Problem P30: Consider the "θ-methods" for $u' = f(t,u)$, namely

$$U^{n+1} = U^n + k[(1-\theta)f(t_n, U^n) + \theta f(t_{n+1}U^{n+1})]$$

where $0 \leq \theta \leq 1$ is a fixed parameter.

a. Cases $\theta = 0, 1/2, 1$ are all familiar methods. Name them.

Solution:

$$\theta = 0: \qquad U^{n+1} = U^n + k[f(t_n, U^n)] \qquad \rightarrow \qquad \text{Forward Euler}$$
$$\theta = 1/2: \qquad U^{n+1} = U^n + \frac{k}{2}[f(t_n, U^n) + f(t_{n+1}U^{n+1})] \qquad \rightarrow \qquad \text{Trapezoid Method}$$
$$\theta = 1: \qquad U^{n+1} = U^n + k[f(t_{n+1}U^{n+1})] \qquad \rightarrow \qquad \text{Backward Euler}$$

b. Find the (absolute) stability regions for $\theta = 0, 1/4, 1/2, 3/4, 1$

Solution:
Applying the test equation $u' = \lambda u$ to our general scheme we get,

$$U^{n+1} = U^n + k[(1-\theta)\lambda U^n + \theta\lambda U^{n+1}].$$

Solving for $U^{n+1} = R(z)U^n$ we get,

$$\begin{aligned}
U^{n+1} &= U^n + k[(1-\theta)\lambda U^n + \theta\lambda U^{n+1}] \\
U^{n+1} &= U^n + (1-\theta)k\lambda U^n + \theta k\lambda U^{n+1} \\
U^{n+1} - \theta k\lambda U^{n+1} &= U^n + (1-\theta)k\lambda U^n \\
(1-\theta k\lambda)U^{n+1} &= (1 + (1-\theta)k\lambda)U^n \\
U^{n+1} &= \frac{(1+(1-\theta)k\lambda)}{(1-\theta k\lambda)}U^n
\end{aligned}$$

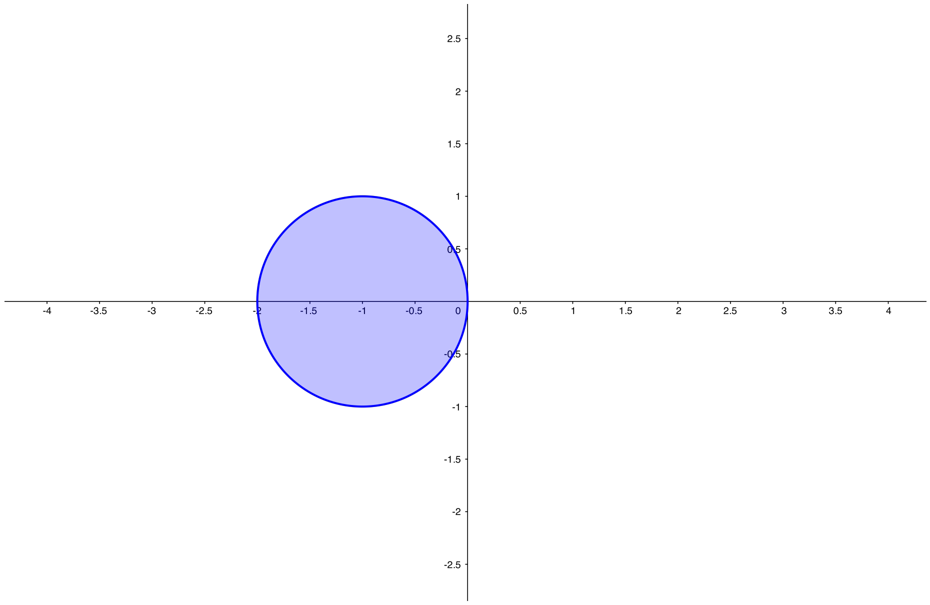
Plotting the stability region, for $\theta = 0$ first note that,

$$R(z) = \frac{(1+(1-\theta)k\lambda)}{(1-\theta k\lambda)} = (1 + k\lambda) = 1 + z$$

Therefore the stability region is given by

$$\begin{aligned}
|1 + z| &\leq 1 \\
|1 + x + iy| &\leq 1 \\
(1+x)^2 + y^2 &\leq 1
\end{aligned}$$

Figure 1: Stability Region for $\theta = 0$ Method.



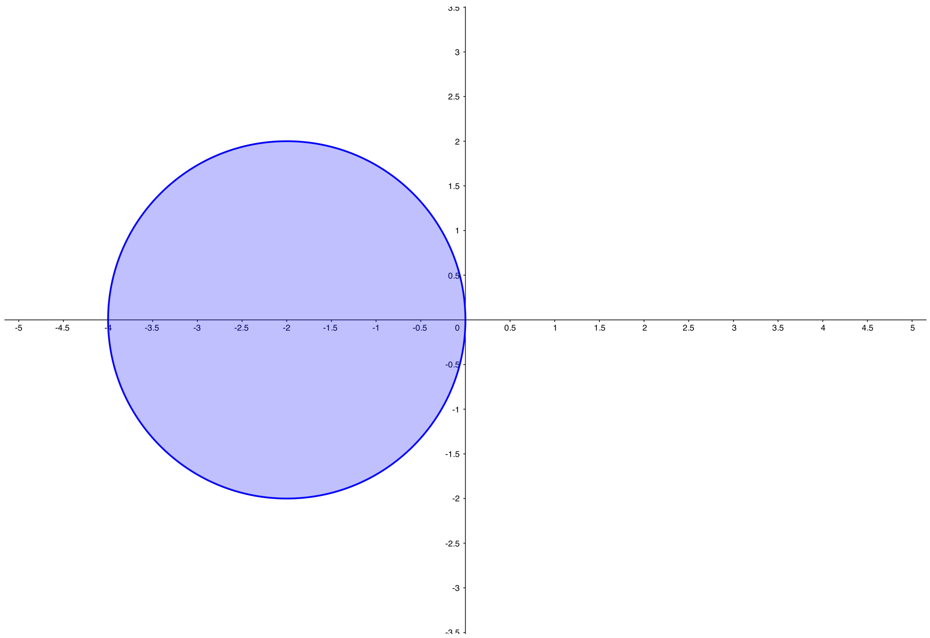
Plotting the stability region, for $\theta = 1/4$ first note that,

$$R(z) = \frac{(1 + \frac{3}{4}k\lambda)}{(1 - \frac{1}{4}k\lambda)} = \frac{(1 + \frac{3}{4}z)}{(1 - \frac{1}{4}z)}$$

Therefore the stability region is given by

$$\begin{aligned}
\left|\frac{(1 + \frac{3}{4}z)}{(1 - \frac{1}{4}z)}\right| &\leq 1 \\
\left|\frac{(1 + \frac{3}{4}(x+iy))}{(1 - \frac{1}{4}(x+iy))}\right| &\leq 1 \\
\left|1 + \frac{3}{4}x + i\frac{3}{4}y\right| &\leq \left|1 - \frac{1}{4}x + i\frac{1}{4}y\right| \\
(1 + \frac{3}{4}x)^2 + (\frac{3}{4}y)^2 &\leq (1 - \frac{1}{4}x)^2 + (\frac{1}{4}y)^2 \\
x^2 + 4x + y^2 &\leq 0 \\
(x^2 + 4x + 4) + y^2 &\leq 4 \\
(x+2)^2 + y^2 &\leq 2^2
\end{aligned}$$

Figure 2: Stability Region for $\theta = 1/4$ Method.



