21801744
Uğur Erdem Seyfi
CS315 - 01

CS 315 Homework 03 Report

In this homework, I want to implement sin(x) as a summation in form of a Taylor series, however for that, I need to be able to also call factorial and exponential functions, since it is not allowed to call functions that are already built-in in the language (such as expt), I will built these necessary functions in the “Helper functions” part and than implement the main code that I wanted to demonstrate in the “Sum function” part.

Remark:

$$\sin(x) = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)!} x^{2i+1}$$
$$= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

1. Helper functions

The functions in this part were created so they help us in the second part. Now since identity function (f(x) = x) is also a function, and we can directly put x as an argument, we can be also considered as we are giving an example for product-of-f kind of an example in 1.1 and 1.2, so I won’t create another function for this.

1.1 exponential function:

```
(define exponential
        (lambda (base power)(
                if (= power 0)
                1
                (* base (exponential base (- power 1)))
                )
        )
)

(display (exponential -1 1)) ; -
(newline)
(display (exponential -1 0)) ; -1
(newline)
(display (exponential 5 2)) ; 5^2
(newline)
```

1.2 factorial function:

```
(define factorial
(lambda (n)(
                if(= n 0)
                1
                (* n (factorial (- n 1)))
            )
        )
)

(display (factorial 6)) ; 720
(newline)
```

2. Sum function

Since I wanted to implement sin(x) as a summation in form of a Taylor series, I needed to implement a function that will give each respective term in the formula above. Now since each term is determined by two parameters, *x* and *i* we need to build our term generating function (f in the homework) based on that.

Term function

```
(define term (
        lambda( x i) (* (/ (exponential -1 i) (factorial (+ (* 2 i) 1)) ) (exponential x (+ (* 2 i) 1)))
    )
)
```

sum-of-term and sum-of-term-tr

```
(define sum-of-term
        (lambda( radian iteration) (
                if (<= iteration -1)
                0
                (+ (term radian iteration) (sum-of-term radian (- iteration 1)))
                )
        )
)

(display "sum-of-term results \n")
(display (sum-of-term (/ PI 2) 1) )
(newline)
(display (sum-of-term (/ PI 2) 5) )
(newline)
(display (sum-of-term (/ PI 2) 10) )
(newline)
(display (sum-of-term PI 1) )
(newline)
(display (sum-of-term PI 5) )
(newline)
```

```
(display (sum-of-term PI 10) )
(newline)

;sum-of-term results
;0.9250178333333334
;0.9999996270418703
;0.9999996829318349
;-2.019857333333333
;0.0011503970403433502
;0.0015926529267152592


(define sum-of-term-tr
        (lambda (radian iteration accumulator) (
                if (<= iteration -1)
                accumulator
                (sum-of-term-tr radian (- iteration 1) (+ accumulator (term radian iteration)))
                )
        )
)

(display "sum-of-term-tr results \n")
(display (sum-of-term-tr (/ PI 2) 1 0) )
(newline)
(display (sum-of-term-tr (/ PI 2) 5 0) )
(newline)
(display (sum-of-term-tr (/ PI 2) 10 0) )
(newline)
(display (sum-of-term-tr (/ PI 2) 1 0) )
(newline)
(display (sum-of-term-tr (/ PI 2) 5 0) )
(newline)
(display (sum-of-term-tr PI 10 0))
(newline)

;sum-of-term-tr results
;0.9250178333333334
;0.9999996270418701
;0.9999996829318347
;0.9250178333333334
;0.9999996270418701
;0.0015926529267153988
```

Tracing with trace

(require racket/trace)
(trace sum-of-term)
(trace sum-of-term-tr)

(sum-of-term (/ PI 2) 10)
(sum-of-term-tr (/ PI 2) 10 0)

After running this command, we encounter the following output:

```
>(sum-of-term 1.57 10)
> (sum-of-term 1.57 9)
> >(sum-of-term 1.57 8)
> > (sum-of-term 1.57 7)
> > >(sum-of-term 1.57 6)
> > > (sum-of-term 1.57 5)
> > > >(sum-of-term 1.57 4)
> > > > (sum-of-term 1.57 3)
> > > > >(sum-of-term 1.57 2)
> > > > > (sum-of-term 1.57 1)
> > > >[10] (sum-of-term 1.57 0)
> > > >[11] (sum-of-term 1.57 -1)
< < < <[11] 0
< < < <[10] 1.57
< < < < < 0.9250178333333334
< < < < <1.0045086604641669
< < < < 0.9998434952309576
< < < <1.0000032058668373
< < < 0.9999996270418703
< < <0.9999996835895989
< < 0.9999996829258632
< <0.999999682931878
< 0.9999996829318347
<0.9999996829318349
>(sum-of-term-tr 1.57 10 0)
>(sum-of-term-tr 1.57 9 2.5441855928783502e-16)
>(sum-of-term-tr 1.57 8 -4.309654695691594e-14)
>(sum-of-term-tr 1.57 7 5.971764180270026e-12)
>(sum-of-term-tr 1.57 6 -6.577639321180497e-10)
>(sum-of-term-tr 1.57 5 5.588996466646491e-08)
>(sum-of-term-tr 1.57 4 -3.5229350024230484e-06)
>(sum-of-term-tr 1.57 3 0.00015618770087726634)
>(sum-of-term-tr 1.57 2 -0.004508977532332046)
>(sum-of-term-tr 1.57 1 0.07498184959850128)
>(sum-of-term-tr 1.57 0 -0.5700003170681653)
>(sum-of-term-tr 1.57 -1 0.9999996829318347)
<0.9999996829318347
```

Now as you can see, as the number of terms that are added increases (which is denoted by “iteration” parameter), the result gets closer to the actual one. As you can see, for 75 iteration pi/2 gets closer to 1 and pi gets closer to 0, which is exactly what we wanted! So in essence we implemented sin formula as a sum of terms described above.

So we are done.