Using Runge-Kutta 2nd (Heun's Method) order to bootstrap for both two-stage Adams-Bashforth (AB2) and two-stage Adams-Moulton(AM2) methods

```
%rk2 solver
function y_next = rk2_step(f,t,y,h)
    k1=f(t,y);
    k2=f(t+h,y+h*k1);
    y_next = y+(h/2)*(k1+k2);
end

function ab2_am2(h,t0,tf)
    f = @(t,y) sin(2*t)+y-y^4;
    dfdy= @(t,y) 1-4*y^3;
    %Parameter
    t=t0:h:tf;
    y_exact =0.9426297327;
    N=(tf-t0)/h;

    y_ab2= zeros(1,N+1);
    y_am2= zeros(1,N+1);

    y_ab2(1)=0;
    y_am2(1)=0;

    y_ab2(2)=rk2_step(f,t(1),y_ab2(1),h);
    y_am2(2)=rk2_step(f,t(1),y_am2(1),h);

    %AB2 method
    for n=2:N
        y_ab2(n+1)=y_ab2(n)+0.5*h*(3*f(t(n),y_ab2(n)) -
f(t(n-1),y_ab2(n-1)));
    end

    %AM2 method (fully-implicit) --> using Newton-Raphson method
    tol=1e-10;
    max_i=20;

    for n=2:N
       y_guess = rk2_step(f,t(n),y_am2(n),h);
       for i=1:max_i
            F = y_guess - y_am2(n) - (1/12)* h * (5*f(t(n+1), y_guess) +
8*f(t(n), y_am2(n)) - f(t(n-1), y_am2(n-1)));
            dF = 1 - (1/12)* h * (5 * dfdy(t(n+1), y_guess));
            delta = -F /dF;
            y_guess = y_guess + delta;

            if abs(delta) < tol
                break;
            end
       end
```

```
        y_am2(n+1) = y_guess;
    end

    %Errors
    err_ab2 = abs(y_ab2(end) - y_exact);
    err_am2 = abs(y_am2(end) - y_exact);


    fprintf('AB2 Method (RK2 Bootstrap):\n  y(1) = %.10f\n  Error =
%.10e\n\n', y_ab2(end), err_ab2);
    fprintf('AM2 Method (RK2 Bootstrap, Newton-Raphson):\n  y(1) = %.10f\n
Error = %.10e\n', y_am2(end), err_am2);
end

ab2_am2(0.1,0,1)
```

```
AB2 Method (RK2 Bootstrap):
  y(1) = 0.9589684529
  Error = 1.6338720207e-02

AM2 Method (RK2 Bootstrap, Newton-Raphson):
  y(1) = 0.9416646250
  Error = 9.6510772171e-04
```

Questions:

(a) What happens to the error if you halve the step size for both method? Why?

```
ab2_am2(0.1/2,0,1)
```

```
AB2 Method (RK2 Bootstrap):
  y(1) = 0.9468798069
  Error = 4.2500741920e-03

AM2 Method (RK2 Bootstrap, Newton-Raphson):
  y(1) = 0.9425163265
  Error = 1.1340617321e-04
```

The errors are reduced for both methods as the step size is halved. This happens because as the error reduced, the number of step increase but then each step is more accurate so the total error decrease. This show how good the method is.

(b) How does the error of AB2 compare to the error in AM2? Why?

It can be seen that the error of AM2 is less than AB2 method. This happens because as AB2 use the past result to predict the future result, the AM2 method use also the future elements to predict the result and it also has stronger stability.

(c) What method did you choose to solve the implicit equation in AM2? Why?

I choose to use Newton-Raphson method. The reason is that the equation is non-linear and using Newton-Raphson method will make it converge faster.

(d) Would you prefer a fully-implicit AM2 or a predictor-corrector method using AB2 and AM2? Why?

In the case where accuracy and robustness is matter more such as solving for stiff ODE, I would prefer to use a fully-implicit AM2 as this method give a more stable and better accuracy which will cost extra to compute but totally worth it.

In the other case where the problem is fairly simple (quick simulation or non-stiff problems) and speed is put into consideration, I would prefer a predictor-corrector method using AB2 and AM2 as this method provide a good enough result while being simplier and cheap to implement.