AMATH 586 Assignment 1

Tyler Chen

Problem 1

Prove that the ODE IVP

$$u'(t) = \frac{1}{t^2 + 2u(t)^2}, \quad t \geq 1,$$
$$u(1) = \eta,$$

has a unique solution for all $t \geq 1$. Find an optimal or near optimal Lipschitz constant for this problem.

Solution

Write $f(u,t) = 1/(t^2 + 2u^2)$. Then,

$$\frac{\partial f}{\partial u} = -\frac{4u}{(t^2 + 2u^2)^2}$$

We now maximize $|f_u(u,t)|$ with respect to u. We have,

$$\frac{\partial^2 f}{\partial u^2} = \frac{32u^2}{(t^2 + 2u^2)^3} - \frac{4}{(t^2 + 2u^2)^2}$$

Clearly $f_{uu} = 0$ at $u = \pm t/\sqrt{6}$. We can easily check that both these points maximize $|f_u(u,t)|$. Therefore, since f_u is a decreasing function of t , for any domain,

$$\mathcal{D} = \{(u,t) : |u - \eta| \leq a, t_0 \leq t \leq t_1\}$$

we have Lipschitz constant,

$$L = \frac{\sqrt{3^3/2^5}}{t_0^3}$$

Therefore, by a theorem from the book, the ODE IVP has a unique solution at least up to time,

$$T^* = \min\{t_1, t_0 + a/S\}, \qquad S = \max_{(u,t)\in\mathcal{D}} |f(u,t)|$$

Observe that,

$$S = \max_{t\geq 1} |f(u,t)| \leq \max_{t\geq 1} \left|\frac{1}{t^2}\right| = 1$$

and that if $u = 0$ this inequality is tight.

Therefore, for any domain $\mathcal{D}$ of the form above, $t_0 + a/S \geq t_0 + a$.

Let $t \geq 1$. Pick $t_0 = 1$, $t_1 > t$, and $a > t - 1$ so that,

$$\min\{t_1, t_0 + a/S\} > t$$

Then there is a unique solution to the ODE IVP up to time $T^* > t$. This proves that the ODE IVP has a unique solution for all time $t \geq 1$. For domains of this form we have Lipschitz constant,

$$L = \sqrt{\frac{3^3}{2^5}}$$

This constant is optimal since the derivative is continuous. To see this consider the expansion of $f_u(t_0/\sqrt{6} + h, t_0)$ about $(t_0/\sqrt{0}, t_0)$. Clearly this can be made arbitrarily close to L by picking h sufficiently small. This means that for any $l < L$, there is some interval for which $|f_u(t_0/\sqrt{6}, t_0) - f_u(t_0/\sqrt{6} + h, t_0)| > l|u_0 + h - u_0|$. That is, l is not a Lipschitz constnat for f with respect to u. □

Problem 2

Consider the equation of harmonic motion:

$$u'' = -ku, \quad u(t_0) = u_0, \quad u'(t_0) = v_0.$$

Here $u(t)$ represents the distance from equilibrium and $k > 0$ is a spring constant. Write this as a system of two first-order differential equations, and show that the right-hand side of your system satisfies a Lipschitz condition on $\mathbf{R}^2$. Determine the (smallest possible) Lipschitz constant for the 1-norm, the 2-norm, and the ∞-norm. (See Appendix A.3.)

Solution

Define $v = u'$. Then $v' = u''$ so the original equation can be written as,

$$\begin{cases} v' = -ku \\ u' = v, \end{cases} \qquad \begin{cases} u(t_0) = u_0 \\ v(t_0) = v_0 \end{cases}$$

We now show that the right hand side is uniformly Lipschitz by finding Lipschitz constants for the 1-norm, 2-norm, and the ∞-norm. Write,

$$x = \begin{bmatrix} v \\ u \end{bmatrix}, \qquad f(x) = \begin{bmatrix} -ku \\ v \end{bmatrix}$$

Define the indicator function $\mathbb{1} : \{\texttt{true}, \texttt{false}\} \to \{0, 1\}$ by $\mathbb{1}[\varphi] = \{1 : \varphi = \texttt{true},\ 0 : \varphi = \texttt{false}\}$. Observe that,

$$\begin{bmatrix} -ku \\ v \end{bmatrix} = \max\{k, 1\} \begin{bmatrix} -u \\ v \end{bmatrix} + \begin{bmatrix} (1-k)\mathbb{1}[k \le 1]u \\ (1-k)\mathbb{1}[k > 1]v \end{bmatrix}$$

