joachim heintz
seminar I (daria, luis, philipp, shadi)
dezember 2020

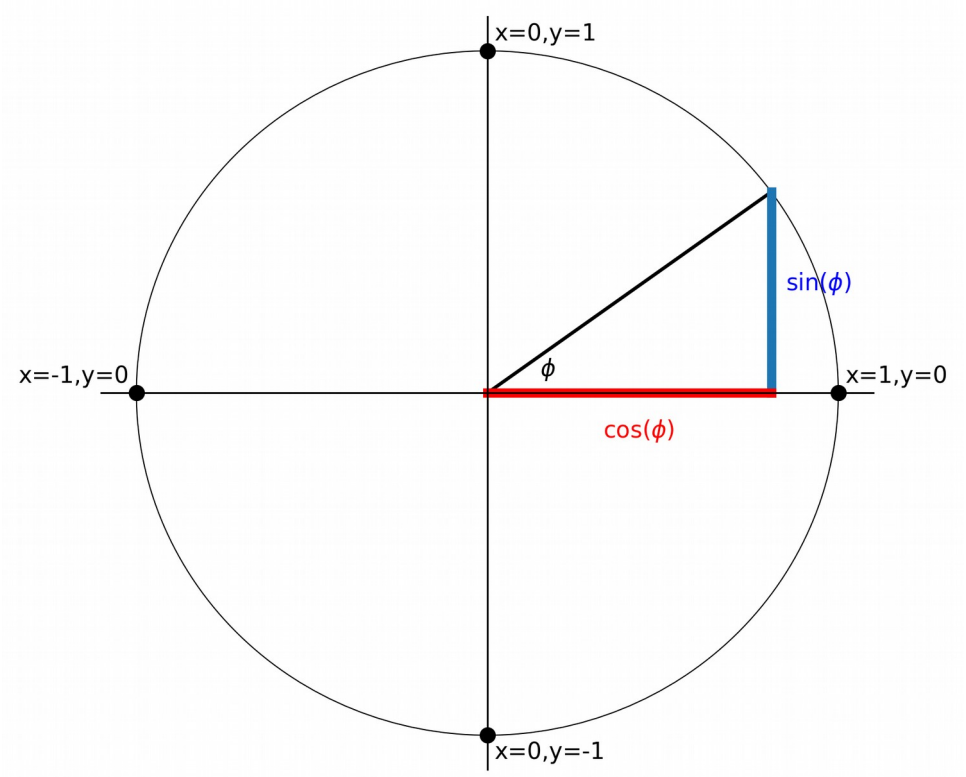
Entwicklung von Code an einem Beispiel: XY-Punkte zu Winkeln für VBAP umformen

1 Aufgabe / Situation

Wir wollen xy-Werte, die wir von einer externen Software bekommen (z.B. über OSC) zur Steuerung eines Klanges durch VBAP [Vector Base Amplitude Panning] benutzen. Deshalb müssen wir die xy-Werte in einen Winkel umformen.

2 Vereinfachung der Aufgabe um anzufangen

Wir wollen den Winkel φ für einen Punkt auf dem Einheitskreis durch seine xy-Koordinaten bestimmen.



Lösungsmöglichkeit mit dem (Arcus-) Sinus

3 Grundansatz

Nehmen wir als Beispiel das eingezeichnete rechtwinklige Dreieck. Die blaue Strecke sin(φ) entspricht dem y-Wert: sin(φ) = y. Wenn wir also den y-Wert schon haben, bekommen wir den Winkel φ als Umkehrung der Sinusfunktion: φ = arcsin(y).

Beispiel 1: Wir wollen den Winkel φ für den y-Wert 1 wissen und bekommen
arcsin(1) -> 1.5707963267948966 (das ist π/2, also 90°)

Beispiel 2: Wir wollen den Winkel φ für den y-Wert -1 wissen und bekommen
arcsin(-1) -> -1.5707963267948966 (also -90°, was dasselbe ist wie 270°)

4 Problem

Diese Ermittlung des Winkels als Umkehrung der Sinusfunktion funktioniert nur in der rechten Hälfte des Kreises. Es gibt aber in der linken Hälfte des Kreises zu jedem y-Wert auch einen Punkt. Für den y-Wert 0 beispielsweise können wir nicht wissen, ob der x-Wert 1 oder der x-Wert -1 gemeint ist. (Für x=1 wäre der Winkel 0°; für x=-1 wäre der Winkel 180° oder π.)

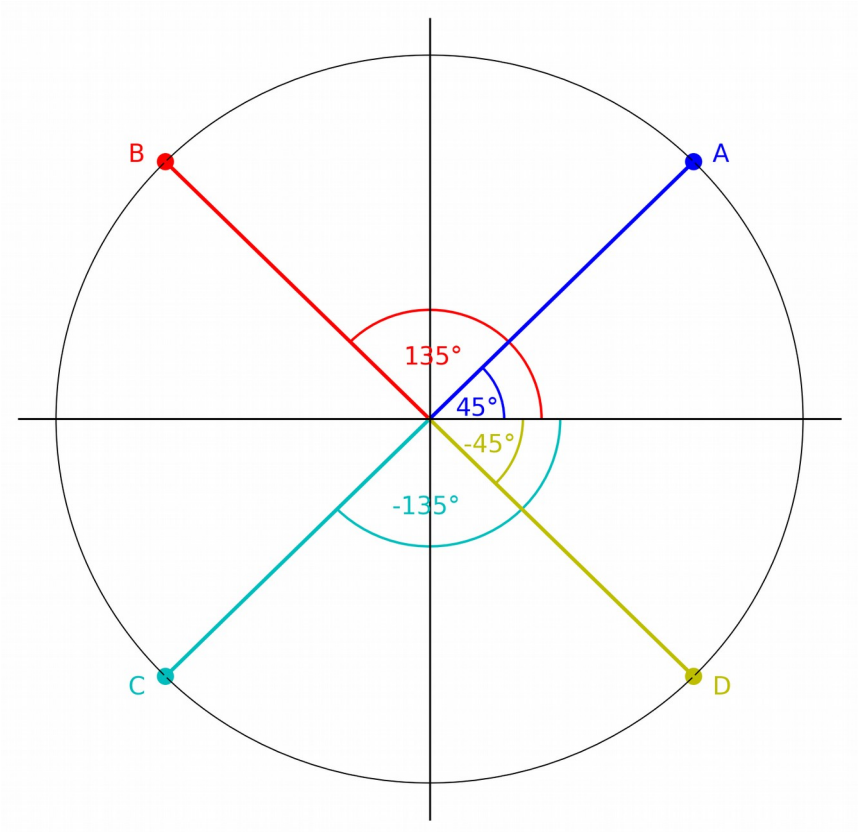
5 Idee zur Problemlösung

Wir unterscheiden an Hand des x-Wertes zwei Fälle:

- In der rechten Hälfte können wir einfach arcsin(y) nehmen.
- In der linken Hälfte müssen wir einen bestimmten Betrag zu dem Winkel addieren bzw. von ihm subtrahieren.

Wie genau dieser Betrag sein muss, den wir addieren bzw. subtrahieren, sehen wir an folgender Darstellung. Dargestellt sind diese Punkte:

Punkt	x	y	Winkel als Grad (0 - 360°)	Winkel als Radiant (Vielfaches von π)
A	0.707	0.707	45	(1/4) π
B	-0.707	0.707	135	(3/4) π
C	-0.707	-0.707	-135	-(3/4) π
D	0.707	-0.707	-45	-(1/4) π



Wenn wir schauen, welche Abweichungen von den richtigen Winkeln sich ergeben, wenn wir für diese Punkte den Arkussinus bestimmen, sehen wir folgendes:

Punkt	x	y	arcsin(y)	eigentlicher Winkel	Abweichung
A	0.707	0.707	(1/4) π	(1/4) π	-
B	-0.707	0.707	(1/4) π	(3/4) π	π / 2
C	-0.707	-0.707	-(1/4) π	-(3/4) π	- (π /2)
D	0.707	-0.707	-(1/4) π	-(1/4) π	-

6 Lösung in normaler Sprache

Wir haben einen Punkt auf dem Einheitskreis mit den Koordinaten x und y.
Wenn x positiv ist (oder Null), ist der Winkel dieses Punktes zur x-Achse arcsin(y).
Wenn x negativ ist und y positiv (oder Null), ist der Winkel arcsin(y) plus π/2.
Wenn x negativ ist und y negativ, ist der Winkel arcsin(y) minus π/2.

7 Lösung in Pseudo-Code

a) Einfach die sprachliche Lösung formalisiert:

```
input = x, y
if x >= 0 then angle = arcsin(y)
if x < 0 and y >= 0 then angle = arcsin(y) + π/2
if x < 0 and y < 0 then angle = arcsin(y) - π/2
output = angle
```

b) Etwas klarere Logik und deutlich formatiert:

```
input = x, y

if x >= 0 then
  angle = arcsin(y)
else
  if y >= 0 then
    angle = arcsin(y) + π/2
  else
    angle = arcsin(y) - π/2

output = angle
```

8 Lösung in Python

```
def xy2angle(x,y):
    from math import asin, pi
    if x >= 0:
        angle = asin(y)
    else:
        if y >= 0:
            angle = asin(y) + pi/2
        else:
            angle = asin(y) - pi/2
    return angle
```

```
xy2angle(.707,.707)        ->   0.785...
xy2angle(-.707,.707)       ->   2.356...
xy2angle(-.707,-.707)      ->   -2.356...
xy2angle(.707,-.707))      ->   -0.785...
```

9 Lösung in JavaScript

```
function xy2angle(x,y) {
  var angle;
  if ( x >= 0 )
    angle = Math.asin(y);
  else
    if ( y >= 0 )
      angle = Math.asin(y) + Math.PI/2;
    else
      angle = Math.asin(y) - Math.PI/2;
  return angle;
}

console.log(xy2angle(.707,.707)) ;
console.log(xy2angle(-.707,.707)) ;
console.log(xy2angle(-.707,-.707)) ;
console.log(xy2angle(.707,-.707)) ;
```

Das in einem Skript geschrieben ergibt ausgeführt:

```
0.785...
2.356...
-2.356...
-0.785...
```

10 Lösung in Lisp

```
(defun xy2angle (x y)
 (if (>= x 0)
  (asin y)
  (if (>= y 0)
   (+ (asin y) (/ pi 2))
   (- (asin y) (/ pi 2))
  )
 )
)

(xy2angle .707 .707)       ->   0.785...
(xy2angle -.707 .707)      ->   2.356...
(xy2angle -.707 -.707)     ->   -2.356...
(xy2angle .707 -.707)      ->   -0.785...
```

11 Lösung in Csound

a) In einem Instrument

```
instr xy2angle
 ix = p4
 iy = p5
 if ix >= 0 then
  iangle = sininv(iy)
 else
  if iy >= 0 then
   iangle = sininv(iy) + $M_PI_2
  else
   iangle = sininv(iy) - $M_PI_2
  endif
 endif
 print iangle
endin
```

Aufruf im Score:

```
i 1 0 0  .707  .707
i 1 0 0 -.707  .707
i 1 0 0 -.707 -.707
i 1 0 0  .707 -.707
```

Ausdruck in der Konsole:

```
instr 1:  iangle = 0.785
instr 1:  iangle = 2.356
instr 1:  iangle = -2.356
instr 1:  iangle = -0.785
```

b) Mit einer eigenen Funktion ("User Defined Opcode")

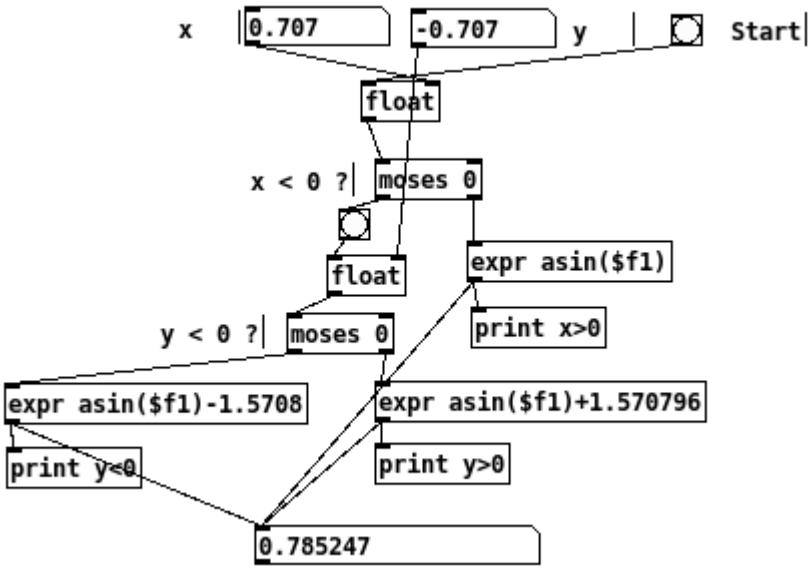
```
opcode xy2angle,i,ii
 ix, iy xin
 if ix >= 0 then
  iangle = sininv(iy)
 else
  if iy >= 0 then
   iangle = sininv(iy) + $M_PI_2
  else
   iangle = sininv(iy) - $M_PI_2
  endif
 endif
 xout iangle
endop
```

```
instr 1
 print xy2angle(.707,.707)
 print xy2angle(-.707,.707)
 print xy2angle(-.707,-.707)
 print xy2angle(.707,-.707)
endin
schedule(1,0,0)
```

Ausdruck in der Konsole:

```
instr 1:  #i0 = 0.785
instr 1:  #i1 = 2.356
instr 1:  #i2 = -2.356
instr 1:  #i3 = -0.785
```

12 Lösung in PD (Pure Data)



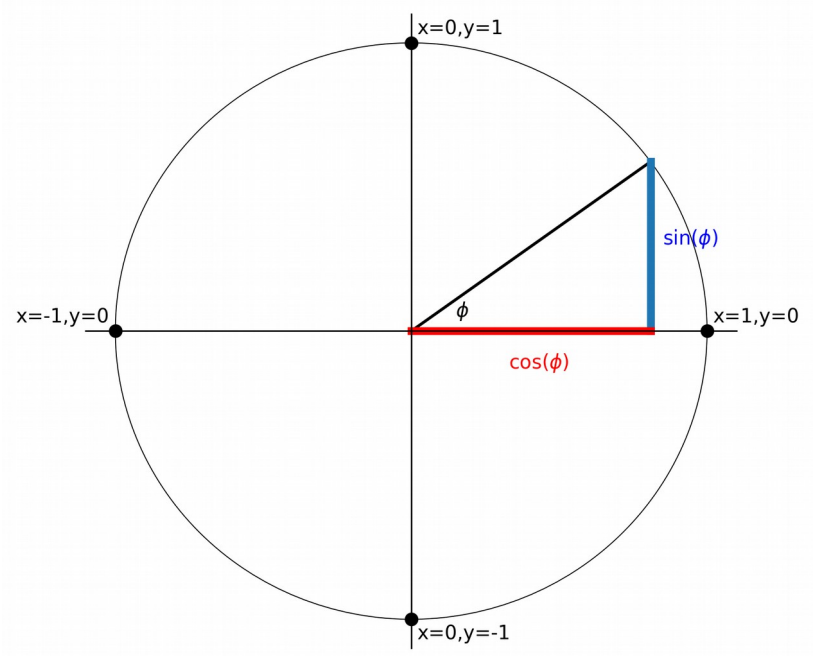
Nach viermaliger Eingabe der Werte für die Punkte A, B, C, D bei x und y gefolgt jeweils vom Klicken auf den Start Button zeigt die PD Konsole:

```
x>0: 0.785247
y>0: 2.35604
y<0: -2.35605
x>0: 0.785247
```

Lösungsmöglichkeit mit dem (Arcus-) Cosinus

13 Am Einheitskreis

Gehen wir noch einmal zurück auf den grundlegenden Ansatz, können wir statt des Sinus auch den Kosinus benutzen. In der Darstellung, die wir oben schon gesehen haben, ist der x-Wert (rot) der Kosinus des Winkels φ:



Da wir den Winkel aus den xy-Werten bestimmen wollen, müssen wir wieder die Umkehrfunktion bemühen. Wenn x = cos(φ), dann ist φ = arccos(x). Die Fallunterscheidungen sind etwas einfacher; statt drei Fälle zu unterscheiden kommt man mit der Unterscheidung zwischen positiven und negativen y-Werten aus, wie wir an dieser Tabelle sehen:

Punkt	x	y	arccos(x)	eigentlicher Winkel	Abweichung
A	0.707	0.707	(1/4) π	(1/4) π	-
B	-0.707	0.707	(3/4) π	(3/4) π	-
C	-0.707	-0.707	(3/4) π	-(3/4) π	* (-1)
D	0.707	-0.707	(1/4) π	-(1/4) π	* (-1)

Mit anderen Worten: Wir bekommen den richtigen Winkel, wenn wir für positive y-Werte den Arkuskosinus von x nehmen, und für negative y-Werte den Arkuskosinus von x noch mit -1 multiplizieren bzw. ihm ein negatives Vorzeichen geben.

14 Lösung in Pseudo-Code

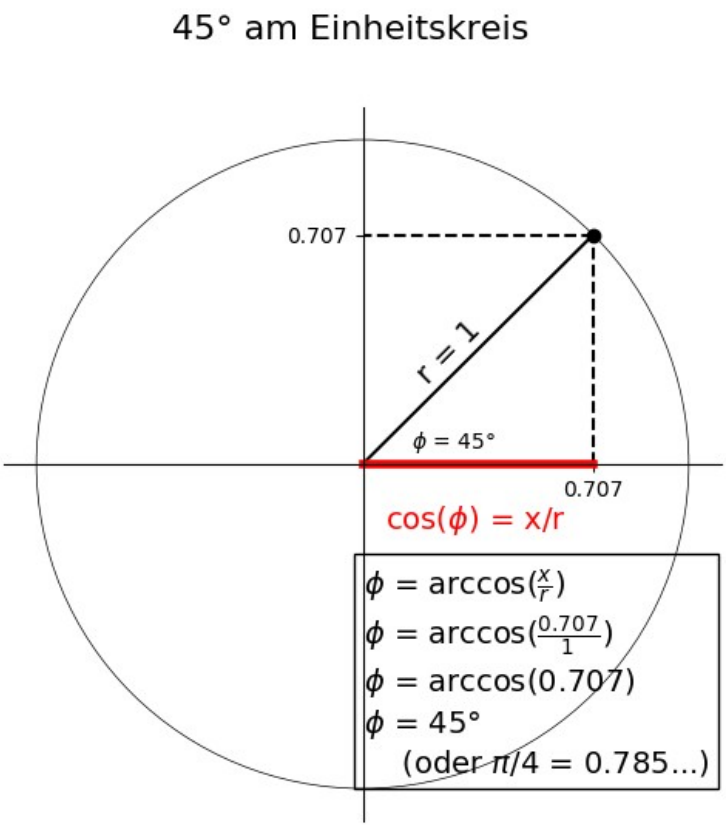
```
input = x, y
if y >= 0 then
  angle = arccos(x)
else
  angle = -arccos(x)
output = angle
```

Das lässt sich noch einfacher als die obigen Beispiele in die jeweiligen Programmiersprachen umsetzen. (Noch einfacher ist es übrigens mit der atan2 / arctan2 / taninv2 Funktion. Aber dabei lernt man weniger das, worum es hier geht: Klar denken und daraus Code entwickeln.)

