Welcome to DrRacket, version 8.13 [cs].
Language: reader "../Scheme-PLUS-for-Racket/main/Scheme-PLUS-for-Racket/src/SRFI-105.rkt", with debugging; memory limit: 8192 MB.

```
SRFI-105 Curly Infix parser with optimization by Damien
MATTEI
(based on code from David A. Wheeler and Alan Manuel K.
Gloria.)

Options :

Infix optimizer is ON.
Infix optimizer on sliced containers is ON.

Possibly skipping some header's lines containing
space,tabs,new line,etc  or comments.

SRFI-105.rkt : number of skipped lines (comments, spaces,
directives,...) at header's beginning : 22

Parsed curly infix code result =

(module exo_retropropagationNhidden_layers_matrix_v2 racket
  (provide (all-defined-out))
  (require plot)
  (require (rename-in srfi/42 (: s42:)))
  (require (rename-in
            flomat
            (repeat repeat-flomat)
            (shape shape-flomat)
            (transpose transpose-flomat)))
  (require "matrix+.rkt")
  (require
"../Scheme-PLUS-for-Racket/main/Scheme-PLUS-for-Racket/src/
Scheme+.rkt")
  (define-overload-existing-operator +)
  (define-overload-procedure uniform)
  (overload-existing-operator + vector-append (vector?
vector?))
  (define (uniform-dummy dummy) (* (random) (if (= (random
2) 0) 1 -1)))
  (define (uniform-interval inf sup)
    (<- gap (- sup inf))
```

```
    (+ inf (* gap (random))))
  (overload-procedure uniform uniform-dummy (number?))
  (overload-procedure uniform uniform-interval (number?
number?))
  (define (σ ž) (/ 1 (+ 1 (exp (- ž)))))
  (define (der_tanh z ž) (- 1 (** z 2)))
  (define (der_σ z ž) (* z (- 1 z)))
  (define (der_atan z ž) (/ 1 (+ 1 (** ž 2))))
  (define ReseauRetroPropagation
    (class object%
      (super-new)
      (init-field
       (nc #(2 3 1))
       (nbiter 10000)
       (η 1.0)
       (activation_function_hidden_layer tanh)
       (activation_function_output_layer tanh)
       (activation_function_hidden_layer_derivative
der_tanh)
       (activation_function_output_layer_derivative
der_tanh))
      (<- lnc (vector-length nc))
      (field (z (vector-ec (:vector lg nc) (make-vector lg
0))))
      (display "z=")
      (display z)
      (newline)
      (field (ž (vector-ec (:vector lg nc) (make-vector lg
0))))
      (display "ž=")
      (display ž)
      (newline)
      (define-pointwise-unary uniform)
      (field
       (M
        (vector-ec
         (s42: n (- lnc 1))
         (.uniform!
          (zeros
           ($bracket-apply$next nc (+ n 1))
           (+ ($bracket-apply$next nc n) 1))))))
      (display "M=")
```

```
      (display M)
      (newline)
      (<- ∇ (for/vector ((lg nc)) (make-vector lg 0)))
      (display "∇=")
      (display ∇)
      (newline)
      (display "nbiter=")
      (display nbiter)
      (newline)
      (field (error 0))
      (define
       (accepte_et_propage x)
       (when (≠ (vector-length x) (vector-length
($bracket-apply$next z 0)))
         (display "Mauvais nombre d'entrées !")
         (newline)
         (exit #f))
       (<- ($bracket-apply$next z 0) x)
       (<- n (vector-length z))
       (declare z_1)
       (declare i)
       (for
        ((<- i 0) (< i (- n 2)) (<- i (+ i 1)))
        (<- z_1 (+ #(1) ($bracket-apply$next z i)))
        (<- ($bracket-apply$next ž (+ i 1)) (*
($bracket-apply$next M i) z_1))
        (<-
         ($bracket-apply$next z (+ i 1))
