

Goals for This Lecture:

- Understand how to name loops
- Understand the use of EXIT statements in iterative DO loops and in nested loops
- Understand nesting loops within conditional constructs and vice versa
- Understand the CYCLE statement
- Using loops to perform numerical integration of a function

Naming DO loops

- It is possible to name loops in a similar fashion to naming block IF constructs

Form:

```
name: do ivar = istart,iend,incr  
    statement block1  
enddo name
```

Form:

```
name: do while(logical_expression)  
    statement block1  
enddo name
```

The EXIT statement

- It is possible to use the exit statement in iterative DO loops as well as DO WHILE and WHILE loops
- EXIT statements can also be used in nested loops.
- EXIT transfers control to the first executable statement following the innermost loop in which it is called
- The EXIT statement can be made to transfer control to an executable line following a loop other than the innermost loop by referring to the name of the loop

Example:

```
outer: do i=1,10,1
  inner: do j=1,10,1
    if(i==j) exit outer
    statement block1
  enddo inner
enddo outer
```

Nesting loops within conditional constructs and vice versa

legal:

```
do i=1,10,1
  if(i==j) then
    write(*,*) `i equal j`
  endif
enddo
```

illegal:

```
if(i==j) then
  do i=1,10,1
    write(*,*) `i equal j`
  endif
enddo
```

legal:

```
if(iflag==1) then
  do i=1,10,1
    write(*,*) `i equal j`
  enddo
endif
```

illegal:

```
do i=1,10,1
  if(iflag==1) then
    write(*,*) `i equal j`
  enddo
endif
```

The CYCLE statement

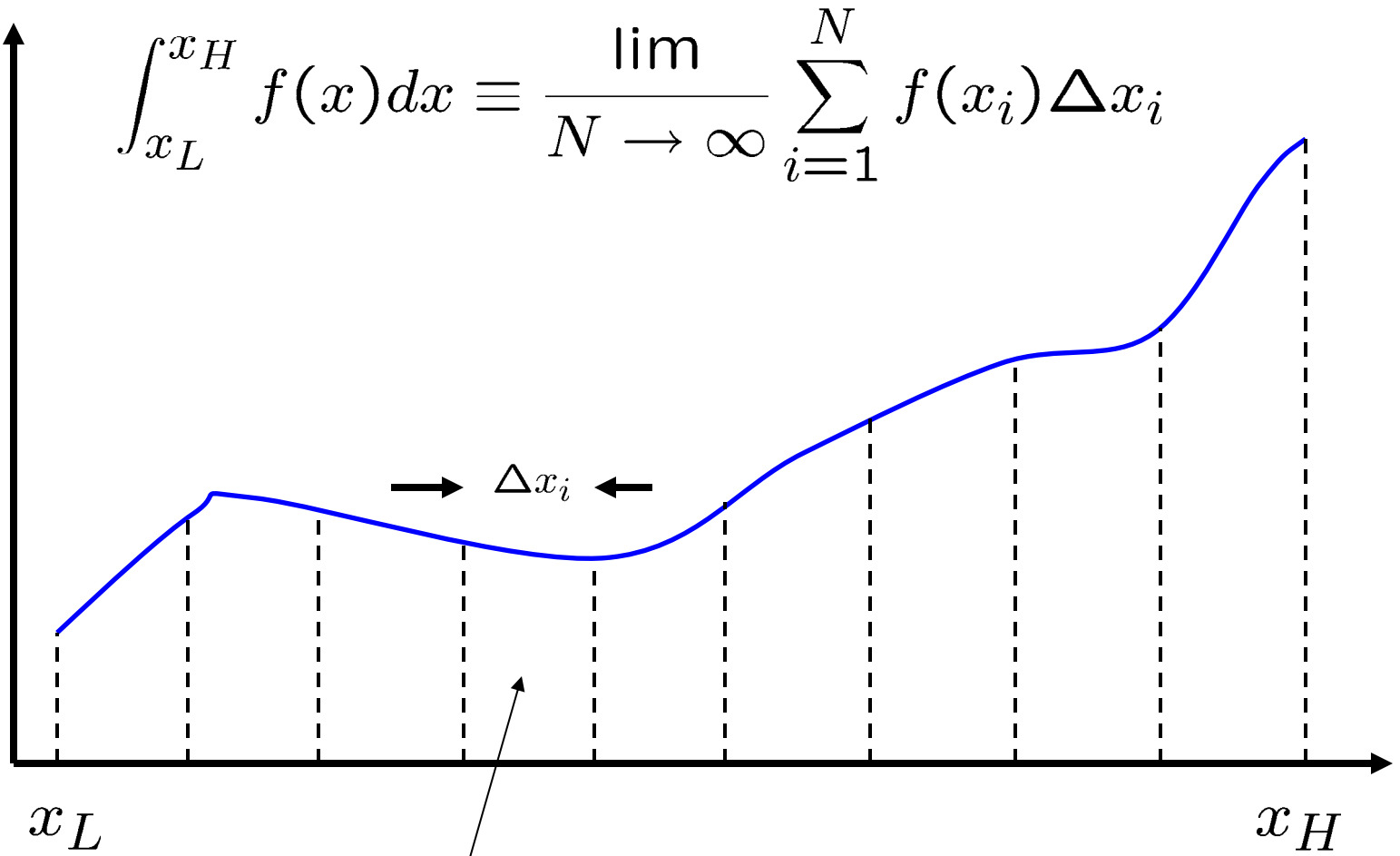
- The CYCLE statement causes the execution of the body of the loop to immediately transfer back to the top of the loop
- If the CYCLE refers to the to a name of a named loop the control transfers to the top of the loop