Verallgemeinerung für alle xy-Werte

15 Einheitskreis noch einmal genauer

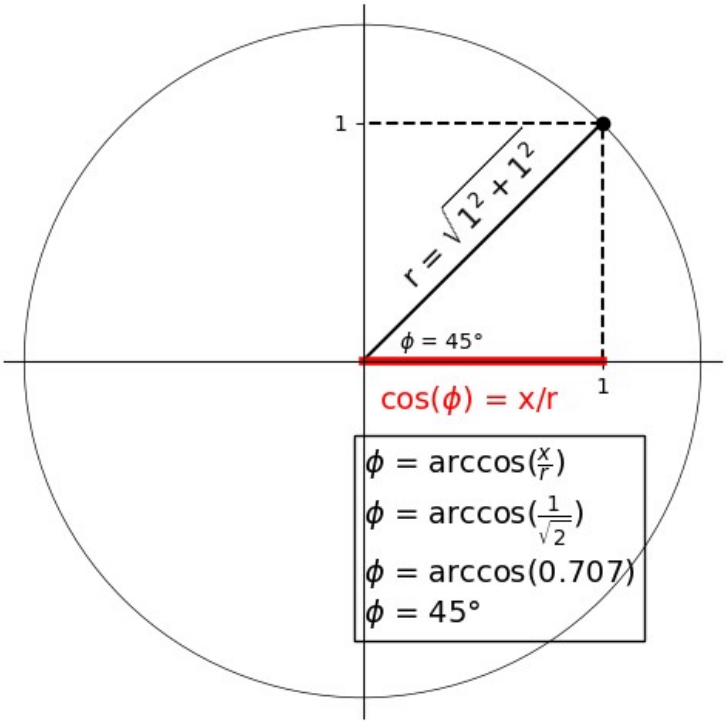
Wir können den gefundenen Ansatz jetzt relativ leicht auf Punkte verallgemeinern, die außerhalb des Einheitskreises liegen. Der Kosinus als Verhältnis der Ankathete zur Hypothenuse im rechtwinkligen Dreieck ist ja eigentlich das Verhältnis des x-Wertes zum Radius r. Nur weil im Einheitskreis dieser Radius 1 ist, vereinfacht sich das Verhältnis auf cos(x) statt cos(x/r). Hier noch einmal die genauere Darstellung am Beispiel des 45° Winkels:



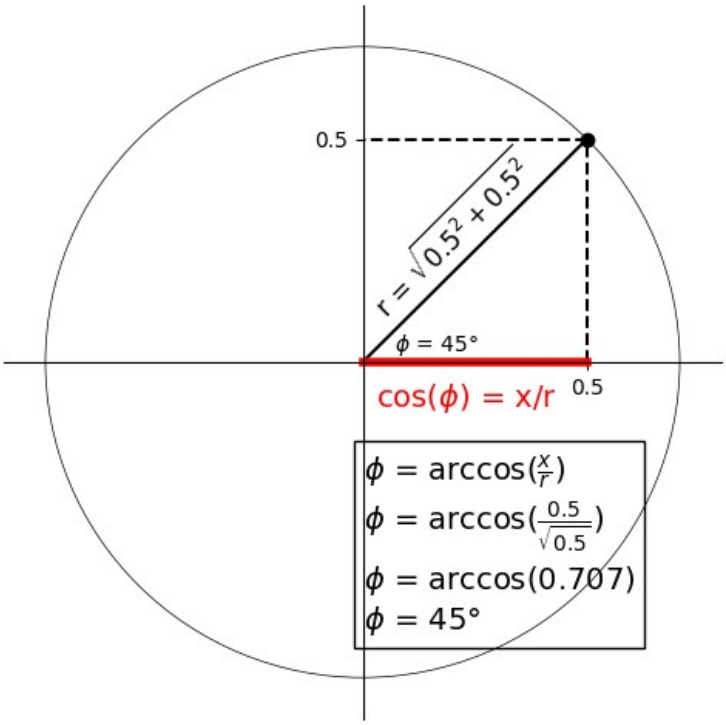
16 Lösung für beliebige x- und y-Werte

Von dort aus kommen wir leicht zur Lösung für beliebige xy-Werte. Wir müssen nur den Radius bestimmen, was durch den Satz des Pythagoras leicht gelingt, der in unserem Fall lautet: $x^2 + y^2 = r^2$. Dadurch berechnet sich r als Wurzel von x^2+y^2. Es folgen zwei Beispiele für die Punkte (1,1) und (0.5,0.5), die ebenfalls einen 45° Winkel bilden.