         (vector-map
          activation_function_hidden_layer
          ($bracket-apply$next ž (+ i 1)))))
       (<- z_1 (+ #(1) ($bracket-apply$next z i)))
       (<- ($bracket-apply$next ž (+ i 1)) (*
($bracket-apply$next M i) z_1))
       (<-
        ($bracket-apply$next z (+ i 1))
        (vector-map
         activation_function_output_layer
         ($bracket-apply$next ž (+ i 1)))))
      (define/public
       (apprentissage Lexemples)
       (<- ip 0)
```

```
       (declare x y)
       (for-racket
        ((it (in-range nbiter)))
        (when (= (% it 1000) 0) (display it) (newline))
        (<- err 0.0)
        (<- x (car ($bracket-apply$next Lexemples ip)))
        (<- y (cdr ($bracket-apply$next Lexemples ip)))
        (accepte_et_propage x)
        (<- i i_output_layer (- (vector-length z) 1))
        (<- ns (vector-length ($bracket-apply$next z i)))
        (for-racket
         ((k (in-range ns)))
         (<-
          ($bracket-apply$next ($bracket-apply$next ∇ i) k)
          (-
           ($bracket-apply$next y k)
           ($bracket-apply$next ($bracket-apply$next z i)
k)))
         (<-
          err
          (+ err (** ($bracket-apply$next
($bracket-apply$next ∇ i) k) 2))))
        (<- err (* err 0.5))
        (when (= it (- nbiter 1)) (<- error err))
        (<- ∂z꘲∂ž
activation_function_output_layer_derivative)
        (modification_des_poids
         ($bracket-apply$next M (- i 1))
         η꘹
         ($bracket-apply$next z (- i 1))
         ($bracket-apply$next z i)
         ($bracket-apply$next ž i)
         ($bracket-apply$next ∇ i)
         ∂z꘲∂ž)
        (<- ∂z꘲∂ž
activation_function_hidden_layer_derivative)
        (for-racket
         ((i (reversed (in-range 1 i_output_layer))))
         (<- nc (vector-length ($bracket-apply$next z i)))
         (<- ns (vector-length ($bracket-apply$next z (+ i
1))))
         (for-racket
```

```
          ((j (in-range nc)))
          (<-
           ($bracket-apply$next ($bracket-apply$next ∇ i)
j)
           (for/sum
            ((k (in-range ns)))
            (*
             (∂z²ᶜ꜀₆∂ž
              ($bracket-apply$next ($bracket-apply$next z
(+ i 1)) k)
              ($bracket-apply$next ($bracket-apply$next ž
(+ i 1)) k))
             ($bracket-apply$next ($bracket-apply$next M
i) k (+ j 1))
             ($bracket-apply$next ($bracket-apply$next ∇
(+ i 1)) k)))))
          (modification_des_poids
           ($bracket-apply$next M (- i 1))
           η²⁰₉ʙ
           ($bracket-apply$next z (- i 1))
           ($bracket-apply$next z i)
           ($bracket-apply$next ž i)
           ($bracket-apply$next ∇ i)
           ∂z²ᶜ꜀₆∂ž))
         (<- ip (random (vector-length Lexemples)))))
      (define
       (modification_des_poids M_i_o η z_input z_output
ž_output ∇_i_o ∂z²ᶜ꜀₆∂ž)
       (<- (len_layer_output len_layer_input_plus1forBias)
(dim M_i_o))
       (<- len_layer_input (- len_layer_input_plus1forBias
1))
       (for-racket
        ((j (in-range len_layer_output)))
        (for-racket
         ((i (in-range len_layer_input)))
         (<-
          ($bracket-apply$next M_i_o j (+ i 1))
          (-
           ($bracket-apply$next M_i_o j (+ i 1))
           (*
            (- η)
```

```
      ($bracket-apply$next z_input i)
      (∂z²ᶜ₆∂ž
