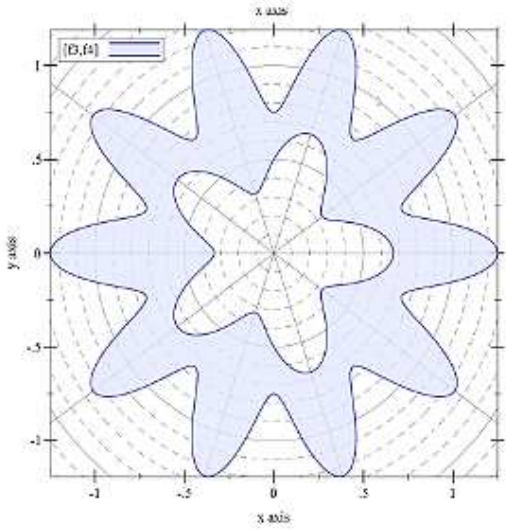
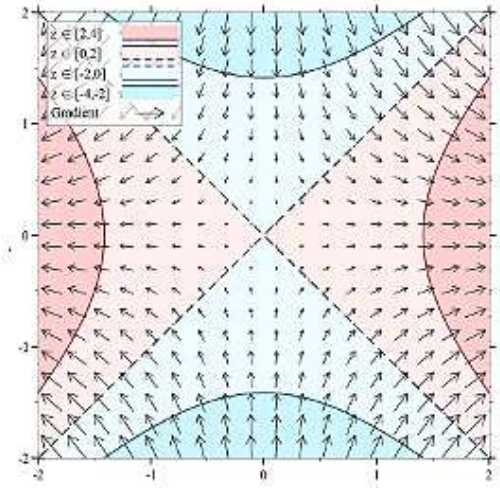
Racket Plot

Inhaltsverzeichnis

1 Kurze Beschreibung

Racket Plot
Programmiersprache: Scheme
Aktionen: plottet Funktionen





2 Code

```
#lang racket
(require plot)

(plot
(function sin -5 5))
(plot
(function sin) #:x-min -5 #:x-max 5 #:y-min -5 #:y-max 5)
;(linear-seq 0 1 5)
;(linear-seq 0 1 5 #:start? #f)
;(linear-seq 0 1 5 #:end? #f)
;(linear-seq 0 1 5 #:start? #f #:end? #f)
;(define xs (linear-seq -1 1 11))

;(plot (lines (map vector xs (map sqr xs))))

(parameterize
 ([plot-x-ticks (time-ticks)]
  [plot-y-far-ticks
   (ticks-scale(plot-y-ticks)
    (linear-scale 9/5 32)
    )
   ]
   [plot-y-label "Temperature (C)"]
   [plot-y-far-label "Temperature (F)"]
  )
 (define data
  (list
   #(0 0)
   #(15 0.6)
   #(30 9.5)
   #(45 10.0)
   #(60 16.6)
   #(75 41.6)
   #(90 42.7)
   #(105 65.5)
   #(120 78.9)
   #(135 78.9)
   #(150 131.1)
   #(165 151.1)
   #(180 176.2)
  )
 )
 (plot
  (list
   (function(lambda(x)(/(sqr x)180))
    0
    180
    #:style 'long-dash
    #:color 3
    #:label "Trend"
   )
  (lines data #:color 2 #:width 2)
  (points data #:color 1 #:line-width 2 #:label "Measured")
  )
  #:y-min -25 #:x-label "Time"
 )
)

(parameterize
```

```
 ([plot-y-transform
  (make-axis-transform
   (invertible-function sqrt sqr)
  )
 ]
 )
 (plot
  (function(lambda(x)x)0 5)
 )
)

(parameterize
 ([plot-x-transform
   (axis-transform-compose log-transform
    (collapse-transform 2 4)
   )
  ]
 )
 (plot
  (function(lambda(x)x)1 5)
 )
)

(parameterize
 ([plot-x-transform
   (axis-transform-bound log-transform 0.01 +inf.0)
  ]
 )
(plot
 (function
  (lambda(x)x)
  -4
  8
  #:label "y = x")
 )
)

(parameterize
 ([plot-x-transform
   (axis-transform-append
    (stretch-transform -2 -1 30)
    (stretch-transform 1 2 30)
    0
   )
  ]
 )
(plot
 (function(lambda(x)x)-3 3)
 )
)

(define(saddle x y z)(-(sqr x)(* 1/2(+(sqr y)(sqr z)))))

(parameterize
 ([plot-x-ticks no-ticks])
 (plot
  (list
   (function sin(- pi)pi)
   (x-ticks
    (list
     (tick(- pi)#t "-p")
```

```
     (tick(* -3/4 pi)#f "")
     (tick(* -1/2 pi)#t "-p/2")
     (tick(* -1/4 pi)#f "")
     (tick 0 #t "0")
     (tick(* 1/4 pi)#f "")
     (tick(* 1/2 pi)#t "p/2")
     (tick(* 3/4 pi)#f "")
     (tick pi #t "p")
    )
   )
  (axes)
  )
 )
)

(plot3d
 (isosurfaces3d saddle
   #:d-min -1
   #:d-max 1
   #:label ""
  )
 #:x-min -2
 #:x-max 2
 #:y-min -2
 #:y-max 2
 #:z-min -2
 #:z-max 2
)

(define(f x)(exp(* -1/2(sqr x))))
(define(f1 t)(vector(* 2(cos(* 4/5 t)))(* 2(sin(* 4/5 t)))))
(define(f2 t)(vector(* 1/2(cos t))(* 1/2(sin t))))
(define(f3 ?)(+ 1/2(* 1/6(cos(* 5 ?)))))
(define(f4 ?)(+ 1(* 1/4(cos(* 10 ?)))))

(parameterize
 ([plot-width 250]
  [plot-height 250]
  [plot-x-label #f]
  [plot-y-label #f]
 )
 (define xs(build-list 20(lambda _ (random))))
 (define ys(build-list 20(lambda _ (random))))
 (list
  (plot
   (list
    (function sin(- pi)pi)
    (invisible-rect #f #f -2 2)
   )
  )
 )
)

(plot3d
 (contour-intervals3d
  (lambda(x y)(+(sqr x)(sqr y)))
  -1.1
  1.1
  -1.1
  1.1
  #:label "z = x^2 + y^2"
```

```
 )
)


(plot3d
 (contours3d
  (lambda(x y)(+(sqr x)(sqr y)))
  -1.1 1.1 -1.1 1.1
  #:label "z=x^2+y^2"
 )
)

(plot
 (stacked-histogram
  (list
   #(a(1 1 1))
   #(b(1.5 3))
   #(c())
   #(d(1/2))
  )
  #:invert? #t
  #:labels '("Red" #f "Blue")
 )
 #:legend-anchor 'top-right
)

(plot
 (list
  (discrete-histogram
   '(
    #(Eggs 1.5)
    #(Bacon 2.5)
    #(Pancakes 3.5)
   )
   #:skip 2.5
   #:x-min 0
   #:label "AMD"
  )
  (discrete-histogram
   '(
    #(Eggs 1.4)
    #(Bacon 2.3)
    #(Pancakes 3.1)
   )
   #:skip 2.5
   #:x-min 1
   #:label "Intel"
   #:color 2
   #:line-color 2
  )
 )
 #:x-label "Breakfast Food"
 #:y-label "Cooking Time (minutes)"
 #:title "Cooking Times For Breakfast Food, Per Processor"
)

(plot
 (list
  (discrete-histogram
   (list
    #(a 1)
```

```
    #(b 2)
    #(c 3)
    #(d 2)
    #(e 4)
    #(f 2.5)
    #(g 1)
   )
  #:label "Numbers per letter"
  )
  (discrete-histogram
   (list
    #(1 1)
    #(4 2)
    #(3 1.5)
   )
   #:x-min 8
   #:label "Numbers per number"
   #:color 2 #:line-color 2
  )
 )
)

(plot
 (discrete-histogram
  (list
   #(A 1)
   #(B 2)
   #(B 3)
   (vector 'C(ivl 0.5 1.5))
  )
 )
)

;(plot(list(area-histogram f(linear-seq -4 4 10))(function f -4 4)))

(plot
 (rectangles
  (list
   (vector(ivl -1 1) (ivl -1 1))
   (vector (ivl 1 2) (ivl 1 2))
  )
 )
)

(plot
 (list
  (contour-intervals
   (lambda(x y)(-(sqr x)(sqr y)))
   -2
   2
   -2
   2
   #:label "z"
  )
  (vector-field
   (lambda(x y)(vector(* 2 x)(* -2 y)))
   #:color "black"
   #:label "Gradient"
  )
 )
)
```

```
(plot
 (list
  (polar-axes #:number 10)
  (polar-interval f3 f4 #:label "[f3,f4]")
 )
)

(plot(function sin(- pi)pi))

(plot(function sqr -2 2))

(plot
 (function(lambda(x)(sin(* 4 x)))-1 1)
 #:x-min -1.5
 #:x-max 1.5
 #:y-min -1.5
 #:y-max 1.5
)

(plot
 (vector-field
  (lambda(x y)(vector(+ x y)(- x y)))
  -2
  2
  -2
  2
 )
)

(plot
 (parametric-interval f1 f2(- pi)pi)
)

(plot
 (isoline
  (lambda(x y)(sqrt(+(sqr x)(sqr y))))
  1.5
  -2
  2
  -2
  2
  #:label "z"
 )
)

(plot(polar(lambda(lambda)1)))

(plot
 (parametric
  (lambda(t)(vector(cos t)(sin t)))
  0
  (* 2 pi)
 )
)

(plot
 (function-interval
  (lambda(x)0)
  (lambda(x)(exp(* -1/2(sqr x))))
```

```
  -4
  4
  #:line1-style 'transparent
 )
)

(plot
 (contours
 (lambda(x y)(-(sqr x)(sqr y)))
  -2
  2
  -2
  2
  #:label "z"
 )
)

(plot
 (inverse-interval sin
 (lambda(x)0)
  (- pi)
  pi
  #:line2-style 'long-dash
 )
)

(plot
 (list
 (tick-grid)
  (lines-interval
   (list
    #(0 0)
    #(1 1/2)
   )
   (list
    #(0 1)
    #(1 3/2)
   )
   #:color 4
   #:line1-color 4
   #:line2-color 4
   #:label "Parallelogram"
  )
 )
)

(plot
 (list
  (function
   (lambda(x)
    (cond
     [(or(x . < . -1)(x . > . 1))0]
     [(x . < . 0)(+ 1 x)]
     [(x . >= . 0)(- 1 x)]
    )
   )
   -1.5
   1.5
  #:label "Density"
 )
 (density
```

```
  (build-list
   2000
   (lambda(n)(-(+(random)(random))1))
  )
  #:color 0
  #:width 2
  #:style 'dot
  #:label "Est. density"
  )
 )
)

(plot
 (lines
  (reverse
   (for/fold
    ([lst(list(vector 0 0))])([i(in-range 1 200)])
    (match-define(vector x y)(first lst))
    (cons(vector i(+ y(* 1/100(-(random)1/2))))lst)
   )
  )
  #:color 6
  #:label "Random walk"
 )
)

(plot
 (list
  (function sqr 1 7)
  (error-bars
   (list
    (vector 2 4 12)
    (vector 4 16 20)
    (vector 6 36 10)
   )
  )
 )
)
```