45° bei x=1 und y=1



45° bei x=0.5 und y=0.5



17 Pseudo-Code für beliebige xy-Werte

Wir müssen noch einen Fall bedenken: x und y dürfen nicht beide gleichzeitig null sein. Wie unser Programm für diesen Sonderfall reagiert, hängt vom Zusammenhang ab. Für unsere Anwendung in VBAP reicht es aus , wenn wir für diesen Fall einen sehr kleinen x- und y-Wert als Substitut setzen.

input = x, y

if x = 0 and y = 0 then
  x = 0.00001
  y = 0.00001

r = sqrt(x^2+y^2)

if y >= 0 then
  angle = arccos(x/r)
else
  angle = -arccos(x/r)

output = angle

18 Csound-Code für beliebige xy-Werte

Das in einen Csound User-Defined Opcode umgesetzt sieht so aus

```
opcode xy2angle,i,ii
 ix, iy xin
 if ix==0 && iy==0 then
  ix = 0.00001
  iy = 0.00001
 endif
 ir = sqrt(ix^2+iy^2)
 if iy >= 0 then
  iangle = cosinv(ix/ir)
 else
  iangle = -cosinv(ix/ir)
 endif
 xout iangle
endop

instr 1
 print xy2angle(.707,.707)
 print xy2angle(1,1)
 print xy2angle(0.5,0.5)
 print xy2angle(10,10)
endin
schedule(1,0,0)
```

Ausdruck in der Konsole:

```
instr 1:  #i0 = 0.785
instr 1:  #i1 = 0.785
instr 1:  #i2 = 0.785
instr 1:  #i3 = 0.785
```

Letzte Kleinigkeiten

19 Umformung Radians zu Grad

VBAP funktioniert in Grad (degree), nicht in Bogenmaß (radians). Wir müssen also die Strecke 2π in 360 Teile aufteilen (bzw. π in 180 Teile). Am besten wir schreiben eine Unterfunktion, die radians nimmt und degree zurückgibt. Dies wäre je eine Version in Python und in Csound:

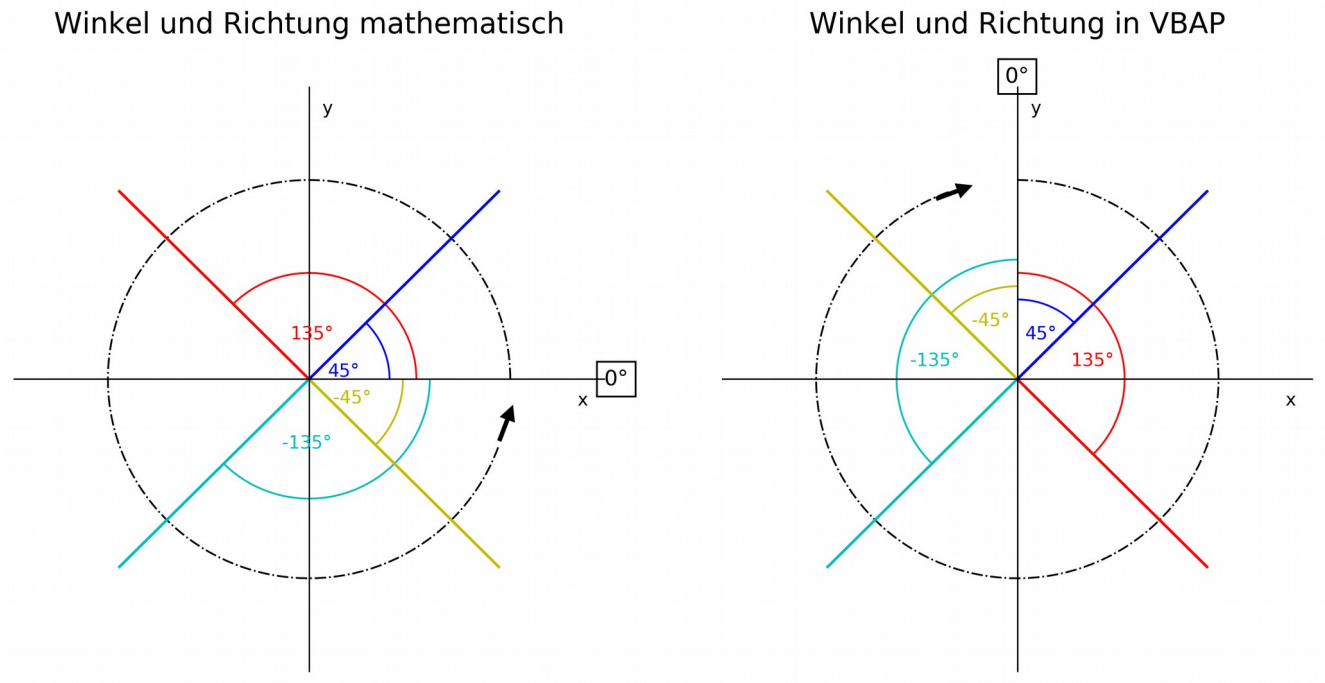
```
def rad2degree(rad):
    from math import pi
    return rad*180 / pi

opcode rad2degree, i, i
 iRadian xin
 xout iRadian*180 / $M_PI
endop
```