       ($bracket-apply$next z_output j)
       ($bracket-apply$next ž_output j))
      ($bracket-apply$next ∇_i_o j)))))
  (<-
   ($bracket-apply$next M_i_o j 0)
   (-
    ($bracket-apply$next M_i_o j 0)
    (*
     (- η)
     1.0
     (∂z²ᶜ₆∂ž
      ($bracket-apply$next z_output j)
      ($bracket-apply$next ž_output j))
     ($bracket-apply$next ∇_i_o j))))))
(define/public
 (test Lexemples)
 (display "Test des exemples :")
 (newline)
 (<- err 0)
 (declare entree sortie_attendue ∇)
 (for-racket
  ((entree-sortie_attendue Lexemples))
  (<- entree (car entree-sortie_attendue))
  (<- sortie_attendue (cdr entree-sortie_attendue))
  (accepte_et_propage entree)
  (printf
   "~a --> ~a : on attendait ~a"
   entree
   ($bracket-apply$next z (- (vector-length z) 1))
   sortie_attendue)
  (newline)
  (<-
   ∇
   (-
    ($bracket-apply$next sortie_attendue 0)
    ($bracket-apply$next
     ($bracket-apply$next z (- (vector-length z) 1))
     0)))
  (<- error (+ error (** ∇ 2))))
```

```
       (<- err (* err 0.5))
       (display "Error on examples=")
       (display error)
       (newline))
      (define/public
       (DL-data-2D)
       (list-ec
        (s42: n 100)
        ($+>
         (<- xp (+ (/ (- pi) 2) (* pi (/ n 100))))
         (accepte_et_propage (vector xp))
         (<-
          xp-DL
          ($bracket-apply$next
           ($bracket-apply$next z (- (vector-length z) 1))
           0))
         (vector xp xp-DL))))
      (define/public
       (DL-plot)
       (<- Lplot-DL (DL-data-2D))
       (plot (points Lplot-DL #:sym 'fullcircle1 #:color
"red")))))
  (printf "################### NOT ###################")
  (newline)
  (<-
   r1
   (new
    ReseauRetroPropagation
    (nc #(1 2 1))
    (nbiter 5000)
    (η 10)
    (activation_function_hidden_layer σ)
    (activation_function_output_layer σ)
    (activation_function_hidden_layer_derivative der_σ)
    (activation_function_output_layer_derivative der_σ)))
  (<- Lexemples1 #((#(1) . #(0)) (#(0) . #(1))))
  (send r1 apprentissage Lexemples1)
  (send r1 test Lexemples1)
  (<- precision 100.0)
  (display "precision=")
  (display precision)
  (newline)
```

```
  (define (trunc x) (/ (round (* precision x)) precision))
  (define-pointwise-unary trunc)
  (define (trunc-matrix mt) (.trunc! mt))
  (<- M (get-field M r1))
  (newline)
  (display "Matrix vector M=")
  (newline)
  (display M)
  (newline)
  (for-racket ((mt M)) (trunc-matrix mt))
  (display "Matrix vector modified M=")
  (newline)
  (display M)
  (newline)
  (send r1 test Lexemples1)
  (newline)
  (printf "################### XOR ###################")
  (newline)
  (<-
   r2
   (new
    ReseauRetroPropagation
    (nc #(2 3 1))
    (nbiter 250000)
    (η 10)
    (activation_function_hidden_layer σ)
    (activation_function_output_layer σ)
    (activation_function_hidden_layer_derivative der_σ)
    (activation_function_output_layer_derivative der_σ)))
  (<-
   Lexemples2
   #((#(1 0) . #(1)) (#(0 0) . #(0)) (#(0 1) . #(1)) (#(1
1) . #(0))))
  (send r2 apprentissage Lexemples2)
  (send r2 test Lexemples2)
  (<- M (get-field M r2))
  (newline)
  (display "Matrix vector M=")
  (newline)
  (display M)
  (newline)
  (for-racket ((mt M)) (trunc-matrix mt))
```

```
  (display "Matrix vector modified M=")
  (newline)
  (display M)
  (newline)
  (send r2 test Lexemples2)
