In [1]:

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

```
# this is a subroutine to find all roots of a func on [a,b]
# with number of steps points nsteps
#
def find_all_roots(func,a,b,nsteps):
    roots=[]
    n=0
    x0=a                #initializations
    step=(b-a)/(nsteps)     #size of a sub-interval
                    #the rest is quite self-explanatory
    while (n<nsteps):
        try:
            roots.append(find_root(func,x0,x0+step))
            root_exist = True
        except RuntimeError:
            root_exist = False
        x0+=step
        n+=1
    roots.sort()
    return roots
```

In [3]:

```
# define a set of grid points and put in A
# A(count index , x_coordinate , y-coordinate , number of roots)
#
A=matrix(QQ,100,4)
n=0
while n < 100:
 A[n,0]=n
 n=n+ 1
n=0
ii=0
jj=0
while ii < 10:
    while jj < 10:
        A[n,1] = ii
        A[n,2] = jj
        n=n+1
        jj = jj + 1
    jj = 0
    ii = ii + 1
```

In [4]:

```
# define DDS as square of the distance from point (ptx,pty)
# to the ellipse at the point with parameter t
#
DDS(ptx,pty,t)=(x(t)-ptx)^2+(y(t)-pty)^2
```

In [5]:
```
# at each n (the count index), find the number of roots when
# solving for the derivative of DS =0
#
n=0
while n < 100:
    DS(t)=DDS(A[n][1],A[n][2],t)
    roots=find_all_roots(diff(DS(t),t),-pi,pi,20)
    A[n,3]=len(roots)
    n=n+ 1
```

In [6]:
```
# count how many are with 4 roots
#
n=0
count=0
while n < 100:
    if A[n,3]==4:
        count=count+1
    n=n+ 1
#
# put these in x1 y1
#
x1=matrix(QQ,count,1)
y1=matrix(QQ,count,1)
n=0
count=0
while n < 100:
    if A[n,3]==4:
        x1[count]=A[n,1]
        y1[count]=A[n,2]
        count=count+1
    n=n+ 1
#
# count how many are with 2 roots
#
n=0
count=0
while n < 100:
    if A[n,3]==2:
        count=count+1
    n=n+ 1
#
# put these in x2 y2
#
x2=matrix(QQ,count,1)
y2=matrix(QQ,count,1)
n=0
count=0
while n < 100:
    if A[n,3]==2:
        x2[count]=A[n,1]
        y2[count]=A[n,2]
        count=count+1
    n=n+ 1
#
# count how many are with 3 roots
#
n=0
count=0
while n < 100:
    if A[n,3]==3:
        count=count+1
    n=n+ 1
#
# put these in x3 y3
```

```
#
x3=matrix(QQ,count,1)
y3=matrix(QQ,count,1)
n=0
count=0
while n < 100:
    if A[n,3]==3:
        x3[count]=A[n,1]
        y3[count]=A[n,2]
        count=count+1
    n=n+ 1
```

In [7]:

```
xxl = x1.column(0)
yyl = y1.column(0)
xx2 = x2.column(0)
yy2 = y2.column(0)
xx3 = x3.column(0)
yy3 = y3.column(0)
```

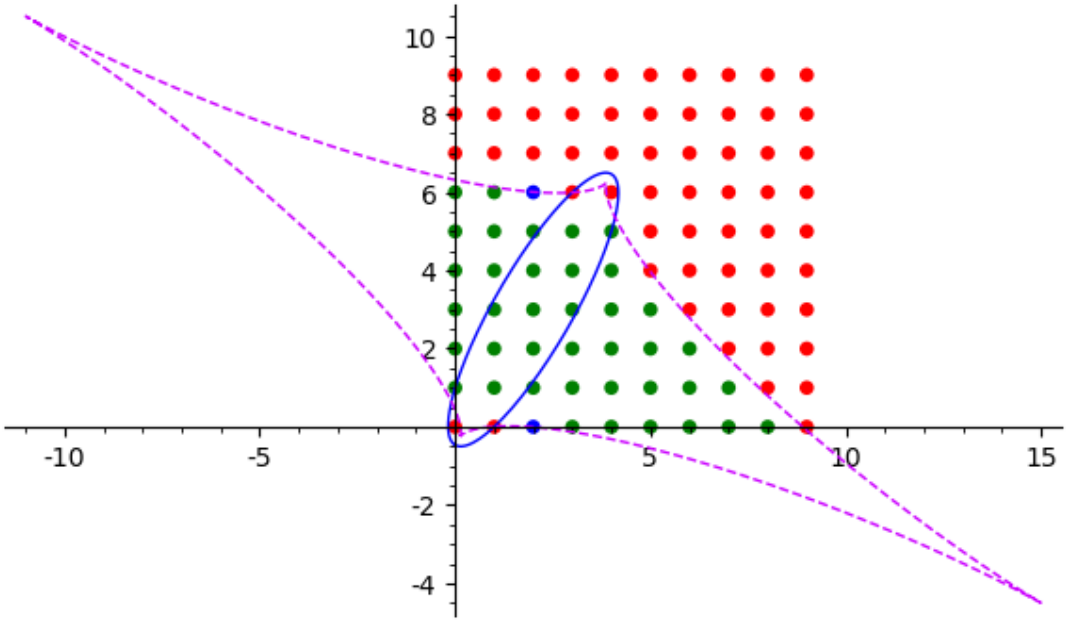
In [8]:

```
# the EVOLUTE of the ellipse
#
re1=((r1^2-r2^2)/r1)
re2=((r2^2-r1^2)/r2)
xe(t)=re1*cos(alpha)*cos(t)^3-re2*sin(alpha)*sin(t)^3+c1
ye(t)=re1*sin(alpha)*cos(t)^3+re2*cos(alpha)*sin(t)^3+c2
```

In [9]:

```
# put locations of points with 4 roots in datapoints1
datapoints1 = zip(xxl, yyl)
# put locations of points with 2 roots in datapoints2
datapoints2 = zip(xx2, yy2)
# put locations of points with 3 roots in datapoints3
datapoints3 = zip(xx3, yy3)
p1=point(datapoints1,rgbcolor='green', pointsize=30)
# points with 4 roots are in green
p2=point(datapoints2,rgbcolor='red', pointsize=30)
# points with 2 roots are in red
p3=point(datapoints3,rgbcolor='blue', pointsize=30)
# points with 3 roots are in blue
p4=parametric_plot((x(t),y(t)),(t, -pi, pi))
p5=parametric_plot((xe(t), ye(t)),(t, -pi, pi),color=hue(0.8),linestyle ="dashed")
p1+p2+p3+p4+p5
```

Out[9]:



In [0]: