

Der Satz von Pick

Michael Nüsken, 15.01.2003.

Ein Gittervieck ist ein Polygon, dessen Ecken im Gitter Z^2 liegen.

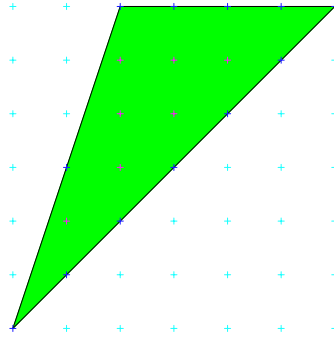
(Trifft dessen Kantenzug sich selbst, so nennen wir das Gittervieck ausgeartet. Im folgenden wollen wir nur nicht ausgeartete Gitterviecke betrachten.)

Der Satz von Pick stellt eine Beziehung zwischen

(a) der Fläche eines Gitterviecks und

(b) den Anzahlen der Gitterpunkte innerhalb des Polygons und auf dem Rand.

Wir wollen diese finden.



```
> restart;  
with(LinearAlgebra):  
with(plots):  
Warning, the name changecoords has been redefined
```

– Fläche eines Gitterviecks

Bestimme die Fläche eines Gitterviecks. Wir gehen davon aus, dass die Ecken in mathematisch positiver Richtung angegeben sind.

```
> area:=proc(A,B,C)  
  if nargs<3 then return 0; end if;  
  Determinant(convert([B-A,C-A],Matrix))/2 +  
  procname(A,C,args[4..-1]);  
end proc;
```

– Innere, äußere und Randpunkte eines Gitterviecks identifizieren und zählen

Bestimme, wo ein (Gitter-)Punkt bzgl. des Polygon liegt. (innerhalb (-1 oder +1), auf dem Rand (-1/2 oder +1/2) oder außerhalb (0).).

– Farbgebung und Bezeichnung der Antwortwerte

```
> col:=proc(a) COLOUR(HUE,evalf(.5+a/3)); end proc;  
txt:=proc(a) option remember; sprintf("%a",a); end proc:  
txt(-1):="-inside":  
txt(-1/2):="-margin":  
txt(0):="outside":  
txt(+1/2):="+margin":
```

```
txt(1):="+inside":
```

Position eines Punktes bezüglich einer Geraden

Wir bestimmen, ob X links(+1), auf(0) oder rechts(-1) der gerichteten Gerade von A nach B liegt.

Eine andere Interpretation ist diese: wir bestimmen die Umlaufrichtung, gesehen von X , wenn wir von A nach B laufen.

```
> lineposition:=proc(X,A,B)
    signum(Determinant(convert([A-X,B-X],Matrix)));
end proc:
orientation:=lineposition:
```

Erster Versuch ...

Berechne, ob X innerhalb oder außerhalb des Dreiecks A, B, C liegt. Die Orientierung des Dreiecks und die Kante C, A spielen eine Sonderrolle, daher werden weitere Informationen darüber bestimmt. Die Antwort p, sb, o besteht aus der Position p , welche 1 innerhalb, 0 auf dem Rand und -1 außerhalb beträgt, der Position sb von X bezüglich der Kante C, A und der Orientierung o des Dreiecks.

```
> extendedtriangleposition:=proc(X,A,B,C)
    option remember;
    local sa,sb,sc,s,o;
    if X in {A,B,C} then return 0; end if;
    sa:=lineposition(X,B,C);
    sb:=lineposition(X,C,A);
    sc:=lineposition(X,A,B);
    s:=sa+sb+sc; # Das ist das dreifache der Umlaufzahl des
    Weges A,B,C um X, falls X nicht auf diesem Weg liegt.
    o:=signum(s); # Falls abs(s)>=2 ist, ist das tatsächlich
    die Orientierung des Dreiecks.
    if s=3 or s=-3 then return 1,sb,s/3; end if;
    if s=2 or s=-2 then return 0,sb,o; end if;
    o:=lineposition(A,B,C); # Die Orientierung ergibt sich in
    diesem Fall nicht automatisch.
    return -1,sb,o;
end proc:
```

Die Position wird rekursiv bestimmt und das klappt jetzt auch meistens. // übersetzt auf neue Werte ...

```
> position1:=proc(X,A,B,C)
    option remember;
    local s1,s2,t,o;
    if nargs<4 then return FAIL; end if; # may be we can
    repair this case later
    if X in {args[2..-1]} then return 1/2; end if;
    s1,t,o:=extendedtriangleposition(X,A,B,C);
    if nargs=4 then return (s1+1)/2; end if;
    s2:=2*position(X,A,C,args[5..-1])-1;
    if s1=0 and s2=0 and t=0 then return (1+o)/2; end if;
    return (1-s1*s2)/2;
end proc:
```

Zweiter Versuch: Umlaufzahlen bzgl. Strecken

Aber mit der Umlaufzahlenmethode klappt's vielleicht noch viel besser!

```

> winkel:=proc(A,B)
  option remember;
  local a,b,d,s;
  #C:=[1,0];
  if convert(A,set)={0} then return FAIL; return
winkel(C,B)+pm*Pi/2; end if;
  if convert(B,set)={0} then return FAIL; return
winkel(A,C)+pm*Pi/2; end if;
  a:=convert(A,Vector);
  LinearAlgebra:-Norm(a,2);
a:=LinearAlgebra:-VectorScalarMultiply(a,1/%) ;
  b:=convert(B,Vector);
  LinearAlgebra:-Norm(b,2);
b:=LinearAlgebra:-VectorScalarMultiply(b,1/%) ;
  d:=signum(Determinant(<a|b>));
  s:=a.b;
  if d=0 then
    if s<0 then
      return pm*Pi;
    end if; # X liegt zwischen A und B!
    return 0;
  end if;
  evalf(d*arccos(s));
end proc;
umlaufzahl1:=proc(X,A,B)
  local w,i,p,C;
  if nargs<4 then return 0; end if;
  if member(X,[args[2..-1]],'p') then
    p:=p+1;
    C:=99998*X;
    if p>2 then C:=C+args[p-1]; else C:=C+args[-1]; end
  if;
  if p<nargs then C:=C+args[p+1]; else C:=C+args[2]; end
  if;
  C:=C/100000;
  w:=umlaufzahl1(C,args[2..p-1],args[p+1..-1]);
  i:=umlaufzahl1(C,args[2..-1]);
  return (w+i)/2+pm*abs(w-i)/2;
end if;
w:=winkel(args[-1]-X,A-X);
for i from 2 to nargs-1 do
  w:=w+winkel(args[i]-X,args[i+1]-X);
end do;
evalf(w/(2*Pi));
subsindets(2*%,float,round)/2;
userinfo(6,umlauf,'procname'(args)=%);
%
end proc;
> position2:=proc()
  return subs(pm=0,umlaufzahl1(args));
end proc;

```

– Dritter Versuch: Umlaufzahlen bezüglich Dreiecken

```

[ > umlaufgerade:=proc(X,A,B)
    (1+orientation(X,A,B))/2;
end proc;
[ > umlaufdreieck:=proc(X,A,B,C)
    option remember;
    local sa,sb,sc,s,o;
    if X in {A,B,C} then return orientation(A,B,C)/2; end
if;
    sa:=lineposition(X,B,C);
    sb:=lineposition(X,C,A);
    sc:=lineposition(X,A,B);
    s:=sa+sb+sc; # Das ist das dreifache der Umlaufzahl des
Weges A,B,C um X, falls X nicht auf diesem Weg liegt.
    o:=signum(s); # Falls abs(s)>=2 ist, ist das tatsächlich
die Orientierung des Dreiecks.
    if s=3 or s=-3 then return signum(s); end if;
    if s=2 or s=-2 then return 1/2*signum(s); end if;
    return 0;
end proc;
[ > umlaufzahl2:=proc(X,A)
    local p;
    if member(X,[0,args[2..-1]],'p') then
        if p=nargs then orientation(args[p-1..p],args[2]);
        elif p=2 then orientation(args[-1],args[p..p+1]);
        else orientation(args[p-1..p+1]);
        end if;
        return %/2+procname(X,args[2..p-1],args[p+1..-1]);
    end if;
    add( umlaufdreieck(X,A,args[i],args[i+1]), i=3..nargs-1
);
end proc;
[ > position3:=umlaufzahl2:
[ > position:=position3: # Beste Wahl

```

– Zählen der wesentlichen Punkte mit Werten

Erstelle eine Tabelle, die die Anzahlen der auftretenden Wert für `position` zählt.

```

[ > scanpositions:=proc()
    local L,x1,x2,y1,y2,x,y;
    L:=[map2(op,1,[args]),map2(op,2,[args])];
    x1:=min(op(L[1])); x2:=max(op(L[1]));
    y1:=min(op(L[2])); y2:=max(op(L[2]));
    L:=proc() option remember; 0 end proc;
    forget(L);
    #L(0),L(1);
    for x from x1 to x2 do
        for y from y1 to y2 do
            position([x,y],args);
            userinfo(4,scan,[x,y],txt(%), % );
            L(%):=L(%)+1;
        end do;
    end do;
    forget(L,0);
    op(4,op(L));

```

```

[ ] end proc:
[ ] > pick:=proc()
[ ]     local L,iL;
[ ]     L:=scanpositions(args);
[ ]     iL:=map(op,[indices(L)]);
[ ]     map( x->x*L[x], iL );
[ ]     convert( %, '+' ) - 1;
[ ] end proc:
[ ] >

```

– Zeichenhilfen

```

[ ] > colouredgridplot:=proc(X)
[ ]     local L,x1,x2,y1,y2,x,y;
[ ]     L:=[map2(op,1,X),map2(op,2,X)];
[ ]     x1:=min(op(L[1])); x2:=max(op(L[1]));
[ ]     y1:=min(op(L[2])); y2:=max(op(L[2]));
[ ]     L:=proc() option remember; NULL end proc;
[ ]     forget(L);
[ ]     for x from x1 to x2 do
[ ]         for y from y1 to y2 do
[ ]             position([x,y],op(X));
[ ]             L(%) := L(%), [x,y];
[ ]         end do;
[ ]     end do;
[ ]     L:=op(4,op(L));
[ ]     map( op, [indices(L)] );
[ ]     map( x->pointplot( {L[x]}, color=col(x), legend=txt(x),
[ ]     args[2..-1] ), % );
[ ]     display(%);
[ ] end proc:
[ ] > show:=proc()
[ ]     print(display([
[ ]         colouredgridplot([args]),
[ ]         polygonplot([args],color=green)
[ ]     ],
[ ]     axes=BOXED,
[ ]     scaling=CONSTRAINED,
[ ]     symbolsize=25
[ ]     ));
[ ] end proc:
[ ] >

```

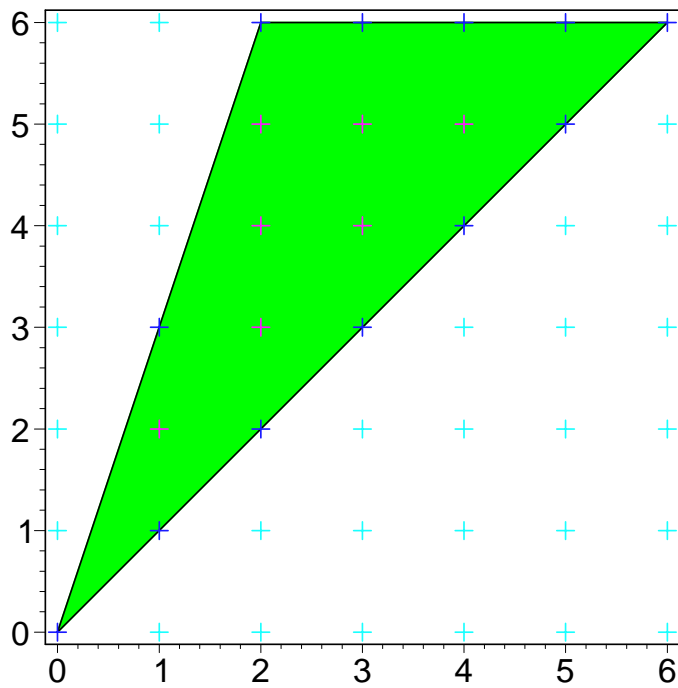
– Suche nach dem Satz von Pick

```

[ ] > ex:=[0,0],[6,6],[2,6];
[ ] show(ex);

```

$ex := [0, 0], [6, 6], [2, 6]$



+++++++ outside
 +++++++ +inside
 +++++++ +margin

```

> for j to 7 do
  ex:=[0,0],[j,j],[2,6];

  a,i,r:=area(ex),scanpositions(ex)[1],scanpositions(ex)[1/2];
  print( a = '?'(i,r) );
end do:

      2=?(1,4)
      4=?(1,8)
      6=?(4,6)
      8=?(5,8)
     10=?(7,8)
     12=?(7,12)
     14=?(10,10)

> EQ:=NULL:
  for j to 7 do
    ex:=[0,0],[j,j],[2,6];
  
```

```

a,i,r:=area(ex),scanpositions(ex)[1],scanpositions(ex)[1/2];
EQ:=EQ, a=alpha*i+beta*r+delta;
end do:
solve({EQ}):
'?`('i','r')=subs(%, alpha*'i'+beta*'r'+delta);

```

$$F(i, r) = i + \frac{r}{2} - 1$$

[>

Der Satz von Pick!

Sei V ein nicht-ausgeartetes Gittervieleck mit i inneren und r Randpunkten. Dann gilt

$$F(V) = i + \frac{r}{2} - 1.$$

[>

Beispiele und Gegenbeispiele

```

> a:='a':b:='b':
X:=<<r,s>|<t,u>>:
Determinant(X);
ex1:=[0,0],[1,2],[4,2],[2,3],[3,5],[3,7],[1,9],[-2,8],[-1,5],
[-3,3],[-2,3],[-2,-2],[0,3]:
ex1b:=[0,0],[1,2],[4,2],[2,3],[3,5],[3,7],[-2,8],[1,9],[-1,5],
[-3,3],[-2,3],[-2,-2],[0,3]:
ex2g:=[0,0],[a,0],[a,b]:
ex2:=op(subs({a=6,b=4},[ex2g])):
ex3g:=[0,0],[a,0],[c,d],[c-a,d]:
ex3:=op(subs({a=4,c=6,d=4},[ex3g])):
ex4:=[5,3],[3,5],[-5,2],[0,1],[1,1],[-3,3],[-5,5],[-5,-4],[-1,-3],
[5,-3],[1,3],[5,-4],[4,-1]:
ex5:=[5,3],[3,5],[-4,2],[0,1],[1,1],[-3,3],[-5,5],[-5,-4],[-1,-3],
[5,-3],[1,3],[5,-4],[4,-1]:
ex6:=[0,0],[3,0],[3,3],[1,2],[1,1],[2,1],[2,3],[0,3]:
ex7:=[0,0],[5,0],[5,5],[3,4],[2,4],[2,2],[3,2],[3,3],[1,4],[1,1],
[4,1],[4,5],[0,5]:
ex8:=[0,-1],[-1,-1],[-1,0],[5,0],[5,5],[3,4],[2,4],[2,2],[3,2],
[3,3],[1,4],[1,1],[4,1],[4,5],[0,5]:
ex9:=op(map(x->[-1,1]-x,[ex7])),ex6:
ex10:=op(map(x->[-1,1]-[x[1],-x[2]],[ex7])),ex6:

```

$$10u - ts$$

Wir nehmen jetzt die Werte des dritten Versuchs.

```
> position:=position3:
```

Beispiele testen.

```

> select(proc(x) local X; X:=convert(x,string); X[1..2]="ex"
and
length(X)>2 and
{}=convert(convert(X[3..-1],list),set) minus
convert(convert("0123456789",list),set); end,[anames()]):
for ex in eval(%,1) do
print(eval(ex,1)=ex);
pick(ex)=area(ex);

```

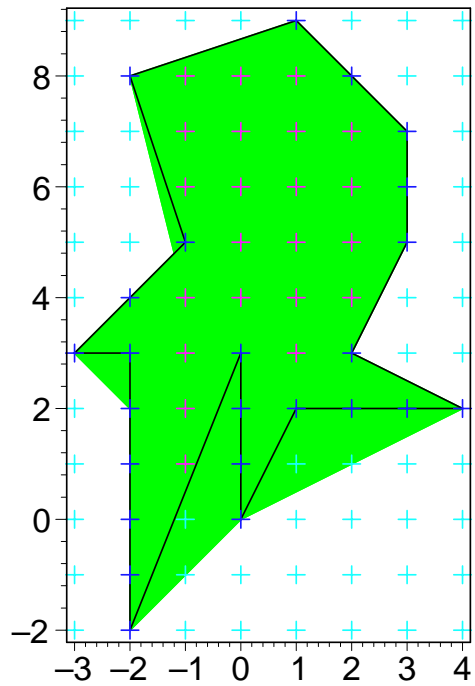
```

print(%);
if not % then
  printf("Picks Formel hat einen FEHLER von %a:",
    lhs(%)-rhs(%));
end if;
show(ex);
end do:

```

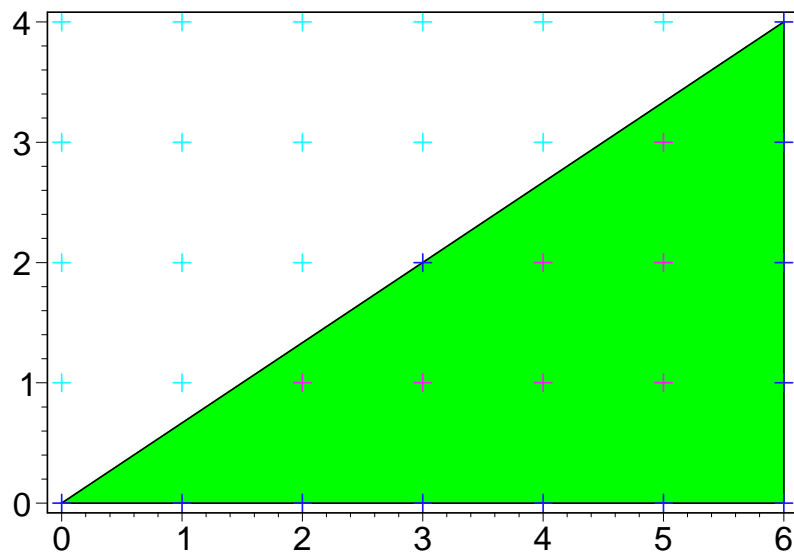
ex1 =

([0, 0], [1, 2], [4, 2], [2, 3], [3, 5], [3, 7], [1, 9], [-2, 8], [-1, 5], [-3, 3], [-2, 3], [-2, -2], [0, 3])
 33 = 33



