Задача № 9

```
with(ComputationalGeometry) :
xy := [[0, 2], [1, 0], [0, -2], [-1, 0]]
```

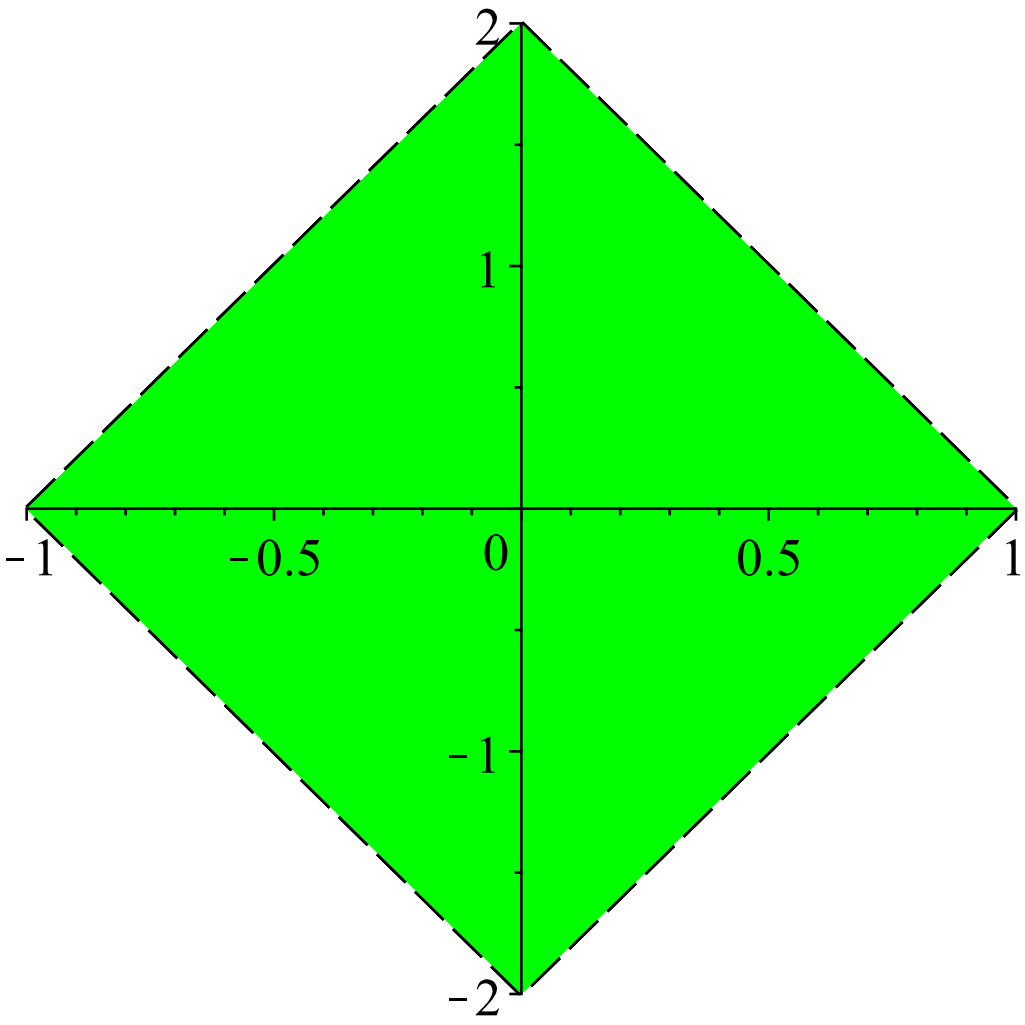
$$xy := [[0, 2], [1, 0], [0, -2], [-1, 0]] \quad \textbf{(1)}$$

```
h := ConvexHull(xy)
```

$$h := [1, 4, 3, 2] \quad \textbf{(2)}$$

```
with(plottools) :
with(plots) :
```

```
plots:-display(map(x→plottools:-polygon(xy), h), color = green, linestyle = dash, thickness = 2)
```




```
with(Student[MultivariateCalculus])
```

[&x, `.`, *Angle*, *ApproximateInt*, *ApproximateIntTutor*, *AreOrthogonal*, *AreParallel*, *AreSkew*, *BoxProduct*, *CenterOfMass*, *ChangeOfVariables*, *Contains*, *CrossProduct*, *CrossSection*, **(3)**

CrossSectionTutor, *Del*, *DirectionalDerivative*, *DirectionalDerivativeTutor*, *Distance*, *DotProduct*, *Equal*, *FunctionAverage*, *GetDimension*, *GetDirection*, *GetIntersection*, *GetNormal*, *GetPlot*, *GetPoint*, *GetRepresentation*, *Gradient*, *GradientTutor*, *Intersects*, *Jacobian*, *LagrangeMultipliers*, *Line*, *MultiInt*,∇, *Norm*, *Normalize*, *Plane*, *Projection*, *Revert*, *SecondDerivativeTest*, *SurfaceArea*, *TaylorApproximation*, *TaylorApproximationTutor*, *TripleScalarProduct*, *diff*]

( )

$linesList := [Line(xy[1], xy[2]), Line(xy[2], xy[3]), Line(xy[3], xy[4]), Line(xy[4], xy[1])]$

$linesList := [<<Line\ 189>>, <<Line\ 190>>, <<Line\ 191>>, <<Line\ 192>>]$ **(4)**

$AreParallel(linesList[3], linesList[2])$

$false$ **(5)**

$parallelList := [\,]$

$parallelList := [\,]$ **(6)**

.

```
for line1 in linesList do
 for line2 in linesList do
 if AreParallel(line1, line2) and not line1 = line2
 then parallelList := [op(parallelList), line1];
 parallelList := [op(parallelList), line2]
 end if
 end do
 end do
```

```
if numelems(parallelList) < 2
 then print(Разложения нет)
 else print(Разложение есть)
 end if
```

Разложение есть **(7)**

parallelList

[<<Line 189>>, <<Line 191>>, <<Line 190>>, <<Line 192>>, **(8)**
<<Line 191>>, <<Line 189>>, <<Line 192>>, <<Line 190>>]

( ) .

, , .

$pointsList1 := [\,]$

$pointsList1 := [\,]$ **(9)**

```
for p in xy do
 if Distance(parallelList[1], p) = 0 then
 pointsList1 := [op(pointsList1), p]
 end if
 end do
```

$pointsList2 := [\,]$

$$pointsList2 := [\,] \tag{10}$$

```
for p in xy do
 if Distance(parallelList[2], p) = 0 then
 pointsList2 := [op(pointsList2), p]
 end if
 end do
```

pointsList1

$$[[-1, 1], [1, 1]] \tag{11}$$

pointsList2

$$[[1, -1], [-1, -1]] \tag{12}$$

, (p o i n t s L i s t 1 , p o i n t s L i s t 2 -)

$$d1 := evalf(Distance(pointsList1[1], pointsList1[2]))$$

$$d1 := 2. \tag{13}$$

$$d2 := evalf(Distance(pointsList2[1], pointsList2[2]))$$

$$d2 := 2. \tag{14}$$

```
if d2 < d1 then
 temp := pointsList1;
 pointsList1 := pointsList2;
 pointsList2 := pointsList1;
 end if
```

- ,

$$xLittleNew := \frac{(pointsList1[1][1] + pointsList1[2][1])}{2}$$

$$xLittleNew := 0 \tag{15}$$

$$yLittleNew := \frac{(pointsList1[1][2] + pointsList1[2][2])}{2}$$

$$yLittleNew := 1 \tag{16}$$

$$newDot1 := [xLittleNew, yLittleNew]$$

$$newDot1 := [0, 1] \tag{17}$$

( )

$$x1mx2 := pointsList1[1][1] - pointsList1[2][1]$$

$$x1mx2 := -2 \tag{18}$$

$$y1my2 := pointsList1[1][2] - pointsList1[2][2]$$

$$y1my2 := 0 \tag{19}$$

$$pair1 := \left[\left[pointsList2[1][1] - \frac{(x1mx2)^2}{4}, pointsList2[1][2] - \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] - \frac{(x1mx2)^2}{4}, pointsList2[2][2] - \frac{(y1my2)^2}{4}\right]\right]$$

$$pair1 := [[0, -1], [-2, -1]] \quad (20)$$

$$Distance(pair1[1], pair1[2])$$

$$2 \quad (21)$$

$$pair2 := \left[\left[pointsList2[1][1] + \frac{(x1mx2)^2}{4}, pointsList2[1][2] - \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] - \frac{(x1mx2)^2}{4}, pointsList2[2][2] - \frac{(y1my2)^2}{4}\right]\right]$$

$$pair2 := [[2, -1], [-2, -1]] \quad (22)$$

$$pair3 := \left[\left[pointsList2[1][1] - \frac{(x1mx2)^2}{4}, pointsList2[1][2] + \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] - \frac{(x1mx2)^2}{4}, pointsList2[2][2] - \frac{(y1my2)^2}{4}\right]\right]$$

$$pair3 := [[0, -1], [-2, -1]] \quad (23)$$

$$pair4 := \left[\left[pointsList2[1][1] - \frac{(x1mx2)^2}{4}, pointsList2[1][2] - \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] + \frac{(x1mx2)^2}{4}, pointsList2[2][2] - \frac{(y1my2)^2}{4}\right]\right]$$

$$pair4 := [[0, -1], [0, -1]] \quad (24)$$

$$pair5 := \left[\left[pointsList2[1][1] - \frac{(x1mx2)^2}{4}, pointsList2[1][2] - \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] - \frac{(x1mx2)^2}{4}, pointsList2[2][2] + \frac{(y1my2)^2}{4}\right]\right]$$

$$pair5 := [[0, -1], [-2, -1]] \quad (25)$$

$$pair6 := \left[\left[pointsList2[1][1] + \frac{(x1mx2)^2}{4}, pointsList2[1][2] + \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] - \frac{(x1mx2)^2}{4}, pointsList2[2][2] - \frac{(y1my2)^2}{4}\right]\right]$$

$$pair6 := [[2, -1], [-2, -1]] \quad (26)$$

$$pair7 := \left[\left[pointsList2[1][1] - \frac{(x1mx2)^2}{4}, pointsList2[1][2] + \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] + \frac{(x1mx2)^2}{4}, pointsList2[2][2] - \frac{(y1my2)^2}{4}\right]\right]$$

$$pair7 := [[0, -1], [0, -1]] \quad (27)$$

$$pair8 := \left[\left[pointsList2[1][1] - \frac{(x1mx2)^2}{4}, pointsList2[1][2] - \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] + \frac{(x1mx2)^2}{4}, pointsList2[2][2] + \frac{(y1my2)^2}{4}\right]\right]$$

$$pair8 := [[0, -1], [0, -1]] \quad (28)$$

$$pair9 := \left[\left[pointsList2[1][1] + \frac{(x1mx2)^2}{4}, pointsList2[1][2] - \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] + \frac{(x1mx2)^2}{4}, pointsList2[2][2] - \frac{(y1my2)^2}{4}\right]\right]$$

$$pair9 := [[2, -1], [0, -1]] \quad (29)$$

$$pair10 := \left[\left[pointsList2[1][1] - \frac{(x1mx2)^2}{4}, pointsList2[1][2] + \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] - \frac{(x1mx2)^2}{4}, pointsList2[2][2] + \frac{(y1my2)^2}{4}\right]\right]$$

$$pair10 := [[0, -1], [-2, -1]] \quad (30)$$

$$pair11 := \left[\left[pointsList2[1][1] + \frac{(x1mx2)^2}{4}, pointsList2[1][2] - \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] - \frac{(x1mx2)^2}{4}, pointsList2[2][2] + \frac{(y1my2)^2}{4}\right]\right]$$

$$pair11 := [[2, -1], [-2, -1]] \quad (31)$$

$$pair12 := \left[\left[pointsList2[1][1] + \frac{(x1mx2)^2}{4}, pointsList2[1][2] + \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] + \frac{(x1mx2)^2}{4}, pointsList2[2][2] - \frac{(y1my2)^2}{4}\right]\right]$$

$$pair12 := [[2, -1], [0, -1]] \quad (32)$$

$$pair13 := \left[\left[pointsList2[1][1] - \frac{(x1mx2)^2}{4}, pointsList2[1][2] + \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] + \frac{(x1mx2)^2}{4}, pointsList2[2][2] + \frac{(y1my2)^2}{4}\right]\right]$$

$$pair13 := [[0, -1], [0, -1]] \quad (33)$$

$$pair14 := \left[\left[pointsList2[1][1] + \frac{(x1mx2)^2}{4}, pointsList2[1][2] - \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] + \frac{(x1mx2)^2}{4}, pointsList2[2][2] + \frac{(y1my2)^2}{4}\right]\right]$$

$$pair14 := [[2, -1], [0, -1]] \quad (34)$$

$$pair15 := \left[\left[pointsList2[1][1] + \frac{(x1mx2)^2}{4}, pointsList2[1][2] + \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] - \frac{(x1mx2)^2}{4}, pointsList2[2][2] + \frac{(y1my2)^2}{4}\right]\right]$$

$$pair15 := [[2, -1], [-2, -1]] \quad (35)$$

$$pair16 := \left[\left[pointsList2[1][1] + \frac{(x1mx2)^2}{4}, pointsList2[1][2] + \frac{(y1my2)^2}{4}\right], \left[pointsList2[2][1] + \frac{(x1mx2)^2}{4}, pointsList2[2][2] + \frac{(y1my2)^2}{4}\right]\right]$$

$$pair16 := [[2, -1], [0, -1]] \quad (36)$$

$$pairs := [pair1, pair2, pair3, pair4, pair5, pair6, pair7, pair8, pair9, pair10, pair11, pair12, pair13, pair14, pair15, pair16] :$$

```
with(ArrayTools) :
with(LinearAlgebra) :
distances := Vector( )
```

$$distances := [\ ] \tag{37}$$

```
for i from 1 to numelems(pairs) do
 Append(distances, Distance(pairs[i][1], pairs[i][2]) )
 end do:

minimum := min(distances)
```

$$minimum := 0 \tag{38}$$

```
for i from 1 to numelems(distances) do
 if distances[i] = minimum then  ind := i
 end if
 end do

                    :

 pairs[ind]
```

$$[[0, -1], [0, -1]] \tag{39}$$

```
xySet := convert(xy, set)
```

$$xySet := \{[-1, -1], [-1, 1], [1, -1], [1, 1]\} \tag{40}$$

```
pointsSet1 := convert(pointsList1, set)
```

$$pointsSet1 := \{[-1, 1], [1, 1]\} \tag{41}$$

```
pointsSet2 := convert(pointsList2, set)
```

$$pointsSet2 := \{[-1, -1], [1, -1]\} \tag{42}$$

```
newPointsSet := (xySet minus pointsSet1) minus pointsSet2
```

$$newPointsSet := \varnothing \tag{43}$$

```
newPointsList := convert(newPointsSet, list)
```

$$newPointsList := [\ ] \tag{44}$$

```
newPointsList2 := [ ]
```

$$newPointsList2 := [\ ] \tag{45}$$

```
dmin := min(d1, d2)
```

$$dmin := 2. \tag{46}$$

```
for p in newPointsList  do
 s := sign(p[1]);
 if (p[2] = 0) then
 xNewP := s·(abs(p[1]) - dmin/2);
 newPointsList2 := [op(newPointsList2), [xNewP, p[2]]] :
 elif (p[1] = 0) then
 yNewP := s·(abs(p[2]) - dmin/2);
 newPointsList2 := [op(newPointsList2), [p[1], yNewP]] :
else
 newPointsList2 := [op(newPointsList2), p] :

 end if
```

end do:

$newPointsList2 := [op(newPointsList2), newDot1]$:
if $d1 = d2$ **then**
$newPointsList2 := [op(newPointsList2), pairs[ind][1]]$:
else
$newPointsList2 := [op(newPointsList2), pairs[ind][1]]$:
$newPointsList2 := [op(newPointsList2), pairs[ind][2]]$:
end if

$$newPointsList2 := [[0, 1], [0, -1]] \quad \textbf{(47)}$$

,
( )

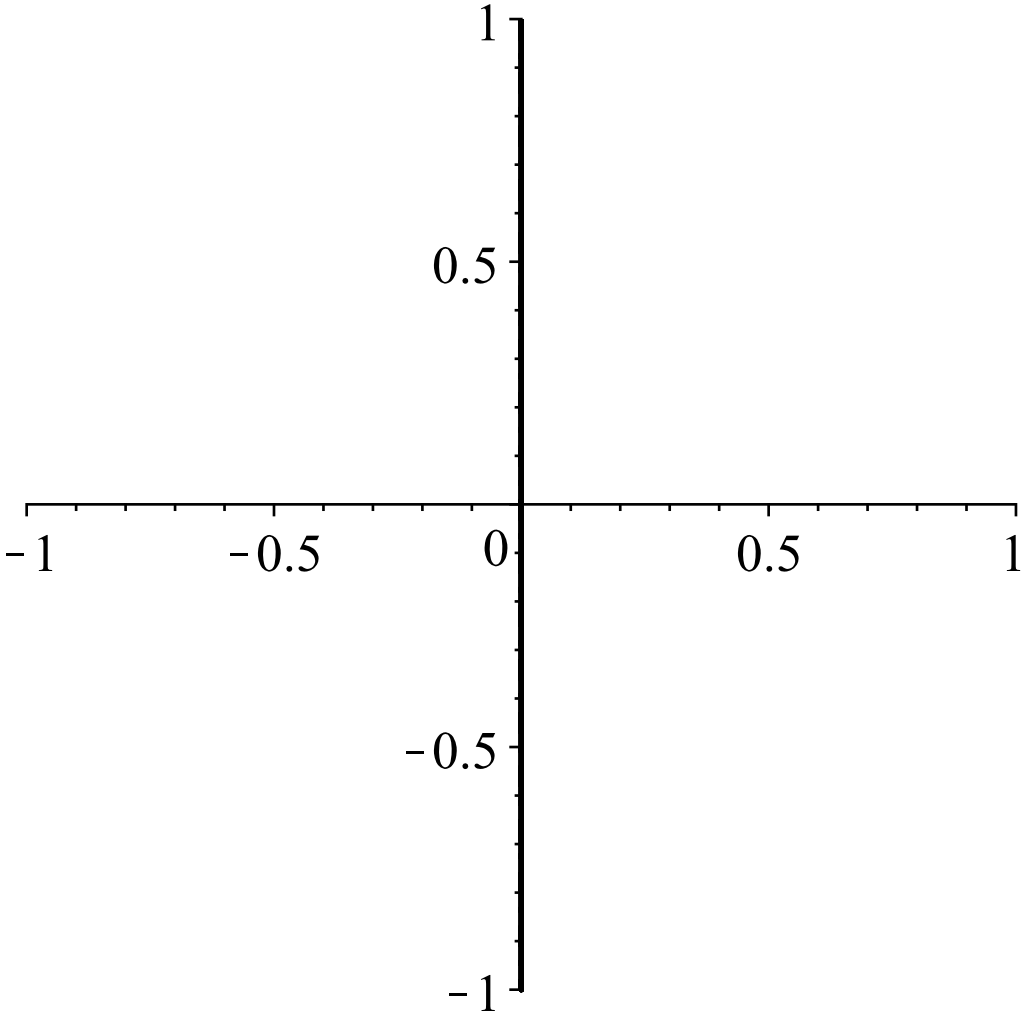
- .

.