  (newline)
  (printf "################### SINUS ###################")
  (newline)
  (<-
   r3
   (new
    ReseauRetroPropagation
    (nc #(1 70 70 1))
    (nbiter 50000)
    (η 0.01)
    (activation_function_hidden_layer atan)
    (activation_function_output_layer tanh)
    (activation_function_hidden_layer_derivative der_atan)
    (activation_function_output_layer_derivative
der_tanh)))
  (<-
   Llearning
   (vector-ec
    (:list x (list-ec (s42: n 10000) (uniform (- pi) pi)))
    (cons (vector x) (vector (sin x)))))
  (<-
   Ltest
   (vector-ec
    (:list x (list-ec (s42: n 10) (uniform (/ (- pi) 2) (/
pi 2))))
    (cons (vector x) (vector (sin x)))))
  (<-
   Lplot-sin
   (list-ec
    (s42: n 100)
    ($+> (<- xp (+ (/ (- pi) 2) (* pi (/ n 100)))) (vector
xp (sin xp)))))
  (display "Lplot-sin =")
  (newline)
  (display Lplot-sin)
  (newline)
  (plot
```

```
   (points Lplot-sin #:sym 'fullcircle1 #:color "blue"
#:label "y = sin(x)"))
  (send r3 apprentissage Llearning)
  (send r3 test Ltest)
  (send r3 DL-plot)
  (<- Lplot-DL-main (send r3 DL-data-2D))
  (plot
   (list
    (points Lplot-sin #:sym 'fullcircle1 #:color "blue"
#:label "y = sin(x)")
    (points Lplot-DL-main #:sym 'circle1 #:color "red"
#:label "neural sine")))
  (<- M (get-field M r3))
  (newline)
  (display "Matrix vector M=")
  (newline)
  (display M)
  (newline)
  (<- precision 1000.0)
  (display "precision=")
  (display precision)
  (newline)
  (define (trunc3 x) (/ (round (* precision x)) precision))
  (define-pointwise-unary trunc3)
  (define (trunc3-matrix mt) (.trunc3! mt))
  (for-racket ((mt M)) (trunc3-matrix mt))
  (display "Matrix vector modified M=")
  (newline)
  (display M)
  (newline)
  (send r3 test Ltest)
  (newline)
  (<- Lplot-DL-trunc (send r3 DL-data-2D))
  (plot
   (list
    (points
     Lplot-DL-trunc
     #:sym
     'circle1
     #:color
     "green"
     #:label
```

```
     "neural sine - matrices with truncated numbers")
    (points
     Lplot-DL-main
     #:sym
     'circle1
     #:color
     "red"
     #:label
     "neural sine"))))

################### NOT ###################
z=#(#(0) #(0 0) #(0))
ž=#(#(0) #(0 0) #(0))
M=#((flomat: ((0.6447519124737936 -0.3786303863756173 6)
(0.7417571261258522 0.9316090421692191))) (flomat:
((-0.28758622841395803 -0.0315167947103021
-0.11238479460031663))))
∇=#(#(0) #(0 0) #(0))
nbiter=5000
0
1000
2000
3000
4000
Test des exemples :
#(1) --> #(0.004587895867126144) : on attendait #(0)
#(0) --> #(0.9947643232569748) : on attendait #(1)
Error on examples=6.218308379232284e-5
precision=100.0

Matrix vector M=
#((flomat: ((1.8152910234817026 -3.1554263757650522)
(-3.0327762531886866 6.228674280658298))) (flomat:
((2.1746890148332936 4.039406836147639
-8.736057058144336))))
Matrix vector modified M=
#((flomat: ((1.82 -3.16) (-3.03 6.23))) (flomat: ((2.17
4.04 -8.74))))
Test des exemples :
#(1) --> #(0.004544191323881111) : on attendait #(0)
#(0) --> #(0.9947479494410946) : on attendait #(1)
Error on examples=0.0001104167936536580 3
```

```
################## XOR ##################
z=#(#(0 0) #(0 0 0) #(0))
ž=#(#(0 0) #(0 0 0) #(0))
M=#((flomat: ((0.4921374168164523 -0.42807507795272776
-0.21352478266068614) (-0.25840107275811564
0.2892869429596896 -0.5136386968281235)
(-0.7564756631261991 -0.4072549577590617