+++++ outside
 +++++ +inside
 +++++ +margin

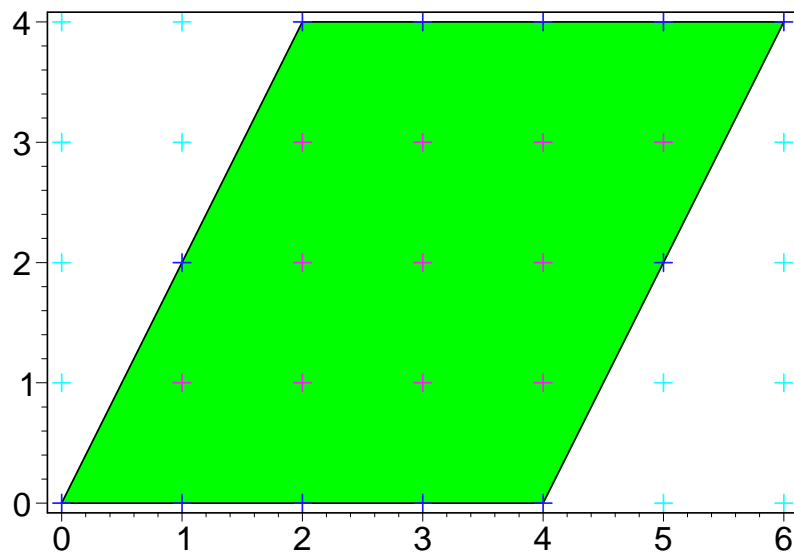
ex2 = ([0, 0], [6, 0], [6, 4])
 12 = 12



+ + + + + + outside
 + + + + + + +inside
 + + + + + + +margin

$$ex3 = ([0, 0], [4, 0], [6, 4], [2, 4])$$

$$16 = 16$$

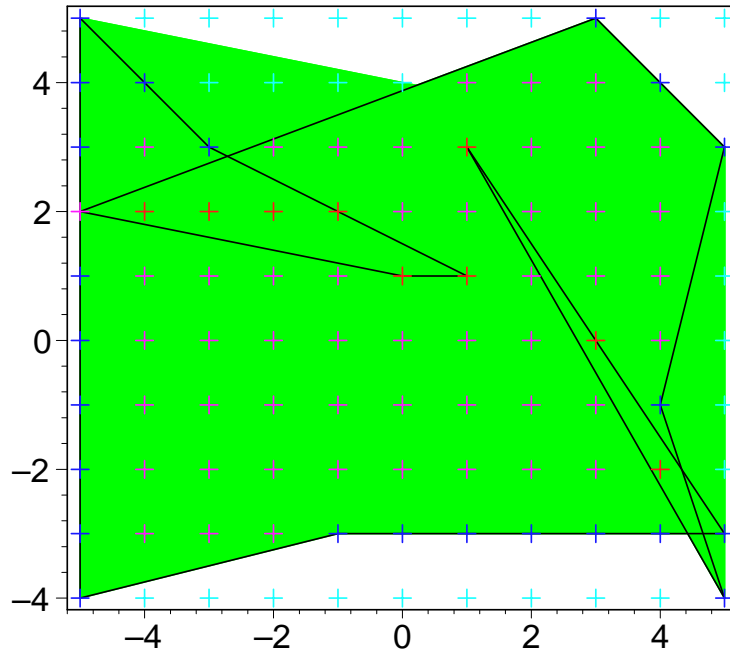


+ + + + + + outside
 + + + + + + +inside
 + + + + + + +margin

$ex4 = ([5, 3], [3, 5], [-5, 2], [0, 1], [1, 1], [-3, 3], [-5, 5], [-5, -4], [-1, -3], [5, -3], [1, 3],$
 $[5, -4], [4, -1])$

$$76 = 74$$

Picks Formel hat einen FEHLER von 2:

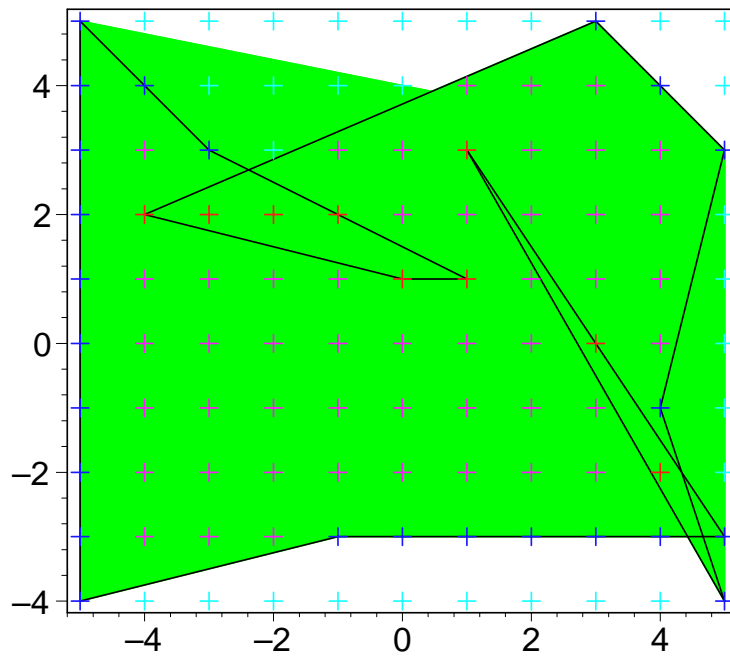


++++++ outside
 ++++++ +inside
 ++++++ 2
 ++++++ +margin
 ++++++ 3/2

$ex5 = ([5, 3], [3, 5], [-4, 2], [0, 1], [1, 1], [-3, 3], [-5, 5], [-5, -4], [-1, -3], [5, -3], [1, 3],$
 $[5, -4], [4, -1])$

$74 = 72$

Picks Formel hat einen FEHLER von 2:

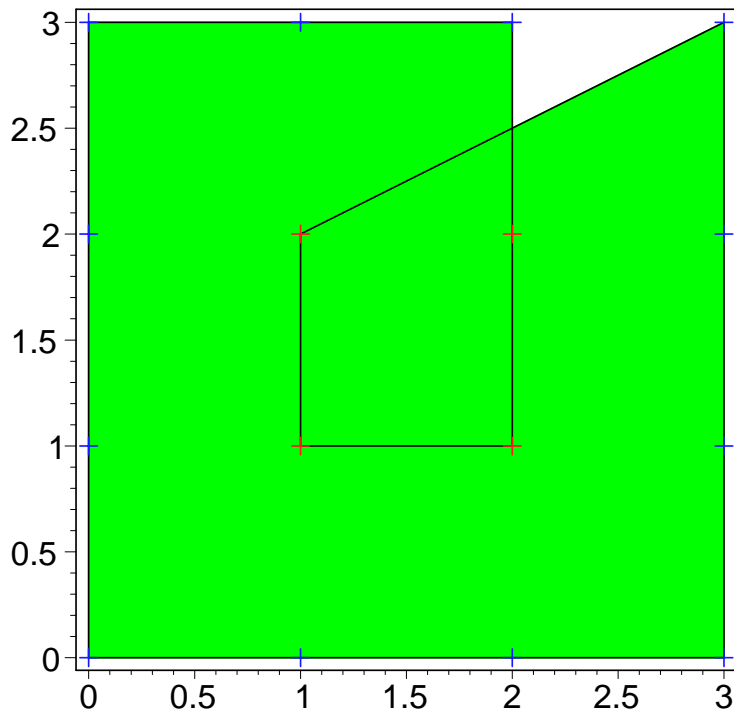


+ + + + + outside
 + + + + + +inside
 + + + + + 2
 + + + + + +margin
 + + + + + 3/2

$ex6 = ([0, 0], [3, 0], [3, 3], [1, 2], [1, 1], [2, 1], [2, 3], [0, 3])$

$$11 = 10$$

Picks Formel hat einen FEHLER von 1:



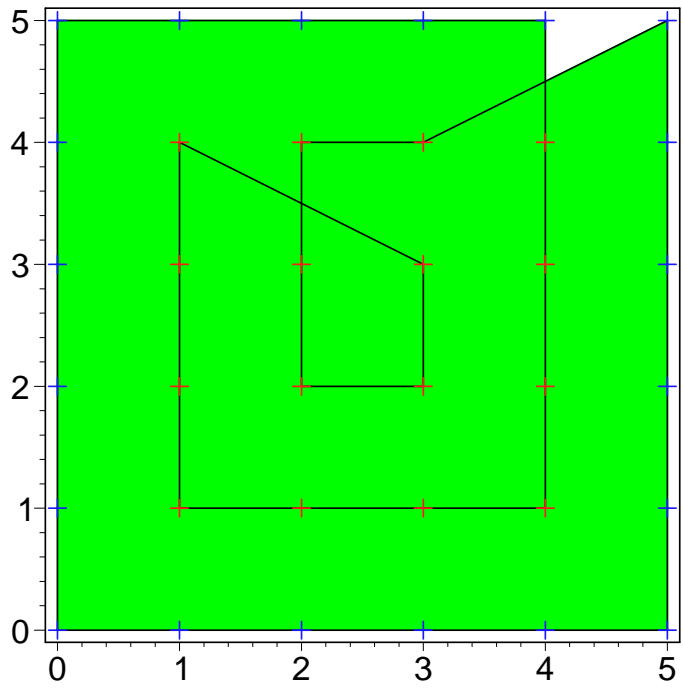
+++++ +margin
 +++++ 3/2

ex7=

([0, 0], [5, 0], [5, 5], [3, 4], [2, 4], [2, 2], [3, 2], [3, 3], [1, 4], [1, 1], [4, 1], [4, 5], [0, 5])

37 = 35

Picks Formel hat einen FEHLER von 2:

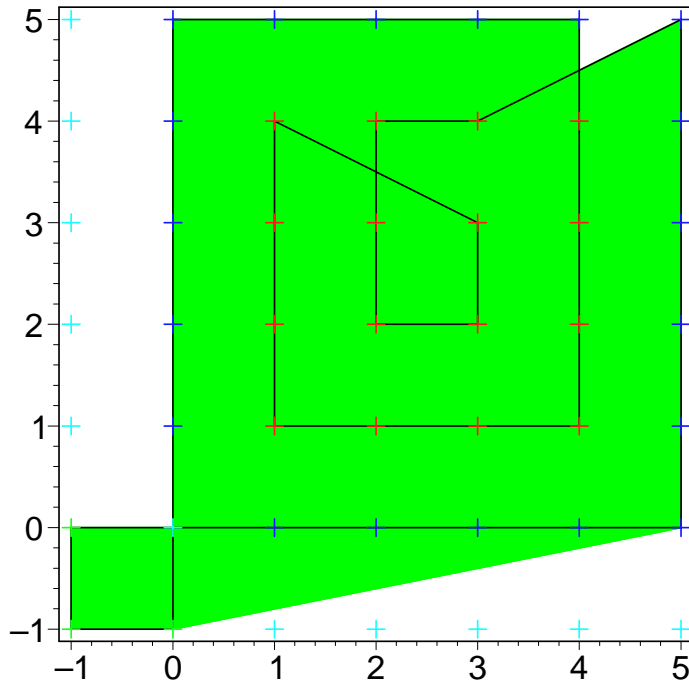


++++++ +margin
 ++++++ 3/2
 ++++++ 5/2

$ex8 = ([0, -1], [-1, -1], [-1, 0], [5, 0], [5, 5], [3, 4], [2, 4], [2, 2], [3, 2], [3, 3], [1, 4], [1, 1],$
 $[4, 1], [4, 5], [0, 5])$

$$35 = 34$$

Picks Formel hat einen FEHLER von 1:

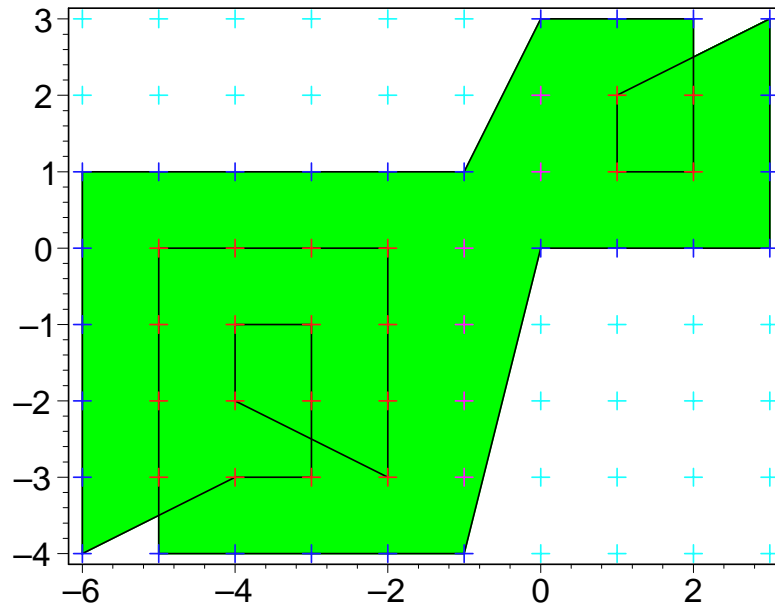


++++++ outside
 ++++++ -margin
 ++++++ +margin
 ++++++ 3/2
 ++++++ 5/2

$ex9 = ([-1, 1], [-6, 1], [-6, -4], [-4, -3], [-3, -3], [-3, -1], [-4, -1], [-4, -2], [-2, -3], [-2, 0], [-5, 0],$
 $[-5, -4], [-1, -4], [0, 0], [3, 0], [3, 3], [1, 2], [1, 1], [2, 1], [2, 3], [0, 3])$

$$52 = 49$$

Picks Formel hat einen FEHLER von 3:

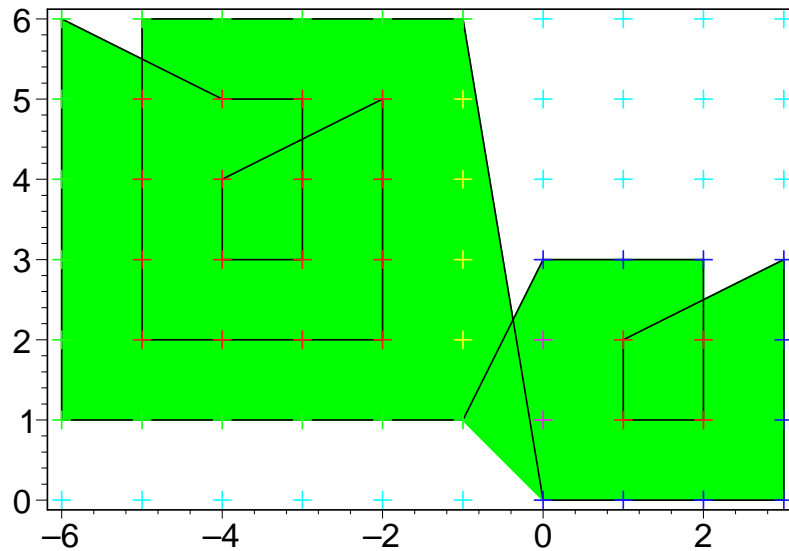


++ + outside
 ++ + inside
 ++ + margin
 ++ + 3/2
 ++ + 5/2

$ex10 = ([-1, 1], [-6, 1], [-6, 6], [-4, 5], [-3, 5], [-3, 3], [-4, 3], [-4, 4], [-2, 5], [-2, 2], [-5, 2],$
 $[-5, 6], [-1, 6], [0, 0], [3, 0], [3, 3], [1, 2], [1, 1], [2, 1], [2, 3], [0, 3])$

$-28 = -26$

Picks Formel hat einen FEHLER von -2:



```

+++++ outside
+++++ +inside
+++++ -margin
+++++ +margin
+++++ -5/2
+++++ 3/2

```

Möglichkeit zur Anpassung von `pick`, um auch für Beispiele mit Selbstüberschneidungen noch funktionieren.

```

> pick:=proc()
  local L,iL;
  L:=scanpositions(args);
  iL:=map(op,[indices(L)]);
  map( x->x*L[x], iL );
  convert( %, '+' )
  - 1; # Originalwert
  #- convert(map(signum,convert(iL,set)),'+'); # geratene
Korrektur
  #- ( max(0,op(iL))+min(0,op(iL)) ); # geratene Korrektur
end proc:

```

Beispiele testen, NUR Fehler ausgeben.

```

> select(proc(x) local X; X:=convert(x,string); X[1..2]="ex"
and
  length(X)>2 and
  {}=convert(convert(X[3..-1],list),set) minus
convert(convert("0123456789",list),set); end,[anames()]):

```

```

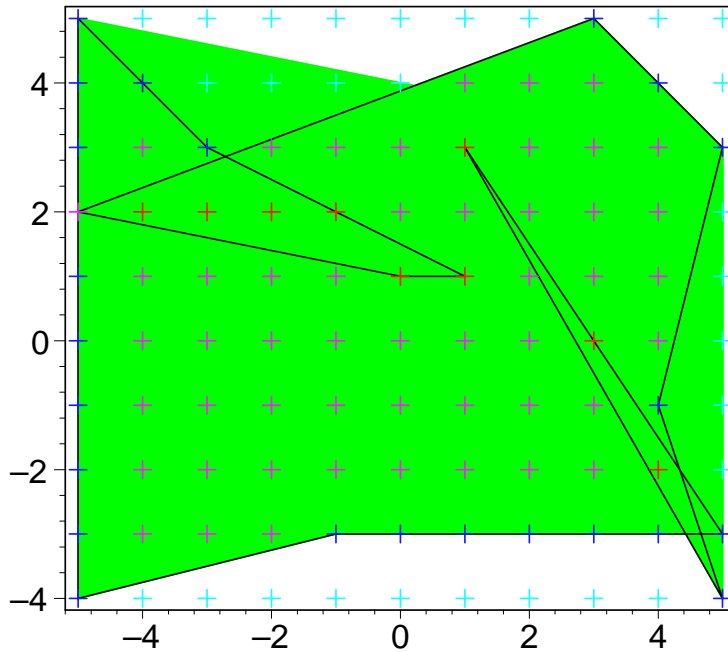
for ex in eval(% ,1) do
  if pick(ex) <> area(ex) then
    print(eval(ex,1)=ex);
    print(scanpositions(ex));
    pick(ex)=area(ex);
    print(%);
    printf("Picks Formel hat einen FEHLER von %a:",
      lhs(%)-rhs(%));
    show(ex);
  end if;
end do:
ex4 = ([5, 3], [3, 5], [-5, 2], [0, 1], [1, 1], [-3, 3], [-5, 5], [-5, -4], [-1, -3], [5, -3], [1, 3],
  [5, -4], [4, -1])

