Dan Card
dcard@umich.edu
Aerospace 523: Computational Fluid Dynamics
Homework: 3
Due: October 30th, 2020

1 A-Stable Backwards Difference

In the BDF methods, the time derivative is approximated using one-sided finite differences. The BDF2 method is A-stable, whereas BDF3 is not. Consider a multi-step method in which the time derivative is approximated by the average of the BDF2 and BDF3 time-derivative approximations:

$$\frac{du}{dt} = f \rightarrow \frac{1}{2}\frac{du}{dt}|_{\text{BDF2}} + \frac{1}{2}\frac{du}{dt}|_{\text{BDF3}} = f$$

a. Determine the coefficients α_k and β_k that define this method. What is its order of accuracy?

BDF2:

The expression for the derivative can be expressed as,

$$u_t = \frac{\frac{3}{2}u^{n+1} - 2u^n + \frac{1}{2}u^{n-1}}{\Delta t} = f^{n+1}$$

BDF3:

The expression for the derivative can be expressed as,

$$u_t = \frac{\frac{11}{6}u^{n+1} - 3u^n + \frac{3}{2}u^{n-1} - \frac{1}{3}u^{n-2}}{\Delta t} = f^{n+1}$$

Averaging BDF2 and BDF3 together will give the time-derivative approximation,

$$\Delta t f^{n+1} = \frac{1}{2}\underbrace{\left(\frac{3}{2}u^{n+1} - 2u^n + \frac{1}{2}u^{n-1}\right)}_{\text{BDF2}} + \frac{1}{2}\underbrace{\left(\frac{11}{6}u^{n+1} - 3u^n + \frac{3}{2}u^{n-1} - \frac{1}{3}u^{n-2}\right)}_{\text{BDF3}}$$

Combining like terms results in,

$$\frac{5}{3}u^{n+1} - \frac{5}{2}u^n + u^{n-1} - \frac{1}{6}u^{n-2} = \Delta t f^{n+1}$$

This results in the coefficients α and β to be,

$$\boxed{\alpha_1 = \frac{5}{3}, \quad \alpha_0 = -\frac{5}{2}, \quad \alpha_{-1} = 1, \quad \alpha_{-2} = -\frac{1}{6}, \quad \beta_1 = 1}$$

Order of Accuracy

Firstly, is to start with the Taylor-Series expansion expression,

$$u^{n+k} = u^n + (k\Delta t)u_t^n + \frac{1}{2}(k\Delta t)^2 u_{tt}^n + \frac{1}{6}(k\Delta t)^3 u_{ttt}^n + \frac{1}{24}(k\Delta t)^4 u_{t^{(4)}}^n + \ldots \mathcal{O}(\Delta t^5)$$

$$f^{n+k} = u_t^{n+k} = u_t^n + (k\Delta t)u_{tt}^n + \frac{1}{2}(k\Delta t)^2 u_{ttt}^n + \frac{1}{6}(k\Delta t)^3 u_{t^{(4)}} + \frac{1}{24}(k\Delta t)^4 u_{t^{(5)}}^n + \ldots \mathcal{O}(\Delta t^5)$$

Conducting Taylor-Expansions:

$$
\begin{aligned}
u^{n+1} &= u^n + \Delta t u_t^n + \frac{1}{2}\Delta t^2 u_{tt}^n + \frac{1}{6}\Delta t^3 u_{ttt}^n + \frac{1}{24}\Delta t^4 u_{t^{(4)}}^n + \ldots \mathcal{O}(\Delta t^5) \\
u^n &= u^n \\
u^{n-1} &= u^n - \Delta t u_t^n + \frac{1}{2}\Delta t^2 u_{tt}^n - \frac{1}{6}\Delta t^3 u_{ttt}^n + \frac{1}{24}\Delta t^4 u_{t^{(4)}}^n + \ldots \mathcal{O}(\Delta t^5) \\
u^{n-2} &= u^n - 2\Delta t u_t^n + 2\Delta t^2 u_{tt}^n - \frac{4}{3}\Delta t^3 u_{ttt}^n + \frac{2}{3}\Delta t^4 u_{t^{(4)}}^n + \ldots \mathcal{O}(\Delta t^5) \\
f^{n+1} &= u_t^n + \Delta t u_{tt}^n + \frac{1}{2}\Delta t^2 u_{ttt}^n + \frac{1}{6}\Delta t^3 u_{t^{(4)}}^n + \frac{1}{24}\Delta t^4 u_{t^{(5)}}^n + \mathcal{O}(\Delta t^5)
\end{aligned}
$$

Then for the order of accuracy the error gives,

$$
\begin{aligned}
\epsilon^{n+1} &= \frac{5}{3}u^{n+1} - \frac{5}{2}u^n + u^{n-1} - \frac{1}{6}u^{n-2} - \Delta t f^{n+1} \\
&= \frac{5}{3}\left(u^n + \Delta t u_t^n + \frac{1}{2}\Delta t^2 u_{tt}^n + \frac{1}{6}\Delta t^3 u_{ttt}^n + \frac{1}{24}\Delta t^4 u_{t^{(4)}}^n\right) + \ldots \\
&- \frac{5}{2}u^n + \ldots \\
&+ \left(u^n - \Delta t u_t^n + \frac{1}{2}\Delta t^2 u_{tt}^n - \frac{1}{6}\Delta t^3 u_{ttt}^n + \frac{1}{24}\Delta t^4 u_{t^{(4)}}^n\right) + \ldots \\
&- \frac{1}{6}\left(u^n - 2\Delta t u_t^n + 2\Delta t^2 u_{tt}^n - \frac{4}{3}\Delta t^3 u_{ttt}^n + \frac{2}{3}\Delta t^4 u_{t^{(4)}}^n\right) + \ldots \\
&- \Delta t\left(u_t^n + \Delta t u_{tt}^n + \frac{1}{2}\Delta t^2 u_{ttt}^n + \frac{1}{6}\Delta t^3 u_{t^{(4)}}^n + \frac{1}{24}\Delta t^4 u_{t^{(5)}}^n\right)
\end{aligned}
$$

Then using Matlab to simplify gives that the error is,

$$\epsilon^{n+1} = -\frac{1}{6}\Delta t^3 u_{ttt} - \frac{1}{6}\Delta t^4 u_{t^{(4)}} + \mathcal{O}(\Delta t^5)$$

Then from the leading term this gives that the convergence is,

$$|\epsilon^{n+1}| = \mathcal{O}(\Delta t^{p+1}) = \mathcal{O}(\Delta t^3)$$

This gives that the order of accuracy is,

$$\boxed{p = 2}$$

Since p = 2, the order of accuracy for this scheme is second-order accurate.

b. Perform an eigenvalue-stability analysis and *prove* (analytically) that this method is A-stable. Plot its stability boundary in the $\lambda\Delta t$ complex number plane, and overlay BDF2 and BDF3.

Starting with the expression for this averaged time-derivative,

$$\Delta t f^{n+1} = \frac{5}{3}u^{n+1} - \frac{5}{2}u^n + u^{n-1} - \frac{1}{6}u^{n-2}$$

Then from here substituting in $g^{n+k}u_0$ for u^{n+k} and $\lambda g^{n+k}u_0$ for f^{n+k},

$$\lambda\Delta t g^{n+1}u_0 = \frac{5}{3}g^{n+1}u_0 - \frac{5}{2}g^n u_0 + g^{n-1}u_0 - \frac{1}{6}g^{n-2}u_0$$

Then taking this expression and dividing by $g^n u_0$ results in,

$$\lambda\Delta t g = \frac{5}{3}g - \frac{5}{2} + g^{-1} - \frac{1}{6}g^{-2}$$

Isolating the $\lambda\Delta t$ term then results in,

$$\lambda\Delta t = \frac{5}{3} - \frac{5}{2}g^{-1} + g^{-2} - \frac{1}{6}g^{-3}$$

Further simplifications without loss of generality gives,

$$\lambda\Delta t = \frac{5}{3} + \frac{1}{6g^3}\left(-15g^2 + 6g - 1\right)$$

Then by definition, this scheme must be stable if the un-stable region (the regions *inside* the marked plots do not extend into the left-hand plan). In this limiting case, this can be re-written as the limit as $\lambda\Delta t \to 0^-$ and solve for the θ value at which this occurs,

$$\lim_{\lambda\Delta t\to 0} = 0 = \frac{5}{3} + \frac{1}{6g^3}\left(-15g^2 + 6g - 1\right)$$

Taking this further, isolating and solving for g gives

$$\lim_{\lambda\Delta t\to 0} = -10g^3 = -15g^2 + 6g - 1$$

Pulling all g terms to one side results in,

$$0 = 10g^3 - 15g^2 + 6g - 1$$

Conducting simple factorization gives,

$$0 = (g-1)\left(10g^2 - 5g + 1\right)$$

Using quadratic formula this gives that g is equivalent to,

$$g = 1, \frac{5 \pm i\sqrt{15}}{20}$$

Solving for the values at which these occurs gives,

$$g = \frac{5 \pm i\sqrt{15}}{20} = \exp[i\theta] = \cos\theta + i\sin\theta$$
$$\theta = 1.1513i \pm 0.6591$$