0.3812685288265008))) (flomat: ((-0.369136149012558
-0.6092928451795392 -0.4507292298953235
0.8905182760739238))))
∇=#(#(0 0) #(0 0 0) #(0))
nbiter=250000
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
21000
22000
23000
24000
25000
26000
27000
28000
```

29000
30000
31000
32000
33000
34000
35000
36000
37000
38000
39000
40000
41000
42000
43000
44000
45000
46000
47000
48000
49000
50000
51000
52000
53000
54000
55000
56000
57000
58000
59000
60000
61000
62000
63000
64000
65000
66000
67000
68000
69000
70000

71000
72000
73000
74000
75000
76000
77000
78000
79000
80000
81000
82000
83000
84000
85000
86000
87000
88000
89000
90000
91000
92000
93000
94000
95000
96000
97000
98000
99000
100000
101000
102000
103000
104000
105000
106000
107000
108000
109000
110000
111000
112000

113000
114000
115000
116000
117000
118000
119000
120000
121000
122000
123000
124000
125000
126000
127000
128000
129000
130000
131000
132000
133000
134000
135000
136000
137000
138000
139000
140000
141000
142000
143000
144000
145000
146000
147000
148000
149000
150000
151000
152000
153000
154000

155000
156000
157000
158000
159000
160000
161000
162000
163000
164000
165000
166000
167000
168000
169000
170000
171000
172000
173000
174000
175000
176000
177000
178000
179000
180000
181000
182000
183000
184000
185000
186000
187000
188000
189000
190000
191000
192000
193000
194000
195000
196000

197000
198000
199000
200000
201000
202000
203000
204000
205000
206000
207000
208000
209000
210000
211000
212000
213000
214000
215000
216000
217000
218000
219000
220000
221000
222000
223000
224000
225000
226000
227000
228000
229000
230000
231000
232000
233000
234000
235000
236000
237000
238000

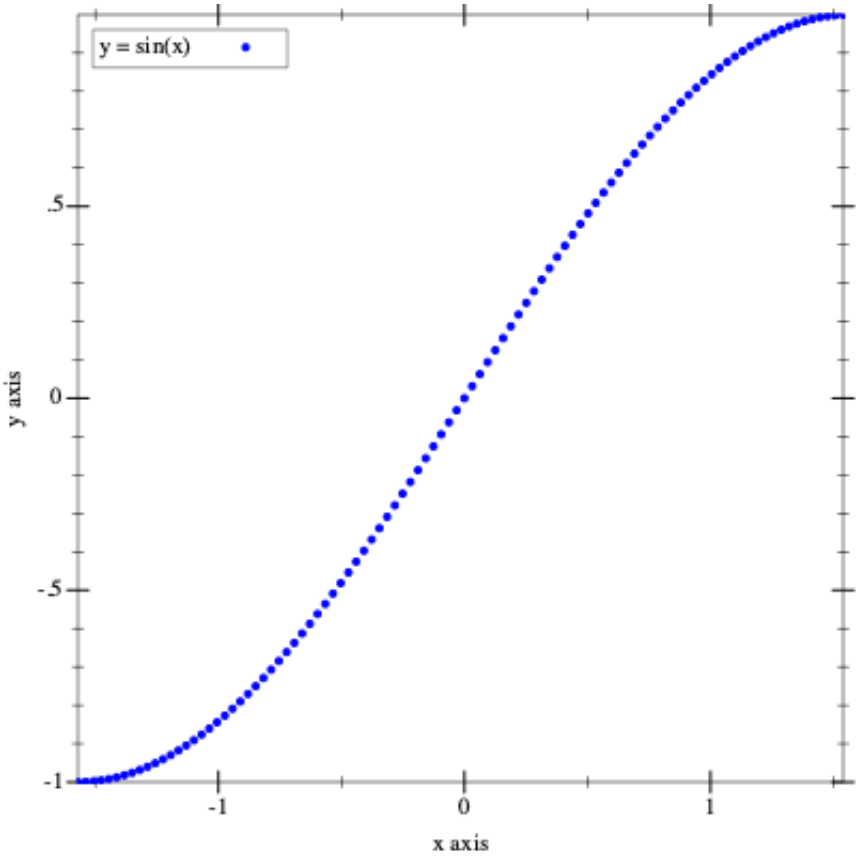
```
239000
240000
241000
242000
243000
244000
245000
246000
247000
248000
249000
Test des exemples :
#(1 0) --> #(0.9982996037110752) : on attendait #(1)
#(0 0) --> #(0.0003641429931353 1826) : on attendait #(0)
#(0 1) --> #(0.9983048494490441) : on attendait #(1)
#(1 1) --> #(0.0021436949699986 68) : on attendait #(0)
Error on examples=1.1938845594195027e-5

Matrix vector M=
#((flomat: ((-0.46650473236101786 -3.42537392369584
8.777321143980888) (0.3094240945499729 9.35224466228428
-4.402127974981613) (-2.422287576212709 9.07129105561903
8.580628866023433))) (flomat: ((3.2002030916915065
-12.961918317770323 -12.929118559430576
16.383702415791838))))
Matrix vector modified M=
#((flomat: ((-0.47 -3.43 8.78) (0.31 9.35 -4.4) (-2.42
9.07 8.58))) (flomat: ((3.2 -12.96 -12.93 16.38))))
Test des exemples :
#(1 0) --> #(0.9982950713443539) : on attendait #(1)
#(0 0) --> #(0.0003683318135450472) : on attendait #(0)
#(0 1) --> #(0.9983006136610401) : on attendait #(1)
#(1 1) --> #(0.0021386043875128167) : on attendait #(0)
Error on examples=2.2442838295240325e-5

################### SINUS ###################
z=#(#(0) #(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0) #(0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0) #(0))
```

```
z̃=#(#(0) #(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0) #(0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0) #(0))
M=#((flomat 70 2 ...) (flomat 70 71 ...) (flomat:
((-0.6802761916763727 -0.4343445115591536
-0.6620598758357704 0.5491575911233153
0.013107789616663998 0.12428473002538641
0.8151242315177434 0.9737410479081186 0.25088104190846366
-0.9128696545671878 -0.5586320716411507
-0.9093501565849486 0.28614271304516226
-0.20020562448603332 -0.04740871089999841
0.11385964036053169 0.087950010153838 -0.22027176311624394
-0.44224552577060405 0.32123357751792864
0.49946483408303133 0.805434004294284 0.9572263392859788
-0.36667783750896116 -0.7582132196306135
0.48744830870750555 0.477443069291338 0.19655593947592087
0.997420256832478 -0.49348361479225383 0.16883526256255216
-0.649191766053412 0.2493853273038166 -0.05173738365093614
-0.46473805295897536 -0.44718997902598134
-0.6705456922001913 -0.7951295635166926
-0.11392906696937186 0.8446318636842602 0.9760684303525449
0.389290780055449 -0.8572008978337485 -0.6389022597325198
-0.5819317698576042 -0.82954836416665 0.3252340666131782
-0.36304657755272657 0.3076102544979502
-0.2556291909820567 -0.7877106098094506
-0.6010812667722124 -0.36567924056697687
0.9455558284361877 0.38134638856166253 -0.350876045409175
-0.6311735066781029 0.2505886473512367 -0.6696135630550845
0.2361602711308134 0.4911660249723432 0.12558575885413165
0.8191048997393389 0.8184686611037426 -0.8514811876481603
-0.9830846021607512 -0.14689028252688674
0.4050332701874246 0.1658776121918446 0.16939672530501124
0.6670648818252365))))
∇=#(#(0) #(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0) #(0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0) #(0))
```

```
nbiter=50000
Lplot-sin =
(#(-1.570796326794896 -1.0) #(-1.5393804002589986
-0.9995065603657316) #(-1.5079644737231006
-0.9980267284282716) #(-1.4765485471872029
-0.99556196460308) #(-1.4451326206513049
-0.9921147013144779) #(-1.413716694115407
-0.9876883405951378) #(-1.382300767579509
-0.9822872507286886) #(-1.350884841043611
-0.9759167619387473) #(-1.3194689145077132
-0.9685831611286311) #(-1.2880529879718152
-0.9602936856769431) #(-1.2566370614359172
-0.9510565162951535) #(-1.2252211349000193
-0.9408807689542255) #(-1.1938052083641213
-0.9297764858882513) #(-1.1623892818282235
-0.9177546256839811) #(-1.1309733552923253
-0.9048270524660195) #(-1.099557428756427 6
-0.8910065241883679) #(-1.0681415022205296
-0.8763066800438636) #(-1.0367255756846316
-0.8607420270039435) #(-1.0053096491487339
-0.8443279255020151) #(-0.9738937226128359
-0.8270805742745618) #(-0.9424777960769379
-0.8090169943749475) #(-0.9110618695410401
-0.7901550123756904) #(-0.8796459430051421
-0.7705132427757893) #(-0.8482300164692441
-0.7501110696304596) #(-0.8168140899333463
-0.7289686274214116) #(-0.7853981633974483
-0.7071067811865475) #(-0.7539822368615503
-0.6845471059286886) #(-0.7225663103256523
-0.6613118653236518) #(-0.6911503837897544
-0.6374239897486896) #(-0.6597344572538566
-0.6129070536529765) #(-0.6283185307179586
-0.5877852522924731) #(-0.5969026041820606
-0.5620833778521306) #(-0.5654866776461627
-0.5358267949789965) #(-0.5340707511102647
-0.5090414157503712) #(-0.5026548245743667
-0.4817536741017151) #(-0.4712388980384689 7
-0.4539904997395468) #(-0.439822971502571
-0.4257792915650726) #(-0.408407044966673
-0.3971478906347805) #(-0.37699111843077526
-0.3681245526846780 3) #(-0.345575191894877 3
-0.3387379202452914) #(-0.3141592653589793
```

-0.3090169943749474) #(-0.2827433382308156
-0.2789911060392294) #(-0.2513274122871836
-0.2486898871648549) #(-0.2199114857512856
-0.21814324139654262) #(-0.1884955921538763
-0.18738131458572466) #(-0.15707963267948966
-0.1564344650402387) #(-0.12566370614359168
-0.1253332335643042) #(-0.09424777960769393
-0.09410831331851445) #(-0.06283185307179595
-0.06279051952931346) #(-0.031415926535897976
-0.03141075907812834) #(0.0 0.0) #(0.031415926535897976
0.03141075907812834) #(0.06283185307179595
0.06279051952931346) #(0.09424777960769393
0.09410831331851445) #(0.1256637061435919
0.12533323356430442) #(0.15707963267948988
0.1564344650402311) #(0.1884955921538785
0.18738131458572488) #(0.2199114857512538
0.21814324139654243) #(0.2513274122871836
0.24868988716485468) #(0.2827433382308134
0.2789911060392292) #(0.3141592653589793
0.3090169943749474) #(0.3455751918948773
0.3387379202452914) #(0.37699111843077526
0.36812455268467803) #(0.40840704496673
0.3971478906347805) #(0.4398229715025712
0.42577929156507277) #(0.4712388980384689
0.4539904997395468) #(0.5026548245743672
0.4817536741017155) #(0.5340707511102649
0.5090414157503713) #(0.5654866776461631
0.5358267949789969) #(0.5969026041820604
0.5620833778521304) #(0.6283185307179586
0.5877852522924731) #(0.6597344572538564
0.6129070536529764) #(0.6911503837897546
0.6374239897486897) #(0.7225663103256523
0.6613118653236518) #(0.7539822368615505
0.6845471059286887) #(0.7853981633974483
0.7071067811865475) #(0.816814089933346
0.7289686274214113) #(0.8482300164692442
0.7501110696304596) #(0.879645943005142
0.7705132427757891) #(0.9110618695410402
0.7901550123756904) #(0.9424777960769379
0.8090169943749475) #(0.9738937226128361
0.8270805742745619) #(1.005309649148734
0.8443279255020149) #(1.0367255756846316

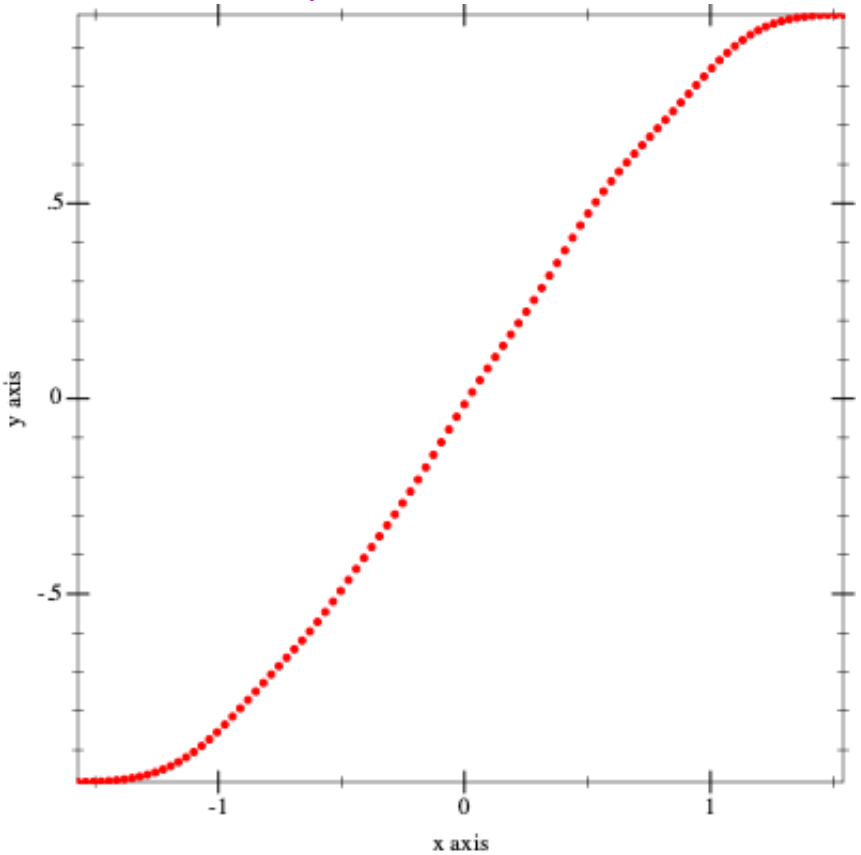
```
0.8607420270039435) #(1.0681415022205294
0.8763066800438635) #(1.0995574287564276
0.8910065241883679) #(1.1309733552923253
0.9048270524660195) #(1.1623892818282235
0.9177546256839811) #(1.1938052083641213
0.9297764858882513) #(1.2252211349000195
0.9408807689542256) #(1.2566370614359172
0.9510565162951535) #(1.2880529879718154
0.9602936856769432) #(1.3194689145077132
0.9685831611286311) #(1.3508848410436114
0.9759167619387474) #(1.3823007675795087
0.9822872507286886) #(1.413716694115407
0.9876883405951378) #(1.4451326206513047
0.9921147013144778) #(1.4765485471872029 0.99556196460308)
#(1.5079644737231006 0.9980267284282716)
#(1.5393804002589988 0.9995065603657316))
```

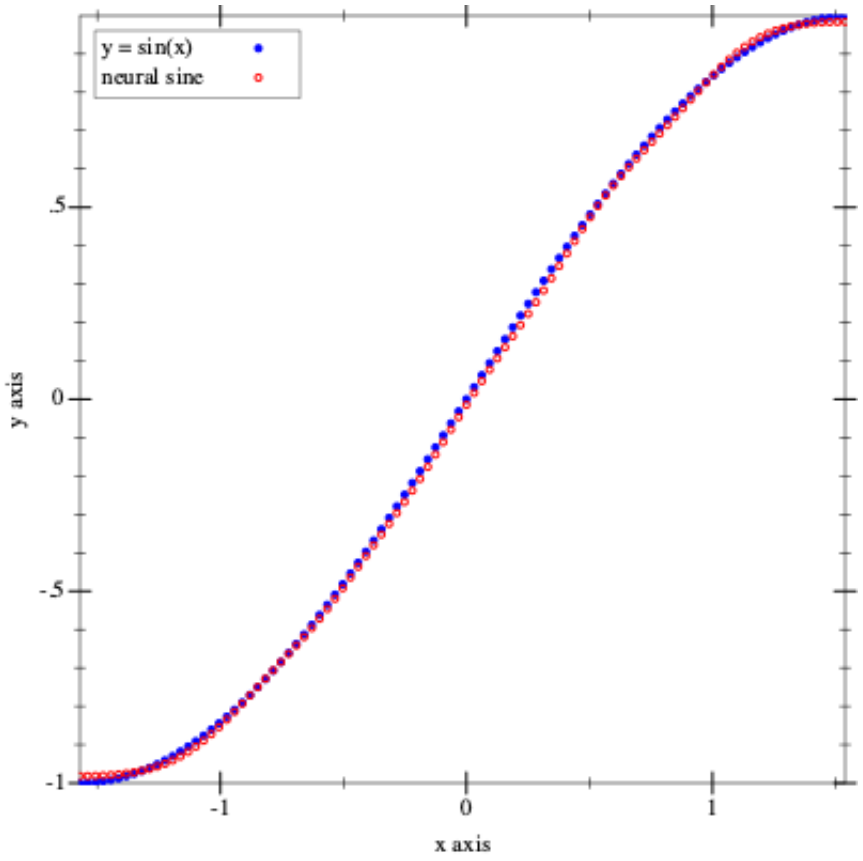



```
0
1000
2000
3000
4000
5000
6000
7000
```

8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
21000
22000
23000
24000
25000
26000
27000
28000
29000
30000
31000
32000
33000
34000
35000
36000
37000
38000
39000
40000
41000
42000
43000
44000
45000
46000
47000
48000
49000

```
Test des exemples :
#(-0.5248011714325682) --> #(-0.511697826724621) : on
attendait #(-0.5010409436480776)
#(-1.3379378026775404) --> #(-0.9704562812353577) : on
attendait #(-0.9730107386543255)
#(0.40632861172910606) --> #(0.3776398244344421) : on
attendait #(0.3952395424739412)
#(-0.6083331511942834) --> #(-0.5808777380352484) : on
attendait #(-0.5715004355905013)
#(-0.3008974370260844) --> #(-0.3126970667714175) : on
attendait #(-0.296377441878989)
#(-0.7479160169861759) --> #(-0.6807551822748271) : on
attendait #(-0.6801124537799255)
#(1.4354078821377456) --> #(0.9815397361762372) : on
attendait #(0.9908489755604684)
