Lines, Curves and Surfaces in 3D

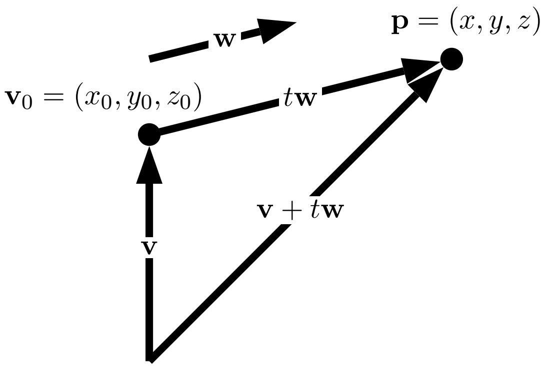
Lines in 3D

In 3D the **implicit equation** of a line is defined as the **intersection of two planes**. (More on this shortly)

The **parametric equation** is a simple extension to 3D of the 2D form:

$$\begin{aligned} x &= x_0 + ft \\ y &= y_0 + gt \\ z &= z_0 + ht \end{aligned}$$

Parametric Lines in 3D



$$
\begin{aligned}
x &= x_0 + ft \\
y &= y_0 + gt \\
z &= z_0 + ht
\end{aligned}
$$

This is simply an **extension** of the vector form in 3D

The line is normalised when $f^2 + g^2 + h^2 = 1$

Perpendicular Distance from a Point to a Line in 3D

For the parametric form,

$$
\begin{aligned}
x &= x_0 + ft \\
y &= y_0 + gt \\
z &= z_0 + ht
\end{aligned}
$$

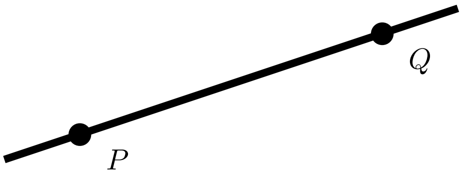
This builds on the 2D example we met earlier, line_par_point_dist_2d. The 3D form is line_par_point_dist_3d.

```
dx =   g * ( f * ( p(2) - y0 ) - g * ( p(1) - x0 ) ) ...
     + h * ( f * ( p(3) - z0 ) - h * ( p(1) - x0 ) );

dy =   h * ( g * ( p(3) - z0 ) - h * ( p(2) - y0 ) ) ...
     - f * ( f * ( p(2) - y0 ) - g * ( p(1) - x0 ) );

dz = - f * ( f * ( p(3) - z0 ) - h * ( p(1) - x0 ) ) ...
     - g * ( g * ( p(3) - z0 ) - h * ( p(2) - y0 ) );

dist = sqrt ( dx * dx + dy * dy + dz * dz ) ...
    / ( f * f + g * g + h * h );
```

The value of parameter, t, where the point intersects the line is given by:

```
t =  (f*( p(1) - x0) + g*(p(2) - y0) + h*(p(3) - z0)/( f * f + g * g + h*h);
```

Line Through Two Points in 3D (parametric form)



The parametric form of a line through two points, $P(x_p, y_p, z_p)$ and $Q(x_q, y_q, z_q)$ comes readily from the vector form of line (again a simple extension from 2D):

- Set base to point P
- Vector along line is $(x_q - x_p, y_q - y_p, z_q - z_p)$
- The equation of the line is:

$$
\begin{aligned}
x &= x_p + (x_q - x_p)t \\
y &= y_p + (y_q - y_p)t \\
z &= z_p + (z_q - z_p)t
\end{aligned}
$$

- As in 2D, $t = 0$ gives P and $t = 1$ gives Q
- Normalise if necessary.

Implicit Surfaces

An implicit surface (just like implicit curves in 2D) of the form

$$f(x, y, z) = 0$$

We simply add the **extra z dimension**.

For example:

- A plane can be represented

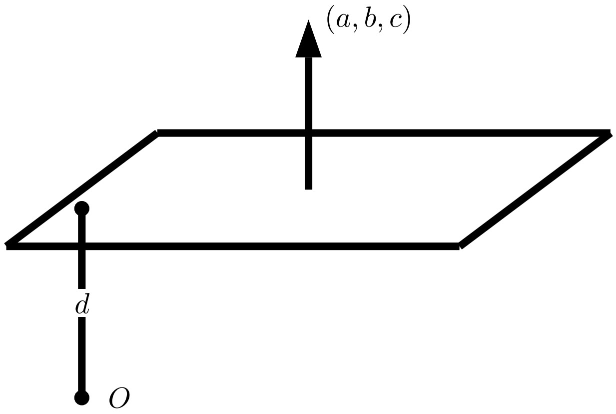
$$ax + by + cz + d = 0$$

- A sphere can be represented as

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - r^2 = 0$$

  which is just the extension of the circle in 2D to 3D where the centre is now (x_c, y_c, z_c) and the radius is r.

