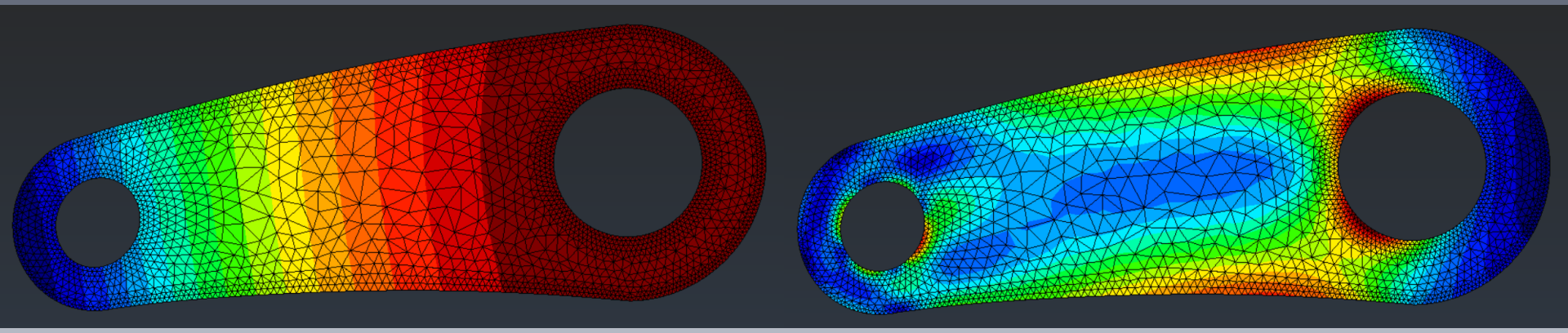
MECÁNICA ESTÁTICA
ME3130



Alejandro Ortiz Bernardin

aortizb@uchile.cl

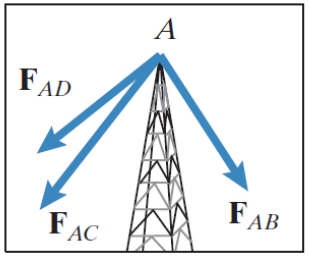
www.camlab.cl/alejandro

DIMEC
INGENIERÍA MECÁNICA
UNIVERSIDAD DE CHILE

ESTÁTICA DE PARTÍCULAS

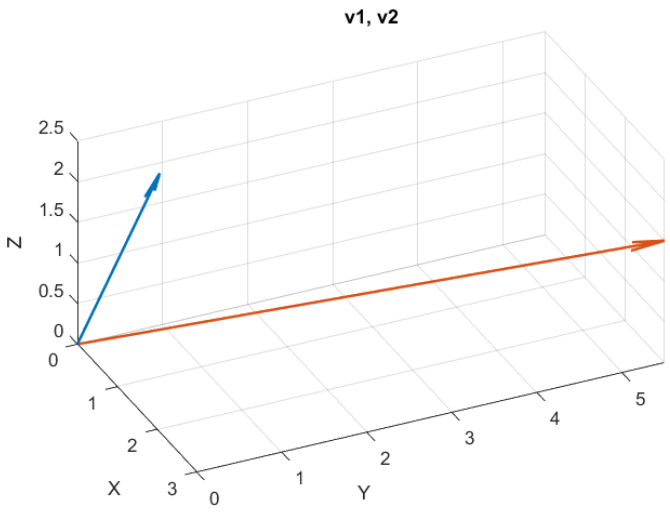
Contenidos

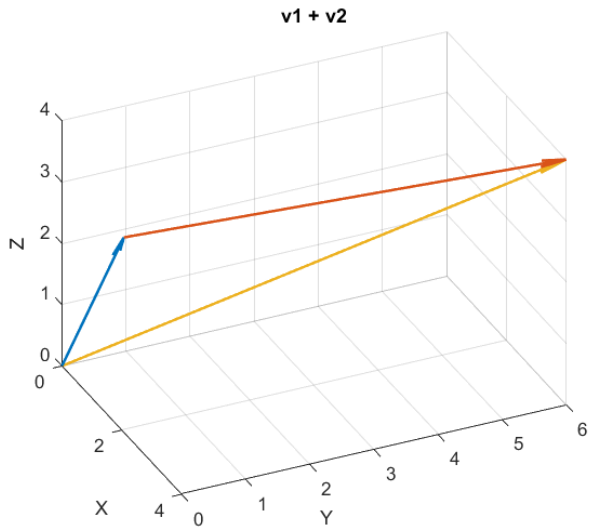
- Estática se asocia al equilibrio de fuerzas de cuerpos en reposo



- Estática de partículas = el problema puede resolverse por equilibrio de una “partícula”

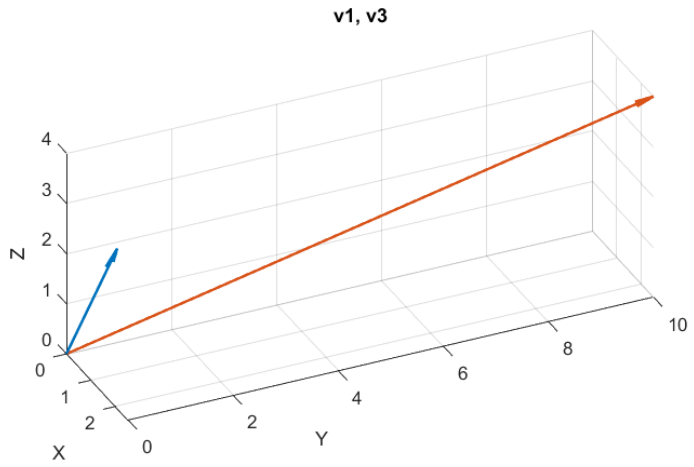
Adición de Vectores

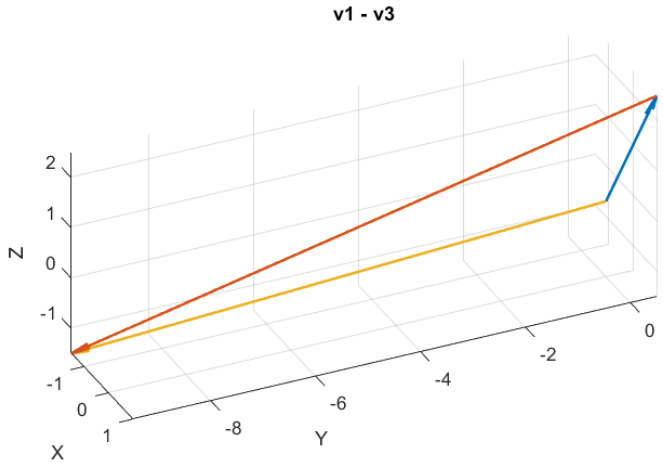





```
%-------------------------------------------------------------------------
%                              Código MATLAB
%-------------------------------------------------------------------------
% Definición de vectores
%  v1 = v1x i + v1y j + v1z k
%  v2 = v2x i + v2y j + v3z k
%  v3 = v2x i + v2y j + v3z k
v1x = 1; v1y = 0.5 ; v1z = 2.5;
v2x = 3.0; v2y = 5.5 ; v2z = 1.5;
v3x = 2.5; v3y = 10 ; v3z = 4.0;
v1 = [v1x, v1y, v1z];
v2 = [v2x, v2y, v2z];
v3 = [v3x, v3y, v3z];

%-- Ejemplo  ---------------------------------------------------------------
a1 = v1 + v2;
% ploteo de la suma de vectores: a1 = v1 + v2
figure; % define una nueva figura
x0 = [0,0,0]; % origen del vector
plotvec3d(x0,v1,'LineWidth',1.5,'MaxHeadSize',0.4);
hold on; % mantiene la figura para seguir ploteando sobre ella
view([65,30]);
%
x0 = v1; % define el origen donde termina v1
plotvec3d(x0,v2,'LineWidth',1.5,'MaxHeadSize',0.2);
%
x0 = [0,0,0]; % define el origen en [0,0,0]
plotvec3d(x0,a1,'LineWidth',1.5,'MaxHeadSize',0.2);
title('v1 + v2');
```

