AM205 HW3. Numerical calculus. Solution

P1. Adaptive integration

(a) The third Legendre polynomial is $P_3(x) = \frac{1}{2}x(5x^2 - 3)$, which has roots at

$$x_1 = -\sqrt{\frac{3}{5}}, \qquad x_2 = 0, \qquad x_3 = \sqrt{\frac{3}{5}}$$

We now solve for the weights by integrating the Lagrange interpolating polynomials through these three points. We have

$$\begin{aligned} w_0 &= \int_{-1}^{1} L_0(x)dx = \int_{-1}^{1} \frac{x-0}{-\sqrt{3/5}-0} \times \frac{x-\sqrt{3/5}}{-\sqrt{3/5}-\sqrt{3/5}}dx \\ &= \frac{5}{6}\int_{-1}^{1}(x^2 - \sqrt{3/5}x)dx = \frac{5}{6}\int_{-1}^{1}(x^2)dx \\ &= \frac{5}{6}\frac{2}{3} = \frac{5}{9} \end{aligned}$$

Similarly, we can do the integration for w_1 to obtain

$$\begin{aligned} w_1 &= \int_{-1}^{1} L_1(x)dx = \int_{-1}^{1} \frac{x+\sqrt{3/5}}{+\sqrt{3/5}} \times \frac{x-\sqrt{3/5}}{-\sqrt{3/5}}dx \\ &= -\frac{5}{3}\int_{-1}^{1}(x^2 - \frac{3}{5})dx = -\frac{5}{3}(\frac{2}{3} - \frac{6}{5}) \\ &= -\frac{5}{3}(-\frac{8}{15}) = \frac{8}{9} \end{aligned}$$

Finally, w_2 is given by

$$\begin{aligned} w_2 &= \int_{-1}^{1} L_2(x)dx = \int_{-1}^{1} \frac{x-0}{\sqrt{3/5}-0} \times \frac{x+\sqrt{3/5}}{\sqrt{3/5}+\sqrt{3/5}}dx \\ &= \frac{5}{6}\int_{-1}^{1}(x^2 + \sqrt{3/5}x)dx = \frac{5}{6}\int_{-1}^{1}(x^2)dx \\ &= \frac{5}{6}\frac{2}{3} = \frac{5}{9} \end{aligned}$$

and therefore $[w_0, w_1, w_2] = [\frac{5}{9}, \frac{8}{9}, \frac{5}{9}]$

We now show that this quadrature rule integrates polynomials of up to degree 5 exactly. We show this property on $[-1, 1]$, and if it holds there, it holds on any arbitrary interval after an affine transformation. An arbitrary fifth-order polynomial can be written as

$$p_5(x) = a + bx + cx^2 + dx^3 + ex^4 + fx^5$$

We can integrate $p_5(x)$ on $[-1, 1]$ and obtain

$$\int_{-1}^{1} p_5(x)dx = a\int_{-1}^{1} dx + b\int_{-1}^{1} xdx + c\int_{-1}^{1} x^2dx + d\int_{-1}^{1} x^3dx + e\int_{-1}^{1} x^4dx + f\int_{-1}^{1} x^5dx$$

We can evaluate the simple integrals and arrive at

$$\int_{-1}^{1} p_5(x)dx = 2a + \frac{2c}{3} + \frac{2e}{5}$$

Evaluating the integral using the weights and quadrature points, we get

$$\int_{-1}^{1} p_5(x)dx = \frac{5}{9}(a - (\frac{3}{5})^{1/2}b + \frac{3}{5}c - (\frac{3}{5})^{3/2}d + \frac{9}{25}e - (\frac{3}{5})^{5/2}f) + \frac{8}{9}(a) + \frac{5}{9}(a + (\frac{3}{5})^{1/2}b + \frac{3}{5}c + (\frac{3}{5})^{3/2}d + \frac{9}{25}e + (\frac{3}{5})^{5/2}f)$$

We can sum the fractions and obtain

$$\int_{-1}^{1} p_5(x)dx = \frac{8}{9}a + 2 \times \frac{5}{9}(a + \frac{3}{5}c + \frac{9}{25}e) = 2a + \frac{2}{3}c + \frac{2}{5}e$$

Hence, the expressions from using quadrature points and weights and from direct integration are identical, so the method is correct for all polynomials up to the fifth order.