Again for $g = 1$,

$$g = 1 = \exp[i\theta] = \cos\theta + i\sin\theta$$
$$\theta = 0, \quad \textbf{Physical answer}$$

As shown above, there are three approximated answers in which this averaged time-derivative scheme will cross into the unstable region. However, two of these three are not physical answers as $\theta \in [\mathbf{0}, \mathbf{2}\pi] \mid \theta \in \mathbb{R}$ $\therefore$ the limiting case occurs at $\theta = \mathbf{0}$ where $g = 1$ resulting in $\lambda\mathbf{\Delta t} = \frac{\mathbf{5}}{\mathbf{3}} - \frac{\mathbf{5}}{\mathbf{3}} = \mathbf{0}$ − resulting in an <u>A-stable scheme.</u>

Plotting these eigenvalue stability regions can show and confirm that $\theta = 0$ is the limiting case and for the averaged time-derivative that it is indeed A-stable as the unstable region never crosses into the left-hand plane like in BDF3 scheme. Plotting the un-stable regions gives Figure 1 shown below,

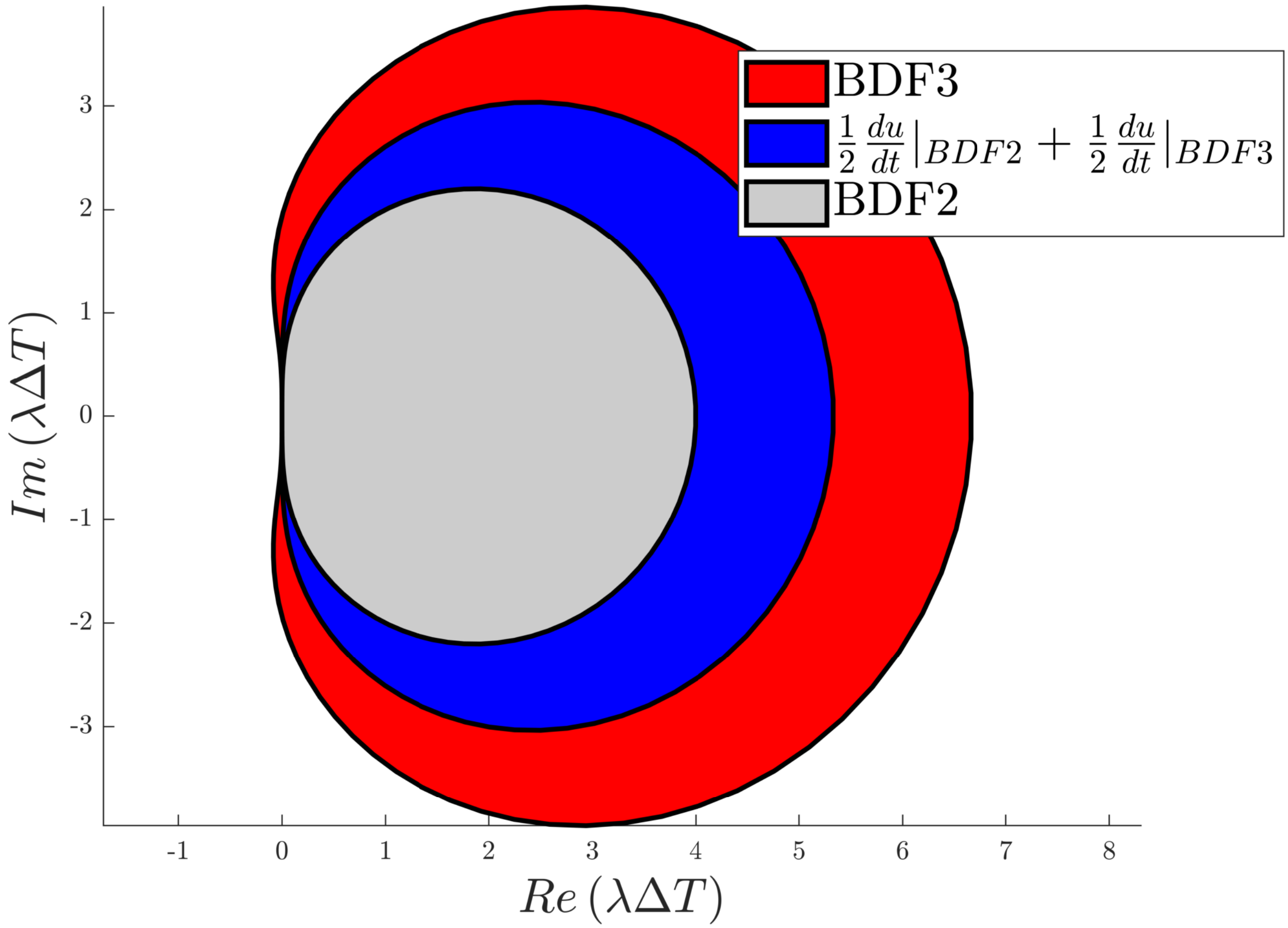

Figure 1: Eigenvalue stability region for BDF2, BDF3, and the averaged time-derivative approximation.

Shown above in Figure 1 are the un-stable regions for BDF2, BDF3, and the averaged time-derivative of the two. Outside of these regions the schemes remain stable where the left-hand plane where Re($\lambda\mathbf{\Delta T}$) < 0 is the A-stable region of the schemes.

c. Calculate the temporal truncation error of this method, $\tau = \text{LHS} - \text{RHS}$ of the multistep formula, and show that the leading term is half the magnitude of that of BDF2.

From part a. of this question, I found that the local error was,

$$\epsilon^{n+1} = -\frac{1}{6}\Delta t^3 u_{ttt} - \frac{1}{6}\Delta t^4 u_{t^{(4)}} + \frac{1}{40}\Delta t^5 u_{t^{(5)}}$$

Thus, the truncation error is

$$\boxed{\epsilon^{n+1} = \underbrace{-\frac{1}{6}\Delta t^3 u_{ttt} - \frac{1}{6}\Delta t^4 u_{t^{(4)}} + \frac{1}{40}\Delta t^5 u_{t^{(5)}} + \ldots \mathcal{O}(\Delta t^6)}_{\text{truncation error: } \mathcal{O}(\Delta t^3)}}$$

However, proving that the leading term is half the magnitude of that of BDF2, I will use the LHS and RHS definitions from BDF2 and use the Taylor-Series expansions from part a. and simplify as,

$$\begin{aligned}
\text{LHS} &= \frac{3}{2}u^{n+1} - 2u^n + \frac{1}{2}u^{n-1} \\
&= \frac{3}{2}\left(u^n + \Delta t u_t^n + \frac{1}{2}\Delta t^2 u_{tt}^n + \frac{1}{6}\Delta t^3 u_{ttt}^n + \frac{1}{24}\Delta t^4 u_{t^{(4)}}^n + \frac{1}{120}\Delta t^5 u_{t^{(5)}}^n\right) + \ldots \\
&- 2u^n + \ldots \\
&+ \frac{1}{2}\left(u^n - \Delta t u_t^n + \frac{1}{2}\Delta t^2 u_{tt}^n - \frac{1}{6}\Delta t^3 u_{ttt}^n + \frac{1}{24}\Delta t^4 u_{t^{(4)}}^n - \frac{1}{120}\Delta t^5 u_{t^{(5)}}^n\right) \\
&= \Delta t u_t^n + \Delta t^2 u_{tt}^n + \frac{1}{6}\Delta t^3 u_{ttt}^n + \frac{1}{12}\Delta t^4 u_{t^{(4)}}^n + \frac{1}{120}\Delta t^5 u_{t^{(5)}}^n \\
\text{RHS} &= \Delta t f^{n+1} \\
&= \Delta t \left(u_t^n + \Delta t u_{tt}^n + \Delta t^2 u_{ttt}^n + \Delta t^3 u_{t^{(4)}}^n + \Delta t^4 u_{t^{(5)}}^n\right)
\end{aligned}$$

Taking the difference between the two gives,

$$\text{LHS} - \text{RHS} = \left(\frac{1}{6} - \frac{1}{2}\right)\Delta t^3 u_{ttt}^n + \left(\frac{1}{12} - \frac{1}{6}\right)\Delta t^4 u_{t^{(4)}}^n + \left(\frac{1}{120} - \frac{1}{24}\right)\Delta t^5 u_{t^{(5)}}^n$$

$$\tau = \text{LHS} - \text{RHS} = -\frac{1}{3}\Delta t^3 u_{ttt}^n - \frac{1}{12}u_{t^{(4)}}^n - \frac{1}{30}\Delta t^5 u_{t^{(5)}}^n$$

Then re-writing both truncation errors gives,

$$\begin{aligned}
\tau_{BDF2} &= -\frac{1}{3}\Delta t^3 u_{ttt}^n - \frac{1}{12}u_{t^{(4)}}^n - \frac{1}{30}\Delta t^5 u_{t^{(5)}}^n \\
\tau_{\text{Avg}} &= -\frac{1}{6}\Delta t^3 u_{ttt} - \frac{1}{6}\Delta t^4 u_{t^{(4)}} + \frac{1}{40}\Delta t^5 u_{t^{(5)}}
\end{aligned}$$

By inspection of the truncation errors above, we see that the leading term for the averaged time-derivative is indeed <u>half</u> that of the BDF2 scheme.

2 The Beam-Warming Method

Consider the Beam-Warming (BW) method applied to the one-dimensional advection equation, $u_t + au_x = 0,\ a > 0$, with initial condition $u(x,0) = u_0(x),\ x \in [0, L]$ and periodic boundaries.

a. Derive the modified equation for the BW method and express it in the form

$$u_t + au_x = \alpha u_{xx} - \beta u_{xxx}$$

Use this equation to determine the order of accuracy of the BW method, and discuss the dispersion relation.

Starting with the modified equation for Beam-Warming method,

$$u_j^{n+1} = u_j^n - \frac{\sigma}{2}\left(3u_j^n - 4u_{j-1}^n + u_{j-2}^n\right) + \frac{\sigma^2}{2}\left(u_{j-2}^n - 2u_{j-1}^n + u_j^n\right)$$

Conducting the Taylor series expansions for these nodes gives,

$$\begin{aligned}
u_{j-2}^n &= u_j^n - 2\Delta x u_x + 2\Delta x^2 u_{xx} - \frac{4}{3}\Delta x^3 u_{xxx} + \frac{2}{3}\Delta x^4 u_{x^{(4)}} + \dots \mathcal{O}(\Delta x^5)\\
u_{j-1}^n &= u_j^n - \Delta x u_x + \frac{1}{2}\Delta x^2 u_{xx} - \frac{1}{6}\Delta x^3 u_{xxx} + \frac{1}{24}\Delta x^4 u_{x^{(4)}} + \dots \mathcal{O}(\Delta x^5)\\
u_j^{n+1} &= u_j^n + \Delta t u_t + \frac{1}{2}\Delta t^2 u_{tt} + \frac{1}{6}\Delta t^3 u_{ttt} + \frac{1}{24}\Delta t^4 u_{t^{(4)}} + \dots \mathcal{O}(\Delta t^5)
\end{aligned}$$

Expanding the right-hand side of the expression I get,

$$\text{RHS} = u_j^n - \frac{\sigma}{2}\left(3u_j^n - 4u_{j-1}^n + u_{j-2}^n\right) + \frac{\sigma^2}{2}\left(u_{j-2}^n - 2u_{j-1}^n + u_j^n\right)$$

Expressing each quantity I get,

$$\begin{aligned}
3u_j^n - 4u_{j-1}^n + u_{j-2}^n &= 3u_j^n + \dots\\
&- 4\left(u_j^n - \Delta x u_x + \frac{1}{2}\Delta x^2 u_{xx} - \frac{1}{6}\Delta x^3 u_{xxx} + \frac{1}{24}\Delta x^4 u_{x^{(4)}} + \dots \mathcal{O}(\Delta x^5)\right) + \dots\\
&+ u_j^n - 2\Delta x u_x + 2\Delta x^2 u_{xx} - \frac{4}{3}\Delta x^3 u_{xxx} + \frac{2}{3}\Delta x^4 u_{x^{(4)}} + \dots \mathcal{O}(\Delta x^5)\\
&= 2\Delta x u_x - \frac{2}{3}\Delta x^3 u_{xxx} + \frac{1}{2}\Delta x^4 u_{x^{(4)}} + \mathcal{O}(\Delta x^5)\\
u_{j-2}^n - 2u_{j-1}^n + u_j^n &= u_j^n - 2\Delta x u_x + 2\Delta x^2 u_{xx} - \frac{4}{3}\Delta x^3 u_{xxx} + \frac{2}{3}\Delta x^4 u_{x^{(4)}} + \dots \mathcal{O}(\Delta x^5) + \dots\\
&- 2\left(u_j^n - \Delta x u_x + \frac{1}{2}\Delta x^2 u_{xx} - \frac{1}{6}\Delta x^3 u_{xxx} + \frac{1}{24}\Delta x^4 u_{x^{(4)}} + \dots \mathcal{O}(\Delta x^5)\right) + \dots\\
&+ u_j^n\\
&= \Delta x^2 u_{xx} - \Delta x^3 u_{xxx} + \frac{7}{12}\Delta x^4 u_{x^{(4)}} - \frac{1}{4}\Delta x^5 u_{x^{(5)}} + \mathcal{O}(\Delta x^6)
\end{aligned}$$

Setting up the relationships I get that,

$$\begin{aligned}
u_j^n + \Delta t u_t + \frac{1}{2}\Delta t^2 u_{tt} + \frac{1}{6}\Delta t^3 u_{ttt} + \frac{1}{24}\Delta t^4 u_{t^{(4)}} + \dots \mathcal{O}(\Delta t^5) &= u_j^n - \dots\\
&\frac{\sigma}{2}\left(2\Delta x u_x - \frac{2}{3}\Delta x^3 u_{xxx} + \frac{1}{2}\Delta x^4 u_{x^{(4)}} + \mathcal{O}(\Delta x^5)\right) + \dots\\
&\frac{\sigma^2}{2}\left(\Delta x^2 u_{xx} - \Delta x^3 u_{xxx} + \frac{7}{12}\Delta x^4 u_{x^{(4)}} - \frac{1}{4}\Delta x^5 u_{x^{(5)}} + \mathcal{O}(\Delta x^6)\right)\\
&= u_j^n - \sigma\Delta x u_x + \frac{\sigma^2}{2}\Delta x^2 u_{xx} - \frac{1}{6}(3\sigma^2 - 2\sigma)\Delta x^3 u_{xxx} + \frac{1}{24}\left(7\sigma^2 - 6\sigma\right)\Delta x^4 u_{x^{(4)}}
\end{aligned}$$

Starting with subtracting the u_j^n terms and expanding σ gives,

$$\begin{aligned}\Delta t u_t + \frac{1}{2}\Delta t^2 u_{tt} + \frac{1}{6}\Delta t^3 u_{ttt} + \frac{1}{24}\Delta t^4 u_{t^{(4)}} + \dots \mathcal{O}(\Delta t^5)\\ = \frac{a\Delta t}{\Delta x}\left(-\Delta x u_x + \frac{\sigma}{2}\Delta x^2 u_{xx} - \frac{1}{6}(3\sigma - 2)\Delta x^3 u_{xxx} + \frac{1}{24}\left(7\sigma - 6\right)\Delta x^4 u_{x^{(4)}}\right)\end{aligned}$$

From here I will simplify by dividing through by Δt and distributing Δx,

$$\begin{aligned}u_t + \frac{1}{2}\Delta t u_{tt} + \frac{1}{6}\Delta t^2 u_{ttt} + \frac{1}{24}\Delta t^3 u_{t^{(4)}} + \dots \mathcal{O}(\Delta t^4)\\ = a\left(-u_x + \frac{\sigma}{2}\Delta x u_{xx} + \frac{a}{6}(3\sigma - 2)\Delta x^2 u_{xxx} + \frac{1}{24}\left(7\sigma - 6\right)\Delta x^4 u_{x^{(4)}}\right)\end{aligned}$$

Collecting the one-dimensional advection term to the same side,

$$\begin{aligned}u_t + a u_x &= -\frac{1}{2}\Delta t u_{tt} - \frac{1}{6}\Delta t^2 u_{ttt} - \frac{1}{24}\Delta t^3 u_{t^{(4)}} + \frac{\sigma a}{2}\Delta x u_{xx} + \dots\\ &+ \frac{a}{6}(3\sigma - 2)\Delta x^2 u_{xxx} + \frac{1}{24}\left(7\sigma - 6\right)\Delta x^4 u_{x^{(4)}}\end{aligned}$$

Now with the expression for the one-dimensional advection solved for, I will relate temporal derivatives to spatial indices by conducting expansions,

$$\begin{aligned}u_{tt} &= -\frac{1}{2}\Delta t u_{ttt} - a u_{xt} + \frac{\sigma a}{2}\Delta x u_{xxt} + \mathcal{O}(\Delta x^2, \Delta t^2)\\ u_{tx} &= -\frac{1}{2}\Delta t u_{ttx} - a u_{xx} + \frac{\sigma a}{2}\Delta x u_{xxx} + \mathcal{O}(\Delta x^2, \Delta t^2)\\ u_{ttt} &= -a u_{xtt} + \mathcal{O}(\Delta x, \Delta t)\\ u_{txx} &= -a u_{xxx} + \mathcal{O}(\Delta x, \Delta t)\\ u_{ttx} &= -a u_{xxt}\end{aligned}$$