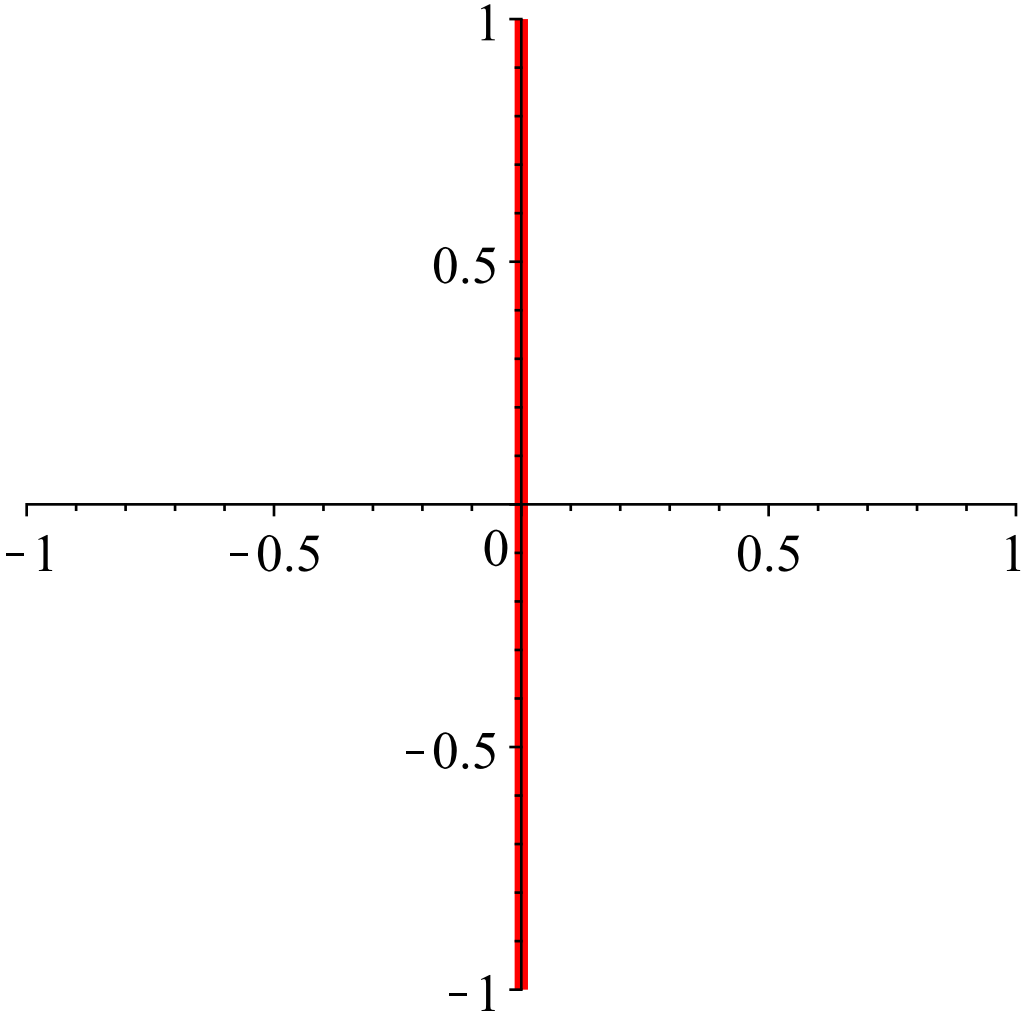
$h2 := ConvexHull(newPointsList2)$

Error, (in ComputationalGeometry:-ConvexHull) from Qhull: QH6214 qhull input error: not enough points(2) to construct initial simplex (need 3)

$plots:-display(map(x \rightarrow plottools:-polygon(newPointsList2), h2), color = red, linestyle = dash, thickness = 2)$



if (*numelems*(*newPointsList2*) = 2) **then**
display(*plottools*:-*line*(*newPointsList2*[1], *newPointsList2*[2], *color* = *red*, *thickness* = 5))
end if