Adición de Vectores






```
%-------------------------------------------------------------------------
%                              Código MATLAB
%-------------------------------------------------------------------------
% Definición de vectores
%  v1 = v1x i + v1y j + v1z k
%  v2 = v2x i + v2y j + v3z k
%  v3 = v2x i + v2y j + v3z k
v1x = 1; v1y = 0.5 ; v1z = 2.5;
v2x = 3.0; v2y = 5.5 ; v2z = 1.5;
v3x = 2.5; v3y = 10 ; v3z = 4.0;
v1 = [v1x, v1y, v1z];
v2 = [v2x, v2y, v2z];
v3 = [v3x, v3y, v3z];

%-- Ejemplo ----------------------------------------------------------------
s1 = v1 - v3;
% ploteo de la suma de vectores: s1 = v1 - v3
figure; % define una nueva figura
x0 = [0,0,0]; % origen del vector
plotvec3d(x0,v1,'LineWidth',1.5,'MaxHeadSize',0.5);
hold on; % mantiene la figura para seguir ploteando sobre ella
view([65,30]);
%
x0 = v1; % define el origen donde termina v1
plotvec3d(x0,-v3,'LineWidth',1.5,'MaxHeadSize',0.1);
%
x0 = [0,0,0]; % define el origen en [0,0,0]
plotvec3d(x0,s1,'LineWidth',1.5,'MaxHeadSize',0.1);
title('v1 - v3');
```

Función de Ploteo de Vectores 3D

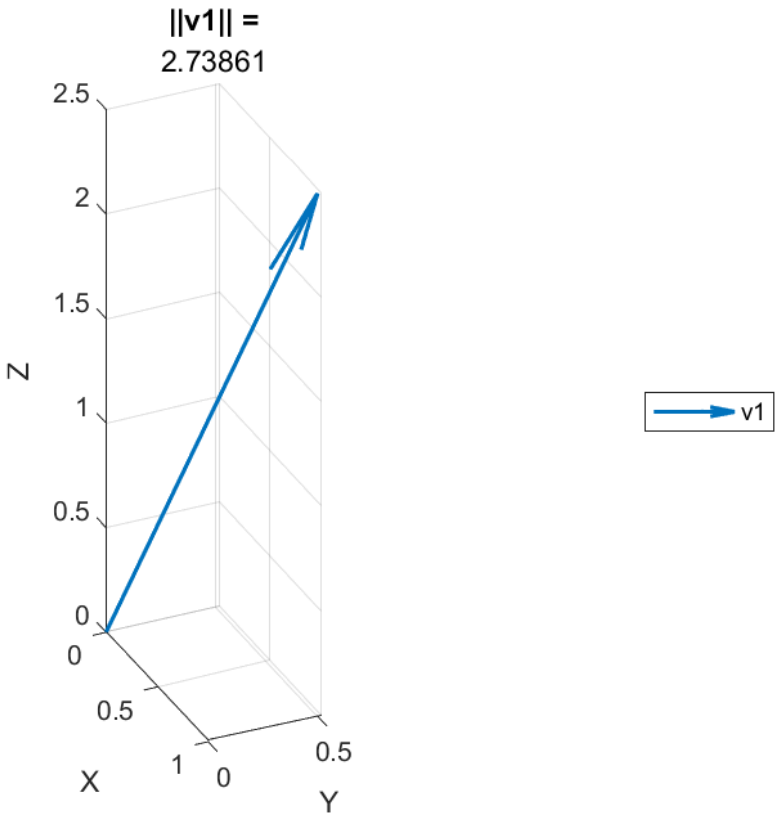
- La función `plotvec3d` es una función “wrapper”(*) de la función de la librería standard de MATLAB llamada `quiver3`

```
%-------------------------------------------------------------------------------------------------------------
%                                          Código MATLAB
%-------------------------------------------------------------------------------------------------------------
function plotvec3d(x0,dx,LineWidthStr,LineWidthVal,MaxHeadSizeStr,MaxHeadSizeVal)
 if length(x0)~=3 || length(dx)~=3
   error('x0 y dx deben ser arreglos de dimension 3');
 end
 quiver3(x0(1),x0(2),x0(3),dx(1),dx(2),dx(3),'off',...
            LineWidthStr,LineWidthVal,MaxHeadSizeStr,MaxHeadSizeVal);
 xlabel('X’);
 ylabel('Y’);
 zlabel('Z’);
 axis equal;
end
```

- Para obtener ayuda sobre `quiver3`, en la línea de comando de MATLAB ejecutar: `help quiver3`

(*) https://en.wikipedia.org/wiki/Wrapper_function

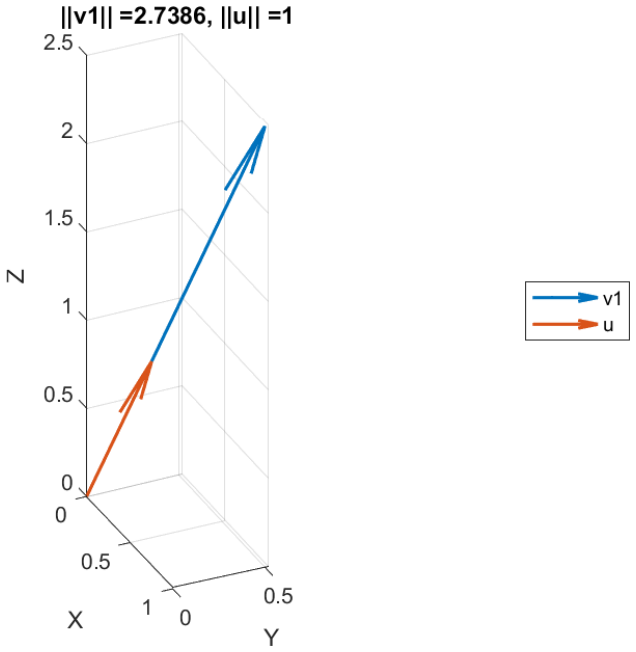
Norma/Magnitud/Módulo de un Vector




```
Norma de un vector
------------------

v1 = [1.000000, 0.500000, 2.500000]
||v1|| = sqrt(v1(1)^2 + v1(2)^2 + v1(3)^2) = 2.738613
||v1|| =  norm(v1) = 2.738613
```

```
%-------------------------------------------------------------------------
%                              Código MATLAB
%-------------------------------------------------------------------------
% Definición de vectores
%  v1 = v1x i + v1y j + v1z k
v1x = 1; v1y = 0.5 ; v1z = 2.5;
v1 = [v1x, v1y, v1z];

%-- Ejemplo ----------------------------------------------------------------
fprintf('Norma de un vector\n');
fprintf('-------------------------\n\n');
fprintf('v1 = [%f, %f, %f]\n',v1); % impresión como arreglo
fprintf('||v1|| = sqrt(v1(1)^2 + v1(2)^2 + v1(3)^2) =
%f\n',sqrt(v1(1)^2+v1(2)^2+v1(3)^2));
fprintf('||v1|| = norm(v1) = %f\n\n',norm(v1));

% ploteo del vector desde el origen
figure; % define una nueva figura
x0 = [0,0,0]; % origen del vector
plotvec3d(x0,v1,'LineWidth',1.5,'MaxHeadSize',0.5); % plotea el
vector v1 con origen en x0
view([65,30]);
legend('v1','Location','eastoutside');
title('||v1|| =',norm(v1));
```

