

Intro to Numerical Computing

Homework 13

December 15, 2024

Alex Knigge & Ryan Scherbarth
CS/MATH 375

1. Gaussian Quadrature.

Remember the integral

$$I = \int_0^{\pi} x \sin(x) dx$$

from the previous problem, where you computed the solution by hand and approximated it by the composite trapezoid method. We will now approximate it using Gaussian quadrature.

- (a) Approximate the integral by hand using the Gaussian quadrature rule with three points. The quadrature rule on the interval $[-1, 1]$ is given by

$$\int_{-1}^1 f(x) dx \approx w_0 f(x_0) + w_1 f(x_1) + w_2 f(x_2)$$

where $x_0 = -\sqrt{3/5}$, $x_1 = 0$, $x_2 = -x_0$, and $w_0 = 5/9$, $w_1 = 8/9$, $w_2 = w_0$. Note: you will need to do the change of interval as described in lecture/book in order to get an approximation over the arbitrary interval $[a, b]$.

- (b) Write a MATLAB function `integral = gauss_int(f,a,b,k)` that implements the k -point Gaussian quadrature rule on the interval $[a, b]$. The values `a` and `b` are the limits of integration and `f` is the function to be integrated. Your function only needs to work for $k \leq 8$. I have provided a MATLAB function `lgwt_table.m` on the Canvas page which you should feel free to use. It produces the Gaussian weights and nodes for the k -point quadrature rule on the interval $[-1, 1]$.

To test your code, use it with $k = 3$ and show that it gives the same answer as your by-hand calculations in the previous question.

- (c) Use your code from the previous question to approximate the integral for $k = 2, 3, \dots, 8$. For each approximation, compute the error. Generate a semi-log plot of error as a function of k . Does your plot look linear? If so, it would imply that the error is $\mathcal{O}(e^{bk})$ for some constant b (i.e. Gaussian quadrature is geometrically convergent).

- (d) Write a MATLAB function `integral = comp_gauss_int(f,a,b,k,n)` that implements composite Gaussian quadrature on the interval $[a, b]$ using n sub-intervals, and k -point quadrature on each sub-interval. Here, f is the function to be integrated.

To write this composite integral function, use `integral = gauss_int(...)` to approximate the integral on each of the (smaller) intervals $[x_{i-1}, x_i]$, for $x_0 = a, x_i = x_0 + ih, h = (b - a)/n$. Then sum up the contributions on each of the smaller intervals to approximate the entire integral over $[a, b]$, again analogous to composite trapezoid.

- (e) Compute an approximate integral using composite Gaussian quadrature with $k = 3$ -point quadrature on each sub-interval for $n = 4, 8, 16, 32$. Tabulate your results by showing three columns: the value of the approximate integral computed by composite Gaussian quadrature, the error, and the observed algebraic order of convergence p for each value of n . Recall that the order of convergence p is computed as

$$p = \frac{\log(err^{(n)}/err^{(n-1)})}{\log(h^{(n)}/h^{(n-1)})}$$

where $err^{(n)}$ is the error for n intervals and $h(n) = (b - a)/n$ is the sub-interval length. Plot the error as a function of n on a log-log plot. Does this plot confirm the order of convergence suggested by the table above?

- (f) Repeat the previous problem with $k = 4$.

Ans:

- (a) Start by mapping the given interval, $[0, \pi]$, into the Gaussian Quadrature interval $[-1, 1]$.

$$x = \frac{b-1}{2}t + \frac{b+1}{2}$$

$$dx = \frac{b-a}{2}dt$$

$$I = \int_0^\pi x \sin(x) dx = \frac{\pi}{2} \int_{-1}^1 \left(\frac{\pi}{2}t + \frac{\pi}{2}\right) \sin\left(\frac{\pi}{2}t + \frac{\pi}{2}\right) dt$$

Approximate using 3 points;

$$\approx w_0 f(t_0) + w_1 f(t_1) + w_2 f(t_2)$$

Splitting the interval, we should be calculating at $t_0 = -\sqrt{\frac{3}{5}}$, $t_1 = 0$, and $t_2 = \sqrt{\frac{3}{5}}$.