With the higher mixed-derivatives solved for, backtracking will find the α and β coefficients,

$$\begin{aligned}u_{txx} &= -a u_{xxx}\\ u_{ttx} &= a^2 u_{xxx}\\ u_{ttt} &= -a^3 u_{xxx}\\ u_{tx} &= -\frac{1}{2}\Delta t a^2 u_{xxx} - a u_{xx} + \frac{\sigma a}{2}\Delta x u_{xxx}\\ &= -a u_{xx} + \cancelto{0}{\left(\frac{\sigma a}{2}\Delta x - \frac{a^2 \Delta t}{2}\right)} u_{xxx} = -a u_{xx}\\ u_{tt} &= \frac{1}{2}\Delta t(a^3 u_{xxx}) + a^2 u_{xx} - \frac{\sigma a^2}{2}\Delta x u_{xxx} = a^2 u_{xx}\\ u_t + a u_x &= -\frac{1}{2}\Delta t a^2 u_{xx} + \frac{1}{6}\Delta t^2 a^3 u_{xxx} + \frac{\sigma a}{2}\Delta x u_{xx} + \frac{a}{6}(3\sigma - 2)\Delta x^2 u_{xxx} + \mathcal{O}(\Delta x^3, \Delta t^3)\\ &= \underbrace{\cancelto{0}{\left(-\frac{1}{2}\Delta t a^2 + \frac{\sigma a}{2}\Delta x\right)}}_{\alpha} u_{xx} + \underbrace{\left(\frac{1}{6}\Delta t^2 a^3 + \frac{a}{6}(3\sigma - 2)\Delta x^2\right)}_{-\beta} u_{xxx}\\ &= 0 \cdot u_{xx} + a\left(\frac{1}{6}\frac{\Delta x^2}{\Delta x^2}\Delta t^2 a^2 + \frac{a}{6}(3\sigma - 2)\Delta x^2\right)\\ &= 0 \cdot u_{xx} + a\left(\frac{\Delta x^2}{6}\sigma^2 + \frac{a}{6}(3\sigma - 2)\Delta x^2\right) u_{xxx}\end{aligned}$$

After further simplifications,

$$u_t + au_x = 0 \cdot u_{xx} + \frac{a\Delta x^2}{6}\left(\sigma^2 - 3\sigma + 2\right) u_{xxx}$$

This gives that the α and β expressions are,

$$\boxed{\alpha = 0, \quad \beta = -\frac{a\Delta x^2}{6}(\sigma^2 - 3\sigma + 2)}$$

Looking above to the dispersion (the coefficient β) will denote how waves of different frequencies will move at different speeds. This dispersion term will be the cause of oscillations where they were not present before. These dispersion effects will be present and more visible if the CFL number goes past its stability limits.

Order of Accuracy

Using the relationship that has been solved for gives,

$$u_t + au_x = \frac{a\Delta x^2}{6}(\sigma^2 - 3\sigma + 2)u_{xxx}$$

This Beam-Warming method solves a modified PDE that contains a dispersion term, but re-writing this scheme from the Taylor-Series expansion to observe the truncation error gives,

$$N(u_j^n) = u_t + au_x + \frac{1}{2}\Delta t u_{tt} + \frac{1}{6}\Delta t^2 u_{ttt} + \frac{1}{24}\Delta t^3 u_{t^{(4)}} - \frac{\sigma a}{2}\Delta x u_{xx} + \dots$$
$$- \frac{a}{6}(3\sigma - 2)\Delta x^2 u_{xxx} - \frac{1}{24}\left(7\sigma - 6\right)\Delta x^4 u_{x^{(4)}}$$

Expanding the above terms into $\Delta x,\ \Delta t$ gives,

$$N(u_j^n) = u_t + au_x + \frac{1}{2}\Delta t u_{tt} + \frac{1}{6}\Delta t^2 u_{ttt} + \frac{1}{24}\Delta t^3 u_{t^{(4)}} - \frac{a^2\Delta t}{2}u_{xx} + \dots$$
$$- a^2\Delta t\Delta x u_{xxx} + \frac{a}{3}\Delta x^2 u_{xxx} - \frac{7}{24}a\Delta t\Delta x^3 u_{x^{(4)}} + \frac{1}{4}\Delta x^4 u_{x^{(4)}}$$

Re-arranging for the leading terms gives,

$$N(u_j^n) = \underbrace{u_t + au_x}_{D(u_j^n)} + \underbrace{\frac{1}{2}\Delta t u_{tt} + \frac{a}{3}\Delta x^2 u_{xxx} - a^2\Delta t\Delta x u_{xxx}}_{\text{truncation error: } \mathcal{O}(\Delta x^2,\ \Delta t)} + \dots$$

Shown above, this scheme is consistent since as $\boldsymbol{\Delta}$x, $\boldsymbol{\Delta}$t $\to$ 0 the solution will become approximate. This scheme is spatially second-order accurate as the power $\boldsymbol{\Delta}$x is 2, and temporally first-order accurate as the pwoer $\boldsymbol{\Delta}$t is 1.

b. Perform a von-Neumann stability analysis of the Beam-Warming method. What is the stability limit for the CFL number σ?

In order to complete the von-Neumann stability analysis, substitute $u_j^n - g^n e^{ij\phi}$ into the equation and calculate the amplification factor g,

$$
\begin{aligned}
u_j^{n+1} &= u_j^n - \frac{\sigma}{2}\left(3u_j^n - 4u_{j-1}^n + u_{j-2}^n\right) + \frac{\sigma^2}{2}\left(u_{j-2}^n - 2u_{j-1}^n + u_j^n\right) \\
g^{n+1}e^{ij\phi} &= g^n e^{ij\phi} - \frac{\sigma}{2}\left(3g^n e^{ij\phi} - 4g^n e^{i(j-1)\phi} + g^n e^{i(j-2)\phi}\right) + \frac{\sigma^2}{2}\left(g^n e^{i(j-2)\phi} - 2g^n e^{i(j-1)\phi} + g^n e^{ij\phi}\right) \\
ge^{ij\phi} &= e^{ij\phi} - \frac{\sigma}{2}\left(3e^{ij\phi} - 4e^{i(j-1)\phi} + e^{i(j-2)\phi}\right) + \frac{\sigma^2}{2}\left(e^{i(j-2)\phi} - 2e^{i(j-1)\phi} + e^{ij\phi}\right) \\
g &= 1 - \frac{\sigma}{2}\left(3 - 4e^{-i\phi} + e^{-2i\phi}\right) + \frac{\sigma^2}{2}\left(e^{-2i\phi} - 2e^{-i\phi} + 1\right)
\end{aligned}
$$

Expanding this term gives,

$$
g = 1 - \frac{3\sigma}{2} + \frac{\sigma^2}{2} + \sigma\left(2-\sigma\right)e^{-i\phi} + \frac{\sigma}{2}\left(\sigma - 1\right)e^{-2i\phi}
$$

Since, $|e^{-i\phi}| \in [0,1]$, taking the norm of g gives

$$
1 = 1 - \frac{3\sigma}{2} + \frac{\sigma^2}{2} + \sigma\left(2-\sigma\right)e^{-i\phi} + \frac{\sigma}{2}\left(\sigma - 1\right)e^{-2i\phi}
$$

Isolating for interms of purely σ and $e^{-i\phi}$ gives,

$$
0 = -\frac{3\sigma}{2} + \frac{\sigma^2}{2} + \sigma\left(2-\sigma\right)e^{-i\phi} + \frac{\sigma}{2}\left(\sigma - 1\right)e^{-2i\phi}
$$

From here one obvious choice for σ is,

$$
\sigma = 0
$$

Looking to the grouped terms trying $\sigma = 1$ gives,

$$
\begin{aligned}
0 &\geq -\frac{3}{2} + \frac{1}{2} + 1\cancelto{1}{|e^{-i\phi}|} \\
0 &= 0, \quad \text{Valid } \sigma
\end{aligned}
$$

Looking to the other grouped term trying $\sigma = 2$ gives,

$$
\begin{aligned}
0 &\geq -\frac{3}{2} + \frac{1}{2} + \cancelto{1}{|e^{-2i\phi}|} \\
0 &= 0, \quad \text{Valid } \sigma
\end{aligned}
$$

With three terms for σ this gives the limiting case for the CFL number,

$$
\boxed{\sigma \leq 2}
$$

c. Implement the BW method in a computer program using $L = 2,\ a = 0.5,\ u_0(x) = exp[-100(x/L - 0.5)^2]$ and a final time of $T = L/a$ (1 period). Perform spatial and temporal convergence studies to demonstrate the order of accuracy in space and time.

Implementing the Beam-Warming method with Matlab attached at the end of the assignment and performing an L_2 norm of the solution approximated at time T for varying levels of N_x and N_t gives that the convergences for each is shown below in Figure 2.

