Visualisasi dan Perhitungan Geometri dengan EMT

Euler menyediakan beberapa fungsi untuk melakukan visualisasi dan perhitungan geometri, baik secara numerik maupun analitik (seperti biasanya tentunya, menggunakan Maxima). Fungsi-fungsi untuk visualisasi dan perhitungan geometeri tersebut disimpan di dalam file program "geometry.e", sehingga file tersebut harus dipanggil sebelum menggunakan fungsi-fungsi atau perintah-perintah untuk geometri.

```
>load geometry
```

```
    Numerical and symbolic geometry.
```

Fungsi-fungsi Geometri

Fungsi-fungsi untuk Menggambar Objek Geometri:

```
defaultd:=textheight()*1.5: nilai asli untuk parameter d
setPlotrange(x1,x2,y1,y2): menentukan rentang x dan y pada bidang koordinat
setPlotRange(r): pusat bidang koordinat (0,0) dan batas-batas sumbu-x dan y adalah -r sd r
plotPoint (P, "P"): menggambar titik P dan diberi label "P"
plotSegment (A,B, "AB", d): menggambar ruas garis AB, diberi label "AB" sejauh d
plotLine (g, "g", d): menggambar garis g diberi label "g" sejauh d
plotCircle (c,"c",v,d): Menggambar lingkaran c dan diberi label "c"
plotLabel (label, P, V, d): menuliskan label pada posisi P
```

Fungsi-fungsi Geometri Analitik (numerik maupun simbolik):

```
turn(v, phi): memutar vektor v sejauh phi
turnLeft(v):   memutar vektor v ke kiri
turnRight(v):  memutar vektor v ke kanan
normalize(v): normal vektor v
crossProduct(v, w): hasil kali silang vektorv dan w.
lineThrough(A, B): garis melalui A dan B, hasilnya [a,b,c] sdh. ax+by=c.
lineWithDirection(A,v): garis melalui A searah vektor v
getLineDirection(g): vektor arah (gradien) garis g
getNormal(g): vektor normal (tegak lurus) garis g
getPointOnLine(g):  titik pada garis g
perpendicular(A, g):  garis melalui A tegak lurus garis g
parallel (A, g):  garis melalui A sejajar garis g
lineIntersection(g, h):  titik potong garis g dan h
projectToLine(A, g):   proyeksi titik A pada garis g
distance(A, B):  jarak titik A dan B
distanceSquared(A, B):  kuadrat jarak A dan B
quadrance(A, B): kuadrat jarak A dan B
areaTriangle(A, B, C):  luas segitiga ABC
computeAngle(A, B, C):   besar sudut <ABC
angleBisector(A, B, C): garis bagi sudut <ABC
circleWithCenter (A, r): lingkaran dengan pusat A dan jari-jari r
getCircleCenter(c):  pusat lingkaran c
getCircleRadius(c):  jari-jari lingkaran c
circleThrough(A,B,C):  lingkaran melalui A, B, C
middlePerpendicular(A, B): titik tengah AB
lineCircleIntersections(g, c): titik potong garis g dan lingkran c
circleCircleIntersections (c1, c2):  titik potong lingkaran c1 dan c2
planeThrough(A, B, C):  bidang melalui titik A, B, C
```

Fungsi-fungsi Khusus Untuk Geometri Simbolik:

```
getLineEquation (g,x,y): persamaan garis g dinyatakan dalam x dan y
getHesseForm (g,x,y,A): bentuk Hesse garis g dinyatakan dalam x dan y dengan titik A pada
sisi positif (kanan/atas) garis
quad(A,B): kuadrat jarak AB
spread(a,b,c): Spread segitiga dengan panjang sisi-sisi a,b,c, yakni sin(alpha)^2 dengan
alpha sudut yang menghadap sisi a.
crosslaw(a,b,c,sa): persamaan 3 quads dan 1 spread pada segitiga dengan panjang sisi a, b, c
triplespread(sa,sb,sc): persamaan 3 spread sa,sb,sc yang memebntuk suatu segitiga
doublespread(sa): Spread sudut rangkap Spread 2*phi, dengan sa=sin(phi)^2 spread a.
```

Contoh 1: Luas, Lingkaran Luar, Lingkaran Dalam Segitiga

Untuk menggambar objek-objek geometri, langkah pertama adalah menentukan rentang sumbu-sumbu koordinat. Semua objek geometri akan digambar pada satu bidang koordinat, sampai didefinisikan bidang koordinat yang baru.

```
>setPlotRange(-0.5,2.5,-0.5,2.5); // mendefinisikan bidang koordinat baru
```

Now set three points and plot them.

```
>A=[1,0]; plotPoint(A,"A"); // definisi dan gambar tiga titik
>B=[0,1]; plotPoint(B,"B");
>C=[2,2]; plotPoint(C,"C");
```

Then three segments.

```
>plotSegment(A,B,"c"); // c=AB
>plotSegment(B,C,"a"); // a=BC
>plotSegment(A,C,"b"); // b=AC
```

The geometry functions include functions to create lines and circles. The format for the line is [a,b,c], which represents the line with equation ax+by=c.

```
>lineThrough(B,C) // garis yang melalui B dan C
```

```
  [-1,  2,  2]
```

Compute the perpendicular line through A on BC.

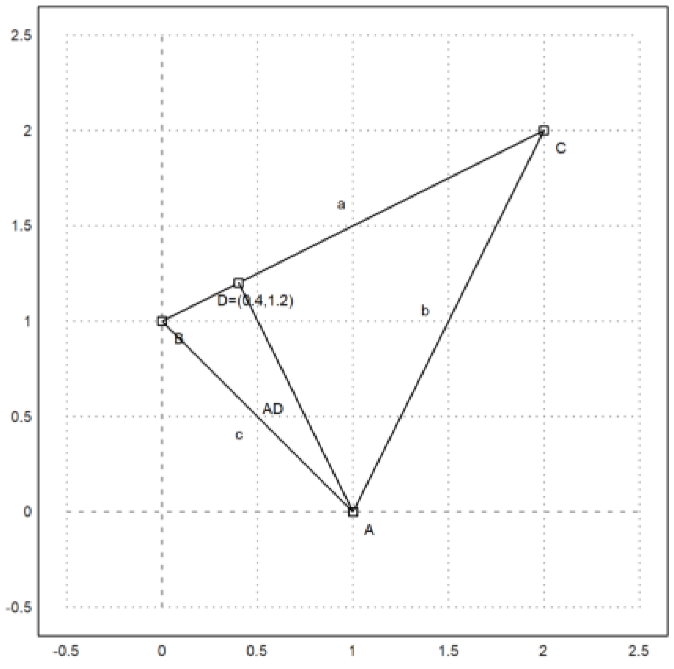
```
>h=perpendicular(A,lineThrough(B,C)); // garis h tegak lurus BC melalui A
```

And its intersection with BC.

```
>D=lineIntersection(h,lineThrough(B,C)); // D adalah titik potong h dan BC
```

Plot that.

```
>plotPoint(D,value=1); // koordinat D ditampilkan
>aspect(1); plotSegment(A,D): // tampilkan semua gambar hasil plot...()
```



Hitung luas ABC:

```
>norm(A-D)*norm(B-C)/2 // AD=norm(A-D), BC=norm(B-C)
```

```
  1.5
```

Compare with determinant formula.

```
>areaTriangle(A,B,C) // hitung luas segitiga langusng dengan fungsi
```

```
1.5
```

Cara lain menghitung luas segitigas ABC:

```
>distance(A,D)*distance(B,C)/2
```

```
1.5
```

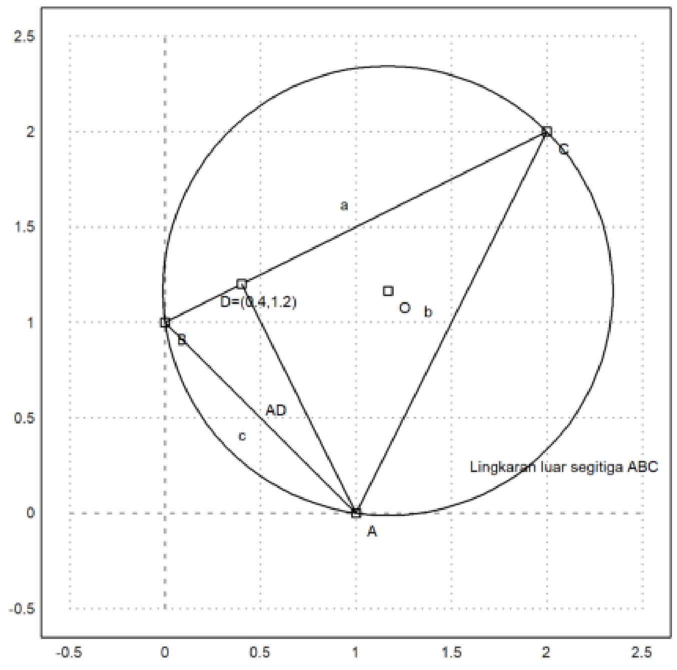
The angle at C.

```
>degprint(computeAngle(B,C,A))
```

```
36°52'11.63''
```

Now the circumcircle of the triangle.

```
>c=circleThrough(A,B,C); // lingkaran luar segitiga ABC
>R=getCircleRadius(c); // jari2 lingkaran luar
>O=getCircleCenter(c); // titik pusat lingkaran c
>plotPoint(O,"O"); // gambar titik "O"
>plotCircle(c,"Lingkaran luar segitiga ABC"):
```



Tampilkan koordinat titik pusat dan jari-jari lingkaran luar.

```
>O, R
```

```
[1.16667,  1.16667]
1.17851130198
```

Sekarang akan digambar lingkaran dalam segitiga ABC. Titik pusat lingkaran dalam adalah titik potong garis-garis bagi sudut.

```
>l=angleBisector(A,C,B); // garis bagi <ACB
>g=angleBisector(C,A,B); // garis bagi <CAB
>P=lineIntersection(l,g) // titik potong kedua garis bagi sudut
```

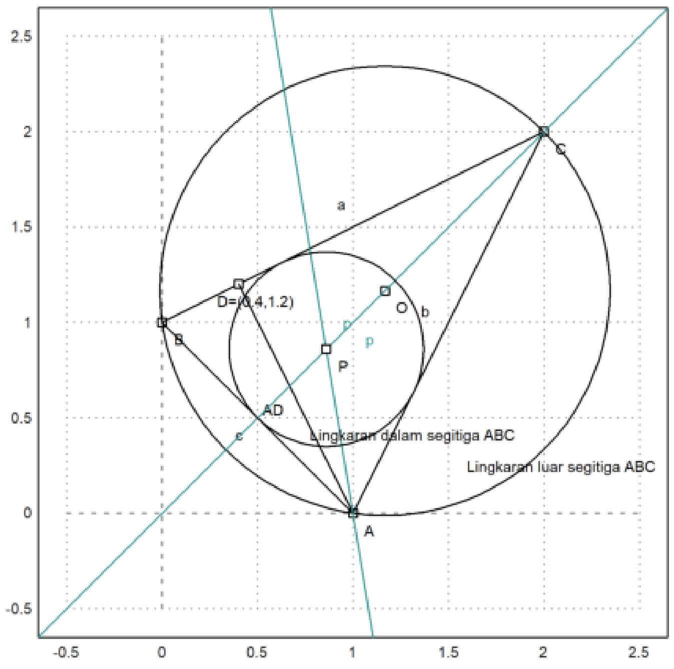
```
[0.86038,  0.86038]
```

Add everything to the plot.

```
>color(5); plotLine(l); plotLine(g); color(1); // gambar kedua garis bagi sudut
>plotPoint(P,"P"); // gambar titik potongnya
>r=norm(P-projectToLine(P,lineThrough(A,B))) // jari-jari lingkaran dalam
```

```
    0.509653732104
```

```
>plotCircle(circleWithCenter(P,r),"Lingkaran dalam segitiga ABC"): // gambar lingkaran dalam
```



Latihan

1. Tentukan ketiga titik singgung lingkaran dalam dengan sisi-sisi segitiga ABC.
2. Gambar segitiga dengan titik-titik sudut ketiga titik singgung tersebut. Merupakan segitiga apakah itu?
3. Hitung luas segitiga tersebut.
4. Tunjukkan bahwa garis bagi sudut yang ke tiga juga melalui titik pusat lingkaran dalam.
5. Gambar jari-jari lingkaran dalam.
6. Hitung luas lingkaran luar dan luas lingkaran dalam segitiga ABC. Adakah hubungan antara luas kedua lingkaran tersebut dengan luas segitiga ABC?

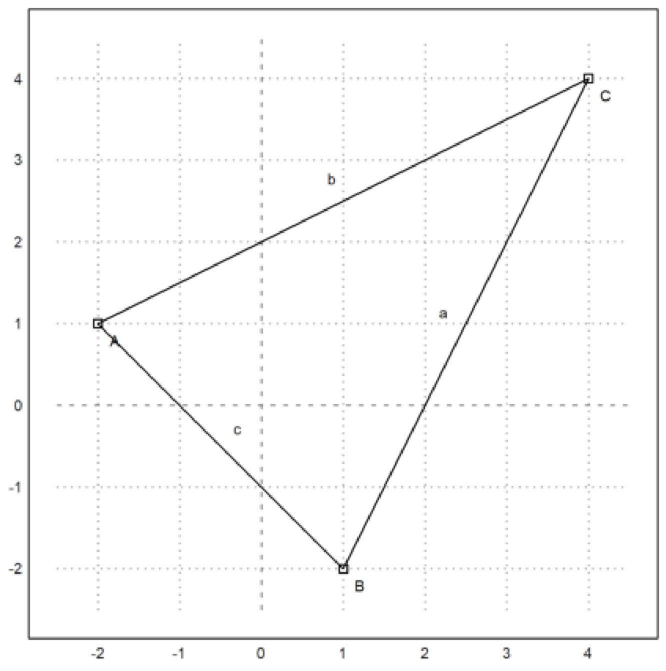
```
>load geometry
```

```
    Numerical and symbolic geometry.
```

```
>setPlotRange(-2.5,4.5,-2.5,4.5);
```

```
    Space between commands expected!
    Found:
     (character 10)
    You can disable this in the Options menu.
    Error in:
    setPlotRange(-2.5,4.5,-2.5,4.5);
     ...
                                    ^
```

```
>A=[-2,1]; plotPoint(A,"A");
>B=[1,-2]; plotPoint(B,"B");
>C=[4,4]; plotPoint(C,"C");
>plotSegment(A,B,"c")
>plotSegment(B,C,"a")
>plotSegment(A,C,"b")
>aspect(1):
```



```
>norm(A-D)*norm(B-C)/2 // AD=norm(A-D), BC=norm(B-C)
```

```
10.0623058987
```

Contoh 2: Geometri Smbolik

We can compute exact and symbolic geometry using Maxima.

The file geometry.e provides the same (and more) functions in Maxima. However, we can use symbolic computations now.

```
>A &= [1,0]; B &= [0,1]; C &= [2,2]; // menentukan tiga titik A, B, C
```

The functions for lines and circle work just like the Euler functions, but provide symbolic computations.

```
>c &= lineThrough(B,C) // c=BC
```

```
[- 1, 2, 2]
```

We can get the equation for a line easily.

```
>$getLineEquation(c,x,y), $solve(%,y) | expand // persamaan garis c
```

$$2\,y-x=2$$

$$\left[y=\frac{x}{2}+1\right]$$

```
>$getLineEquation(lineThrough([x1,y1],[x2,y2]),x,y), $solve(%,y) // persamaan garis melalui(x1, y1)
```

$$x\,\left(y_1-y_2\right)+\left(x_2-x_1\right)\,y=x_1\,\left(y_1-y_2\right)+\left(x_2-x_1\right)\,y_1$$

$$\left[y=\frac{-\left(x_1-x\right)\,y_2-\left(x-x_2\right)\,y_1}{x_2-x_1}\right]$$

```
>$getLineEquation(lineThrough(A,[x1,y1]),x,y) // persamaan garis melalui A dan (x1, y1)
```

$$\left(x_1-1\right)\,y-x\,y_1=-y_1$$

```
>h &= perpendicular(A,lineThrough(B,C)) // h melalui A tegak lurus BC
```

```
[2, 1, 2]
```

```
>Q &= lineIntersection(c,h) // Q titik potong garis c=BC dan h
```

```
                                 2  6
                                [-, -]
                                 5  5
```

```
>$projectToLine(A,lineThrough(B,C)) // proyeksi A pada BC
```

$$\left[\frac{2}{5},\frac{6}{5}\right]$$

```
>$distance(A,Q) // jarak AQ
```

$$\frac{3}{\sqrt{5}}$$

```
>cc &= circleThrough(A,B,C); $cc // (titik pusat dan jari-jari) lingkaran melalui A, B, C
```

$$\left[\frac{7}{6},\frac{7}{6},\frac{5}{3\sqrt{2}}\right]$$

```
>r&=getCircleRadius(cc); $r , $float(r) // tampilkan nilai jari-jari
```

$$\frac{5}{3\sqrt{2}}$$

$$1.178511301977579$$

```
>$computeAngle(A,C,B) // nilai <ACB
```

$$\arccos\left(\frac{4}{5}\right)$$

```
>$solve(getLineEquation(angleBisector(A,C,B),x,y),y)[1] // persamaan garis bagi <ACB
```

$$y=x$$

```
>P &= lineIntersection(angleBisector(A,C,B),angleBisector(C,B,A)); $P // titik potong 2 garis bagi s
```

$$\left[\frac{\sqrt{2}\sqrt{5}+2}{6},\frac{\sqrt{2}\sqrt{5}+2}{6}\right]$$

```
>P() // hasilnya sama dengan perhitungan sebelumnya
```

```
    [0.86038,  0.86038]
```

Intersecting Lines and Circles

Of course, we can also intersect lines with circles, and circles with circles.

