In [1]:

```
# an ellipse given in parametric form
# that the ellipse is titled and shifted
# with centre at (x=c1=3, y=c2=0)
# and major radius r1=2
# and minor radius r2=1
# and titled by angle alpha=pi/4
c1=3
c2=0
r1=2
r2=1
alpha=pi/4
x(t)=r1*cos(alpha)*cos(t)-r2*sin(alpha)*sin(t)+c1
y(t)=r1*sin(alpha)*cos(t)+r2*cos(alpha)*sin(t)+c2
```

In [2]:

```
# define a fixed point P at (ptx=3, pty=2)
#
ptx=3
pty=2
```

In [3]:

```
# define the square distance DS(t) between the point
# and the ellipse with parameter t
#
DS(t)=(x(t)-ptx)^2+(y(t)-pty)^2
```

In [4]:

```
# to find the closest point on the ellipse to point P,
# we find t minimizing DS
# thus, we differentiate DS w.r.t. t,
# let it equals zero, and solve for t
#
show(solve(diff(DS(t),t)==0,t))
```

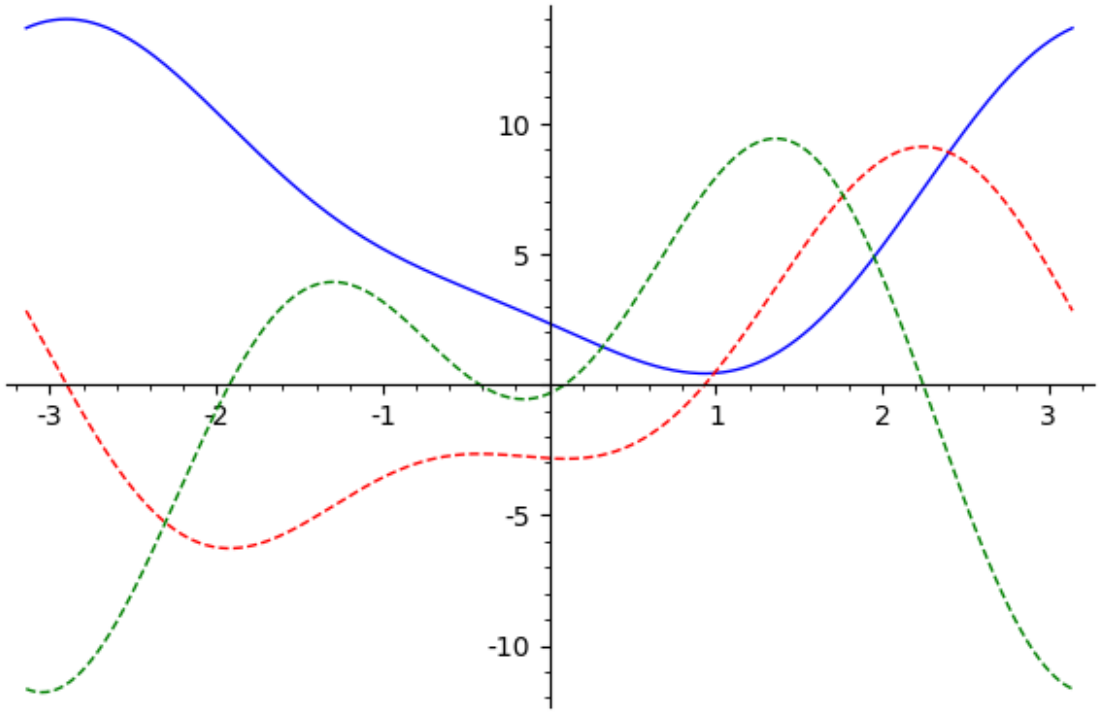
Out[4]:

$$\left[\sin(t) = -\frac{2\,\cos(t)}{3\,\sqrt{2}\cos(t) - 4}\right]$$

In [5]:

```
# since CoCalc cannot show analytic exact answer for t
# we need to find t numerically
#
p1=plot(DS(t),t,-pi,pi)
p2=plot(diff(DS(t),t),t,-pi,pi,rgbcolor='red', linestyle = "dashed" )
p3=plot(diff(DS(t),t,2),t,-pi,pi,rgbcolor='green', linestyle = "dashed")
p1+p2+p3
```

Out[5]:



In [6]:
```
# there are two roots, one between -3 to -2, and one between 0 and 1
# we find the root t_0 between t=0 to t=1
# note that this root is minimum as the second derivative is positive
#
t0=(diff(DS(t),t)==0).find_root(0,1,t)
show(t0)
```