```

$$\text{table}([1 = 50, 2 = 4, \frac{1}{2} = 23, \frac{3}{2} = 5])$$

$$76 = 74$$

Picks Formel hat einen FEHLER von 2:



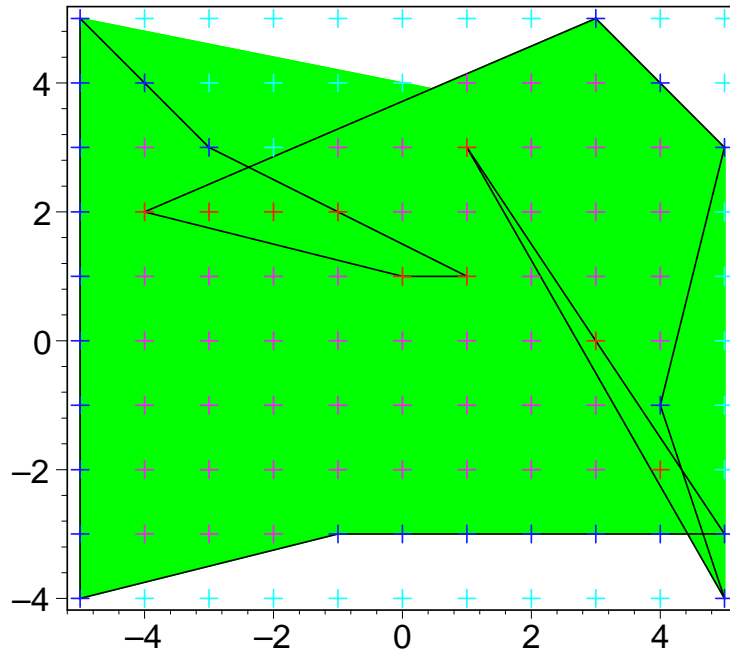
- +++++ outside
- +++++ +inside
- +++++ 2
- +++++ +margin
- +++++ 3/2

ex5 = ([5, 3], [3, 5], [-4, 2], [0, 1], [1, 1], [-3, 3], [-5, 5], [-5, -4], [-1, -3], [5, -3], [1, 3],
 [5, -4], [4, -1])

$$\text{table}([1 = 48, 2 = 3, \frac{1}{2} = 24, \frac{3}{2} = 6])$$

$$74 = 72$$

Picks Formel hat einen FEHLER von 2:



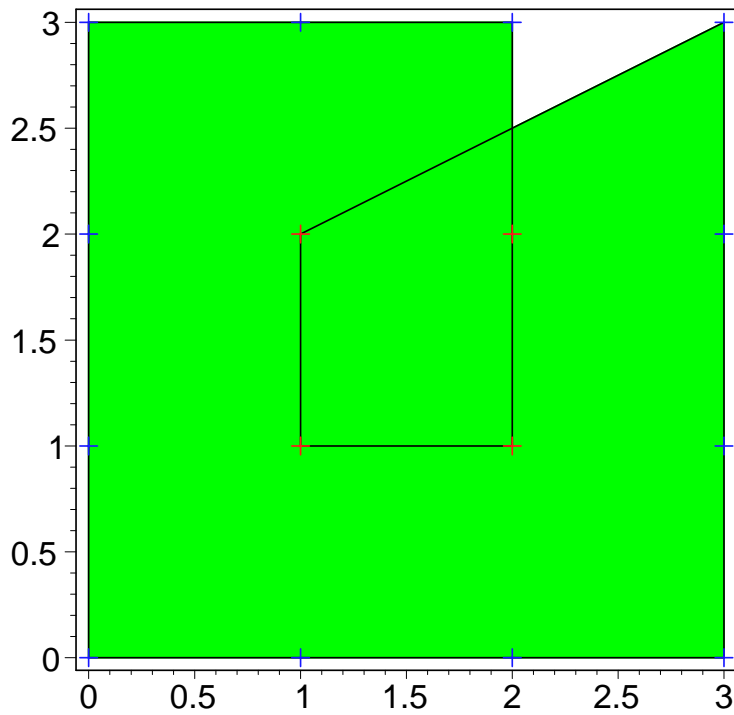
+++++ outside
 +++++ +inside
 +++++ 2
 +++++ +margin
 +++++ 3/2

ex6 = ([0, 0], [3, 0], [3, 3], [1, 2], [1, 1], [2, 1], [2, 3], [0, 3])

$$\text{table}([\frac{1}{2} = 12, \frac{3}{2} = 4])$$

$$11 = 10$$

Picks Formel hat einen FEHLER von 1:



+++++ +margin
 +++++ 3/2

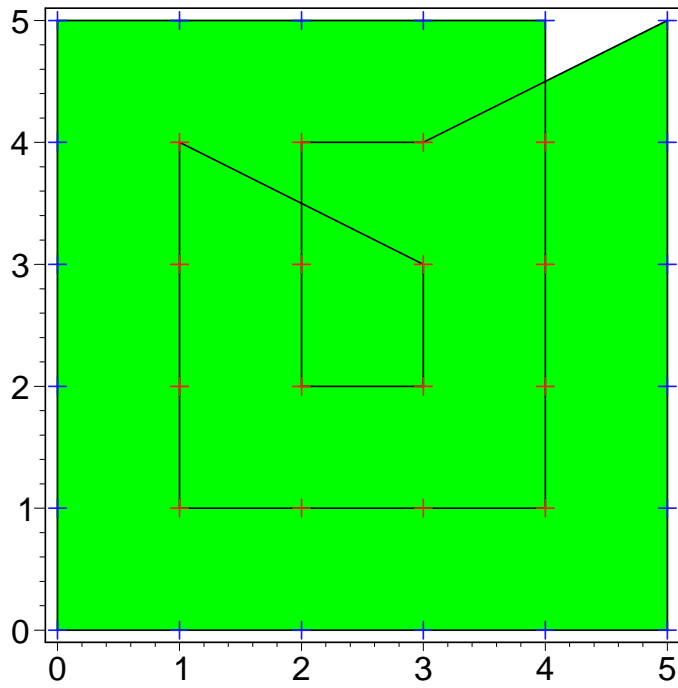
ex7=

([0, 0], [5, 0], [5, 5], [3, 4], [2, 4], [2, 2], [3, 2], [3, 3], [1, 4], [1, 1], [4, 1], [4, 5], [0, 5])

table($\frac{1}{2}=20$, $\frac{3}{2}=12$, $\frac{5}{2}=4$)

37 = 35

Picks Formel hat einen FEHLER von 2:



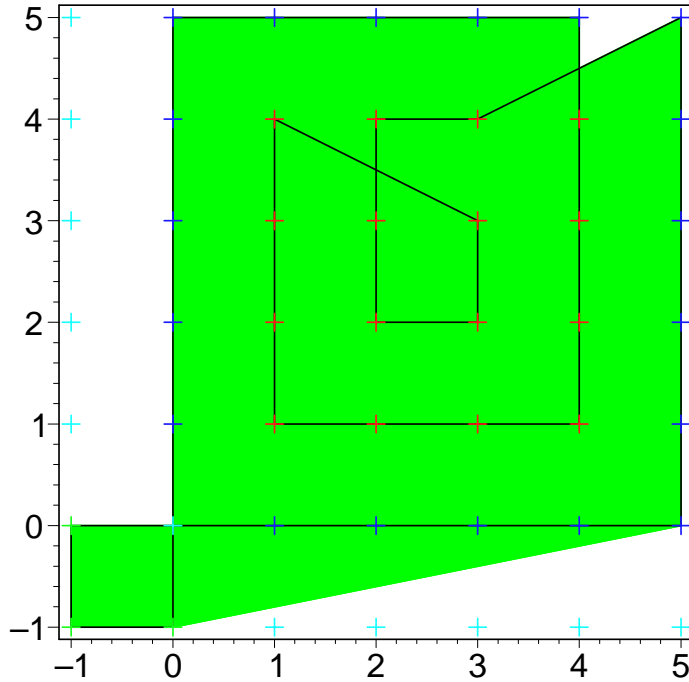
++++++ +margin
 ++++++ 3/2
 ++++++ 5/2

$ex8 = ([0, -1], [-1, -1], [-1, 0], [5, 0], [5, 5], [3, 4], [2, 4], [2, 2], [3, 2], [3, 3], [1, 4], [1, 1],$
 $[4, 1], [4, 5], [0, 5])$