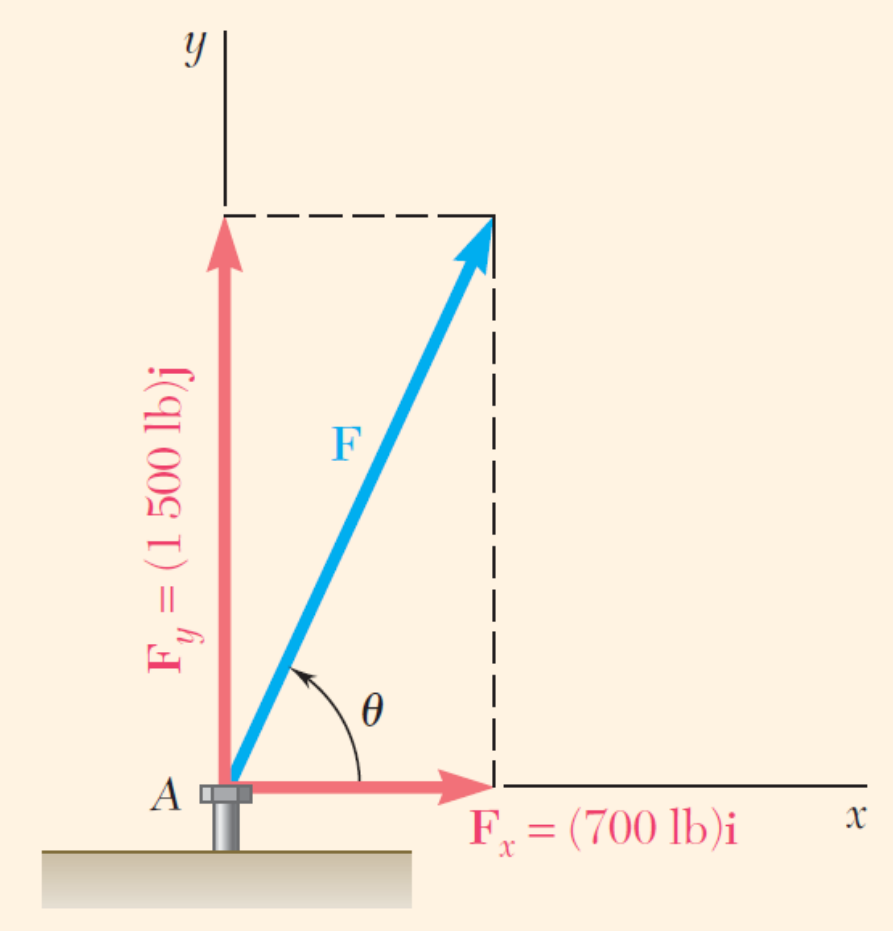
Vector Unitario




```
Vector unitario
---------------

v1 = [1.000000, 0.500000, 2.500000]

Norma del vector v1:
||v1|| = sqrt(v1(1)^2 + v1(2)^2 + v1(3)^2) = norm(v1) = 2.738613

Vector unitario u:
u =  v1/||v1|| = [0.365148, 0.182574, 0.912871]

Norma del vector unitario:
||u|| = sqrt(u(1)^2 + u(2)^2 + u(3)^2) = norm(u) = 1.000000

Reconstrucción de v1 a partir del vector unitario u:
v1 = ||v1||u = [1.000000, 0.500000, 2.500000]
```

```
%-------------------------------------------------------------------------
%                              Código MATLAB
%-------------------------------------------------------------------------
% Definición de vectores
%  v1 = v1x i + v1y j + v1z k
v1x = 1; v1y = 0.5 ; v1z = 2.5;
v1 = [v1x, v1y, v1z];
%-- Ejemplo ----------------------------------------------------------------
fprintf('v1 = [%f, %f, %f]\n\n',v1);
fprintf('Norma del vector v1:\n');
fprintf('||v1|| = sqrt(v1(1)^2 + v1(2)^2 + v1(3)^2) = norm(v1) =
%f\n\n',norm(v1));
fprintf('Vector unitario u:\n');
u = v1/norm(v1); % vector unitario
fprintf('u = v1/||v1|| = [%f, %f, %f]\n\n',u);
fprintf('Norma del vector unitario:\n');
fprintf('||u|| = sqrt(u(1)^2 + u(2)^2 + u(3)^2) = norm(u) =
%f\n\n',norm(u));
fprintf('Reconstrucción de v1 a partir del vector unitario u:\n');
fprintf('v1 = ||v1||u = [%f, %f, %f]\n\n',norm(v1)*u);
figure; % define una nueva figura
x0 = [0,0,0]; % origen del vector
plotvec3d(x0,v1,'LineWidth',1.5,'MaxHeadSize',0.5); % vector v1
hold on; % mantiene la figura para seguir ploteando sobre ella
view([65,30]);
x0 = [0,0,0]; % origen del vector
plotvec3d(x0,u,'LineWidth',1.5,'MaxHeadSize',1.5); % vector u
 legend('v1','u','Location','eastoutside');
title(['||v1|| =',num2str(norm(v1)),', ||u|| =',num2str(norm(u))]);
```

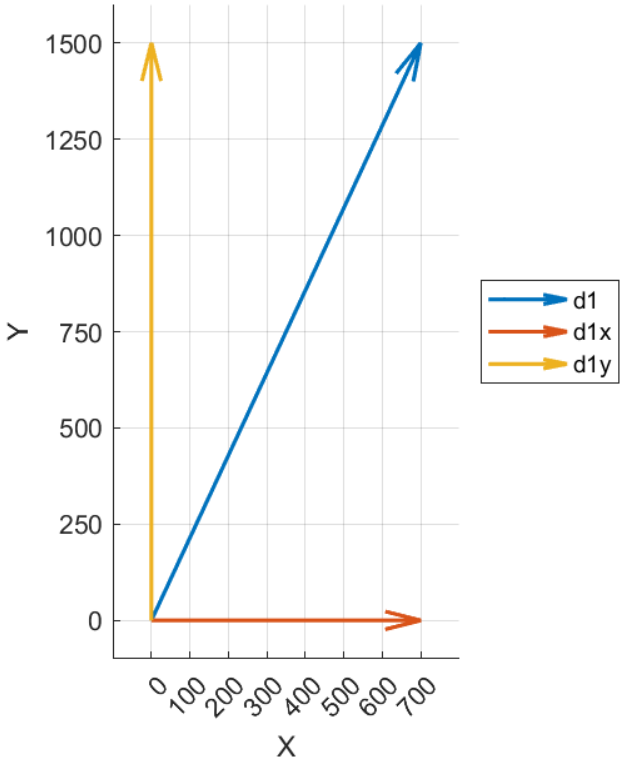
Descomposición de Vectores



Ejemplo: Fuerza sobre un perno (caso 2D).

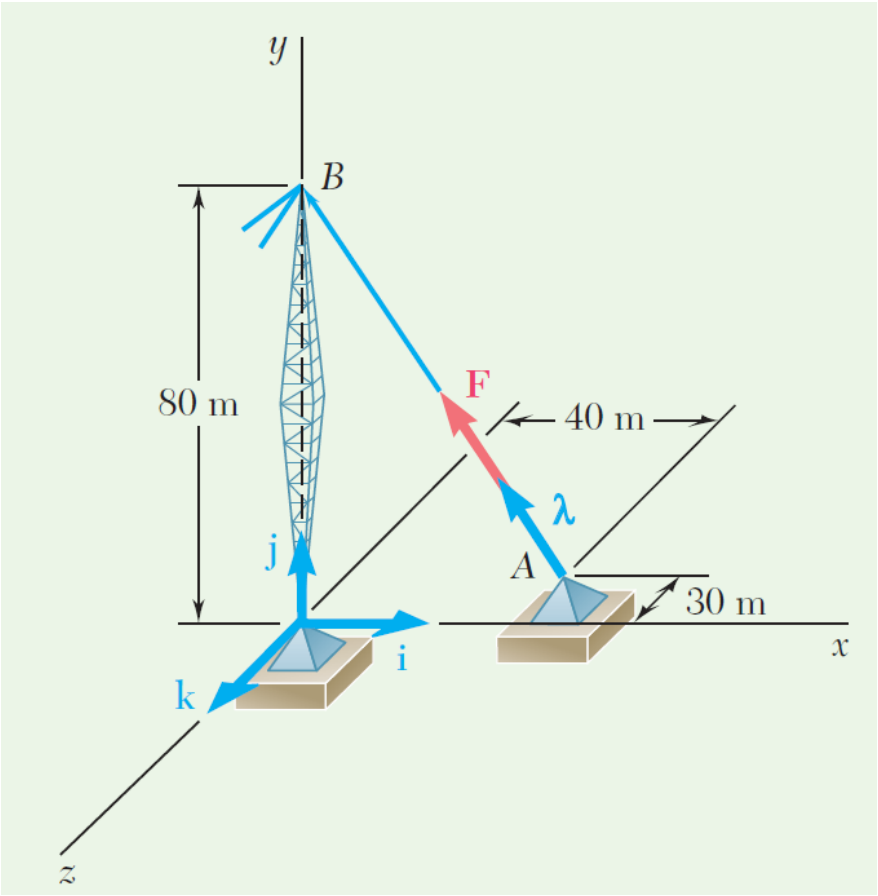
Calcular la magnitud de la fuerza F y el ángulo entre F y el eje x.

Descomposición de Vectores




```
Descomposición de vectores 2D
-----------------------------

Ángulo entre vector d1 y eje x:

alpha_x = atan(d1y/d1x)*180/pi() = 64.983107°

Magnitud del vector d1:

d1_mag = d1x/cos(alpha_x) = 1655.294536
```

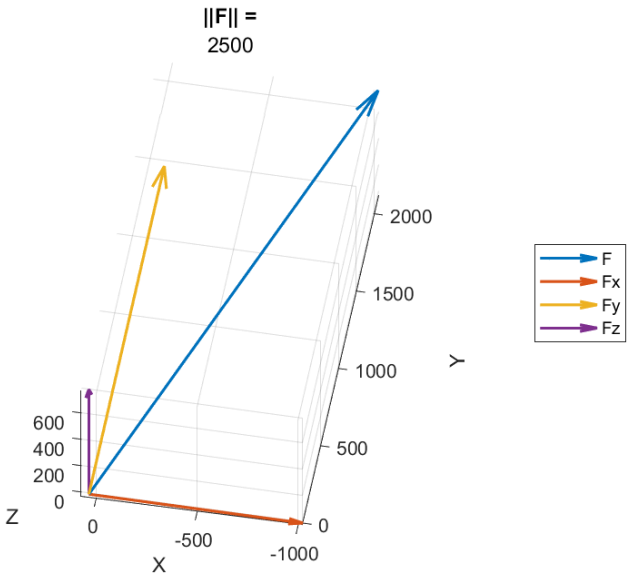
```
%--------------------------------------------------------------------------
%                                Código MATLAB
%--------------------------------------------------------------------------
% definición de un vector 2D
d1 = [700,1500,0]; % también puede ser: d1 = [700 1500 0];
% ploteo del vector d1
figure; % define una nueva figura
x0 = [0,0,0]; % origen del vector
plotvec3d(x0,d1,'LineWidth',1.5,'MaxHeadSize',0.2); % plotea el vector d1
view(2); % vista frontal por defecto para caso 2D
% descomposición del vector en el eje x e y
d1x = d1(1); d1y = d1(2); % componente x e y del vector d1
% ploteo de las componentes
hold on; % mantiene la figura para seguir ploteando sobre ella
plotvec3d(x0,[d1x 0 0],'LineWidth',1.5,'MaxHeadSize',0.4); % plotea vector d1x
plotvec3d(x0,[0 d1y 0],'LineWidth',1.5,'MaxHeadSize',0.2); % plotea vector d1y
axis([-100 800 -100 1600]); % fija los límites de los ejes
xticks(0:100:700); % especifica ciertos valores en el eje x
yticks([0,250,500,750,1000,1250,1500]); % especifica ciertos valores en el eje y
legend('d1','d1x','d1y','Location','eastoutside');
% ángulo entre vector d1 y eje x
fprintf('Ángulo entre vector d1 y eje x:\n\n');
alpha_x = atan(d1y/d1x);
fprintf('alpha_x = atan(d1y/d1x)*180/pi() = %f °\n\n', alpha_x*180/pi());
% magnitud del vector d1
fprintf('Magnitud del vector d1:\n\n');
d1_mag = d1x/cos(alpha_x);
fprintf('d1_mag = d1x/cos(alpha_x) = %f\n\n', d1_mag);
```



Ejemplo: Fuerza sobre un perno (caso 3D).

Si F = 2500 N, determinar sus componentes Fx, Fy y Fz, y los ángulos θx, θy y θz que definen la dirección de la fuerza .

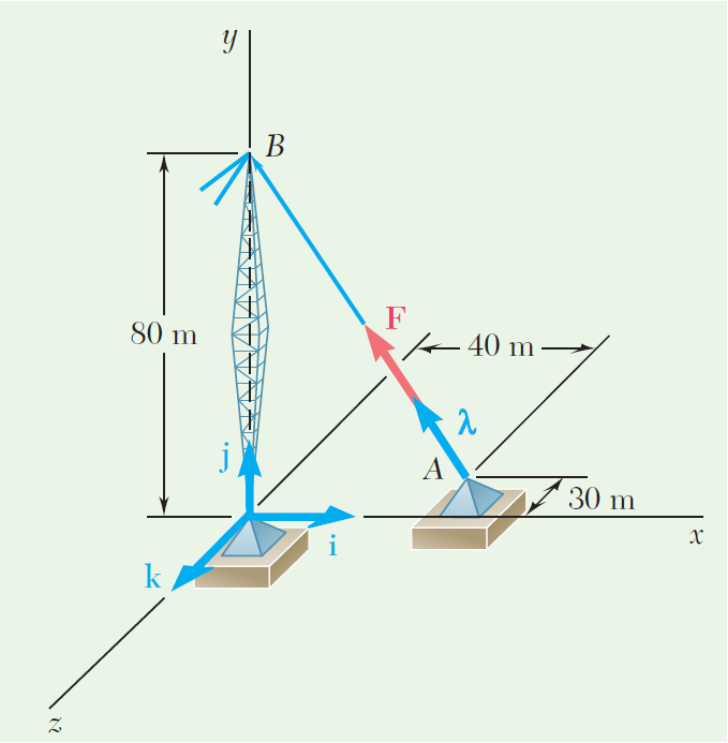
Descomposición de Vectores




```
Descomposición de vectores 3D
-----------------------------

F_mag = ||F|| = 2500.000000
AB = [dx,dy,dz] = [-40.000000, 80.000000, 30.000000]

u = AB/||AB|| = [-0.423999, 0.847998, 0.317999]

F = ||F||u = [-1059.997880, 2119.995760, 794.998410]

Cosenos directores:
cos(tx) = Fx/||F|| = -0.423999
cos(ty) = Fx/||F|| = 0.847998
cos(tz) = Fx/||F|| = 0.317999

Ángulos:
tx = acos(costx)*180/pi() = 115.087329°
ty = acos(costy)*180/pi() = 32.005383°
tz = acos(costz)*180/pi() = 71.458022°
```

```
%--------------------------------------------------------------------------
%                              Código MATLAB
%--------------------------------------------------------------------------
fprintf('Descomposición de vectores 3D\n');
fprintf('-----------------------------------------\n\n');
F_mag = 2500; % dado un vector F, su magnitud es F_mag = norm(F)
fprintf('F_mag = ||F|| = %f\n',F_mag);
% definición del vector unitario a lo largo de la dirección de F_mag
dx = -40; dy = 80; dz = 30; % componentes del vector distancia AB
AB = [dx,dy,dz]; % vector distancia desde A hasta B
u = AB/norm(AB); % vector unitario en la dirección del vector AB
fprintf('AB = [dx,dy,dz] = [%f, %f, %f]\n\n',AB);
fprintf('u = AB/||AB|| = [%f, %f, %f]\n\n',u);
% construcción del vector F a partir del vector unitario u
F = F_mag*u; fprintf('F = ||F||u = [%f, %f, %f]\n\n',F);
% ploteo del vector F
figure; % define una nueva figura
x0 = [40,0,-30]; % origen del vector es el origen del vector AB
plotvec3d(x0,F,'LineWidth',1.5,'MaxHeadSize',0.2); % plotea el vector F
hold on; view([170,-50]);
plotvec3d(x0,[F(1),0,0],'LineWidth',1.5,'MaxHeadSize',0.2); % componente x
plotvec3d(x0,[0,F(2),0],'LineWidth',1.5,'MaxHeadSize',0.2); % componente y
plotvec3d(x0,[0,0,F(3)],'LineWidth',1.5,'MaxHeadSize',0.2); % componente z
legend('F','Fx','Fy','Fz','Location','eastoutside'); title('||F|| =',norm(F));
% dirección de la fuerza
fprintf('Cosenos directores:\n');
costx = F(1)/norm(F); costy = F(2)/norm(F); costz = F(3)/norm(F);
tx = acos(costx); ty = acos(costy); tz = acos(costz);
fprintf('cos(tx) = Fx/||F|| = %f\n',costx); fprintf('cos(ty) = Fx/||F|| = %f\n',costy);
fprintf('cos(tz) = Fx/||F|| = %f\n\n',costz); fprintf('Ángulos:\n');
fprintf('tx = acos(costx)*180/pi() = %f°\n',tx*180/pi());
fprintf('ty = acos(costy)*180/pi() = %f°\n',ty*180/pi());
fprintf('tz = acos(costz)*180/pi() = %f°\n\n',tz*180/pi());
```

