where we remind that performing the dot product is equivalent to perform the matrix product and take half of the trace. It is left as an exercise for the reader to recover the results in the previous section by using Pauli matrices multiplication.

Pauli matrices in Cylindrical coordinates

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
/* Pauli.mac */
print(" ***********************************************************")$
print(" Pauli Matrices Definition ")$
print(" ***********************************************************")$

Sigma[0] : matrix([1,0],[0,1])$
Sigma[1] : matrix([0,1],[1,0])$
Sigma[2] : matrix([0,-%i],[%i,0])$
Sigma[3] : matrix([1,0],[0,-1])$

print("Sigma[0] =",Sigma[0])$
print("Sigma[1] =",Sigma[1])$
print("Sigma[2] =",Sigma[2])$
print("Sigma[3] =",Sigma[3])$

print(" ***********************************************************")$
print("Obtaining Pauli matrices in cylindrical coordinates")$
print(" ***********************************************************")$
print("Representation of the unit vector rho as")$
print(cos(phi)* Sigma[1] + sin(phi)* Sigma[2])$
rho : cos(phi)* Sigma[1] + sin(phi)* Sigma[2]$
rho : ratsimp(expand(rho))$

print("Representation of the unit vector (phi as")$
print(- sin(phi)* Sigma[1] + cos(phi)* Sigma[2])$
aphi : - sin(phi)* Sigma[1] + cos(phi)* Sigma[2]$
aphi : ratsimp(expand(aphi));

print(" ***********************************************************")$
print(" Pauli Matrices Definition Cylindrical")$
print(" ***********************************************************")$

Sigma[%rho] : matrix([0,%e^(-%i * phi)],[%e^(%i * phi),0])$
Sigma[phi] : matrix([0,-%i * %e^(-%i * phi)],[%i * %e^(%i * phi),0])$
Sigma[z] : matrix([1,0],[0,-1])$

print(Sigma[%rho]," = ",Sigma[%rho])$
print(Sigma[phi]," = ",Sigma[phi])$
print(Sigma[z]," = ",Sigma[z])$

print("Squared matrices")$
Sigma[%rho] . Sigma[%rho];
Sigma[phi] . Sigma[phi];
Sigma[z] . Sigma[z];

print("Alternating rules")$
Sigma[%rho] . Sigma[phi] + Sigma[phi] . Sigma[%rho];
Sigma[phi] . Sigma[z] + Sigma[z] . Sigma[%rho];
Sigma[z] . Sigma[%rho] + Sigma[%rho] . Sigma[z];

print("Determinants")$
determinant(Sigma[%rho]);
determinant(Sigma[phi]);
determinant(Sigma[z]);

print("vector in cylindrical coordinate")$
Acyl : factor(A[%rho]*Sigma[%rho]+A[phi]*Sigma[phi]+A[z]*Sigma[3]);

print("vector decomposition")$
print("obtaining A[rho] components")$

print("Dot multiplication with Sigma[%rho] allows to retrieve the rho component")$
w1 : A[x] * trigsimp(Sigma[%rho] . Sigma[1])
   + A[y] * trigsimp(Sigma[%rho] . Sigma[2])
   + A[z] * trigsimp(Sigma[%rho] . Sigma[3]) ;
Arho : factor(realpart((1/2 * (w1[1,1]+w1[2,2]))));

print("Dot multiplication with Sigma[%phi] allows to retrieve the phi component")$
w1 : A[x] * trigsimp(Sigma[phi] . Sigma[1])
   + A[y] * trigsimp(Sigma[phi] . Sigma[2])
   + A[z] * trigsimp(Sigma[phi] . Sigma[3]) ;
Aphi : ratsimp(factor(realpart((1/2 * (w1[1,1]+w1[2,2])))));

w1 : A[x] * trigsimp(Sigma[3] . Sigma[1])
   + A[y] * trigsimp(Sigma[3] . Sigma[2])
   + A[z] * trigsimp(Sigma[3] . Sigma[3])
Az : ratsimp(factor(realpart((1/2 * (w1[1,1]+w1[2,2])))));

print(" ***********************************************************")$
print("Numerical example (see example 2.7)")$
print(" ***********************************************************")$

