4.1

The code for evaluating composite trapezoid

```
def eval_composite_trap(M,a,b,f):
  """
  put code from prelab with same returns as gauss_quad
  you can return None for the weights
  """
  h = (b-a)/M
  x = np.linspace(a,b,M+1)
  f_i = f(x)

  I_trap = (h/2)*(f_i[0]+2*sum(f_i[1:M])+f_i[M])

  return I_trap
```

The code for evaluating composite simpsons

```
def eval_composite_simpsons(M,a,b,f):
  """
  put code from prelab with same returns as gauss_quad
  you can return None for the weights
  """
  h = (b-a)/M
  x = np.linspace(a,b,M+1)
  f_i = f(x)

  I_simp = (h/3) *(f_i[0]+2*sum(f_i[:n-1:2])+4*sum(f_i[1:n:2])+f_i[n-1])

  return I_simp
```

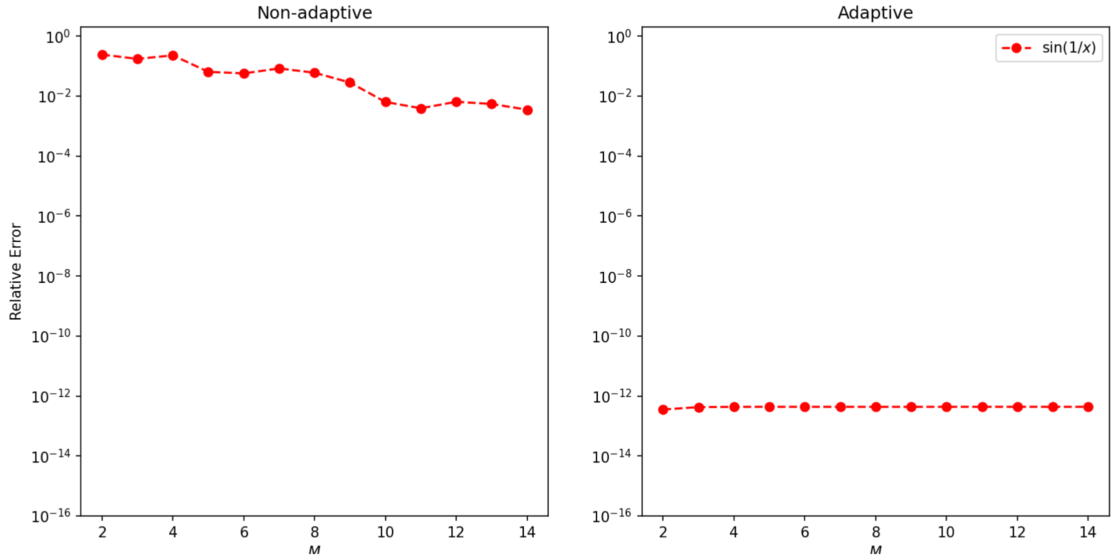
Gaussian quadrature

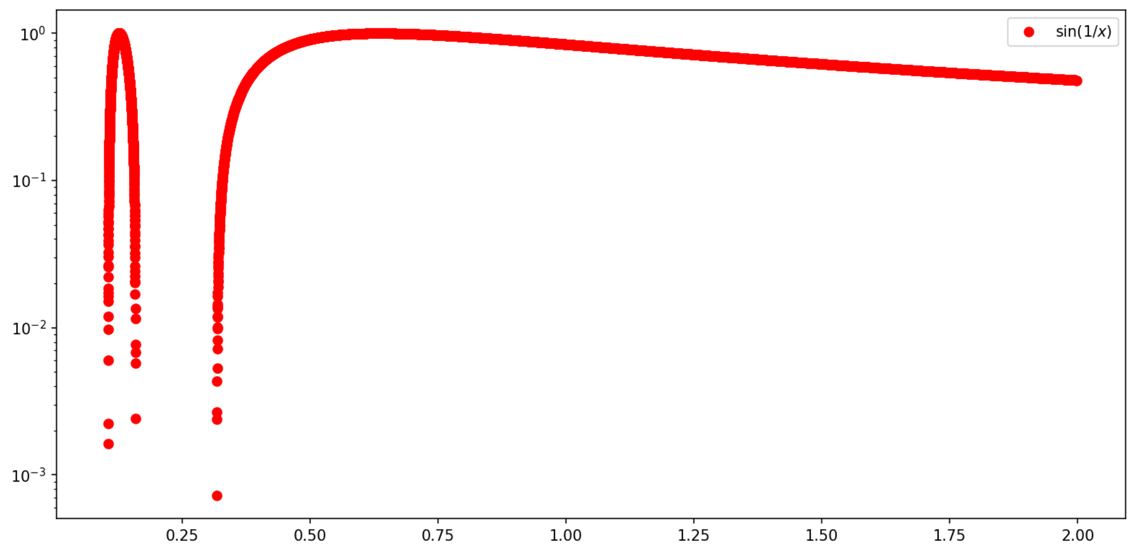
```
def eval_gauss_quad(M,a,b,f):

  x,w = lgwt(M,a,b)
  I_hat = np.sum(f(x)*w)
  return I_hat,x,w
```

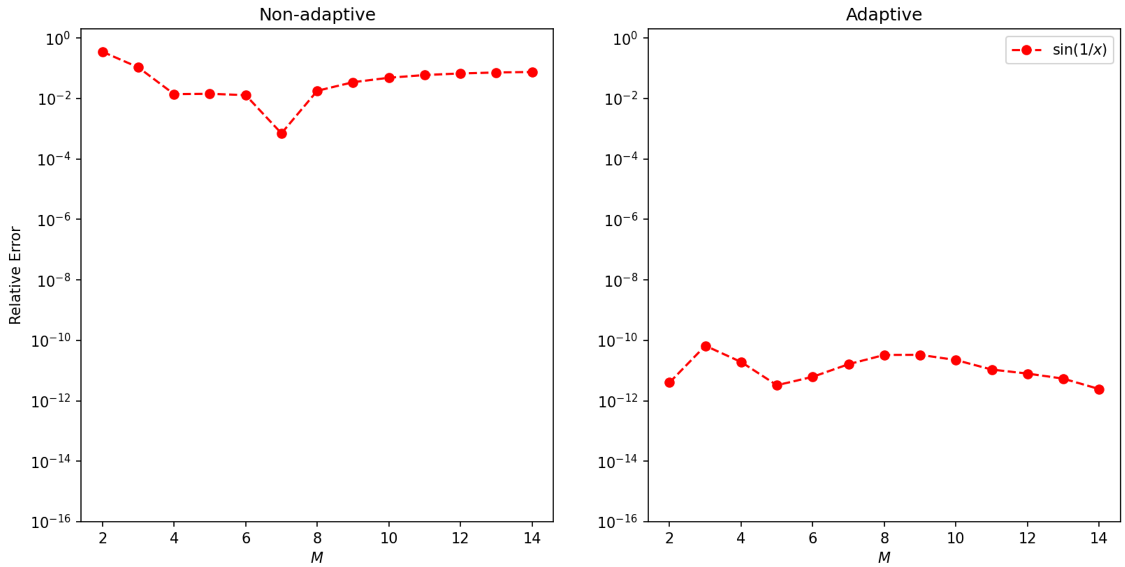
4.2

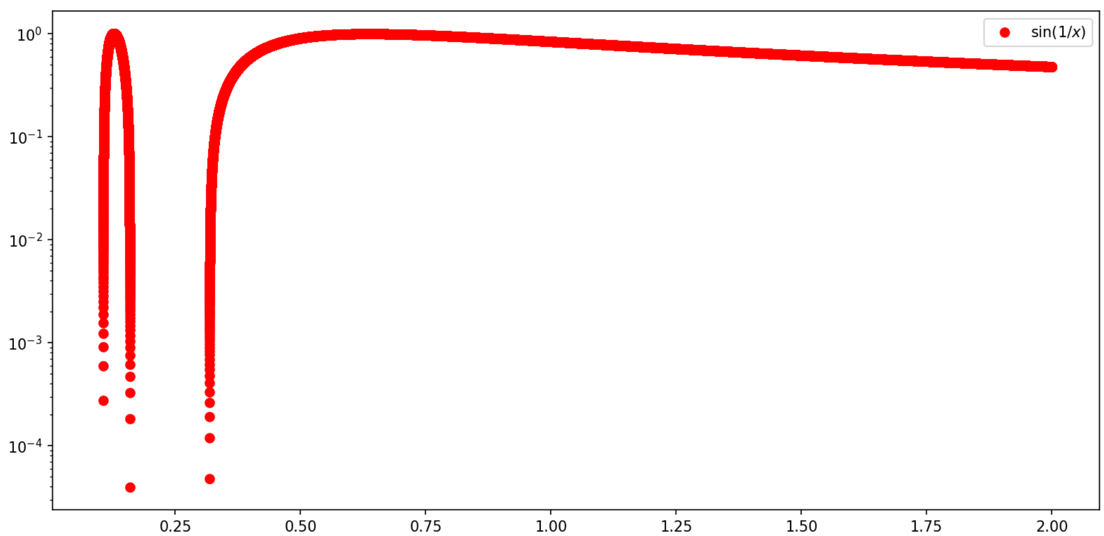
We get the following plot when evaluating using gaussian quadrature



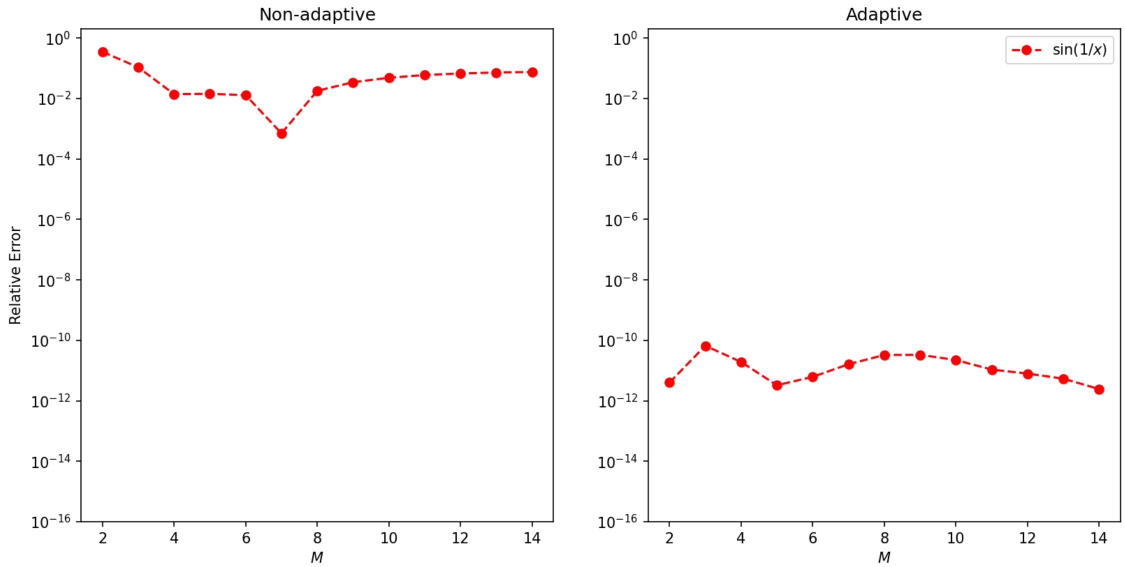


When we use composite trapezoid we get the following graphs





Using composite simpsons we get the following graphs


Non-adaptive
Adaptive
sin(1/x)
Relative Error
M
M