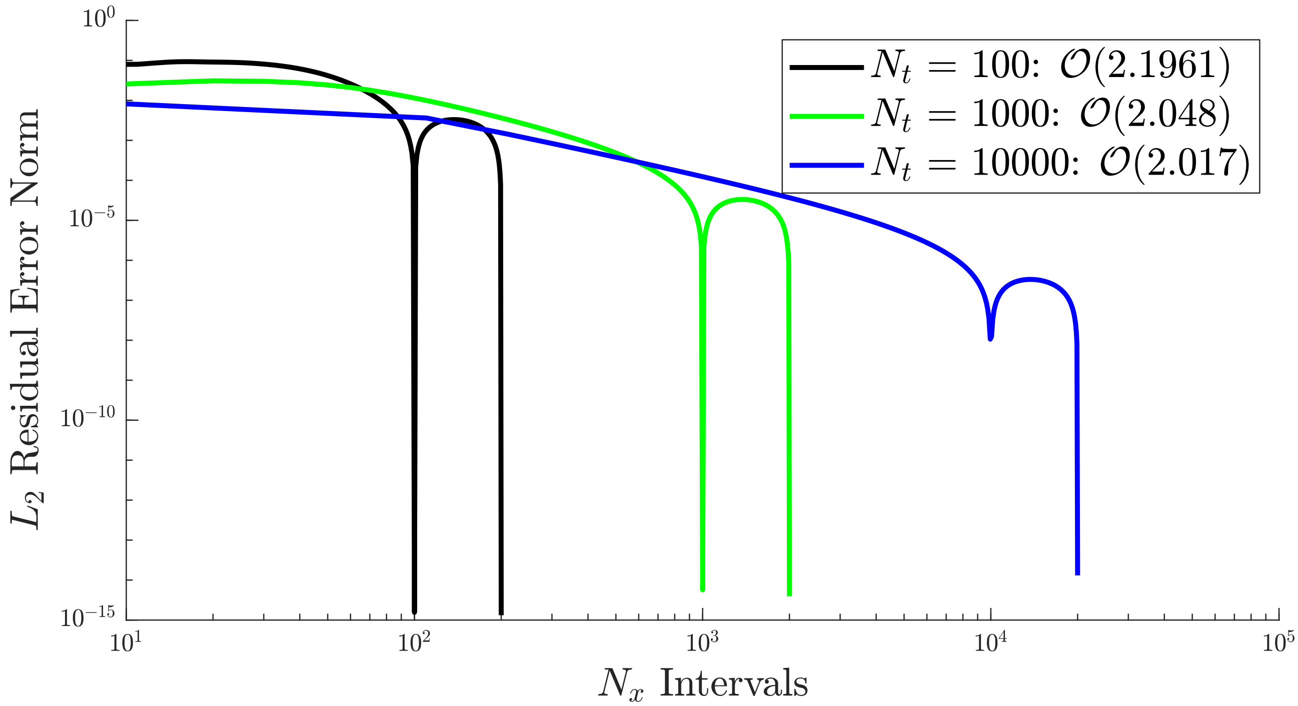

Figure 2: Beam-Warming implementation and convergence of spatial and temporal convergences.

Shown above in Figure 2 are the convergences of the $\mathbf{L_2}$ norm. As $\mathbf{N_x}$ increases for each given $\mathbf{N_t}$ value, it will reach the CFL number stability limits of $\sigma = 1,\ 2$. When $\sigma = 1$ for each combination of $\mathbf{N_x},\ \mathbf{N_t}$ it is seen as the vertical asymptote in which the residual norm decreases significantly. Computing the orders of accuracy in the spatial domain confirms that this scheme is second-order accurate in the spatial domain such that $\mathcal{O}(\mathbf{\Delta x^2})$. Looking at the temporal convergence can be done through visual inspection to be first order accurate $\mathcal{O}(\mathbf{\Delta t})$, but will be shown on the following page.

Continued on the next page...

Implementing the Beam-Warming method again but only varying N_t gives that the spatial convergence below in Figure 3,

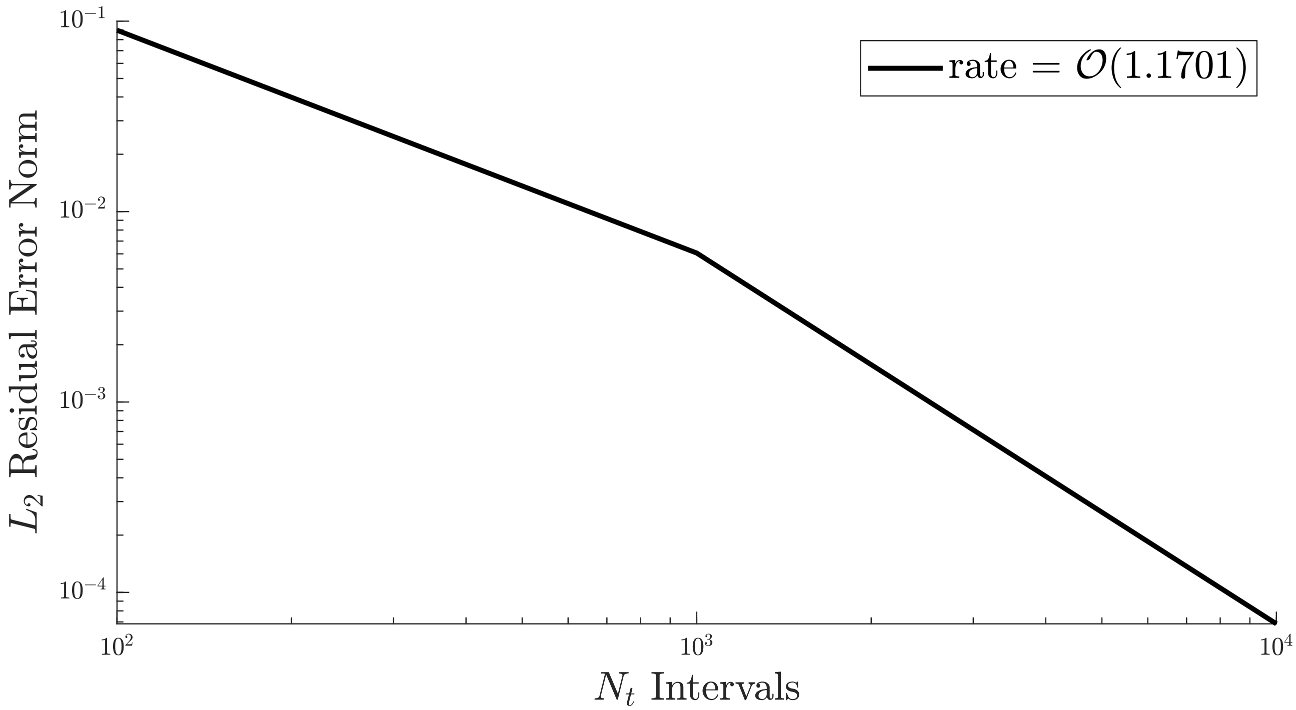

Figure 3: Temporal only convergences of Beam-Warming method.

Shown above in Figure 3 are the temporal convergences of the Beam-Warming method. Again it is shown and confirmed that it is first-order accurate.

3 Manufactured Solutions

Discretize the one-dimensional advection-diffusion equation, $u_t + au_x - \nu u_{xx} = 0$, using the trapezoidal method in time and second-order central differences in space. Assume a grid of length L, periodic boundaries, and N spatial intervals.

a. Write a computer program that implements the given method, and run a simulation using the parameters given in problem 2c, $\nu = 0.1,\ N = 64$, and a CFL number of $\sigma = 0.5$. Plot the state at the final time, $u(x, T)$.

First starting with the expressions for the expansions one-dimensional advection-diffusion equation gives,

$$\begin{aligned} u_j^{n+1} &= u_j^n + \frac{\Delta t}{2}\left(f^{n+1} + f^n\right) \\ u_x &= \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} \\ u_{xx} &= \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2} \end{aligned}$$

Substituting in and solve for u_t gives,

$$\begin{aligned} u_t &= \nu\frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2} - a\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = \dot{u} = f = \underline{\underline{A}}u_j^n \\ \underline{\underline{A}}u_j^n &= -\frac{2\nu}{\Delta x^2}u_j^n + \left(\frac{\nu}{\Delta x^2} + \frac{a}{2\Delta x}\right)u_{j-1}^n + \left(\frac{\nu}{\Delta x^2} - \frac{a}{2\Delta x}\right)u_{j+1}^n \end{aligned}$$

Using this relationship of $\underline{\underline{A}}$, solve for the update relationship for u_j^{n+1},

$$\begin{aligned} u_j^{n+1} - \frac{\Delta t}{2}f^{n+1} &= u_j^n + \frac{\Delta t}{2}f^n \\ u_j^{n+1} - \frac{\Delta t}{2}\underline{\underline{A}}u^{n+1} &= u_j^n + \frac{\Delta t}{2}\underline{\underline{A}}u^n \\ \left(\underline{\underline{I}} - \frac{\Delta t}{2}\underline{\underline{A}}\right)u_j^{n+1} &= \left(\underline{\underline{I}} + \frac{\Delta t}{2}\underline{\underline{A}}\right)u_j^n \end{aligned}$$

$$\boxed{u_j^{n+1} = \left(\underline{\underline{I}} - \frac{\Delta t}{2}\underline{\underline{A}}\right)^{-1}\left(\underline{\underline{I}} + \frac{\Delta t}{2}\underline{\underline{A}}\right)u_j^n}$$

Continued on next page . . .