$table(\frac{-1}{2} = 3, \frac{1}{2} = 19, \frac{3}{2} = 12, \frac{5}{2} = 4)$

35 = 34

Picks Formel hat einen FEHLER von 1:



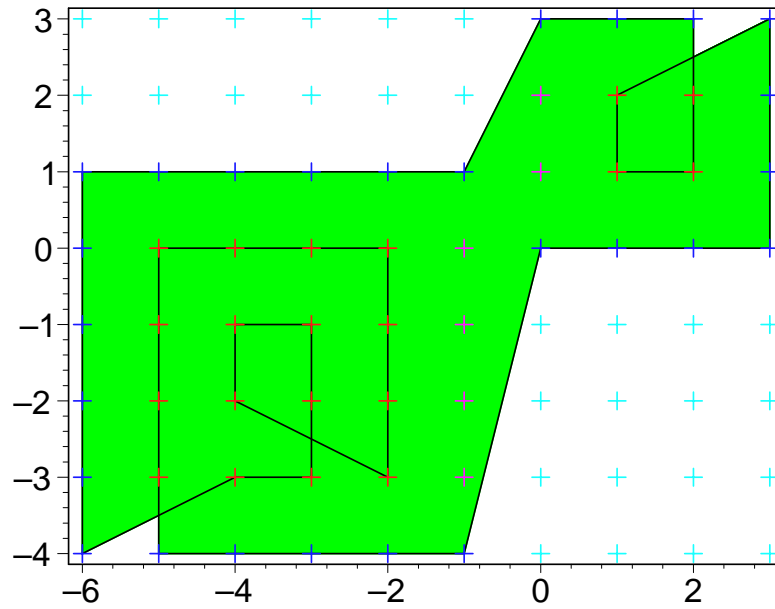
++++++ outside
 ++++++ -margin
 ++++++ +margin
 ++++++ 3/2
 ++++++ 5/2

$ex9 = ([-1, 1], [-6, 1], [-6, -4], [-4, -3], [-3, -3], [-3, -1], [-4, -1], [-4, -2], [-2, -3], [-2, 0], [-5, 0],$
 $[-5, -4], [-1, -4], [0, 0], [3, 0], [3, 3], [1, 2], [1, 1], [2, 1], [2, 3], [0, 3])$

$$\text{table}([1 = 6, \frac{1}{2} = 26, \frac{3}{2} = 16, \frac{5}{2} = 4])$$

$$52 = 49$$

Picks Formel hat einen FEHLER von 3:



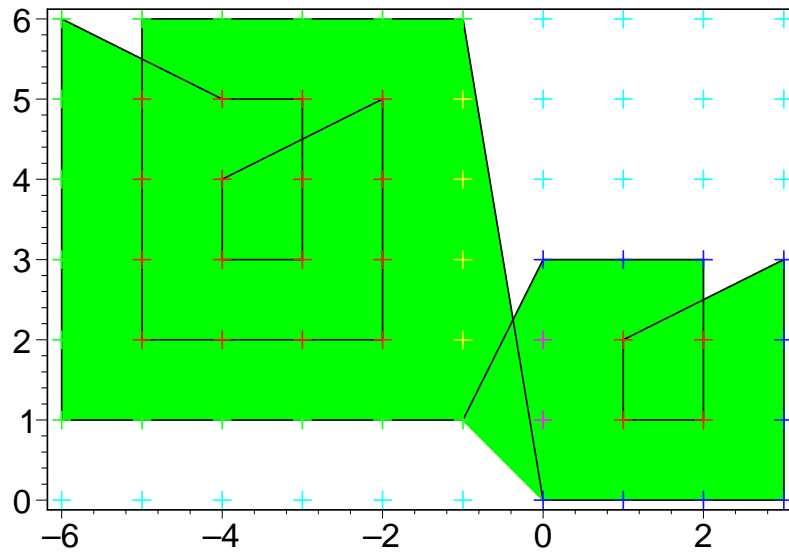
++ + outside
 ++ + inside
 ++ + margin
 ++ + 3/2
 ++ + 5/2

$exIO = ([-1, 1], [-6, 1], [-6, 6], [-4, 5], [-3, 5], [-3, 3], [-4, 3], [-4, 4], [-2, 5], [-2, 2], [-5, 2],$
 $[-5, 6], [-1, 6], [0, 0], [3, 0], [3, 3], [1, 2], [1, 1], [2, 1], [2, 3], [0, 3])$

$table([1 = 2, \frac{-1}{2} = 16, \frac{1}{2} = 10, \frac{-5}{2} = 4, \frac{3}{2} = 4, \frac{-3}{2} = 12, -1 = 4])$

-28 = -26

Picks Formel hat einen FEHLER von -2:



```

+++++ outside
+++++ +inside
+++++ -margin
+++++ +margin
+++++ -5/2
+++++ 3/2

```

Teilabschnitte von Beispiel ex1 durchprobieren,
 NUR die Fälle ausgeben, wo der Satz (bzw. unsere Verfeinerung) nicht stimmt.

```

> for a to 13 do
  for b from a+2 to 13 do
    ex:=ex1[a..b]:
    untrace(position,triangleposition,lineposition):
    infolevel[scan]:=0:
    exT:=scanpositions(ex);
    exT:=op(op(exT));
    exT:=sort( exT, (x,y)->evalb(lhs(x)<lhs(y)) );
    exeq:=pick(ex)=area(ex);
    if false or not exeq then
      printf("Beispiel ex1[%a]\n" ||
        "Umlaufzahlzahlen: %a\n" ||
        "Fläche=PickscheFormel? %a (%a)\n",
        a..b, exT, exeq, evalb(exeq) );
      show(ex);
    fi;
  od;

```

od:

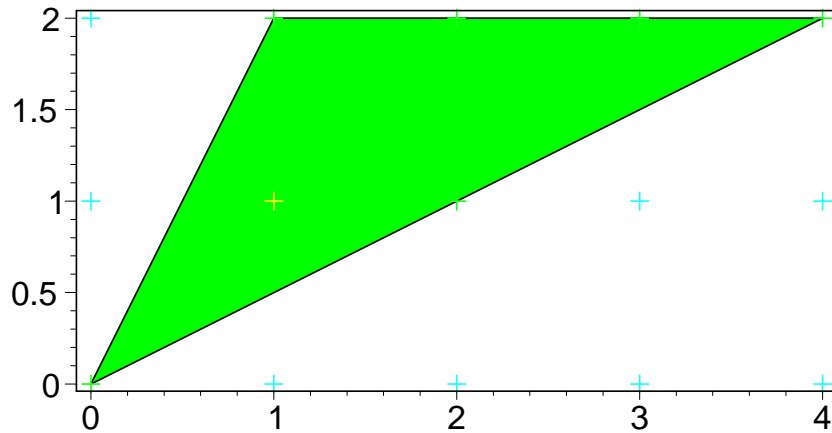
Offenbar funktioniert mit dem dritten Versuch jetzt alles, außer wenn Kanten des Vielecks sich an nicht vorhergesehenen Stellen treffen.

Für `ex[8..12]` und `ex[10..13]` wird dadurch die Position `[-2,3]` falsch bewertet.

