AMATH 586 Assignment 3

Tyler Chen

Problem 1

An ODE system, $\mathbf{u}'(t) = \mathbf{f}(\mathbf{u}(t), t)$, $t \geq 0$, is said to be *monotone* (with respect to the Euclidean inner product) if the real-valued function $\mathbf{f}$ satisfies

$$\langle (\mathbf{u} - \mathbf{v}), \mathbf{f}(\mathbf{u}, t) - \mathbf{f}(\mathbf{v}, t) \rangle \leq 0$$

for all $t \geq 0$ and all vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$.

(a) Show that if $\mathbf{u}(t)$ and $\mathbf{v}(t)$ are two real solutions to a monotone ODE system with $\mathbf{u}(0) = \mathbf{u_0}$ and $\mathbf{v}(0) = \mathbf{v_0}$, then $\|\mathbf{u}(t) - \mathbf{v}(t)\|$ decreases (weakly) monotonically for $t \geq 0$. [Hint: Look at $\frac{d}{dt}(\|\mathbf{u}(t) - \mathbf{v}(t)\|^2) = \frac{d}{dt}\langle \mathbf{u}(t) - \mathbf{v}(t), \mathbf{u}(t) - \mathbf{v}(t) \rangle$.]

(b) Consider the linear ODE with variable coefficients

$$\mathbf{u}' = A(t)\mathbf{u}, \quad t \geq 0,$$

where A is a real d by d matrix function of t. Prove that this ODE system is monotone if and only if all of the eigenvalues of the symmetric matrix $B(t) = \frac{1}{2}[A(t) + A^T(t)]$ are non-positive.

(c) Suppose the backward Euler method is applied to the ODE system in part (b). Let $\{\mathbf{U}^n\}_{n=0}^{n=N}$ be the sequence of vectors generated using initial value $\mathbf{U}^0$, and let $\{\mathbf{V}^n\}_{n=0}^{n=N}$ be the sequence of vectors generated using initial value $\mathbf{V}^0$ (and the same stepsize k). What conditions must k satisfy to ensure that $\|\mathbf{U}^n - \mathbf{V}^n\|$ is a decreasing function of $n \geq 0$?

Solution

(a) First observe that if $g(t) = (g_1(t), \ldots, g_d(t))$,

$$\frac{\mathrm{d}}{\mathrm{d}t} \langle g(t), g(t) \rangle = \frac{\mathrm{d}}{\mathrm{d}t} \left(g_1(t)^2 + \cdots + g_d(t)^2 \right) = 2g_1(t)g_1'(t) + \cdots 2g_d(t)g_d'(t) = 2 \langle g(t), g'(t) \rangle$$

Suppose $u(t)$ and $v(t)$ are real solutions to a monotone ODE system. Then, since $(u(t) - v(t))' = u'(t) - v'(t) = f(u,t) - f(v,t)$,

$$\frac{\mathrm{d}}{\mathrm{d}t} \|u(t) - v(t)\|^2 = \frac{\mathrm{d}}{\mathrm{d}t} \langle u(t) - v(t), u(t) - v(t) \rangle = 2 \langle u(t) - v(t), f(u,t) - f(v,t) \rangle \leq 0$$

Therefore $\|u(t) - v(t)\|^2$ is monotonically decreasing on $[0, \infty)$. As $\|u(t) - v(t)\| \geq 0$ on this interval, we also have that $\|u(t) - v(t)\|$ is monotonically decreasing on $[0, \infty)$. □

(b) Observe, that if $z, A(t)$ are real that $A^H(t) = A^T(t)$ and $\langle z, A(t)z \rangle$ is real. Therefore,

$$\langle z, A(t)z \rangle = \left\langle A^H(t)z, z \right\rangle = \overline{\langle z, A^H(t)z \rangle} = \left\langle z, A^T(t)z \right\rangle$$

Using the above equality it is clear that,

$$\langle z, B(t)z \rangle = \left\langle z, \frac{1}{2}(A(t) + A^T(t))z \right\rangle = \frac{1}{2} \langle z, A(t)z \rangle + \frac{1}{2} \left\langle z, A^T(t)z \right\rangle = \langle z, A(t)z \rangle$$

Let $v, w \in \mathbb{R}^d$ and write $z = v - w$. Since $A(t)$ is linear,

$$\langle u - v, A(t)(u - v)\rangle = \langle z, A(t)z\rangle = z^T A(t) z$$

Finally note that the eigenvalues of $B(t)$ are all non-positive if and only if for all z, $\langle z, B(t)z\rangle \leq 0$ (i.e. $B(t)$ is negative semi-definite).