20 Anpassung der 0° Position und der Richtung

VBAP hat noch zwei Setzungen, die anders sind als das normale mathematische Verfahren, das sich in den Standardfunktionen der Programmiersprachen widerspiegelt:

a) Die Winkel werden im Uhrzeigersinn abgetragen, nicht gegen den Uhrzeigersinn.
b) Die 0° Position ist "vorn", also in Richtung der y-Achse (statt in Richtung der x-Achse).



Wenn wir also von einem xy-Punkt den Winkel in mathematischer Weise bestimmen, müssen wir
a) den Winkel mit -1 multiplizieren, um die Richtung zu ändern (aus 45° wird -45°), und dann
b) 90° zu dem Ergebnis addieren, damit die 0° Position an der y-Achse ausgerichtet wird.

21 Zwei Lösungen für alles zusammen

Um die verschiedenen Programmiersprache vergleichen zu können, zum Abschluss nochmal eine Lösung in Python und eine in Csound. Bei beiden komprimiere ich den Code noch etwas, wodurch er zwar etwas schlechter lesbar ist, aber weniger Platz wegnimmt.

Python

```
def xy2vbap(x,y):
    from math import pi, acos, sqrt
    r = sqrt(x**2 + y**2)
    if y >= 0: rad_math = acos(x/r)
    else: rad_math = -acos(x/r)
    deg_math = rad_math * 180 / pi
    deg_vbap = -deg_math + 90
    return deg_vbap

#test mit fünf punkten
for x,y in zip([1,0,-1,0,-1],[0,1,0,-1,1]):
    print(xy2vbap(x,y))
```

```
->    90.0
      0.0
      -90.0
      180.0
      -45.0
```

Csound

```
opcode xy2vbap,k,kk
 kx, ky xin
 kradius = sqrt(kx^2+ky^2)
 k_rad_math = (ky>=0) ? cosinv(kx/kradius) : -cosinv(kx/kradius)
 xout -(k_rad_math * 180 / $M_PI -90)
endop

//eine bewegung durch das koordinatensystem
instr 1
 kx = linseg:k(0,2,1,4,-1,2,0)
 ky = linseg:k(1,4,-1,4,1)
 kwinkel = xy2vbap(kx,ky)
 printks("x = %4.1f, y = %4.1f, Winkel = %3d°\n",1,kx,ky,kwinkel)
endin
schedule(1,0,8)
```

```
->    x =  0.0, y =  1.0, Winkel =   0°
      x =  0.5, y =  0.5, Winkel =  45°
      x =  1.0, y = -0.0, Winkel =  90°
      x =  0.5, y = -0.5, Winkel = 135°
      x = -0.0, y = -1.0, Winkel = 180°
      x = -0.5, y = -0.5, Winkel = 225°
      x = -1.0, y =  0.0, Winkel = -90°
      x = -0.5, y =  0.5, Winkel = -45°
```