Beispiel `ex1[1 .. 3]`

Umlaufzahlanzahlen: `[-1 = 1, -1/2 = 6]`

Fläche=PickischeFormel? `-5 = -3 (false)`

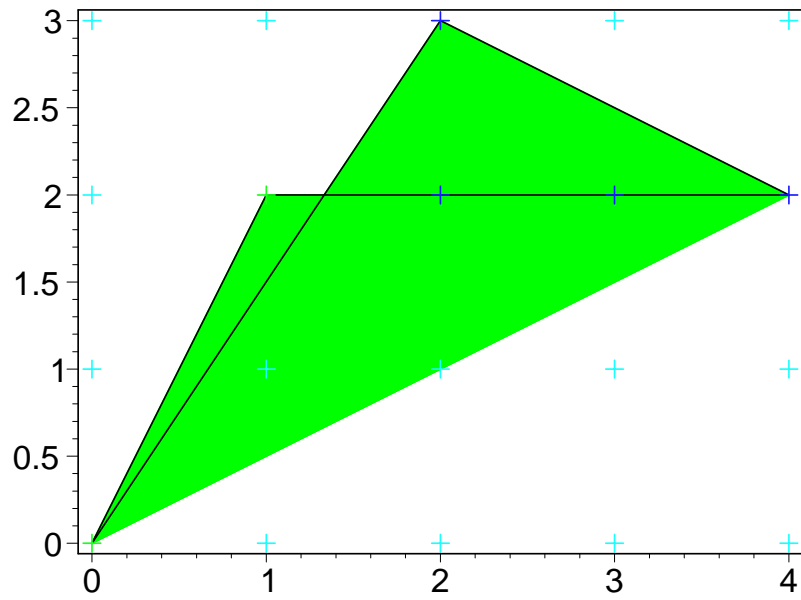


+ + + + + outside
+ + + + + -margin
+ + + + + -inside

Beispiel `ex1[1 .. 4]`

Umlaufzahlanzahlen: `[-1/2 = 2, 1/2 = 4]`

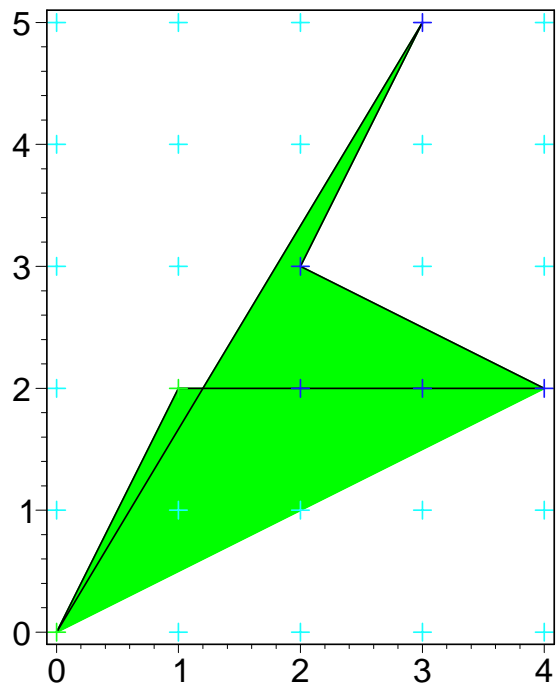
Fläche=PickischeFormel? `0 = 1 (false)`



++ ++ ++ ++ ++ outside
 ++ ++ ++ ++ ++ -margin
 ++ ++ ++ ++ ++ +margin

```

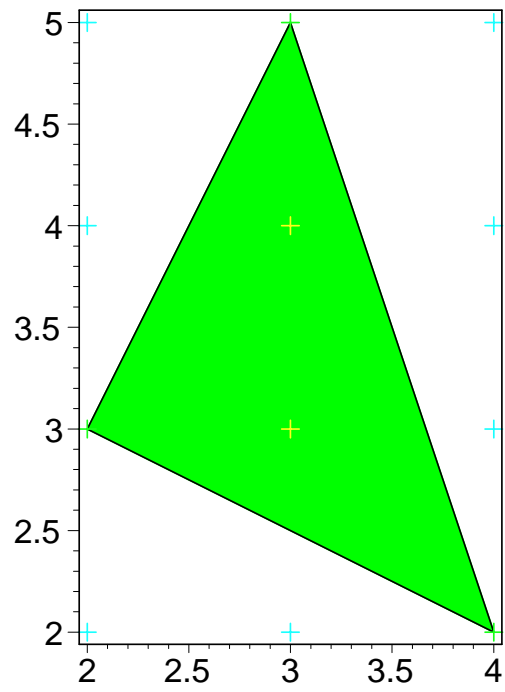
Beispiel ex1[1 .. 5]
Umlaufzahlen: [-1/2 = 2, 1/2 = 5]
Fläche=PickscheFormel? 1/2 = 3/2 (false)
  
```



++++++ outside
 ++++++ -margin
 ++++++ +margin

```

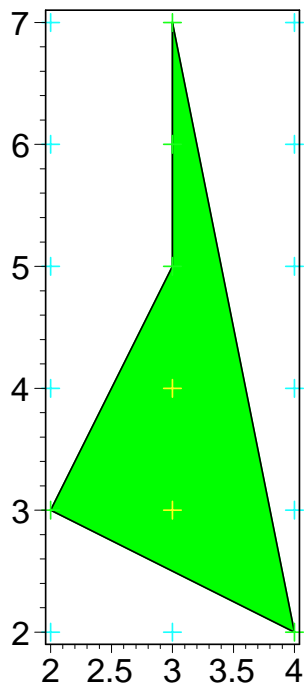
Beispiel ex1[3 .. 5]
Umlaufzahlen: [-1 = 2, -1/2 = 3]
Fläche=PickischeFormel? -9/2 = -5/2 (false)
  
```



+++++ outside
 +++++ -margin
 +++++ -inside

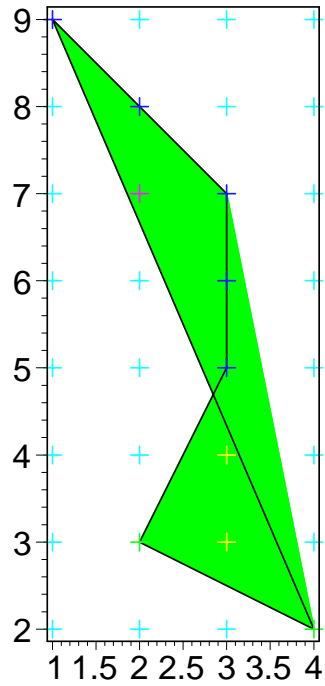
```

Beispiel ex1[3 .. 6]
Umlaufzahlen: [-1 = 2, -1/2 = 5]
Fläche=PickscheFormel? -11/2 = -7/2 (false)
  
```



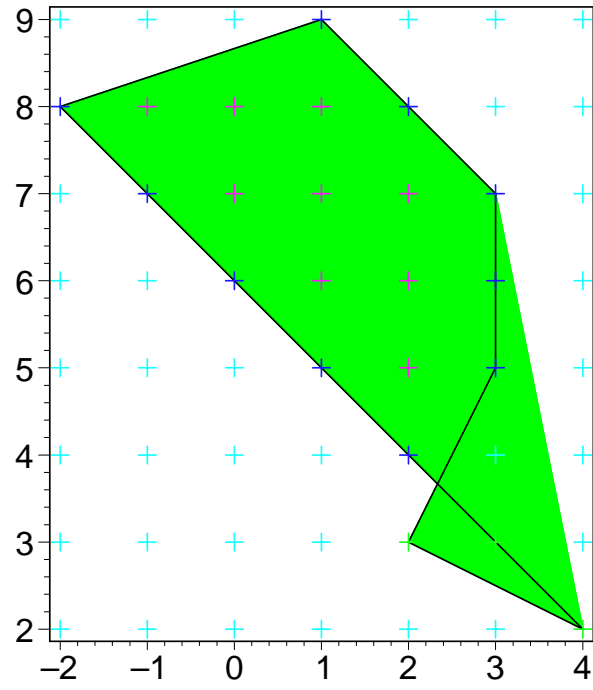
++++ outside
 +++ margin
 +++ inside

Beispiel ex1[3 .. 7]
 Umlaufzahlen: [-1 = 2, -1/2 = 2, 1/2 = 5, 1 = 1]
 Fläche=PickischeFormel? -1/2 = 1/2 (false)



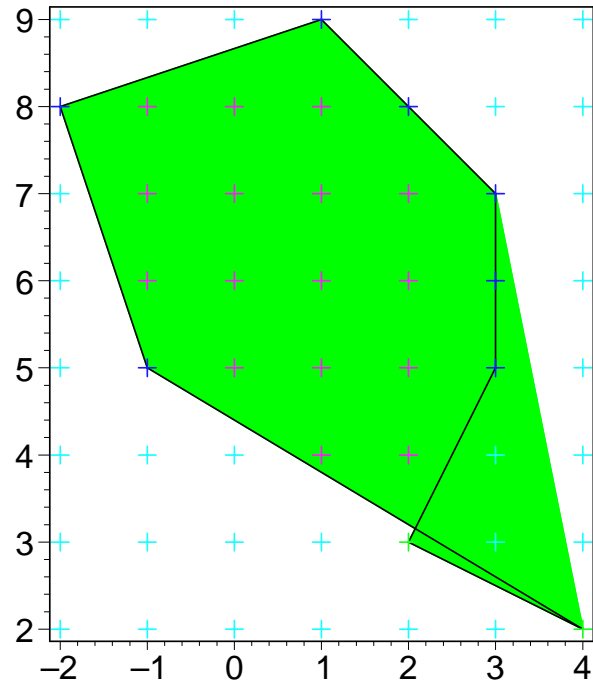
+++++ outside
 +++++ +inside
 +++++ -margin
 +++++ +margin
 +++++ -inside

Beispiel ex1[3 .. 8]
 Umlaufzahlen: [-1/2 = 3, 1/2 = 10, 1 = 9]
 Fläche=PickscheFormel? 23/2 = 25/2 (false)



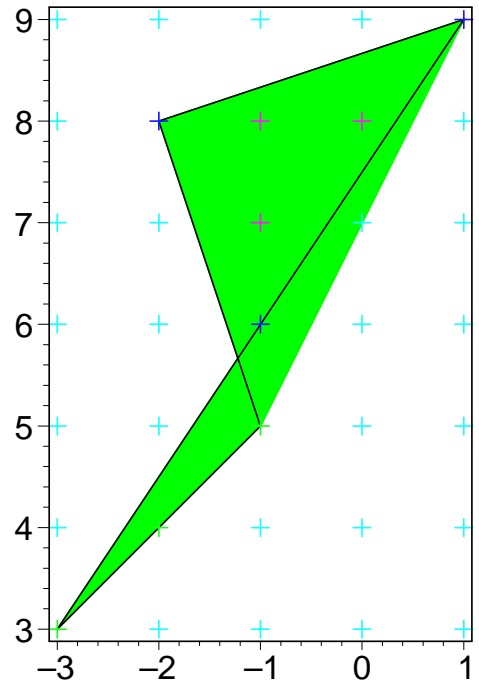
++++++ outside
 ++++++ +inside
 ++++++ -margin
 ++++++ +margin

