APPM 5610 - Homework 8

Eappen Nelluvelil

April 6, 2022

1. A popular explicit Runge-Kutta method is defined by the following formulas:

$$\begin{aligned}
k_1 &= hf\left(x_n, y_n\right) \\
k_2 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right) \\
k_3 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right) \\
k_4 &= hf\left(x_n + h, y_n + k_3\right) \\
y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right).
\end{aligned}$$

Estimate the region of absolute stability of this method by calculating all intersections of the region of absolute stability with the real and imaginary axes.

We consider the model problem $y'\left(x\right) = f\left(x, y\left(x\right)\right)$, where

$$\begin{aligned}
y'\left(x\right) &= \lambda y, \\
y\left(x_0\right) &= y_0.
\end{aligned}$$

We then perform all the intermediate calculations and arrive at

$$\begin{aligned}
k_1 &= \lambda h y_n, \\
k_2 &= \lambda h y_n + \frac{1}{2}\left(\lambda h\right)^2 y_n, \\
k_3 &= \lambda h y_n + \frac{1}{2}\left(\lambda h\right)^2 y_n + \frac{1}{4}\left(\lambda h\right)^3 y_n, \\
k_4 &= \lambda h y_n + \left(\lambda h\right)^2 y_n + \frac{1}{2}\left(\lambda h\right)^3 y_n + \frac{1}{4}\left(\lambda h\right)^4 y_n, \\
y_{n+1} &= \left(1 + \lambda h + \frac{1}{2}\left(\lambda h\right)^2 + \frac{1}{6}\left(\lambda h\right)^3 + \frac{1}{24}\left(\lambda h\right)^4\right) y_n.
\end{aligned}$$

For this scheme to converge for the model problem, we require that

$$\left|\left(1 + \lambda h + \frac{1}{2}\left(\lambda h\right)^2 + \frac{1}{6}\left(\lambda h\right)^3 + \frac{1}{24}\left(\lambda h\right)^4\right)\right| < 1$$

for $\lambda h \in \mathbb{C}$. We compute the absolute stability region in Mathematica by finding all $\lambda h \in \mathbb{C}$ for which $\left|\left(1 + \lambda h + \frac{1}{2}\left(\lambda h\right)^2 + \frac{1}{6}\left(\lambda h\right)^3 + \frac{1}{24}\left(\lambda h\right)^4\right)\right| < 1$.

2. One seeks the solution of the eigenvalue problem

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\,x}\left[\left(\frac{1}{1+x}\right)\frac{\mathrm{d}\,y}{\mathrm{d}\,x}\right] + \lambda y &= 0, \\
y\left(0\right) &= 0, \\
y\left(1\right) &= 0,
\end{aligned}$$

by integrating, for a few values of λ, an equivalent system of two first order differential equations with initial values $y(0) = 0$ and $y'(0) = 1$, using the trapezoidal method combined with Richardson's extrapolation developed in a previous assignment. Taking λ in the range $[6.7, 6.8]$, compute the value of λ for which $y(1) = 0$.

We define auxiliary variables u and v as follows:

$$\begin{aligned} u &= y, \\ v &= u' = y', \\ v' &= u'' = y''. \end{aligned}$$

The eigenvalue problem can be written as

$$\left(\frac{1}{1+x}\right) y'' - \left(\frac{1}{(1+x)^2}\right) y' + \lambda y = 0.$$

Substituting the auxiliary variables that we introduced earlier, we obtain the equivalent first-order system:

$$\begin{aligned} \begin{bmatrix} u'(x) \\ v'(x) \end{bmatrix} &= \begin{bmatrix} v(x) \\ \left(\frac{1}{1+x}\right) v(x) - (1+x)\lambda u(x) \end{bmatrix}, \\ \begin{bmatrix} u(0) \\ v(0) \end{bmatrix} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \end{aligned}$$

We then use the trapezoidal method and Richardson extrapolation code developed earlier for assignment six, loop through discrete values for λ in the range $[6.7, 6.8]$ (in the code, this is given by 200 equispaced points between 6.7 and 6.8), and we obtain $\lambda \approx 6.7738693467336679$.

```matlab
clc;
clear;
close all;

n = 200;
lambdas = linspace(6.7, 6.8, n);
y_t_f = zeros(length(lambdas), 1);

t_0 = 0;
t_f = 1;
y_0 = [0; 1];

% Maximum number of rows for Richardson extrapolation
max_rows = 17;
tol = 1e-15;

for k = 1:length(lambdas)
    lambda = lambdas(k);

    % Richardon extrapolation matrix
    A = zeros(max_rows, max_rows);

    h = (t_f - t_0);
    y = trapezoid(@odefun, t_0, t_f, lambda, h, y_0);

    A(1, 1) = y(1);

    for i = 1:(max_rows - 1)
        h = h/2;
        y = trapezoid(@odefun, t_0, t_f, lambda, h, y_0);
        A(i + 1, 1) = y(1);

        for j = 1:i
            A(i + 1, j + 1) = ((4^j)*A(i + 1, j) - A(i, j))/((4^j) - 1);
        end

        if abs(A(i + 1, i + 1) - A(i, i)) < tol
            break;
        end
    end

    y_t_f(k) = abs(A(max_rows, max_rows));
end

% Find the value of lambda that makes y(1) closest to 0
optimal_lambda_idx = find(y_t_f == min(y_t_f));
fprintf("Optimal lambda: %0.16f\n", lambdas(optimal_lambda_idx));

function [f] = odefun(t, y, lambda)
    f = [y(2); ...
        (1/(1+t))*y(2) - (1+t)*lambda*y(1)];
end
```

```
function [y] = trapezoid(odefun, t_0, t_f, lambda, h, y_0)
    y = y_0;

    while t_0 <= t_f
        E_y = y + (h/2)*odefun(t_0, y, lambda);
        I_m = [1, -h/2;
               (h/2)*(1 + (t_0 + h))*lambda, 1 - (h/2)*(1/(1 + (t_0 + h)))];
        I_y = I_m\E_y;

        y = E_y + (h/2)*odefun(t_0 + h, I_y, lambda);
        t_0 = t_0 + h;
    end
end
```

```
Optimal lambda: 6.7738693467336679
```

Published with MATLAB® R2022a

In[2]:=
```
ComplexRegionPlot[Abs[1 + r + (1 / 2) * r^(2) + (1 / 6) * r^(3) + (1 / 24) * r^(4)] ≤ 1,
 {r, 3}, FrameLabel → {"Real", "Imaginary"},
 PlotLabel → "Absolute stability region for explicit RK4"]
```

Out[2]=