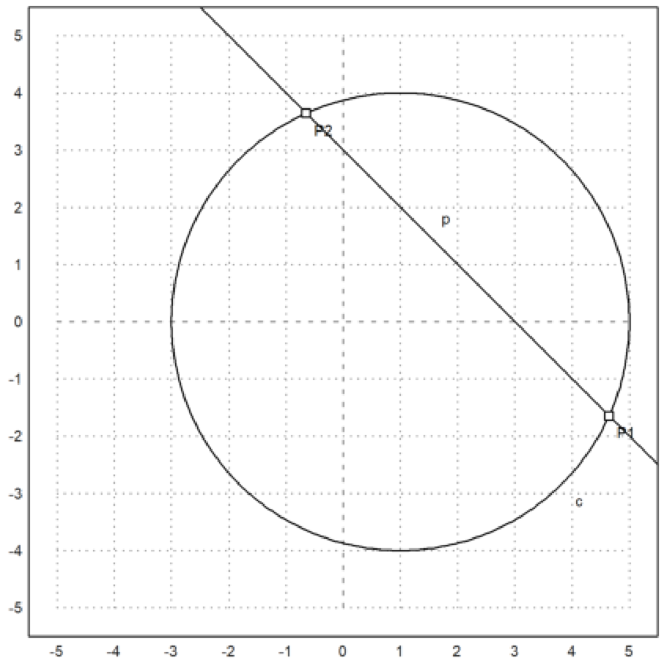
```
>A &:= [1,0]; c=circleWithCenter(A,4);
>B &:= [1,2]; C &:= [2,1]; l=lineThrough(B,C);
>setPlotRange(5); plotCircle(c); plotLine(l);
```

The intersection of a line with a circle returns two points and the number of intersection points.

```
>{P1,P2,f}=lineCircleIntersections(l,c);
>P1, P2, f
```

```
    [4.64575,  -1.64575]
    [-0.645751,  3.64575]
    2
```

```
>plotPoint(P1); plotPoint(P2):
```



The same in Maxima.

```
>c &= circleWithCenter(A,4) // lingkaran dengan pusat A jari-jari 4
```

```
                                [1, 0, 4]
```

```
>l &= lineThrough(B,C) // garis l melalui B dan C
```

```
                                [1, 1, 3]
```

```
>$lineCircleIntersections(l,c) | radcan, // titik potong lingkaran c dan garis l
```

$$\left[\left[\sqrt{7}+2,1-\sqrt{7}\right],\left[2-\sqrt{7},\sqrt{7}+1\right]\right]$$

Akan ditunjukkan bahwa sudut-sudut yang menghadap bsuusr yang sama adalah sama besar.

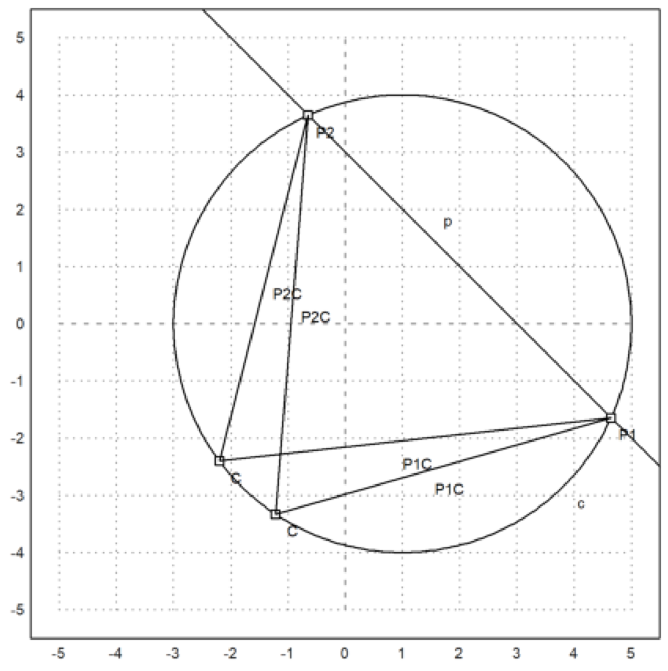
```
>C=A+normalize([-2,-3])*4; plotPoint(C); plotSegment(P1,C); plotSegment(P2,C);
>degprint(computeAngle(P1,C,P2))
```

```
    69°17'42.68''
```

```
>C=A+normalize([-4,-3])*4; plotPoint(C); plotSegment(P1,C); plotSegment(P2,C);
>degprint(computeAngle(P1,C,P2))
```

```
    69°17'42.68''
```

```
>insimg;
```

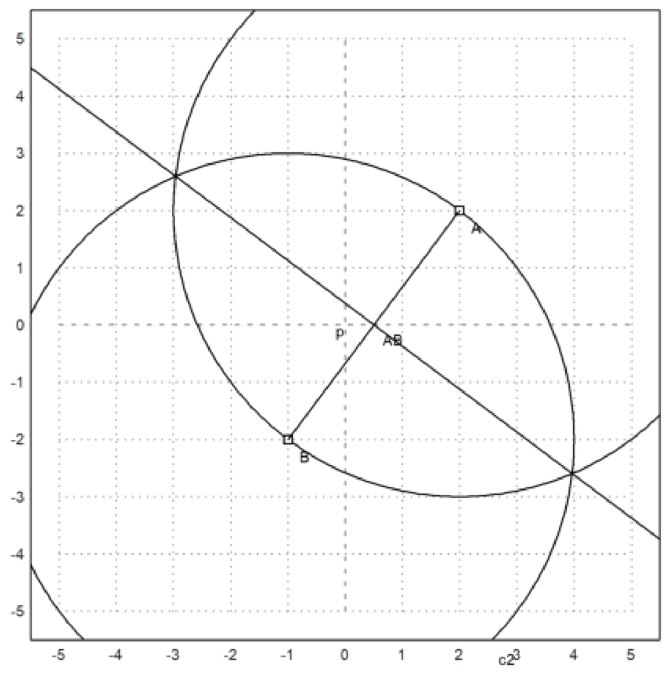


Garis Sumbu

Berikut adalah langkah-langkah menggambar garis sumbu ruas garis AB:

1. Gambar lingkaran dengan pusat A melalui B.
2. Gambar lingkaran dengan pusat B melalui A.
3. Tarik garis melallui kedua titik potong kedua lingkaran tersebut. Garis ini merupakan garis sumbu (melalui titik tengah dan tegak lurus) AB.

```
>A=[2,2]; B=[-1,-2];
>c1=circleWithCenter(A,distance(A,B));
>c2=circleWithCenter(B,distance(A,B));
>{P1,P2,f}=circleCircleIntersections(c1,c2);
>l=lineThrough(P1,P2);
>setPlotRange(5); plotCircle(c1); plotCircle(c2);
>plotPoint(A); plotPoint(B); plotSegment(A,B); plotLine(l):
```



Next, we do the same in Maxima with general coordinates.

```
>A &= [a1,a2]; B &= [b1,b2];
>c1 &= circleWithCenter(A,distance(A,B));
>c2 &= circleWithCenter(B,distance(A,B));
>P &= circleCircleIntersections(c1,c2); P1 &= P[1]; P2 &= P[2];
```

The equations for the intersections are quite involved. But we can simplify, if we solve for y.

```
>g &= getLineEquation(lineThrough(P1,P2),x,y);
>$solve(g,y)
```

$$\left[y=\frac{-\left(2\,b_1-2\,a_1\right)\,x+b_2^2+b_1^2-a_2^2-a_1^2}{2\,b_2-2\,a_2}\right]$$

This is indeed the same as the middle perpendicular, which is computed in a completely different way.

```
>$solve(getLineEquation(middlePerpendicular(A,B),x,y),y)
```

$$\left[y=\frac{-\left(2\,b_1-2\,a_1\right)\,x+b_2^2+b_1^2-a_2^2-a_1^2}{2\,b_2-2\,a_2}\right]$$

```
>h &=getLineEquation(lineThrough(A,B),x,y);
>$solve(h,y)
```

$$\left[y=\frac{\left(b_2-a_2\right)\,x-a_1\,b_2+a_2\,b_1}{b_1-a_1}\right]$$

Perhatikan hasil kali gradien garis g dan h adalah:

Artinya kedua garis tegak lurus.

Contoh 3: Rumus Heron

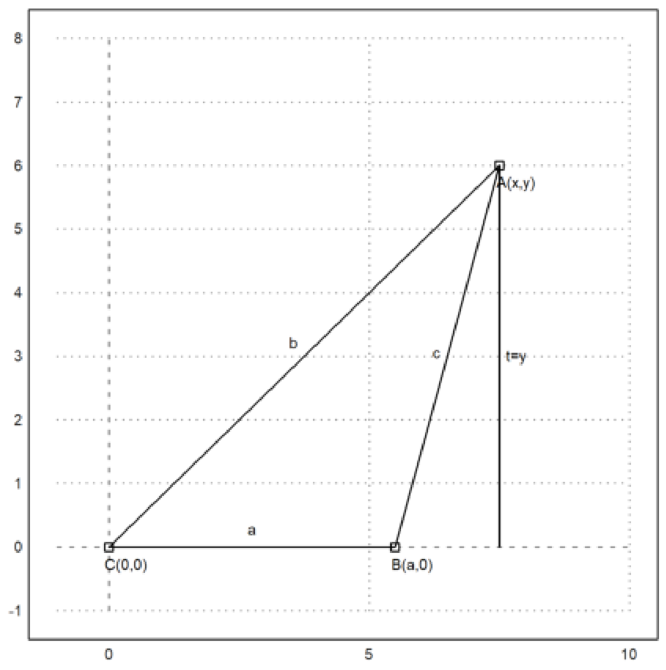
Rumus Heron menyatakan bahwa luas segitiga dengan panjang sisi-sisi a, b dan c adalah:

atau bisa ditulis dalam bentuk lain:

Untuk membuktikan hal ini kita misalkan C(0,0), B(a,0) dan A(x,y), b=AC, c=AB. Luas segitiga ABC adalah

Nilai y didapat dengan menyelesaikan sistem persamaan:

```
>setPlotRange(-1,10,-1,8); plotPoint([0,0], "C(0,0)"); plotPoint([5.5,0], "B(a,0)");  ...
  plotPoint([7.5,6], "A(x,y)");
>plotSegment([0,0],[5.5,0], "a",25); plotSegment([5.5,0],[7.5,6],"c",15);  ...
 plotSegment([0,0],[7.5,6],"b",25);
>plotSegment([7.5,6],[7.5,0],"t=y",25):
```



```
>&assume(a>0); sol &= solve([x^2+y^2=b^2,(x-a)^2+y^2=c^2],[x,y])
```

```
                                   []
```

Extract the solution y.

```
>ysol &= y with sol[2][2]; $'y=sqrt(factor(ysol^2))
```

```
Maxima said:
part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);

Error in:
ysol &= y with sol[2][2]; $'y=sqrt(factor(ysol^2)) ...
                        ^
```

We get the Heron formula.

```
>function H(a,b,c) &= sqrt(factor((ysol*a/2)^2)); $'H(a,b,c)=H(a,b,c)
```

$$H\left(a , b , \left[1 , 0 , 4 \right] \right)=\frac{a\,\left| ysol\right| }{2}$$

```
>$'Luas=H(2,5,6) // luas segitiga dengan panjang sisi-sisi 2, 5, 6
```

$$Luas=\left| ysol\right|$$

Of course, each rectangular triangle is a well known case.

```
>H(3,4,5) //luas segitiga siku-siku dengan panjang sisi 3, 4, 5
```

```
Variable or function ysol not found.
Try "trace errors" to inspect local variables after errors.
H:
    useglobal; return a*abs(ysol)/2
Error in:
H(3,4,5) //luas segitiga siku-siku dengan panjang sisi 3, 4, 5 ...
        ^
```

And it is also obvious, that this is the triangle with maximal area and the two sides 3 and 4.

```
>aspect (1.5); plot2d(&H(3,4,x),1,7): // Kurva luas segitiga sengan panjang sisi 3, 4, x (1<= x <=7)
```

```
Variable or function ysol not found.
Error in expression: 3*abs(ysol)/2
 %ploteval:
    y0=f$(x[1],args());
adaptiveevalone:
    s=%ploteval(g$,t;args());
Try "trace errors" to inspect local variables after errors.
plot2d:
    dw/n,dw/n^2,dw/n,auto;args());
```

The general case works too.

```
>$solve(diff(H(a,b,c)^2,c)=0,c)
```

```
Maxima said:
diff: second argument must be a variable; found [1,0,4]
 -- an error. To debug this try: debugmode(true);

Error in:
 $solve(diff(H(a,b,c)^2,c)=0,c) ...
                               ^
```

Now let us find the set of all points where b+c=d for some constant d. It is well known that this is an ellipse.

```
>s1 &= subst(d-c,b,sol[2]); $s1
```

```
Maxima said:
part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);

Error in:
s1 &= subst(d-c,b,sol[2]); $s1 ...
                         ^
```

And make functions of this.

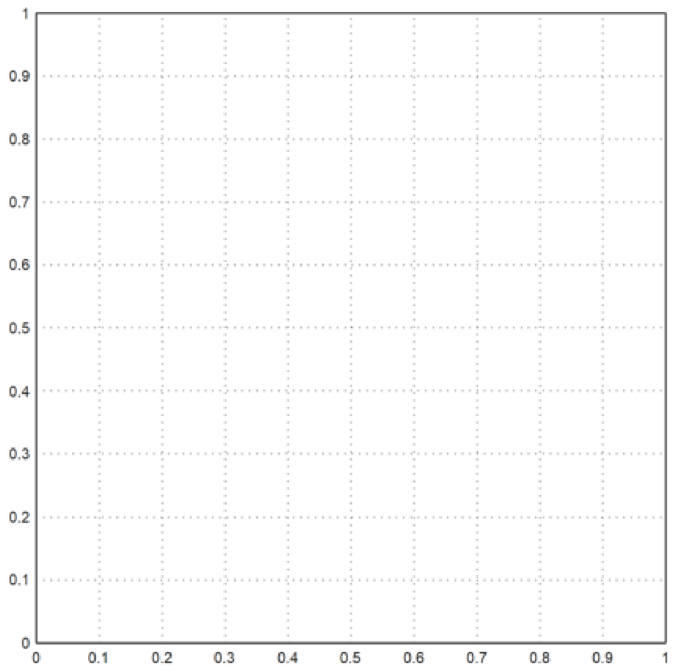
```
>function fx(a,c,d) &= rhs(s1[1]); $fx(a,c,d), function fy(a,c,d) &= rhs(s1[2]); $fy(a,c,d)
```

$$0$$

$$0$$

Now we can draw the set. The side b varies from 1 to 4. It is well known that we get an ellipse.

```
>aspect(1); plot2d(&fx(3,x,5),&fy(3,x,5),xmin=1,xmax=4,square=1):
```



We can check the general equation for this ellipse, i.e.

where (xm,ym) is the center, and u and v are the half axes.

```
>$ratsimp((fx(a,c,d)-a/2)^2/u^2+fy(a,c,d)^2/v^2 with [u=d/2,v=sqrt(d^2-a^2)/2])
```

$$\frac{a^2}{d^2}$$

We see that the height and thus the area of the triangle is maximal for x=0. Thus the area of a triangle with a+b+c=d is maximal, if it is equilateral. We wish to derive this analytically.

```
>eqns &= [diff(H(a,b,d-(a+b))^2,a)=0,diff(H(a,b,d-(a+b))^2,b)=0]; $eqns
```

$$\left[\frac{a\,ysol^2}{2}=0,0=0\right]$$

We get some minima, which belong to triangles with one side 0, and the solution a=b=c=d/3.

```
>$solve(eqns,[a,b])
```

$$[[a=0,b=\%r_1]]$$

There is also the Lagrange method, maximizing H(a,b,c)^2 with respect to a+b+d=d.

```
>&solve([diff(H(a,b,c)^2,a)=la,diff(H(a,b,c)^2,b)=la, ...
    diff(H(a,b,c)^2,c)=la,a+b+c=d],[a,b,c,la])
```

```
 Maxima said:
 diff: second argument must be a variable; found [1,0,4]
  -- an error. To debug this try: debugmode(true);

 Error in:
 ... la,    diff(H(a,b,c)^2,c)=la,a+b+c=d],[a,b,c,la]) ...
                                                      ^
```