Implicit Equation of a Plane

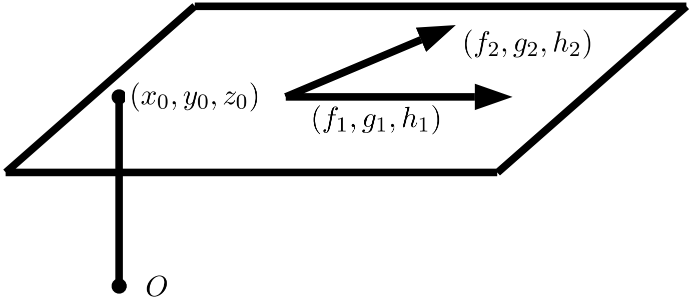


The plane equation:

$$ax + by + cz + d = 0$$

- Is normalised if $a^2 + b^2 + c^2 = 1$.
- Like 2D the **normal vector** — ***the surface normal*** — is given by a vector $\mathbf{n} = (a, b, c)$
  - a, b and c are the cosine angles which the normal makes with the x-,y- and z-axes respectively.

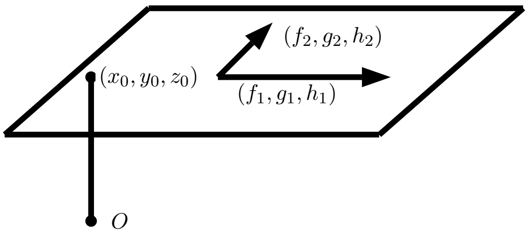
Parametric Equation of a Plane



$$
\begin{aligned}
x &= x_0 + f_1 u + f_2 v \\
y &= y_0 + g_1 u + g_2 v \\
z &= z_0 + h_1 u + h_2 v
\end{aligned}
$$

- This is an extension of parametric line into 3D where we now have two variable parameters u and v that vary.
- (f_1, g_1, h_1) and (f_2, g_2, h_2) are two **different** vectors **parallel** to the plane.

Parametric Equation of a Plane (Cont.)

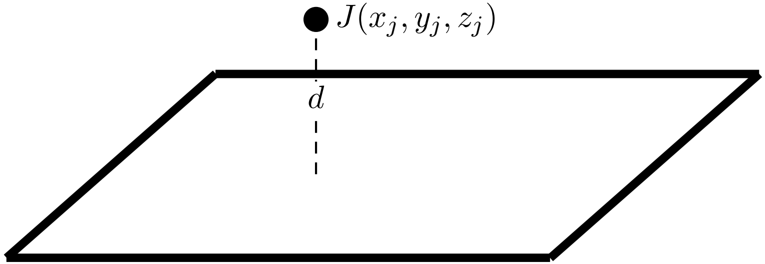


$$
\begin{aligned}
x &= x_0 + f_1 u + f_2 v \\
y &= y_0 + g_1 u + g_2 v \\
z &= z_0 + h_1 u + h_2 v
\end{aligned}
$$

- A point in the plane is found by adding proportion u of one vector to a proportion v of the other vector
- If the two vectors are have unit length and are perpendicular, then:

$$
\begin{aligned}
f_1^2 + g_1^2 + h_1^2 &= 1 \\
f_2^2 + g_2^2 + h_2^2 &= 1 \\
f_1 f_2 + g_1 g_2 + h_1 h_2 &= 0 \quad \text{(scalar product)}
\end{aligned}
$$

Distance from a 3D point and a Plane



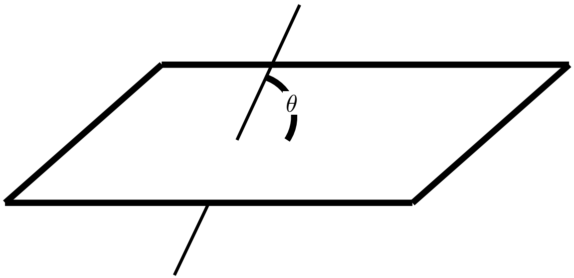
The distance, d, between a point, $J(x_j, y_j, z_j)$, and an **implicit** plane, $ax + by + cz + d = 0$ is:

$$d = \frac{ax_j + by_j + cz_j + d}{\sqrt{a^2 + b^2 + c^2}}$$

This is very similar the 2D distance of a point to a line.
The MATALB code to achive this is, plane_imp_point_dist_3d.m:

```
norm = sqrt ( a * a + b * b + c * c );
if ( norm == 0.0
      error ( 'PLANE Normal = 0!' );
end
dist = abs ( a * p(1) + b * p(2) + c * p(3) + d ) / norm;
```

Angle Between a Line and a Plane



If the plane is in implicit form $ax + by + cz + d = 0$ and line is in parametric form:

$$
\begin{aligned}
x &= x_0 + ft \\
y &= y_0 + gt \\
z &= z_0 + ht
\end{aligned}
$$

then the angle, γ between the line and the normal the plane (a, b, c) is:

$$\gamma = \cos^{-1}(af + bg + ch)$$

The angle, θ, between the line and the plane is then:

$$\theta = \frac{\pi}{2} - \gamma$$

Angle Between a Line and a Plane (cont)

If either line or plane equations are not normalised the we must normalise:

$$\gamma = \cos^{-1}\frac{(af+bg+ch)}{\sqrt{(a^2+b^2+c^2)}}, \gamma = \cos^{-1}\frac{(af+bg+ch)}{\sqrt{(f^2+g^2+h^2)}}, \gamma = \cos^{-1}\frac{(af+bg+ch)}{\sqrt{(a^2+b^2+c^2)(f^2+g^2+h^2)}}$$