fpprintprec :5$
ratprint : false$
print("Data for given point P1")$
P1[x] : 2.0$
P1[y] : -4.0$
P1[z] : 3.0$

print("and a given vector A defined in cartesian coordinates")$
A[x] : 2.0$
A[y] : -3.0$
A[z] : 4.0$

print(" ***********************************************************")$
print("Transformation of coordinates !!!")$
print(" ***********************************************************")$

print("phi value for P1")$
phi : float(atan2(P1[y],P1[x]));

print("point P1 as column vector")$
P1xyz : matrix([P1[x]],[P1[y]],[P1[z]]);

print("vector A as column vector")$
Axyz : matrix([A[x]],[A[y]],[A[z]]);

print("From rectangular to cylindrical ...")$
print("Point P1 expressed in rectangular is transformed in point P1 in cylindrical")$

P1rho : sqrt(P1[x]^2+P1[y]^2);
P1phi : float(atan2(P1[y],P1[x]));
P1z : P1[z];

print("the phi component in degrees is" phi*180.0/float(%pi))$
print(" or equivalently ", 360.0 + phi*180.0/float(%pi))$

print(" ***********************************************************")$
print("Transformation of vector components !!!")$
print(" ***********************************************************")$

print("A is considered at a vector")$
print("transformation matrix from rect to cyl")$
r2c : matrix([cos(phi),sin(phi),0],[-sin(phi),cos(phi),0],[0,0,1]);

print("Vector in cylindrical coordinates")$
Acyl : r2c . Axyz;

print(" ***********************************************************")$
print("Using Pauli matrices")$
print(" ***********************************************************")$

print("Pauli vector in rectangular coordinates")$
/* Arect : matrix([A[x]], A[y], A[z]]); */
Arect : ratsimp(A[x] * Sigma[1] + A[y] * Sigma[2]+ A[z] * Sigma[3]);


print("Pauli matrices for the cylindrical coordinate system")$
print("They are dependent on the phi value!")$
Sigma[%rho] : matrix([0,%e^(-%i * phi)],[%e^(%i * phi),0]);
Sigma[phi] : matrix([0,-%i * %e^(-%i * phi)],[%i * %e^(%i * phi),0]);
Sigma[z] : matrix([1,0],[0,-1]);

print("write the A[%rho] . A[phi] . A[z] components from conventional analysis")$
Arho : Axyz[1][1]$
Aaphi : Axyz[2][1]$
Az : Axyz[3][1]$

print("Pauli vector in cylindrical coordinates")$
Apcyl : rectform(Arho * Sigma[%rho] + Aaphi * Sigma[phi] + Az * Sigma[3]);
print("and the difference between Pauli matrix in cylindrical and in rectangular coordinates")$

(Arect-Apcyl);

print("THEY ARE THE SAME !!!")$

print("How to derive the cylindrical components from the Pauli matrix?")$
print("In order to obtain A[rho] just multiply the Pauli matrix ")$
print(" with Sigma[%rho] and take half of the trace")$
Apc : rectform(Arect . Sigma[%rho]);
Aprho : 1.0/2.0 *(Apc[1,1]+Apc[2,2]);

print("proceed similarly for the components along phi")$
Apq : rectform(Arect . Sigma[phi]);
Apa : 1.0/2.0 *(Apq[1,1]+Apq[2,2]);

print("and finally")$
Apz : rectform(Arect . Sigma[3]);
Apz : 1.0/2.0 *(Apz[1,1]+Apz[2,2]);

print("end")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Pauli matrices in Spherical coordinates

Pauli matrices in Spherical coordinates

So far we have introduced the Pauli matrices for the rectangular and cylindrical coordinate systems. We can proceed in the same way that we have followed for the cylindrical coordinate system. By using (9) we can write:

$$\begin{aligned} \mathbf{u}_r &= \mathbf{u}_x \sin\theta \cos\phi + \mathbf{u}_y \sin\theta \sin\phi + \mathbf{u}_z \cos\theta \\ \sigma_r &= \sin\theta \cos\phi\, \sigma_1 + \sin\theta \sin\phi\, \sigma_2 + \cos\theta\, \sigma_3 \\ &= \begin{pmatrix} \cos\theta & e^{-i\phi} \sin\theta \\ e^{i\phi} \sin\theta & -\cos\theta \end{pmatrix}. \end{aligned} \tag{23}$$

