Parametric Formulas for Villarceau Circles

Kurt Nalty

December 31, 2012

Abstract

At every point on a torus, four perfect circles intersect. Two of these circles are the toroidal and poloidal circles commonly used for coordinates on a torus. The other two circles are the Villarceau circles, created by slicing the torus at an angle bitangent to the interior opening of the torus. These circles can be used as an alternative coordinate system for the torus. These circles are also of technological interest for high frequency, resonant, air core transformers.

Villarceau Circles

At every point on a torus, four perfect circles intersect. Two of these circles are the toroidal and poloidal circles commonly used for coordinates on a torus. The other two circles are the Villarceau circles, which are circles of the same radius as the major radius R of the torus, tilted to the center plane by a slope of $\pm r/R$ and offset from the center by minor radius distance r.

From David Burke and Wikipedia, we have a simple illustration of standard coordinates on a torus shown in Figure 1. These coordinates are often called Tokamak coordinates, as the phrase "toroidal coordinates" usually refers to a different sphere and ring coordinate system.

Many fine illustrations of Villarceau circles exist. Figure 2, by Lucas Barbosa, shows the circles in red on a torus.

Lucas Barbosa has a nice animation on Wikipedia showing a slicing process, and the resulting circles. Figure 3 is a cross-section taken from his animation.

My favorite is a POV contest award winner by Tor Olav Kristensen.

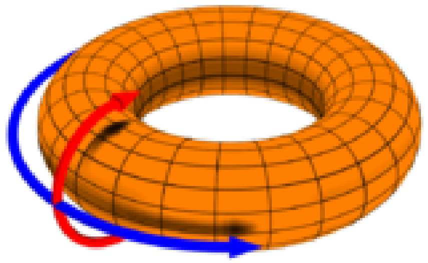

Figure 1: (David Burke - Wikipedia) Toroidal Direction Blue, Poloidal Direction Red

My goal in this note, is to provide simple, explicit formula correlating Cartesian, toroidal and Villarceau formulas.

My primary coordinate system will be Tokamak coordinates, θ and ϕ, as illustrated in Figure 1.

Orient the toriod in the XY plane, with θ beginning on the X axis and increasing in a counter-clockwise direction as seen from the positive Z axis. Let R represent the major axis, r the minor axis, and ρ be the distance of the point from the Z axis. Define the poloidal angle ϕ as zero at the maximum radial distance, and increasing initially in the positive Z direction.

We find the cartesian coordinates to be

$$\begin{aligned} \rho &= R + r\cos(\phi) \\ x &= \rho\cos(\theta) \\ y &= \rho\sin(\theta) \\ z &= \sin(\phi) \end{aligned}$$

Working backwards from x, y, and z, we have

$$\begin{aligned} \theta &= \text{atan2}(y, x) \\ \phi &= \text{atan2}(z, \sqrt{x^2+y^2} - R) \end{aligned}$$