(b) See solution code in [p1_adaptive_integration.py]. We implement the adaptive scheme using the 3-point Gauss quadrature as discussed in part **(a)**. For the integrals of the form

$$\int_{-1}^{7/2} (-x^m - (x+3)^2 + 4)dx$$

the computed integral values, estimated error, and number of intervals are given below

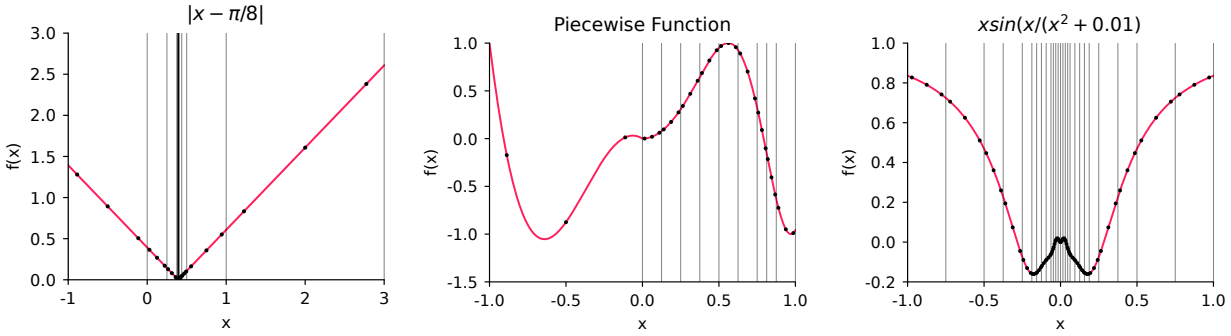
m	**Integral value**	**Estimated error**	**Num. of intervals**
4	-176.119	0	1
5	-377.086	0	1
6	-990.151	7.83×10^{-7}	16
7	-2885.59	7.00×10^{-7}	27
8	-8828.28	1.33×10^{-6}	29

As expected from part **(a)**, the integnrals for $m \leq 5$ are computed exactly with a single integration step. The integrals for $m \geq 6$ are determined to high accuracy with a small number of subdivisions of the interval.

(c) We use the same integration routine to compute the integrals for this part. The answers are given below

Integral	Integral value	Estimated error	Num. of intervals
$\int_{-1}^{3} \lvert x - \frac{\pi}{8} \rvert dx$	4.37	0	16
$\int_{-1}^{1} g(x)dx$	-0.17	1.11×10^{-7}	10
$\int_{-1}^{1} x \sin(\frac{x}{x^2+0.01})dx$	0.76	7.24×10^{-7}	26

Plots of the functions with the quadrature points and intervals are shown below.



P2. Finite differences and least squares

(a) First, we evaluate the Taylor expansion around x at five points as suggested

$$f(x-2h) = f(x) - 2hf'(x) + \frac{4h^2 f^{(2)}(x)}{2} - \frac{8h^3 f^{(3)}(x)}{6} + \frac{16h^4 f^{(4)}(x)}{24} - \frac{32h^5 f^{(5)}(x)}{120} + \cdots$$
$$f(x-h) = f(x) - hf'(x) + \frac{h^2 f^{(2)}(x)}{2} - \frac{h^3 f^{(3)}(x)}{6} + \frac{h^4 f^{(4)}(x)}{24} - \frac{h^5 f^{(5)}(x)}{120} + \cdots$$
$$f(x+h) = f(x) + hf'(x) + \frac{h^2 f^{(2)}(x)}{2} + \frac{h^3 f^{(3)}(x)}{6} + \frac{h^4 f^{(4)}(x)}{24} + \frac{h^5 f^{(5)}(x)}{120} + \cdots$$
$$f(x+2h) = f(x) + 2hf'(x) + \frac{4h^2 f^{(2)}(x)}{2} + \frac{8h^3 f^{(3)}(x)}{6} + \frac{16h^4 f^{(4)}(x)}{24} + \frac{32h^5 f^{(5)}(x)}{120} + \cdots$$