Implementing the trapezoidal method derived above and implementing into Matlab with the given parameters above, I generate Figure 4 shown below.

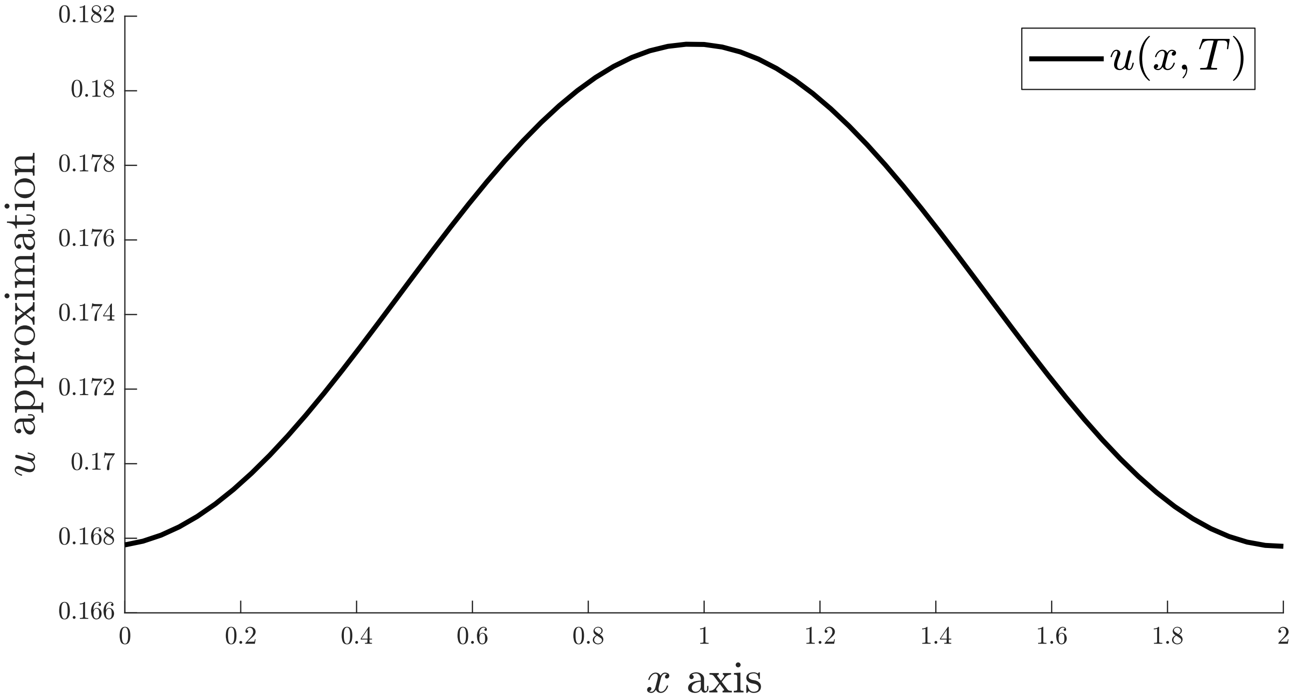

Figure 4: Implementation of trapezoidal method.

Shown above in Figure 4, is the approximated solution through implementation of of the trapezoidal method. This shows the solution after one period which should return the initial condition decayed by some exponential factor.

b. Apply the method of manufactured solutions to your discretization. The PDE will now need a source term: $u_t + au_x - \nu u_{xx} = s(x,t)$. Derive the form of $s(x,t)$ for the manufactured solution $u^{MS} = \sin(kx - \omega t)$, where k and ω are known constants.

Using the manufactured solution, and substituting in the expression for the PDE I get,

$$s(x,t) = \frac{\partial}{\partial t}u^{\text{MS}} + a\frac{\partial}{\partial x}u^{\text{MS}} - \nu\frac{\partial^2}{\partial x^2}u^{\text{MS}}$$
$$s(x,t) = -\omega\cos(kx - \omega t) + ak\cos(kx - \omega t) + \nu k^2\sin(kx - \omega t)$$

Condensing and simplfying the solution for $s(x,t)$ further gives,

$$\boxed{s(x,t) = (ak - \omega)\cos(kx - \omega t) + \nu k^2\sin(kx - \omega t)}$$

Since we are implementing the trapezoidal method, we take the average in time such that,

$$\underline{s} = \frac{\Delta t}{2}\left(s(x,t^{n+1}) + s(x,t^n)\right)$$

Then following the steps from part a. but with the additional source term gives,

$$u_j^{n+1} = \left(\underline{\underline{I}} - \frac{\Delta t}{2}\underline{\underline{A}}\right)^{-1}\left(\left(\underline{\underline{I}} + \frac{\Delta t}{2}\underline{\underline{A}}\right)u_j^n + \underline{s}\right)$$

Therefore, the trapezoidal update for u_j^{n+1} is,

$$\boxed{u_j^{n+1} = \left(\underline{\underline{I}} - \frac{\Delta t}{2}\underline{\underline{A}}\right)^{-1}\left(\left(\underline{\underline{I}} + \frac{\Delta t}{2}\underline{\underline{A}}\right)u_j^n + \frac{\Delta t}{2}\left(s(x,t^{n+1}) + s(x,t^n)\right)\right)}$$

c. Implement the method of manufactured solutions in your discretization, and present the solution at $t = T = L/a$ for $N = 64,\ \sigma = 0.5$. Use $k = 4\pi/L$ and $\omega = 5a/L$.

Using the source term $s(x,t)$ derived in part c. above and implementing through Matlab gives that the final value is below in Figure 5

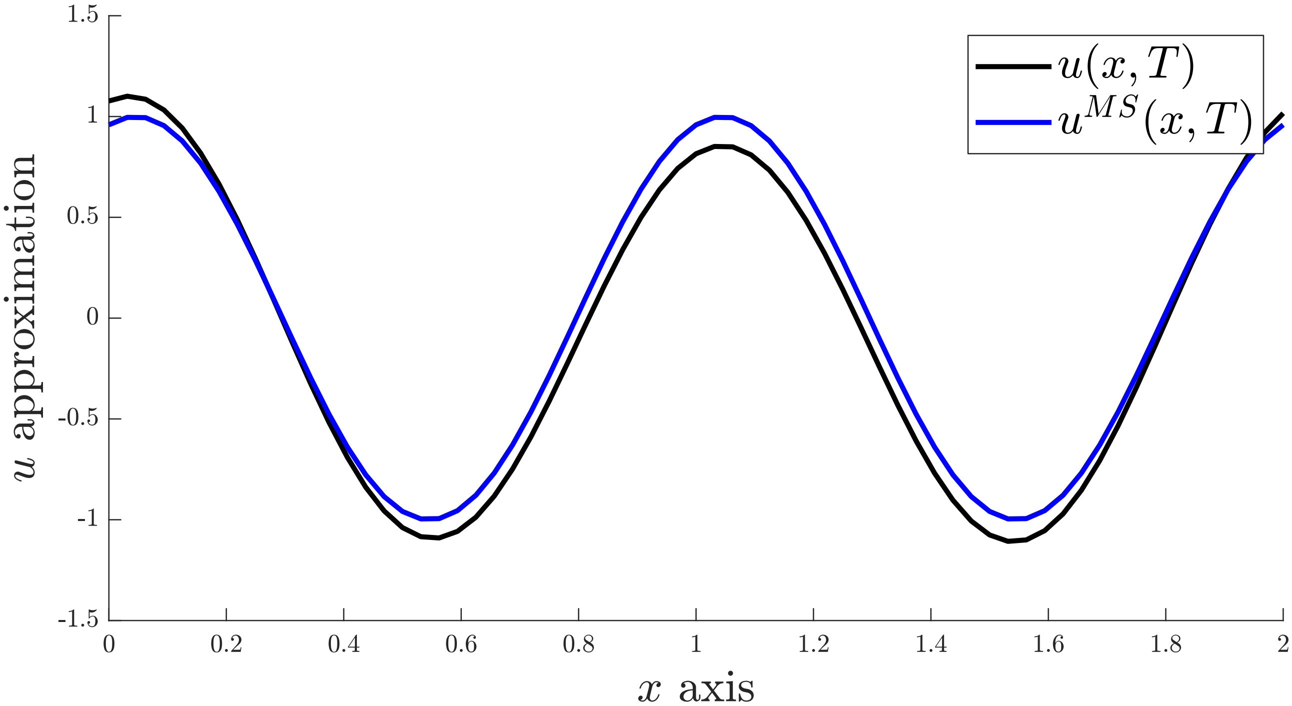

Figure 5: Implementation method of manufactured solutions for $T = L/a, N = 64,\ \sigma = 0.5$.

