```
[E_adaptive, n_adaptive] = adaptive();
[E_comp, n_comp] = composite();

fprintf("composite n=%d  error=%e  adaptive n=%d error=%e\n", n_comp, E_comp,↙
n_adaptive, E_adaptive)
a = 0.1;
b = 1;

global xpoints;
f = @(x) sin(1./x);

fplot(f, [a, b], "r")
hold on
plot(xpoints, 0, "ob")
title("sin(1/x) actual vs adaptive simpson")



function [error, n] = adaptive()
    a = 0.1;
    b = 1;
    global xpoints;
    xpoints = [a, b];

    f = @(x) sin(1./x);
    F = integral(f, a, b);
    %f =@(x) sin(1/x);

    [r, level] = ad_simpson( f, a, b, 10^(-4), 0, 1000, 0 );

    n = 3+2*level;

    error = abs(r-F);
end

function [error, n] = composite()
    n = 2;
    a = 0.1;
    b = 1;

    %E = max|-f''''(thing)|*(b-a)^5/n

    f = @(x) sin(1./x);
    F = integral(f, a, b);

    while 1
        h = (b-a)/n;

        even = 0;
```

```
        for i = 1:(n/2-1)
            even = even + f(a+2*i*h);
        end

        odd = 0;
        for i = 1:(n/2)
            odd = odd + f(a+(2*i-1)*h);
        end

        comp = (h/3)*(f(a) + 2*even + 4*odd + f(b));

        error = abs(F-comp);


        if error < 10^(-4)
            break
        end
        n = n*2;
    end
end



function [r, level, funs ] = ad_simpson( f, a, b, eps, level, level_max, funs )
     global xpoints
     level = level+1;
     h = b-a;
     c = (a+b)/2;

     ab = feval(f,a) + feval(f,b);
     cc = feval(f, c);

     one = h*( ab + 4*cc  )/6;

     d = (a+c)/2;
     e = (c+b)/2;

     xpoints = unique([xpoints, a, b, c, d, e]);
     two = h*( ab + 4*feval(f,d) + 2*cc + 4*feval(f,e)  )/12;


     funs = funs+5;

     if level >= level_max
           r = two;
           disp('max level reached');
     else

         if abs(two-one) < 15*eps
           r = two + (two-one)/15;
```

```
        else
            [left,  level, funs] = ad_simpson( f, a, c, eps/2, level, level_max, funs↙
);
            [right, level, funs] = ad_simpson( f, c, b, eps/2, level, level_max, funs↙
);
            r = left + right;
        end
    end
end
```