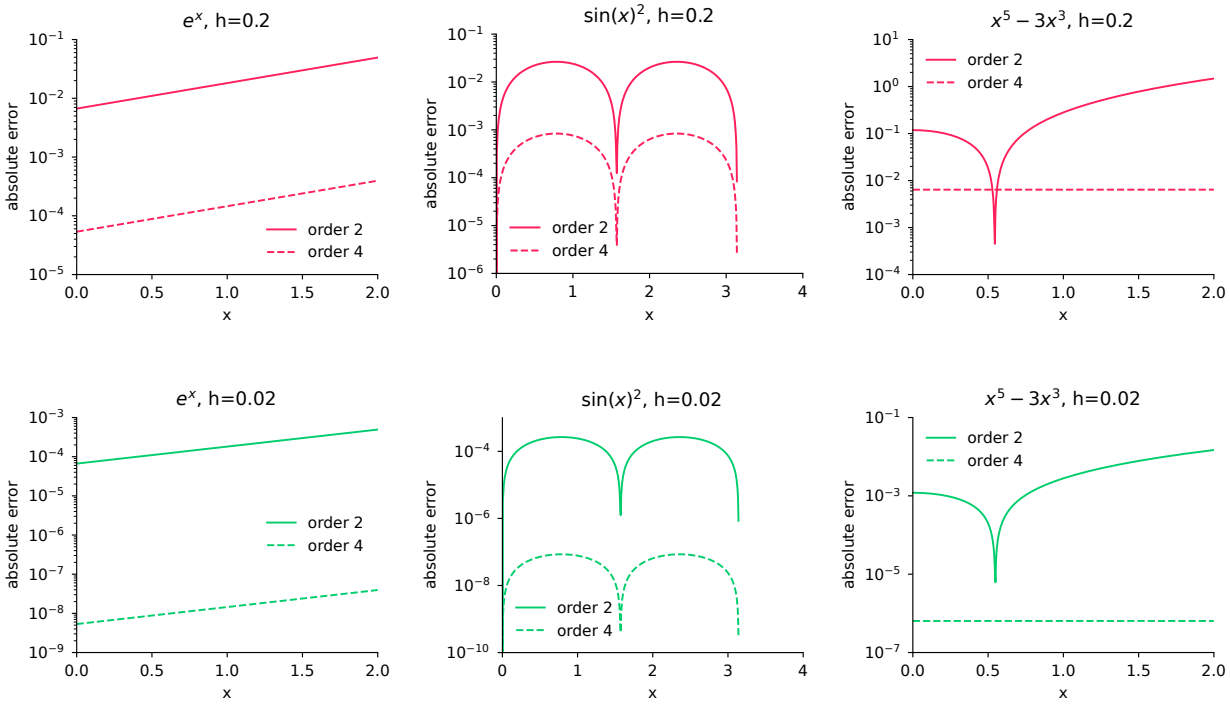
We then rearrange and to get the $\mathcal{O}(h^2)$, $\mathcal{O}(h^3)$, and $\mathcal{O}(h^4)$ terms to cancel:

$$8 \times (f(x+h) - f(x-h)) - (f(x+2h) - f(x-2h)) = 12hf'(x) - \frac{48h^5 f^{(5)}}{120}$$

So,

$$f'(x) = \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} + \mathcal{O}(h^4)$$

(b) We observe that despite the choice of h, the scaling between the errors of the two approximations remains the same.



(c) See solution code in [p2_finite_differences.py]. We aim to build an approximation to the first derivative $f'(0)$ of a function $f(x)$ given its values on a stencil of m points $x_i = t_i h$, where $t_i = (i - \frac{m-1}{2})$ for $i = 0, \ldots, m-1$. The points are centered at zero. Denote $\bar{f} = [f(x_0), \ldots, f(x_{m-1})]^T \in \mathbb{R}^m$. The approximation is based on a polynomial of degree p fitted to the function values. Define a polynomial of degree p with coefficients $b = [b_0, \ldots, b_p]^T \in \mathbb{R}^{p+1}$ as

$$g(x) = \sum_{j=0}^{p} b_j x^j.$$

Denote its values at points x_i as $\bar{g} = [g(x_0), \ldots, g(x_{m-1})]^T \in \mathbb{R}^m$. Then

$$\bar{g} = Vb$$

where $V \in \mathbb{R}^{m,p+1}$ is the Vandermonde matrix

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^p \\ 1 & x_1 & x_1^2 & \cdots & x_1^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m-1} & x_{m-1}^2 & \cdots & x_{m-1}^p \end{bmatrix}.$$