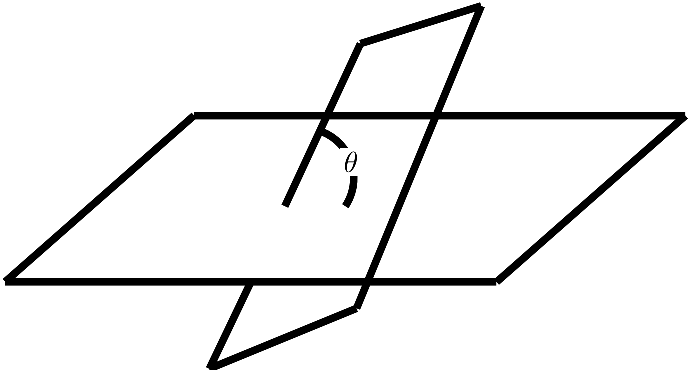
The angle, θ, is as before:

$$\theta = \frac{\pi}{2} - \gamma$$

The MATLAB code to so this is planes_imp_angle_line_3d.m:

```
norm1 = sqrt ( a1 * a1 + b1 * b1 + c1 * c1 );
 if ( norm1 == 0.0 )
    angle = Inf;
    return
 end

norm2 = sqrt ( f * f + g *g + h * h );
if ( norm2 == 0.0 )
    angle = Inf;
    return
  end

cosine = ( a1 * f + b1 * g + c1 * h) / ( norm1 * norm2 );
angle = pi/2 - acos( cosine );
```

Angle Between Two Planes



Given two **normalised** **implicit** planes

$$a_1x + b_1y + c_1z + d_1 = 0$$

and

$$a_2x + b_2y + c_2z + d_2 = 0$$

The angle between them, θ, is the angle between the normals:

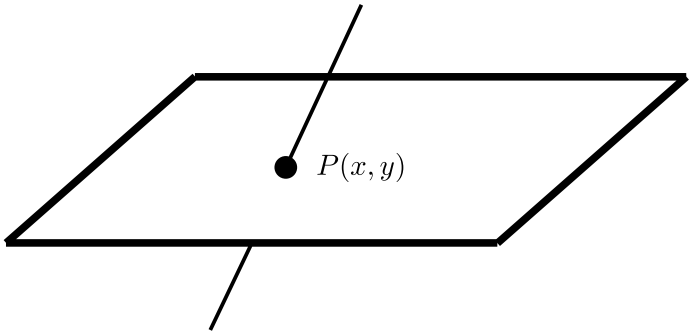
$$\theta = \cos^{-1}(a_1a_2 + b_1b_2 + c_1c_2)$$

Angle Between Two Planes (MATLAB Code)

The MATLAB code to do this is planes_imp_angle_3d.m:

```
norm1 = sqrt ( a1 * a1 + b1 * b1 + c1 * c1 );
if ( norm1 == 0.0 )
    angle = Inf;
    return
  end

norm2 = sqrt ( a2 * a2 + b2 * b2 + c2 * c2 );
if ( norm2 == 0.0 )
    angle = Inf;
    return
  end
cosine = ( a1 * a2 + b1 * b2 + c1 * c2 ) / ( norm1 * norm2 );
angle = acos ( cosine );
```

Intersection of a Line and Plane



If the plane is in implicit form $ax + by + cz + d = 0$ and line is in parametric form:

$$\begin{aligned} x &= x_0 + ft \\ y &= y_0 + gt \\ z &= z_0 + ht \end{aligned}$$

The the point, $P(x, y)$, where the is given by parameter, t:

$$t = \frac{-(ax_0 + by_0 + cz_0 + d)}{af + bg + ch}$$

Intersection of a Line and Plane (MATLAB Code)

The MATLAB code to do this is plane_imp_line_par_int_3d.m

```
tol = eps;

norm1 = sqrt ( a * a + b * b + c * c );
if ( norm1 == 0.0 )
  error ( 'Norm1 = 0 - Fatal error!' )
end

norm2 = sqrt ( f * f + g * g + h * h );
if ( norm2 == 0.0 )
  error ( 'Norm2 = 0 - Fatal error!' )
end

denom = a * f + b * g + c * h;

if ( abs ( denom ) < tol * norm1 * norm2 ) %  The line and the plane may be parallel.
  if ( a * x0 + b * y0 + c * z0 + d == 0.0 )
    intersect = 1;
    p(1) = x0;
    p(2) = y0;
    p(3) = z0;
  else
    intersect = 0;
    p(1:dim_num) = 0.0;
  end
else
  intersect = 1;
  t = - ( a * x0 + b * y0 + c * z0 + d ) / denom; %  they must intersect.
  p(1) = x0 + t * f;
  p(2) = y0 + t * g;
  p(3) = z0 + t * h;
end
```

Intersection of Three Planes

- Three planes intersect at point.
- Two planes intersect in a line $\rightarrow$ two lines intersect at a point
- Similar problem to solving in 2D for two line intersecting:
  - Solve *three* simultaneous linear equations:

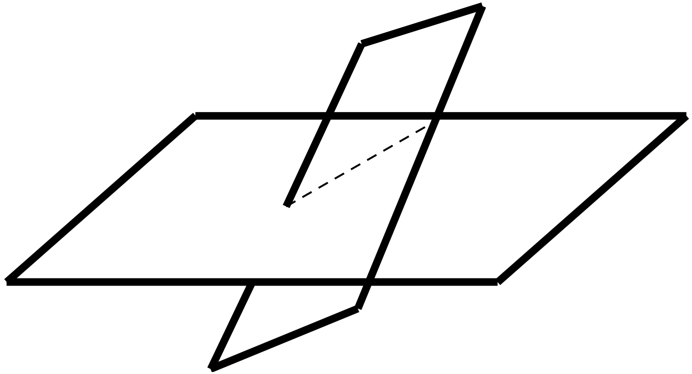
$$\begin{aligned}
a_1 x + b_1 y + c_1 z + d_1 &= 0 \\
a_2 x + b_2 y + c_2 z + d_2 &= 0 \\
a_3 x + b_3 y + c_3 z + d_3 &= 0
\end{aligned}$$

Intersection of Three Planes (MATLAB Code)

The MATLAB code to do this is, planes_3d_3_intersect.m:

```
tol = eps;

bc = b2*c3 - b3*c2;
ac = a2*c3 - a3*c2;
ab = a2*b3 - a3*b2;


det = a1*bc - b1*ac + c1*ab

if (abs(det) < tol)
  error('planes_3d_3_intersct: At least to planes are parallel');
end;

else
 dc = d2*c3 - d3*c2;
 db = d2*b3 - d3*b2;
 ad = a2*d3 - a3*d2;

 detinv = 1/det;

 p(1) = (b1*dc - d1*bc - c1*db)*detinv;
 p(2) = (d1*ac - a1*dc - c1*ad)*detinv;
 p(3) = (b1*ad + a1*db - d1*ab)*detinv;

 return;
end;
```

Intersection of Two Planes



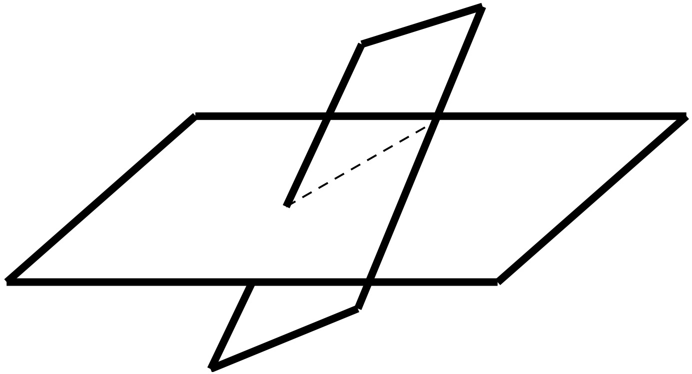
Two planes

$$
\begin{aligned}
a_1x + b_1y + c_1z + d_1 &= 0 \\
a_2x + b_2y + c_2z + d_2 &= 0
\end{aligned}
$$

intersect to form a straight line:

$$
\begin{aligned}
x &= x_0 + ft \\
y &= y_0 + gt \\
z &= z_0 + ht
\end{aligned}
$$

Intersection of Two Planes (Cont.)



- (f, g, h) may be found by finding a vector along the line. This is given by the vector cross product of (a_1, b_1, c_1) and (a_2, b_2, c_2):

$$\begin{vmatrix} f & g & h \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{vmatrix}$$

- (x_0, y_0, z_0) then readily follows.

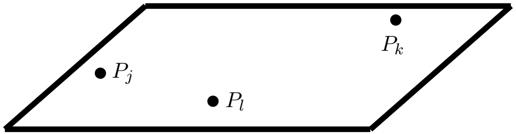
Intersection of Two Planes (Matlab Code)

The MATLAB code to do this is, planes_3d_3_intersect.m:

```
tol = eps;

f = b1*c2 - b2*c1;
g = c2*a2 - c2*a1;
h = a1*b2 - a2*b1;


det = f*f + g*g + h*h;

if (abs(det) < tol)
  error('planes_3d_2intersect_line: Planes are parallel');
end;

else
 dc = d1*c2 - c1*d2;
 db = d1*b2 - b1*d2;
 ad = a1*d1 - a2*d1;

 detinv = 1/det;

 x0 = (g*dc - h*db)*detinv;
 y0 = -(f*dc +h*ad)*detinv;
 z0 = (f*db +g*ab)*detinv;

 return;
end;
```

Plane Through Three Points

Just as two points define a line, ***three points*** define a plane (this is the **explicit** form of a plane):



A plane may be found as follows:

- Let P_j be the to point (x_0, y_0, z_0) in the plane
- Form two vectors in the plane $P_l - P_j$ and $P_k - P_j$
- Form another vector from a general point $P(x, y)$ in the plane to P_j
- The the equation of the plane is given by:

$$\begin{vmatrix} x - x_j & y - y_j & z - z_j \\ x_k - x_j & y_k - y_j & z_k - z_j \\ x_l - x_j & y_l - y_j & z_l - z_j \end{vmatrix}$$

- a, b, c and d can be found by expanding the determinant above

Back
Close

Plane Through Three Points (MATLAB Code)