Descomposición de Vectores 3D




```
Propiedades de los cosenos directores
-------------------------------------

Reconstrucción del vector unitario:
u = cos(tx)i + cos(ty)j + cos(tz)k
  = [costx, costy, costz]
  = [-0.423999, 0.847998, 0.317999]

Relación entre cosenos directores:
cos(tx)^2 + cos(ty)^2 + cos(tz)^2 = 1.000000
```

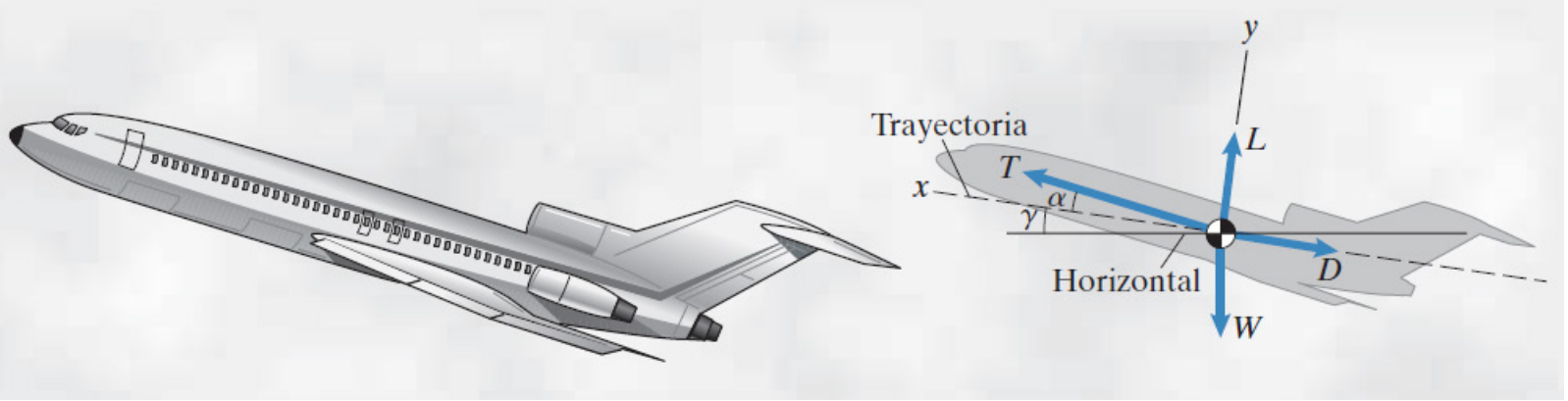
Propiedades cosenos directores

```
F_mag = ||F|| = 2500.000000
AB = [dx,dy,dz] = [-40.000000, 80.000000, 30.000000]

u = AB/||AB|| = [-0.423999, 0.847998, 0.317999]

F = ||F||u = [-1059.997880, 2119.995760, 794.998410]

Cosenos directores:
cos(tx) = Fx/||F|| = -0.423999
cos(ty) = Fx/||F|| = 0.847998
cos(tz) = Fx/||F|| = 0.317999
```

```
%--------------------------------------------------------------------------
%                          Código MATLAB
%--------------------------------------------------------------------------
% … continúa de slide anterior
%
fprintf('Propiedades de los cosenos directores\n');
fprintf('----------------------------------------------\n\n');
fprintf('Reconstrucción del vector unitario:\n');
fprintf('u = (costx)i + (costy)j + (costz)k\n');
fprintf(' = [costx, costy, costz]\n');
fprintf(' = [%f, %f, %f]\n\n',costx,costy,costz);
fprintf('Relación entre cosenos directores:\n');
fprintf('costx*costx + costy*costy + costz*costz = %f\n\n',...
costx*costx + costy*costy + costz*costz);
```

Primera Ley del Movimiento de Newton:

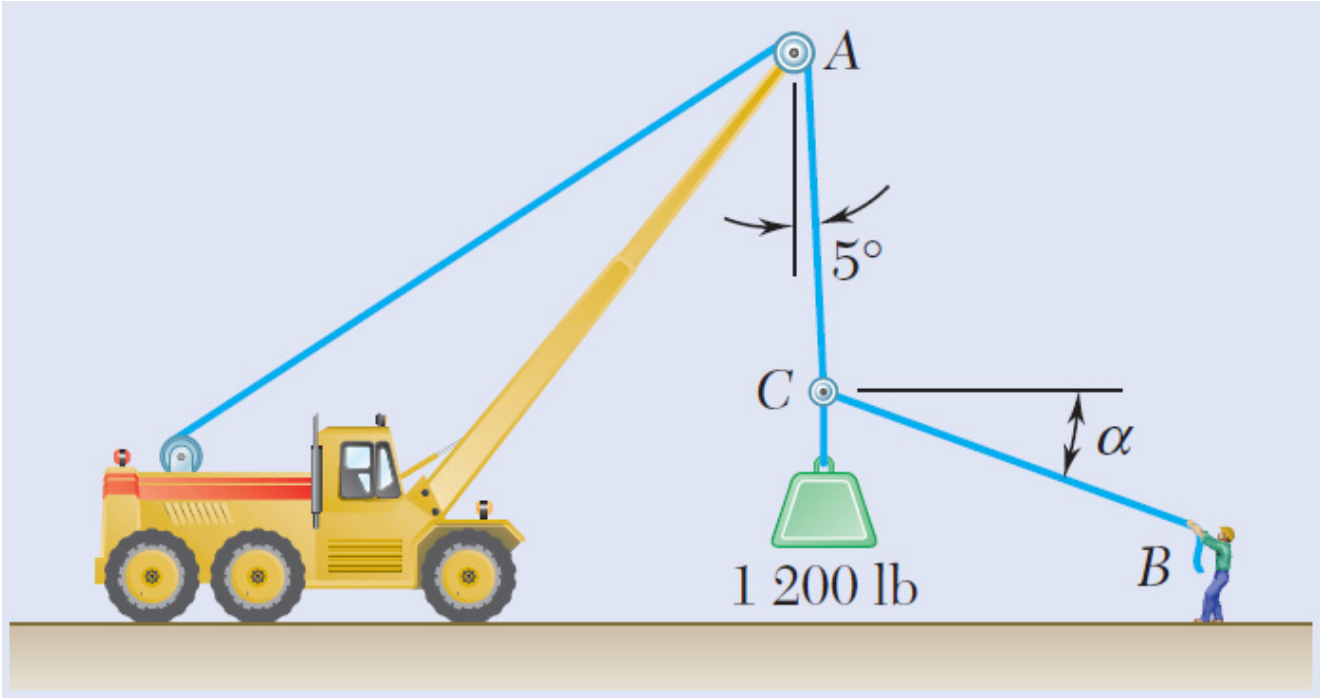
Si una fuerza resultante que actúa sobre una partícula es cero, la partícula permanecerá en reposo (si originalmente estaba en reposo) o se moverá con velocidad constante en línea recta (si originalmente estaba en movimiento).