We can make a plot of the situation

First set the points in Maxima.

```
>A &= at([x,y],sol[2]); $A
```

```
Maxima said:
part: invalid index of list or matrix.
 -- an error. To debug this try: debugmode(true);

Error in:
A &= at([x,y],sol[2]); $A ...
                     ^
```

```
>B &= [0,0]; $B, C &= [a,0]; $C
```

$$[0,0]$$

$$[a,0]$$

Then set the plot range, and plot the points.

```
>setPlotRange(0,5,-2,3); ...
 a=4; b=3; c=2; ...
 plotPoint(mxmeval("B"),"B"); plotPoint(mxmeval("C"),"C"); ...
 plotPoint(mxmeval("A"),"A"):
```

```
Variable a1 not found!
Use global variables or parameters for string evaluation.
Error in Evaluate, superfluous characters found.
Try "trace errors" to inspect local variables after errors.
mxmeval:
    return evaluate(mxm(s));
Error in:
... otPoint(mxmeval("C"),"C"); plotPoint(mxmeval("A"),"A"): ...
                                                     ^
```

Plot the segments.

```
>plotSegment(mxmeval("A"),mxmeval("C")); ...
 plotSegment(mxmeval("B"),mxmeval("C")); ...
 plotSegment(mxmeval("B"),mxmeval("A")):
```

```
Variable a1 not found!
Use global variables or parameters for string evaluation.
Error in Evaluate, superfluous characters found.
Try "trace errors" to inspect local variables after errors.
mxmeval:
    return evaluate(mxm(s));
Error in:
plotSegment(mxmeval("A"),mxmeval("C")); plotSegment(mxmeval("B ...
                        ^
```

Compute the middle perpendicular in Maxima.

```
>h &= middlePerpendicular(A,B); g &= middlePerpendicular(B,C);
```

And the center of the circumference.

```
>U &= lineIntersection(h,g);
```

We get the formula for the radius of the circumcircle.

```
>&assume(a>0,b>0,c>0); $distance(U,B) | radcan
```

$$\frac{\sqrt{a_2^2+a_1^2}\,\sqrt{a_2^2+a_1^2-2\,a\,a_1+a^2}}{2\,|a_2|}$$

Let us add this to the plot.

```
>plotPoint(U()); ...
 plotCircle(circleWithCenter(mxmeval("U"),mxmeval("distance(U,C)"))):
```

```
Variable a2 not found!
Use global variables or parameters for string evaluation.
Error in ^
Error in expression: [a/2,(a2^2+a1^2-a*a1)/(2*a2)]
Error in:
```

```
plotPoint(U()); plotCircle(circleWithCenter(mxmeval("U"),mxmev ...
                           ^
```

Using geometry, we derive the simple formula

for the radius. We can check, if this is really true with Maxima. Maxima will factor this only if we square it.

```
>$c^2/sin(computeAngle(A,B,C))^2  | factor
```

$$\left[\frac{a_2^2+a_1^2}{a_2^2}, 0, \frac{16\left(a_2^2+a_1^2\right)}{a_2^2}\right]$$

Contoh 4: Garis Euler dan Parabola

Euler line is a line determined from any triangle that is not equilateral. It is a central line of the triangle, and it passes through several important points determined from the triangle, including the orthocenter, the circumcenter, the centroid, the Exeter point and the center of the nine-point circle of the triangle.

For a demonstration, we compute and plot the Euler line in a triangle.

First, we define the corners of the triangle in Euler. We use a definition, which is visible in symbolic expressions.

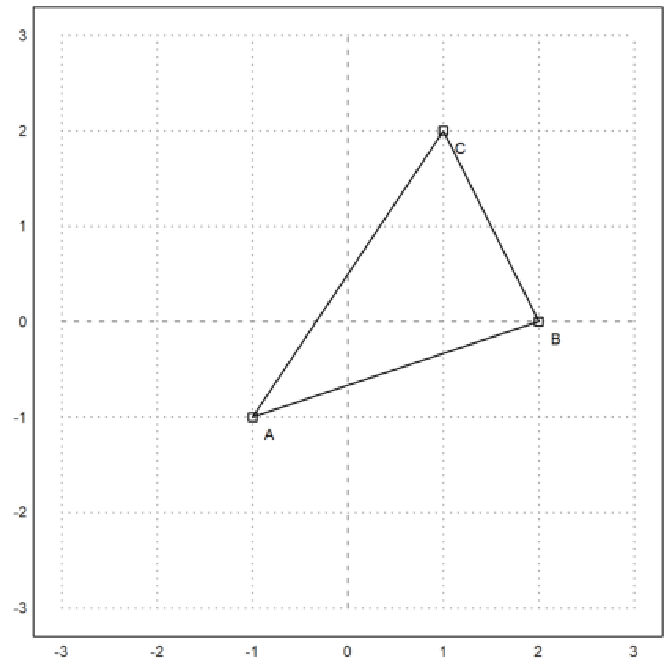
```
>A::=[-1,-1]; B::=[2,0]; C::=[1,2];
```

To plot geometric objects, we setup a plot area, and add the points to it. All plots of geometric objects are added to the current plot.

```
>setPlotRange(3); plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C");
```

We can also add the sides of the triangle.

```
>plotSegment(A,B,""); plotSegment(B,C,""); plotSegment(C,A,""):
```



Here is the area of the triangle, using the determinant formula. Of course, we have to take the absolute value of this result.

```
>$areaTriangle(A,B,C)
```

$$-\frac{7}{2}$$

We can compute the coefficients of the side c.

```
>c &= lineThrough(A,B)
```

```
[- 1, 3, - 2]
```

And also get a formula for this line.

```
>$getLineEquation(c,x,y)
```

$$3\,y-x=-2$$

For the Hesse form, we need to specify a point, such that the point is on the positive side of the Hesseform. Inserting the point yields the positive distance to the line.

```
>$getHesseForm(c,x,y,C), $at(%,[x=C[1],y=C[2]])
```

$$\frac{3\,y-x+2}{\sqrt{10}}$$

$$\frac{7}{\sqrt{10}}$$

Now we compute the circumcircle of ABC.

```
>LL &= circleThrough(A,B,C); $getCircleEquation(LL,x,y)
```

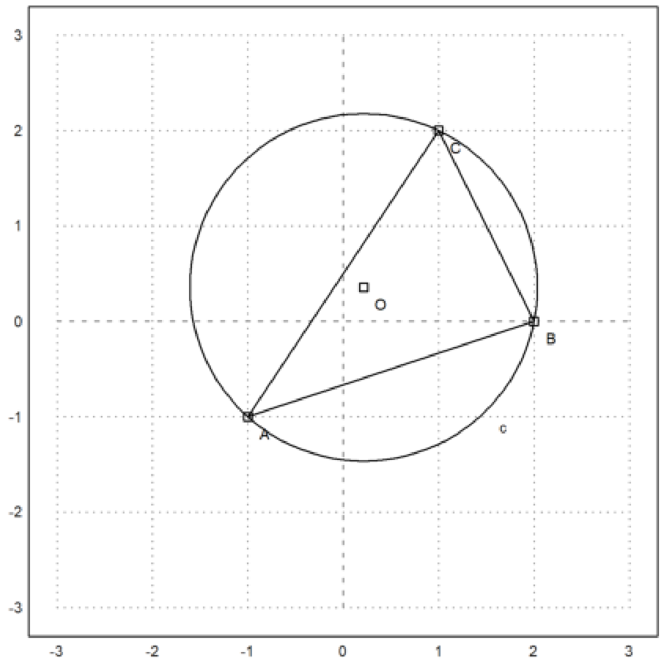
$$\left(y-\frac{5}{14}\right)^2+\left(x-\frac{3}{14}\right)^2=\frac{325}{98}$$

```
>O &= getCircleCenter(LL); $O
```

$$\left[\frac{3}{14},\frac{5}{14}\right]$$

Plot the circle and its center. Cu and U are symbolic. We evaluate these expressions for Euler.

```
>plotCircle(LL()); plotPoint(O(),"O"):
```



We can compute the intersection of the heights in ABC (the orthocenter) numerically with the following command.

```
>H &= lineIntersection(perpendicular(A,lineThrough(C,B)),...
   perpendicular(B,lineThrough(A,C))); $H
```

$$\left[\frac{11}{7},\frac{2}{7}\right]$$

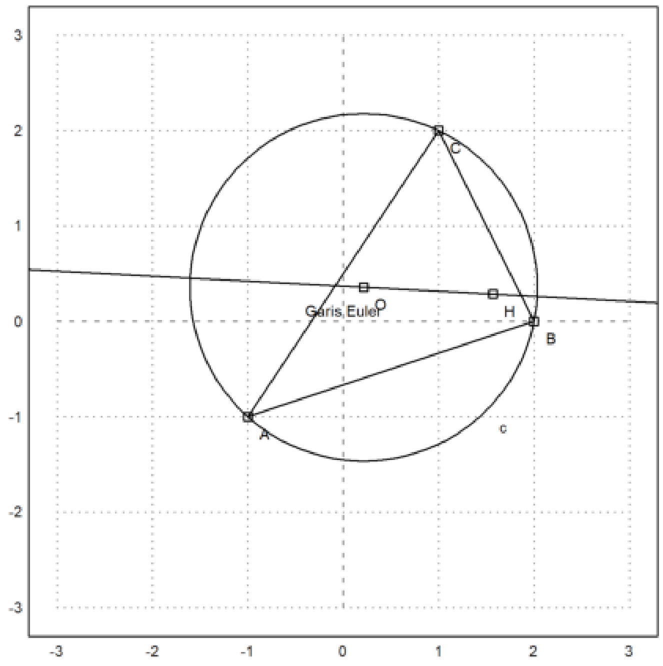
Now we can compute the Euler line of the triangle.

```
>el &= lineThrough(H,O); $getLineEquation(el,x,y)
```

$$-\frac{19\,y}{14}-\frac{x}{14}=-\frac{1}{2}$$

Add it to our plot.

```
>plotPoint(H(),"H"); plotLine(el(),"Garis Euler"):
```

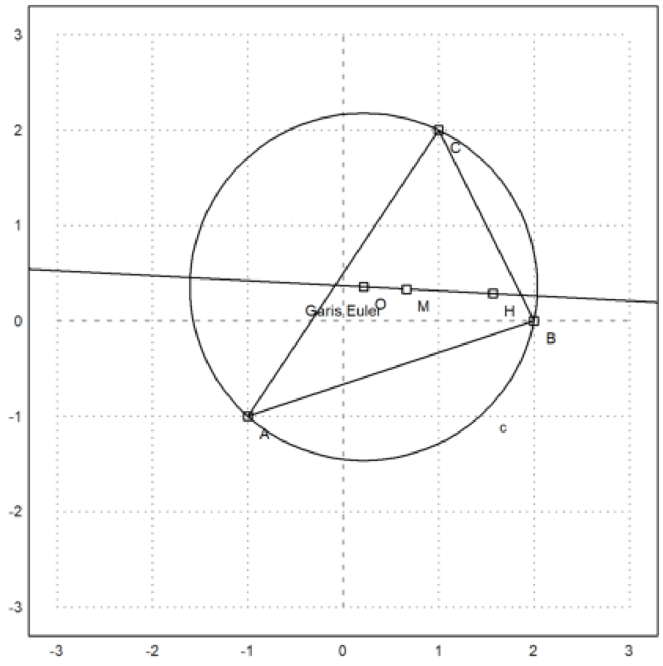


The gravitational center should be on this line.

```
>M &= (A+B+C)/3; $getLineEquation(el,x,y) with [x=M[1],y=M[2]]
```

$$-\frac{1}{2}=-\frac{1}{2}$$

```
>plotPoint(M(),"M"): // titik berat
```



The theory tells us MH=2*MO. We need to simplify with radcan to achieve this.

```
>$distance(M,H)/distance(M,O)|radcan
```

$$2$$

The functions include functions for angles too.

```
>$computeAngle(A,C,B), degprint(%())
```

$$\arccos\left(\frac{4}{\sqrt{5}\,\sqrt{13}}\right)$$

```
60°15'18.43''
```

The equation for the center of the incircle is not very nice.

```
>Q &= lineIntersection(angleBisector(A,C,B),angleBisector(C,B,A))|radcan; $Q
```

$$\left[\frac{\left(2^{\frac{3}{2}}+1\right)\,\sqrt{5}\,\sqrt{13}-15\,\sqrt{2}+3}{14},\frac{\left(\sqrt{2}-3\right)\,\sqrt{5}\,\sqrt{13}+5\,2^{\frac{3}{2}}+5}{14}\right]$$

Let us compute also the expression for the radius of the inscribed circle.

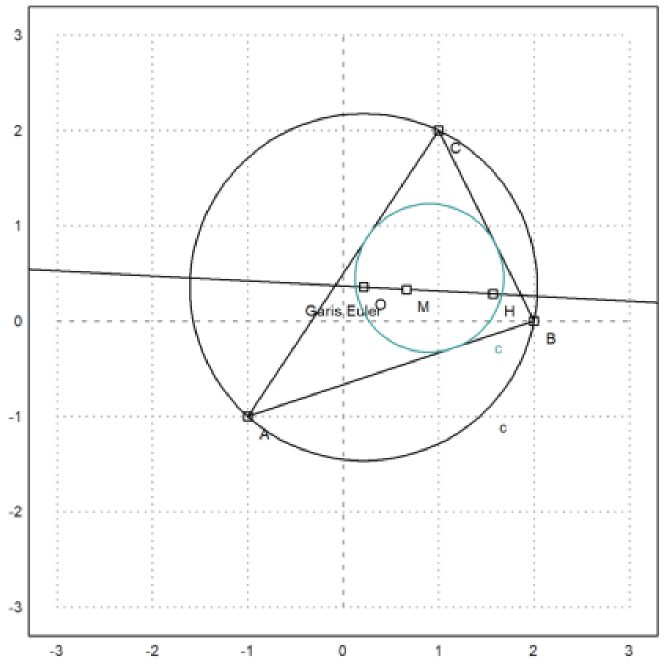
```
>r &= distance(Q,projectToLine(Q,lineThrough(A,B)))|ratsimp; $r
```

$$\frac{\sqrt{\left(-41\,\sqrt{2}-31\right)\,\sqrt{5}\,\sqrt{13}+115\,\sqrt{2}+614}}{7\,\sqrt{2}}$$

```
>LD &=  circleWithCenter(Q,r); // Lingkaran dalam
```

Let us add this to the plot.

```
>color(5); plotCircle(LD()):
```



Parabola

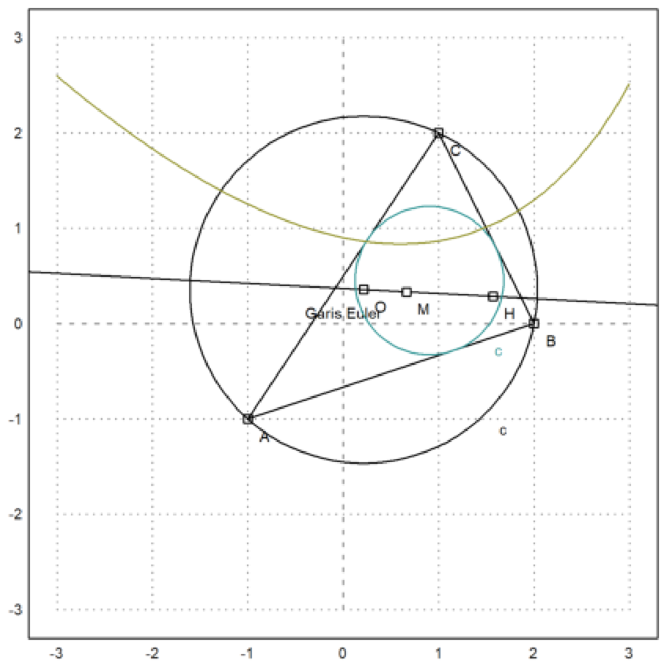
Selanjutnya akan dicari persamaan tempat kedudukan titik-titik yang berjarak sama ke titik C dan ke garis AB.

```
>p &= getHesseForm(lineThrough(A,B),x,y,C)-distance([x,y],C); $p='0
```

$$\frac{3\,y-x+2}{\sqrt{10}}-\sqrt{(2-y)^2+(1-x)^2}=0$$

Persamaan tersebut dapat digambar menjadi satu dengan gambar sebelumnya.

```
>plot2d(p,level=0,add=1,contourcolor=6):
```