This proves, the ODE system is monotone ($\langle u - v, A(t)(u - v)\rangle = z^T B(t) z \leq 0$ for all u, v) if and only all of the eigenvalues of $B(t)$ are non-positive. □

(c) Backwards Euler gives,

$$(U^{n+1} - U^n)/k = A(t)U^{n+1} \iff U^{n+1} = (I - kA(t))^{-1} U^n = \left((I - kA(t))^{-1}\right)^{n+1} U^0$$
$$(V^{n+1} - V^n)/k = A(t)V^{n+1} \iff V^{n+1} = (I - kA(t))^{-1} V^n = \left((I - kA(t))^{-1}\right)^{n+1} V^0$$

Therefore,

$$\|U^n - V^n\| = \left\|\left((I - kA(t))^{-1}\right)^n (U^0 - V^0)\right\| \leq \left\|(I - kA(t))^{-1}\right\|^n \left\|(U^0 - V^0)\right\|$$

We should then pick k such that $I - kA(t)$ is invertible and,

$$\left\|(I - kA(t))^{-1}\right\| < 1$$

Clearly a necessary condition for this is that $|1 - k\lambda_i| < 1$ for $i = 1, \ldots, d$. (If $A(t)$ is symmetric then this condition is also sufficient.)

Problem 2

Consider the midpoint method $U^{n+1} = U^{n-1} + 2kf(U^n)$ applied to the test problem $u' = \lambda u$. The method is zero-stable and second order accurate, and hence convergent. For $\lambda < 0$ and any $k > 0$, however, the point $z = k\lambda$ is *never* in the region of absolute stability of this method (see Example 7.7). Thus the numerical solution will grow exponentially over time. Yet the true solution $u(t) = e^{\lambda t}u(0)$ decays exponentially over time. How can numerical solutions that grow exponentially over time converge to something that decays exponentially over time?

(a) To try and resolve this seeming paradox, let $U^0 = \eta$, use forward Euler to generate U^1, and then use the midpoint method for $n = 2, 3, \ldots$. Find an analytic expression for U^n by solving the linear difference equation. What happens to this expression as $k \to 0$?

(b) Devise some numerical experiments to illustrate the resolution of the paradox.

Solution

(a) Using forward Euler we have,

$$U^1 = U^0 + kf(U^0) = U^0 + k\lambda U^0 = (1 + k\lambda)\eta$$

Using the midpoint method for $n = 1, 2, \ldots$ we have difference equation,

$$U^{n+1} = U^{n-1} + 2kf(U^n) = U^{n-1} + 2k\lambda U^n$$

Equivalently, for $n = 0, 1, \ldots$

$$U^{n+2} - 2k\lambda U^n - U^n = 0, \qquad U^0 = \eta, \qquad U^1 = (1 + k\lambda)\eta$$

This has characteristic polynomial,

$$\chi(x) = x^2 - 2k\lambda x - 1 = \left(x - k\lambda + \sqrt{1 + k^2\lambda^2}\right)\left(x - k\lambda - \sqrt{1 + k^2\lambda^2}\right)$$

We therefore have general solution to the difference equations,

$$U^n = c_1\left(k\lambda - \sqrt{1 + k^2\lambda^2}\right)^n + c_2\left(k\lambda + \sqrt{1 + k^2\lambda^2}\right)^n$$

Using the initial conditions we have,

$$\eta = c_1 + c_2, \qquad (1 + k\lambda)\eta = c_1\left(k\lambda - \sqrt{1 + k^2\lambda^2}\right) + c_2\left(k\lambda + \sqrt{1 + k^2\lambda^2}\right)$$

This means,

$$c_1 = \frac{\eta\left(\sqrt{k^2\lambda^2 + 1} - 1\right)}{2\sqrt{k^2\lambda^2 + 1}}, \qquad c_2 = \frac{\eta\left(\sqrt{k^2\lambda^2 + 1} + 1\right)}{2\sqrt{k^2\lambda^2 + 1}}$$

Therefore,

$$U^n = \left(\frac{\eta\left(\sqrt{k^2\lambda^2 + 1} - 1\right)}{2\sqrt{k^2\lambda^2 + 1}}\right)\left(k\lambda - \sqrt{1 + k^2\lambda^2}\right)^n + \left(\frac{\eta\left(\sqrt{k^2\lambda^2 + 1} + 1\right)}{2\sqrt{k^2\lambda^2 + 1}}\right)\left(k\lambda + \sqrt{1 + k^2\lambda^2}\right)^n$$

Finally,

$$\lim_{k\to 0} U^n = \frac{\eta(1-1)}{2}(1) + \frac{\eta(1+1)}{2}(1) = \eta$$

To explain the "paradox" consider the expression for U^n for k sufficiently small. In this limit we can take all $\mathcal{O}(k^2)$ terms to be zero. Then,

$$U^n = \frac{\eta(1-1)}{2})(k\lambda - 1)^n + \frac{\eta(1+1)}{2}(1+k\lambda)^n = \eta(1+k\lambda)^n$$

Now, with $t = nk$ we have,

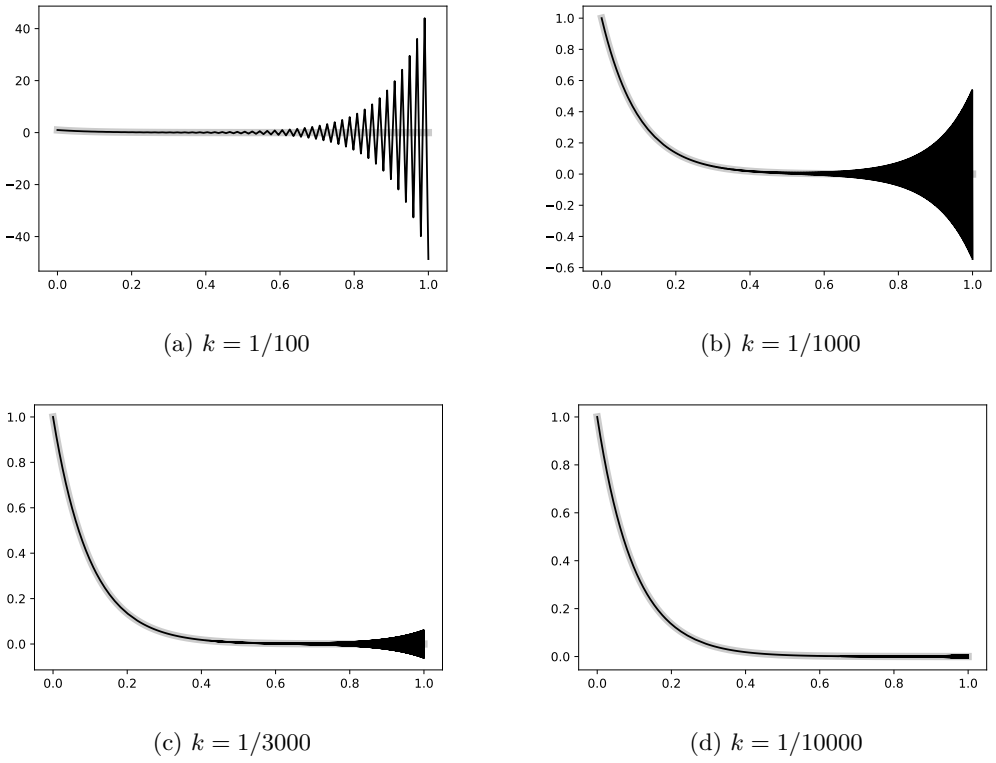
$$u(t) = \eta e^{t\lambda} = \eta e^{nk\lambda} = \eta\left(e^{k\lambda}\right)^n = \eta(1 + k\lambda + \mathcal{O}(k^2))^n$$

These agree with one another for small enough k.

(b) We take $\eta = 1$, $\lambda = -10$, and solve on $[0, 1]$ using the midpoint rule. We implement this in Python,

```
def mp(f,u0,u1,N,k):
    u = np.zeros(N)
    u[0] = u0
    u[1] = u1
    for n in np.arange(1,N-1):
        u[n+1] = u[n-1] + 2*k*f(u[n])
    return u
```

Figure 1 shows the results at various values of k.



(a) $k = 1/100$

(b) $k = 1/1000$

(c) $k = 1/3000$

(d) $k = 1/10000$

Figure 1: Midpoint method at various mesh sizes. Actual solution in grey.

Clearly the solution converges on our fixed, finite interval. However, it is also apparent that the solution will blow up if plotted on a larger interval (see analytic expression for U^n which diverges as $n \to \infty$). This is exactly in agreement with our definition for convergence.

Problem 3

Use the boundary locus method described in Secs. 7.6.1-7.6.2 to plot the region of absolute stability for the TR-BDF2 method (8.6). Observe that the method is A-stable and show that it is also L-stable.

Solution

Recall the TR-BDF2 method,

$$U^{n+1} = \frac{1}{3}\left(4U^* - U^n + kf(U^{n+1})\right), \qquad U^* = U^n + \frac{k}{4}\left(f(U^n) + f(U^*)\right)$$

Consider this method applied to the test equation $u' = \lambda u$. Then,

$$U^* = U^n + \frac{k}{4}(\lambda U^n + \lambda U^*) = U^n + \frac{k\lambda}{4}(U^n + U^*) \qquad \iff \qquad U^* = \left(\frac{1 + k\lambda/4}{1 - k\lambda/4}\right) U^n$$

Therefore,

$$U^{n+1} = \frac{1}{3}\left(4\left(\frac{1 + k\lambda/4}{1 - k\lambda/4}\right) U^n - U^n + k\lambda U^{n+1}\right)$$

Writing $z = k\lambda$ we have,

$$U^{n+1}\left(1 - \frac{z}{3}\right) = \frac{1}{3}\left(4\left(\frac{1 + z/4}{1 - z/4}\right) - 1\right) U^n$$

Equivalently,

$$U^{n+1} = R(z)U^n, \qquad R(z) = \frac{12 + 5z}{12 - 7z + z^2}$$

The region of stability is,

$$\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\}$$

To find the boundary of $\mathcal{S}$ we set $R(z) = e^{i\theta}$,

$$\begin{aligned} & 12 + 5z = e^{i\theta}\left(12 - 7z + z^2\right) = 12e^{i\theta} - 7e^{i\theta}z + e^{i\theta}z^2 \\ \iff \quad & e^{i\theta}z^2 - (7e^{i\theta} + 5)z + 12e^{i\theta} - 12 = 0 \\ \iff \quad & z^2 - (7 + 5e^{-i\theta})z + 12(1 - e^{-i\theta}) = 0 \end{aligned}$$

This has solutions,

$$z = \frac{(7 + 5e^{-i\theta}) \pm \sqrt{(7 + 5e^{-i\theta})^2 - 4(12(1 - e^{-i\theta}))}}{2}$$

Figure 2 shows the contour plot of $R(z)$ along with points of the form above evaluated at 30 values of θ equally spaced from 0 to 2π. Since $R(z) \to 0$ as $z \to \infty$ we easily verify that $\mathcal{S}$ is the region on the outside of this boundary.

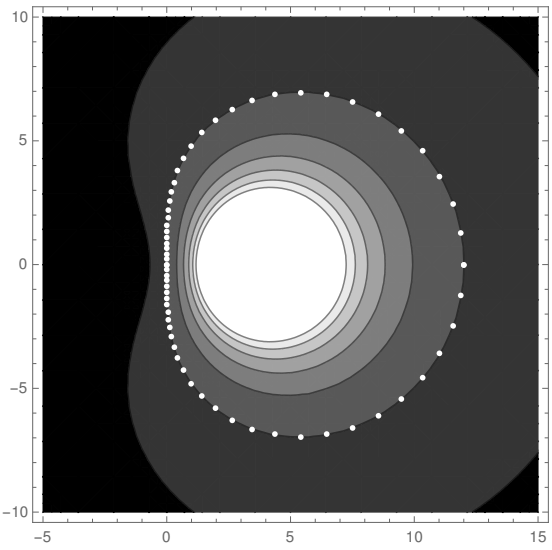


Figure 2: Contour plot of $R(z)$ and points evaluated on the boundary (white)

Clearly the leftmost point of the boundary is $z = 0$. Since $R(0) = 1$, the entire left half plane is contained in the region of stability. This means the method is A-stable.

Clearly $|R(z)| \to 0$ as $z \to \infty$ as the degree of the denominator of $R(z)$ is larger than the degree of the numerator. This, along with A-stability mean the method is also L-stable. □

Problem 4

Consider the A-stable trapezoidal method applied to the test problem $u' = \lambda u$:

$$U^{n+1} = U^n + \frac{k}{2}\lambda[U^n + U^{n+1}].$$

Of course, one can easily solve this linear equation for U^{n+1}, but suppose we used *fixed point iteration* to solve for U^{n+1}: Taking $w^{(0)}$ as an initial guess for U^{n+1}, set

$$w^{(j+1)} = U^n + \frac{k}{2}\lambda[U^n + w^{(j)}], \quad j = 0, 1, \ldots.$$

What conditions must $k\lambda$ satisfy so that $w^{(j)}$ will converge to U^{n+1}? [Note that if we use the trapezoidal rule but solve for U^{n+1} using fixed point iteration, we lose the benefits of A-stability!]

Solution

We write the iteration in the form,

$$w^{(j+1)} = U^n + k\lambda(U^n + w^{(j)})/2$$

Observe then that,

$$\left\|w^{(j+1)} - U^{n+1}\right\| = \left\|(U^n + k\lambda(U^n + w^{(j)})/2) - (U^n + k\lambda(U^n + U^{n+1})/2)\right\| = \frac{|k\lambda|}{2}\left\|w^{(j)} - U^{n+1}\right\|$$

Therefore,

$$\left\|w^{(j+1)} - U^{n+1}\right\| = \left(\frac{|k\lambda|}{2}\right)^{j+1}\left\|w^{(0)} - U^{n+1}\right\|$$

This proves that fixed point iteration will converge if and only if $k\lambda < 2$. □

Problem 5

The following simple model describes the switching behavior for a muscle that controls a valve in the heart. Let $x(t)$ denote the position of the muscle at time t and let $\alpha(t)$ denote the concentration at time t of a chemical stimulus. Suppose that the dynamics of x and α are controlled by the system of differential equations

$$\begin{aligned} \frac{dx}{dt} &= -\frac{x^3}{3} + x + \alpha \\ \frac{d\alpha}{dt} &= -\epsilon x. \end{aligned}$$

Here $\epsilon > 0$ is a parameter; its inverse estimates roughly the time that x spends near one of its rest positions.

(a) Taking $\epsilon = 1/100$, $x(0) = 2$, and $\alpha(0) = 2/3$, solve this system of differential equations using an explicit method of your choice. Integrate out to, say, $T = 400$, and turn in plots of $x(t)$ and $\alpha(t)$. Explain why you chose the method that you used and approximately how accurate you think your computed solution is and why. Comment on whether you seemed to need restrictions on the step size for stability or whether accuracy was the only consideration in choosing your step size.

(b) Solve the same problem using the backward Euler method and solving the nonlinear equations at each step via Newton's method. Write down the Jacobian matrix for the system and explain what initial guess you will use. Were you able to use a larger time step using the backward Euler method than you were with the explicit method used in part (a)?

Solution

(a) We implement forward Euler and the standard fourth order Runge-Kutta methods in Python:

```
def fw_euler(f,u0,N,T):
    U = np.zeros((len(u0),N))
    U[:,0] = u0
    for i in range(N-1):
        U[:,i+1] = U[:,i] + k*f(U[:,i])
    return U
```

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

We pick these methods since they are the standard benchmarks for explicit one step methods. Our solutions do not really seem very accurate until at least $N = 600$ at this point they seem to resemble the actual solution.

(b) Write,

$$f\left(\begin{bmatrix} x \\ \alpha \end{bmatrix}\right) = \begin{bmatrix} -x^3/3 + x + \alpha \\ -\epsilon x \end{bmatrix}$$

Backwards Euler updates U^{n+1} by solving,

$$(U^{n+1} - U^n)/k = f(U^{n+1})$$

The Jacobian of $f : \mathbb{R}^2 \to \mathbb{R}^2$ is,

$$J_f(u) = \begin{bmatrix} -x^2 + 1 & 1 \\ -\epsilon & 0 \end{bmatrix}$$

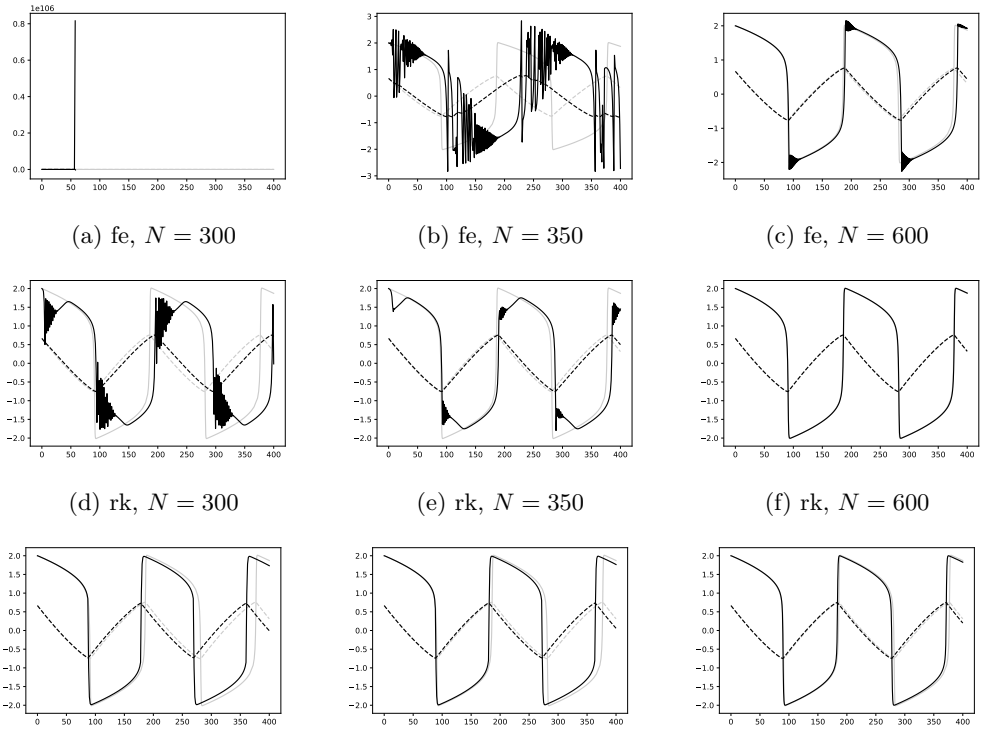
We apply Newton's method to solve for U^{n+1}. At each step we compute,

$$w^{(j+1)} = w^{(j)} - \Delta w^{(j)}, \qquad (I - kJ_f(w^{(j)}))\Delta w^{(j)} = w^{(j)} - kf(w^{(j)}) - U^n$$

We will take $w^{(0)} = U^n + kf(U^n)$, the value forward Euler would pick for U^{n+1} given current value U^n. We iterate until the difference is around machine precision, or until we have reached our maximum iteration count (in our tests 100).

We implement this in Python:

```
def bw_euler(f,u0,N,T,Jf,tol,max_iter):
    U = np.zeros((2,N))
    U[:,0] = u0
    for n in range(N-1):
        # run Newton's method with initial guess from foward Euler
        U[:,n+1] = U[:,n] + k*f(U[:,n])
        for i in range(max_iter):
            dw = np.linalg.solve(np.identity(2) - k*Jf(U[:,n+1]),U[:,n
                +1]-k*f(U[:,n+1])-U[:,n])
            if np.linalg.norm(dw) < tol:
                break
            U[:,n+1] -= dw
        print("max iteration reached, Newton's method did not converge
            to specified tolerance")
    return U
```



(a) fe, $N = 300$ (b) fe, $N = 350$ (c) fe, $N = 600$

(d) rk, $N = 300$ (e) rk, $N = 350$ (f) rk, $N = 600$

(g) be, $N = 300$ (h) be, $N = 350$ (i) be, $N = 600$

Figure 3: Numerical solutions of $x(t)$ (solid), $\alpha(t)$ (dashed) on time interval $[0, 400]$ with output of Python's `solve_ivp` (with tolerance of 10^{-10}) shown in grey

While fewer mesh points are required for a reasonable solution, unless we iterate Newton's method for a long time, the solution does not always converge. When we allow Newton's method to converge more fully we can take larger steps and still have convergence. Sample outputs are shown in Figure 3.