MAT 305: Mathematical Computing

Repeating a task with loops

John Perry

University of Southern Mississippi

Spring 2016

Outline

Outline

1. Loops
2. Indefinite loops
3. Summary

Loops?

- **loop:** a sequence of statements that is repeated

big time bug: *infinite loops*

Loops?

- **loop:** a sequence of statements that is repeated

big time bug: *infinite loops*

“infinite loop”?

see infinite loop

— *AmigaDOS* manual, ca. 1993

Why loops?

- like functions: avoid retyping code
  - many patterns repeated
  - same behavior, different data

- don't know number of repetitions when programming

Types of loops

- definite
    - number of repetitions known at beginning of loop
- indefinite
    - number of repetitions not known at beginning of loop
    - number of repetitions unknowable at beginning of loop

Types of loops

- definite
  - number of repetitions known at beginning of loop
- indefinite
  - number of repetitions not known at beginning of loop
  - number of repetitions unknowable at beginning of loop

Most languages use different constructions for each

Outline

The `while` command

`while` *condition* `:`

  statement1

  statement2

  ...

where

- statements are executed while *condition* remains true
  - statements will *not* be executed if *condition* is false from the get-go
- like definite loops, variables in *condition* can be modified
- unlike definite loops, variables in *condition* **should** be modified

Pseudocode for indefinite loop

while *condition*
    statement1
    statement2
    ...
out-of-loop statement 1

Pseudocode for indefinite loop

while *condition*
    statement1
    statement2
    ...
out-of-loop statement 1

Notice:

- indentation ends at end of loop
- no colon

Example

```
sage: f = x**10
sage: while f != 0:
        f = diff(f)
        print f
10*x^9
90*x^8
720*x^7
5040*x^6
30240*x^5
151200*x^4
604800*x^3
1814400*x^2
3628800*x
3628800
0
```

More interesting example

Use the Method of Bisection to approximate a root of $\cos x - x$ on the interval $[0, 1]$, correct to the hundredths place.

More interesting example

Use the Method of Bisection to approximate a root of $\cos x - x$ on the interval $[0, 1]$, correct to the hundredths place.

Hunh?!?

Method of Bisection?

The Method of Bisection is based on:

Theorem (Intermediate Value Theorem)

If

- f *is a continuous function on* $[a,b]$, *and*
- $f(a) \neq f(b)$,

then

- *for any* y *between* $f(a)$ *and* $f(b)$,
- $\exists c \in (a,b)$ *such that* $f(c) = y$.

Continuous?

f *continuous* at $x = a$ if

- can evaluate limit at $x = a$ by computing $f(a)$, or
- can draw graph without lifting pencil

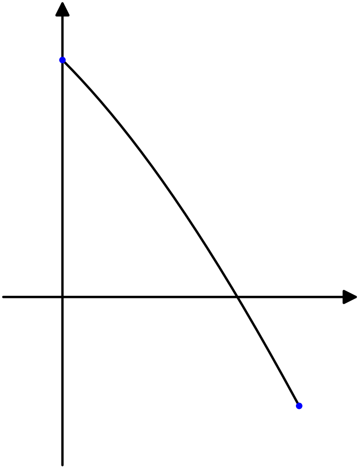
Continuous?

f *continuous* at $x = a$ if

- can evaluate limit at $x = a$ by computing $f(a)$, or
- can draw graph without lifting pencil

Upshot: To find a root of a continuous function f, start with two x values a and b such that $f(a)$ and $f(b)$ have different signs, then bisect the interval.

1 Animation = 1000 Words



(need Acrobat Reader to see animation)

Back to the example...

Check hypotheses...

- $f(x) = \cos x - x$
  - x, $\cos x$ continuous
  - difference of continuous functions also continuous

  $\therefore$ f continuous
- $a = 0$ and $b = 1$
  - $f(a) = 1 > 0$
  - $f(b) \approx -0.4597 < 0$

Intermediate Value Theorem applies: can start Method of Bisection.

How to solve it?

Idea: Interval endpoints a and b are not close enough as long as their digits differ through the hundredths place.

How to solve it?

Idea: Interval endpoints a and b are not close enough as long as their digits differ through the hundredths place.

Application: While their digits differ through the hundredths place, halve the interval.

How to solve it?

Idea: Interval endpoints a and b are not close enough as long as their digits differ through the hundredths place.

Application: While their digits differ through the hundredths place, halve the interval.

"Halve" the interval? Pick the half containing a root!

Pseudocode

algorithm *method_of_bisection*

Pseudocode

algorithm *method_of_bisection*

inputs

f, a continuous function

$a, b \in \mathbb{R}$ such that $a \neq b$ **and** $f(a)$ and $f(b)$ have different signs

Pseudocode

algorithm *method_of_bisection*

inputs

f, a continuous function

$a, b \in \mathbb{R}$ such that $a \neq b$ **and** $f(a)$ and $f(b)$ have different signs

outputs

$c \in [a, b]$ such that $f(c) \approx 0$ **and** c accurate to hundredths place

Pseudocode

algorithm *method_of_bisection*
inputs
f, a continuous function
$a, b \in \mathbb{R}$ such that $a \neq b$ **and** $f(a)$ and $f(b)$ have different signs
outputs
$c \in [a, b]$ such that $f(c) \approx 0$ **and** c accurate to hundredths place
do
while the digits of a and b differ through the hundredths
Let $c = \frac{a+b}{2}$
if $f(a)$ **and** $f(c)$ have the same sign
Let $a = c$ — Interval now $\left(\frac{a+b}{2}, b\right)$
else if $f(a)$ and $f(c)$ have opposite signs
Let $b = c$ — Interval now $\left(a, \frac{a+b}{2}\right)$
else — we must have $f(c) = 0$
return c
return a, rounded to hundredths place

Try it!

```
sage:  def method_of_bisection(f,a,b,x=x):
          while round(a,2) != round(b,2):
```

Try it!

```
sage:  def method_of_bisection(f,a,b,x=x):
         while round(a,2) != round(b,2):
           c = (a + b)/2
```

Try it!

```
sage:  def method_of_bisection(f,a,b,x=x):
         while round(a,2) != round(b,2):
           c = (a + b)/2
           if f({x:a})*f({x:c}) > 0:
             a = c
           elif f({x:a})*f({x:c}) < 0:
             b = c
           else:
             return c
         return round(a,2)
```

Try it!

```
sage: def method_of_bisection(f,a,b,x=x):
        while round(a,2) != round(b,2):
          c = (a + b)/2
          if f({x:a})*f({x:c}) > 0:
            a = c
          elif f({x:a})*f({x:c}) < 0:
            b = c
          else:
            return c
        return round(a,2)

sage: method_of_bisection(cos(x)-x,x,0,1)
0.74
```

Outline

1. Loops
2. Indefinite loops
3. Summary

Summary

Two types of loops

- definite: n repetitions known at outset
  - **for** $c \in C$
    - collection C of n elements controls loop
    - don't modify C
- indefinite: number of repetitions not known at outset
  - **while** *condition*
    - Boolean *condition* controls loop