

Iteration and Recursion

Computational Physics

Iteration and Recursion

Outline

- Definitions
 - Iteration
 - Recursion
- Examples
 - Fibonacci Sequence
 - Computing Sums
 - Searching for a Root

Definitions

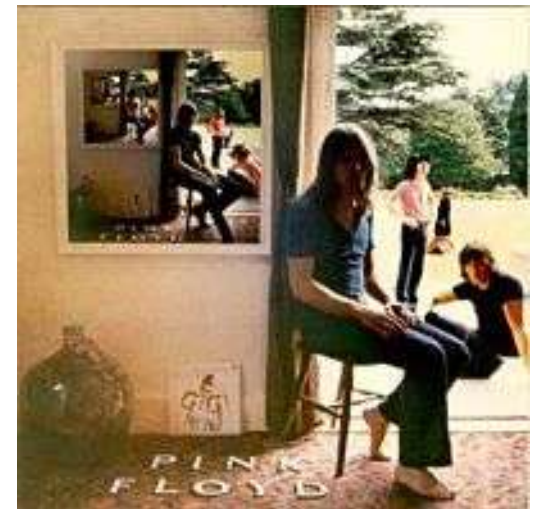
Many numerical algorithms involve repeating simple operations again and again to achieve a solution.

Iteration

- Definition of function or sequence through repetition of a process
- Solution technique involving successive approximations

Recursion

- Special case of Iteration wherein function or sequence is defined by calling itself.



Fibonacci Sequence

- A famous number sequence derived from a simple formula:

$$F_n = F_{n-1} + F_{n-2}$$

with seed values: $F_1 = 0$; $F_2 = 1$

- Sequence: 0 1 1 2 3 5 8 13 21 34 55 ...
- Features
 - Natural Phenomena (Petals on flowers)
 - F_n/F_{n-1} approaches Golden Ratio as $n \rightarrow \infty$

Fibonacci Sequence

Iterative Approach

```
% compute fibonacci sequence

% initialize first two terms
f(1) = 0;
f(2) = 1;

% start computing with next term
for i=3:10 %compute to 10th term
    f(i) = f(i-1)+f(i-2);
end

% show result
f
```

New value in series
is sum of previous
two values



Fibonacci Sequence

Recursive Approach

```
function result = fib(m)

% recursive function to compute
% fibonacci sequence

if(m==0)
    result = 0;
elseif(m==1)
    result = 1;
else
    result = fib(m-1) + fib(m-2);
end

return result;
```

Creates a MATLAB function (in file *fib.m*) to compute the *m*th term in the Fibonacci sequence.

Function calls itself to add two previous values in the sequence

Computing a Sum

```
% compute sum of numbers
% 1 through 10

% initialize variable
Sum = 0.0;

% now add in each element
for i=1:10
    Sum = Sum + i;
end

% print result
fprintf("Result = %f\n", Sum);
```

$$\sum_{i=1}^{10} i$$

The variable "Sum" is initialized with value of 0.0

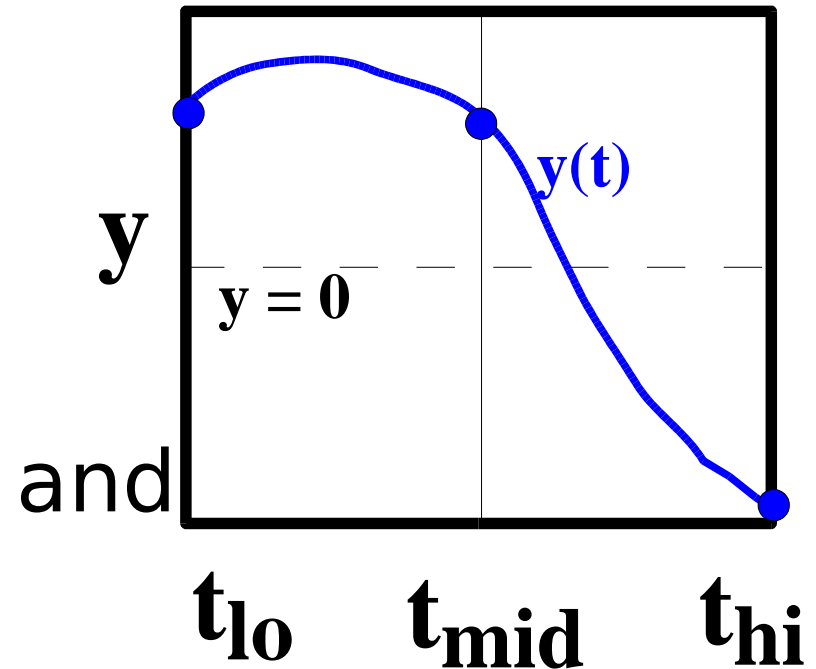
First pass through loop:
 $i = 1$ so $\text{Sum} = 0 + 1 = 1$

Next pass through loop:
 $i = 2$ so $\text{Sum} = 1 + 2 = 3$

Third pass through loop:
 $i = 3$ so $\text{Sum} = 3 + 3 = 6$

Iterative Search

- Find value of t where $y(t)=0$
- if $y(t_{lo})$ and $y(t_{hi})$ have opposite signs then a root exists.
- Compute midpoint
 $t_{mid} = (t_{hi}+t_{lo})/2$
determine which half of domain contains the root
- Repeat until search domain has narrowed to give an answer that is “close enough”



Bisection Algorithm

```
% initial range is lo, hi
% initial values at end points are ylo, yhi
% note  $ylo \cdot yhi < 0$  assures a root exists

epsilon = 1e-10; %convergence criteria

while(abs(hi-lo)>2*epsilon)
    mid = (hi+lo)/2;
    ymid = myf(mid);
    if(ymid*ylo>0)
        lo = mid;
    else
        hi = mid;
    end
end
end
```

myf is MATLAB function which computes function for root search

true if *ymid* is on same side of 0 as *ylo*. In this case new low end of search is *midpt*.