This should be some function, but the default solver of Maxima can find the solution only, if we square the equation. Consequently, we get a fake solution.

```
>akar &= solve(getHesseForm(lineThrough(A,B),x,y,C)^2-distance([x,y],C)^2,y)
```

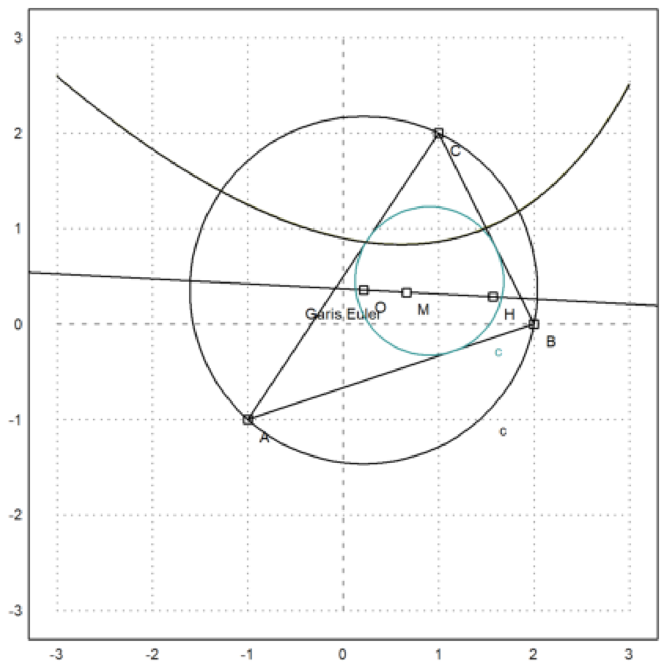
```
            [y = - 3 x - sqrt(70) sqrt(9 - 2 x) + 26, 
                                  y = - 3 x + sqrt(70) sqrt(9 - 2 x) + 26]
```

The first solution is

maxima: akar[1]

Adding the first solution to the plot show, that it is indeed the path we are looking for. The theory tells us that it is a rotated parabola.

```
>plot2d(&rhs(akar[1]),add=1):
```



```
>function g(x) &= rhs(akar[1]); $'g(x)= g(x)// fungsi yang mendefinisikan kurva di atas
```

$$g\left(x\right)=-3\,x-\sqrt{70}\,\sqrt{9-2\,x}+26$$

```
>T &=[-1, g(-1)]; // ambil sebarang titik pada kurva tersebut
>dTC &= distance(T,C); $fullratsimp(dTC), $float(%) // jarak T ke C
```

$$\sqrt{1503-54\,\sqrt{11}\,\sqrt{70}}$$

$$2.135605779339061$$

```
>U &= projectToLine(T,lineThrough(A,B)); $U // proyeksi T pada garis AB
```

$$\left[\frac{80-3\,\sqrt{11}\,\sqrt{70}}{10},\frac{20-\sqrt{11}\,\sqrt{70}}{10}\right]$$

```
>dU2AB &= distance(T,U); $fullratsimp(dU2AB), $float(%) // jatak T ke AB
```

$$\sqrt{1503-54\,\sqrt{11}\,\sqrt{70}}$$

$$2.135605779339061$$

Ternyata jarak T ke C sama dengan jarak T ke AB. Coba Anda pilih titik T yang lain dan ulangi perhitungan-perhitungan di atas untuk menunjukkan bahwa hasilnya juga sama.

Contoh 5: Trigonometri Rasional

This is inspired by a talk N.J.Wildberger. In his book "Divine Proportions", Wildberger proposes to replace the classical notions of distance and angle by quadrance and spread. Using these, it is indeed possible to avoid trigonometric functions in many examples, and to stay "rational".

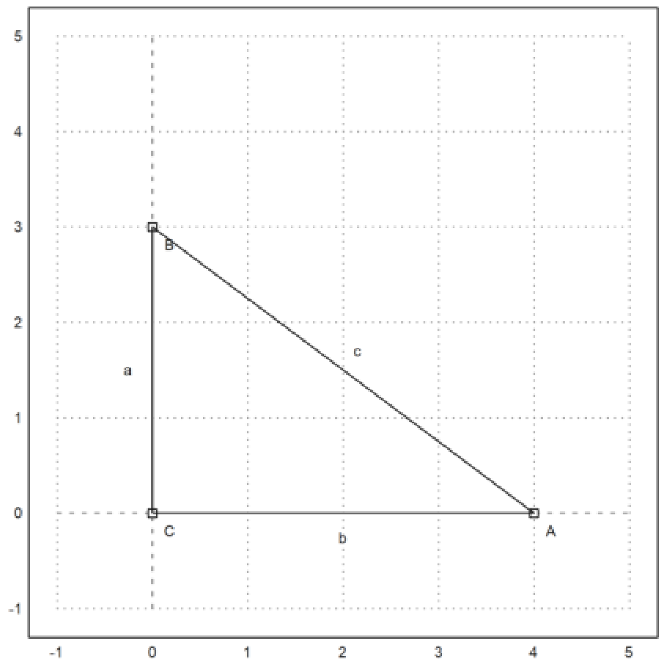
In the following, I introduce the concepts, and solve a few problems. I am using symbolic computations Maxima here, which hides the main advantage of rational trigonometry that the computations can be performed by paper and pencil only. You are invited to check the results without a computer.

The point is that symbolic rational computations often yield simple results. In contrast, classical trigonometry yields complicated trigonometric results, which evaluate to numerical approximations only.

```
>load geometry;
```

For a first introduction, we use the rectangular triangle with the famous Egyptian proportions 3, 4 and 5. The following commands are Euler commands to plot plane geometry contained in the Euler file "geometry.e".

```
>C&:=[0,0]; A&:=[4,0]; B&:=[0,3]; ...
 setPlotRange(-1,5,-1,5); ...
 plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
 plotSegment(B,A,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
 insimg(30);
```



Of course,

$$\sin(w_a) = \frac{a}{c},$$

where wa is the angle at A. The usual way to compute this angle, is to take the inverse of the sine function. The result is an undigestible angle, which can only be printed approximately.

```
>wa := arcsin(3/5); degprint(wa)
```

```
 36°52'11.63''
```

Rational trigonometry tries to avoid this.

The first notion of rational trigonometry is a quadrance, which replaces the distance. In fact, it is just the distance squared. In the following, a, b, and c denote the quadrances of the sides.

The Pythogoras theorem simply becomes a+b=c then.

```
>a &= 3^2; b &= 4^2; c &= 5^2; &a+b=c
```

```
                                 25 = 25
```

The second notion of rational trigonometry is the spread. The spread measures the opening between lines. It is 0, if the lines are parallel, and 1, if the lines are rectangular. It is the square of the sine of the angle between the two lines.

The spread of the lines AB and AC in the image above is defined as

$$s_a = \sin(\alpha)^2 = \frac{a}{c},$$

where a and c are the quadrances of any rectangular triangle with one corner in A.

```
>sa &= a/c; $sa
```

$$\frac{9}{25}$$

This is easier to compute than the angle, of course. But you lose the property that angles can be added easily.

Of course, we can convert our approximate value for the angle wa to a sprad, and print it as a fraction.

```
>fracprint(sin(wa)^2)
```

```
 9/25
```

The cosine law of classical trgonometry translates into the following "cross law".

$$(c+b-a)^2 = 4bc\,(1-s_a)$$

Here a, b, and c are quadrances of the sides of a triangle, and sa is the spread of at the corner A. The side a is, as usual, opposite to the corner A.

These laws are implemented in the geometry.e file we loaded into Euler.

```
>$crosslaw(aa,bb,cc,saa)
```

$$crosslaw\,(aa, bb, cc, saa)$$

In our case we get

```
>$crosslaw(a,b,c,sa)
```

$$1024 = 1024$$

Let us use this crosslaw to find the spread at A. To do this, we produce the crosslaw for the quadrances a, b, and c, and solve it for the unknown spread sa.

You can do this by hand easily, but I use Maxima. Of course, we get the result, we already had.

```
>$crosslaw(a,b,c,x), $solve(%,x)
```

$$1024 = 1600\,(1-x)$$

$$\left[x = \frac{9}{25}\right]$$

We know this already. The definition of the spread is a special case of the crosslaw.

We can also solve this for general a,b,c. The result is a formula which computes the spread of an angle of a triangle given the quadrances of the three sides.

```
>$solve(crosslaw(aa,bb,cc,x),x)
```

$$\left[\left[\frac{168\,bb\,x+36\,bb^2+(-72\,aa-84)\,bb+36\,aa^2-84\,aa+49}{36},\frac{168\,bb\,x+36\,bb^2+(-72\,aa-84)\,bb+36\,aa^2-84\,aa+49}{36},\frac{15\,2^{\frac{1}{3}}\,bb\,x+18\,bb^2+\left(-36\,aa-15\,2^{\frac{1}{3}}\right)\,bb+18\,aa^2-15\,2^{\frac{1}{3}}\,aa+25}{18}\right]=0\right]$$

We could make a function of the result. Such a function is already defined in the geometry.e file of Euler.

```
>$spread(a,b,c)
```

$$\frac{9}{25}$$

As an example, we can use it to compute the angle of a triangle with sides

$$a,\quad a,\quad \frac{4a}{7}$$

The result is rational, which is not so easy to get if we use classical trigonometry.

```
>$spread(a,a,4*a/7)
```

$$\frac{6}{7}$$

This is the angle in degrees.

```
>degprint(arcsin(sqrt(6/7)))
```

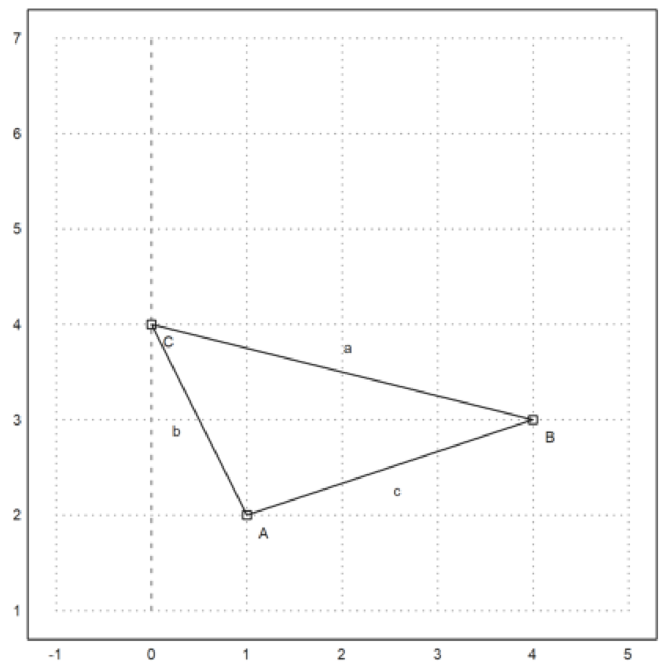
```
 67°47'32.44''
```

Another Example

Now, let us try a more advanced example.

We set three corners of a triangle as follows.

```
>A&:=[1,2]; B&:=[4,3]; C&:=[0,4]; ...
 setPlotRange(-1,5,1,7); ...
 plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
 plotSegment(B,A,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
 insimg;
```



Using Pythogoras, it is easy to compute the distance between two points. I first use the function distance of the Euler file for geometry. The function distance uses classical geometry.

```
>$distance(A,B)
```

$$\sqrt{10}$$

Euler does also contain functions for the quadrance between two points.

In the following example, since c+b is not a, the triangle is not rectangular.

```
>c &= quad(A,B); $c, b &= quad(A,C); $b, a &= quad(B,C); $a,
```

$$10$$

$$5$$

$$17$$

First, let us compute the traditional angle. The function computeAngle uses the usual method based on the dot product of two vectors. The result is some floating point approximation.

$$A=<1,2>\quad B=<4,3>,\quad C=<0,4>$$

$$\mathbf{a}=C-B=<-4,1>,\quad \mathbf{c}=A-B=<-3,-1>,\quad \beta=\angle ABC$$

$$\mathbf{a}.\mathbf{c}=|\mathbf{a}|.|\mathbf{c}|\cos\beta$$

$$\cos\angle ABC=\cos\beta=\frac{\mathbf{a}.\mathbf{c}}{|\mathbf{a}|.|\mathbf{c}|}=\frac{12-1}{\sqrt{17}\sqrt{10}}=\frac{11}{\sqrt{17}\sqrt{10}}$$

```
>wb &= computeAngle(A,B,C); $wb, $(wb/pi*180)()
```

$$\arccos\left(\frac{11}{\sqrt{10}\,\sqrt{17}}\right)$$

```
32.4711922908
```

Using pencil and paper, we can do the same with the cross law. We insert the quadrances a, b, and c into the cross law and solve for x.

```
>$crosslaw(a,b,c,x), $solve(%,x), //(b+c-a)^=4b.c(1-x)
```

$$4=200\,(1-x)$$

$$\left[x=\frac{49}{50}\right]$$

That is, what the function spread defined in "geometry.e" does.

```
>sb &= spread(b,a,c); $sb
```

$$\frac{49}{170}$$

Maxima gets the same result using ordinary trigonometry, if we force it. It does resolve the sin(arccos(...)) term to a fractional result. Most students could not do this.

```
>$sin(computeAngle(A,B,C))^2
```

$$\frac{49}{170}$$

Once, we have the spread at B, we can compute the height ha on the side a. Remember that

by definition.

```
>ha &= c*sb; $ha
```

$$\frac{49}{17}$$

The following image has been produced with the geometry program C.a.R., which can draw quadrances and spreads.

image: (20) Rational_Geometry_CaR.png

By definition the length of ha is the square root of its quadrance.

```
>$sqrt(ha)
```

$$\frac{7}{\sqrt{17}}$$

Now we can compute the area of the triangle. Do not forget, that we are dealing with quadrances!

```
>$sqrt(ha)*sqrt(a)/2
```

$$\frac{7}{2}$$

The usual determinant formula yields the same result.

```
>$areaTriangle(B,A,C)
```

$$\frac{7}{2}$$

The Heron Formula

Now, let us solve this problem in general!

```
>&remvalue(a,b,c,sb,ha);
```

We first compute the spread at B for a triangle with sides a, b, and c. Then we compute the area squared (the "quadrea"?), factor it with Maxima, and we get the famous formula of Heron.

Admittedly, this is tough to do with pencil and paper.

```
>$spread(b^2,c^2,a^2), $factor(%*c^2*a^2/4)
```

$$\frac{-c^4-\left(-2\,b^2-2\,a^2\right)\,c^2-b^4+2\,a^2\,b^2-a^4}{4\,a^2\,c^2}$$

$$\frac{\left(-c+b+a\right)\,\left(c-b+a\right)\,\left(c+b-a\right)\,\left(c+b+a\right)}{16}$$

The Triple Spread Rule

The disadvantage of spreads is that they do no longer simply add like angles.

However, the three spreads of a triangle satisfy the following "triple spread" rule.

```
>&remvalue(sa,sb,sc); $triplespread(sa,sb,sc)
```

$$\left(sc+sb+sa\right)^2=2\,\left(sc^2+sb^2+sa^2\right)+4\,sa\,sb\,sc$$

This rule is valid for any three angles that add to 180°.

Since the spreads of

are equal, the triple spread rule is also true, if

Since the spread of the negative angle is the same, the triple spread rule also holds, if

For example, we can compute the spread of the 60° angle. It is 3/4. The equations have a second solution, however, where all spreads are 0.

```
>$solve(triplespread(x,x,x),x)
```

$$\left[x=\frac{3}{4} , x=0\right]$$

The spread of 90° is obviously 1. If two angles add to 90°, their spread solves the triple spread equation with a,b,1. By the following computation we get a+b=1.

```
>$triplespread(x,y,1), $solve(%,x)
```

$$\left(y+x+1\right)^2=2\,\left(y^2+x^2+1\right)+4\,x\,y$$

$$\left[x=1-y\right]$$

Since the spread of 180°-t is the same as the spread of t, the triple spread formula also holds, if one angle is the sum or difference of the two other angles.