T = empuje ejercido por motores; W = peso del avión

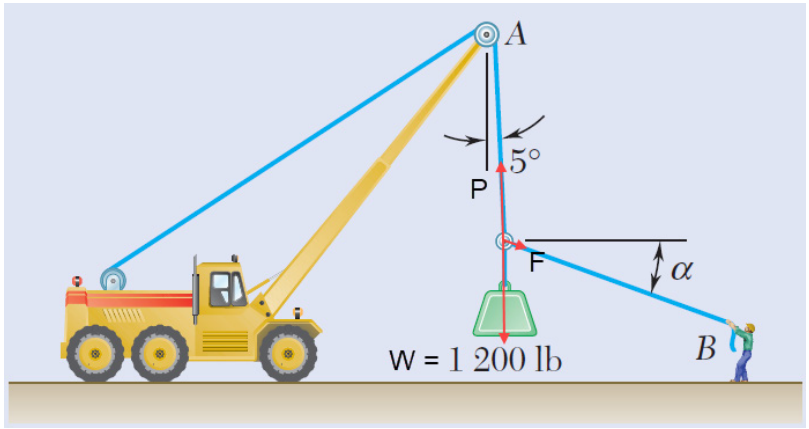
Fuerzas hidrodinámicas debido a presión sobre superficie:

L = fuerza de levante; D = fuerza de arrastre



Ejemplo: Si $\alpha = 20°$, determinar la tensión en el cable AC y en la cuerda BC.

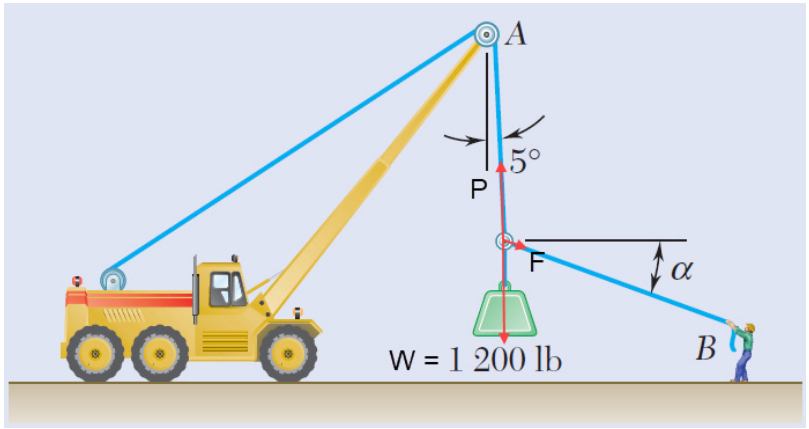
Equilibrio en el Plano: DCL Partícula



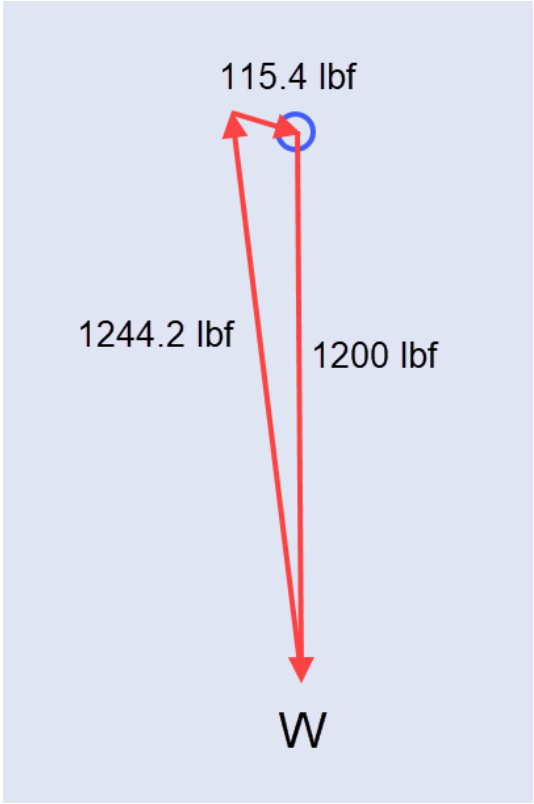

```
Equilibrio en el plano
----------------------

Datos:
W = 1200.000000 lbf
alpha = 0.349066 rad
beta = 0.087266 rad

Sumatoria de fuerzas en eje horizontal = 0:
F*cos(alpha) - P*sin(beta) = 0
--> F = P*sin(beta)/cos(alpha)

Sumatoria de fuerzas en eje vertical = 0:
-F*sin(alpha) + P*cos(beta) - W = 0
--> -P*sin(beta)/cos(alpha)*sin(alpha) + P*cos(beta) - W = 0
--> -P*sin(beta)*tan(alpha) + P*cos(beta) - W = 0
-->  P*(cos(beta) - sin(beta)*tan(alpha)) - W = 0

Solución:
P = W/(cos(beta) - sin(beta)*tan(alpha)) = 1244.203306 lbf
-->  F = P*sin(beta)/cos(alpha) = 115.398866 lbf
```

```
%-------------------------------------------------------------------------
%                              Código MATLAB
%-------------------------------------------------------------------------
fprintf('Equilibrio en el plano\n');
fprintf('----------------------\n\n');
% Datos:
W = 1200; % peso soportado
alpha = 20*pi()/180; % ángulo en radianes
beta = 5*pi()/180; % ángulo en radianes
fprintf('Datos:\n');
fprintf('W = %f lbf\n',W);
fprintf('alpha = %f rad\n',alpha);
fprintf('beta = %f rad\n\n',beta);
fprintf('Sumatoria de fuerzas en eje horizontal = 0:\n');
fprintf('F*cos(alpha) - P*sin(beta) = 0\n');
fprintf('--> F = P*sin(beta)/cos(alpha)\n\n');
fprintf('Sumatoria de fuerzas en eje vertical = 0:\n');
fprintf('-F*sin(alpha) + P*cos(beta) - W = 0\n');
fprintf('--> -P*sin(beta)/cos(alpha)*sin(alpha) + P*cos(beta) - W = 0\n');
fprintf('--> -P*sin(beta)*tan(alpha) + P*cos(beta) - W = 0\n');
fprintf('--> P*(cos(beta) - sin(beta)*tan(alpha)) - W = 0\n\n');
fprintf('Solución:\n');
P = W/(cos(beta) - sin(beta)*tan(alpha));
fprintf('P = W/(cos(beta) - sin(beta)*tan(alpha)) = %f lbf\n',P);
fprintf('--> F = P*sin(beta)/cos(alpha) = %f
lbf\n\n',P*sin(beta)/cos(alpha));
```