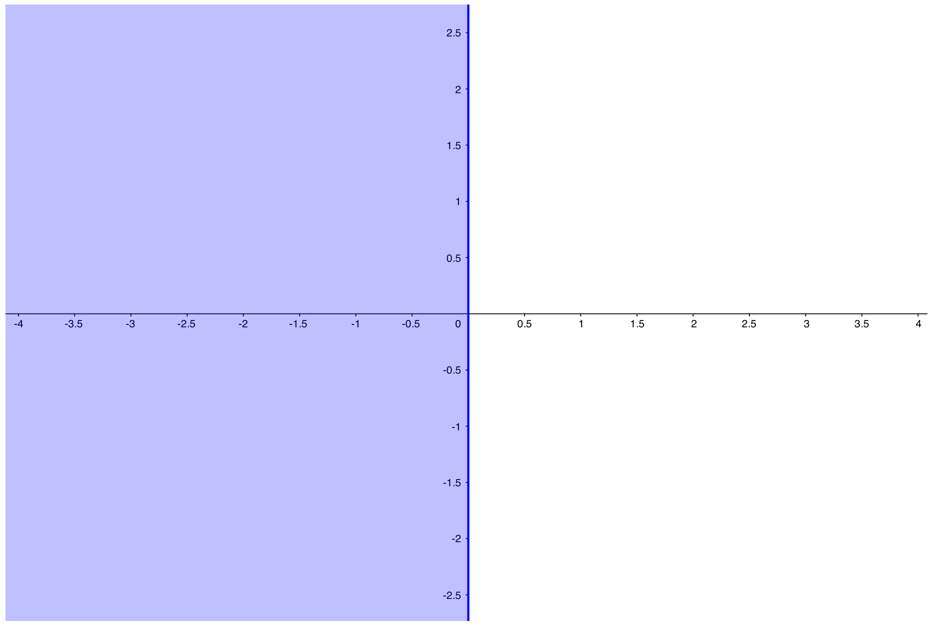
Plotting the stability region, for $\theta = 1/2$ first note that,

$$R(z) = \frac{(1 + \frac{1}{2}k\lambda)}{(1 - \frac{1}{2}k\lambda)} = \frac{(1 + \frac{1}{2}z)}{(1 - \frac{1}{2}z)}$$

Therefore the stability region is given by

$$\begin{gathered}
\left|\frac{(1 + \frac{1}{2}z)}{(1 - \frac{1}{2}z)}\right| \leq 1 \\
\left|\frac{(1 + \frac{1}{2}(x+iy))}{(1 - \frac{1}{2}(x+iy))}\right| \leq 1 \\
\left|1 + \frac{1}{2}x + i\frac{1}{2}y\right| \leq \left|1 - \frac{1}{2}x + i\frac{1}{2}y\right| \\
(1 + \frac{1}{2}x)^2 + (\frac{1}{2}y)^2 \leq (1 - \frac{1}{2}x)^2 + (\frac{1}{2}y)^2 \\
\frac{1}{4}x^2 + x + 1 + \frac{1}{4}y^2 \leq \frac{1}{4}x^2 - x + 1 + \frac{1}{4}y^2 \\
x \leq -x
\end{gathered}$$

Figure 3: Stability Region for $\theta = 1/2$ Method.