So we can find the spread of the doubled angle. Note that there are two solutions again. We make this a function.

```
>$solve(triplespread(a,a,x),x), function doublespread(a) &= factor(rhs(%[1]))
```

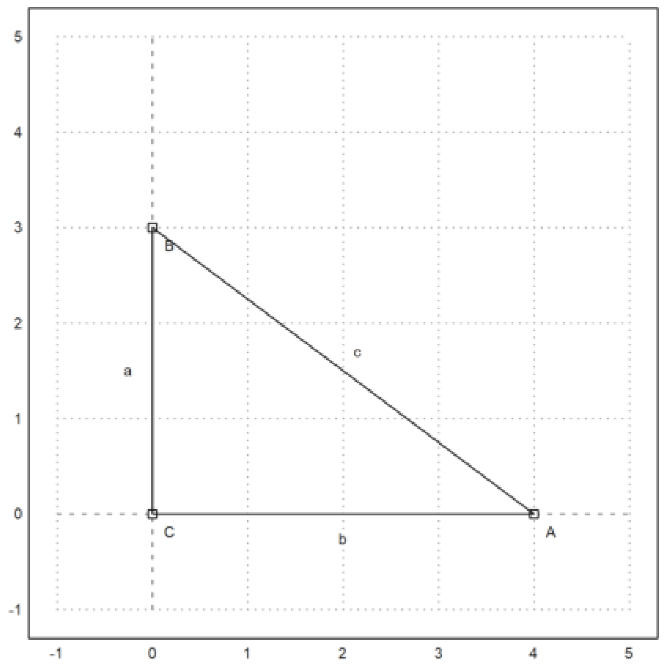
$$\left[x=4\,a-4\,a^2,x=0\right]$$

```
- 4 (a - 1) a
```

Angle Bisectors

This is the situation, we already know.

```
>C&:=[0,0]; A&:=[4,0]; B&:=[0,3]; ...
 setPlotRange(-1,5,-1,5); ...
 plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
 plotSegment(B,A,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
 insimg;
```



Let us compute the length of the angle bisector at A. But we want to solve that for general a,b,c.

```
>&remvalue(a,b,c);
```

So we first compute the spread of the bisected angle at A, using the triple spread formula.

The problem with this formula shows up again. It has two solutions. We have to pick the correct one. The other solution refers to the bisected angle 180°-wa.

```
>$triplespread(x,x,a/(a+b)), $solve(%,x), sa2 &= rhs(%[1]); $sa2
```

$$\left(2\,x+\frac{a}{b+a}\right)^2=2\,\left(2\,x^2+\frac{a^2}{(b+a)^2}\right)+\frac{4\,a\,x^2}{b+a}$$

$$\left[x=\frac{-\sqrt{b}\,\sqrt{b+a}+b+a}{2\,b+2\,a},x=\frac{\sqrt{b}\,\sqrt{b+a}+b+a}{2\,b+2\,a}\right]$$

$$\frac{-\sqrt{b}\,\sqrt{b+a}+b+a}{2\,b+2\,a}$$

Let us check for the Egyptian rectangle.

```
>$sa2 with [a=3^2,b=4^2]
```

$$\frac{1}{10}$$

We can print the angle in Euler, after transferring the spread to radians.

```
>wa2 := arcsin(sqrt(1/10)); degprint(wa2)
```

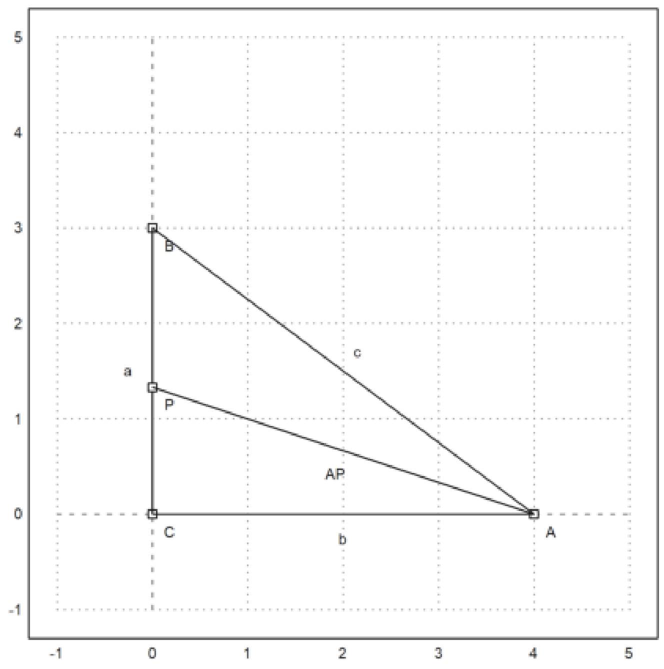
```
18°26'5.82''
```

The point P is the intersection of the angle bisector with the y-axis.

```
>P := [0,tan(wa2)*4]
```

```
[0,  1.33333]
```

```
>plotPoint(P,"P"); plotSegment(A,P):
```



Let us check the angles in our specific example.

```
>computeAngle(C,A,P), computeAngle(P,A,B)
```

```
0.321750554397
0.321750554397
```

Now we compute the length of the bisector AP.

We use the sine theorem in the triangle APC. This theorem states that

$$\frac{BC}{\sin(w_a)} = \frac{AC}{\sin(w_b)} = \frac{AB}{\sin(w_c)}$$

holds in any triangle. Square it, it translates into the so-called "spread law"

$$\frac{a}{s_a} = \frac{b}{s_b} = \frac{c}{s_b}$$

where a,b,c denote qudrances.

Since the spread CPA is 1-sa2, we get from it bisa/1=b/(1-sa2) and can compute bisa (quadrance of the angle bisector).

```
>&factor(ratsimp(b/(1-sa2))); bisa &= %; $bisa
```

$$\frac{2\,b\,(b+a)}{\sqrt{b}\,\sqrt{b+a}+b+a}$$

Let us check this formula for our Egyptian values.

```
>sqrt(mxmeval("at(bisa,[a=3^2,b=4^2])")), distance(A,P)
```

```
    4.21637021356
    4.21637021356
```

We can also compute P using the spread formula.

```
>py&=factor(ratsimp(sa2*bisa)); $py
```

$$-\frac{b\left(\sqrt{b}\,\sqrt{b+a}-b-a\right)}{\sqrt{b}\,\sqrt{b+a}+b+a}$$

The value is the same we got with trigonometric formulas.

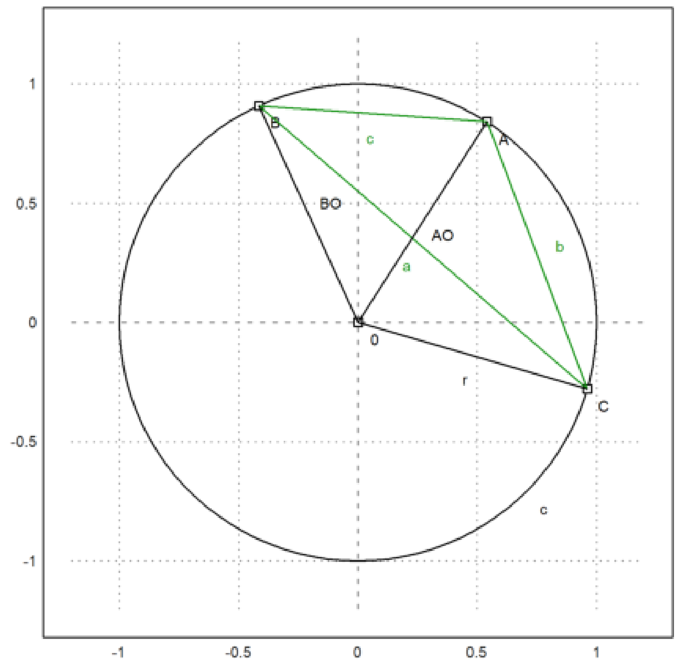
```
>sqrt(mxmeval("at(py,[a=3^2,b=4^2])"))
```

```
    1.33333333333
```

The Chord Angle

Have a look at the following situation.

```
>setPlotRange(1.2); ...
 color(1); plotCircle(circleWithCenter([0,0],1)); ...
 A:=[cos(1),sin(1)]; B:=[cos(2),sin(2)]; C:=[cos(6),sin(6)]; ...
 plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
 color(3); plotSegment(A,B,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
 color(1); O:=[0,0];  plotPoint(O,"0"); ...
 plotSegment(A,O); plotSegment(B,O); plotSegment(C,O,"r"); ...
 insimg;
```



We can use Maxima to solve the triple spread formula for the angles at the center O for r. Thus we get a formula for the quadratic radius of the pericircle in terms of quadrances of the sides.

This time, Maxima produces some complex zeros, which we ignore.

```
>&remvalue(a,b,c,r); // hapus nilai-nilai sebelumnya untuk perhitungan baru
>rabc &= rhs(solve(triplespread(spread(b,r,r),spread(a,r,r),spread(c,r,r)),r)[4]); $rabc
```

$$-\frac{a\,b\,c}{c^2-2\,b\,c+a\,\left(-2\,c-2\,b\right)+b^2+a^2}$$

We can make that an Euler function.

```
>function periradius(a,b,c) &= rabc;
```

Let us check the result for our points A,B,C.

```
>a:=quadrance(B,C); b:=quadrance(A,C); c:=quadrance(A,B);
```

The radius is indeed 1.

```
>periradius(a,b,c)
```

```
1
```

The fact is, that the spread CBA depends only on b and c. This is the chord angle theorem.

```
>$spread(b,a,c)*rabc | ratsimp
```

$$\frac{b}{4}$$

In fact the spread is b/(4r), and we see that the chord angle of the chord b is half the center angle.

```
>$doublespread(b/(4*r))-spread(b,r,r) | ratsimp
```

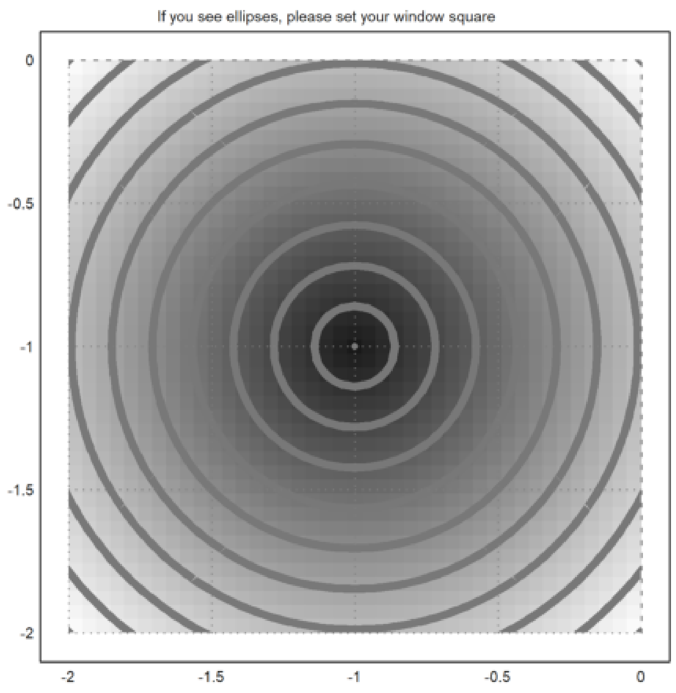
$$0$$

Contoh 6: Jarak Minimal pada Bidang

Preliminary remark

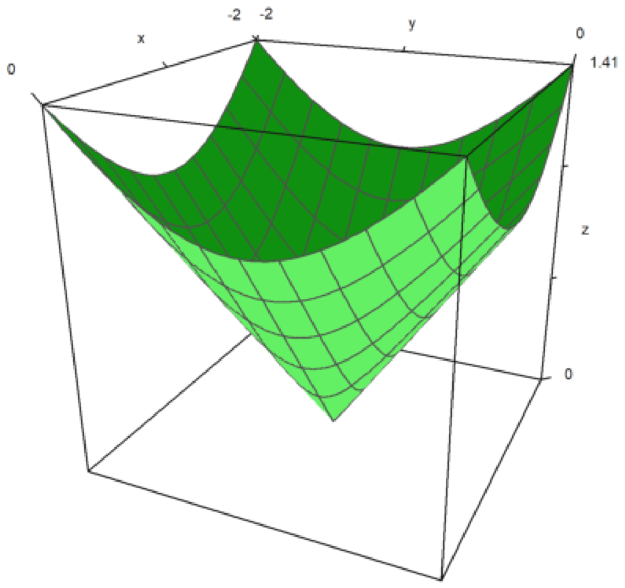
The function which, to a point M in the plane, assigns the distance AM between a fixed point A and M, has rather simple level lines: circles centered in A.

```
>&remvalue();
>A=[-1,-1];
>function d1(x,y):=sqrt((x-A[1])^2+(y-A[2])^2)
>fcontour("d1",xmin=-2,xmax=0,ymin=-2,ymax=0,hue=1, ...
 title="If you see ellipses, please set your window square"):
```



and the graph is rather simple too: the upper part of a cone:

```
>plot3d("d1",xmin=-2,xmax=0,ymin=-2,ymax=0):
```

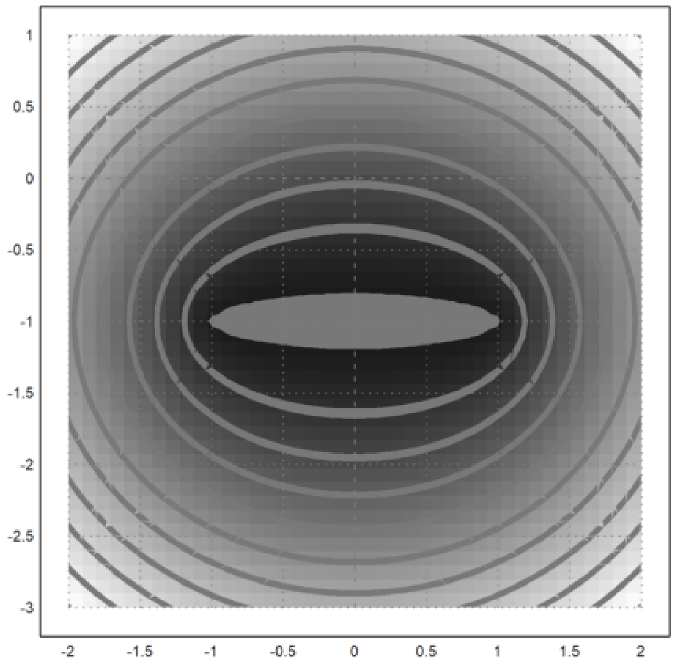


Of course the minimum 0 is attained in A.

Two points

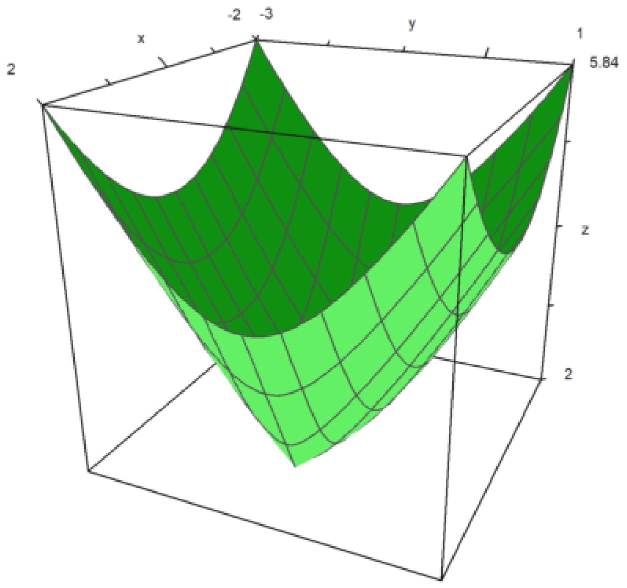
Now we look at the function MA+MB where A and B are two points (fixed). It is a "well-known fact" that the level curves are ellipses, the focal points being A and B; except for the minimum AB which is constant on the segment [AB]:

```
>B=[1,-1];
>function d2(x,y):=d1(x,y)+sqrt((x-B[1])^2+(y-B[2])^2)
>fcontour("d2",xmin=-2,xmax=2,ymin=-3,ymax=1,hue=1):
```



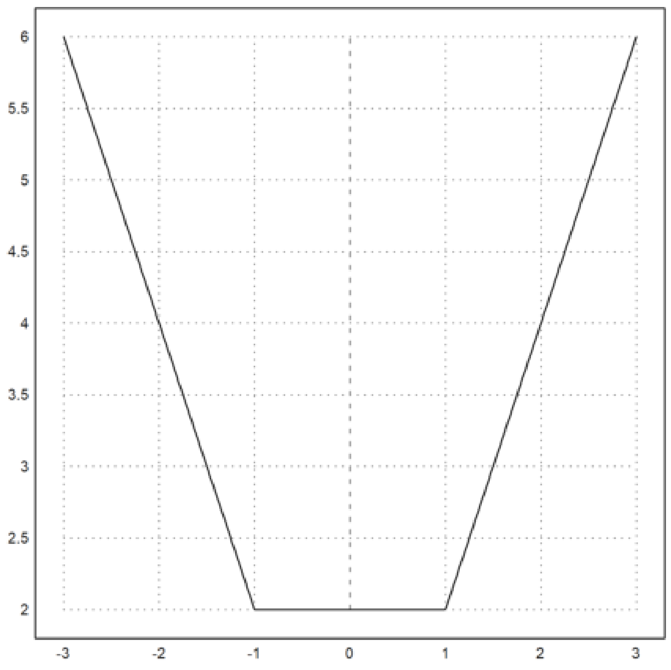
The graph is more interesting:

```
>plot3d("d2",xmin=-2,xmax=2,ymin=-3,ymax=1):
```



The restriction to line (AB) is more famous:

```
>plot2d("abs(x+1)+abs(x-1)",xmin=-3,xmax=3):
```