#(-0.6426716627825968) --> #(-0.606964141325650 4) : on
attendait #(-0.599336236873342)
#(-0.08369883581716264) --> #(-0.100615423729940 3) : on
attendait #(-0.083601144744553 67)
#(1.1867540244821444) --> #(0.9408665614611825) : on
attendait #(0.9271576797121787)
Error on examples=0.0015390590410949 64
```

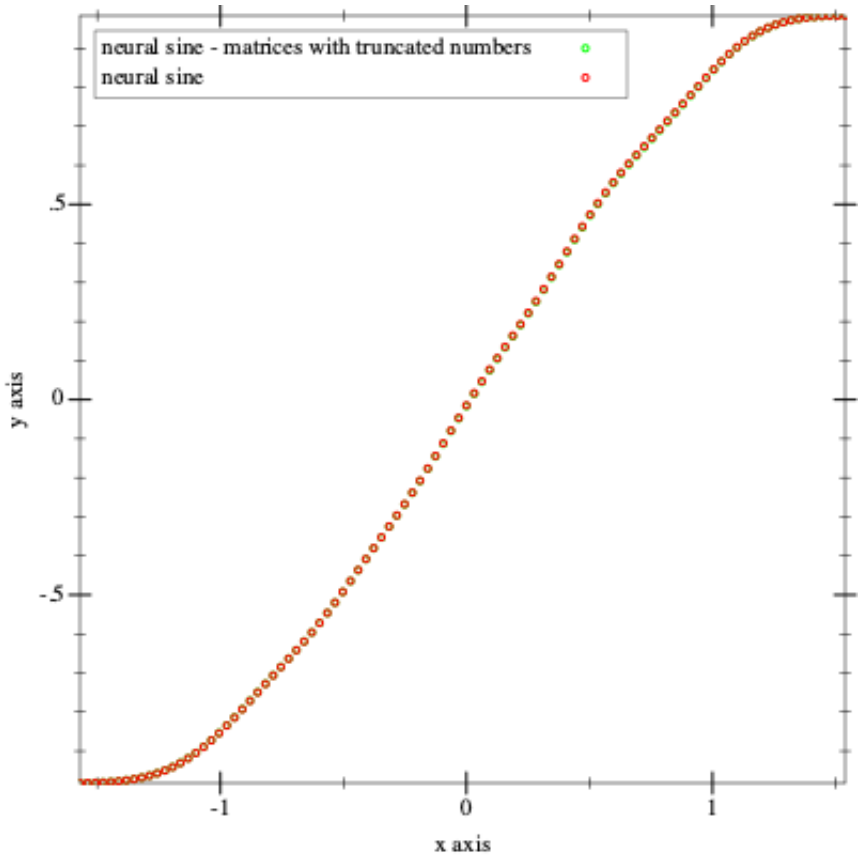





```
Matrix vector M=
#((flomat 70 2 ...) (flomat 70 71 ...) (flomat:
((-0.6619807901582975 -0.4737768371139214
-0.7078011565109767 0.5791929935098333
-0.03935637226518891 0.16777652387129116
0.7997643516588997 0.9804547263186936 0.2994968327318799
-0.8991063079717566 -0.5730044815690072
-0.9217502859178812 0.23583358938652954
-0.17151102185021743 -0.10257283522008478
0.09594303895683351 0.11519689291565628
-0.1418571542200748 -0.33888755929676706
0.37532846189470714 0.4527198383947423 0.80780274978186
0.8914288834552173 -0.322802018089066 -0.7068875761963405
0.4576912290974067 0.4813823822125027 0.1470182925760294
0.9325935703512306 -0.4360719171360888 0.11910593931320643
-0.5894023374540589 0.2598412711488805 0.03523066361372713
-0.4119550497418043 -0.39730242442664665
-0.6786486893144837 -0.8106296361859988
-0.13594731538301716 0.8598534621501353 1.0032341402375573
0.460100598566314 -0.8675590342046242 -0.6970879036585963
-0.5426707567016125 -0.823298099352991 0.3172527016640863
-0.3393079103963251 0.30226128085533654
-0.26749645438365693 -0.8137505417842985
```

```
-0.6610973715674748 -0.33723588630517415
0.8763280389294331 0.4398993777073877 -0.3650706649617116
-0.6879368538347391 0.16385891459581572
-0.6446146608843102 0.25666388736308443 0.4978582419589434
0.1288641247813719 0.7881651525682878 0.8065061036524872
-0.7937868210849799 -0.9026399473415112
-0.14814465072659982 0.4546796543220005
0.10663826254270432 0.1825506822628948
0.6677809417025773))))
precision=1000.0
Matrix vector modified M=
#((flomat 70 2 ...) (flomat 70 71 ...) (flomat: ((-0.662
-0.474 -0.708 0.579 -0.039 0.168 0.8 0.98 0.299 -0.899
-0.573 -0.922 0.236 -0.172 -0.103 0.096 0.115 -0.142
-0.339 0.375 0.453 0.808 0.891 -0.323 -0.707 0.458 0.481
0.147 0.933 -0.436 0.119 -0.589 0.26 0.035 -0.412 -0.397
-0.679 -0.811 -0.136 0.86 1.003 0.46 -0.868 -0.697 -0.543
-0.823 0.317 -0.339 0.302 -0.267 -0.814 -0.661 -0.337
0.876 0.44 -0.365 -0.688 0.164 -0.645 0.257 0.498 0.129
0.788 0.807 -0.794 -0.903 -0.148 0.455 0.107 0.183
0.668))))
Test des exemples :
#(-0.5248011714325682) --> #(-0.51191228899564) : on
attendait #(-0.5010409436480776)
#(-1.3379378026775404) --> #(-0.9703078541075946) : on
attendait #(-0.9730107386543255)
#(0.4063286117291060 6) --> #(0.3753566273293885 7) : on
attendait #(0.3952395424739412)
#(-0.6083331511942834) --> #(-0.5809191644180132) : on
attendait #(-0.5715004355905013)
#(-0.3008974370260844) --> #(-0.31342246870674) : on
attendait #(-0.296377441878989)
#(-0.7479160169861759) --> #(-0.6804990500291508) : on
attendait #(-0.6801124537799255)
#(1.4354078821377456) --> #(0.9813359934526824) : on
attendait #(0.9908489755604684)
#(-0.6426716627825968) --> #(-0.6069325887998773) : on
attendait #(-0.599336236873342)
#(-0.083698835817162 64) --> #(-0.1020898911855870 6) : on
attendait #(-0.0836011447445536 7)
#(1.1867540244821444) --> #(0.940142782025123) : on
attendait #(0.9271576797121787)
```

Error on examples=0.0030979239574375275



>