The MATLAB code to do this, plane_exp2imp_3d.m

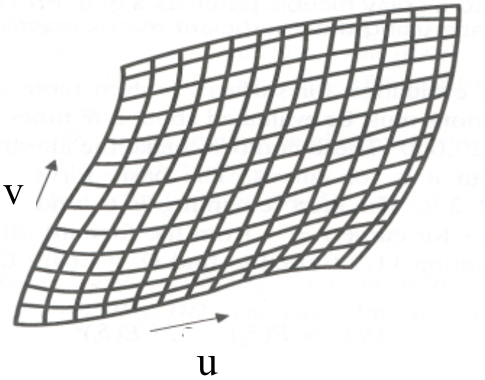
```
a = ( p2(2) - p1(2) ) * ( p3(3) - p1(3) ) ...
    - ( p2(3) - p1(3) ) * ( p3(2) - p1(2) );

b = ( p2(3) - p1(3) ) * ( p3(1) - p1(1) ) ...
    - ( p2(1) - p1(1) ) * ( p3(3) - p1(3) );

c = ( p2(1) - p1(1) ) * ( p3(2) - p1(2) ) ...
    - ( p2(2) - p1(2) ) * ( p3(1) - p1(1) );

d = - p2(1) * a - p2(2) * b - p2(3) * c;
```

Parametric Surfaces

The general form of a parametric surface is

$$\mathbf{r} = (x(u, v), y(u, v), z(u, v)).$$

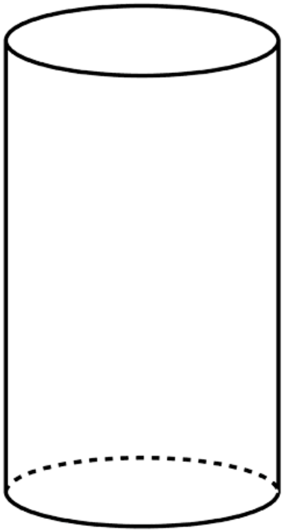
This is just like a parametric curve except we now have two parameters u and v that vary.

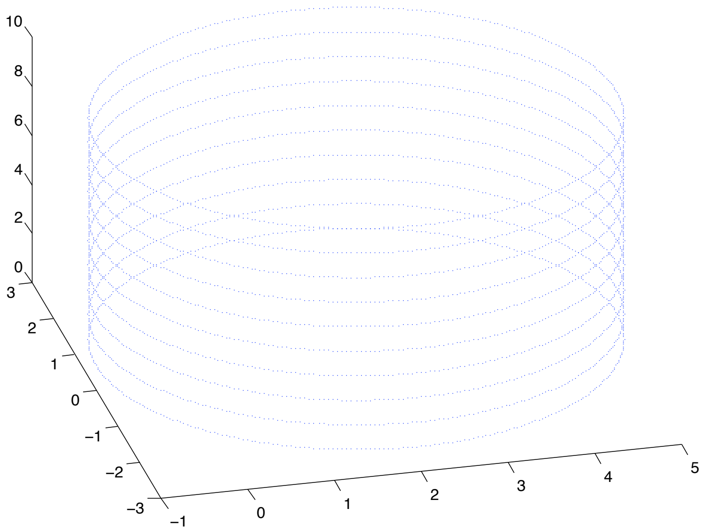


Parametric Surface: Cylinder

For example, a cylindrical may be represented in parametric form as

$$x = x_0 + r\cos u \qquad y = y_0 + r\sin u \qquad z = z_0 + v.$$





Back
Close

Parametric Surface: Cylinder (MATLAB Code

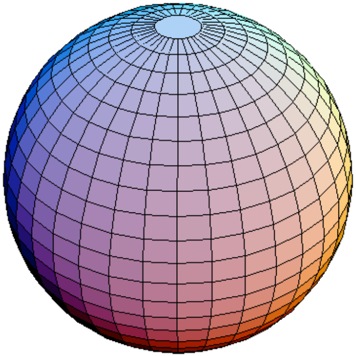
The MATLAB code to plot the cylinder figure is cyl_plot.m

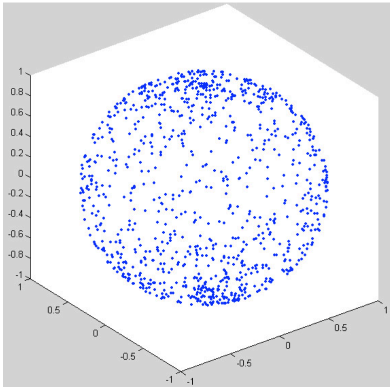
```
p0 = [2,0,0] % x_0, y_0, z_0
r = 3; %radius

n = 360;

hold on;
for v = 1:10
for u = 1:360
theta = ( 2.0 * pi * ( u - 1 ) ) / n;
x = p0(1) + r * cos(theta);
y = p0(2) + r * sin(theta);
z = p0(3) + v;

plot3(x,y,z);
end
end
```

Parametric Surface: Sphere

A sphere is represented in parametric form as

$$x = x_c + \sin(u) * \sin(v) \qquad y = y_c + \cos(u) * \sin(v) \qquad z = z_c + \cos(v)$$