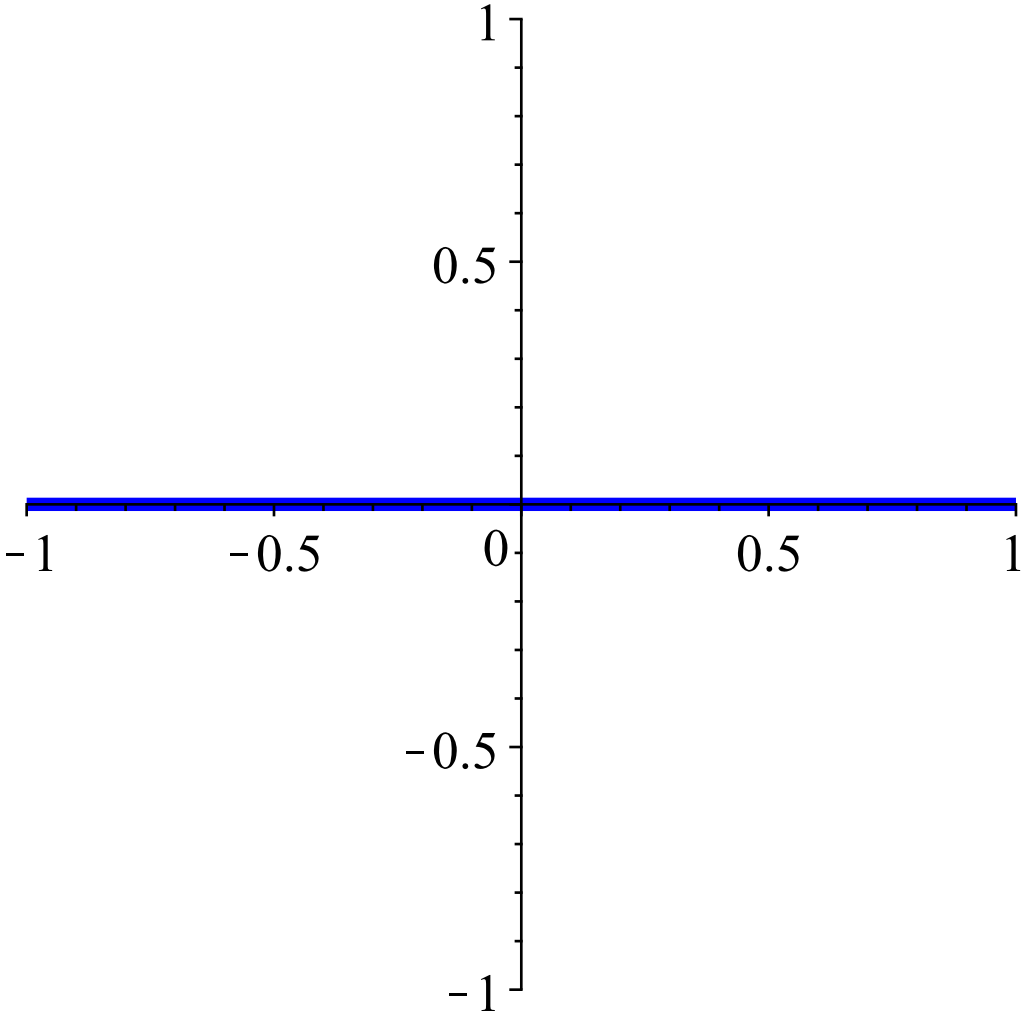

```
otrPoint1X := pointsList1[1][1] - newDot1[1]:
otrPoint1Y := pointsList1[1][2] - newDot1[2]:
otrPoint2X := pointsList1[2][1] - newDot1[1]:
otrPoint2Y := pointsList1[2][2] - newDot1[2]:

otrPoint1 := [otrPoint1X, otrPoint1Y]
```

$$otrPoint1 := [-1, 0] \quad \textbf{(48)}$$

```
otrPoint2 := [otrPoint2X, otrPoint2Y]
```