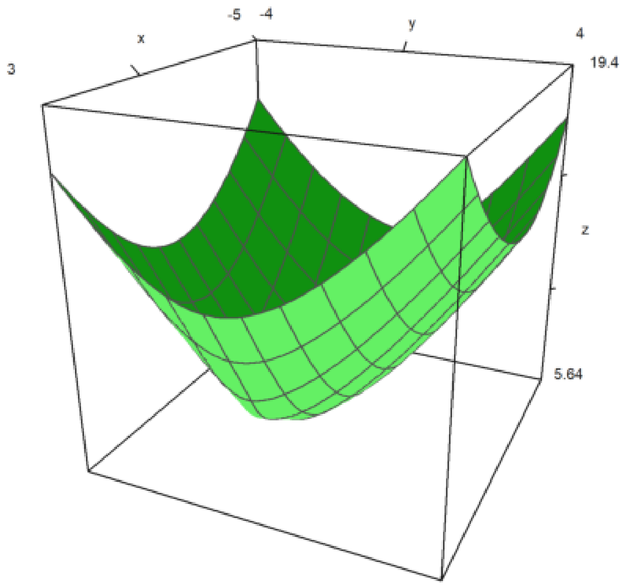
Three points

Now things are less simple: It is a little less well-known that MA+MB+MC attains its minimum at one point of the plane but to determine it is less simple:

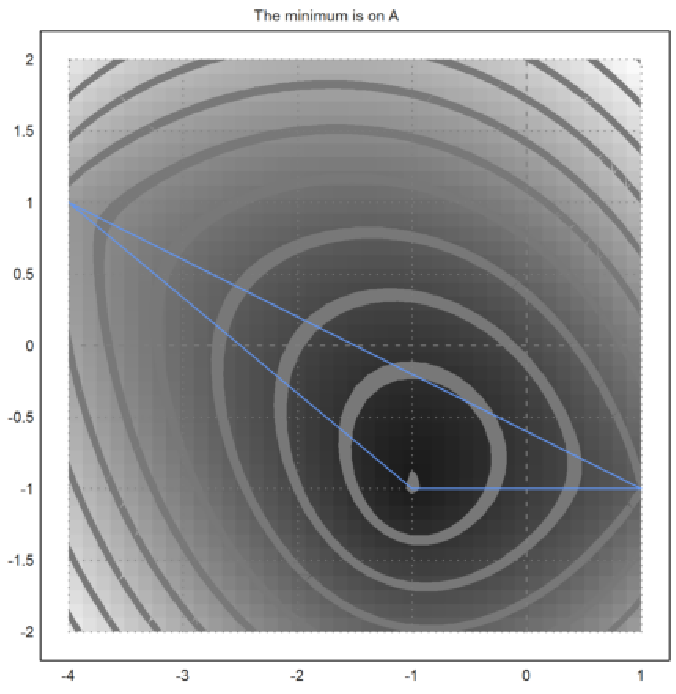
1) If one of the angles of the triangle ABC is more than 120° (say in A), then the minimum is attained at this very point (say AB+AC).

Example:

```
>C=[-4,1];
>function d3(x,y):=d2(x,y)+sqrt((x-C[1])^2+(y-C[2])^2)
>plot3d("d3",xmin=-5,xmax=3,ymin=-4,ymax=4);
>insimg;
```

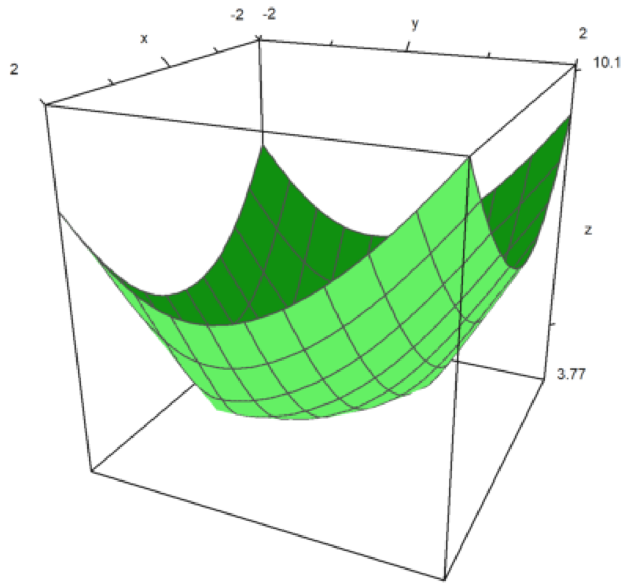


```
>fcontour("d3",xmin=-4,xmax=1,ymin=-2,ymax=2,hue=1,title="The minimum is on A");
>P=(A_B_C_A)'; plot2d(P[1],P[2],add=1,color=12);
>insimg;
```

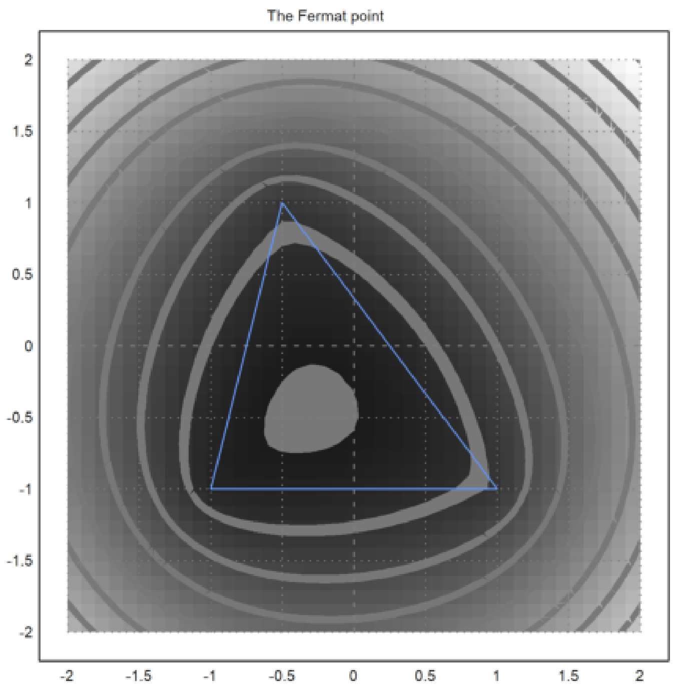


2) But if all the angles of triangle ABC are less than 120°, the minimum is on a point F in the interior of the triangle, which is the only point which sees the sides of ABC with the same angles (then 120° each):

```
>C=[-0.5,1];
>plot3d("d3",xmin=-2,xmax=2,ymin=-2,ymax=2):
```



```
>fcontour("d3",xmin=-2,xmax=2,ymin=-2,ymax=2,hue=1,title="The Fermat point");
>P=(A_B_C_A)'; plot2d(P[1],P[2],add=1,color=12);
>insimg;
```



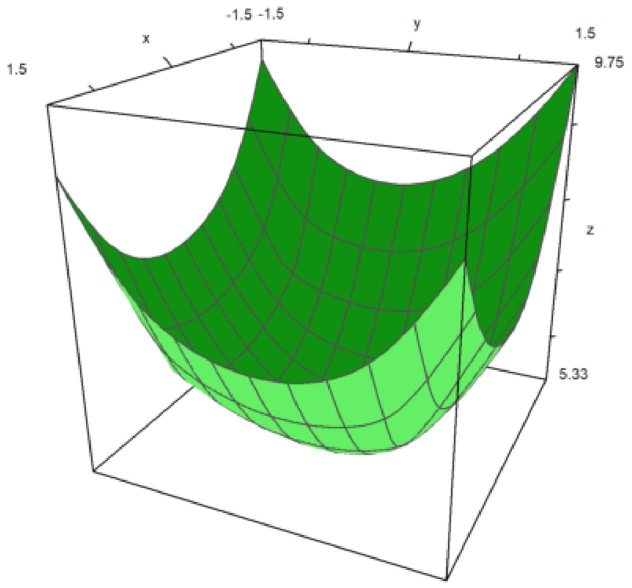
It is an interesting activity to realize the above figure with a geometry software; for example, I know a soft written in Java which has a "contour lines" instruction...

All of this above have been discovered by a french judge called Pierre de Fermat; he wrote letters to other dilettants like the priest Marin Mersenne and Blaise Pascal who worked at the income taxes. So the unique point F such that FA+FB+FC is minimal, is called the Fermat point of the triangle. But it seems that a few years before, the italian Torriccelli had found this point before Fermat did! Anyway the tradition is to note this point F...

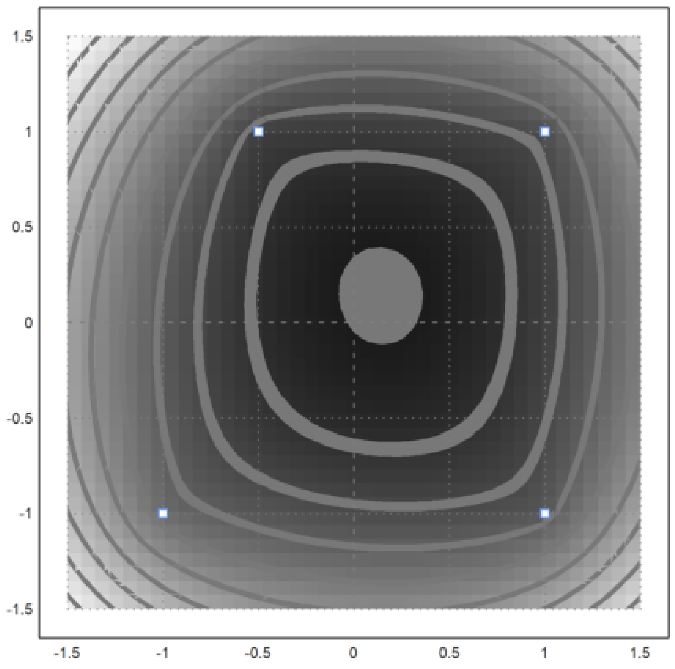
Four points

The next step is to add a 4th point D and try and minimize MA+MB+MC+MD; say that you are a cable TV operator and want to find in which field you must put your antenna so that you can feed four villages and use as little cable length as possible!

```
>D=[1,1];
>function d4(x,y):=d3(x,y)+sqrt((x-D[1])^2+(y-D[2])^2)
>plot3d("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5):
```



```
>fcontour("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5,hue=1);
>P=(A_B_C_D)'; plot2d(P[1],P[2],points=1,add=1,color=12);
>insimg;
```



There is still a minimum and it is attained at none of the vertices A, B, C nor D:

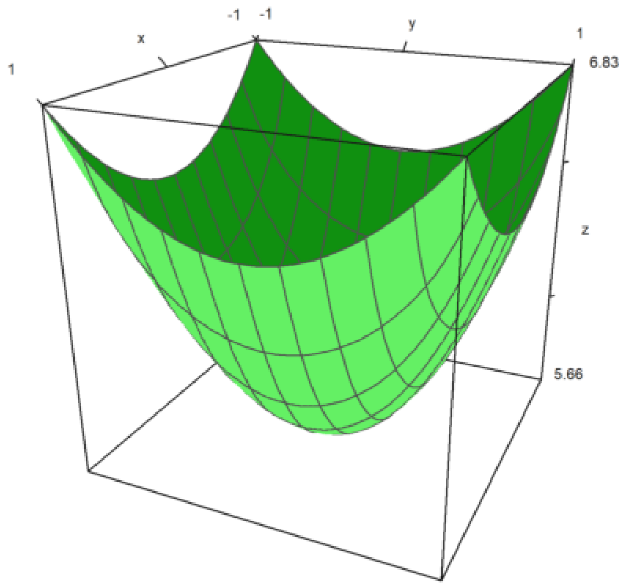
```
>function f(x):=d4(x[1],x[2])
>neldermin("f",[0.2,0.2])
```

```
    [0.142858,  0.142857]
```

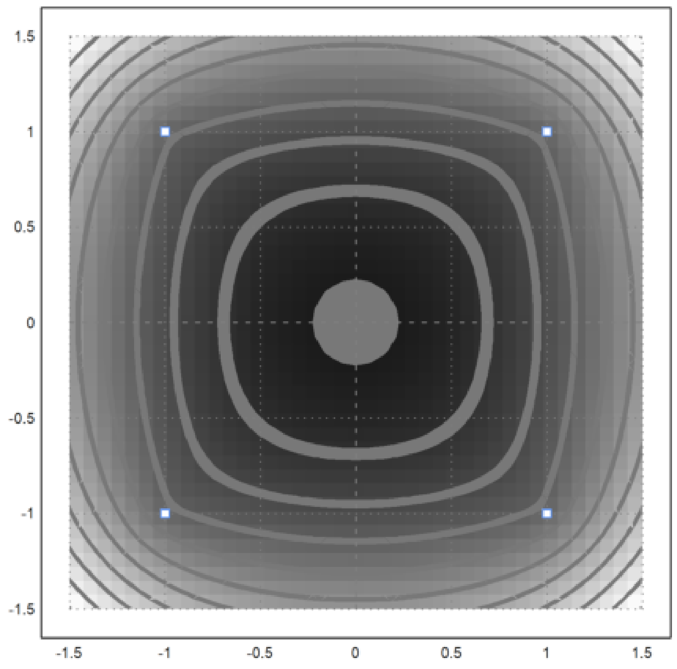
It seems that in this case, the coordinates of the optimal point are rational or near-rational...

Now ABCD is a square we expect that the optimal point will be the center of ABCD:

```
>C=[-1,1];
>plot3d("d4",xmin=-1,xmax=1,ymin=-1,ymax=1):
```



```
>fcontour("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5,hue=1);
>P=(A_B_C_D)'; plot2d(P[1],P[2],add=1,color=12,points=1);
>insimg;
```



Contoh 7: Bola Dandelin dengan Povray

You can run this demonstration, if you have Povray installed, and pvengine.exe in the program path.

First we compute the radii of the spheres.

If you look at the figure below, you see that we need two circles touching the two lines which form the cone, and one line which forms the plane cutting the cone.

We use the geometry.e file of Euler for this.

```
>load geometry;
```

First the two lines forming the cone.

```
>g1 &= lineThrough([0,0],[1,a])
```

```
[- a, 1, 0]
```

```
>g2 &= lineThrough([0,0],[-1,a])
```

```
[- a, - 1, 0]
```

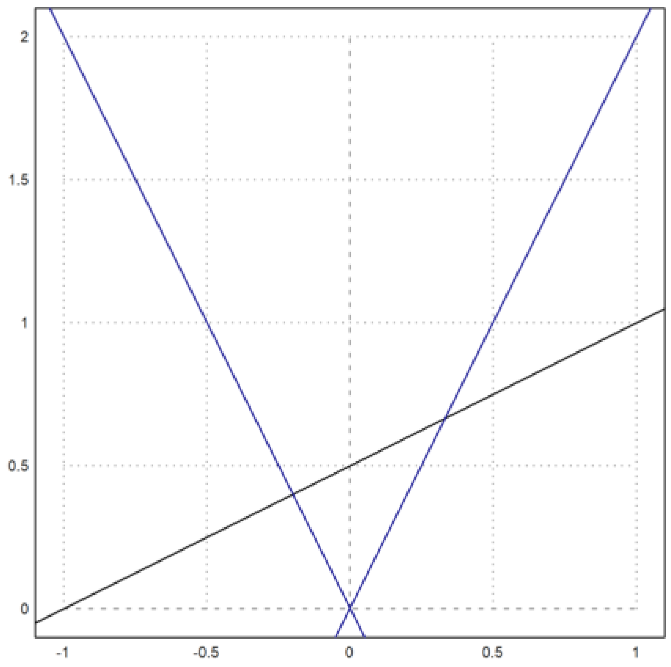
Thenm a third line.

```
>g &= lineThrough([-1,0],[1,1])
```

```
[- 1, 2, 1]
```

We plot everything so far.

```
>setPlotRange(-1,1,0,2);
>color(black); plotLine(g(),"")
>a:=2; color(blue); plotLine(g1(),""), plotLine(g2(),""):
```



Now we take a general point on the y-axis.

```
>P &= [0,u]
```

```
[0, u]
```

Compute the distance to g1.

```
>d1 &= distance(P,projectToLine(P,g1)); $d1
```

$$\sqrt{\left(\frac{a^2\,u}{a^2+1}-u\right)^2+\frac{a^2\,u^2}{\left(a^2+1\right)^2}}$$

Compute the distance to g.

```
>d &= distance(P,projectToLine(P,g)); $d
```

$$\sqrt{\left(\frac{u+2}{5}-u\right)^2+\frac{\left(2\,u-1\right)^2}{25}}$$

And find the centers of the two circles, where the distances are equal.

```
>sol &= solve(d1^2=d^2,u); $sol
```

$$\left[u=\frac{-\sqrt{5}\,\sqrt{a^2+1}+2\,a^2+2}{4\,a^2-1} , u=\frac{\sqrt{5}\,\sqrt{a^2+1}+2\,a^2+2}{4\,a^2-1} \right]$$

There are two solutions.

