```
n = 500; %Initial n value as a starting point
a = -4;%other initial conditions and formula
b = -1;
f = @(x) (2+ x^3)* sin(exp(x));
tol = 10^(-6);
err = 1;%a start point for err

while err > tol%checking if the error is below my tolerance value and if cycling again

    n = n+1;%went for n + 1 as this is a pretty high tolerance. for smaller tolerances the ju
mp in n as well as the start point of n can be increased to reduce computing time
    h = (b-a)/n; %calculating h using given formula
    xi = a:h:b; %creating a vector of evenly spaced points from x0 to xn
    fi = zeros(1,n);% resetting fi as n2 is significantly larger than n so summing would give
 error. initializing as a 1 by n size vector for computing time reduction rather than resizin
g every ittertation of for loop
    for i = 1:n%calculaing each value of fi
        fi(1,i) = h*f((xi(i)+xi(i+1))/2); %using the formula to work out each midpoint approx
imation and saving to vector fi
    end
    Int1 = abs(sum(fi)); %summing fi and taking its absolute value to give an integral approx
imation

    n2 = 3*n;%calculating n2
    h = (b-a)/n2; %calculating h using given formula
    xi = a:h:b; %creating a vector of evenly spaced points from x0 to xn
    fi = zeros(1,n2);%resizing vector once rather than doing so every for loop
    for i = 1:n2%calculting each value of fi
        fi(1,i) = h*f((xi(i)+xi(i+1))/2); %using the formula to work out each midpoint approx
imation and saving to vector fi
    end
    Int2 = abs(sum(fi)); %summing fi and taking its absolute value to give an integral approx
imation

    err = Int1 - Int2;%calculating err

end
fprintf('The Integral is %4.6f and took%4i intervals \n', [Int1,n]) %printing the integral to
 console
```

```
The Integral is 2.583692 and took 748 intervals
```

Published with MATLAB® R2019b