Example:

```
do i=1,10,1
  if(i==3) cycle
  write(*,*) ' i = ',i
enddo
```

Using DO loops to perform numerical integration

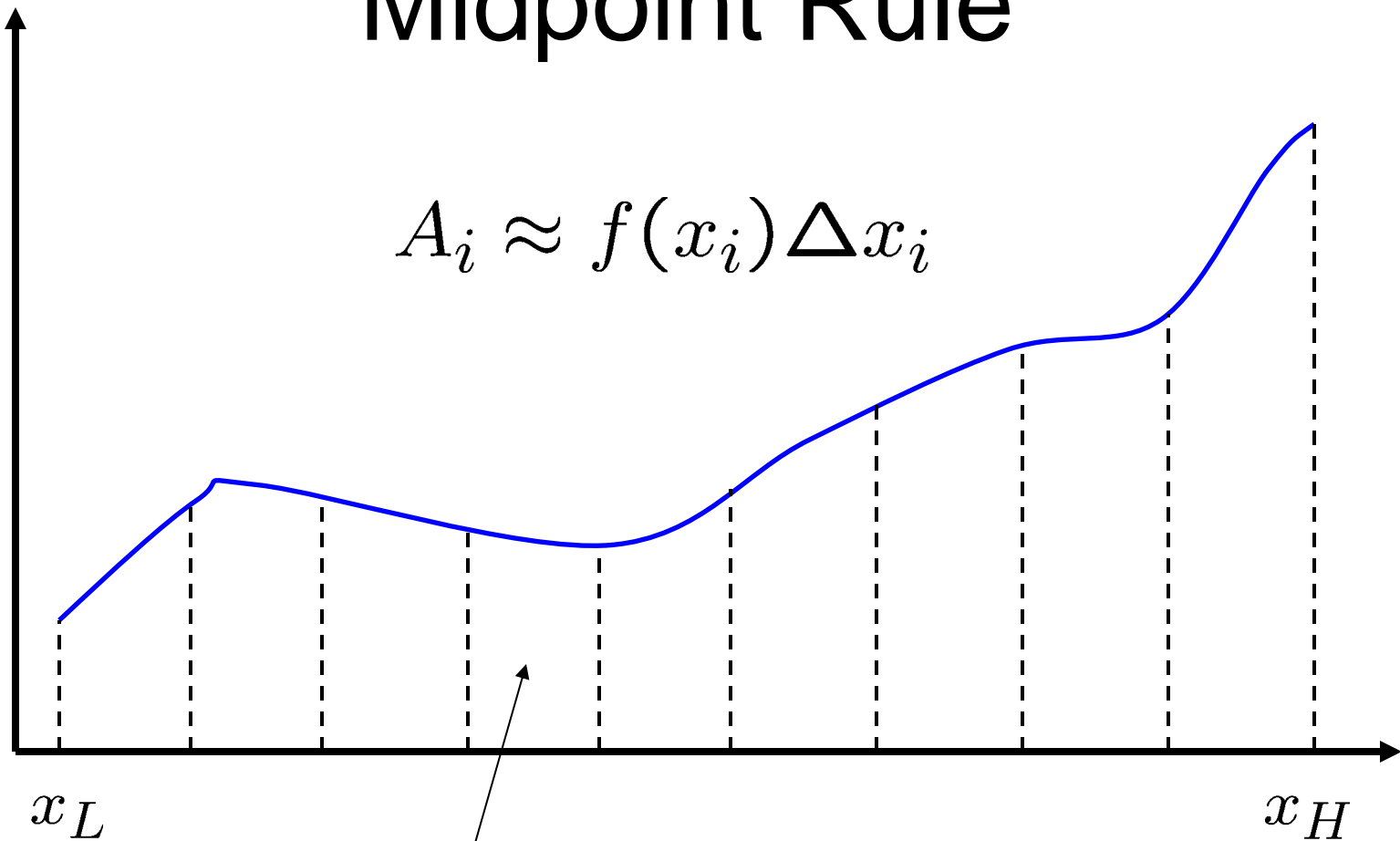
- We can now discuss a simple way to numerically find the definite integral of a function
 -
- Remember: what we are trying to do is find the area under the curve of a function between points a and b
- We can break up the interval into sub-intervals in order to try to find the area of each
- This approximation should get better as we break the interval between a and b up into more and more sub-intervals
- In order to estimate the area of the sub-intervals we can evaluate the function at the midpoint of each sub-interval and treat the area of the sub-interval as a rectangle



A_i = Area of i th subinterval

$$\int_{x_L}^{x_H} f(x) dx \approx \sum_{i=1}^N A_i$$

Midpoint Rule



$A_i =$ Area of i th subinterval

Midpoint Rule Program

```
! Purpose: Integrate x^2 from 1.0 to 2.0 via midpoint rule
! Author:  F. Douglas Swesty
! Date:    9/28/2005
program midpoint
implicit none      ! Turn off implicit typing
Integer, parameter :: n=100 ! Number of subintervals
integer :: i       ! Loop index
real :: xlow=1.0, xhi=2.0 ! Bounds of integral
real :: dx        ! Variable to hold width of subinterval
real :: sum       ! Variable to hold sum
real :: xi        ! Variable to hold location of ith subinterval
real :: fi        ! Variable to value of function at ith subinterval

dx = (xhi-xlow)/(1.0*n) ! Calculate width of subinterval

sum = 0.0                ! Initialize sum
xi = xlow+0.5*dx         ! Initialize value of xi
do i = 1,n,1             ! Initiate loop
  fi = xi**2             ! Evaluate function at ith point
  sum = sum+fi*dx        ! Accumulate sum
  xi = xi+dx             ! Increment location of ith point
enddo                    ! Terminate loop

write(*,*) ' sum = ',sum
stop                     ! Stop execution of the program
end program midpoint
```

Reading Assignment

- Read Sections 4.1,4.3