Beispiel ex1[3 .. 9]
 Umlaufzahlen: [-1/2 = 2, 1/2 = 7, 1 = 16]
 Fläche=PickischeFormel? 35/2 = 37/2 (false)



++++++ outside
 ++++++ +inside
 ++++++ -margin
 ++++++ +margin

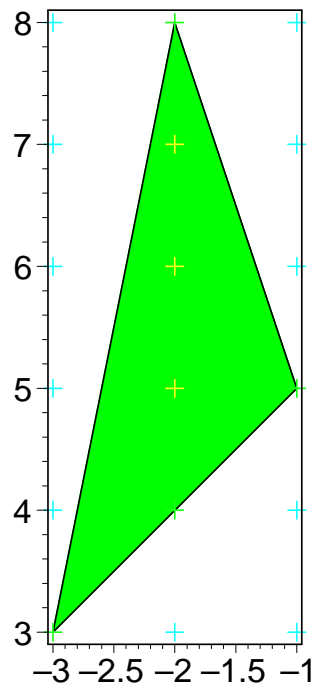
Beispiel ex1[7 .. 10]
 Umlaufzahlen: [-1/2 = 3, 1/2 = 3, 1 = 3]
 Fläche=PickscheFormel? 2 = 3 (false)



++++++ outside
 ++++++ +inside
 ++++++ -margin
 ++++++ +margin

```

Beispiel ex1[8 .. 10]
Umlaufzahlen: [-1 = 3, -1/2 = 4]
Fläche=PickscheFormel? -6 = -4 (false)
  
```

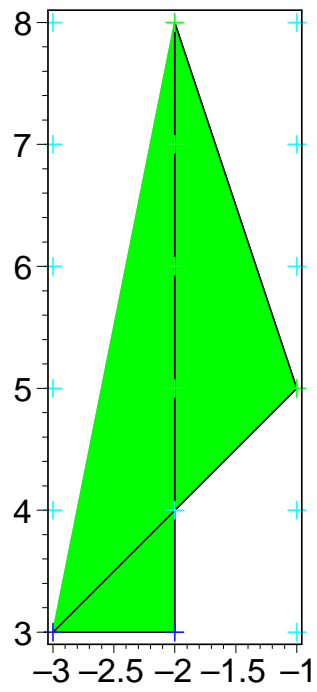


++++ outside
 ++++ -margin
 ++++ -inside

```

Beispiel ex1[8 .. 11]
Umlaufzahlen: [-1/2 = 5, 1/2 = 2]
Fläche=PickscheFormel? -5/2 = -3/2 (false)

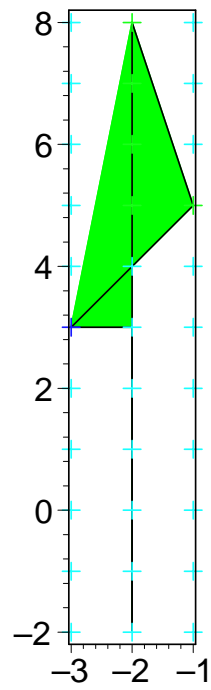
```



+++++ outside
 +++++ -margin
 +++++ +margin

```

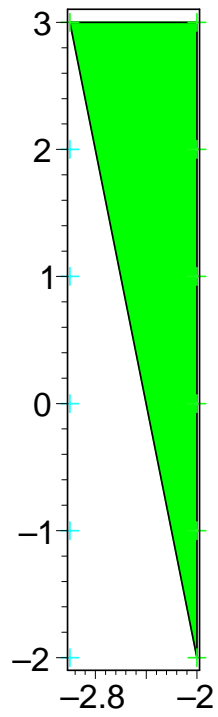
Beispiel ex1[8 .. 12]
Umlaufzahlen: [-1/2 = 5, 1/2 = 1]
Fläche=PickscheFormel? -3 = -3/2 (false)
  
```



+margin
 -margin
 outside

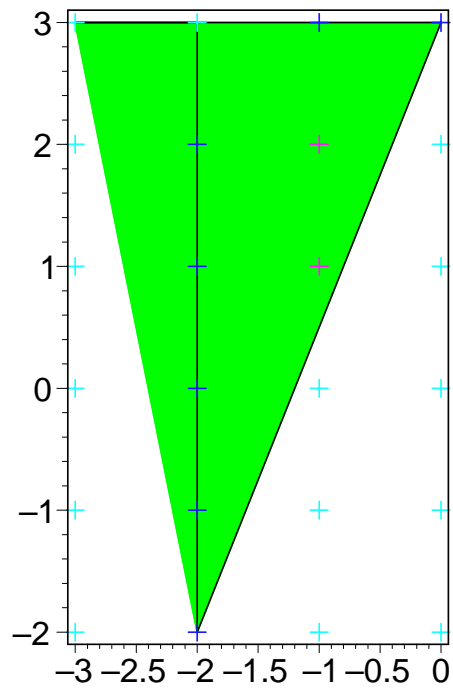
```

Beispiel ex1[10 .. 12]
Umlaufzahlen: [-1/2 = 7]
Fläche=PickischeFormel? -9/2 = -5/2 (false)
  
```



outside
-margin

```
Beispiel ex1[10 .. 13]  
Umlaufzahlen: [1/2 = 7, 1 = 2]  
Fläche=PickischeFormel? 9/2 = 5 (false)
```



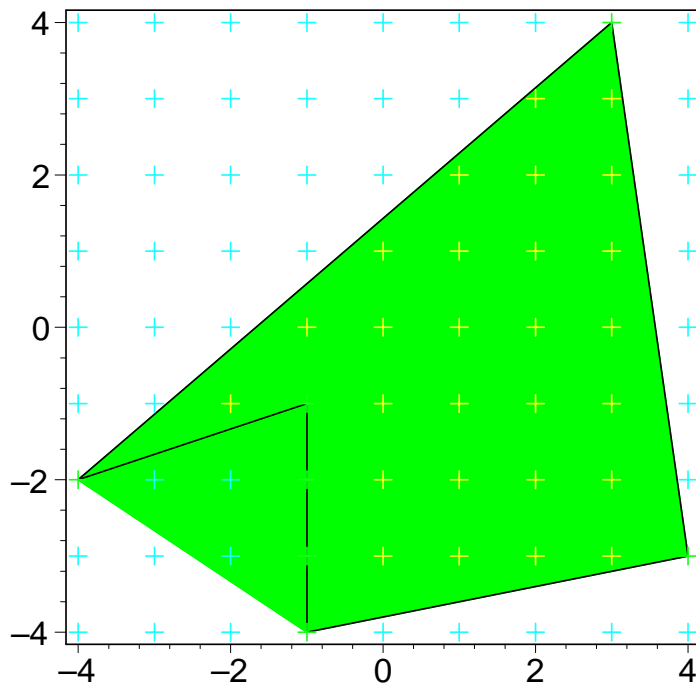
+++++ outside
 +++++ +inside
 +++++ +margin

- Zufällige Beispiele erzeugen und testen

```
> ex:=seq( [rand(-5..5)(),rand(-5..5)()], i=1..rand(3..15)() );
> pick(ex)=area(ex);
show(ex);
```

```
ex := [-1, -4], [-1, -1], [-4, -2], [3, 4], [4, -3]
```

$$\frac{-63}{2} = \frac{-59}{2}$$



+++++++ outside
 +++++++ -margin
 +++++++ -inside

– Zufällige fast sternförmige Beispiele erzeugen und testen

Hier werden zufällige Punkte gewählt und dann nach dem Winkel (arg) sortiert.

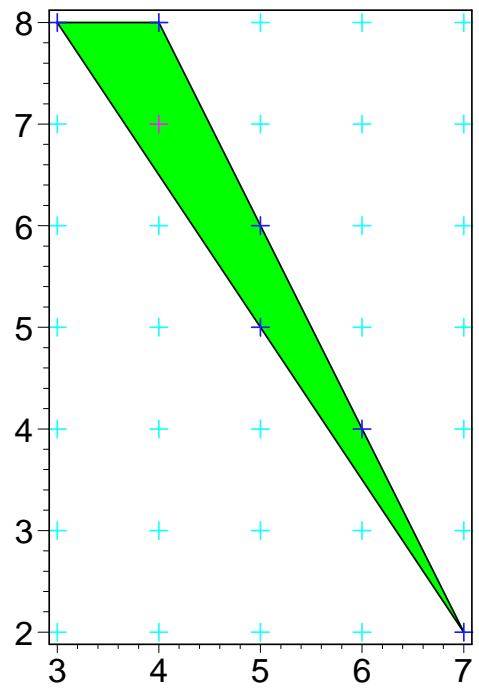
```
> arg:=proc(x)
  option remember;
  evalf(argument(x[1]+I*x[2]));
end proc;
> exu:=seq( [rand(-2..8)(),rand(-2..8)()], i=1..rand(3..15)()
);
pick(exu)=area(exu);
ex:=op(sort([exu], (x,y)->evalb(arg(x)<arg(y)) ));
pick(ex)=area(ex);
show(ex);
```

exu := [7, 2], [3, 8], [4, 8]

-5 = -3

ex := [7, 2], [4, 8], [3, 8]

3 = 3



+++++ outside
 ++++ inside
 ++++ margin

[>
 [>
 [>
 [>