Solving the weights using the formula $w_i = \int_{-1}^1 \ell_i(x) dx$

$w_0 = \frac{5}{9}$, $w_1 = \frac{8}{9}$, $w_2 = \frac{5}{9}$.
Substitute the terms in;

$$\approx \frac{\pi}{2} \left[\frac{5}{9} (x_0 \sin(x_0)) + \frac{8}{9} \left(\frac{\pi}{2} \right) + \frac{5}{9} (x_2 \sin(x_2)) \right]$$

Where $x_0 = \frac{\pi}{2} (1 - \sqrt{\frac{3}{5}})$, and $x_2 = \frac{\pi}{2} (1 + \sqrt{\frac{3}{5}})$

(b) Script

```
function integral = gauss_int(f, a, b, k)
    [n, w] = lgwt_table(k);

    new_nodes = (b - a) / 2 * n + (b + a) / 2;
    new_weights = (b - a) / 2 * w;

    function_values = arrayfun(f, new_nodes);

    integral = sum(new_weights .* function_values);
end
```

(c)

n	Approx	Err	Convergence
4	3.14159302715971	3.73569917222483e-07	0
8	3.14159265933494	5.7451479129611e-09	6.02289043581827
16	3.14159265367921	8.94155860464707e-11	6.00567389659077
32	3.14159265359119	1.397992832608e-12	5.99909746048548

cs375-img12.png

(d) Script

```
function integral = comp_gauss_int(f, a, b, k, n)
    h = (b - a) / n;
    integral = 0;
```

```
for i = 1:n
    x_i = a + (i - 1) * h;
    x_i_next = x_i + h;
    integral = integral + gauss_int(f, x_i, x_i_next, k);
end
end
```

(e)

n	Approx	Err	Convergence
4	3.14159265332829	2.61503263487839e-10	0
8	3.14159265358879	1.00452979268084e-12	8.0241647881531
16	3.14159265358979	3.5527136788005e-15	8.14338321398982
32	3.1415926535898	2.66453525910038e-15	0.415037499278844

cs375-img11.png

2. Write a Matlab script to compute an Euler approximation to the IVP

$$x' = f(t, x), \quad x(t_0) = x_0.$$

Your method should have the following form: `[t,x] = my_euler(f,x0,t0,tend,n)`. The input parameters are (in order) a function that evaluates the right-hand side of the ODE, the initial condition x_0 , the left end-point of the interval (t_0), the right end-point of the interval, and the number of steps to take. The outputs should both be size $n + 1$ vectors which contain the discretized interval t_0, t_1, \dots, t_n and the approximate solution at these points x_0, x_1, \dots, x_n .

Ans:

```
function [t,x] = my_euler(f, x0, t0, tend, n)
    h = (tend - t0) / n;

    t = linspace(t0, tend, n + 1);
    x = zeros(1, n + 1);

    x(1) = x0;

    for i = 1:n
        x(i + 1) = x(i) + h * f(t(i), x(i));
    end
end
```

3. Consider this initial value problem (IVP)

$$\begin{cases} y'(t) = (y(t))^2 - y(t), & t \in [0, 2], \\ y(0) = 0.5. \end{cases}$$

Approximate the solution to this IVP with the Euler method, using $h = 1.0, 0.5, 0.25$, and 0.125 . Plot the approximate solutions on the same axes.

Ans:

Using the Matlab script:

```
f = @(t, y) y^2 - y;
y0 = 0.5;
t0 = 0;
tend = 2;

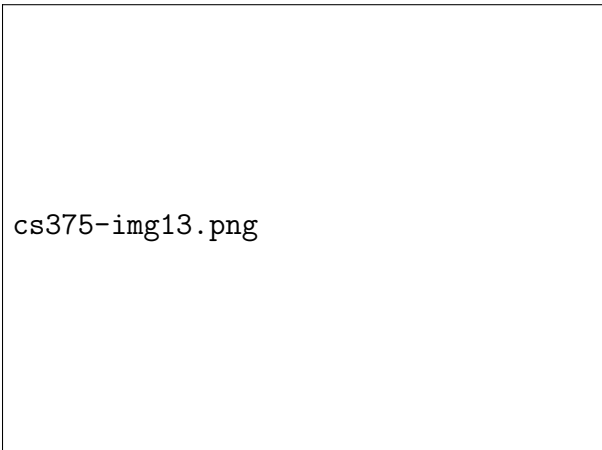
h_values = [1.0, 0.5, 0.25, 0.125];

figure;
hold on;

for h = h_values
    n = (tend - t0) / h;
    [t, y] = my_euler(f, y0, t0, tend, n);

    plot(t, y, '-o', 'DisplayName', sprintf('h = %.3f', h));
end
```

We get the following graph of values:



cs375-img13.png

4. Consider the following IVP

$$y'(t) = (y(t))^{(1/3)}$$

$$y(0) = 0$$

Show that both $y(t) = 0$ and $y(t) = (\frac{2}{3}t)^{3/2}$ are solutions to this IVP. Trying approximating the solution with Euler's method and $h = 0.1$ Which of these two solutions does Euler's method find?

Ans:

To show that $y(t) = 0$ and $y(t) = (\frac{2}{3}t)^{3/2}$ are solutions to the IVP we need to find the derivative of each and $y(0)$ and verify that they equal the IVP equations.

For $y(t) = 0$:

$$y'(t) = 0$$

$$0 = 0^{1/3}$$

$$0 = 0 \checkmark$$

$$y(0) = 0 \checkmark$$

For $y(t) = (\frac{2}{3}t)^{3/2}$:

$$y'(t) = \frac{2}{3} \cdot \frac{3}{2} (\frac{2}{3}t)^{1/2}$$

$$y'(t) = 1 \cdot (\frac{2}{3}t)^{1/2}$$

$$y'(t) = (\frac{2}{3}t)^{1/2}$$

$$(\frac{2}{3}t)^{1/2} = (y(t))^{(1/3)}$$

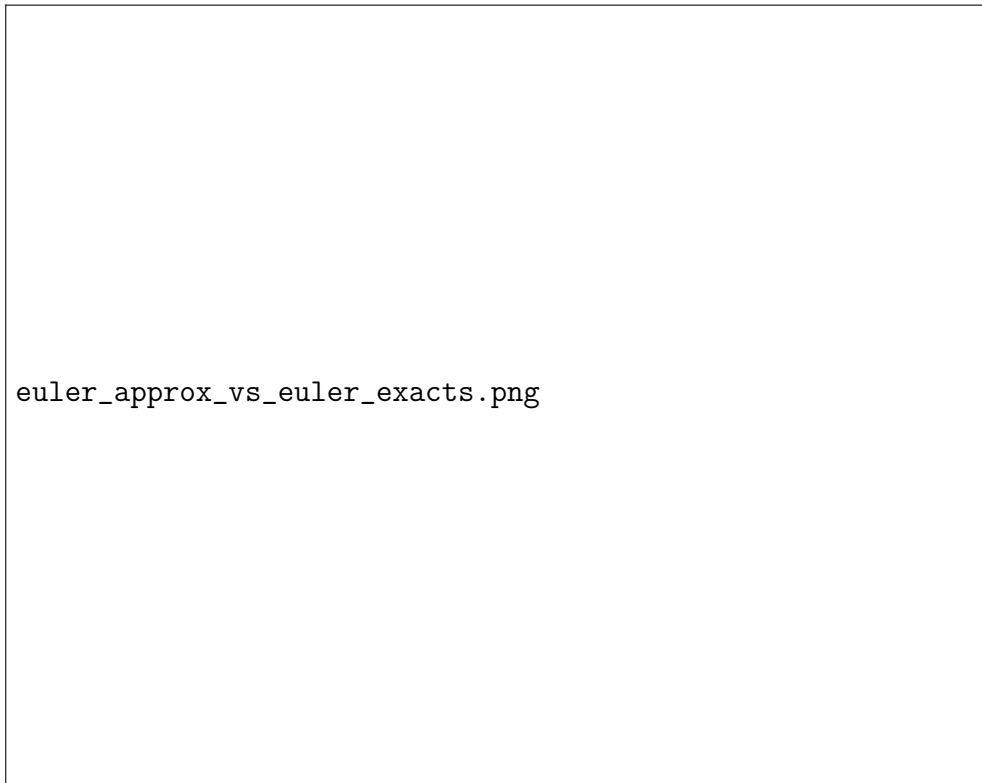
$$(\frac{2}{3}t)^{1/2} = (\frac{2}{3}t)^{3/2 \cdot 1/3}$$

$$(\frac{2}{3}t)^{1/2} = (\frac{2}{3}t)^{1/2}$$

$$y(0) = (\frac{2}{3} \cdot 0)^{3/2} \checkmark$$

$$y(0) = 0 \checkmark$$

Plotting the approximation with Euler's method against the two solutions:



euler_approx_vs_euler_exacts.png

The Euler's method approximation we are using matches with and finds the $y(t) = 0$ solution.

5. Suppose that an ordinary differential equation is solved numerically on an interval $[a, b]$ and that the local truncation error is ch^p . Show that if all truncation errors have the same sign (the worst possible case), then the total truncation error is $(b-a)ch^{p-1}$, where $h = (b-a)/n$.

Ans:

For each step, the local error will be proportional to h^p . We can represent the local error at each step as ch^p .

For the interval $[a, b]$, we divide the interval into n steps, s.t. $h = \frac{b-a}{n}$.

We find the worst case error when each of the errors have the same sign, since this means we aren't ever canceling some portion of the previous error out (as we would if we were oscillating signs). Therefore, the worst case error can be represented as $n * ch^p$.

Putting the two together we get

$$\begin{aligned}
 \text{Total error} &= n * ch^p \\
 &= \frac{b-a}{h} * ch^p \\
 &= \frac{b-a}{h} (c * h * h^{p-1}) \\
 &= (b-a)ch^{p-1}
 \end{aligned}$$

6. Consider the following ODE:

$$\begin{cases} u' = -u^2 - 2\sin(2t) + (\cos(2t))^2, & t \in [0, 1] \\ u(0) = 1 \end{cases}$$

- (a) Verify that $u(t) = \cos(2t)$ satisfies both the ODE and the initial condition.
- (b) Use your Euler method to solve the above ODE using $n = 10, 20, 40, 80$ time steps ($h = 0.1, 0.05, 0.025, 0.0125$) Tabulate your results by showing four columns: the step size h , the value of the approximation at $t = 1$, the error, and the observed order of convergence p for each value of h . Recall that the order of convergence p is computed as

$$p = \frac{\log(err^{(n)}/err^{(n-1)})}{\log(h^{(n)}/h^{(n-1)})}$$

where $err(h_k)$ is the error at $t = 1$ corresponding to a step size of h_k . Do you see the expected convergence? If not, try to explain your results.

Ans:

- (a) To verify that $u(t) = \cos(2t)$ satisfies both the ODE and the initial condition we need to compute u' , and plug in u and u' to the ODE and verify that the left and right-hand sides equal each other. Then we need to verify that $u(0)$ does equal 1. First,

$$u(t) = \cos(2t)$$

$$u'(t) = -2\sin(2t)$$

Plugging into u and u' into the ODE equation:

$$-2\sin(2t) = -(\cos(2t))^2 - 2\sin(2t) + (\cos(2t))^2$$

$$-2\sin(2t) = -(\cos(2t))^2 + (\cos(2t))^2 - 2\sin(2t)$$

$$-2\sin(2t) = -2\sin(2t) \checkmark$$

Now we verify $u(0) = 1$

$$u(t) = \cos(2t)$$

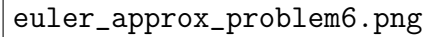
$$u(0) = \cos(0)$$

$$\cos(0) = 1$$

$$u(0) = 1 \checkmark$$

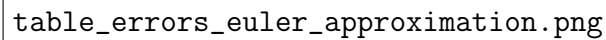
Thus, $u(t) = \cos(2t)$ satisfies both the ODE and the initial condition.

(b) The graph we get using our Euler method:



euler_approx_problem6.png

The table of errors and order of convergence p for each h :



table_errors_euler_approximation.png

The Euler method has order of convergence $p = 1$ which the p values in the table are getting closer to with decreasing values of h so we do see the expected convergence. Additionally, the error is getting smaller as h values decrease.

7. Write a Matlab script to compute a fourth order Runge-Kutta approximation to the IVP

$$x' = f(t, x), \quad x(t_0) = x_0$$

Your method should have the following form: `[t,x] = my_rkr4(f,x0,t0,tend,n)`. The input parameters are (in order) a function that evaluates the right hand side of the ODE, the initial condition x_0 , the left end-point of the interval (t_0), the right end- point of the interval, and the number of steps to take. The outputs should both be size $n + 1$ vectors which contain the discretized interval t_0, t_1, \dots, t_n and the approximate solution at these points x_0, x_1, \dots, x_n .

Ans:

```
function [t, x] = my_rkr4(f, x0, t0, tend, n)
    h = (tend - t0) / n;
    t = linspace(t0, tend, n + 1);
    x = zeros(size(t));
    x(1) = x0;

    for i = 1:n
        k1 = f(t(i), x(i));
        k2 = f(t(i) + h/2, x(i) + h*k1/2);
        k3 = f(t(i) + h/2, x(i) + h*k2/2);
        k4 = f(t(i) + h, x(i) + h*k3);

        x(i + 1) = x(i) + h * (k1 + 2*k2 + 2*k3 + k4) / 6;
    end
end
```

8. Next, consider an IVP that represents a flame problem (from Mathworks). When combustion starts (say a match is lit), then the ball of flame expands quickly until it reaches a steady state. This steady state represents the balance between the fuel on the surface of the combusting material and the fuel in the interior of the material expanding. A straight-forward IVP models this process well,

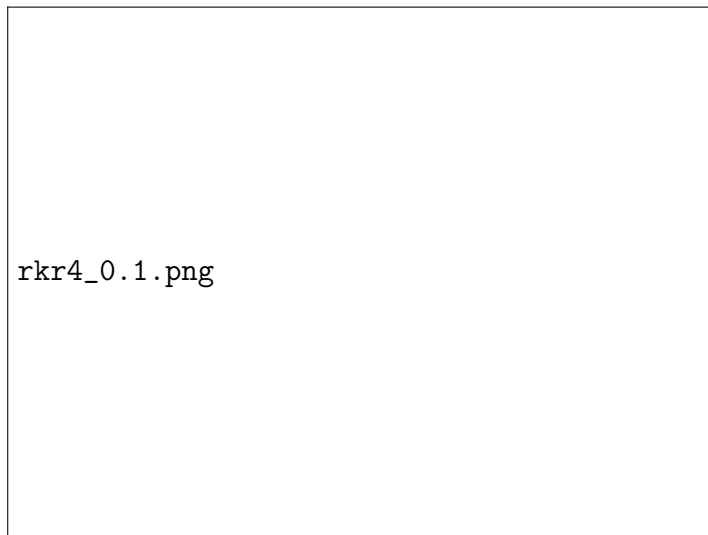
$$y'(t) = (y(t))^2 - (y(t))^3, \quad 0 \leq t \leq 2/\delta$$

$$y(0) = \delta,$$

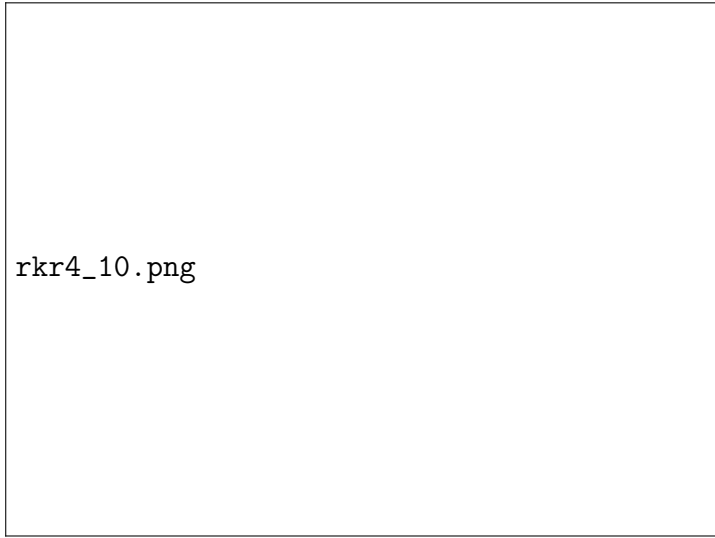
where $y(t)$ is the radius of the flame (and $(y(t))^2$ and $(y(t))^3$ are the surface and volume contributions). The value δ is the initial radius.

Use the Runge-Kutta Method you wrote in the previous problem to solve this ODE with $\delta = 0.01$. Try using $h = 10.0$ and $h = 0.1$. For both h -values, provide a plot of the solution, with the x-axis representing time, and the y-axis representing the flame radius. Provide one or two sentences interpreting the overall shape of the plot for $h = 0.1$. That is, how is the flame radius varying over time? Provide one or two sentences explaining the reason(s) for any differences between your two plots.

Ans: Runge-Kutta with $h = 0.1$



As time increases, the flame radius slowly increases until it reaches $t = 100s$ where the radius then jumps up to 1 and stays at the radius for the rest of the time.



rkr4_10.png

The difference we see in this solution with a larger h value is that once the solution starts to rapidly increase there is numerical instability and our Runge-Kutta method fails. Thus, a suitable h needs to be chosen for the method to work as we want it.

9. Find the region of stability for the RK2a method (from lecture). Express your answer as an inequality using the variable $z = \lambda h$, where h is the step-size of the method and λ is the constant from the Dahlquist problem. Graph the region in the complex plane.

Ans: The Dahlquist test problem is:

$$f(t_i, y_i) = y' = \lambda y_i$$

To derive the region of stability for the RK2a method, we need to substitute $f(y)$ into the RK2a method

$$\begin{aligned} k_1 &= \lambda y_i \\ k_2 &= \lambda(y_i + hK_1) \\ k_2 &= \lambda(y_i + h\lambda y_i) \\ k_2 &= \lambda y_i(1 + h\lambda) \end{aligned}$$

Now we substitute K_1 and K_2 into

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{2}(K_1 + K_2) \\ y_{i+1} &= y_i + \frac{h}{2}(\lambda y_i + \lambda y_i(1 + h\lambda)) \\ y_{i+1} &= y_i + \frac{h\lambda y_i}{2}(1 + (1 + h\lambda)) \\ y_{i+1} &= y_i + \frac{h\lambda y_i}{2}(2 + h\lambda) \end{aligned}$$

Now substituting $z = h\lambda$ to simplify, we get

$$y_i + zy_i + \frac{z^2 y_i}{2}$$

Amplification Factor $G(z)$

$$G(z) = \frac{y_{i+1}}{y_i}$$

Substituting our equation from above,

$$G(z) = 1 + z + \frac{z^2}{2}$$

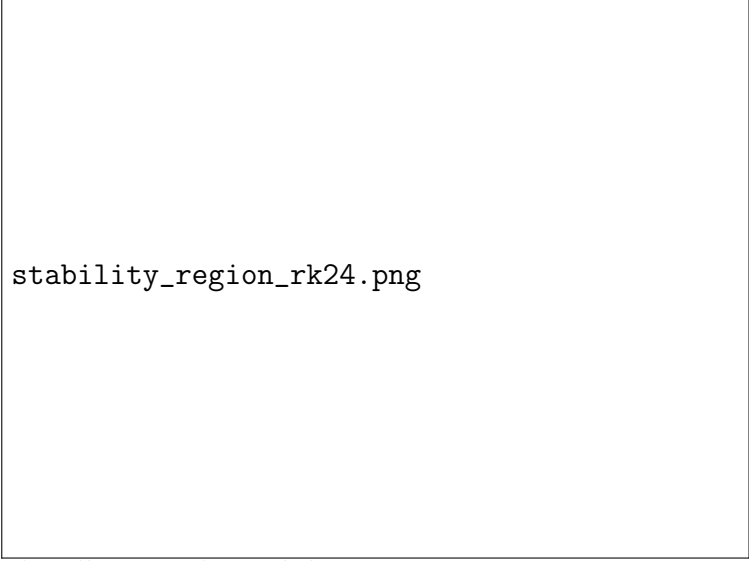
We know for stability,

$$|G(z)| < 1$$

Thus, the region of stability for RK2a is

$$\left|1 + z + \frac{z^2}{2}\right| < 1$$

Graphing the region of stability in the complex plane:



stability_region_rk24.png

The region within the ellipse is the stability region.