アルゴリズムとデータ構造入門 第五回課題

1029-24-9540 山崎啓太郎

February 7, 2013

1 Section 1.29

```
1  (define sum (lambda (term a next b)
2          (define iter (lambda(a result)
3                  (if (> a b)
4                          result
5                          (iter (next a) (+ result (term a)))
6                  )
7          ))
8          (iter a 0)
9  ))
10
11 (define simpson-integral (lambda(f a b n)
12         (define h (/ (- b a) n))
13         (define term (lambda (k)
14                 (f (+ a (* h k)))
15         ))
16         (define next (lambda(k)
17                 (+ k 2)
18         ))
19         (* (/ h 3) (+
20                 (f a)
21                 (* 4 (sum term 1 next (- n 1)))
22                 (* 2 (sum term 2 next (- n 2)))
23                 (f b)
24         ))
25 ))
26
27 (define integral (lambda(f a b n)
28         (define h (/ (- b a) n))
29         (* (sum (lambda (k) (f (+ a (* h k))))
```

```
30                      0
31                      (lambda (k) (+ k 1))
32                      n)
33              h)
34  ))
```

出力結果
(simpson-integral (lambda (x) (* x x x)) 0 1 100) => 0.25
(simpson-integral (lambda (x) (* x x x)) 0 1 1000) => 0.25
(integral (lambda (x) (* x x x)) 0 1 1000) => 0.25050025000000004
(integral (lambda (x) (* x x x)) 0 1 100) => 0.25502500000000006

比較結果
integral は n を増やすごとに 0.25 に近づいていることがわかる。
simpson の公式を使った場合、分割数が少なくても正確な値が出る。

2 Section 1.31

```
1  (define product-iter (lambda (term a next b)
2          (define iter (lambda(a result)
3                  (if (> a b)
4                          result
5                          (iter (next a) (* result (term a)))
6                  )
7          ))
8          (iter a 1)
9  ))
10
11 (define product-recur (lambda (term a next b)
12         (if (> a b) 1
13                 (* (term a) (product-recur term (next a) next b))
14         )
15 ))
16
17 (define factorial (lambda (n)
18         (product-iter
19                 (lambda (n) n)
```

```
20                  1
21                  (lambda (n) (+ n 1))
22                  n
23          )
24  ))
25
26  (define pi (lambda (n)
27          (define term (lambda (n) n))
28          (define next (lambda (n) (+ n 2)))
29          (/ (* 4
30                      (product-iter term 2 next (* 2 n))
31                      (product-iter term 4 next (* 2 (+ n 1)))
32                  )
33                  (* (product-iter term 3 next (+ (* 2 n) 1))
34                      (product-iter term 3 next (+ (* 2 n) 1))
35                  )
36          )
37  ))
```

a. 出力結果
(pi 80) => 3.1513038442382775

JAKLDではn=80までしか出力できなかったため、以下はgaucheで実行した。

(display (exact->inexact (pi 1000))) => 3.142377365093878
(display (exact->inexact (pi 10000))) => 3.1416711865344635

nが増えるごとにπに近づいていることがわかる。

b. 再帰型、反復型の級数の積
再帰型はproduct-recur、反復型はproduct-iterで定義してある。