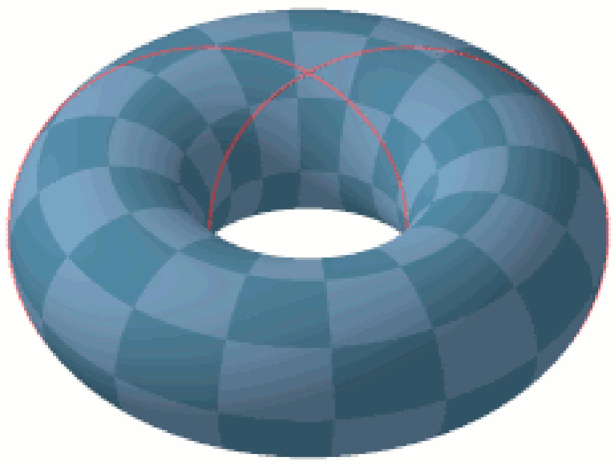

Figure 2: (Lucas Barbosa - Wikipedia) Villarceau Circle On Torus

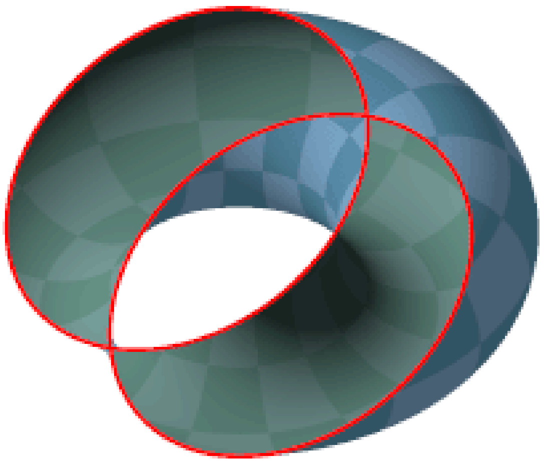

Figure 3: (Lucas Barbosa - Wikipedia) Sliced Torus

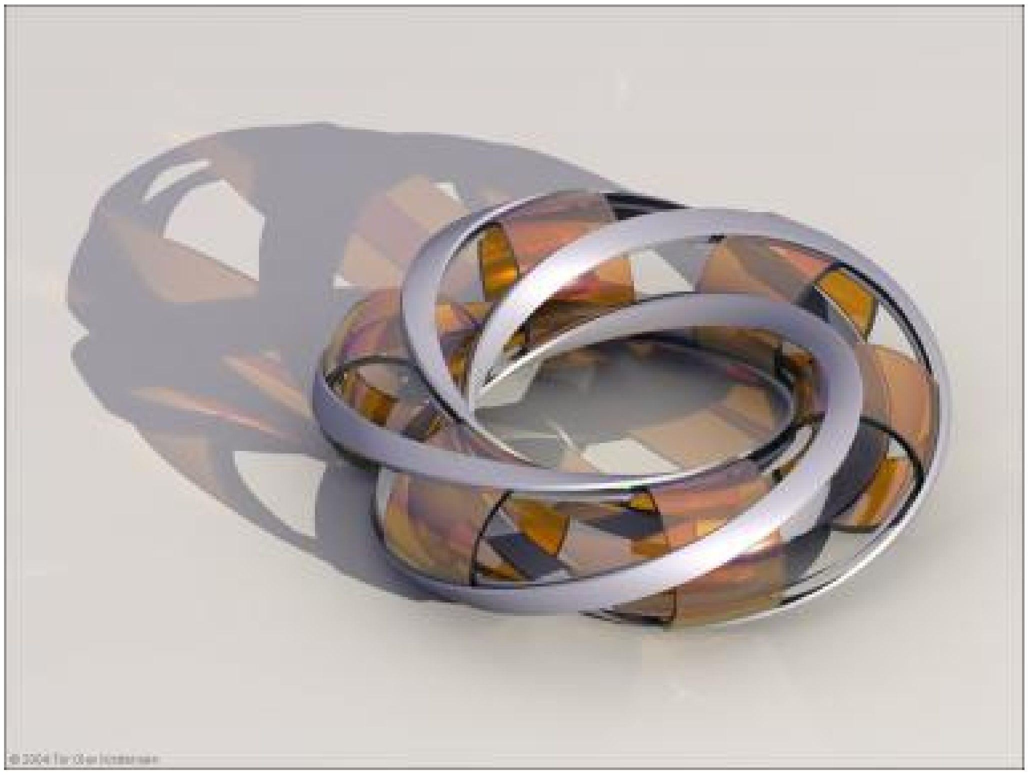

Figure 4: One Family of Villarceau Circles by Tor Olav Kristensen

First Villarceau Family Coordinates

The Villarceau circles are of radius R, offset from the origin by r, and tilted by angle $\pm\sin^{-1}(r/R)$. This tilted circle is then rotated around the Z axis by angle ψ to sweep the family of curves associated with that particular tilt sign. Any spot on the torus can be specified by an angle along the circle γ(which turns out to be ϕ), and the angle ψ. In a sense, I have three coordinates along the surface of the torus, being tokamak coordinates, and 'right hand' and 'left hand' Villarceu circles.

As mentioned above, the poloidal angle ϕ from tokamak coordinates and the angle along the tilted circle γ can be chosen to coincide, resulting in the z coordinate being

$$\begin{aligned}
\alpha &= \sin^{-1}(r/R) \quad \text{tilt angle} \\
z &= r\sin(\phi) \\
&= R\sin(\gamma)\sin(\alpha) = R\sin(\gamma)(r/R) = r\sin(\gamma) \\
\gamma &= \phi
\end{aligned}$$

The inverse sine has two solutions over 0 to 180 degrees. In this case, we have chosen the first solution. For the second family, we will find we need the second solution.