Pauli matrices in Spherical coordinates

Similarly, for the unit vector along θ we can write:

$$\begin{aligned} \mathbf{u}_\theta &= \mathbf{u}_x \cos\theta \cos\phi + \mathbf{u}_y \cos\theta \sin\phi - \mathbf{u}_z \sin\theta \\ \sigma_\theta &= \cos\theta \cos\phi\, \sigma_1 + \cos\theta \sin\phi\, \sigma_2 - \sin\theta\, \sigma_3 \\ &= \begin{pmatrix} -\sin\theta & e^{-i\phi}\cos\theta \\ e^{i\phi}\cos\theta & \sin\theta \end{pmatrix}. \end{aligned} \tag{24}$$

Not surprisingly for the ϕ coordinate the result is the same as in cylindrical coordinates.

Pauli matrices in Spherical coordinates

Summarizing, we have the following three matrices in spherical coordinates

$$\begin{aligned}
\sigma_r &= \begin{pmatrix} \cos\theta & e^{-i\,\phi}\,\sin\theta \\ e^{i\,\phi}\,\sin\theta & -\cos\theta \end{pmatrix} \\
\sigma_\theta &= \begin{pmatrix} -\sin\theta & e^{-i\,\phi}\,\cos\theta \\ e^{i\,\phi}\,\cos\theta & \sin\theta \end{pmatrix} \\
\sigma_\phi &= \begin{pmatrix} 0 & -i\,e^{-i\,\phi} \\ i\,e^{i\,\phi} & 0 \end{pmatrix} .
\end{aligned} \tag{25}$$

Pauli matrices in Spherical coordinates

It is readily proved that the matrices $\sigma_r, \sigma_\theta, \sigma_\phi$ when multiplied by themselves give the identity matrix σ_0, their trace is null, their determinant is always -1 and their dot product is zero if they are not the same. In addition we have:

$$\begin{aligned} i\,\sigma_\phi &= \sigma_r\sigma_\theta \\ i\,\sigma_r &= \sigma_\theta\sigma_\phi \\ i\,\sigma_\theta &= \sigma_\phi\sigma_r\,. \end{aligned} \tag{26}$$

Pauli matrices in Spherical coordinates

Therefore, the generic vector **A** can be expressed in spherical coordinates in terms of Pauli matrices as:

$$\begin{aligned}\tilde{A} &= A_r\sigma_r + A_\theta\sigma_\theta + A_\phi\sigma_\phi \\ &= \begin{pmatrix} A_r\,\cos\theta - A_\theta\,\sin\theta & e^{-i\,\phi}\,(A_r\,\sin\theta\, - i\,A_\phi + A_\theta\,\cos\theta) \\ e^{i\,\phi}\,(A_r\,\sin\theta\, + i\,A_\phi + A_\theta\,\cos\theta) & -A_r\,\cos\theta + A_\theta\,\sin\theta \end{pmatrix} \end{aligned} \quad (27)$$

Pauli matrices in Spherical coordinates

It is noted that since the vector is the same when expressed in rectangular, cylindrical or spherical components we have the identity

$$\begin{aligned} \tilde{A} &= A_x\sigma_1 + A_y\sigma_2 + A_z\sigma_3 \\ &= A_\rho\sigma_\rho + A_\phi\sigma_\phi + A_z\sigma_z \\ &= A_r\sigma_r + A_\theta\sigma_\theta + A_\phi\sigma_\phi \,. \end{aligned} \tag{28}$$

By performing the dot product of the above expression with a selected sigma matrix we can recover the desired field component in terms of the other basis.

Thus all the transformation between vectors in different coordinate systems can be simply obtained by matrix multiplication and trace operation.

