2021 Spring MAS 365: Homework 8

posted on May 27; due by June 3

1. [10+5 points]

   (a) Determine constants a, b, c and d that will produce a quadrature formula

   $$\int_{-1}^{1} f(x)dx = af(-1) + bf(1) + cf'(-1) + df'(1)$$

   that has degree of precision three.

   (b) Show that the formula $\sum_{i=1}^{n} c_i P(x_i)$ cannot have degree of precision greater than $2n-1$, regardless of the choice of $c_1, \ldots, c_n$ and $x_1, \ldots, x_n$. [Hint: Construct a polynomial that has a double root at each of the x_j's.]

   Solution:

   (a) We wan to find a, b, c, d such that the formula is exact for $1, x, x^2, x^3$, *i.e.*,

   $$\begin{aligned} 2 &= \int_{-1}^{1} 1dx = a + b, \\ 0 &= \int_{-1}^{1} xdx = -a + b + c + d, \\ \frac{2}{3} &= \int_{-1}^{1} x^2dx = a + b - 2c + 2d, \\ 0 &= \int_{-1}^{1} x^3dx = -a + b + 3c + 3d. \end{aligned}$$

   The solution of the above linear system is

   $$a = 1,\ b = 1,\ c = \frac{1}{3},\ d = -\frac{1}{3}.$$

   (b) Let $P(x) = \prod_{i=1}^{n}(x - x_i)^2$. This satisfies $\sum_{i=1}^{n} c_i P(x_i) = 0$, while $\int_{-1}^{1} P(x)dx \neq 0$. So the formula cannot have degree of precision greater than $2n-1$.

2. [10 points] Show that the initial-value problem

   $$y' = t^2y + 1, \quad 0 \leq t \leq 1, \quad y(0) = 1$$

   is well-posed.

   Solution: Let $f(t, y) = t^2y + 1$ and $D = \{(t, y) \mid 0 \leq t \leq 1,\ -\infty < y < \infty\}$. We have

   $$\left|\frac{\partial f}{\partial y}\right| = \left|t^2\right| \leq 1, \quad \text{for all } (t, y) \in D,$$

   and thus f satisfies a Lipschitz condition on D in the variable y with $L = 1$. Since $f(t, y)$ is continuous, the problem is well-posed.

3. [5+10 points]

   (a) Use Euler's method with $h = 0.5$ to approximate the solution to

$$y' = \frac{1+t}{1+y}, \quad 1 \le t \le 2, \quad y(1) = 2.$$

   (b) Given that the solution to the above problem is $y(t) = \sqrt{t^2 + 2t + 6} - 1$, find the bound for the approximation error. [Hint: Consider a set $D = \{(t, y) \mid 1 \le t \le 2 \text{ and } 2 \le y < \infty\}$ for the Lipschitz condition in the variable y.]

Solution:

   (a) Euler method is

$$w_0 = 2,$$
$$w_{i+1} = w_i + h\frac{1+t_i}{1+w_i}$$

   and the approximations are given below.

t_i	1.0	1.5	2.0
w_i	2.0000	2.3333	2.7083

   (b) Let $f(t, y) = \frac{1+t}{1+y}$. Since $\left|\frac{\partial f(t,y)}{\partial y}\right| = \left|-\frac{1+t}{(1+y)^2}\right|$, f satisfies Lipschitz condition with $L = \frac{1}{3}$ in the variable y on $D = \{(t, y) \mid 1 \le t \le 2 \text{ and } 2 \le y < \infty\}$. In addition, y satisfies

$$|y''(t)| = \left|\frac{5}{(t^2 + 2t + 6)^{\frac{3}{2}}}\right| \le M = \frac{5}{27}$$

   for $t \in [1, 2]$. Thus, for each $i = 0, 1, 2$, we have

$$|y(t_i) - w_i| \le \frac{0.5 \times \frac{5}{27}}{2 \times \frac{1}{3}}[e^{\frac{1}{3} \times 0.5i} - 1]$$

   gives the error bound below.

t_i	1.0	1.5	2.0
error bound	0.0000	0.0252	0.0549

4. [10+5 points]

   (a) Derive the Adams-Bashforth Two-Step method by using the Lagrange form of the interpolating polynomial.

   (b) Derive Milne's method in page 313 of the textbook by applying the open Newton-Cotes formula to the integral, and report its local truncation error.

$$y(t_{i+1}) - y(t_{i-3}) = \int_{t_{i-3}}^{t_{i+1}} f(t, y(t))dt.$$

Solution:

(a) Let $P_1(t)$ be the linear Lagrange polynomial

$$P_1(t) = \frac{t - t_{i-1}}{t_i - t_{i-1}} f(t_i, y(t_i)) + \frac{t - t_i}{t_{i-1} - t_i} f(t_{i-1}, y(t_{i-1})).$$

Then we have

$$\begin{aligned}
y(t_{i+1}) - y(t_i) &= \int_{t_i}^{t_{i+1}} f(t, y(t))dt \approx \int_{t_i}^{t_{i+1}} P_1(t)dt \\
&= \frac{f(t_i, y(t_i))}{t_i - t_{i-1}} \int_{t_i}^{t_{i+1}} (t - t_{i-1})dt + \frac{f(t_{i-1}, y(t_{i-1}))}{t_{i-1} - t_i} \int_{t_i}^{t_{i+1}} (t - t_i)dt \\
&= \frac{3}{2} h f(t_i, y(t_i)) - \frac{1}{2} h f(t_{i-1}, y(t_{i-1})).
\end{aligned}$$

Replacing $y(t_j)$ with w_j, for $j = i-1, i, i+1$ gives

$$\begin{aligned}
w_0 = \alpha, \quad w_1 = \alpha_1, \\
w_{i+1} = w_i + \frac{h}{2}[3f(t_i, w_i) - f(t_{i-1}, w_{i-1})],
\end{aligned}$$

where $i = 1, 2, \ldots, N-1$.

(b) Open Newton Cotes formula (for $n = 2$) gives

$$\begin{aligned}
y(t_{i+1}) - y(t_{i-3}) &= \int_{t_{i-3}}^{t_{i+1}} f(t, y(t))dt \\
&= \frac{4h}{3}[2f(t_i, y(t_i)) - f(t_{i-1}, y(t_{i-1})) + 2f(t_{i-2}, y(t_{i-2})] + \frac{14h^5}{45} f^{(4)}(\xi, y(\xi))
\end{aligned}$$

for some number ξ in (t_{i-3}, t_{i+1}). The corresponding difference equation is

$$w_{i+1} = w_{i-3} + \frac{4h}{3}[2f(t_i, w_i) - f(t_{i-1}, w_{i-1}) + 2f(t_{i-2}, w_{i-2})],$$

with the local truncation error

$$\tau_{i+1}(h) = \frac{14h^5}{45} f^{(4)}(\xi, y(\xi)) = \frac{14h^4}{45} y^{(5)}(\xi)$$

for some ξ in (t_{i-3}, t_{i+1}).

5. [10+10 points]

(a) Implement Taylor's method of order two with cubic Hermite interpolation via MATLAB grader.

(b) Implement Adams fourth-order predictor-corrector method via MATLAB grader.

Solution:

(a)
```
function w = IVP_Taylor_Hermite(f, fp, a, b, alpha, N, t0)
    h = (b-a)/N;
    tt = a:h:b;
    ww = zeros(N+1,1);
    % Taylor
    ww(1) = alpha;
    for i = 1:N
        ww(i+1) = ww(i) + h*f(tt(i),ww(i)) + h^2/2*fp(tt(i),ww(i));
```

```
        end
        i1 = floor((t0-a)/h) + 1;
        i2 = ceil((t0-a)/h) + 1;
        if i1 == i2
            w = ww(i1);
        else
            % cubic Hermite interpolation
            z = [tt(i1); tt(i1); tt(i2); tt(i2)];
            Q = zeros(4,4);
            Q(:,1) = [ww(i1); ww(i1); ww(i2); ww(i2)];
            Q([2 4],2) = [f(tt(i1),ww(i1)); f(tt(i2),ww(i2))];
            Q(3,2) = (Q(3,1) - Q(2,1))/(z(3) - z(2));
            for i=2:3
            for j=2:i
                Q(i+1,j+1) = (Q(i+1,j) - Q(i,j))/(z(i+1) - z(i-j+1));
            end
            end
            w = Q(1,1) + (t0-tt(i1))*Q(2,2) + (t0-tt(i1))^2*Q(3,3) ...
                + (t0-tt(i1))^2*(t0-tt(i2))*Q(4,4);
        end
    end
```

(b)

```
    function w = IVP_Adams_Predictor_Corrector(f, a, b, alpha, N)
        h = (b-a)/N;
        t = a:h:b;
        w = zeros(N+1,1);
        w(1) = alpha;
        for i = 1:3
            k1 = h*f(t(i), w(i));
            k2 = h*f(t(i) + h/2, w(i) + 1/2*k1);
            k3 = h*f(t(i) + h/2, w(i) + 1/2*k2);
            k4 = h*f(t(i) + h, w(i) + k3);
            w(i+1) = w(i) + 1/6*(k1 + 2*k2 + 2*k3 + k4);
        end
        for i=4:N
            wp = w(i) + h/24*(55*f(t(i),w(i)) - 59*f(t(i-1),w(i-1)) ...
                 + 37*f(t(i-2),w(i-2)) - 9*f(t(i-3),w(i-3)));
            w(i+1) = w(i) + h/24*(9*f(t(i+1),wp) + 19*f(t(i),w(i)) ...
                 - 5*f(t(i-1),w(i-1)) + f(t(i-2),w(i-2)));
        end
    end
    end
```