Backward Difference Method

BDM_12_2_A(alpha,h,0.0005,t)

```
x_w_u_e =

         0.1     0.002342    0.0022224   0.00011954
         0.2    0.0044547    0.0042273   0.00022738
         0.3    0.0061313    0.0058184   0.00031296
         0.4    0.0072078    0.0068399   0.00036791
         0.5    0.0075787    0.0071919   0.00038684
         0.6    0.0072078    0.0068399   0.00036791
         0.7    0.0061313    0.0058184   0.00031296
         0.8    0.0044547    0.0042273   0.00022738
         0.9     0.002342    0.0022224   0.00011954
```

alpha = 1;

h = 0.1;

v = 0.01;

t = 0.5;

BDM_12_2_A(alpha,h,v,t)

```
x_w_u_e =
```

```
        0.1     0.002898    0.0022224    0.0006756
        0.2    0.0055124    0.0042273    0.0012851
        0.3    0.0075871    0.0058184    0.0017688
        0.4    0.0089192    0.0068399    0.0020793
        0.5    0.0093782    0.0071919    0.0021863
        0.6    0.0089192    0.0068399    0.0020793
        0.7    0.0075871    0.0058184    0.0017688
        0.8    0.0055124    0.0042273    0.0012851
        0.9     0.002898    0.0022224    0.0006756
```

Source Code

language-matlab

```matlab
clear;
clc;
close all;

format shortg

alpha = 1;
h = 0.1;
v = 0.01;
t = 0.5;

BDM_12_2_A(alpha,h,0.0005,t)
BDM_12_2_A(alpha,h,v,t)
```

```
function BDM_12_2_A(al,h,v,t)
    m = 1/h;
    iterations = t/v;
%     s = al^2*v/h^2;
    x = zeros(m-1,1);

    %go horizontal
    for i =1:m-1
        x(i) = i*h;
    end

    w = sin(pi*x);
    wn = zeros(m-1,1);

    %go vertical
    [c,d,e] = compute_tridiag(wn, al, v, h);

    for j = 1:iterations
        wn = gaussian_diag(c,d,e, w);
        w = wn;
    end

    %compute true solution
    u = true_solution(x,t);
    e = abs(u-w);
    x_w_u_e = [x,w,u,e]
end
```

```
function [u] = true_solution(x,t)
    u = exp(-pi^2*t)*sin(pi*x);
end

function [c,d,e] = compute_tridiag(W, alpha, v,
h)
    n = length(W);
    c = zeros(n,1);
    d = zeros(n,1);
    e = zeros(n,1);

    lambda = alpha^2*(v/h^2);

    d(1) = (1 + (2*lambda));
    e(1) = -lambda;

    for i = 2:n-1
        c(i) = -lambda;
        d(i) = (1 + (2*lambda));
        e(i) = -lambda;
    end

    c(n) = -lambda;
    d(n) = (1 + (2*lambda));
end
```

```
function W = gaussian_diag(c,d,e,b)
    n = length(d);
    W = zeros(n,1);
    for k = 2:n
        mult = -c(k)/d(k-1);
        d(k) = mult*e(k-1) + d(k);
        b(k) = mult*b(k-1) + b(k);
    end
    W(n) = b(n)/d(n);
    for k = n-1:-1:1
        W(k) = (b(k)-e(k)*W(k+1))/d(k);
    end

end
```