Looking above to Figure 5 and comparing the approximated solution from the trapezoidal method gives that it is close in approximation to the analytical solution (manufactured solution) when N = 64. At higher values of N the solution is much closer in comparison.

d. Using the manufactured solution, perform spatial and temporal convergence studies of your discretization, using the L_2 solution error, and verify that the orders of accuracy match your expectations.

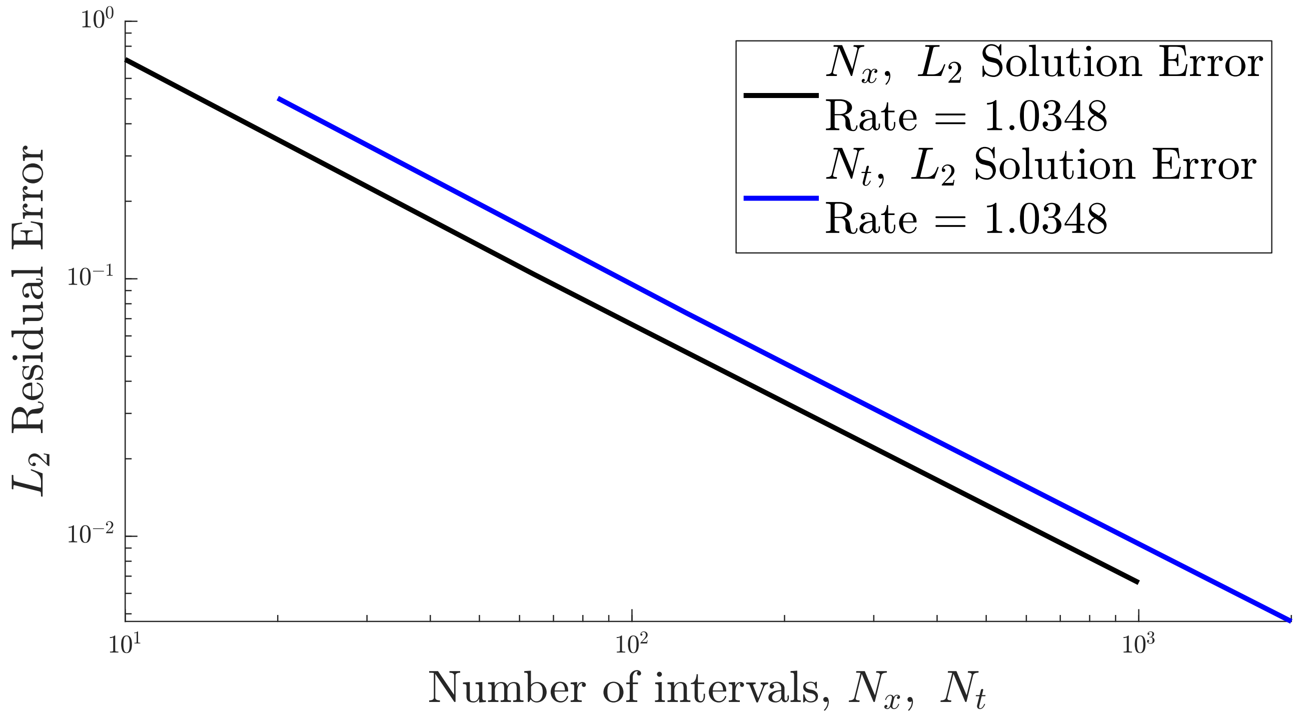

Figure 6: Spatial and temporal convergence study using manufactured solution.

Looking above in Figure 6 to the spatial and temporal convergences shows that this is first-order accurate such that $\mathcal{O}(\boldsymbol{\Delta}\mathbf{t})$ when converging.

Tabulating the spatial and temporal solution errors from the graph above gives that the convergences are as follows in Tables 1, 2 below confirming first-order accuracy.

Table 1: Spatial convergence varying N_x

Rates	Order of accuracy
$\text{rate}_{N=10}$	$= 1.0348$
$\text{rate}_{N=64}$	$= 1.0085$
$\text{rate}_{N=100}$	$= 1.0032$
$\text{rate}_{N=500}$	$= 1.0009$

Table 2: Temporal convergence varying N_t

Rates	Order of accuracy
$\text{rate}_{N=20}$	$= 1.0348$
$\text{rate}_{N=128}$	$= 1.0085$
$\text{rate}_{N=200}$	$= 1.0032$
$\text{rate}_{N=1000}$	$= 1.0009$

Matlab Code for A-Stable Backwards Difference

Algorithm 1: Matlab Code for determining A-Stable backwards differences.

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
clear all; clc; close all
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% a
syms u ut utt uttt utttt uttttt dt

unp1 = u + dt*ut + 1/2*dt^2*utt + 1/6*dt^3*uttt + 1/24*dt^4*utttt + 1/120*dt^5*uttttt;
un = u;
unm1 = u - dt*ut + 1/2*dt^2*utt - 1/6*dt^3*uttt + 1/24*dt^4*utttt - 1/120*dt^5*uttttt;
unm2 = u - 2*dt*ut + 2*dt^2*utt - 4/3*dt^3*uttt + 2/3*dt^4*utttt + (-2)^5/120*dt^5*uttttt;
fnp1 = ut + dt*utt + 1/2*dt^2*uttt + 1/6*dt^3*utttt + 1/24*dt^4*uttttt;

epsi = 5/3*unp1 - 5/2*un + unm1 - 1/6*unm2 - dt*fnp1;
pretty(simplify(epsi))
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% b
num = 100;
bdf2 = zeros(num, 1);
bdf3 = zeros(num, 1);
avg_der = zeros(num, 1);
thetlin = linspace(0, 2*pi, num);
g = exp(thetlin.*1i);
syms ldt
for i = 1:num
    eqn = g(i) == 4/3 - 1/(3*g(i)) + 2/3*ldt*g(i);
    sol = double(solve(eqn, ldt));
    bdf2(i) = sol;

    eqn = g(i) == 18/11 -9/(11*g(i)) + 2/(11*g(i)^2) + 6/11*ldt*g(i);
    sol = double(solve(eqn, ldt));
    bdf3(i) = sol;

    eqn = 5/3*g(i) - 5/2 + g(i)^(-1) - 1/6*g(i)^(-2) == ldt*g(i);
    sol = double(solve(eqn, ldt));
    avg_der(i) = sol;
end

figure()
hold on
fill(real(bdf3), imag(bdf3),[1,1,1],'facealpha', 1, 'FaceColor',[1,0,0],'EdgeColor','k','linewidth',1.8)
fill(real(avg_der), imag(avg_der),[1,1,1],'facealpha', 1, 'FaceColor',[0,0,1],'EdgeColor','k','linewidth',1.8)
fill(real(bdf2), imag(bdf2),[1,1,1],'facealpha', 1, 'FaceColor',[0.8,0.8,0.8],'EdgeColor','k','linewidth',1.8)
xlim([-2.5, 0])
axis equal
xlabel('$Re \left( \lambda \Delta T \right)$','fontsize', 18)
ylabel('$Im \left( \lambda \Delta T \right)$','fontsize', 18)
legend({'BDF3','$\frac{1}{2}\frac{du}{dt}|_{BDF2} + \frac{1}{2}\frac{du}{dt}|_{BDF3}$','BDF2'}, 'fontsize', 18, 'location', 'best', 'interpreter', 'latex')
set(gcf, 'Color', 'w', 'Position', [100 100 1000 500]);
export_fig('eigs.eps')

