使用 Runge—Kutta 法估計多區間內之波方程

KTHFAN

January 2022

Contents

1 簡介

當波通過不同的介質時，其波速可能會隨著材料的性質而改變。而在本報告中，將會討論當波在不同的區間內，其對應的波速不一致時，波方程的估計格式應該如何設計，並且對此估計格式的穩定性進行討論以及實驗。

2 定義

在本章節裡，將會對問題進行定義以及描述。首先，假設波方程的定義如式(1)，其中波會經過 $I^{(1)}, I^{(2)}, ..., I^{(M)}$ 共 M 個相鄰的區間，而波在區間 $I^{(m)}$ 上的波速爲 $a^{(m)}$、初始條件爲 $f^{(m)}(x)$，如圖1所示，其中 $a^{(m)} > 0$。至於波的來源，則是在 x_0 上由 $g(t)$ 定義，x_0 爲 $I^{(1)}$ 的左端點。

$$\begin{aligned} &\frac{\partial u(x,t)}{\partial t} + a^{(m)} \frac{\partial u(x,t)}{\partial x} = 0 \\ &u(x,0) = f^{(m)}(x) \\ &u(x_0,t) = g(t) \\ &x \in I^{(m)} \\ &m = 1, 2, ..., M \end{aligned} \tag{1}$$

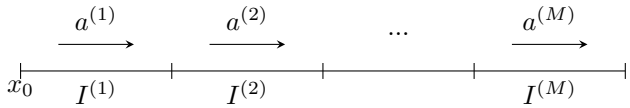

Figure 1: 波行進於多個區間上

3 估計格式

在本章節中，將會說明式(1)的估計格式應該如何設計。

在估計過程中，假設每個區間 $I^{(m)} = [l^{(m)}, r^{(m)}]$ 爲閉區間，如此一來，每個相鄰的區間在端點將會發生重疊。由於波是由左向右傳遞的，因此當波由區間 $I^{(m)}$ 傳遞至緊鄰的區間 $I^{(n)}$ 時，u 在 $l^{(n)}$ 上的數值應該與 u 在 $r^{(m)}$ 上的值有關，因此考慮一個半離散化格式(2)，其中當 $m = 1$ 時 $g = g(t)$，至於 $m = 2, 3, ..., M$ 時，g 爲 u 在 $r^{(m-1)}$ 的估計值。透過補償項 $-a^{(m)}\tau^{(m)}(u_0 - g)$ 的加入，格式將有能力把上一個區間 $I^{(m)}$ 的輸出資訊作爲緊鄰區間 $I^{(n)}$ 的訊息來源。

$$\begin{aligned}
&\frac{du_0}{dt} + a^{(m)}\frac{u_1 - u_0}{\Delta x} = -a^{(m)}\tau^{(m)}(u_0 - g) \\
&\frac{du_j}{dt} + a^{(m)}\frac{u_{j+1} - u_{j-1}}{2\Delta x} = 0 \qquad\qquad , j = 1, 2, ..., N-1 \\
&\frac{du_N}{dt} + a^{(m)}\frac{u_N - u_{N-1}}{\Delta x} = 0
\end{aligned} \tag{2}$$

接下來，我們將(2)寫成矩陣形式：

$$\frac{d\mathbf{u}}{dt} = -a^{(m)}D\mathbf{u} - a^{(m)}\tau^{(m)}(u_0 - g)e_0 \tag{3}$$

其中 $\mathbf{u} = [u_0, u_1, ..., u_N]^T$, $e_0 = [1, 0, ..., 0]^T$ 以及

$$D = \frac{1}{2\Delta x}\begin{bmatrix} -2 & 2 & 0 & \dots & 0 \\ -1 & 0 & 1 & \dots & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & \dots & -1 & 0 & 1 \\ 0 & \dots & 0 & -2 & 2 \end{bmatrix}$$

最後，利用 Runge—Kutta 方法即可求出波方程 u 的數值解。

4 穩定性分析

接下來的章節裡將會分別討論在邊界條件下，以及區間內之系統的穩定情況。

4.1 邊界穩定性分析

在上一個章節中，提出了加入補償項 $-a^{(m)}\tau^{(m)}(u_0 - g)$ 爲左端點提供資訊，然而 $\tau^{(m)}$ 的值尚未給定。因此在本章節中，會將半離散化格式簡化，進而推導出系統在邊界條件的穩定情況，找出合理的 $\tau^{(m)}$ 值。

