```
/* p is the position of the sensors 6x3 */
/* 0 =X1; 1=X2; 2=Y1; 3=Y2; 4=Z1; 5=z2; */

/* R is the radius */

/* vector r = p - s */
float64 r[6][4];

// resulting matrix S_ij
float64 S[3][3];
float64 ROT[3][3];
int c;

/* calculate r */
/* 0=x; 1=y; 2=z; 3=abs^5; */

for( c=0; c<6; c++)
{
   r[c][0] = p[c][0] - R*cos( phi);
   r[c][1] = p[c][1] - R*sin( phi);
   r[c][2] = p[c][2];
   r[c][3] = (r[c][0]**2+r[c][1]**2+r[c][2]**2)**(5/2);
}


/* MATRIX S_ij */
S[0][0] = (2*r[0][0]**2-r[0][1]**2-r[0][2]**2)/r[0][3]-(2*r[1][0]**2-r[1][1]**2-r[1][2]**2)/r[1][3];
S[0][1] = 3*(r[0][0]*r[0][1]/r[0][3]-r[1][0]*r[1][1]/r[1][3]);
S[0][2] = 3*(r[0][0]*r[0][2]/r[0][3]-r[1][0]*r[1][2]/r[1][3]);

S[1][0] = 3*(r[2][0]*r[2][1]/r[2][3]-r[3][0]*r[3][1]/r[3][3]);
S[1][1] = (2*r[2][1]**2-r[2][0]**2-r[2][2]**2)/r[2][3]-(2*r[3][1]**2-r[3][0]**2-r[3][2]**2)/r[3][3];
S[1][2] = 3*(r[2][1]*r[2][2]/r[2][3]-r[3][1]*r[3][2]/r[3][3]);

S[2][0] = 3*(r[4][0]*r[4][1]/r[4][3]-r[5][0]*r[5][1]/r[5][3]);
S[2][1] = (2*r[4][1]**2-r[4][0]**2-r[4][2]**2)/r[4][3]-(2*r[5][1]**2-r[5][0]**2-r[5][2]**2)/r[5][3];
S[2][2] = 3*(r[4][1]*r[4][2]/r[4][3]-r[5][1]*r[5][2]/r[5][3]);

/* Rotation Matrix */
ROT[0][0] = cos( phi);
ROT[0][1] = sin(phi);
ROT[0][2] = 0;

ROT[1][0] = ROT[0][1];
ROT[1][1] = ROT[0][0];
ROT[1][2] = 0;

ROT[2][0] = 0;
ROT[2][1] = 0;
ROT[2][2] = 1;
```