Pauli matrices in Spherical coordinates

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$

print("  ***************************************************************")$
print("  Pauli Matrices Definition")$
print("  ***************************************************************")$

%sigma[0] :  matrix([1,0],[0,1])$
%sigma[1] :  matrix([0,1],[1,0])$
%sigma[2] :  matrix([0,-%i],[%i,0])$
%sigma[3] :  matrix([1,0],[0,-1])$

print("%sigma[0] = ",%sigma[0])$
print("%sigma[1] = ",%sigma[1])$
print("%sigma[2] = ",%sigma[2])$
print("%sigma[3] = ",%sigma[3])$

Axyz : A[x] * %sigma[1] + A[y] * %sigma[2]+ A[z] * %sigma[3]$
print("vector A in cartesian coordinates",":" = ,Axyz)$

print("  ***************************************************************")$
print("  Obtaining Pauli matrices in spherical coordinates")$
print("  ***************************************************************")$

print("Representation of the unit vector ur as")$
print(sin(%itheta) * cos(phi)* %sigma[1] + sin(%itheta) * sin(phi)* %sigma[2] + cos(%itheta) * %sigma[3])$
ur : sin(%itheta) * cos(phi)* %sigma[1] + sin(%itheta) * sin(phi)* %sigma[2] + cos(%itheta) * %sigma[3]$
ur : factor(ur);
xdot : rationp(exponentialize((ur[2,1]/ sin(%itheta)))$
ur[2,1] : xdot*sin(%itheta)$
xdot : rationp(exponentialize((ur[1,2]/ sin(%itheta)))$
ur[1,2] : xdot*cos(%itheta)$
print("ur")$
ur;

print("Representation of the unit vector utheta as")$
print(sin(%itheta) * cos(phi)* %sigma[1] + sin(%itheta) * sin(phi)* %sigma[2] + cos(%itheta) * %sigma[3])$
utheta : cos(%itheta) * cos(phi)* %sigma[1] + cos(%itheta) * sin(phi)* %sigma[2] - sin(%itheta) * %sigma[3]$
utheta : factor(utheta);
xdot : rationp(exponentialize((utheta[2,1]/ cos(%itheta)))$
utheta[2,1] : xdot*cos(%itheta)$
xdot : rationp(exponentialize((utheta[1,2]/ cos(%itheta)))$
utheta[1,2] : xdot*cos(%itheta)$
print("ur")$
utheta;

print("Representation of the unit vector \phi as")$
print(- sin(phi)* %sigma[1] + cos(phi)* %sigma[2])$
uphi : - sin(phi)* %sigma[1] + cos(phi)* %sigma[2]$
uphi : rationp(exponentialize(uphi));

print("  ***************************************************************")$
print(" Pauli Matrices Definition Spherical")$
print("  ***************************************************************")$

ct : cos(%itheta)$
st : sin(%itheta)$

%sigma[r] : matrix([ct,st * %e^(-%i * phi)],[st * %e^(%i * phi),-ct])$
%sigma[%itheta] : matrix([-st,ct  * %e^(-%i * phi)],[ct * %e^(%i * phi),st])$
%sigma[phi] : matrix([0,-%i * %e^(-%i * phi)],[%i * %e^(%i * phi),0])$

print("%sigma[r] = ",%sigma[r])$
print("%sigma[%itheta] = ",%sigma[%itheta])$
print("%sigma[phi] = ",%sigma[phi])$

print("  ***************************************************************")$
print("vector in spherical coordinates")$
print("  ***************************************************************")$
Aspk :  ratsimp(factor(A[r]*%sigma[r]+A[%itheta]*%sigma[%itheta]+A[ phi]*%sigma[phi]));

print("  ***************************************************************")$
print(" properties")$
print("  ***************************************************************")$

print("Squared matrices")$
trigsimp(%sigma[r] . %sigma[r]);
trigsimp(%sigma[%itheta] . %sigma[%itheta]);
%sigma[phi] . %sigma[phi];

print("  ***************************************************************")$
print(" Alternating rules")$
trigsimp(%sigma[r] . %sigma[%itheta] - %i * %sigma[phi]);
%sigma[%itheta] . %sigma[phi] - %i * %sigma[r];
%sigma[phi] . %sigma[r] - %i * %sigma[%itheta];
print("%sigma[r] . %sigma[%itheta] + %i * %sigma[phi]")$
print("%sigma[%itheta] . %sigma[phi] = %i * %sigma[r]")$
print("%sigma[phi] . %sigma[r] = %i * %sigma[%itheta]")$

print("  ***************************************************************")$
print("Determinants")$
trigsimp(determinant(%sigma[r]));
trigsimp(determinant(%sigma[%itheta]));
determinant(%sigma[phi]);