首先，令 $H = \Delta x\,\mathrm{diag}[\frac{1}{2}, 1, ..., 1, \frac{1}{2}]$，透過矩陣格式(3)可以得到

$$\begin{cases} \mathbf{u}^T H\dfrac{d\mathbf{u}}{dt} = -a^{(m)}\mathbf{u}^T HD\mathbf{u} - a^{(m)}\tau^{(m)}(u_0 - g(t))\mathbf{u}^T He_0 \\ \dfrac{d\mathbf{u}^T}{dt}H^T\mathbf{u} = -a^{(m)}\mathbf{u}^T D^T H^T\mathbf{u} - a^{(m)}\tau^{(m)}(u_0 - g)e_0^T H^T\mathbf{u} \end{cases}$$

$$\overset{+}{\Longrightarrow} \frac{d}{dt}E_D(t) = a^{(m)}\Big(u_0^2 - u_N^2\Big) - a^{(m)}\tau^{(m)}\Delta x u_0(u_0 - g)$$

其中 $E_D(t) = \sum_{j=0}^{N} \frac{1}{c_j}u_j^2$, $\begin{cases} c_0, c_1 = 2 \\ c_j = 1 \; j = 1, ..., N-1 \end{cases}$。

在這裡可以發現 $E_D(t)$ 即爲連續型能量總和 $\int_0^t u^2(x,t)\,dx$ 的離散形式。最後，假設 $\tau^{(m)} = \frac{2}{\Delta x}$，則可以得到

$$\Rightarrow \frac{d}{dt}E_D(t) = a^{(m)}\Big[-u_N^2 - (u_0 - g)^2 + g^2\Big] \le a^{(m)}g^2$$
$$\Rightarrow E_D(t) \le E_D(0) + a^{(m)}\int_0^t g^2\,dt \tag{4}$$

在此，我們發現到系統的總能量會小於系統初始能量加上邊界項提供的能量，這表示系統總能量會隨著時間的經過慢慢被訊息來源 g 所影響。

4.2 整個區間內的穩定性

上一個章節主要是討論系統在邊界條件以及區間交界處的穩定情況，然而區間內的穩定性並沒有被討論。如前面的章節所提及，本報告所討論的波方程是透過 Runge—Kutta 法來進行估計，而在本報告中，不會對其穩定性進行數學上的推導，而是透過實驗來測試其穩定性。

5 實驗

在本章節裡，將會討論在不同的波速、區間大小、CFL 以及起始條件下所模擬出來的波形。

至於 Δx 與 Δt 之間的關係則需要隨著區間而改變。在本實驗裡，在區間 $I^{(m)}$ 中，Δx 與 Δt 之間的關係爲 $\frac{\Delta t}{\Delta x} = a^{(m)}\lambda$，其中 λ 爲 CFL，是用來控制時間步長的變數。

在接下來的實驗裡，如過沒有特別指明，則給定時間 $t = 5$、$\lambda = 1$、x 樣本點總數 $N = 160$、起始函數 $f^{(m)}(x) = \sin(x)$、邊界函數 $g(t) = f^{(1)}(-a^{(1)}t)$。

5.1 多區間波形模擬

首先圖2以及圖3展示波在不同區間裡波速的變化。圖2模擬了波在行經兩個不同介質區間時所形成的波形，其中圖2上下部分的邊界條件皆爲 $g(t) = \sin(-1.6t)$，區間 $I^{(1)} = [0, \pi]$, $I^{(2)} = [\pi, 2\pi]$。

我們可以看到圖2在波速較快的地方波長較長、波速慢的地方波長較短。此現象可以從簡單的數學推導中看出，因爲邊界條件 $g(t)$ 的週期不變 (皆爲 1.6)，因此將根據週期公式 $T = \frac{\ell^{(m)}}{a^{(m)}}$，$\ell^{(m)}$ 爲波在區間 $I^{(m)}$ 內的波長，得出波速會與波長成正比，並與模擬的結果符合。圖3展示了波在四個區間 $[0, \frac{3}{2}\pi, 2\pi, 3\pi, 4\pi]$ 行進時所形成的波形。

5.2 初始條件與邊界條件

接下來討論初始條件、邊界條件以及 τ 對模擬的影響。圖4展示在每個區間的起始條件 $f^{(m)}(x)$ 不同的情況下，波形會如何隨著時間變化。其中模擬條件如(5)所示。

在圖4中發現波形在區間交接處出現上下抖動的現象，隨著時間經過，抖動會趨於平緩，而波形也慢慢被上一個區間的波形所取代。此外，因爲邊界條件 $g(t) = 1$，左側水平的波形逐漸向右傳遞，不難想像隨著時間推移，整個波形會變成一條水平線，此現象符合章節 4.1 裡的結論(4)，系統總能量會隨著時間的經過慢慢被邊界項 g 所影響。

圖5展示給定不同 τ 對模擬結果的影響，其中 τ 分別取 $0, 0.1, 100$、區間爲 $[0, 2\pi]$。在實驗裡發現，τ 值越大，波形受邊界條件的影響就越大，當 $\tau = 0$ 時，模擬程序不會考慮上風的訊息來源; 隨著 τ 取得越大，波形上下抖動的幅度將會越小，當 τ 足夠大時，波形將完全由邊界項所決定。

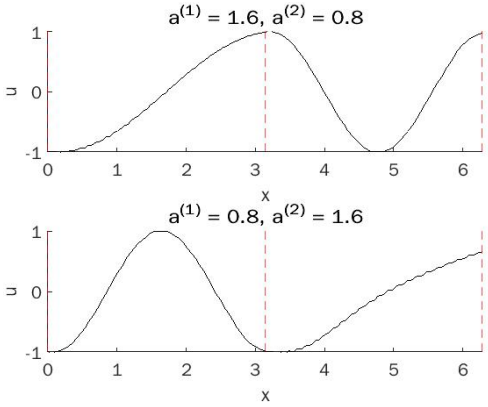

Figure 2: 二區間波形之模擬

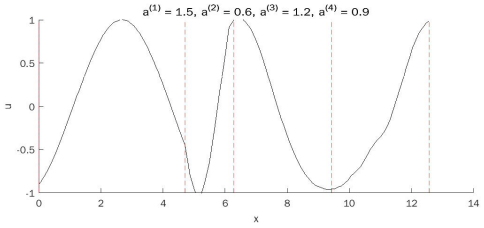

Figure 3: 四區間波形之模擬

$$\begin{cases} a^{(m)} = 1 & m = 1, 2, 3 \\ f^{(1)}(x) = \sin x & x \in [0, \pi] \\ f^{(2)}(x) = \cos x & x \in [\pi, 2\pi] \\ f^{(3)}(x) = \sin x + 0.2 & x \in [2\pi, 3\pi] \\ g(t) = 1 \end{cases} \tag{5}$$

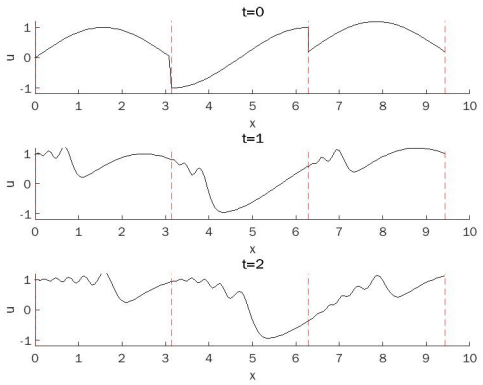

Figure 4: 不連續起始條件與邊界條件

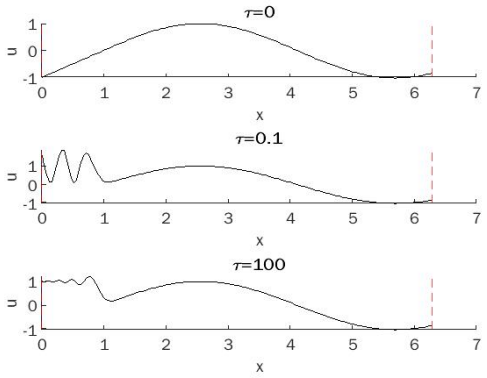

Figure 5: τ 對波形的影響

5.3 CFL 與鋸齒狀波形

圖6展示給定不同的 λ，對模擬出的波形影響不大，其中區間爲 $[0, \frac{1}{2}\pi]$。

在圖7裡，區間爲 $[0, \frac{1}{4}\pi]$，相較於前面其他實驗的區間，此區間的長度較小。在實驗裡，我們發現波形上下震盪並呈現鋸齒狀，而此現象無法透過降低 λ 消除，反而是要透過降低 x 的樣本數 N 來解決。在此有一點需要注意，在本實驗中，x 的樣本數 N 已經給定，因此降低 λ 只會降低時間的步長，不會增加 Δx。因此，在本實驗中我們發現若希望波形平滑，λ 需要與區間大小成正比，也就是樣本點的密度應該爲某個常數。

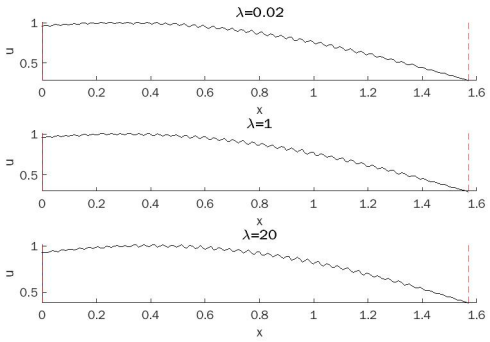

Figure 6: CFL 對波形的影響

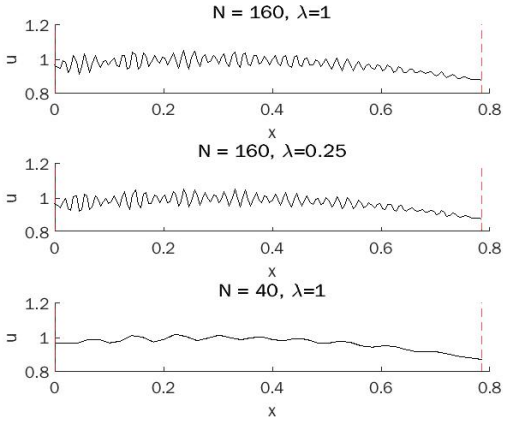

Figure 7: 鋸齒狀波形

6 結語

在實驗裡，我們發現在使用 Runge—Kutta 法估計波方程時，其收斂性似乎不錯，相較於 Upwind-scheme、Downwind-scheme 等格式，Runge—Kutta 法無論 $\frac{\triangle t}{\triangle x}$ 取多少，都沒有發生波形嚴重變形的現象，雖然給定不合適的 λ 值可能導致波形產生鋸齒狀，但是波形還是保留了大致上的形狀。至於在邊界條件裡，τ 的值會影響邊界項對波形的影響程度，但是也沒有發現因爲 τ 過大或過小而導致波形不穩定。

最後有一個地方需要特別討論，在章節 4.1 裡推導出了式(4)：$E_D(t) \leq E_D(0) + a^{(m)} \int_0^t g^2 \, dt$，並且在章節 5.2 裡加以佐證。但是上式並沒有考慮系統所流失的能量。隨著波的傳遞，波會把能量帶出區間，因此到最後初始能量 $E_D(0)$ 將會流失，整個系統的能量會完全由邊界條件 $g(t)$ 所決定。此現象也與現實中的情況相符，在流動的河流中倒下一桶汙水，起初河流會被汙水染黑，但是隨著河流不斷地流動，最後黑色的汙水會消失在視野中，而河流也會恢復清澈。

7 附錄：程式碼

```
%% init
clear
clc
close all

%% two domain same speed
a = [1, 1];
intervals = [0, pi, 2*pi];
N_x = 160;
t_end = 5;
init_fun = {@(x) sin(x), @(x) sin(x)};
bdy_fun = @(t) init_fun{1}(-a(1)*t);
cfl = [1, 1];
tau = [1, 1];

[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

figure(1); clf('reset')
hold on;
plot_u(t, x, u, 5); xline(intervals,'--r');
xlabel('x'); ylabel('u'); ylim([-1, 1]); title('t=5');
hold off
% animation_u(t, x, u);

%% two domain different speed
figure(2); clf('reset')

a = [1.6, 0.8]; intervals = [0, pi, 2*pi]; N_x = 160; t_end = 5; tau = [1, 1];
init_fun = {@(x) sin(x), @(x) sin(x)}; bdy_fun = @(t) init_fun{1}(-a(1)*t); cfl = [1, 1];
[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(2, 1, 1); hold on;
plot_u(t, x, u, 20); xline(intervals,'--r');
xlabel('x'); ylabel('u'); ylim([-1, 1]); title('a^{(1)} = 1.6, a^{(2)} = 0.8');
hold off

a = [0.8, 1.6];
[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(2, 1, 2); hold on;
plot_u(t, x, u, 20); xline(intervals,'--r');
xlabel('x'); ylabel('u'); ylim([-1, 1]); title('a^{(1)} = 0.8, a^{(2)} = 1.6');
hold off

%% four domain different speed
figure(3); clf('reset')
a = [1.5, 0.6, 1.2, 0.9]; intervals = [0, 3/2*pi, 2*pi, 3*pi, 4*pi]; N_x = 160; t_end = 5; tau = [1, 1, 1, 1];
init_fun = {@(x) sin(x), @(x) sin(x), @(x) sin(x), @(x) sin(x)}; bdy_fun = @(t) init_fun{1}(-a(1)*t); cfl = [1, 1, 1, 1];

[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

hold on;
plot_u(t, x, u, 5); xline(intervals,'--r');
xlabel('x'); ylabel('u'); ylim([-1, 1]); title('a^{(1)} = 1.5, a^{(2)} = 0.6, a^{(3)} = 1.2, a^{(4)} = 0.9');
hold off

%% boundaries are not continuous
figure(4); clf('reset')
a = [1, 1, 1]; intervals = [0, pi, 2*pi, 3*pi]; N_x = 160; t_end = 5; tau = [1, 1, 1];
init_fun = {@(x) sin(x), @(x) cos(x), @(x) sin(x)+0.2}; bdy_fun = @(t) 1; cfl=[1, 1, 1];
[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(3, 1, 1);hold on;
plot_u(t, x, u, 0); xline(intervals,'--r');
xlabel('x'); ylabel('u'); ylim([-1.2, 1.2]); title('t=0');
hold off
subplot(3, 1, 2);hold on;
plot_u(t, x, u, 1); xline(intervals,'--r');
xlabel('x'); ylabel('u'); ylim([-1.2, 1.2]); title('t=1');
hold off
subplot(3, 1, 3);hold on;
plot_u(t, x, u, 2); xline(intervals,'--r');
xlabel('x'); ylabel('u'); ylim([-1.2, 1.2]); title('t=2');
hold off
```

```
%% small domain
figure(5); clf('reset')
a = [1]; intervals = [0, pi/4]; N_x = 160; t_end = 5; tau = 1;
init_fun = {@(x) sin(x)}; bdy_fun = @(t) init_fun{1}(-a(1)*t); cfl=1;

[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(3, 1, 1); hold on;
plot_u(t, x, u, 5); xline(intervals,'--r');
xlabel('x'); ylabel('u'); ylim([0.8, 1.2]); title('N = 160, \lambda=1');
hold off

cfl = 0.25;
[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(3, 1, 2); hold on;
plot_u(t, x, u, 5); xline(intervals,'--r');
xlabel('x'); ylabel('u'); ylim([0.8, 1.2]); title('N = 160, \lambda=0.25');
hold off

cfl = 1; N_x = 40;
[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(3, 1, 3); hold on;
plot_u(t, x, u, 5); xline(intervals,'--r');
xlabel('x'); ylabel('u'); ylim([0.8, 1.2]); title('N = 40, \lambda=1');
hold off

%% different tau
figure(6); clf('reset')
a = [1]; intervals = [0, 2*pi]; N_x = 160; t_end = 5; tau=0;
init_fun = {@(x) sin(x)}; bdy_fun = @(t) 1; cfl = 1;

[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(3, 1, 1); hold on;
plot_u(t, x, u, 1); xline(intervals,'--r');
xlabel('x'); ylabel('u'); title('\tau=0');
hold off

tau=0.1;
[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(3, 1, 2); hold on;
plot_u(t, x, u, 1); xline(intervals,'--r');
xlabel('x'); ylabel('u'); title('\tau=0.1');
hold off

tau = 100;
[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(3, 1, 3); hold on;
plot_u(t, x, u, 1); xline(intervals,'--r');
xlabel('x'); ylabel('u'); title('\tau=100');
hold off

%% different CFL
figure(7); clf('reset')
a = [1]; intervals = [0, pi/2]; N_x = 160; t_end = 5; tau = 1;
init_fun = {@(x) sin(x)}; bdy_fun = @(t) init_fun{1}(-a(1)*t); cfl = 0.02;

[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(3, 1, 1);hold on;
plot_u(t, x, u, 20); xline(intervals,'--r');