Out[6]:

$$0.9342955781066485$$

In [7]:
```
# we find the root t_1 between t=-3 to t=-2
# note that this root is maximum as the second derivative is negative
#
t1=(diff(DS(t),t)==0).find_root(-3,-2,t)
show(t1)
```

Out[7]:

$$-2.9000801835272654$$

In [8]:
```
# check the second derivative to see if at t_0 is positive for minimum
#
var('tt')
show(bool(diff(DS(tt),tt,2).subs(tt=t0)>0))
```

Out[8]:

$$\text{True}$$

In [9]:
```
show(RR(x(t0)))
```

Out[9]:

$$3.27194586646520$$

In [10]:
```
show(RR(y(t0)))
```

Out[10]:

$$1.40922894904544$$

In [11]:

```
show(RR(sqrt(DS(t0))))
```

Out[11]:

$$0.650357585435483$$

In [12]:

```
# find the slope of tangent line m
# m(t) = dy/dx = (dy/dt)/(dx/dt)
#
var('yy xx')
xdash(t)=diff(x(t),t)
ydash(t)=diff(y(t),t)
m(t)=ydash(t)/xdash(t)
```

In [13]:

```
# tangent line at t=t0
#
tangent(xx)=solve((yy-y(t0))/(xx-x(t0))==m(t0),yy)[0].rhs()
```

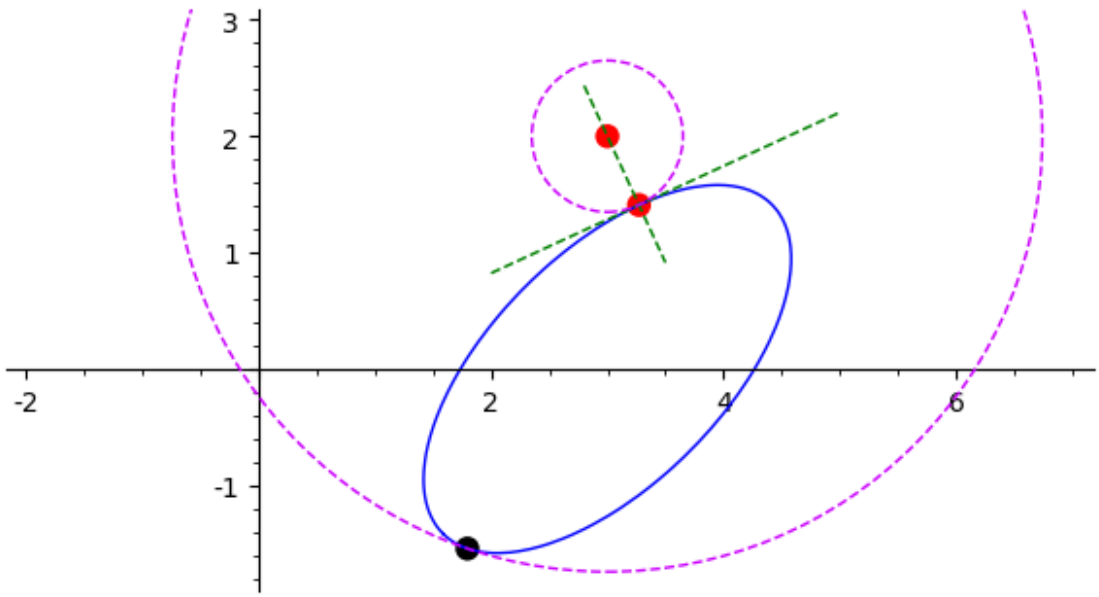
In [14]:

```
# normal line at t=t0
#
normal(xx)=solve((yy-y(t0))/(xx-x(t0))==(-1/m(t0)),yy)[0].rhs()
```

In [15]:

```
p1=parametric_plot( (x(t), y(t)), (t, -pi, pi) )
p2=plot(tangent(xx),xx,2,5, rgbcolor='green', linestyle = "dashed")
p3=plot(normal(xx),xx,2.8,3.5, rgbcolor='green', linestyle = "dashed")
pt0 = point((x(t0),y(t0)), rgbcolor='red', pointsize=80)
pt1 = point((x(t1),y(t1)), rgbcolor='black', pointsize=80)
pt00 = point((ptx,pty), rgbcolor='red', pointsize=80)
cplt0=circle((ptx,pty),sqrt(DS(t0)),color=hue(0.8), linestyle = "dashed")
cplt1=circle((ptx,pty),sqrt(DS(t1)),color=hue(0.8), linestyle = "dashed")
plotall=p1+p2+p3+pt00+pt0+pt1+cplt0+cplt1
(plotall).show(xmin=-2, xmax=7, ymin=-1.8, ymax=3,aspect_ratio=1)
```

Out[15]:



In [0]: