The following Matlab and R functions will compute Cartesian coordinates from a distance matrix of chromatic perceptual distances calculated using Vorobyev & Osorio's (1998) model.

Matlab:

```
function coords = jnd2cart(d,nc)
%COORDS = JND2CART(D,NC)
%Calculates the Cartesian coordinates COORDS, given a symmetrical distance
%matrix of chromatic perceptual distances (in jnds), D, and the number of
%cones classes used in the chromatic discrimination, NC.

%check input
n = length(d); %number of data points
if n < 2
    error('The chromatic distance matrix D must measure at least 2x2.');
end

%get chromatic coordinates
if nc == 3 %trichromat
    coords = zeros(n,2);
    %coordinates of the first three points
    coords(1,:) = [0 0];
    coords(2,:) = [d(1,2) 0];
    if n >= 3
        coords(3,:) = intersection(d(1,2),d(1,3),d(2,3));
        if n >=4
            %coordinates of the fourth and subsequent points
            for i = 4:n
                [p1 p2] = intersection(d(1,2),d(1,i),d(2,i));
                d1 = abs(norm(p1 - coords(3,:)) - d(3,i));
                d2 = abs(norm(p2 - coords(3,:)) - d(3,i));
                if d1 < d2
                    coords(i,:) = p1;
                else
                    coords(i,:) = p2;
                end
            end
        end
    end
elseif nc == 4 %tetrachromat
    coords = zeros(n,3);
    %coordinates of the first three points (assumes all on xy plane)
    coords(1,:) = [0 0 0];
    coords(2,:) = [d(1,2) 0 0];
    if n >= 3
        coords(3,:) = [intersection(d(1,2),d(1,3),d(2,3)) 0];
        if n >= 4
            %coordinates of fourth point
            coords(4,:) = intersection3(d(1,2),coords(3,:),d(1,4), ...
                                        d(2,4),d(3,4));
            if n >= 5
                %coordinates of the fifth and subsequent points
                for i = 5:n
                    [p1 p2] = intersection3(d(1,2),coords(3,:),d(1,i), ...
                                            d(2,i),d(3,i));
                    if abs(norm(p1 - coords(4,:)) - d(4,i)) < ...
                       abs(norm(p2 - coords(4,:)) - d(4,i))
                        coords(i,:) = p1;
                    else
                        coords(i,:) = p2;
                    end
                end
            end
```

```
        end
    end
else
    error('NC must equal 3 (for a trichromat) or 4 (for a tetrachromat).');
end
```

This helper function calculates the intersection points of two circles:

```
function [p1 p2] = intersection(d,r1,r2)
%[P1 P2] = INTERSECTION(D,R1,R2)
%Find the intersection points between two circles. Assumes that the centres
%of the two source circles lie at [0,0] and [D,0], and that they have radii
%of R1 and R2, respectively. Returns the two possible solution vectors, P1
%and P2.

x = (r1^2 - r2^2 + d^2)/(2*d);
y = sqrt(r1^2 - x^2);
p1 = real([x y]);
p2 = real([x -y]);
```

This helper function calculates the intersection points of two spheres:

```
function [p1 p2] = intersection3(d,ij,r1,r2,r3)
%[P1 P2] = INTERSECTION3(D,IJ,R1,R2,R3)
%Calculate sphere-sphere intersection points. Assumes that the centres of
%the three source spheres lie on the xy plane, with centres at [0,0,0],
%[D,0,0] and [IJ(1),IJ(2),0], and the spheres have radii of R1, R2 and R3,
%respectively. Returns the two possible solution vectors, P1 and P2.

i = ij(1); j = ij(2);
x = (r1^2 - r2^2 + d^2)/(2*d);
y = ((r1^2 - r3^2 + j^2 + i^2)/(2*j)) - ((i/j)*x);
z = sqrt(r1^2 - x^2 - y^2);
p1 = real([x y z]);
p2 = real([x y -z]);
```

R:

```
jnd2cart <- function(d,nc)
#Usage: jnd2cart(d,nc)
#Calculates the Cartesian coordinates given a symmetrical distance
#matrix of chromatic perceptual distances (in jnds), d, and the number of
#cones classes used in the chromatic discrimination, nc.
{

#check input
n <- ncol(d); #number of data points
if (n < 2)
{
    stop("The chromatic distance matrix d must measure at least 2x2.");
}

#get chromatic coordinates
if (nc == 3) #trichromat
{
    coords <- array(0,c(n,2));
    #coordinates of the first three points
    coords[1,] <- c(0,0);
    coords[2,] <- c(d[1,2],0);
```

```
    if (n >= 3)
    {
        p <- intersection(d[1,2],d[1,3],d[2,3]);
        coords[3,] <- p[1,];
        if (n >=4)
        {
            #coordinates of the fourth and subsequent points
            for (i in 4:n)
            {
                p <- intersection(d[1,2],d[1,i],d[2,i]);
                d1 <- abs(vdist(p[1,],coords[3,]) - d[3,i]);
                d2 <- abs(vdist(p[2,],coords[3,]) - d[3,i]);
                if (d1 < d2)
                {
                    coords[i,] <- p[1,];
                }
                else
                {
                    coords[i,] <- p[2,];
                }
            }
        }
    }
} else
if (nc == 4) #tetrachromat
{
    coords <- array(0,c(n,3));
    #coordinates of the first three points (assumes all on xy plane)
    coords[1,] <- c(0,0,0);
    coords[2,] <- c(d[1,2],0,0);
    if (n >= 3)
    {
        p <- intersection(d[1,2],d[1,3],d[2,3]);
        coords[3,] <- c(p[1,],0);
        if (n >= 4)
        {
            #coordinates of fourth point
            p <- intersection3(d[1,2],coords[3,],d[1,4],d[2,4],d[3,4]);
            coords[4,] <- p[1,];
            if (n >= 5)
            {
                #coordinates of the fifth and subsequent points
                for (i in 5:n)
                {
                    p <- intersection3(d[1,2],coords[3,],d[1,i], +
                                       d[2,i],d[3,i]);
                    diff1 <- abs(vdist(p[1,],coords[4,]) - d[4,i]);
                    diff2 <- abs(vdist(p[2,],coords[4,]) - d[4,i]);
                    if (diff1 < diff2)
                    {
                        coords[i,] <- p[1,];
                    }
                    else
                    {
                        coords[i,] <- p[2,];
                    }
                }
            }
        }
    }
} else
if (nc < 3 || nc > 4) #other
{
    stop("nc must equal 3 (for a trichromat) or 4 (for a tetrachromat).");
```

```
}
return (coords)

} #end function


intersection <- function(d,r1,r2)
#helper function for jnd2cart
{
    x <- (r1^2 - r2^2 + d^2)/(2*d);
    y <- sqrt(r1^2 - x^2);
    p1 <- Re(c(x,y));
    p2 <- Re(c(x,-y));
    return (rbind(p1[1:2],p2[1:2]));
}


intersection3 <- function(d,ij,r1,r2,r3)
#helper function for jnd2cart
{
    i <- ij[1]; j <- ij[2];
    x <- (r1^2 - r2^2 + d^2)/(2*d);
    y <- ((r1^2 - r3^2 + j^2 + i^2)/(2*j)) - ((i/j)*x);
    z <- 0;
    val <- (r1^2 - x^2 - y^2);
    if (val >= 0)
    {
        z <- sqrt(val);
    }
    p1 <- Re(c(x,y,z));
    p2 <- Re(c(x,y,-z));
    return (rbind(p1[1:3],p2[1:3]));
}


vdist <- function(a,b)
#helper function for jnd2cart
{
    d <- 0;
    for (i in 1:length(a))
    {
        d <- d + (a[i] - b[i])^2;
    }
    return (sqrt(d));
}
```

References

Vorobyev M, Osorio D. 1998. Receptor noise as a determinant of colour thresholds. Proc R Soc B. 265:351-358.