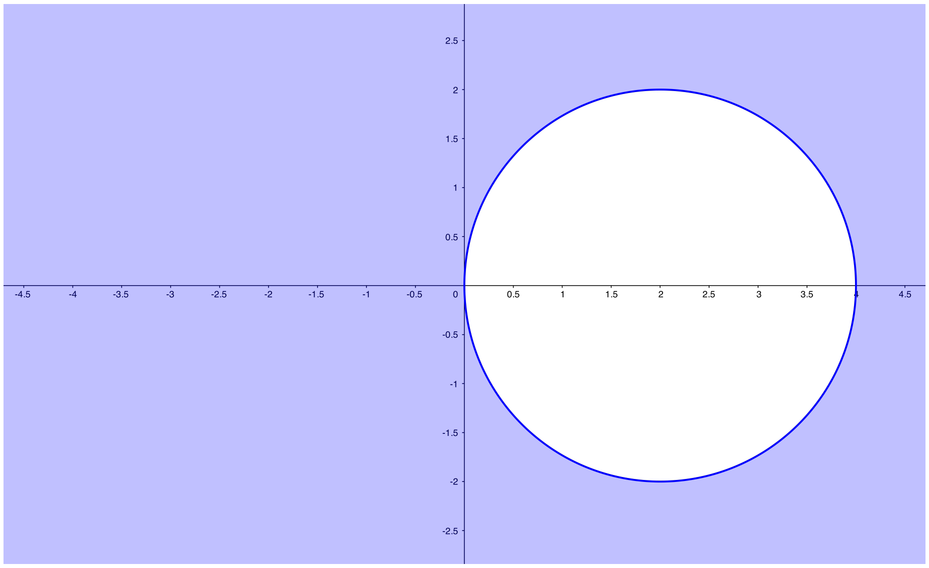
Plotting the stability region, for $\theta = 3/4$ first note that,

$$R(z) = \frac{(1 + \frac{1}{4}k\lambda)}{(1 - \frac{3}{4}k\lambda)} = \frac{(1 + \frac{1}{4}z)}{(1 - \frac{3}{4}z)}$$

Therefore the stability region is given by

$$\begin{aligned}
\left|\frac{(1 + \frac{1}{4}z)}{(1 - \frac{3}{4}z)}\right| &\leq 1\\
\left|\frac{(1 + \frac{1}{4}(x+iy))}{(1 - \frac{3}{4}(x+iy))}\right| &\leq 1\\
\left|1 + \frac{1}{4}x + i\frac{1}{4}y\right| &\leq \left|1 - \frac{3}{4}x + i\frac{3}{4}y\right|\\
(1 + \frac{1}{4}x)^2 + (\frac{1}{4}y)^2 &\leq (1 - \frac{3}{4}x)^2 + (\frac{3}{4}y)^2\\
y^2 - 4x + x^2 &\geq 0\\
y^2 + 4 - 4x + x^2 &\geq 4\\
y^2 + (x-2)^2 &\geq 2^2
\end{aligned}$$

Figure 4: Stability Region for $\theta = 3/4$ Method.



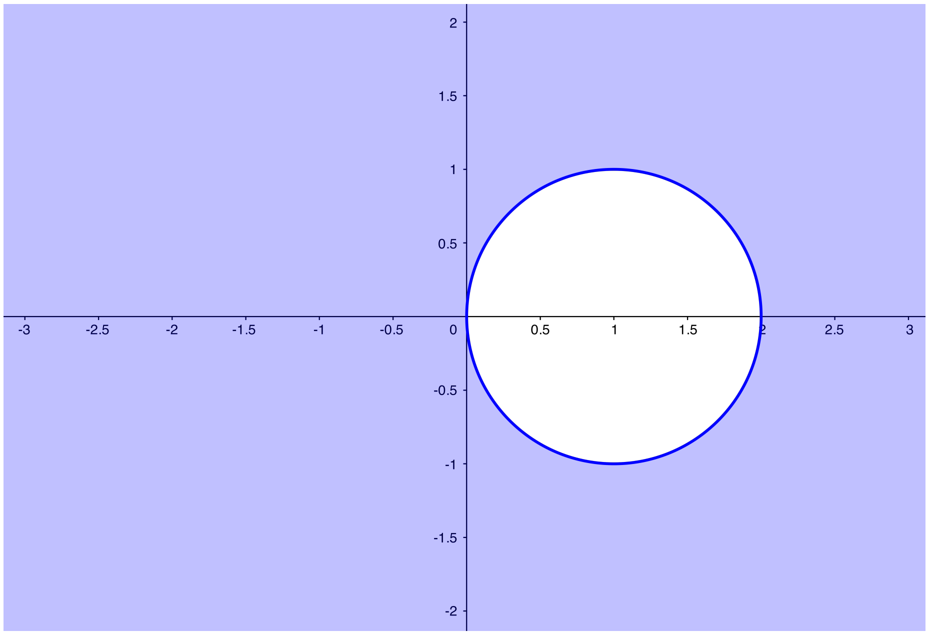
Plotting the stability region, for $\theta = 1$ first note that,