For the x and y coordinates, I usually obtain their value in two steps. I first find coordinates on the simple tilted circle, then rotate those coordinates to obtain the actual values. For the first family, I offset the circle by r in the positive x direction.

$$\begin{aligned}
\alpha &= \sin^{-1}(r/R) \quad \text{tilt angle} \\
x_1 &= r + R\cos(\gamma) \\
y_1 &= R\sin(\gamma)\cos(\alpha) \\
z_1 &= R\sin(\gamma)\sin(\alpha) = r\sin(\gamma) \\
\\
x_2 &= x = x_1\cos(\psi) - y_1\sin(\psi) \\
y_2 &= y = x_1\sin(\psi) + y_1\cos(\psi) \\
z_2 &= z = r\sin(\gamma)
\end{aligned}$$

Now, given x, y and z, how do we recover γ and ψ? It turns out to be fairly easy. As γ and ϕ coincide, we have

$$\gamma = \operatorname{atan2}(z, \sqrt{x^2 + y^2} - R)$$

Knowing γ, R and r, we can calculate x_1 and y_1.

$$\begin{aligned} x_1 &= r + R\cos(\gamma) \\ y_1 &= R\sin(\gamma)\cos(\alpha) \end{aligned}$$

Knowing x_1, y_1, and given $x_2 = x$, and $y_2 = y$, we can then solve the equations

$$\begin{aligned} x_2 &= x_1\cos(\psi) - y_1\sin(\psi) \\ y_2 &= x_1\sin(\psi) + y_1\cos(\psi) \end{aligned}$$

for $\cos(\phi)$ and $\sin(\phi)$. Equivalently, using our expressions for cross product and dot product of two vectors, we achieve the same result,

$$\phi = \operatorname{atan2}(x_1y_2 - x_2y_1, x_1x_2 + y_1y_2)$$

Second Villarceau Family Coordinates

The second Villarceau family is found fairly similar to the first. The initial circle is now offset in the negative x direction.

$$x_1 = -r + R\cos(\gamma)$$

The inverse sine has two solutions over the span 0 to 180 degrees. This time, we use the second solution relating γ and ϕ.

$$\begin{aligned}
\alpha &= \sin^{-1}(r/R) \quad \text{tilt angle} \\
z &= r\sin(\phi) \\
&= R\sin(\gamma)\sin(\alpha) = R\sin(\gamma)(r/R) = r\sin(\gamma) \\
\gamma &= \pi/2 - \phi
\end{aligned}$$

For the x and y coordinates, I usually obtain their value in two steps. I first find coordinates on the simple tilted circle, then rotate those coordinates to obtain the actual values. For the first family, I offset the circle by r in the positive x direction.

$$\begin{aligned}
\alpha &= \sin^{-1}(r/R) \quad \text{tilt angle} \\
x_1 &= -r + R\cos(\gamma) \\
y_1 &= R\sin(\gamma)\cos(\alpha) \\
z_1 &= R\sin(\gamma)\sin(\alpha) = r\sin(\gamma) \\
\\
x_2 &= x = x_1\cos(\psi) - y_1\sin(\psi) \\
y_2 &= y = x_1\sin(\psi) + y_1\cos(\psi) \\
z_2 &= z = r\sin(\gamma)
\end{aligned}$$

Now, given x, y and z, how do we recover γ and ψ? It turns out to be fairly easy. As γ and ϕ coincide, we have

$$\gamma = \text{atan2}(z, \sqrt{x^2+y^2} - R)$$

Knowing γ, R and r, we can calculate x_1 and y_1.

$$\begin{aligned}
x_1 &= r + R\cos(\gamma) \\
y_1 &= R\sin(\gamma)\cos(\alpha)
\end{aligned}$$