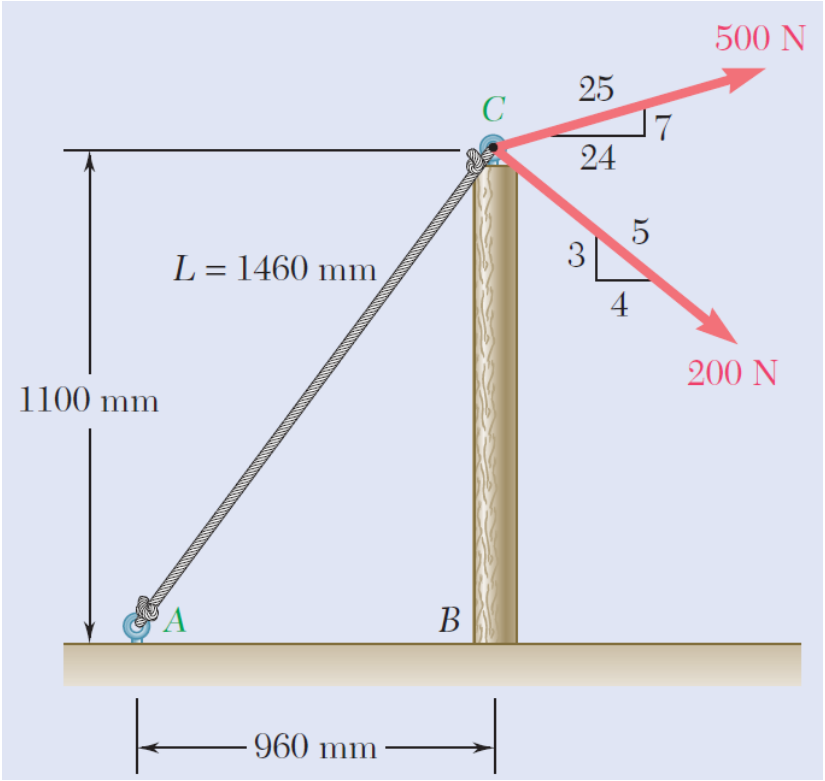
Equilibrio en el Plano




```
Equilibrio en el plano
----------------------

Datos:
W = 1200.000000 lbf
alpha = 0.349066 rad
beta = 0.087266 rad

Sumatoria de fuerzas en eje horizontal = 0:
F*cos(alpha) - P*sin(beta) = 0
--> F = P*sin(beta)/cos(alpha)

Sumatoria de fuerzas en eje vertical = 0:
-F*sin(alpha) + P*cos(beta) - W = 0
--> -P*sin(beta)/cos(alpha)*sin(alpha) + P*cos(beta) - W = 0
--> -P*sin(beta)*tan(alpha) + P*cos(beta) - W = 0
-->  P*(cos(beta) - sin(beta)*tan(alpha)) - W = 0

Solución:
P = W/(cos(beta) - sin(beta)*tan(alpha)) = 1244.203306 lbf
-->  F = P*sin(beta)/cos(alpha) = 115.398866 lbf
```

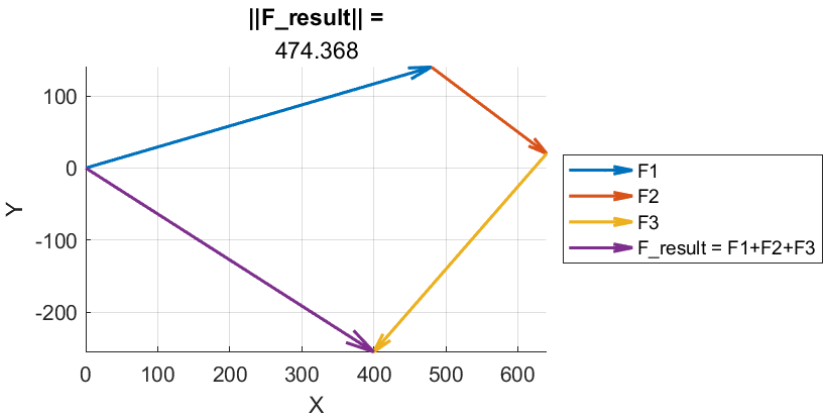


Ley del polígono muestra que la resultante sobre el punto de análisis es 0.



Ejemplo: La tensión en la cuerda AC es 365 N. (a) Determinar la resultante de las tres fuerzas que se ejercen en el punto C del poste BC; (b) determinar la fuerza axial y cortante en el poste BC que permiten el equilibrio del punto C.

Equilibrio en el Plano



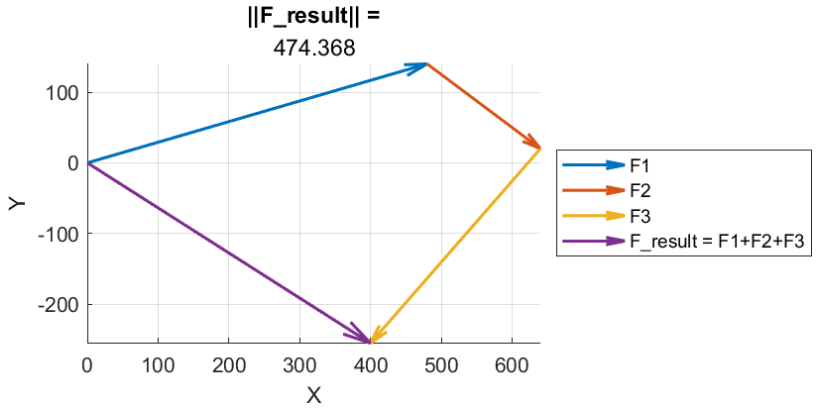

```
Datos:
||F1|| = 500.000000 lbf
||F2|| = 200.000000 lbf
||F3|| = 365.000000 lbf
Magnitud resultante: ||F_result|| = 474.368001 lbf
Ángulo resultante: alpha = -32.517511°
```

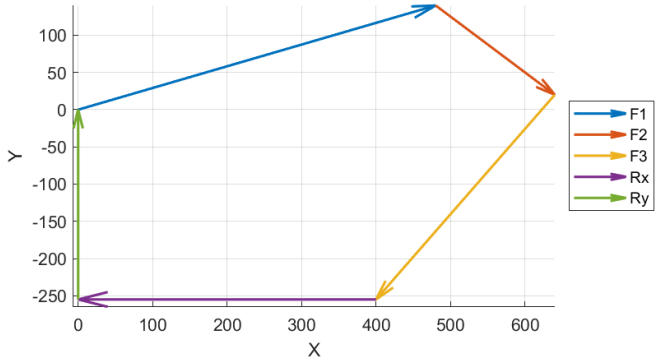
La reacciones equilibrantes están dadas por las componentes de la fuerza que se opone a la resultante

```
Reacciones equilibrantes sobre el poste:
Rx = -400.000000 lbf
Ry = 255.000000 lbf
```

```
%-------------------------------------------------------------------------
%                          Código MATLAB
%-------------------------------------------------------------------------
F1_mag = 500; F1x = F1_mag*24/25; F1y = F1_mag*7/25;
F2_mag = 200; F2x = F2_mag*4/5; F2y = -F2_mag*3/5;
F3_mag = 365; F3x = -F3_mag*960/1460; F3y = -F3_mag*1100/1460;
% plotea vector F1
figure; % define una nueva figura
x0 = [0,0,0]; % define origen del vector
plotvec3d(x0,[F1x,F1y,0],'LineWidth',1.5,'MaxHeadSize',0.2);
hold on; view(2);
% plotea vector F2 a continuación del vector F1
x0 = [F1x,F1y,0]; % define origen del vector
plotvec3d(x0,[F2x,F2y,0],'LineWidth',1.5,'MaxHeadSize',0.5);
% plotea vector F3 a continuación del vector F1 + F2
x0 = [F1x,F1y,0]+[F2x,F2y,0]; % define el origen del vector
plotvec3d(x0,[F3x,F3y,0],'LineWidth',1.5,'MaxHeadSize',0.3);
legend('F1','F2','F3','Location','eastoutside');
% plotea vector resultante desde [0,0,0] hasta donde termina F1+F2+F3
x0 = [0,0,0]; % define origen del vector
F_result = [F1x,F1y,0]+[F2x,F2y,0]+[F3x,F3y,0];
F_result_x = F_result(1); F_result_y = F_result(2);
plotvec3d(x0,F_result,'LineWidth',1.5,'MaxHeadSize',0.3);
legend('F1','F2','F3','F\_result = F1+F2+F3','Location','eastoutside');
title('||F\_result|| =',norm(F_result));
fprintf('Magnitud resultante: ||F_result|| = %f lbf\n',norm(F_result));
fprintf('Ángulo resultante: alpha =%f°\n',...
atan(F_result_y/F_result_x)*180/pi());
Rx = -F_result_x; Ry = -F_result_y;
fprintf('Reacciones equilibrantes sobre el poste:\n');
fprintf('Rx = %f lbf\n',Rx); fprintf('Ry = %f lbf\n',Ry);
```

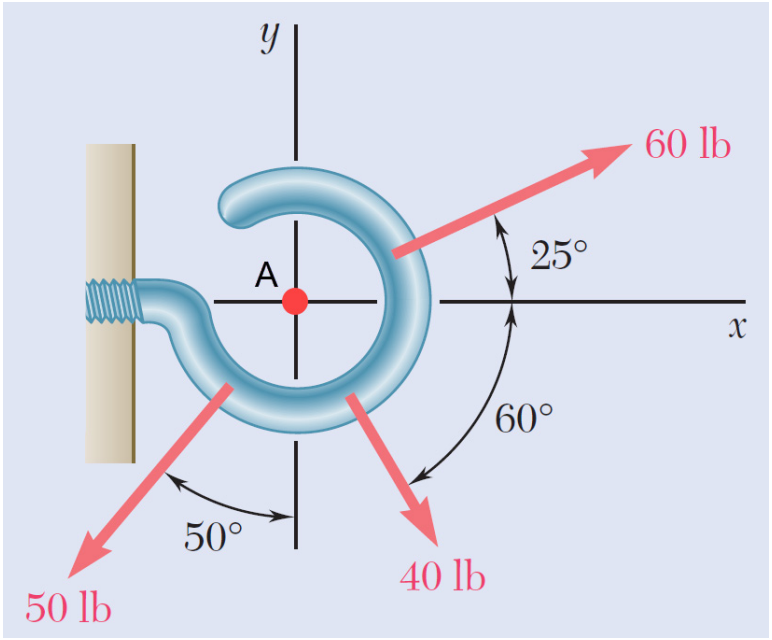
Equilibrio por método gráfico (ley del polígono)






```
%-------------------------------------------------------------------------
%                              Código MATLAB
%-------------------------------------------------------------------------
F1_mag = 500; F1x = F1_mag*24/25; F1y = F1_mag*7/25;
F2_mag = 200; F2x = F2_mag*4/5; F2y = -F2_mag*3/5;
F3_mag = 365; F3x = -F3_mag*960/1460; F3y = -F3_mag*1100/1460;
F_result = [F1x,F1y,0]+[F2x,F2y,0]+[F3x,F3y,0];
F_result_x = F_result(1); F_result_y = F_result(2);
Rx = -F_result_x; Ry = -F_result_y;
% plotea suma de vectores en equilibrio: F1 + F2 + F3 - F_result
figure; % define una nueva figura
x0 = [0,0,0]; % define el origen del vector
plotvec3d(x0,[F1x,F1y,0],'LineWidth',1.5,'MaxHeadSize',0.2); hold on;
view(2); % vista frontal por defecto para caso 2D
%
x0 = [F1x,F1y,0]; % define el origen del vector
plotvec3d(x0,[F2x,F2y,0],'LineWidth',1.5,'MaxHeadSize',0.5);
x0 = [F1x,F1y,0]+[F2x,F2y,0]; % define el origen del vector
plotvec3d(x0,[F3x,F3y,0],'LineWidth',1.5,'MaxHeadSize',0.3);
legend('F1','F2','F3','Location','eastoutside');
%
x0 = [F1x,F1y,0]+[F2x,F2y,0]+[F3x,F3y,0]; % define el origen del vector
plotvec3d(x0,[Rx,0,0],'LineWidth',1.5,'MaxHeadSize',0.3);
x0 = [F1x,F1y,0]+[F2x,F2y,0]+[F3x,F3y,0]+[Rx,0,0]; % define origen vector
plotvec3d(x0,[0,Ry,0],'LineWidth',1.5,'MaxHeadSize',0.3);
legend('F1','F2','F3','Rx','Ry','Location','eastoutside');
```

Problema 1: Equilibrio en el plano



Todas las fuerzas son concurrentes al punto A.

Programar una función en MATLAB para resolver:

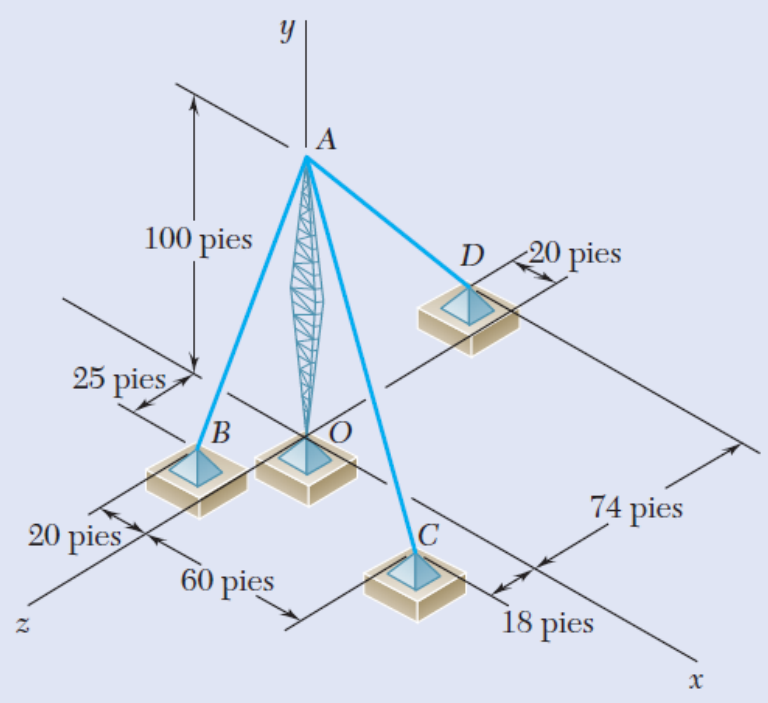
a) La fuerza resultante ejercida por las tres fuerzas en el punto A.

b) Las reacciones equilibrantes del Sistema.

Instrucciones:

- Tarea en grupos de máximo 3 integrantes.
- Presentar resultados en slides de PowerPoint (puede ser slides en PDF).
- Adjuntar código MATLAB.
- Enviar por sistema de tareas de u-cursos

Problema 2: Equilibrio en el espacio



Torre de transmisión sostenida mediante tres alambres unidos a un pasador en A y anclados mediante pernos en B, C y D. Se sabe que la tensión en el alambre AC es 590 lbf.

Programar una función en MATLAB para determinar la fuerza vertical P ejercida por la torre sobre el pasador en A.

Instrucciones:

- Tarea en grupos de máximo 3 integrantes.
- Presentar resultados en slides de PowerPoint (puede ser slides en PDF).
- Adjuntar código MATLAB.
- Enviar por sistema de tareas de u-cursos