$$R(z) = \frac{(1 + (1-\theta)k\lambda)}{(1-\theta k\lambda)} = \frac{1}{1-k\lambda} = \frac{1}{1-z}$$

Therefore the stability region is given by,

$$\begin{aligned} \frac{1}{1-z} &\leq 1 \\ \frac{1}{|1-x-iy|} &\leq 1 \\ |1-x-iy| &\geq 1 \\ (1-x)^2 + y^2 &\geq 1 \end{aligned}$$

Figure 5: Stability Region for $\theta = 1$ Method.



c. Show that the θ-methods are A-stable for $\theta \geq 1/2$.

Solution:
Note that we have already show that for $\theta = 1/2$ the θ-method is A-stable. Let $\theta > 1/2$

$$R(z) = \frac{(1 + (1-\theta)k\lambda)}{(1-\theta k\lambda)} = \frac{(1 + (1-\theta)z)}{(1-\theta z)}.$$

Therefore the stability region is given by

$$\begin{aligned}
\left|\frac{(1 + (1-\theta)z)}{(1-\theta z)}\right| &\leq 1 \\
\frac{|1 + (1-\theta)z|}{|1-\theta z|} &\leq 1 \\
|1 + (1-\theta)z| &\leq |1-\theta z| \\
|1 + (1-\theta)(x+iy)| &\leq |1-\theta(x+iy)| \\
|1 + (1-\theta)x + i(1-\theta)y| &\leq |1-\theta x + i\theta y| \\
(1 + (1-\theta)x)^2 + (1-\theta)^2y^2 &\leq (1-\theta x)^2 + \theta^2 y^2
\end{aligned}$$

Expanding out and collecting like terms, since $\theta > 1/2$ we find that,

$$
\begin{aligned}
2x + (1-2\theta)x^2 + (1-2\theta)y^2 &\leq 0 \\
\frac{2}{(1-2\theta)}x + x^2 + y^2 &\geq 0 \\
\left(\frac{1}{(1-2\theta)}\right)^2 + \frac{2}{(1-2\theta)}x + x^2 + y^2 &\geq \left(\frac{1}{(1-2\theta)}\right)^2 \\
\left(x + \frac{1}{(1-2\theta)}\right)^2 + y^2 &\geq \left(\frac{1}{(1-2\theta)}\right)^2
\end{aligned}
$$

Note that since $\theta > 1/2$ the term $\frac{1}{(1-2\theta)} < 0$ and therefore, for some $r > 0$ our stability region looks like,

$$(x-r)^2 + y^2 \geq (r)^2$$

which will always be A-stable.

Problem P31: Consider this Runge-Kutta method, a one-step and implicit interpretation of the multistep midpoint method:

$$
\begin{aligned}
U^* &= U^n + \frac{k}{2}f(t_n + k/2, U^*) \\
U^{n+1} &= U^n + kf(t_n + k/2, U^*)
\end{aligned}
$$

The first stage is backward Wuler to dertermin an approximation to the value at the midpoint in time. The second stage is a midpoint method using this value.

a. Determine the order of accuracy of this method. That is, compute the truncation error accurately enough to know the power p in $\tau = O(k^p)$.