/*
print("  ***************************************************************")$
print(" Transformation matrices")$
print("  ***************************************************************")$
%sigma[r] : %sigma[1]$
%sigma[%itheta] : %sigma[2]$
%sigma[phi] : %sigma[3]$
*/

print("  ***************************************************************")$
print(" vector decompositions")$
print("  ***************************************************************")$
print(" obtaining A vector components")$
print("  ***************************************************************")$
print(" from rectangular to spherical")$
print("  ***************************************************************")$
print(" original vector ",Axyz)$
print("Dot multiplication with %sigma[r] provides the r component")$
ur1 :A[x] * trigsimp(%sigma[r] . %sigma[1]) +
     A[y] * trigsimp(%sigma[r] . %sigma[2]) +
     A[z] * trigsimp(%sigma[r] . %sigma[3]) $
Ar : factor(realpart(1/2 * (ur1[1,1]+ur1[2,2])))$
print(" Ar, " = ",Ar)$

print("Dot multiplication with %sigma[%itheta] provides the %itheta component")$
ur1 :A[x] * trigsimp(%sigma[%itheta] . %sigma[1]) +
     A[y] * trigsimp(%sigma[%itheta] . %sigma[2]) +
     A[z] * trigsimp(%sigma[%itheta] . %sigma[3])$
Atheta : ratsimp(factor(realpart(1/2 * (ur1[1,1]+ur1[2,2]))))$
print(" Atheta, " = ",Atheta)$

print("Dot multiplication with %sigma[phi] provides the phi component")$
ur1 :A[x] * trigsimp(%sigma[phi] . %sigma[1]) +
     A[y] * trigsimp(%sigma[phi] . %sigma[2]) +
     A[z] * trigsimp(%sigma[phi] . %sigma[3])$
Aphi : ratsimp(factor(realpart(1/2 * (ur1[1,1]-ur1[2,2]))))$
print(" Aphi, " = ",Aphi)$

print("  ***************************************************************")$
print("From spherical to rectangular")$
print("  ***************************************************************")$

print("Dot multiplication with %sigma[1] provides the x component")$
ur1 :A[r] * trigsimp(%sigma[r] . %sigma[1]) +
     A[%itheta] * trigsimp(%sigma[%itheta] . %sigma[1]) +
     A[phi] * trigsimp(%sigma[phi] . %sigma[1]) $
Ax : factor(realpart(1/2 * (ur1[1,1]+ur1[2,2])))$
print(" Ax, " = ",Ax)$

print("Dot multiplication with %sigma[2] provides the y component")$
ur1 :A[r] * trigsimp(%sigma[r] . %sigma[2]) +
     A[%itheta] * trigsimp(%sigma[%itheta] . %sigma[2]) +
     A[phi] * trigsimp(%sigma[phi] . %sigma[2]) $
Ay : factor(realpart(1/2 * (ur1[1,1]-ur1[2,2])))$
print(" Ay, " = ",Ay)$

print("Dot multiplication with %sigma[3] provides the z component")$
ur1 :A[r] * trigsimp(%sigma[r] . %sigma[3]) +
     A[%itheta] * trigsimp(%sigma[%itheta] . %sigma[3]) +
     A[phi] * trigsimp(%sigma[phi] . %sigma[3]) $
Az : factor(realpart(1/2 * (ur1[1,1]+ur1[2,2])))$
print(" Az, " = ",Az)$

print("end")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Pauli matrices in Spherical coordinates

An example of application is provided with the following listing.

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect)$
/* example3.8.wxm */

/* */
print("Example of of transformation from rectangular to spherical")$
print("given the vector")$
print(Ax," = ",-x+y)$
print(Ay," = ",-x-y)$
print(Az," = ",z)$
print("Find the vector in spherical coordinates")$

print("---------------------------------")$
print("Solution")$
print("---------------------------------")$
print("Express x,y,z in terms of r, theta ,phi")$
x : r * sin(theta) * cos(phi)$
y : r * sin(theta) * sin(phi)$
z : r *cos(theta)$

print("spherical to rectangular")$
print("x," = ",x)$
print("y," = ",y)$
print("z," = ",z)$

print("write the vector A in rectangular coordinates")$
Ax : trigsimp(x+y) ;
Ay : trigsimp(y-x);
Az : trigsimp(z);