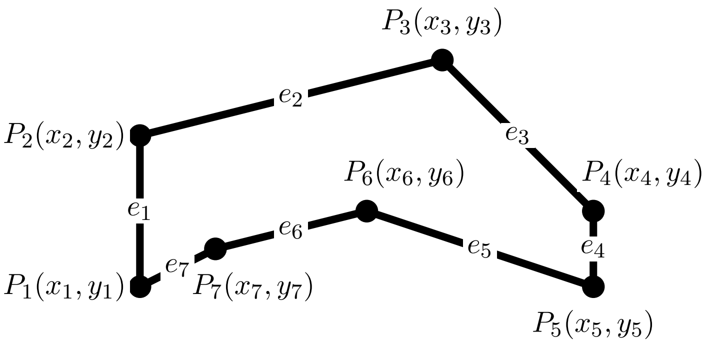
MATLAB code to produce a parametric sphere is at HyperSphere.m

Piecewise Shape Models

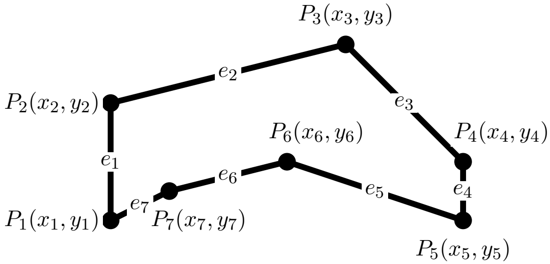
Polygons

A polygon is a 2D shape that is bounded by a closed path or circuit, composed of a finite sequence of straight line segments.

We can represent a polygon by a series of connected lines:



Polygons (Cont.)



- Each line is defined by two vertices — the start and end points.
- We can define a data structure which stores a list of points (coordinate positions) and the edges define by indexes to two points:

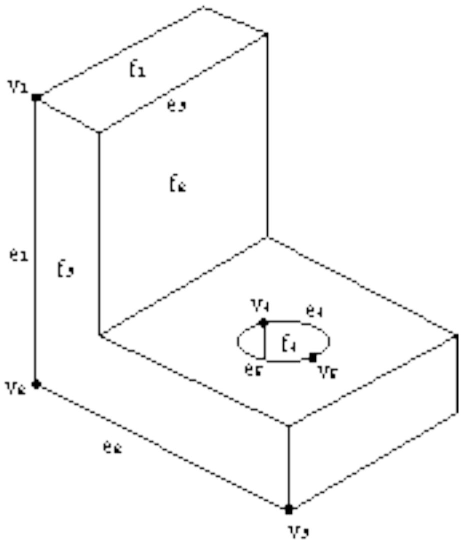
  Points : $\{P_1(x_1, y_1), P_2(x_2, y_2), P_3(x_3, y_3), \ldots\}$
  Points define the **geometry** of the polygon.

  Edges : $\text{Edges} : \{e_1 = (P_1, P_2), e_2 = (P_2, P_3), \ldots$
  Edges define the **topology** of the polygon.

- If you traverse the polygon points in an ordered direction (clockwise) then the lines define a closed shape with the inside on the right of each line.

3D Objects: Boundary Representation

In 3D we need to represent the object’s faces, edges and vertices and how they are joined together:



3D Objects: Boundary Representation

Topology — The topology of an object records the connectivity of the faces, edges and vertices. Thus,

- Each edge has a vertex at either end of it (e_1 is terminated by v_1 and v_2), and,
- each edge has two faces, one on either side of it (edge e_3 lies between faces f_1 and f_2).
- A face is either represented as an implicit surface or as a parametric surface
- Edges may be straight lines (like e_1), circular arcs (like e_4), and so on.

Geometry – This described the exact shape and position of each of the edges, faces and vertices. The geometry of a vertex is just its position in space as given by its (x, y, z) coordinates.

Back
Close

Geometric Transformations

Some Common Geometric Transformations:

2D Scaling: $\begin{pmatrix} x_k & 0 \\ 0 & y_k \end{pmatrix}$

2D Rotation:

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

2D Shear (x axis): $\begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$

2D Shear (y axis): $\begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix}$

3D Scaling: $\begin{pmatrix} x_k & 0 & 0 \\ 0 & y_k & 0 \\ 0 & 0 & z_k \end{pmatrix}$

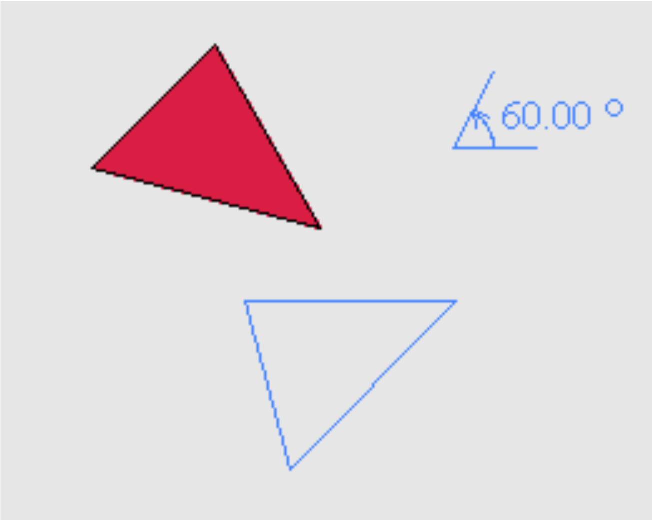
3D Rotation about z axis:

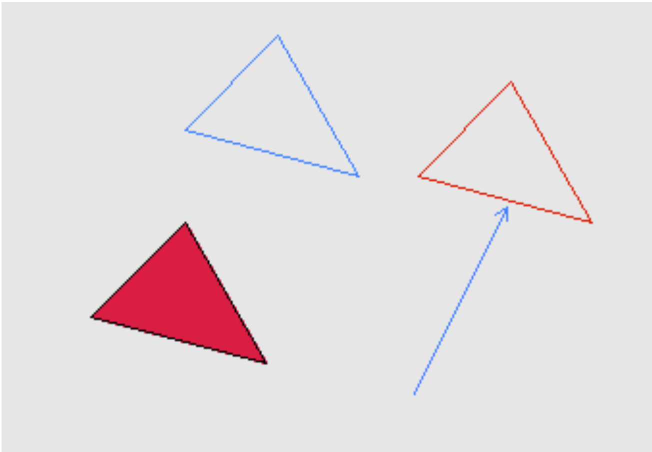
$$\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2D Translation (Homogeneous Coords):

$$\begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation and Translation





$$\begin{bmatrix} X_{rotated} \\ Y_{rotated} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_{translated} \\ Y_{translated} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & D_x \\ 0 & 1 & D_y \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Applying Geometric Transformations

When applying geometric transformations to objects

- Rotate the point coordinate and vectors (normals etc.)
  - Simply do matrix-vector multiplication
- For polygons and Boundary representations only need to apply transformation to the point (geometric) data – **the topology is unchanged**

Compound Geometric Transformations

It is a simple task to create compound transformations (*e.g.* A rotation followed by a translation:

- Perform Matrix-Matrix multiplications, *e.g.* $\mathbf{T}.\mathbf{R}_\theta$
  - Note in general $\mathbf{T}.\mathbf{R}_\theta \neq \mathbf{R}_\theta.\mathbf{T}$ (Basic non-commutative law of multiplication)
- One problem exists in that a 2D Translation **cannot** be represented as a 2D matrix — it must be represented a 3D matrix:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{pmatrix}$$

- So how do we combine a 2D rotation $\mathbf{R}_\theta. = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$ with a translation?

  We need to define **Homogeneous Coordinates** to deal with this issue.

Homogeneous Coordinates

To work in homogeneous coordinates do the following:

- Increase the dimension of the space by 1, *i.e. 2D* $\rightarrow$ 3D, 3D$\rightarrow$ 4D. We can now accommodate the translation matrix.
- Convert all standard other 2D transformations via the following:

$$\begin{pmatrix} X & X & 0 \\ X & X & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

  where $\begin{pmatrix} X & X \\ X & X \end{pmatrix}$ is the regular 2D transformation

- Perform compound transformation by matrix-matrix multiplication in homogeneous matrices

Homogeneous Coordinates Example: 2D Rotation

The transformation for a 2D rotation (angle of rotation θ is:

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

The corresponding ***homogenous form*** is:

$$\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Compound Geometric Transformations MATLAB Examples

- Simply create the matrices for individual transformations in MATALB (2D, 3D or homogeneous)
- Assemble compound via matrix multiplication.
- Apply compound transform to coordinates/vectors etc. via matrix-vector multiplication.

2D Geometric Transformation MATLAB Demo Code

3D Transformations

3D Rotation about x axis:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix}$$

3D Rotation about y axis:

$$\begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

3D Rotation about z axis:

$$\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3D Scaling:

$$\begin{pmatrix} x_k & 0 & 0 \\ 0 & y_k & 0 \\ 0 & 0 & z_k \end{pmatrix}$$

3D Translation (Homogeneous Coords (4D):

$$\begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Need to put all 3D transformation into homogenous form for compound transformations.

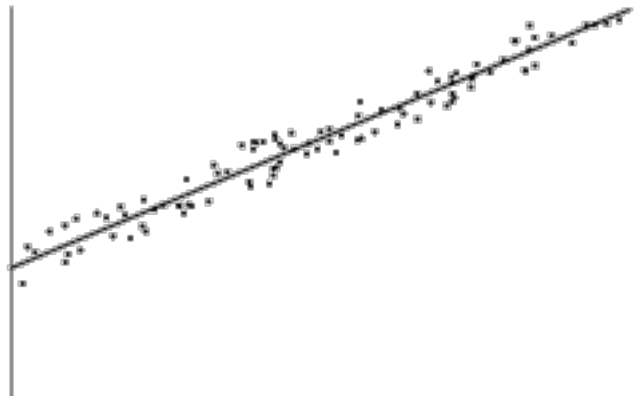
Least Squares Fitting

We have already seen that we need 2 points to define a line and 3 points to define circle and a plane — the minimum number of free parameters.

However, in many data processing applications we will have many more samples than the minimum number required.

- Errors in the positions of these points mean that in practice they do not all lie exactly on the expected line/curve/surface.
- ***Least squares approximation*** methods find the surface of the given type which best represents the data points.
  - Minimise some error function

Least Squares Fit of a Line



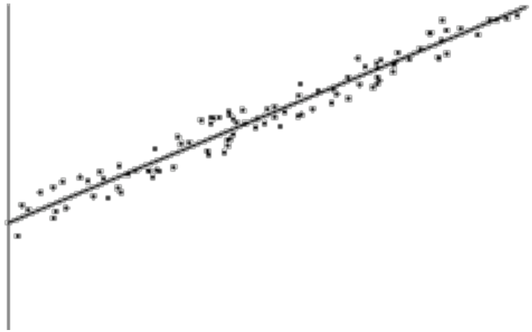
We use the general explicit equation of a line here:

$$y = mx + c$$

We can define an error, σ_i as the distance from the line:

$$\sigma_i = y_i - mx_i - c$$

Least Squares Fit of a Line (Cont.)



The best fitting plane to a set of n points $\{P_i\}$ occurs when

$$\chi^2 = \sum \sigma_i^2 = \sum (y_i - m x_i - c)^2$$

is a minimum, where this and all subsequent sums are to be taken over all points, $P_i, i = 1, \ldots, n$.

This occurs when

$$\begin{aligned} \frac{\partial \chi^2}{\partial m} &= 0 \\ \frac{\partial \chi^2}{\partial c} &= 0 \end{aligned}$$

Least Squares Fit of a Line (Cont.)

Now

$$\frac{\partial \chi^2}{\partial m} = -2\sum(y_i - mx_i - c)x_i = 0$$

$$\frac{\partial \chi^2}{\partial c} = -2\sum(y_i - mx_i - c) = 0$$

which leads to

$$\begin{aligned} \sum y_i - m\sum x_i - nc &= 0 \\ \sum x_i y_i - m\sum x_i^2 - c\sum x_i &= 0 \end{aligned}$$

which can be solved for m and c to give:

$$\begin{aligned} m &= \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \\ c &= \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2} \end{aligned}$$

Least Squares Fit of a Line (MATLAB Code)

The MATLAB code to perform least squares line fitting is, line_fit.m

```
function [m c] = line_fit(x,y)

[n1 n] = size(x);

sumx = sum(x);
sumy = sum(y);

sumx2 = sum(x.*x);

sumxy = sum(x.*y);

det = n* sumx2 - sumx*sumx;

m = (n*sumxy - sumx*sumy)/det;

c = (sumy*sumx2 - sumx*sumxy)/det;
```

See line_fit_demo.m for example call.

Least Squares Fit of a Plane

Let us suppose we are given a set of points in the form $z = z(x, y)$, *i.e.* we are given z_i values for the points lying at a set of known (x_i, y_i) positions.

The equation of a plane can be written as

$$z = ax + by + c.$$

For a given point $P_i = (x_i, y_i, z_i)$ the error in the fit is measured by

$$\sigma_i = z_i - (ax_i + by_i + c).$$

Least Squares Fit of a Plane (Cont.)

The best fitting plane to a set of n points $\{P_i\}$ occurs when $\chi^2 = \sum \sigma_i^2$ is a minimum, where this and all subsequent sums are to be taken over all points, $P_i, i = 1, \ldots, n$.

This occurs when

$$\frac{\partial \chi^2}{\partial a} = 0 \tag{1}$$

and

$$\frac{\partial \chi^2}{\partial b} = 0 \tag{2}$$

and and when

$$\frac{\partial \chi^2}{\partial c} = 0 \tag{3}$$

Least Squares Fit of a Plane (Cont.)

Now $\frac{\partial \chi^2}{\partial a} = 0$ gives:

$$\sum x_i z_i - a \sum x_i^2 - b \sum x_i y_i - c \sum x_i = 0,$$

and $\frac{\partial \chi^2}{\partial b} = 0$ gives:

$$\sum y_i z_i - a \sum x_i y_i - b \sum y_i^2 - c \sum y_i = 0,$$

and $\frac{\partial \chi^2}{\partial c} = 0$ gives

$$\sum z_i - a \sum x_i - b \sum y_i - nc = 0.$$

Simplifying a Least Squares Fit of a Plane

Now, some of these sums can become quite large and any attempt to solve the equations directly can lead to poor results due to rounding errors.

- One common method (for any least squares approximation) of reducing the rounding errors is to re-express the coordinates of each point relative to the centre of gravity of the set of points, (x_g, y_g, z_g).
- All subsequent calculations are performed under this translation with the final results requiring the inverse translation to return to the original coordinate system.

The translation also simplifies our minimisation problem since now

$$\sum_{\forall i} x_i = \sum_{\forall i} y_i = \sum_{\forall i} z_i = 0.$$

Least Squares Fit of a Plane (Cont.)

Therefore, using the above fact in previous equations and solving for a, b and c gives:

$$
\begin{aligned}
a &= \frac{\sum x_i z_i \sum y_i^2 - \sum y_i z_i \sum x_i y_i}{\sum x_i^2 \sum y_i^2 - (\sum x_i y_i)^2}, && (4)\\
b &= \frac{\sum y_i z_i \sum x_i^2 - \sum x_i z_i \sum x_i y_i}{\sum x_i^2 \sum y_i^2 - (\sum x_i y_i)^2}, && (5)\\
c &= 0. && (6)
\end{aligned}
$$

All we now need to do is to translate back.

Example Applications Visited

- Geographic Information Systems: Point Location
- Geometric Modelling: Spline Fitting
- Computer Graphics: Ray Tracing
- Image Processing: Hough Transform
- Mobile Systems: Spatial Location Sensing

Application-specific techniques are out of the scope of this module but geometric computing is widely used in all of these applications.

THE
END