Solution:
First we will compute the one-step error $\mathcal{L}^*$ for the first equation. Suppose $U^n = u(t_n)$ and $U^* = u(t_*)$, and not that by substitution we get,

$$U^* = u(t_n) + \frac{k}{2}f(u(t_*)).$$

Applying the exact solution, and expanding $u(t_n)$ around t_* we get the following,

$$
\begin{aligned}
U^* &= \left(u(t_*) - \frac{k}{2}u'(t_*) + O(k^2)\right) + \frac{k}{2}u'(t_*) \\
U^* &= u(t_*) + O(k^2).
\end{aligned}
$$

This means that $u(t_*)$ serves as a second order accurate approximation for U^*. Proceeding to compute the truncation error, note that we can reorder the second equation and our truncation error for t^* comes from,

$$\tau^* = \frac{u(t_{n+1}) - u(t_n)}{k} - f(t_*, U^*)$$

Applying our approximation for U^* evaluated at t^* we get $f(t_*, U^*) = f(u(t_*)+O(k^2))$, and substitution of the exact solution gives $f(t_*, U^*) = u'(t_*) + O(k^2)$. Expanding $u(t_{n+1})$ and $u(t_n)$ about t_* we get,

$$\begin{aligned}\tau^* = \frac{1}{k}\Bigg(&\left(u(t_*) + \frac{1}{2}ku'(t_*) + \frac{1}{8}k^2u''(t_*) + O(k^3)\right) - \\ &\left(u(t_*) - \frac{1}{2}ku'(t_*) + \frac{1}{8}k^2u''(t_*) + O(k^3)\right)\Bigg) - \\ &(u'(t_*) + O(k^2))\end{aligned}$$

$$\begin{aligned}\tau^* &= \frac{1}{k}\left(ku'(t_*) + O(k^3)\right) - (u'(t_*) + O(k^2)), \\ &= O(k^2).\end{aligned}$$

b. Determine the stability region. Is this method A-stable? Is it L-stable?

Solution:
Applying the test equation $u' = \lambda u$ to our scheme we get the following equations,

$$U^* = U^n + \frac{1}{2}k\lambda U^*$$

$$U^{n+1} = U^n + k\lambda U^*$$

Solving for our function $R(z)$ we get the following,

$$\begin{aligned}U^{n+1} &= \left(1 + \frac{k\lambda}{1 - \frac{1}{2}k\lambda}\right)U^n, \\ U^{n+1} &= \left(\frac{1 + \frac{1}{2}k\lambda}{1 - \frac{1}{2}k\lambda}\right)U^n,\end{aligned}$$

$$R(z) = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z}.$$

This will result in the same stability region as was plotted for $\theta = 1/2$ of the θ-methods in the previous problem. This method is A-stable. This method is not L-stable, since clearly $\lim_{z\to\infty} |R(z)| \to 1$ and not 0.

Problem P32: Reproduce Table 7.1. In particular, consider the scalar ODE IVP,

$$u'(t) = \lambda(u(t) - \cos(t)) - sin(t), \qquad u(0) = 1$$

with the particular value $\lambda = -2100$. Use an implementation of forward Euler, to compute approximations of $u(T)$ for $T = 2$, for the given values of k, and report the final-time numerical errors. $|U^n - u(T)|$ as in the Table. Confirm by this experiment that there is a critical value of k around 0.00095 where the error finally drops from enormous values to something comparable to, then much smaller than, the solution magnitude itself.

Solution:
Consider the following Matlab code, table71.m which generates the desired table

Code:

```
f = @(t, u) -2100.*(u - cos(t)) - sin(t);
% Step sizes from table
k = [.001, .000976, .000952, .0008, .0004]';
% Convert to number of steps
N = floor(2./k);
% Exact Solution
exact = cos(2);
tFinalError = zeros(5, 1);

for i = 1:5
    [tt,zz] = feulerED(f,1,0,2, N(i));
    tFinalError(i) = abs(zz(end) - exact);
end
% Final Table
FinalTable = [k tFinalError];
table(FinalTable)
```