Knowing x_1, y_1, and given $x_2 = x$, and $y_2 = y$, we can then solve the equations

$$\begin{aligned}
x_2 &= x_1\cos(\psi) - y_1\sin(\psi) \\
y_2 &= x_1\sin(\psi) + y_1\cos(\psi)
\end{aligned}$$

for $\cos(\phi)$ and $\sin(\phi)$. Equivalently, using our expressions for cross product and dot product of two vectors, we achieve the same result,

$$\phi = \operatorname{atan2}(x_1 y_2 - x_2 y_1, x_1 x_2 + y_1 y_2)$$

Listing of Villarceau.c

As a numerical illustration of the above, here is a (mangled) listing of a demonstration program in c, available at
`http://www.kurtnalty.com/Villarceau.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


int main(void)
{
int i,j,k;
double x,y,z,x1,y1,z1,x2,y2,z2;
int phid, thetad, gammad, psid; // integer degree angles
double phi, theta, gamma, psi;
double pi = 3.1415926;
double tilt, alpha;
double gamma1, gamma2;


double relative_error_squared; // actually, relative_error_squared
double r2;

double R = 5.0;
double r = 1.0;

double rho;
float a;
double xref, yref, zref;
```

```
FILE* Output;

Output = fopen("toroid.xyz","w");
r = 1.0;  // minor radius
R = 5.0; // major radius

// draw 24 minor radial rings

for (i=0;i<24;i++) {
theta = (i*pi)/(12.0);
printf("i = %d Theta = %g \n",i,theta);
phi = 0.0;
z = r*sin(phi);
rho = R + r*cos(phi);
x = rho*cos(theta);
y = rho*sin(theta);
fprintf(Output,"%g   %g   %g  0 \n",x,y,z);  // moveto

for (j=1;j<25;j++) {
phi = (j*pi/12.0);
z = r*sin(phi);
rho = R + r*cos(phi);
x = rho*cos(theta);
y = rho*sin(theta);
fprintf(Output,"%g   %g   %g  30 \n",x,y,z);  // moveto

}

}


// draw 24 major radial rings

for (i=0;i<24;i++) {
phi = (i*pi)/(12.0);
printf("i = %d Theta = %g \n",i,theta);
theta = 0.0;
z = r*sin(phi);
rho = R + r*cos(phi);
x = rho*cos(theta);
y = rho*sin(theta);
fprintf(Output,"%g   %g   %g  0 \n",x,y,z);  // moveto

for (j=1;j<25;j++) {
theta = (j*pi/12.0);
z = r*sin(phi);
rho = R + r*cos(phi);
x = rho*cos(theta);
y = rho*sin(theta);
fprintf(Output,"%g   %g   %g  90 \n",x,y,z);  // moveto

}

}




phid = 30.0;
thetad = 0.0;
printf("\nPlease enter phi (degrees) ");
scanf("%d",&phid);

printf("\nPlease enter theta (degrees) ");
scanf("%d",&thetad);
phi = phid*pi/180.0;
theta = thetad*pi/180.0;

// knowing R, r , theta, and phi, we calculate x, y, and z

z = r*sin(phi);
rho = R + r*cos(phi); //distance from z axis
x = rho*cos(theta);
y = rho*sin(theta);
```

```
xref = x;
yref = y;
zref = z;

printf("x = %g   y = %g   z = %g \n",x,y,z);

// given x, y, z calculate phi and theta

theta = atan2(y,x);
phi   = atan2(z,(sqrt(x*x + y*y) - R));

printf("recalculated from x, y, z, we have (degree)\nphi = %g    theta  =  %g \n\r


// *************** plot the hoop and ring for phi and theta ****************

double phiref = phi;
double thetaref = theta;

phi = 0.0;
theta = thetaref;
z = r*sin(phi);
rho = R + r*cos(phi);
x = rho*cos(theta);
y = rho*sin(theta);
fprintf(Output,"%g   %g   %g  0 \n",x,y,z);  // moveto

for (j=1;j<25;j++) {
phi = (j*pi/12.0);
z = r*sin(phi);
rho = R + r*cos(phi);
x = rho*cos(theta);
y = rho*sin(theta);
fprintf(Output,"%g   %g   %g  160 \n",x,y,z);  // moveto

}

phi = phiref;
```

```
theta = 0.0;
z = r*sin(phi);
rho = R + r*cos(phi);
x = rho*cos(theta);
y = rho*sin(theta);
fprintf(Output,"%g   %g   %g  0 \n",x,y,z);  // moveto

for (j=1;j<25;j++) {
theta = (j*pi/12.0);
z = r*sin(phi);
rho = R + r*cos(phi);
x = rho*cos(theta);
y = rho*sin(theta);
fprintf(Output,"%g   %g   %g  160 \n",x,y,z);  // moveto