xlabel('x'); ylabel('u'); title('\lambda=0.02');
hold off

cfl = 1;
[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(3, 1, 2);hold on;
plot_u(t, x, u, 20); xline(intervals,'--r');
xlabel('x'); ylabel('u'); title('\lambda=1');
hold off

cfl = 20;
[t, x, u] = MultidomainWave(init_fun, bdy_fun, a, intervals, cfl, tau, N_x, t_end);

subplot(3, 1,3);hold on;
plot_u(t, x, u, 20); xline(intervals,'--r');
xlabel('x'); ylabel('u'); title('\lambda=20');
hold off
```

```
%% function definition
% Estimate wave equation on multiple domain with different initial
% condition and speed.
% Args:
%   f      {cell<fun>} The functions for initial condition.
%   g            {fun} The function for left boundary condition.
%   a         {vector} The coefficients(speed) of wave equation.
%   intervals {vector} Endpointe of multidomain.
%   cfl       {vector} Controls dx/dt.
%   tau       {vector} Controls the boundary.
%   N_x      {integer} Number of points of x.
%   t_end     {number} Time to stop.
% Returns:
%   t {vector}  Sequence of time.
%   x {vector}  Sequence of x.
%   u {matrix}  Estimated u, size(u) == [length(t), length(x)].
%
function [t, x, u] = MultidomainWave(f, g, a, intervals, cfl, tau, N_x, t_end)
  % initialize x
  intervals_length = diff(intervals);
  n_domain = length(intervals_length); % numbers of domain
  total_domain_length = intervals(n_domain) - intervals(1);

  dt = cfl.*a * intervals_length' / N_x; % May influence the convergence.

  N_x_list = ceil(cfl.*a .* intervals_length / dt); % N_x corresponding to each domain

  N_x = sum(N_x_list); % reset N_x
  index_list = [0, cumsum(N_x_list)]; % The index of x and u that will be used in ensuing code.

  x_list = cell(1, n_domain);  % sequence x corresponding to each domain
  dx_list = zeros(1, n_domain); % dx corresponding to each domain
  for m=1:n_domain
    x0 = intervals(m); % start point of domain_m
    xn = intervals(m+1);% end point of domain_m

    x_list{m} = linspace(x0, xn, N_x_list(m));
    dx_list(m) = (xn - x0) / (N_x_list(m) - 1);
  end

  % initialize t
  dt = repmat(dt, 1, ceil(t_end / dt)); % scalar to vector
  N_t = length(dt);
  t = [0, cumsum(dt)];
  t_end = t(N_t+1); % reset t_end

  % initialize u (equation to estimate)
  D_list = cell(1, n_domain); % matrix list for differential
  e0_list = cell(1, n_domain); % vector list for boundary term
  for m=1:n_domain
    n = N_x_list(m);

    D = circshift(eye(n), -1) - circshift(eye(n), 1);
    D(n, 1) = 0; D(1, n) = 0;
    D(1, 1:2) = [-2, 2];
    D(n, n-1:n) = [-2, 2];
    D = D / (2*dx_list(m));
    D_list{m} = D;

    e0 = zeros(n, 1);
    e0(1) = 1;
    e0_list{m} = e0;
   end

   u_list = cell(1, n_domain);
   for m = 1:n_domain
     u_list{m} = f{m}(x_list{m});
   end

   % In order to run ode23 ode45, u needs to be flattened into a vector
   % Note that each sequence of u_list and x_list contain the endpoint of
   % the corrsponding domain.
   u = cell2mat(u_list);
   x = cell2mat(x_list);
   % solve ode
   [t, u] = ode45(@(t, u) vdp1(t, u, n_domain, index_list, D_list, e0_list, dx_list, a, g, tau), t, u);

   % Remove the redundant(overlapping) boundary term in u.
   u(:, N_x_list(2:n_domain)) = [];
   x(N_x_list(2:n_domain)) = [];
end
```

```
% Used by MultidomainWave
function u = vdp1(t, u, n_domain, index_list, D_list, e0_list, dx_list, a_list, bdy_g, tau)
  g = bdy_g(t);

  for m=1:n_domain
    j0 = index_list(m) + 1; jn = index_list(m+1);
    D = D_list{m};   e0 = e0_list{m};
    dx = dx_list(m);
    a = a_list(m);

    tmp_g = u(jn); % For boundary condition of next iteration.
    u(j0:jn) = -a*D*u(j0:jn) - a*tau(m)*(2/dx)*e0*(u(j0) - g); % "The Main Equation"

    g = tmp_g; % For boundary condition of next iteration.
  end
end

function u0 = get_u_at_t(t, u, t_now)
  idx = max(1, length(t(t<=t_now))); % Since t is increasing
  u0 = u(idx,:);
end

function [] = plot_u(t, x, u, t_now)
  u0 = get_u_at_t(t, u, t_now);
  plot(x, u0, 'Color', 'black');
end

function [] = animation_u(t, x, u)
  n = length(t);
  % dt = [0; diff(t)];
  for j = 1:n
    % pause(dt(j)) % too slow

    t_now = t(j);
    plot_u(t, x, u, t_now);

    figure(1)
    drawnow
  end
end
```