Console:

```
>> table71

          FinalTable
     ______________________

       0.001      1.4525e+76
    0.000976      3.9534e+36
    0.000952      3.2109e-07
      0.0008       7.923e-08
      0.0004      3.9603e-08
```

Problem P33: For a famously stiff problem, consider the heat PDE,

$$u_t = u_{xx}$$

Here $u(t, x)$ is the temperature in a rod of length one ($0 \leq x \leq 1$) and we set boundary temperatures to zero. For an initial temperature distribution we set one part hotter than the rest:

$$u(0, x) = \begin{cases} 1, & 0.25 < x < .5, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Suppose we seek $u(1, x)$, i.e. we set $t_f = 1$. We apply the method of lines(1). That is we discretized the spatial (x) derivatives using the notation from Chapter 2. Specifically, use $m + 1$ sub-interval, let $h = 1/(m + 1)$ and let $x_j = jh$ for $j = 0, 1, 2, \ldots, m + 1$. Now $U_j(t) \approx u(t, x_j)$. By eliminating unknowns $U_0 = 0$ and $U_{m+1} = 0$, and keeping the time derivatives as ordinary derivatives we get a linear ODE system of dimensions m,

$$U(t)' = AU(t) \tag{2}$$

where $U(t) \in \mathbb{R}^m$ and A is exactly the matrix in (2.10). For a given m, note $U(0)$ is computed from the above formula for $u(0, x)$.

a. Implement both forward and backward Euler on (2). Use backslash for linear solve.

Solution:

Code:

```
function [tt,zz] = HEATbeulerED(m,eta,t0,tf,N)
% HEATBACKWARDEULER  Solve
%    u_t = u_xx ,  u(0, x) = eta
% for u(t, x) on the interval [t0,tf] with N steps in time
% and m+2 steps in space. Backward Euler in time via method of lines.
%
% Usage: [tt,zz] = HEATbeulerED(m,eta,t0,tf,N)

% Compute step size in space
h = 1/(m + 1);

% Generate matrix for system of IVPs in time
A = (1/h^2).*(diag(((- 2)*ones(m, 1))) ...
    + diag(ones(m - 1, 1), 1) ...
    + diag(ones(m - 1, 1), -1));

% Define step in time.
dt = (tf - t0) / N;
% Sparse representation of
% backward euler system,
% U(n+1) = U(n) + kAU(n+1)
% (1 - kA)U(n+1) = U(n)
fU = sparse(diag((ones(m, 1))) - dt.*A);

tt = t0:dt:tf;          % row vector of times
eta = eta(:);           % force to be column vector
s = length(eta);
zz = zeros(s,N+1);      % jth column is U at t_{j-1}
zz(:,1) = eta;

for j = 1:N         % Backward Euler Solve
    zz(:,j+1) = (fU)\zz(:,j);
end
```

Code:

```
function [tt,zz] = HEATfeulerED(m,eta,t0,tf,N)
% HEATFORWARDEULER  Solve
%   u_t = u_xx ,  u(0, x) = eta
% for u(t, x) on the interval [t0,tf] with N steps in time
% and m+2 steps in space. Forward Euler in time via method of lines.
%
% Usage: [tt,zz] = HEATfeulerED(m,eta,t0,tf,N)

% Compute step size in space
h = 1/(m + 1);
% Generate matrix for system of IVPs in time
A = (1/h^2).*(diag(((- 2)*ones(m, 1))) + ...
     diag(ones(m - 1, 1), 1) + ...
     diag(ones(m - 1, 1), -1));

% Define inline function for
% Solving system of IVP U(t)' = AU(t)
f = @(t, u) A*u;

% Define step in time.
dt = (tf - t0) / N;
tt = t0:dt:tf;          % row vector of times
eta = eta(:);           % force to be column vector
s = length(eta);
zz = zeros(s,N+1);      % jth column is U at t_{j-1}
zz(:,1) = eta;

for j = 1:N             % forward Euler is (5.19) in LeVeque
    zz(:,j+1) = zz(:,j) + dt .* f(tt(j),zz(:,j));
end
```

b. Now consider the $m = 99$ case, so $h = .01$ and let $k = t_f/N = 1/N$ be the time step length. For BE, compute and show the solution using $N = 100$ time steps. For FE, $N = 100$ will generate extraordinary explosion.