print("Transformation matrix x2rmat")$
r2xmat : matrix([sin(theta)* cos(phi),sin(theta)*sin(phi),cos(theta)],
                [cos(theta)*cos(phi),cos(theta)*sin(phi),-sin(theta)],
                [-sin(phi),cos(phi),0]);

print("vector A in rectangular coordinates")$
Arect  : [Ax,Ay,Az];

print("Transform vector A in spherical coordinates")$
print("rectangular to spherical")$
Aspher : r2xmat . Arect ;
print("after xemplification we recover")$
Aspher : trigsimp(Aspher);

print("-----------------------------------------")$
print(" same excercise but with Pauli matrices")$
print("-----------------------------------------")$

print("------------------------------------------------------------------")$
print("Pauli Matrices Definition")$
print("------------------------------------------------------------------")$

%sigma[0] : matrix([1,0],[0,1])$
%sigma[1] : matrix([0,1],[1,0])$
%sigma[2] : matrix([0,-%i],[%i,0])$
%sigma[3] : matrix([1,0],[0,-1])$

print("%sigma[0]," = ",%sigma[0])$
print("%sigma[1]," = ",%sigma[1])$
print("%sigma[2]," = ",%sigma[2])$
print("%sigma[3]," = ",%sigma[3])$

print("Vector A in rectangular coordinates")$
Axyz : Ax*%sigma[1] + Ay*%sigma[2] + Az*%sigma[3];


print("------------------------------------------------------------------")$
print("Pauli Matrices Definition Spherical")$
print("------------------------------------------------------------------")$

ct : cos(theta)$
st : sin(theta)$

%sigma[r] : matrix([ct,st * %e^(-%i * phi)],[st * %e^(%i * phi),-ct])$
%sigma[theta] : matrix([-st,ct  * %e^(-%i * phi)],[ct * %e^(%i * phi),st])$
%sigma[phi] : matrix([0,-%i * %e^(-%i * phi)],[%i * %e^(%i * phi),0])$

print("%sigma[r]," = ",%sigma[r])$
print("%sigma[theta]," = ",%sigma[theta])$
print("%sigma[phi]," = ",%sigma[phi])$

A[x] : Ax$
A[y] : Ay$
A[z] : Az$

print("Dot multiplication with %sigma[r] allows to retrieve the r component")$
sr1 :A[x] * trigsimp(%sigma[r] . %sigma[1]) + A[y] * trigsimp(%sigma[r] . %sigma[2]) + A[z] * trigsimp(%sigma[r] . %sigma[3])
$
sr1 : trigsimp(sr1)$
Ar : factor(realpart(1/2 * (sr1[1,1]+sr1[2,2])))$
Ar : trigsimp(Ar)$
print("Ar," = ",Ar)$


print("Dot multiplication with %sigma[theta] allows to retrieve the theta component")$
sr1 :A[x] * trigsimp(%sigma[theta] . %sigma[1]) + A[y] * trigsimp(%sigma[theta] . %sigma[2]) + A[z] * trigsimp(%sigma[theta] . %sigma[3])
$
Atheta : ratsimp(factor(realpart(1/2 * (sr1[1,1]+sr1[2,2]))))$
Atheta : trigsimp(Atheta)$
print("Atheta," = ",Atheta)$


print("Dot multiplication with %sigma[phi] allows to retrieve the phi component")$
sr1 :A[x] * trigsimp(%sigma[phi] . %sigma[1]) + A[y] * trigsimp(%sigma[phi] . %sigma[2]) + A[z] * trigsimp(%sigma[phi] . %sigma[3])
$
Aphi : ratsimp(factor(realpart(1/2 * (sr1[1,1]+sr1[2,2]))))$
Aphi : trigsimp(Aphi)$
print("Aphi," = ",Aphi)$

print("------------------------------------------------------------------")$
print("vector in spherical coordinates")$
print("------------------------------------------------------------------")$
Asph : ratsimp(factor(Ar*%sigma[r]+Atheta*%sigma[theta]+Aphi*%sigma[phi]));

print("Try to find the difference with the cartesian representation")$
ratsimp(demoivre(Asph - Axyz));
print("There is no difference!")$
print("The vector representation with Pauli is independent from the coordinate system")$

print("bye")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```