$$otrPoint2 := [1, 0] \quad \textbf{(49)}$$

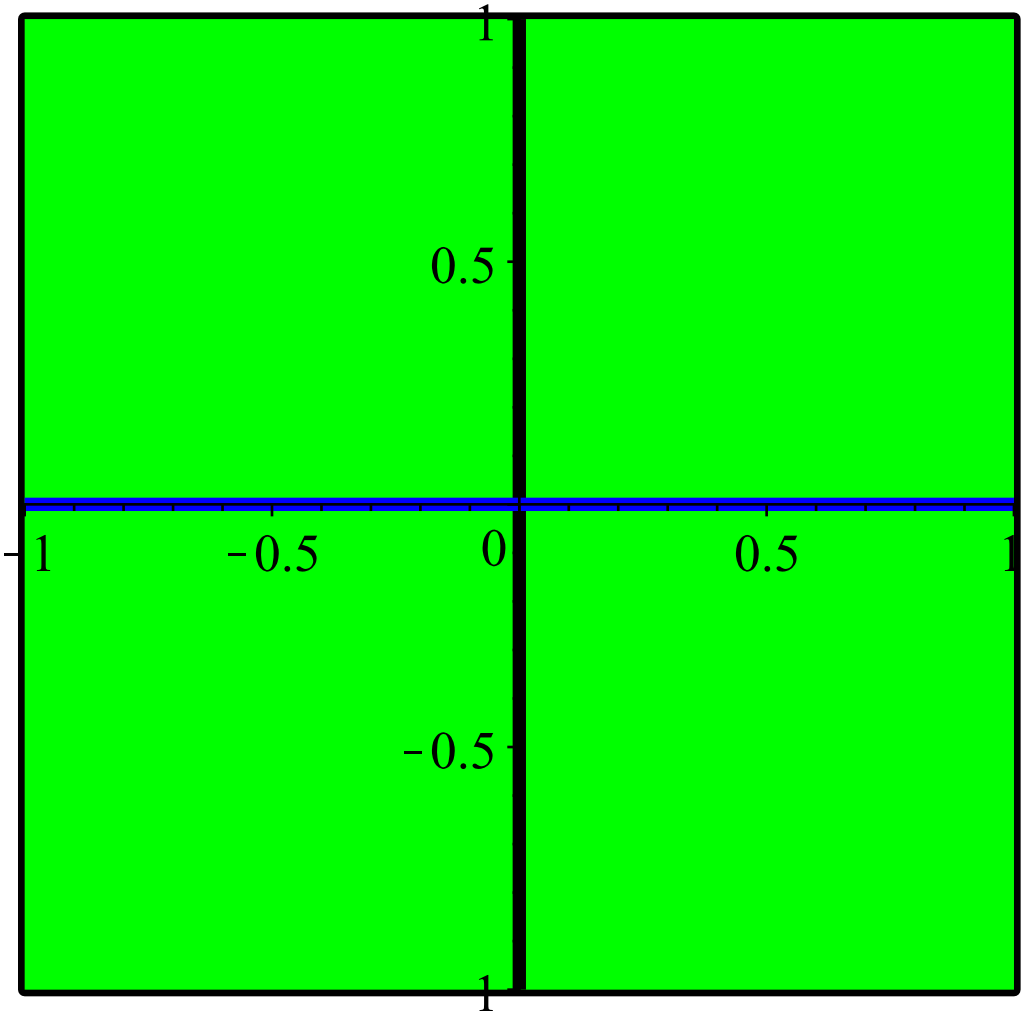
```
display(line(otrPoint1, otrPoint2, color = blue, thickness = 5))
```




```
for i from numelems(xy) to 1 by -1 do
 if not(member(i, h))
 then Remove(xy, i);
 end if
 end do

for i from numelems(newPointsList2) to 1 by -1 do
 if not(member(i, h2))
 then Remove(newPointsList2, i);
 end if
 end do
```

(-
, -)

display(*line*(*otrPoint1*, *otrPoint2*, *color* = *blue*, *thickness* = 5), *polygon*(*newPointsList2*, *color* = *red*, *thickness* = 5), *polygon*(*xy*, *color* = *green*, *transparency* = 0.80, *thickness* = 5))



( ) :

R X 0 , Y 0

$R := 1$:
$X0 := 0$:
$Y0 := 0$:
$X\_5 := [\,]$:
$Y\_5 := [\,]$:

for i **from** 1 **to** 5 **do**

$$X\_5 := \left[op(X\_5), evalf\left(X0 + R \cdot \cos\left(\frac{\pi}{5} \cdot (1 + 2 \cdot i) \right) \right) \right];$$

$$Y\_5 := \left[op(Y\_5), evalf\left(Y0 + R \cdot \sin\left(\frac{\pi}{5} \cdot (1 + 2 \cdot i) \right) \right) \right]$$

end do:

X_5

$$[-0.3090169938, -1., -0.3090169938, 0.8090169943, 0.8090169943] \quad \textbf{(50)}$$

```
points5 := [ ]:
for i from 1 to 5 do
points5 := [op(points5), [X_5[i], Y_5[i]]]
end do:
```

```
points5
```

$$[[-0.3090169938, 0.9510565165], [-1., 0.], [-0.3090169938, -0.9510565165], [0.8090169943, -0.5877852524], [0.8090169943, 0.5877852524]] \quad \textbf{(51)}$$