Determine the largest-possible absolutely stable time step k from the eigenvalues of A and the the stability region of FE. Finally, compare the computation costs of the two runs, by counting floating-point multiplications. You will conclude that an implicit method is indeed effective in this case.

Solution:
Recall that for a linear, constant coefficient, system of IVPs as described by (2), if A is diagonalizable then the system can be decoupled in to a system of test equations.

$$\begin{aligned}
U(t)' &= AU(t)\\
U(t)' &= R\Lambda R^{-1}U(t)\\
R^{-1}U(t)' &= \Lambda R^{-1}U(t)\\
(R^{-1}U)(t)' &= \Lambda(R^{-1}U)(t)\\
W(t)' &= \Lambda W(t)
\end{aligned}$$

Let $W(t) = (R^{-1}U)(t)$ and note that $R^{-1}U(t)' = (R^{-1}U)(t)'$ since R is constant. Recall that the eigenvalues of A are well documented and are given by $\lambda_i = \frac{2}{h}(\cos(i\pi h) - 1)$. Applying our test equations to the Forward Euler scheme we get that in order to achieve absolute convergence given spacing $h = .01$,

$$\begin{aligned}
|1 + k\lambda_i| &\leq 1 \\
|1 + k\frac{2}{h^2}(\cos(i\pi h) - 1)| &\leq 1 \\
|1 + k\frac{2}{.0001}(\cos(i\pi .01) - 1)| &\leq 1
\end{aligned}$$

Note that $\cos(i\pi.01) \approx -1$ and therefore,

$$\begin{aligned}
|1 + k\frac{2}{.0001}(-2)| &\leq 1, \\
|1 - k\frac{4}{.0001}| &\leq 1, \\
-1 \leq 1 - k\frac{4}{.0001} &\leq 1, \\
2 \geq k\frac{4}{.0001} &\geq 0, \\
\frac{1}{20000} \geq k &\geq 0.
\end{aligned}$$

Therefore in order to achieve absolute convergence using forward euler we must take at lease 20,000 times steps.

Considering the floating-point multiplications required we find that for each of the 20,000 timesteps we need to perform the $AU(t)$ matrix multiplication, and since A is an $m \times m$ tridiagonal matrix we get a total of $20,000(3m)$ multiplications. However since backward euler is A-stable it will produce a quality solution in as little as 100 time steps and since it takes only $5m$ multiplications to solve a comparable tridiagonal system, backward euler is substantially better suited for this problem.

Here is a script which exports both forward euler and backward euler solutions.
Code:

```
m = 99;
N = 100;
eta = ((linspace(0, 1, m+2)>.25).*(linspace(0, 1, m+2)<.5))';
eta = eta(2:(length(eta) - 1));
[tt,zz] = HEATfeulerED(m,eta,0,1,N);

xx = linspace(0, 1, m+2);
xx = xx(2:(length(xx) - 1));


p = plot(xx,zz(:,1));
axis([0 1 0 1])
for i = 1:200:N
    p.YData = zz(:,i);
    exportgraphics(gca,"FEheat.gif","Append",true)
end




[btt,bzz] = HEATbeulerED(m,eta,0,1,N);
pb = plot(xx,bzz(:,1));
axis([0 1 0 1])
for i = 1:200:N
    pb.YData = bzz(:,i);
    exportgraphics(gca,"BEheat.gif","Append",true)
end
```

Here is a link to the compiled gifs,
https://github.com/StefanoFochesatto/NumericalDifferentialEquation

backward