}



// **************** plot first Villarceau circle in purple *********************

printf("\nFirst Villarceau circle is purple\n");

tilt = r/R;
alpha = asin(tilt);
phi = phid*pi/180.0;
theta = thetad*pi/180.0;
gamma = phi; // easy

x2 = xref; // these are the called out point
y2 = yref;
z2 = zref;

x1 = r + R*cos(gamma); // this is the vector prior to rotation
y1 =     R*sin(gamma)*cos(alpha);
z1 = r*sin(gamma);

psi = atan2(x1*y2 - x2*y1, x1*x2 + y1*y2); // recover the angle
```

```
printf("Reference values x = %g  y = %g  z = %g  \n",xref,yref,zref);

x2 =  x1*cos(psi) - y1*sin(psi);
y2 =  x1*sin(psi) + y1*cos(psi);
z2 =  z1;
printf("Recovered values x = %g  y = %g  z = %g  \n",x2,y2,z2);

// set the initial point for plotting the circle

gamma = 0.0;
x1 = r + R*cos(gamma);
y1 =     R*sin(gamma)*cos(alpha);
z1 = r*sin(gamma);

// rotate circle by angle psi around the x axis

x2 =  x1*cos(psi) - y1*sin(psi);
y2 =  x1*sin(psi) + y1*cos(psi);
z2 =  z1;

fprintf(Output,"%g   %g   %g  0 \n",x2,y2,z2);  // moveto

for (j=1;j<25;j++) {
gamma = (j*pi/12.0);
x1 = r + R*cos(gamma);
y1 =     R*sin(gamma)*cos(alpha);
z1 = r*sin(gamma);

// rotate circle by angle psi around the x axis

x2 =  x1*cos(psi) - y1*sin(psi);
y2 =  x1*sin(psi) + y1*cos(psi);
z2 =  z1;
fprintf(Output,"%g   %g   %g  220 \n",x2,y2,z2);  // lineto

}
```

```
// ***************** Plot second Villarceau circle in orange ***************


// We now look at the other Villarceau curve through this point.
printf("\nThe second Villarceua circle is yelloish-orange\n");


tilt = r/R;
alpha = asin(tilt);
phi = phid*pi/180.0;
theta = thetad*pi/180.0;
gamma = pi - phi; // easy

x2 = xref; // these are the called out point
y2 = yref;
z2 = zref;

x1 = -r + R*cos(gamma); // this is the vector prior to rotation
y1 =      R*sin(gamma)*cos(alpha);
z1 =  r*sin(gamma);

psi = atan2(x1*y2 - x2*y1, x1*x2 + y1*y2); // recover the angle

printf("Reference values x = %g  y = %g  z = %g  \n",xref,yref,zref);

x2 =  x1*cos(psi) - y1*sin(psi);
y2 =  x1*sin(psi) + y1*cos(psi);
z2 =  z1;
printf("Recovered values x = %g  y = %g  z = %g  \n",x2,y2,z2);

// set the initial point for plotting the circle

gamma = 0.0;
x1 = -r + R*cos(gamma);
y1 =      R*sin(gamma)*cos(alpha);
z1 =  r*sin(gamma);
```

```
// rotate circle by angle psi around the x axis

x2 =  x1*cos(psi) - y1*sin(psi);
y2 =  x1*sin(psi) + y1*cos(psi);
z2 =  z1;

fprintf(Output,"%g   %g   %g  0 \n",x2,y2,z2);  // moveto

for (j=1;j<25;j++) {
gamma = (j*pi/12.0);
x1 = -r + R*cos(gamma);
y1 =      R*sin(gamma)*cos(alpha);
z1 =  r*sin(gamma);

// rotate circle by angle psi around the x axis

x2 =  x1*cos(psi) - y1*sin(psi);
y2 =  x1*sin(psi) + y1*cos(psi);
z2 =  z1;
fprintf(Output,"%g   %g   %g  310 \n",x2,y2,z2);  // lineto

}

fclose(Output);

}
```