We evaluate the symbolic solutions, and find both centers, and both distances.

```
>u := sol()
```

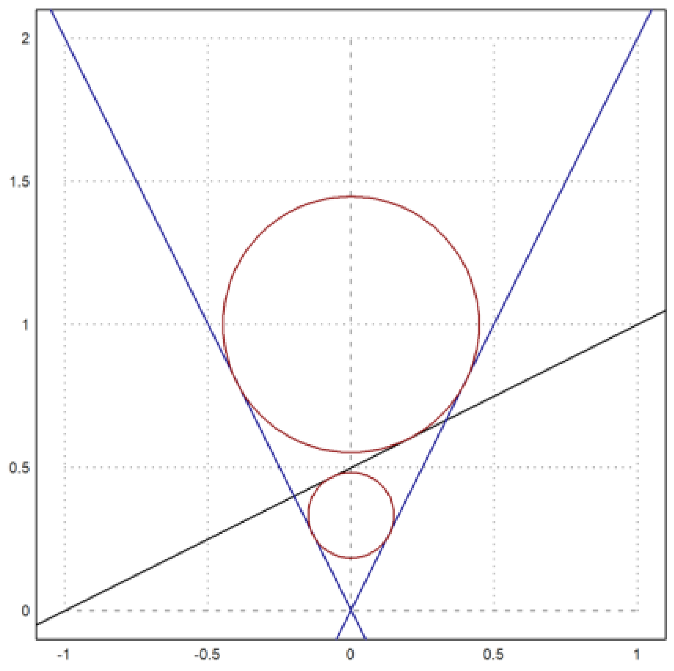
```
    [0.333333,  1]
```

```
>dd := d()
```

```
    [0.149071,  0.447214]
```

Plot the circles into the figure.

```
>color(red);
>plotCircle(circleWithCenter([0,u[1]],dd[1]),"");
>plotCircle(circleWithCenter([0,u[2]],dd[2]),"");
>insimg;
```



Plot with Povray

Next we plot everything with Povray. Note that you change any command in the following sequence of Povray commands, and rerun all commands with Shift-Return.

First we load the povray functions.

```
>load povray;
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

```
    C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe
```

We setup the scene appropriately.

```
>povstart(zoom=11,center=[0,0,0.5],height=10°,angle=140°);
```

Next we write the two spheres to the Povray file.

```
>writeln(povsphere([0,0,u[1]],dd[1],povlook(red)));
>writeln(povsphere([0,0,u[2]],dd[2],povlook(red)));
```

And the cone, transparent.

```
>writeln(povcone([0,0,0],0,[0,0,a],1,povlook(lightgray,1)));
```

We generate a plane restricted to the cone.

```
>gp=g();
>pc=povcone([0,0,0],0,[0,0,a],1,"");
>vp=[gp[1],0,gp[2]]; dp=gp[3];
>writeln(povplane(vp,dp,povlook(blue,0.5),pc));
```

Now we generate two points on the circles, where the spheres touch the cone.

```
>function turnz(v) := return [-v[2],v[1],v[3]]
>P1=projectToLine([0,u[1]],g1()); P1=turnz([P1[1],0,P1[2]]);
>writeln(povpoint(P1,povlook(yellow)));
>P2=projectToLine([0,u[2]],g1()); P2=turnz([P2[1],0,P2[2]]);
>writeln(povpoint(P2,povlook(yellow)));
```

Then we generate the two points where the spheres touch the plane. These are the foci of the ellipse.

```
>P3=projectToLine([0,u[1]],g()); P3=[P3[1],0,P3[2]];
>writeln(povpoint(P3,povlook(yellow)));
>P4=projectToLine([0,u[2]],g()); P4=[P4[1],0,P4[2]];
>writeln(povpoint(P4,povlook(yellow)));
```

Next we compute the intersection of P1P2 with the plane.

```
>t1=scalp(vp,P1)-dp; t2=scalp(vp,P2)-dp; P5=P1+t1/(t1-t2)*(P2-P1);
>writeln(povpoint(P5,povlook(yellow)));
```

We connect the points with line segments.

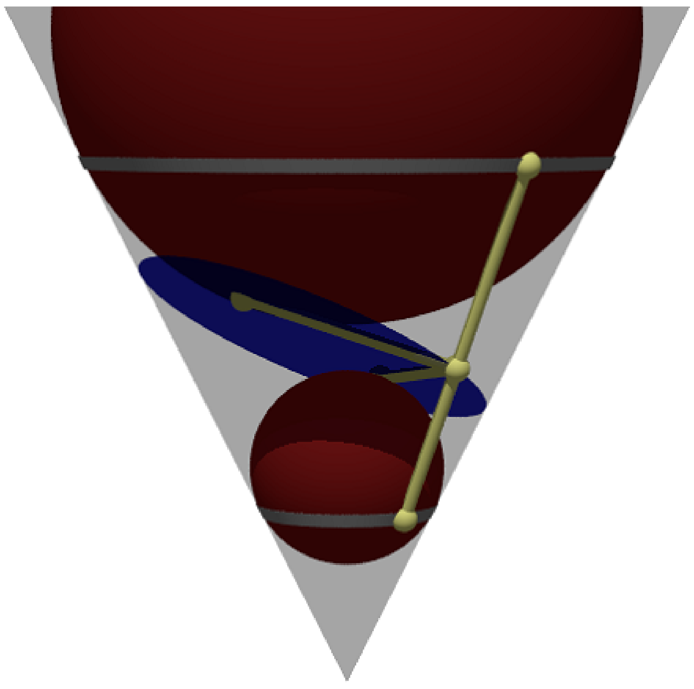
```
>writeln(povsegment(P1,P2,povlook(yellow)));
>writeln(povsegment(P5,P3,povlook(yellow)));
>writeln(povsegment(P5,P4,povlook(yellow)));
```

Now we generate a gray band, where the spheres touch the cone.

```
>pcw=povcone([0,0,0],0,[0,0,a],1.01);
>pc1=povcylinder([0,0,P1[3]-defaultpointsize/2],[0,0,P1[3]+defaultpointsize/2],1);
>writeln(povintersection([pcw,pc1],povlook(gray)));
>pc2=povcylinder([0,0,P2[3]-defaultpointsize/2],[0,0,P2[3]+defaultpointsize/2],1);
>writeln(povintersection([pcw,pc2],povlook(gray)));
```

Start the Povray program.

```
>povend();
```

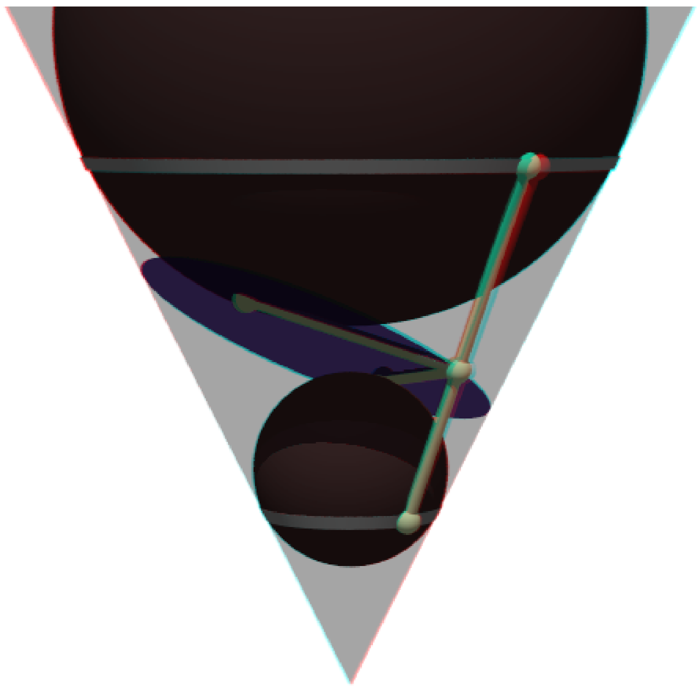


To get an Anaglyph of this we need to put everything into a scene function. This function will be used twice later.

```
>function scene () ...
 global a,u,dd,g,g1,defaultpointsize;
 writeln(povsphere([0,0,u[1]],dd[1],povlook(red)));
 writeln(povsphere([0,0,u[2]],dd[2],povlook(red)));
 writeln(povcone([0,0,0],0,[0,0,a],1,povlook(lightgray,1)));
 gp=g();
 pc=povcone([0,0,0],0,[0,0,a],1,"");
 vp=[gp[1],0,gp[2]]; dp=gp[3];
 writeln(povplane(vp,dp,povlook(blue,0.5),pc));
 P1=projectToLine([0,u[1]],g1()); P1=turnz([P1[1],0,P1[2]]);
 writeln(povpoint(P1,povlook(yellow)));
 P2=projectToLine([0,u[2]],g1()); P2=turnz([P2[1],0,P2[2]]);
 writeln(povpoint(P2,povlook(yellow)));
 P3=projectToLine([0,u[1]],g()); P3=[P3[1],0,P3[2]];
 writeln(povpoint(P3,povlook(yellow)));
 P4=projectToLine([0,u[2]],g()); P4=[P4[1],0,P4[2]];
 writeln(povpoint(P4,povlook(yellow)));
 t1=scalp(vp,P1)-dp; t2=scalp(vp,P2)-dp; P5=P1+t1/(t1-t2)*(P2-P1);
 writeln(povpoint(P5,povlook(yellow)));
 writeln(povsegment(P1,P2,povlook(yellow)));
 writeln(povsegment(P5,P3,povlook(yellow)));
 writeln(povsegment(P5,P4,povlook(yellow)));
 pcw=povcone([0,0,0],0,[0,0,a],1.01);
 pc1=povcylinder([0,0,P1[3]-defaultpointsize/2],[0,0,P1[3]+defaultpointsize/2],1);
 writeln(povintersection([pcw,pc1],povlook(gray)));
 pc2=povcylinder([0,0,P2[3]-defaultpointsize/2],[0,0,P2[3]+defaultpointsize/2],1);
 writeln(povintersection([pcw,pc2],povlook(gray)));
 endfunction
```

You need red/cyan glasses to appreciate the following effect.

```
>povanaglyph("scene",zoom=11,center=[0,0,0.5],height=10°,angle=140°);
```



Contoh 8: Geometri Bumi

In this notebook, we want to do some spherical computations. The functions are contained in the file "spherical.e" in the examples folder. We need to load that file first.

```
>load "spherical.e";
```

To enter a geographical position, we use a vector with two coordinates in radians (north and east, negative values for south and west). The following are the coordinates for the Campus of the FMIPA UNY.

```
>FMIPA=[rad(-7,-46.467),rad(110,23.05)]
```

```
    [-0.13569,  1.92657]
```

You can print this position with sposprint (spherical position print).

```
>sposprint(FMIPA) // posisi garis lintang dan garis bujur FMIPA UNY
```

```
 S 7°46.467' E 110°23.050'
```

Let us add two more towns, Solo and Semarang.

```
>Solo=[rad(-7,-34.333),rad(110,49.683)]; Semarang=[rad(-6,-59.05),rad(110,24.533)];
>sposprint(Solo), sposprint(Semarang),
```

```
 S 7°34.333' E 110°49.683'
 S 6°59.050' E 110°24.533'
```

First we compute the vector from one to the other on an ideal ball. This vector is [heading,distance] in radians. To compute the distance on the earth, we multiply with the earth radius at a latitude of 7°.

```
>br=svector(FMIPA,Solo); degprint(br[1]), br[2]*rearth(7°)->km // perkiraan jarak FMIPA-Solo
```

```
 65°20'26.60''
 53.8945384608
```

This is a good approximation. The following routines use even better approximations. On such a short distance the result is almost the same.

```
>esdist(FMIPA,Semarang)->" km" // perkiraan jarak FMIPA-Semarang
```

```
 Commands must be separated by semicolon or comma!
 Found:  // perkiraan jarak FMIPA-Semarang (character 32)
 You can disable this in the Options menu.
 Error in:
 esdist(FMIPA,Semarang)->" km" // perkiraan jarak FMIPA-Semaran ...
                              ^
```

There is a function for the heading, taking the elliptical shape of the earth into account. Again, we print in an advanced way.

```
>sdegprint(esdir(FMIPA,Solo))
```

```
      65.34°
```

The angle of a triangle exceeds 180° on the sphere.

```
>asum=sangle(Solo,FMIPA,Semarang)+sangle(FMIPA,Solo,Semarang)+sangle(FMIPA,Semarang,Solo); degprint(
```

```
 180°0'10.77''
```

This can be used to compute the area of the triangle. Note: For small triangles, this is not accurate due to the subtraction error in asum-pi.

```
>(asum-pi)*rearth(48°)^2->" km^2" // perkiraan luas segitiga FMIPA-Solo-Semarang
```

```
 Commands must be separated by semicolon or comma!
 Found:  // perkiraan luas segitiga FMIPA-Solo-Semarang (character 32)
 You can disable this in the Options menu.
 Error in:
 (asum-pi)*rearth(48°)^2->" km^2" // perkiraan luas segitiga FM ...
                                 ^
```

There is a function for this, which uses the mean latitude of the triangle to compute the earth radius, and takes care of rounding errors for very small triangles.

```
>esarea(Solo,FMIPA,Semarang)->" km^2", //perkiraan yang sama dengan fungsi esarea()
```

```
 2123.64310526 km^2
```

We can also add vectors to positions. A vector contains the heading and the distance, both in radians. To get a vector, we use svector. To add a vector to a position, we use saddvector.

```
>v=svector(FMIPA,Solo); sposprint(saddvector(FMIPA,v)), sposprint(Solo),
 S 7°34.333' E 110°49.683'
 S 7°34.333' E 110°49.683'
```

These functions assume an ideal ball. The same on the earth.

```
>sposprint(esadd(FMIPA,esdir(FMIPA,Solo),esdist(FMIPA,Solo))), sposprint(Solo),
```

```
 S 7°34.333' E 110°49.683'
 S 7°34.333' E 110°49.683'
```

Let us turn to a larger example, Tugu Jogja dan Monas Jakarta (menggunakan Google Earth untuk mencari koordinatnya).

```
>Tugu=[-7.7833°,110.3661°]; Monas=[-6.175°,106.811944°];
>sposprint(Tugu), sposprint(Monas)
```

```
 S 7°46.998' E 110°21.966'
 S 6°10.500' E 106°48.717'
```

According to Google Earth, the distance is 429.66km. We get a good approximation.

```
>esdist(Tugu,Monas)->" km" // perkiraan jarak Tugu Jogja - Monas Jakarta
```

```
 Commands must be separated by semicolon or comma!
 Found:  // perkiraan jarak Tugu Jogja - Monas Jakarta (character 32)
 You can disable this in the Options menu.
 Error in:
 esdist(Tugu,Monas)->" km" // perkiraan jarak Tugu Jogja - Mona ...
                          ^
```

The heading is the same as the one computed in Google Earth.

```
>degprint(esdir(Tugu,Monas))
```

```
 294°17'2.85''
```

However, we do no longer get the exact target position, if we add the heading and distance to the orginal position. This is so, since we do not compute the inverse function exactly, but take an approximation of the earth radius along the path.

```
>sposprint(esadd(Tugu,esdir(Tugu,Monas),esdist(Tugu,Monas)))
```

```
 S 6°10.500' E 106°48.717'
```

The error is not large, however.

```
>sposprint(Monas),
```

```
 S 6°10.500' E 106°48.717'
```

Of course, we cannot sail with the same heading from one destination to another, if we want to take the shortest path. Imagine, you fly NE starting at any point on the earth. Then you will spiral to the north pole. Great circles do not follow a constant heading!

The following computation shows that we are way off the correct destination, if we use the same heading during our travel.

```
>dist=esdist(Tugu,Monas); hd=esdir(Tugu,Monas);
```

Now we add 10 times one-tenth of the distance, using the heading to Monas, we got in Tugu.

```
>p=Tugu; loop 1 to 10; p=esadd(p,hd,dist/10); end;
```

The result is far off.

```
>sposprint(p), skmprint(esdist(p,Monas))
```

```
 S 6°11.250' E 106°48.372'
      1.529km
```

As another example, let us take two points on the earth at the same lattitude.

```
>P1=[30°,10°]; P2=[30°,50°];
```

The shortest path from P1 to P2 is not the circle of lattitude 30°, but a shorter path starting 10° further north at P1.

```
>sdegprint(esdir(P1,P2))
```

```
      79.69°
```

But, if we follow this compass reading, we will spiral to the north pole! So we must adjust our heading along the way. For rough purposes, we adjust it at 1/10 of the total distance.

```
>p=P1;  dist=esdist(P1,P2); ...
   loop 1 to 10; dir=esdir(p,P2); sdegprint(dir), p=esadd(p,dir,dist/10); end;
```

```
      79.69°
      81.67°
      83.71°
      85.78°
      87.89°
      90.00°
      92.12°
      94.22°
      96.29°
      98.33°
```

The distances are not right, since we will add a bit off error, if we follow the same heading for too long.

```
>skmprint(esdist(p,P2))
```

```
      0.203km
```

We get a good approximation, if we adjust out heading after each 1/100 of the total distance from Tugu to Monas.

```
>p=Tugu; dist=esdist(Tugu,Monas); ...
   loop 1 to 100; p=esadd(p,esdir(p,Monas),dist/100); end;
>skmprint(esdist(p,Monas))
```

```
      0.000km
```

For navigational purposes, we can get a sequence of GPS position along the great circle to Monas with the function navigate.

```
>load spherical; v=navigate(Tugu,Monas,10); ...
   loop 1 to rows(v); sposprint(v[#]), end;
```

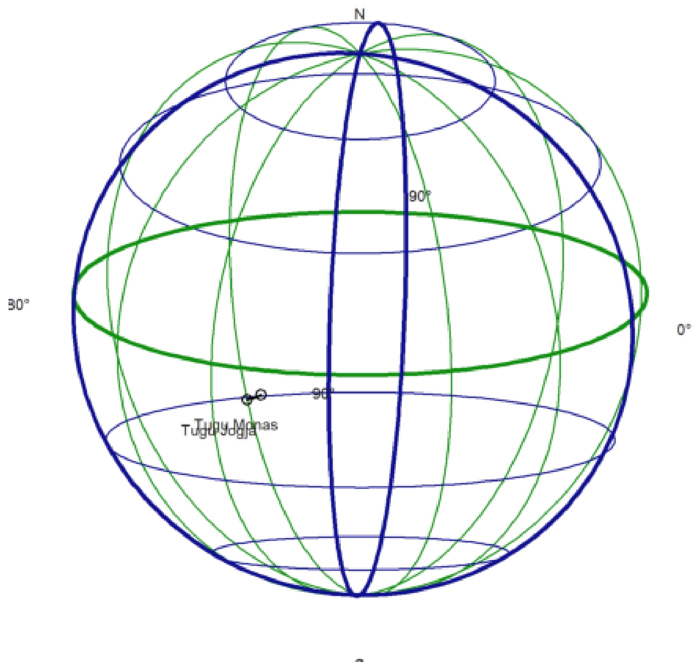
```
 S 7°46.998' E 110°21.966'
 S 7°37.422' E 110°0.573'
 S 7°27.829' E 109°39.196'
 S 7°18.219' E 109°17.834'
 S 7°8.592' E 108°56.488'
 S 6°58.948' E 108°35.157'
 S 6°49.289' E 108°13.841'
 S 6°39.614' E 107°52.539'
 S 6°29.924' E 107°31.251'
 S 6°20.219' E 107°9.977'
 S 6°10.500' E 106°48.717'
```

We write a function, which plots the earth, the two positions, and the positions in between.

```
>function testplot ...
 useglobal;
 plotearth;
 plotpos(Tugu,"Tugu Jogja"); plotpos(Monas,"Tugu Monas");
 plotposline(v);
 endfunction
```

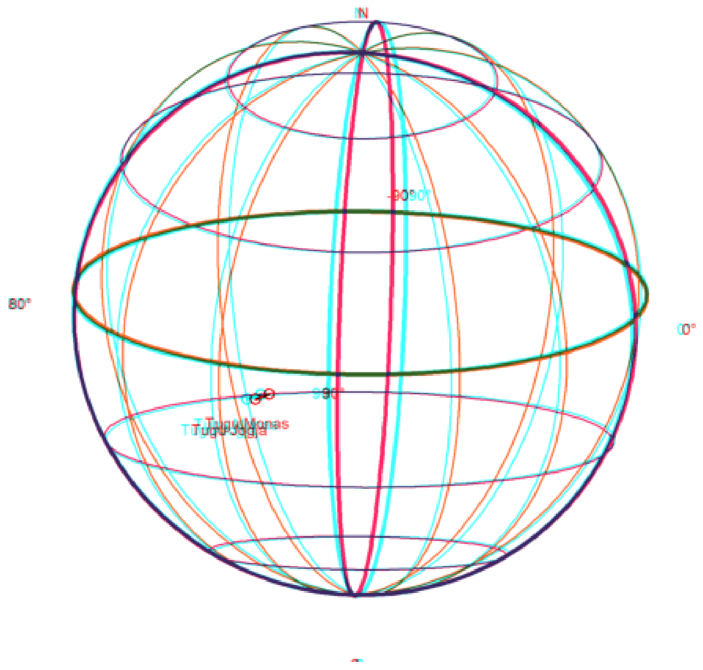
Now plot everything.

```
>plot3d("testplot",angle=25, height=6,>own,>user,zoom=4):
```



Or use plot3d to get an anaglyph view of it. This looks really great with red/cyan glasses.

```
>plot3d("testplot",angle=25,height=6,distance=5,own=1,anaglyph=1,zoom=4):
```



Latihan

1. Gambarlah segi-n beraturan jika diketahui titik pusat O, n, dan jarak titik pusat ke titik-titik sudut segi-n tersebut (jari-jari lingkaran luar segi-n), r.

Petunjuk:

- Besar sudut pusat yang menghadap masing-masing sisi segi-n adalah (360/n).
- Titik-titik sudut segi-n merupakan perpotongan lingkaran luar segi-n dan garis-garis yang melalui pusat dan saling membentuk sudut sebesar kelipatan (360/n).
- Untuk n ganjil, pilih salah satu titik sudut adalah di atas.
- Untuk n genap, pilih 2 titik di kanan dan kiri lurus dengan titik pusat.
- Anda dapat menggambar segi-3, 4, 5, 6, 7, dst beraturan.

2. Gambarlah suatu parabola yang melalui 3 titik yang diketahui.

Petunjuk:
- Misalkan persamaan parabolanya y= ax^2+bx+c.
- Substitusikan koordinat titik-titik yang diketahui ke persamaan tersebut.
- Selesaikan SPL yang terbentuk untuk mendapatkan nilai-nilai a, b, c.

3. Gambarlah suatu segi-4 yang diketahui keempat titik sudutnya, misalnya A, B, C, D.
- Tentukan apakah segi-4 tersebut merupakan segi-4 garis singgung (sisinya-sisintya merupakan garis singgung lingkaran yang sama yakni lingkaran dalam segi-4 tersebut).
- Suatu segi-4 merupakan segi-4 garis singgung apabila keempat garis bagi sudutnya bertemu di satu titik.
- Jika segi-4 tersebut merupakan segi-4 garis singgung, gambar lingkaran dalamnya.
- Tunjukkan bahwa syarat suatu segi-4 merupakan segi-4 garis singgung apabila hasil kali panjang sisi-sisi yang berhadapan sama.

4. Gambarlah suatu ellips jika diketahui kedua titik fokusnya, misalnya P dan Q. Ingat ellips dengan fokus P dan Q adalah tempat kedudukan titik-titik yang jumlah jarak ke P dan ke Q selalu sama (konstan).

5. Gambarlah suatu hiperbola jika diketahui kedua titik fokusnya, misalnya P dan Q. Ingat ellips dengan fokus P dan Q adalah tempat kedudukan titik-titik yang selisih jarak ke P dan ke Q selalu sama (konstan).

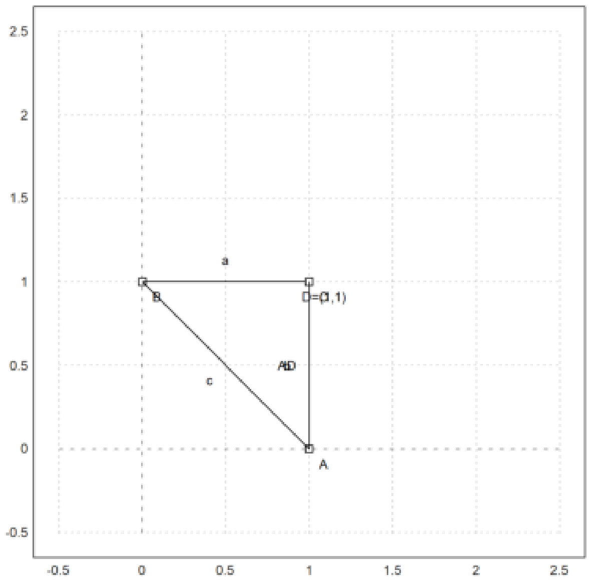
```
>load geometry
```

```
    Numerical and symbolic geometry.
```

```
>setPlotRange(-0.5,2.5,-0.5,2.5);
>A=[1,0]; plotPoint(A,"A");
>B=[0,1]; plotPoint(B,"B");
>C=[1,1]; plotPoint(C,"C");
>plotSegment(A,B,"c");
>plotSegment(B,C,"a");
>plotSegment(A,C,"b");
>lineThrough(B,C)
```

```
    [0,  1,  1]
```

```
>h=perpendicular(A,lineThrough(B,C));
>D=lineIntersection(h,lineThrough(B,C));
>plotPoint(D,value=1);
>aspect(1); plotSegment(A,D):
```



```
>norm(A-D)*norm(B-C)/2
```

```
    0.5
```

```
>areaTriangle(A,B,C)
```

```
    0.5
```

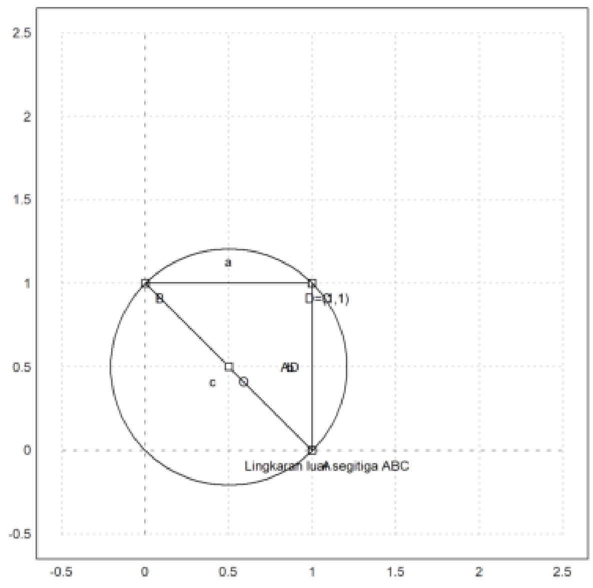
```
>distance(A,D)*distance(B,C)/2
```

```
    0.5
```

```
>degprint(computeAngle(B,C,A))
```

```
    90°
```

```
>c=circleThrough(A,B,C);
>R=getCircleRadius(c);
>O=getCircleCenter(c);
>plotPoint(O,"O");
>plotCircle(c,"Lingkaran luar segitiga ABC"):
```



```
>O, R
```

```
    [0.5,  0.5]
    0.707106781187
```

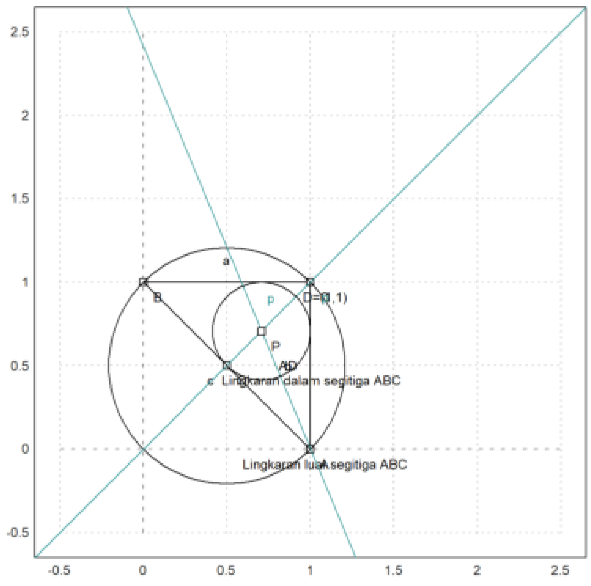
```
>l=angleBisector(A,C,B);
>g=angleBisector(C,A,B);
>P=lineIntersection(l,g)
```

```
    [0.707107,  0.707107]
```

```
>color(5); plotLine(l); plotLine(g); color(1);
>plotPoint(P,"P");
>r=norm(P-projectToLine(P,lineThrough(A,B)))
```

```
    0.292893218813
```

```
>plotCircle(circleWithCenter(P,r),"Lingkaran dalam segitiga ABC"):
```



```
// Define the points
x1 = 1; y1 = 0;
x2 = 2; y2 = 1;
x3 = 3; y3 = 4;

// Create the coefficient matrix and the constant vector
A = [x1^2, x1, 1; x2^2, x2, 1; x3^2, x3, 1];
b = [y1; y2; y3];

// Solve the system of equations
solution = A \ b;
a = solution[1];
b = solution[2];
c = solution[3];

// Define the parabola function
function y(x) := a*x^2 + b*x + c;

// Plot the parabola
plot2d("y(x)", -1, 4, -1, 5);
```

```
>p &= getHesseForm(lineThrough(A,B),x,y,C)-distance([x,y],C); $p='0
```

[illegible]

```
>p &= getHesseForm(lineThrough(A,B),x,y,C)-distance([x,y],C); $p='0
```

[illegible]

```
>p &= getHesseForm(ax^2+bx+c); $p='0
```

```
Maxima said:
Too few arguments supplied to getHesseForm(g,x,y,p); found:
       [c+bx+ax^2]
 -- an error. To debug this try: debugmode(true);

Error in:
p &= getHesseForm(ax^2+bx+c); $p='0 ...
                         ^
```

```
>load geometry
```

```
Numerical and symbolic geometry.
```

```
>x1 := point1[1]; y1 := point1[2];
```

```
point1 is not a variable!
Error in:
x1 := point1[1]; y1 := point1[2]; ...
               ^
```

```
>x2 := point2[1]; y2 := point2[2];
```

```
    point2 is not a variable!
    Error in:
    x2 := point2[1]; y2 := point2[2]; ...
                   ^
```

```
>x3 := point3[1]; y3 := point3[2];
```

```
    point3 is not a variable!
    Error in:
    x3 := point3[1]; y3 := point3[2]; ...
                   ^
```

```
>y1 = ax1^2 + bx1 + c
```

```
    Variable ax1 not found!
    Error in ^
    Error in:
    y1 = ax1^2 + bx1 + c ...
               ^
```

```
>y2 = ax2^2 + bx2 + c
```

```
    Variable ax2 not found!
    Error in ^
    Error in:
    y2 = ax2^2 + bx2 + c ...
               ^
```

```
>y3 = ax3^2 + bx3 + c
```

```
    Variable ax3 not found!
    Error in ^
    Error in:
    y3 = ax3^2 + bx3 + c ...
               ^
```

```
>A := [[x1^2, x1, 1], [x2^2, x2, 1], [x3^2, x3, 1]];
```

```
    Variable x1 not found!
    Error in ^
    Error in:
    A := [[x1^2, x1, 1], [x2^2, x2, 1], [x3^2, x3, 1]]; ...
               ^
```

```
>B := [y1, y2, y3];
```

```
    Variable y1 not found!
    Error in:
    B := [y1, y2, y3]; ...
            ^
```

```
>coeffs := A \ B; // Solusi SPL (A invers B)
```

```
    Need nxn matrix A, nxm matrix or vector b for \
    Error in:
    coeffs := A \ B; // Solusi SPL (A invers B) ...
                   ^
```

```
>a := coeffs[1];
```

```
coeffs is not a variable!
Error in:
a := coeffs[1]; ...
              ^
```

```
>b := coeffs[2];
```

```
coeffs is not a variable!
Error in:
b := coeffs[2]; ...
              ^
```

```
>c := coeffs[3];
```

```
coeffs is not a variable!
Error in:
c := coeffs[3]; ...
              ^
```

```
> print("Koefisien parabola: a = ", a, ", b = ", b, ", c = ", c);
```

```
Variable or function a not found.
Error in:
 print("Koefisien parabola: a = ", a, ", b = ", b, ", c = ", c ...
                                    ^
```

```
>x_range := -10..10;
```

```
Variable range not found!
Error in:
x_range := -10..10;  ...
        ^
```

```
>parabola_points := [];
```

```
Variable parabola not found!
Error in:
parabola_points := []; ...
                ^
```

```
>for x : x_range do
```

```
Syntax error in for!
Error in:
for x : x_range do ...
      ^
```

```
>y := a*x^2 + b*x + c;
```

```
Variable a not found!
Error in:
y := a*x^2 + b*x + c; ...
           ^
```

```
>parabola_points := append(parabola_points, [x, y]);
```

```
Variable parabola not found!
Error in:
parabola_points := append(parabola_points, [x, y]); ...
                ^
```

```
>plot2d(parabola_points, style="lines", title="Parabola through 3 points");
```

```
    plot2d([point1, point2, point3], style="dots", color="red", title="Titik-titik Diketahui");
end;
Variable parabola not found!
```

```
Error in:
plot2d(parabola_points, style="lines", title="Parabola through ...
                      ^
```