

```

> restart;
> with(plottools);
[arc, arrow, circle, cone, cuboid, curve, cutin, cutout, cylinder, disk, dodecahedron, ellipse,
  ellipticArc, hemisphere, hexahedron, homothety, hyperbola, icosahedron, line, octahedron,
  parallelepiped, pieslice, point, polygon, project, rectangle, reflect, rotate, scale, semitorus,
  sphere, stellate, tetrahedron, torus, transform, translate, vrmf]

```

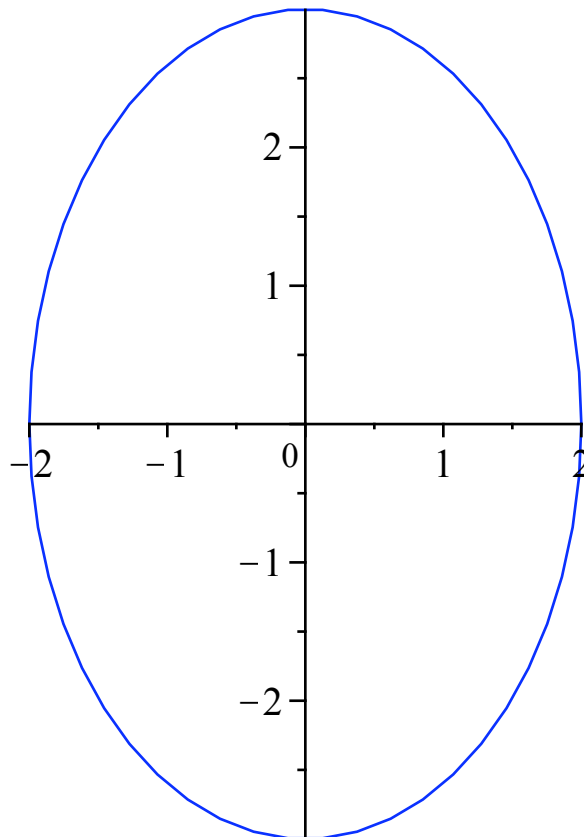
(1)

```

> with(plots);
[animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d,
  conformal, conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot,
  display, dualaxisplot, fieldplot, fieldplot3d, gradplot, gradplot3d, graphplot3d, implicitplot,
  implicitplot3d, inequal, interactive, interactiveparams, intersectplot, listcontplot,
  listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot, matrixplot, multiple,
  odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d,
  polyhedra_supported, polyhedraplot, rootlocus, semilogplot, setcolors, setoptions,
  setoptions3d, spacecurve, sparsematrixplot, surfdata, textplot, textplot3d, tubeplot]
> elli:= ellipse([0,0], 2, 3,color=blue):
  display(elli, scaling=constrained);

```

(2)



```

> ParEllipse:=proc(c,d,t)
  [c[1]+d[1]*(t+1/t)/2,c[2]+d[2]*(t-1/t)/(2*I)]
end;
ParEllipse:=proc(c,d,t)
  [c[1]+1/2*d[1]*(t+1/t),c[2]-1/2*I*d[2]*(t-1/t)]
end proc

```

(3)

```

  ParEllipse:=proc(c,d,t) [c[1]+d[1]*cos(t),c[2]+d[2]*sin(t)] end proc
> TangentToEllipse:=proc(c,d,t)

```

(4)

```

local P;
P:=ParEllipse(c,d,t);
diff(P[1],t)*(y-P[2])-diff(P[2],t)*(x-P[1])
end;

```

TangentToEllipse:=proc(c,d,t)

```

local P;
P:=ParEllipse(c,d,t); (diff(P[1],t))*(y-P[2])-(diff(P[2],t))*(x-P[1])

```

end proc

```

> Pt:=proc(L1,L2)
local q;
q:=solve({L1,L2},{x,y}):
normal([subs(q,x),subs(q,y)])
end;
> PointsBrianchon:=proc(c,d) local L,t,i,P;
for i from 0 to 5 do L[i]:=TangentToEllipse(c,d,t[i]): od;
for i from 0 to 5 do P[i]:=Pt(L[i],L[i+1 mod 6]): od;
unapply([seq(P[i],i=0..5),P[0]],seq(t[i],i=0..5));
end;
> PB:=PointsBrianchon([0,0],[2,3]);

```

$PB := (t_0, t_1, t_2, t_3, t_4, t_5) \rightarrow \left[\left[\frac{2(t_0 t_1 + 1)}{t_1 + t_0}, -\frac{3I(t_0 t_1 - 1)}{t_1 + t_0} \right], \right.$

$\left[\frac{2(t_1 t_2 + 1)}{t_2 + t_1}, -\frac{3I(t_1 t_2 - 1)}{t_2 + t_1} \right], \left[\frac{2(t_2 t_3 + 1)}{t_3 + t_2}, -\frac{3I(t_2 t_3 - 1)}{t_3 + t_2} \right],$

$\left[\frac{2(t_3 t_4 + 1)}{t_4 + t_3}, -\frac{3I(t_3 t_4 - 1)}{t_4 + t_3} \right], \left[\frac{2(t_4 t_5 + 1)}{t_5 + t_4}, -\frac{3I(t_4 t_5 - 1)}{t_5 + t_4} \right],$

$\left. \left[\frac{2(t_0 t_5 + 1)}{t_5 + t_0}, -\frac{3I(t_0 t_5 - 1)}{t_5 + t_0} \right], \left[\frac{2(t_0 t_1 + 1)}{t_1 + t_0}, -\frac{3I(t_0 t_1 - 1)}{t_1 + t_0} \right] \right]$

```

> h:=rand(0..1000);

