

Dot (or Scalar) Product for Vectors

```
> restart:with(plots):with(LinearAlgebra):  
> setoptions3d(axes=NORMAL,labels=["x","y","z"],orientation=[20,  
70]);
```

Although we are using the [LinearAlgebra](#) package here, we could get much the same functionality from the [VectorCalculus](#) package. Since we are using the **LinearAlgebra** package, we do not need to use the **BasisFormat** statement to get vectors in row or column form. Although examples are given in three dimensions, the same commands work in two dimensions. We enter two column vectors abstractly.

Dot Product

```
> V:=Vector([v[1],v[2],v[3]]);W:=<w[1],w[2],w[3]>;
```

$$V := \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$W := \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

We enter two row vectors abstractly.

```
> X:=Vector[row]([x[1],x[2],x[3]]);Z:=<z[1]|z[2]|z[3]>;
```

$$X := \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

$$Z := \begin{bmatrix} z_1 & z_2 & z_3 \end{bmatrix}$$

To find the **dot** or **scalar product** of two vectors, we use the **DotProduct** command from the **LinearAlgebra** package. As seen below, it makes no difference if you use two column vectors, two row vectors, or one of each.

```
> DotProduct(V,W);
```

$$\overline{v_1} w_1 + \overline{v_2} w_2 + \overline{v_3} w_3$$

```
> DotProduct(X,Z);
```

$$\overline{z_1} x_1 + \overline{z_2} x_2 + \overline{z_3} x_3$$

```
> DotProduct(Z,V);
```

$$\overline{v_1} z_1 + \overline{v_2} z_2 + \overline{v_3} z_3$$

The bar over some components indicate complex conjugates, Maple assumes all components of vectors are complex numbers. Thus the definition here differs from that of the text. However, if all the components are real, their conjugates are just the real numbers themselves, matching up with the formula in the text. We can use the [assume](#) command to make all of our components real. For instance:

```
> assume(v[1],real);assume(v[2],real);assume(v[3],real);assume(w  
[1],real);assume(w[2],real);assume(w[3],real);
```

We reenter the vectors.

```
> V:=Vector([v[1],v[2],v[3]]);W:=Vector([w[1],w[2],w[3]]);
```

$$V := \begin{bmatrix} v_{\sim 1} \\ v_{\sim 2} \\ v_{\sim 3} \end{bmatrix}$$

$$W := \begin{bmatrix} w_{\sim 1} \\ w_{\sim 2} \\ w_{\sim 3} \end{bmatrix}$$

The "~" after each component indicates that there is an assumption on the variable. We recompute the dot product.

```
> DotProduct(V,W);
```

$$v_{\sim 1} w_{\sim 1} + v_{\sim 2} w_{\sim 2} + v_{\sim 3} w_{\sim 3}$$

Before going on, we remove the assumptions from the components.

```
> V:='V';W:='W';
```

$$V := V$$

$$W := W$$

We now look at a particular example.

```
> v:=Vector([1,2,3]);w:=Vector([-3,5,7]);
```

$$v := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$w := \begin{bmatrix} -3 \\ 5 \\ 7 \end{bmatrix}$$

```
> v_dot_w:=DotProduct(v,w);
```

$$v_dot_w := 28$$

This command can be abbreviated as follows.

```
> v_dot_w:=v.w;
```

$$v_dot_w := 28$$

We next find the angle between the two vectors.

```
> theta:=VectorAngle(v,w);
```

$$\theta := \arccos\left(\frac{2}{83} \sqrt{14} \sqrt{83}\right)$$

Using the angle, we find the dot product using the **geometric definition**.

```
> v_dot_w:=Norm(v,2)*Norm(w,2)*cos(theta);
```

$$v_dot_w := 28$$

Components and Projections

Finally, we resolve a planar vector **a** into components parallel to and perpendicular to a nonzero vector

b. But we need the **VectorCalculus** package for the graphics. We begin with vectors **a** and **b**.

```
> restart:with(plots):with(VectorCalculus):BasisFormat(false):
```

We begin with vectors **a** and **b**.

```
> a:=Vector([4,3]);b:=Vector([3,1]);
```

$$a := \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

$$b := \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

First, we find the **component of a along b**.

```
> compaparb:=DotProduct(a,b)/Norm(b,2);
```

$$\text{compaparb} := \frac{3}{2} \sqrt{10}$$

We next find the **projection of a onto b**.

```
> aparallel:=compaparb*b/Norm(b,2);
```

$$\text{apara} := \begin{bmatrix} \frac{9}{2} \\ \frac{3}{2} \end{bmatrix}$$

Then we find the projection of **a** perpendicular to **b**.

```
> aperp:=a-apara;
```

$$\text{aperp} := \begin{bmatrix} -\frac{1}{2} \\ \frac{3}{2} \end{bmatrix}$$

We see that the two components do indeed sum to **a**.

```
> a:=apara+aperp;
```

$$a := \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

We view the resolution graphically.

```
> p1:=PlotVector(a,width=.1,head_width=.2,head_length=.2,scaling=
constrained,color=red):
p2:=PlotVector(b,width=.05,head_width=.2,head_length=.2,scaling=
constrained,color=green):
p3:=PlotVector(apara,width=.1,head_width=.2,head_length=.2,
scaling=constrained,color=blue):
p4:=PlotVector(apara,aperp,width=.1,head_width=.2,
head_length=.2,scaling=constrained,color=blue):
p5:=textplot([2,2,"a"],font=[TIMES,BOLD,14],color=red):
p6:=textplot([2,.3,"b"],font=[TIMES,BOLD,14],color=green):
p7:=textplot([4,.8,"Proj[b]a"],[5.5,2,"Proj[bperp]a"],font=
[TIMES,BOLD,14],color=blue):
display(p1,p2,p3,p4,p5,p6,p7);
```