The coefficients b are obtained by least-squares fitting, i.e from minimizing the fitting error $\|\bar{g} - \bar{f}\|_2 = \|Vb - \bar{f}\|_2$, and therefore satisfy the normal equations $V^T V b = V^T \bar{f}$, for which

$$b = (V^T V)^{-1} V^T \bar{f}.$$

The approximation to first derivative $f'(0)$ is then given by

$$f'(0) \approx g'(0) = b_1 = e_1^T b = [e_1^T (V^T V)^{-1} V^T] \bar{f},$$

which is a linear combination of the function values and can be written as

$$f'(0) \approx \frac{w_0 f(x_0) + \ldots + w_{m-1} f(x_{m-1})}{h}$$

with $w = h e_1^T (V^T V)^{-1} V^T \in \mathbb{R}^m$. Note that w only depends on m and does not depend on h or $\bar{f}$, since the fitting is invariant to translation and scaling. The same approximation holds at an arbitrary point x

$$f'(x) \approx \frac{w_0 f(x - h\frac{m-1}{2}) + w_1 f(x - h\frac{m-1}{2} + h) + \ldots + w_{m-1} f(x + h\frac{m-1}{2})}{h}.$$

Now using this general procedure, we can compute the coefficients of various finite difference approximations.

(d) Below, we tabulate the computed coefficients w for all (m, p) pairs:

- $(m, p) = (3, 2), (5, 4)$

$$w_{3,2} = [-0.5, 0.0, 0.5]$$
$$w_{5,4} = [0.0833, -0.6667, 0.0, 0.6667, -0.0833]$$

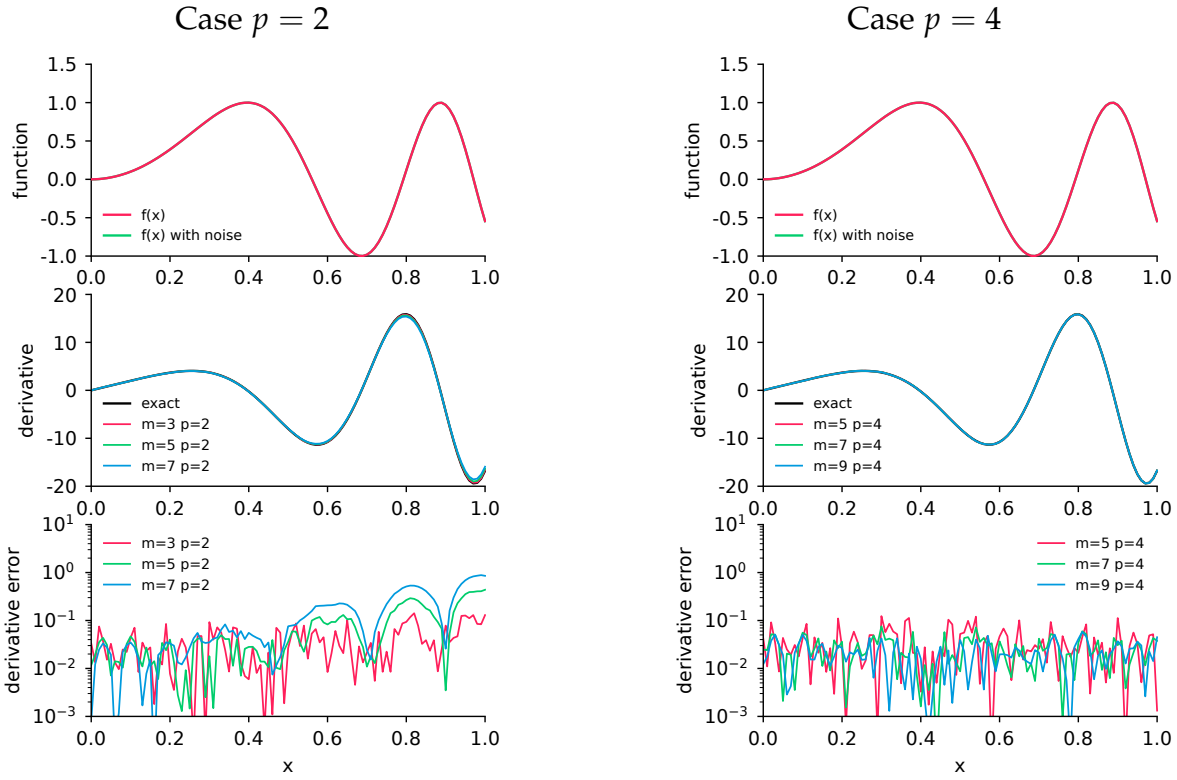
- $(m, p) = (5, 2), (7, 4)$

$$w_{5,2} = [-0.2, -0.1, 0.0, 0.1, 0.2]$$
$$w_{7,4} = [0.0873, -0.2659, -0.2302, 0.0, 0.2302, 0.2659, -0.0873]$$

- $(m, p) = (7, 2), (9, 4)$

$$w_{7,2} = [-0.1071, -0.0714, -0.0357, 0.0, 0.0357, 0.0714, 0.1071]$$
$$w_{9,4} = [0.0724, -0.1195, -0.1625, -0.1061, 0.0, 0.1061, 0.1625, 0.1195, -0.0724]$$

(e) Results:



Below are the RMS errors for all (m, p) pairs considered:

$$\begin{aligned}
\text{RMSE}_{3,2} &= 0.05437 \\
\text{RMSE}_{5,2} &= 0.1367 \\
\text{RMSE}_{7,2} &= 0.2761 \\
\text{RMSE}_{5,4} &= 0.04888 \\
\text{RMSE}_{7,4} &= 0.03130 \\
\text{RMSE}_{9,4} &= 0.02543
\end{aligned}$$

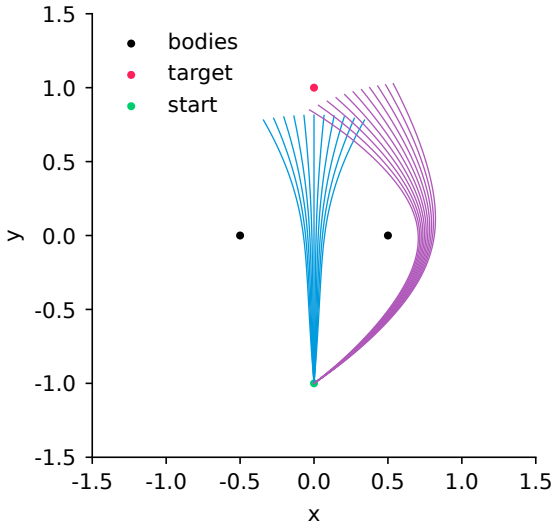
The approximation with the lowest error is given by $(p, m) = (4, 9)$. In general, we see that the error for the same stencil size is lower for larger p. This is intuitive as a quartic will better approximate our function than a quadratic. Interestingly, increasing the stencil size decreases the error for $p = 4$ but increases the error for $p = 2$. In part **(d)**, we highlighted how for $m > p + 1$, we observe a stronger smoothing effect on the stencil of m points. So, for $p = 2$, larger stencils are poorly fit by our polynomial, leading to low-quality weights for our approximation of $f'(x)$.

More specifically, we can think of our choice of m and p as affecting two different contributions to the error: (i) the truncation error and (ii) the error from the noise. For a larger stencil with $p = 2$, the increased truncation error dominates the reduction of the noise.

P3. Gravity assist

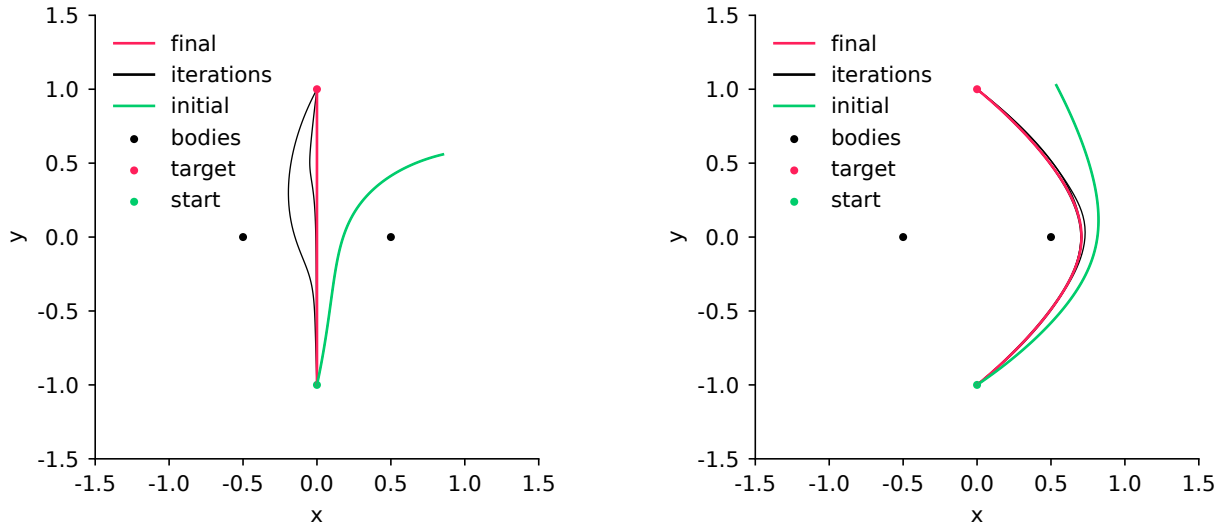
See solution code in [p3_gravity.py].

(a) Below are the solutions of the initial value problem.



(b) The reference implementation adapts the TDMA algorithm to 2×2 matrices.

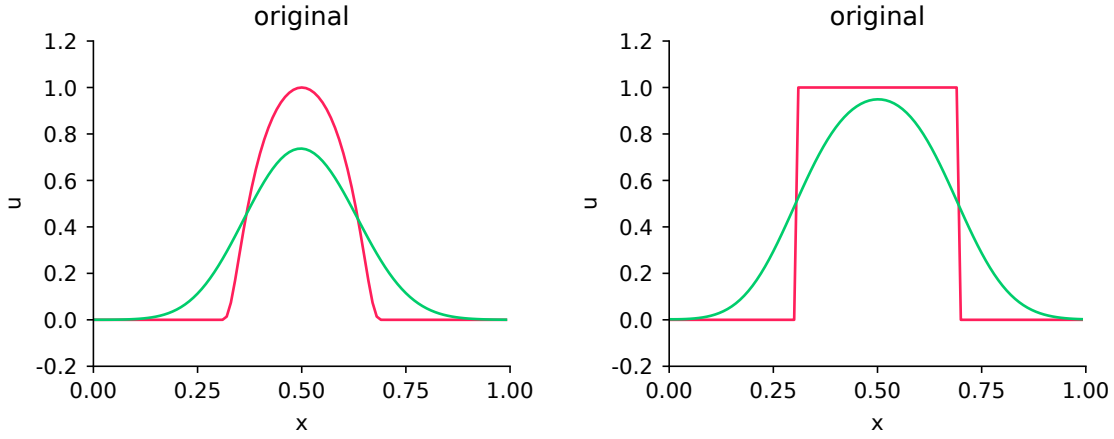
(c) Below are the solutions of the boundary value problem for two choices of the initial guess. In both cases, the iterations converge to a trajectory that satisfies the boundary conditions and the original nonlinear equation. The obtained trajectories qualitatively match those from part **(a)**. For the first case, the trajectory passes between the two planets, takes 6 iterations until convergence, and results in the initial velocity of $(2.5749 \cdot 10^{-18}, 1.4822)$. For the first case, the trajectory passes the two planets on the right, takes 8 iterations until convergence, and results in the initial velocity of $(1.5816, 1.3038)$.



P4. Advection and anti-diffusion [23 pts]

See solution code in [p4_advection.py].

(a) Below are the results for the original first-order upwind scheme. In both cases, we observe smearing of the solution. However, there is not additional oscillation or phase shift, i.e. the maximum remains at the expected position. The magnitude has decreased. This behavior is consistent with the truncation error analysis, which reveals an additional diffusion term added by the discretization.



(b) Taylor expansion of $u(x)$ about x_i gives

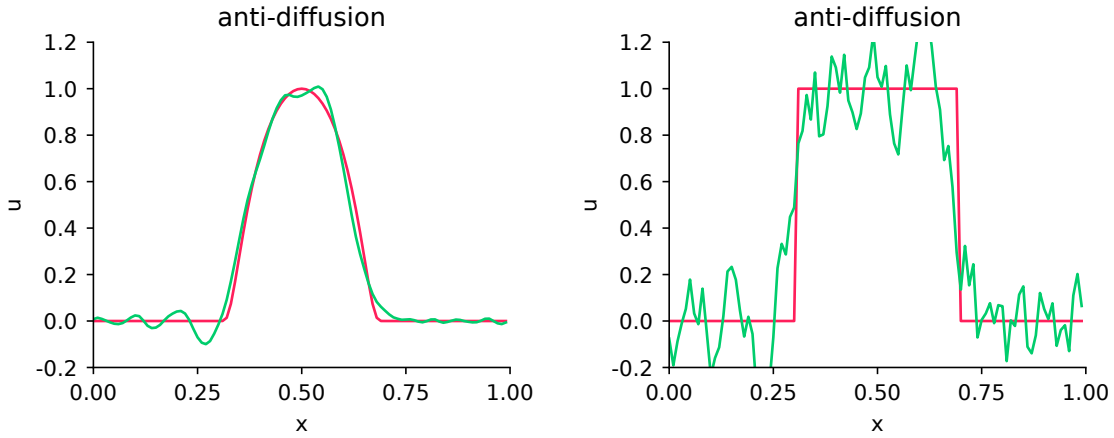
$$u\left(x_i - h\right) = u(x_i) - u_x(x_i)h + \frac{1}{2}u_{xx}(x_i)h^2 + \mathcal{O}(h^3). \tag{1}$$

Therefore, the backward difference evaluates to

$$\frac{u(x_i) - u(x_i - \Delta x)}{\Delta x} = u_x(x_i) - \frac{1}{2} h u_{xx}(x_i) + \mathcal{O}(h^2), \tag{2}$$

which results in $A = 0.5$.

Below are the results for the scheme with the anti-diffusion term added unconditionally. In fact, in this case we obtain the central difference scheme. We do not observe any smearing of the solution. For the smooth function, there is moderate oscillation, no phase shift, and no change of magnitude. For the step function, there is significant oscillation, no phase shift, and no change of magnitude, although the magnitude is harder to determine due to oscillation.

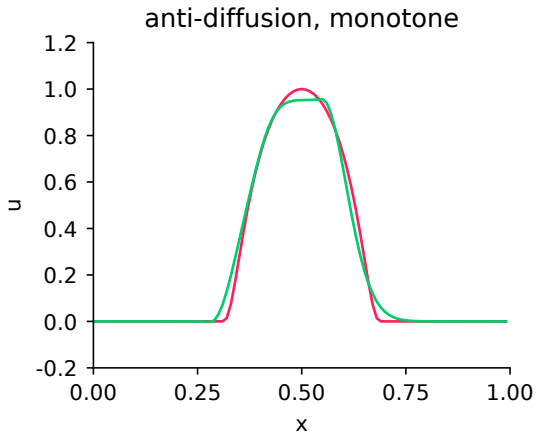


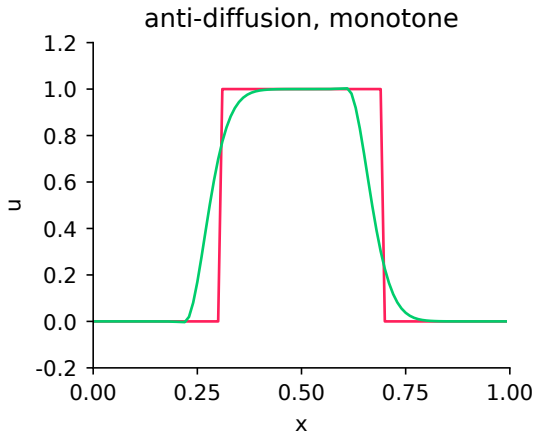
(c) Below are the results for the scheme with the anti-diffusion term added only in points where the solution is monotonic. As expected, this reduces the numerical diffusion without introducing oscillation. The magnitude has slightly decreased for the smooth function and has not changed for the step function. There is no significant phase shift.

Note that the function `scipy.integrate.odeint()` with default parameters may fail in this case, unable to choose a proper time step. Overriding the default parameters to set the time manually may solve the issue. For example, the following

```
scipy.integrate.odeint(rhs, u_init, tt, hmax=1e-3, atol=1e10)
```

should produce a correct solution. The function `scipy.integrate.solve_ivp()` gives a correct solution with default parameters.


anti-diffusion, monotone
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
u
0.00
0.25
0.50
0.75
1.00
x


anti-diffusion, monotone
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
u
0.00
0.25
0.50
0.75
1.00
x