```

h:=proc()

```

proc() option builtin=RandNumberInterface; end proc(6, 1001, 10)

```

end proc

```

> choix:=proc()
local T,S,i;
for i from 0 to 5 do T[i]:=evalf(h()/1000*2*Pi) od;
S:=sort([seq(T[i],i=0..5)]);
print(S);
S:=seq(exp(I*S[i]),i=1..6);
T:=PB(S);
[seq(map(Re,T[i]),i=1..7)]
end;

```

choix:=proc()

```

local T,S,i;
for i from 0 to 5 do T[i]:=evalf(1/500*h()*Pi) end do;
S:=sort([seq(T[i],i=0..5)]);
print(S);
S:=seq(exp(S[i]*I),i=1..6);
T:=PB(S);
[seq(map(Re,T[i]),i=1..7)]

```

end proc

```

> Points:=choix();
[0.9361946109, 1.413716694, 1.966637001, 2.990796207, 3.462035105, 5.491503959]
Points:= [[0.7936834076, 2.848834725], [-.2475961212, 3.096218438], [-1.808422235,

```

```
2.117816988], [-2.049435436, -.2613860872], [-.8845215426, -5.526446083],  
[3.071502416, 0.3334852067], [0.7936834076, 2.848834725]]
```

```
> A:=plot(Points);
```

```
A:=PLOT(...)
```

(10)

```
> B1:=plot([Points[1],Points[4]],color=green);
```

```
B2:=plot([Points[2],Points[5]],color=green);
```

```
B3:=plot([Points[3],Points[6]],color=green);
```

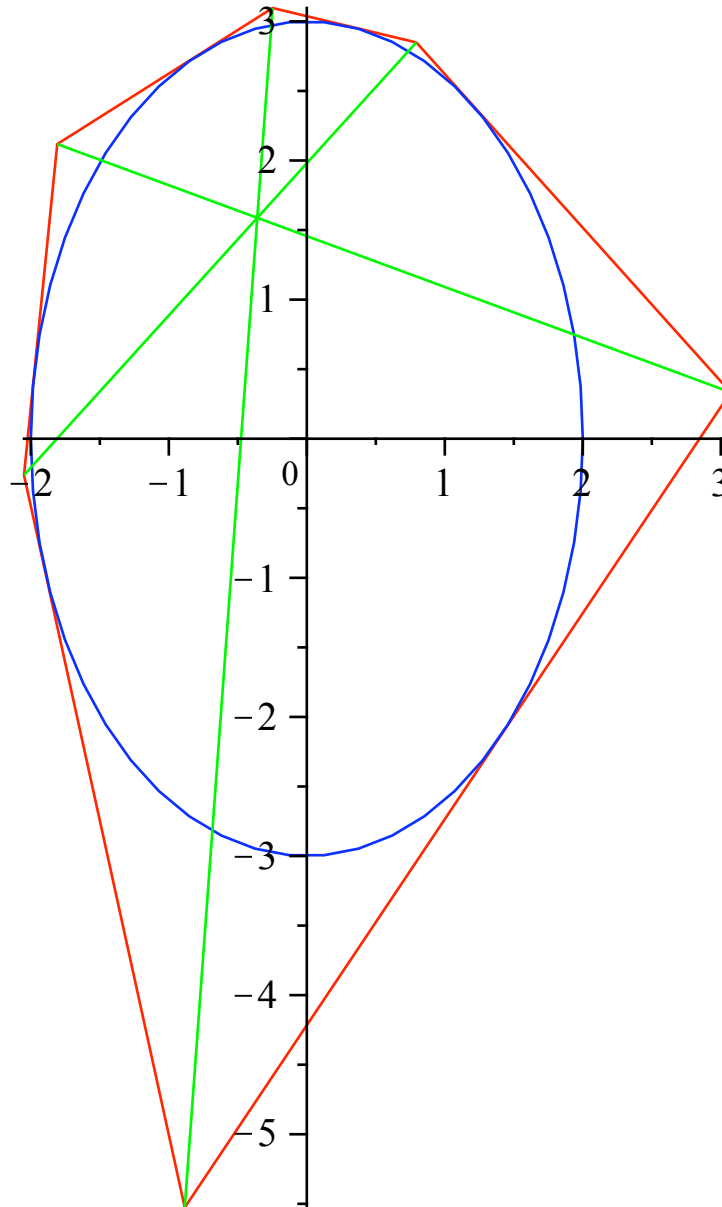
```
B1:=PLOT(...)
```

(11)

```
B2:=PLOT(...)
```

```
B3:=PLOT(...)
```

```
> display(A,elli,B1,B2,B3,scaling=constrained);
```



```
> AREA:=proc(A,B,C):
```

```
normal(expand( (B[1]*C[2]-B[2]*C[1]-A[1]*C[2]+A[2]*C[1] -B[1]*  
A[2]+B[2]*A[1])/2 ))):
```

```

end:
> Le:=proc(A,B)
  AREA(A,B,[x,y]):
end:
>
>
>
>
>
>
> Concurrent:=proc():
  not evalb(solve({args},{x,y})=NULL):
end:
> Brianchon:=proc() local L,t,i,c,d,P:
  for i from 0 to 5 do L[i]:= TangentToEllipse(c,d,t[i]): od:
  for i from 0 to 5 do P[i]:= Pt (L[i],L[i+1 mod 6]): od:
  Concurrent ( Le (P[0],P[3]) , Le (P[1],P[4]) , Le (P[2],
P[5]) ):
end:
> Brianchon();

```

true

(12)