syms ldt theta
g = exp(1i*theta);
eqn = 0 == 5/3 - 5/2*g^-1 + g^-2 - 1/6*g^-3;
sol = double(solve(eqn, theta));
```

Matlab Code for Beam-Warming Method

Algorithm 2: Matlab Code for implementation of Beam-Warming method.

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
clear all; clc; close all
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
num = 200;
ntlin = 10.^(2:4);
l2norms = zeros(max(size(ntlin)), num);
intervals = zeros(max(size(ntlin)), num);
for j = 1:max(size(ntlin))
    intervals(j,:) = floor(linspace(10, 2*ntlin(j), num));
    for i = 1:max(size(intervals))
        [us, xlin] = BW_method(intervals(j,i), ntlin(j));

        l2norms(j,i) = sqrt(1./ntlin(j).*sum((us(1,:) - us(end,:)).^2));
    end
    fprintf('Nt = %.f \n', ntlin(j))
end
val = floor(num/3);
rate1 = abs(log10(l2norms(1,2*val)/l2norms(1,val))/log10(intervals(1,2*val)/intervals(1,val)));
rate2 = abs(log10(l2norms(2,floor(2.5*val))/l2norms(2,floor(1.1*val)))/log10(intervals(2,floor(2.5*val))/intervals(2,floor(1.1*val))));
rate3 = abs(log10(l2norms(3,floor(2.4*val))/l2norms(3,floor(1.1*val)))/log10(intervals(3,floor(2.4*val))/intervals(3,floor(1.1*val))));

figure()
hold on
plot(intervals(1,:), l2norms(1,:), 'k', 'linewidth', 2)
plot(intervals(2,:), l2norms(2,:), 'g', 'linewidth', 2)
plot(intervals(3,:), l2norms(3,:), 'b', 'linewidth', 2)
xlabel('$N_x$ Intervals', 'fontsize', 16)
ylabel('$L_2$ Residual Error Norm', 'fontsize', 16)
legend({['$N_t$ = 100: $\mathcal{O}$(', num2str(rate1),')'],...
        ['$N_t$ = 1000: $\mathcal{O}$(', num2str(rate2),')'],...
        ['$N_t$ = 10000: $\mathcal{O}$(', num2str(rate3),')']}, 'fontsize', 16, 'location', 'northeast')
set(gca, 'yscale', 'log')
set(gca, 'xscale', 'log')
set(gcf, 'Color', 'w', 'Position', [200 200 800 400]);
export_fig('BW_convergence.eps')
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
l2err = l2norms(:,15);
rate = abs(log10(l2err(2)/l2err(1))/log10(ntlin(2)/ntlin(1)));

figure()
hold on
plot(ntlin, l2err, 'k', 'linewidth', 2)
xlabel('$N_t$ Intervals', 'fontsize', 16)
ylabel('$L_2$ Residual Error Norm', 'fontsize', 16)
legend(['rate = $\mathcal{O}$(', num2str(rate),')'], 'fontsize', 16, 'location', 'northeast')
set(gca, 'yscale', 'log')
set(gca, 'xscale', 'log')
set(gcf, 'Color', 'w', 'Position', [200 200 800 400]);
export_fig('BW_convergence_nt.eps')
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
function [us, xlin] = BW_method(nx, nt)
    L = 2; a = 0.5; T = L/a;
    xlin = linspace(0, L, nx + 1);
    tlin = linspace(0, T, nt + 1);
    us = zeros(nt+1, nx+1);

    us(1,:) = exp(-100.*(xlin./L - 0.5).^2);
    dx = xlin(2); dt = tlin(2);
    sig = a*dt/dx;

    for n = 1:nt
        for j = 1:(nx+1)
            if j-1 == 0
                ujm1 = us(n,nx);
                ujm2 = us(n,nx-1);
            elseif j-1 == 1
```

```
                ujm1 = us(n,j-1);
                ujm2 = us(n,nx);
            else
                ujm1 = us(n,j-1);
                ujm2 = us(n,j-2);
            end

            uj = us(n,j);
            us(n+1,j) = us(n,j) - sig/2*(3*uj - 4*ujm1 + ujm2) + sig^2/2*(ujm2 - 2*ujm1 + uj);
        end
    end
end
```

Matlab Code for Trapezoidal Method

Algorithm 3: Matlab Code for implementation of trapezoidal method.

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
clear all; clc; close all
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Input Variables
L = 2; a = 0.5; T = L/a;
nu = 0.1; sig = 0.5;
N = 64;

% Variables for input
xlin = linspace(0, L, N + 1);
dx = xlin(2);
dt = sig * dx/a;
nt = max(size(0:dt:T));

A = sparse(N + 1); % Pre-allocate A matrix
for i = 1:(N+1)
   A(i,i) = -2*nu/dx^2; % Diagonal

   if i > 1 % U, D entries
        A(i,i-1) = nu/dx^2 + a/(2*dx);
   end
   if i < N+1
        A(i,i+1) = nu/dx^2 - a/(2*dx);
   end
   if i == 1  % Handle Periodic Boundaries
       A(i,N+1) = nu/dx^2 + a/(2*dx);
   elseif i == N+1
       A(i,1) = nu/dx^2 - a/(2*dx);
   end
end
% Create super matrix for trapezoidal method
bigA = (eye(N+1) + dt/2*A)*inv((eye(N+1) - dt/2*A));

us = zeros(N+1, nt+1); % Pre-allocation and initial condition
us(:,1) = exp(-100.*(xlin./L - 0.5).^2);
for j = 1:nt
    us(:,j+1) = bigA*us(:,j); % Iterate with trapozoidal iteration
end

figure()
hold on
plot(xlin, us(:,end), 'k', 'linewidth', 2)
xlabel('$x$ axis', 'fontsize', 18)
ylabel('$u$ approximation', 'fontsize', 18)
legend('$u(x,T)$', 'location', 'best','fontsize', 18)
set(gcf, 'Color', 'w', 'Position', [200 200 800 400]);
export_fig('q3a.eps')
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% c
L = 2; a = 0.5; T = L/a;
nu = 0.1; sig = 0.5;
k = 4*pi/L; omega = 5*a/L;
% Variables for input
nums = [10, 64, 100, 500, 1000];
ntvals = zeros(1, max(size(nums)));
l2err = zeros(1, max(size(nums)));
l2errnt= zeros(1, max(size(nums))); idx = 1;
for N = nums
    fprintf('Iteration - %.f\n', N)
    xlin = linspace(0, L, N + 1);
    dx = xlin(2);
    dt = sig * dx/a;
    nt = N/a;
    tlin = linspace(0, T, nt + 1);
    ntvals(idx) = nt;

    us = zeros(N+1, nt+1); % Pre-allocation and initial condition
    us(:,1) = sin(k.*xlin - omega*0);
    A = buildA(N, nu, dx, a);
```

```
    for j = 1:nt
        t = tlin(j);
        t2 = tlin(j+1);
        source1 = (a*k - omega).*cos(k.*xlin - omega*t) + nu*k^2*sin(k.*xlin - omega*t);
        source2 = (a*k - omega).*cos(k.*xlin - omega*t2) + nu*k^2*sin(k.*xlin - omega*t2);

        source_contrib = (eye(N+1) - dt/2*A)^(-1)*dt/2*(source1' + source2');
        us(:,j+1) = (eye(N+1) - dt/2*A)^(-1)*(eye(N+1) + dt/2*A)*us(:,j) + source_contrib;
    end

    uexact = sin(k.*xlin - omega*T);
    l2err(idx) = sqrt(1/N*sum((uexact' - us(:,end)).^2));
    l2errnt(idx) = sqrt(1/nt*sum((uexact' - us(:,end)).^2));
    idx = idx + 1;
    if N == 64
        figure()
        hold on
        plot(xlin, us(:,end), 'k-', 'linewidth', 2)
        plot(xlin, uexact, 'b', 'linewidth', 2)
        xlabel('$x$ axis', 'fontsize', 18)
        ylabel('$u$ approximation', 'fontsize', 18)
        legend({'$u(x,T)$', '$u^{MS}(x,T)$'}, 'location', 'northeast','fontsize', 18)
        set(gcf, 'Color', 'w', 'Position', [200 200 800 400]);
        export_fig('q3c.eps')
    end
end
ratesnx = zeros(1, (max(size(l2err)) - 1));
fid = fopen('rates_nx','w');
for i = 1:(max(size(l2err)) - 1)
    ratesnx(i) = abs(log10(l2err(i+1)/l2err(i))/log10(nums(i+1)/nums(i)));
    fprintf(fid,'$ \\text{rate}_{N = %.f} $ & = %.4f\\\\ \n',nums(i), ratesnx(i));
end
fclose(fid);
ratesnt = zeros(1, (max(size(l2err)) - 1));
fid = fopen('rates_nt','w');
for i = 1:(max(size(l2err)) - 1)
    ratesnt(i) = abs(log10(l2errnt(i+1)/l2errnt(i))/log10(ntvals(i+1)/ntvals(i)));
    fprintf(fid,'$ \\text{rate}_{N = %.f} $ & = %.4f\\\\ \n',ntvals(i), ratesnt(i));
end
fclose(fid);

figure()
hold on
plot(nums, l2err, 'k-', 'linewidth', 2)
plot(ntvals, l2errnt, 'b-', 'linewidth', 2)
xlabel('Number of intervals, $N_x,\ N_t$', 'fontsize', 18)
ylabel('$L_2$ Residual Error', 'fontsize', 18)
legend({['$N_x,\ L_2$ Solution Error', newline, 'Rate = ', num2str(ratesnx(1))], ...
    ['$N_t,\ L_2$ Solution Error', newline, 'Rate = ', num2str(ratesnt(1))]}, 'location', 'northeast','fontsize', 18)
set(gca, 'xscale', 'log')
set(gca, 'yscale', 'log')
set(gcf, 'Color', 'w', 'Position', [200 200 800 400]);
export_fig('q3d.eps')
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
function A = buildA(N, nu, dx, a)
    A = sparse(N + 1); % Pre-allocate A matrix
    for i = 1:(N+1)
       A(i,i) = -2*nu/dx^2; % Diagonal

       if i > 1 % U, D entries
            A(i,i-1) = nu/dx^2 + a/(2*dx);
       end
       if i < N+1
            A(i,i+1) = nu/dx^2 - a/(2*dx);
       end

       if i == 1  % Handle Periodic Boundaries
           A(i,N+1) = nu/dx^2 + a/(2*dx);
       elseif i == N+1
           A(i,1) = nu/dx^2 - a/(2*dx);
       end
    end
end
```