Let $\tilde{x} = x - x^*$. Note that k is linear so that for any x, x^*, $f(x) - f(x^*) = f(x - x^*) = f(\tilde{x})$. Therefore, for any norm $\|\cdot\|$, by the triangle inequality,

$$\|f(\tilde{x})\| \le \max\{k, 1\} \|\tilde{x}\|$$

Note that $\tilde{x} = [\mathbb{1}[k \ge 1], \mathbb{1}[k < 1]]$ attains the bound above. This proves that $\max\{k, 1\}$ is the optimal Lipschitz constant for any norm. $\square$

Problem 3

Consider the one-step method

$$U^{n+1} = U^n + k[\theta f(U^n, t_n) + (1-\theta) f(U^{n+1}, t_{n+1})],$$

where $\theta \in [0,1]$ is given. Note that this method is *explicit* if $\theta = 1$ and otherwise it is implicit. Show that the local truncation error is $O(k^2)$ if $\theta = 1/2$ and otherwise it is $O(k)$.

Solution

Note,

$$f(u(t_n), t_n) = u'(t_n), \qquad f(u(t_{n+1}), t_{n+1}) = u'(t_{n+1})$$

We expand,

$$u(t_{n+1}) = u(t_n) + u'(t_n)k + \frac{1}{2!}u''(t_n)k^2 + \frac{1}{3!}u'''(t_n)k^3 + \mathcal{O}(k^4)$$
$$u'(t_{n+1}) = u'(t_n) + u''(t_n)k + \frac{1}{2!}u'''(t_n)k^2 + \frac{1}{3!}u''''(t_n)k^3 + \mathcal{O}(k^4)$$

Therefore,

$$\frac{1}{k}\left(u(t_{n+1}) - u(t_n)\right) = u'(t_n) + \frac{1}{2!}u''(t_n)k + \frac{1}{3!}u'''(t_n)k^2 + \mathcal{O}(k^3)$$
$$\theta u'(t_n) + (1-\theta)u'(t_{n+1}) = u'(t_n) + (1-\theta)\left(u''(t_n)k + \frac{1}{2!}u'''(t_n)k^2 + \mathcal{O}(k^3)\right)$$

We have local truncation error,

$$\begin{aligned}\tau^n &= \frac{1}{k}\left(u(t_{n+1}) - u(t_n)\right) - \theta f(u(t_n), t_n) - (1-\theta) f(u(t_{n+1}), t_{n+1}) \\ &= \left(\frac{1}{2!}u''(t_n)k + \frac{1}{3!}u'''(t_n)k^2 + \mathcal{O}(k^3)\right) - (1-\theta)\left(u''(t_n)k + \frac{1}{2!}u'''(t_n)k^2 + \mathcal{O}(k^3)\right)\end{aligned}$$

Clearly the $\mathcal{O}(k)$ terms cancel if and only if $\theta = 1/2$, in which case the leading order terms are $\mathcal{O}(k^2)$. Otherwise the leading order terms are $\mathcal{O}(k)$. □

Problem 4

Compute the leading term in the local truncation error of the following methods:

(a) Heun's method:

$$\begin{aligned} \tilde{U}^{n+1} &= U^n + kf(U^n, t_n), \\ U^{n+1} &= U^n + \frac{k}{2}\left[f(\tilde{U}^{n+1}, t_{n+1}) + f(U^n, t_n)\right]. \end{aligned}$$

(b) The Runge-Kutta method (5.32).

$$\begin{aligned} \tilde{U}^{n+1} &= U^n + \frac{1}{2}kf(U^n, t_n), \\ U^{n+1} &= U^n + kf\left(\tilde{U}^{n+1}, t_n + \frac{k}{2}\right) \end{aligned}$$

Solution

(a) We expand,

$$u(t+k) = u(t) + u'(t)k + \frac{1}{2!}u''(t)k^2 + \frac{1}{3!}u'''(t)k^3 + \mathcal{O}(k^4)$$

Similarly, in two dimensions,

$$\begin{aligned} f(u + ku', t + k) = f(u,t) + f_u(u,t)u'k + f_t(u,t)k \\ + \frac{1}{2!}\Big(f_{uu}(u,t)u'^2k^2 + 2f_{ut}(u,t)u'k^2 + f_{tt}(u,t)k^2\Big) + \mathcal{O}(k^3) \end{aligned}$$

Now observe that by the limit definition of the derivative,

$$u'' = \frac{\partial}{\partial t}u' = \frac{\partial}{\partial t}f(u,t) = f_u(u,t)u' + f_t(u,t)$$

We can therefore write,

$$f(u + ku', t + k) = u' + u''k + \frac{1}{2!}\left(f_{uu}(u,t)u'^2k^2 + 2f_{ut}(u,t)u'k^2 + f_{tt}(u,t)k^2\right) + \mathcal{O}(k^3)$$

Using the fact that $f(u(t_n)) = u'(t_n)$,

$$\begin{aligned} \frac{1}{k}\left(u(t_{n+1}) - u(t_n)\right) &= u'(t_n) + \frac{1}{2}u''(t_n)k + \frac{1}{3!}u'''(t_n)k^2 + \mathcal{O}(k^3) \\ f(u(t_n) + kf(u(t_n))) + f(u(t_n)) &= 2u'(t_n) + u''(t_n)k + \frac{1}{2!}\big(f_{uu}(u(t_n), t_n)u'(t_n)^2 \\ &\quad + 2f_{ut}(u(t_n), t_n)u'(t_n) + f_{tt}(u(t_n), t_n)\big)k^2 + \mathcal{O}(k^3) \end{aligned}$$

The local truncation error is then,

$$\begin{aligned}\tau^n &= \frac{1}{k}\left(u(t_{n+1}) - u(t_n)\right) - \frac{1}{2}\left(f\left(u(t_n) + kf(u(t_n))\right) - f(u(t_n))\right) \\ &= \left(\frac{1}{6}u'''(t_n) - \frac{1}{4}\Big(f_{uu}(u(t_n),t_n)u'(t_n)^2 + 2f_{ut}(u(t_n),t_n)u'(t_n) + f_{tt}(u(t_n),t_n)\Big)\right)k^2 + \mathcal{O}(k^3)\end{aligned}$$

(b) We expand,

$$u(t+k) = u(t) + u'(t)k + \frac{1}{2!}u''(t)k^2 + \frac{1}{3!}u'''(t)k^3 + \mathcal{O}(t^4)$$

Similarly, in two dimensions,

$$\begin{aligned}f\left(u + \frac{k}{2}u', t + \frac{k}{2}\right) = f(u,t) + f_u(u,t)\frac{u'}{2}k + f_t(u,t)\frac{1}{2}k \\ + \frac{1}{2!}\Big(f_{uu}(u,t)\frac{u'^2}{4}k^2 + 2f_{ut}(u,t)\frac{u'}{4}k^2 + f_{tt}(u,t)\frac{k^2}{4}\Big) + \mathcal{O}(k^3)\end{aligned}$$

Now observe that by the limit definition of the derivative,

$$u'' = \frac{\partial}{\partial t}u' = \frac{\partial}{\partial t}f(u,t) = f_u(u,t)u' + f_t(u,t)$$

We can therefore write,

$$f\left(u + \frac{k}{2}u', t + \frac{k}{2}\right)W = u' + \frac{1}{2}u''k + \frac{1}{2}\Big(f_{uu}(u,t)\frac{u'^2}{4}k^2 + 2f_{ut}(u,t)\frac{u'}{4}k^2 + f_{tt}(u,t)\frac{k^2}{4}\Big) + \mathcal{O}(k^3)$$

Using the fact that $f(u(t_n),t_n) = u'(t_n)$,

$$\begin{aligned}\frac{1}{k}\left(u(t_n+k) - u(t_n)\right) &= u'(t_n) + \frac{1}{2}u''(t_n)k + \frac{1}{6}u'''(t_n)k^2 + \mathcal{O}(k^3) \\ f\left(u(t_n) + \frac{1}{2}kf(u(t_n),t_n), t_n + \frac{k}{2}\right) &= u'(t_n) + \frac{1}{2}u''(t_n)k + \frac{1}{2}\Big(\frac{1}{4}f_{uu}(u(t_n),t_n)u'(t_n)^2 \\ &\quad + \frac{1}{2}f_{ut}(u(t_n),t_n)u'(t_n) + \frac{1}{4}f_{tt}(u(t_n),t_n)\Big)k^2 + \mathcal{O}(k^3)\end{aligned}$$

The local truncation error is then,

$$\begin{aligned}\tau^n &= \frac{1}{k}\left(u(t_n+k) - u(t_n)\right) - f\left(u(t_n) + \frac{1}{2}kf(u(t_n),t_n), t_n + \frac{k}{2}\right) \\ &= \left(\frac{1}{6}u'''(t_n) - \frac{1}{2}\left(\frac{1}{4}f_{uu}(u(t_n),t_n)u'(t_n)^2 + \frac{1}{2}f_{ut}(u(t_n),t_n)u'(t_n) + \frac{1}{4}f_{tt}(u(t_n),t_n)\right)\right)k^2 + \mathcal{O}(k^3)\end{aligned}$$

Problem 5

The initial value problem

$$u'(t) = u(t)^2 - \sin(t) - \cos^2(t), \quad u(0) = 1$$

has the solution $u(t) = \cos(t)$. Write a computer code to solve this problem up to time $T = 8$ with various different time steps $k = T/N$, with

$$N = 25,\ 50,\ 100,\ 200,\ 400,\ 800,\ 1600.$$

Do this using two different methods:

(a) Forward Euler.

(b) The classical fourth-order Runge-Kutta method (5.33).

Plot the true solution (with a solid line) and the approximate solution (with o's) for $N = 25$. Also make a log-log plot of the errors (the maximum absolute value of the difference between the true and computed solution at the mesh points) for each method vs. the stepsize k.

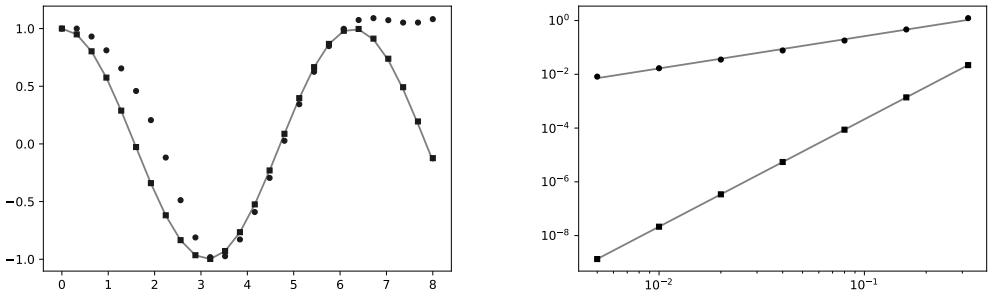
Solution

We implement these methods in Python as,

```
def fw_euler(f,u0,N,T):
    k = T/N
    u = np.zeros(N+1)
    u[0] = u0
    for n in range(N):
        u[n+1] = u[n] + k*f(u[n],k*n)
    return u
```

```
def r_k(f,u0,N,T):
    k = T/N
    u = np.zeros(N+1)
    u[0] = u0
    for n in range(N):
        tn = k*n
        Y1 = u[n]
        Y2 = u[n] + k/2*f(Y1,tn)
        Y3 = u[n] + k/2*f(Y2,tn+k/2)
        Y4 = u[n] + k*f(Y3,tn+k/2)
        u[n+1] = u[n] + k/6*( f(Y1,tn) + 2*f(Y2,tn+k/2) + 2*f(Y3,tn+k/2) + f
            (Y4,tn+k))
    return u
```

A sample solution along with the errors of each methods vs. mesh size used are show in Figure 1. The slope of the infinity norms of the errors for Forward Euler and Runge-Kutta were 1.20 and 3.99 respectively. As we have shown, simple methods have global error equal in order to the local truncation error. Our results match what is expected.



(a) computed solutions with $N = 25$ mesh points (b) infinity norm of error vs mesh spacing

Figure 1: circle: Forward Euler, square: Runge-Kutta

Problem 6

Use a method of your choice or try `ode45` in Matlab to solve the Lotka-Volterra predator-prey equations:

$$\begin{aligned} R' &= (1 - .02F)R, \\ F' &= (-1 + .03R)F, \end{aligned}$$

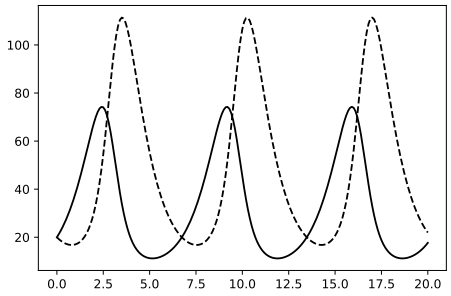
starting with $R_0 = F_0 = 20$. Plot $R(t)$ and $F(t)$ vs. t on the same plot, labeling each curve. Present the solution in another way with a plot of F vs. R.

Solution

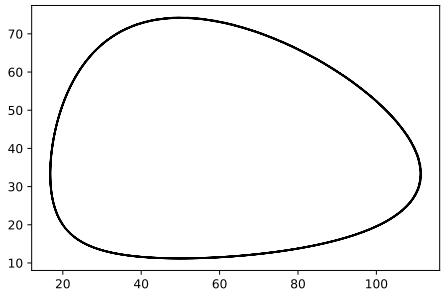
We implement this using Scipy's IVP solver, requring it to use the eplicit Runge-Kutta method of order 5(4). According to the documentation, "The error is controlled assuming 4th order accuracy, but steps are taken using a 5th oder accurate formula (local extrapolation is done)." The right hand side function is define inline as a lambda function.

```
sol = solve_ivp(lambda t,rf: [(1-0.02*rf[1])*rf[0], (-1+0.03*rf[0])*rf
    [1]],[0,20],[20,20],method='RK45',max_step=.1)
```

The outputs R and F are plotted against time in Figure 2a and against eachother in Figure 2b. The solution is periodic which is seen in the closed contour of the phase portrait. We run the solver from time 0 to 20 which covers about three cycles.



(a) solid: R, dashed: F



(b) F vs. R

Figure 2: output of RK45 solver on give IVP