Newton-Raphson

- General approximation algorithm to find roots of a polynomial in one variable

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

- Want to find r such that $p(r) = 0$
- For example, to find the square root of 24, find the root of $p(x) = x^2 - 24$
- Newton showed that if g is an approximation to the root, then

$$g - p(g)/p'(g)$$

is a better approximation; where p' is derivative of p

Newton-Raphson

- Simple case: $cx^2 + k$
- First derivative: $2cx$

Square root

- So if polynomial is $x^2 + k$, then derivative is $2x$
- Newton-Raphson says given a guess g for root, a better guess is

$$g - (g^2 - k)/2g$$

Newton-Raphson

- This gives us another way of generating guesses, which we can check; very efficient

```
epsilon = 0.01
y = 24.0
guess = y/2.0

while abs(guess*guess - y) >= epsilon:
    guess = guess - (((guess**2) - y)/(2*guess))
print('Square root of ' + str(y) + ' is about '
  + str(guess))
```

Close enough

Generation

Iterative algorithms

- Guess and check methods build on reusing same code
  - Use a looping construct to generate guesses, then check and continue
- Generating guesses
  - Exhaustive enumeration
  - Bisection search
  - Newton-Raphson (for root finding)