

Returning procedures as values

```
(define (sqrt x)
  (fixed-point
   (lambda (y)
     (average y (/ x y)))
   1.0))
```

We used (lambda (y) (average y (/ x y) y)) instead of (lambda (y) (/ x y))

Average damping

```
(define (average-damp f)
  (lambda (x) (average x (f x))))
(define (sqrt x)
  (fixed-point
   (average-damp
    (lambda (y) (/ x y)))
   1.0))
```

Cube roots

```
(define (cube-root x)
  (fixed-point
   (average-damp
    (lambda (y)
      (/ x (* y y))))
   1.0))
```

A limitation

- Average damping doesn't work for fourth-root, fifth-root, etc.
- Weighted damping:

```
(define (damped f)
  (lambda (x)
    (+ (* (- 1 weight) x)
       (* weight (f x)))))
```

Fifth root

```
(define (5th-root x)
  (fixed-point
    (damped
      (lambda (y)
        (/ x (* y y y y))))
    1.0))
(define weight 0.1)
```

Newton's method for finding roots

- If x is a guess for the root, the next guess is

$$x - \frac{f(x)}{\frac{df(x)}{dx}}$$

Newton's method code

```
(define (derivative f)
  (let ((halfdx (/ dx 2)))
    (lambda (x)
      (/ (- (f (+ x halfdx))
            (f (- x halfdx))))
         dx))))
(define dx 0.0001)
```

(code continued)

```
(define (newton-transform f)
  (lambda (x)
    (- x (/ (f x)
            ((derivative f) x)))))
(define (newtons-method f guess)
  (fixed-point
    (newton-transform f)
    guess))
```

Let's try it!

What is root of $x^2 + 4x + 1$?

```
(newtons-method  
 (lambda (x)  
   (+ (* x x) (* 4 x) 1)))  
 1.0)  
  
-0.267949...
```

Reminder!

In Scheme, procedures are first-class objects

- They can be named by variables
- They can be inputs to procedures
- They can be outputs from procedures
- They can be parts of data structures