# Algorithm for Integration Using the Left Rectangle Rule

We want to calculate using rectangles, with the height of the rectangle given by the left edge, to some desired precision.

**Straightforward (simple but somewhat inefficient) version of the Left Rectangle Rule**

Given *left* (or *a* in the notation above), *right* (or *b* in the notation above), *f*(*x*), *precision, maxLoops*

// **Start with an initial approximation** - in this case, 1 rectangle

*numRectangles* 🡨 1

*totalWidth* 🡨 *right* - *left*

*height* 🡨 *f*(*left*)

*estimate* 🡨 *height* \* *totalWidth*

// setup for loop:

*numLoops* 🡨 0

*keepGoing* 🡨 **true**

// **While we are not "close enough"**

While (*keepGoing*)

Add 1 to *numLoops*

*oldEstimate* 🡨 *estimate*

// **Find a better approximation** - use twice as many rectangles

Multiply *numRectangles* by 2

*currentWidth* 🡨 *totalWidth* / *numRectangles*

*estimate* 🡨 0

for each rectangle *i* starting at *i* = 0

// calculate area of current rectangle and add to total

*currentLeft* 🡨 *left* + *i* \* *currentWidth*

*currentHeight* 🡨 *f*(*currentLeft*)

*currentArea* 🡨 *currentHeight* \* *currentWidth*

Add *currentArea* to *estimate*

// **Check if we are "close enough"**

*error* 🡨 absolute value of *estimate* - *oldEstimate*

*relError* 🡨 absolute value of *error* / *estimate*

if *relError* ≤ *precision*

*keepGoing* 🡨 **false**

else if *numLoops* ≥ *maxLoops*

indicate error

*keepGoing* 🡨 **false**

return *estimate*

**Implementation issues:**

* Make sure all of your files are in the same project!
* Make sure that your loop is doing the correct number of rectangles.
  + Since we start at *i* = 0, the correct code should be something like   
    for (i = 0; i < numRectangles; i++)  
    Make sure that you have the right initialization and condition!
* Make sure that the order of operations is correct in calculating error.
* If implementing as a method in the IFunction / ACFunction classes:
  + Don't forget to put the method definition in your interface (IFunction)!
  + If the integration rule is a method in ACFunction, you don't need to pass the function in as a parameter – it is the invoking object.
    - To calculate *f*(*x*), we will call this.calculate(x) – for instance, *f*(*left*) would be something like this.calculate(left), depending on your variable names.
* If implementing as a class of its own (such as Integral):
  + Consider what attributes you need and what values you want to pass into the constructor or constructors
  + Consider what results you want to use – might include:
    - Estimate
    - Boolean indication of convergence
    - Number of loops (or number of rectangles) used

**Possible ways to increase efficiency:**

*Note that answers should be the same (with slight differences based on representation/propagation/round-off error) with efficiencies implemented (just calculated faster)!*

* Take common steps (like multiplication of the width) out of the loop (may decrease clarity, but increases efficiency):

for each rectangle *i* starting at *i* = 0

*currentLeft* 🡨 *left* + *i* \* *currentWidth*

*currentHeight* 🡨 *f*(*currentLeft*)

*currentArea* 🡨 *currentHeight* ~~\*~~ *~~currentWidth~~*

Add *currentArea* to *estimate*

Multiply *estimate* by *currentWidth*

* Combine steps in calculations in loop:

for each rectangle *i* starting at *i* = 0

Add *f*(*left* + *i* \* *currentWidth*) to *estimate*

Multiply *estimate* by *currentWidth*

* Store previously calculated heights so we don't have to call the function to calculate them again – calculate heights only for the odd-numbered rectangles (new heights)
  + Could use previous estimate as a starting point – must multiply by 0.5 to account for the smaller width; OR
  + Keep track of all heights calculated separately and then multiply by the width:

// in the initial estimate

*heights* 🡨 *f*(*left*)

*estimate* 🡨 *heights* \* *totalWidth*

...

// *estimate* 🡨 0 - remove this line, since we don't need to reset any more

...

// Change the for loop so it only looks at the new odd-numbered rectangles

For every new rectangle starting at *i* = 1 up to *numRectangles* by 2

Add *f*(*left* + *i* \* *currentWidth*) to *heights*

*estimate* 🡨 *heights* \* *currentWidth*

**Efficient version of the Left Rectangle Rule**

Given *left* (or *a* in the notation above), *right* (or *b* in the notation above), *f*(*x*), *precision, maxLoops*

// Start with an initial approximation - in this case, 1 rectangle

*numRectangles* 🡨 1

*totalWidth* 🡨 *right* - *left*

*heights* 🡨 *f*(*left*)

*estimate* 🡨 *heights* \* *totalWidth*

// setup for loop:

*numLoops* 🡨 0

*keepGoing* 🡨 **true**

// While we are not "close enough"

While (*keepGoing*)

Add 1 to *numLoops*

*oldEstimate* 🡨 *estimate*

// Find a better approximation - use twice as many rectangles

Multiply *numRectangles* by 2

*currentWidth* 🡨 *totalWidth* / *numRectangles*

For every new rectangle starting at *i* = 1 up to *numRectangles* by 2

Add *f*(*left* + *i* \* *currentWidth*) to *heights*

*estimate* 🡨 *heights* \* *currentWidth*

// Check if we are "close enough"

*relError* 🡨 absolute value of ((*estimate* - *oldEstimate*) / *estimate)*

if *relError* ≤ *precision*

*keepGoing* 🡨 **false**

else if *numLoops* ≥ *maxLoops*

indicate error

*keepGoing* 🡨 **false**

return *estimate*