A Numerical Solution to a Second Order Ordinary Differential Equation

Hunter, Marcus Tiger, Matthew Wigfield, Jacob

November 28, 2015

Contents

1. Introduction

The authors were tasked by the client with finding the solution to the following family of differential equations

$$\begin{cases} -u''(x) + cu(x) = f(x) \\ 0 \leq x \leq 1 \\ u(0) = \epsilon \\ u(1) = \delta. \end{cases}$$

Additionally, the client has also requested to be provided with a means of plotting the solution once obtained.

Throughout this report, the above family of differential equations together with the interval of definition and initial conditions will be represented by $Lu = f$.

Assumptions were placed on this family so that $c \in \mathbb{R}$ with $c > 0$ and $f \in C^k([0,1])$ for sufficiently large k so that f is relatively well-behaved on the defined interval.

In this report we will detail the analytical solution to this family of differential equations showing the the above problem is well-posed and explain why this solution is not amenable to practical use. We therefore provide a numerical scheme to approximate the solution to the family of differential equations and examine the convergence, consistency and stability of the numerical scheme. Using the solution provided by the numerical scheme, we then explore the different options for plotting the solution.

2. Analytical Solution

The family of differential equations $Lu = f$ represents a second order linear differential equation and therefore well-known techniques can be used to find the solution $u(x)$.

The solution $u(x)$ is given by $u(x) = u_h(x) + u_p(x)$ where $u_h(x)$ is the solution to the homogeneous equation $-u''(x) + cu(x) = 0$ and $u_p(x)$ is a particular solution of $-u''(x) + cu(x) = f(x)$.

To find the homogeneous solution, note that the characteristic equation of this family of differential equations is given by $-m^2 + c = 0$, the roots of which are $m_1 = \sqrt{c} = \omega$ and $m_2 = -\sqrt{c} = -\omega$. Note that since $c > 0$, these roots are real and distinct suggesting that the homogeneous solution is given by

$$u_h(x) = c_1 e^{\omega x} + c_2 e^{-\omega x}. \tag{1}$$

To find the particular solution, we assume the particular solution is of the form $u_p(x) = \kappa(x)e^{\omega x}$ for some unknown function $\kappa(x)$. Thus,

$$u_p''(x) = \kappa''(x)e^{\omega x} + 2\omega\kappa'(x)e^{\omega x} + \omega^2\kappa(x)e^{\omega x}$$

and substituting the above into the original differential equation $Lu = f$ with $u_p(x) = \kappa(x)e^{\omega x}$ we have

$$\kappa''(x) + 2\omega\kappa'(x) = -f(x)e^{-\omega x}. \tag{2}$$

Making the substitution $\lambda(x) = \kappa'(x)$ into (2) we can reduce the above second order linear differential equation into the first order linear differential equation

$$\lambda'(x) + 2\omega\lambda(x) = -f(x)e^{-\omega x}. \tag{3}$$

The homogeneous solution to this first order differential equation is given by $\lambda_h(x) = c_3e^{-2\omega x}$ suggesting the particular solution to the first order differential equation is of the form $\lambda_p(x) = \mu(x)e^{-2\omega x}$.

Repeating the same process as above, we see that

$$\lambda_p'(x) = \mu'(x)e^{-2\omega x} - 2\omega\mu(x)e^{-2\omega x}$$

and substituting into (3) with $\lambda_p(x) = \mu(x)e^{-2\omega x}$ we find that the first order linear differential equation becomes the separable first order differential equation

$$\mu'(x) = -f(x)e^{\omega x}.$$

We readily see the solution to the above differential equation is given by

$$\mu(x) = -\int_0^x f(r)e^{\omega r}dr.$$

As $\kappa'(x) = \lambda_p(x) = \mu(x)e^{-2\omega x}$, we deduce that

$$\kappa(x) = -\int_0^x e^{-2\omega s}\left[\int_0^s f(r)e^{\omega r}dr\right]ds$$

and

$$u_p(x) = \kappa(x)e^{\omega x} = -e^{\omega x}\int_0^x e^{-2\omega s}\left[\int_0^s f(r)e^{\omega r}dr\right]ds. \tag{4}$$

Combining the homogeneous solution (1) and the particular solution (4) we have that the general solution to $Lu = f$ is given by

$$\begin{aligned} u(x) &= u_h(x) + u_p(x) \\ &= c_1e^{\omega x} + c_2e^{-\omega x} - e^{\omega x}\int_0^x e^{-2\omega s}\left[\int_0^s f(r)e^{\omega r}dr\right]ds. \end{aligned} \tag{5}$$

Using the boundary values provided in $Lu = f$, the general solution is specified by the system of linear equations

$$\begin{aligned} u(0) &= c_1 + c_2 = \epsilon \\ u(1) &= c_1e^{\omega} + c_2e^{-\omega} - e^{\omega}\int_0^1 e^{-2\omega s}\left[\int_0^s f(r)e^{\omega r}dr\right]ds = \delta. \end{aligned}$$

The solution to this system in terms of the unknowns c_1 and c_2 is given by

$$c_1 = \frac{\epsilon e^{-\omega} - \delta - e^{\omega} \int_0^1 e^{-2\omega s} \left[\int_0^s f(r) e^{\omega r} dr\right] ds}{e^{-\omega} - e^{\omega}}$$
$$c_2 = \frac{-\epsilon e^{\omega} + \delta + e^{\omega} \int_0^1 e^{-2\omega s} \left[\int_0^s f(r) e^{\omega r} dr\right] ds}{e^{-\omega} - e^{\omega}}.$$

Using these constants in the general solution (5) gives us the unique analytical solution to the family of differential equation $Lu = f$. Furthermore, we deduce that the problem is in fact well-posed.

From this solution, we must make the following additional assumption on this problem: $f(x)$ must be integrable on the interval $[0, 1]$.

As the analytical solution depends on the symbolic integration of $f(x)$, we will be unable to use this solution for functions $f(x)$ in which the closed-form of the integral is not known.

3. Numerical Scheme

As mentioned in the previous section, the analytical solution is not practical to use for most functions $f(x)$. Thus, we present a numerical solution to approximate the analytical solution for the problem $Lu = f$.

3.1. Description

Our solution is derived from the method of finite differences. We define a finite set of points on the interval $[0, 1]$ called the grid D_h where the parameter h is the size of the grid where a smaller h denotes a finer grid. For our purposes, we consider $h = 1/N$ for positive N and create the uniform grid

$$D_h = \{x_n | x_n = hn \text{ for } 0 \leq n \leq N\}. \tag{6}$$

Define on this grid the discretized solution to the problem $Lu = f$ as $[u]_h = \{u(x_n)\}$ and define the discretized function $f^{(h)} = \{f(x_n)\}$. We wish to create a scheme L_h that computes an approximate solution $u^{(h)} = \left\{u_0^{(h)}, u_1^{(h)}, \ldots, u_N^{(h)}\right\}$ to the problem $Lu = f$, i.e. a scheme such that $L_h u^{(h)} = f^{(h)}$.

Finding an approximation to $u''(x)$ should suggest how to construct the scheme L_h. To find an approximation for $u''(x)$, we investigate the Taylor expansion of $u(x)$ centered at h and $-h$. These expansions are given by

$$u(x+h) = u(x) + hu'(x) + \frac{h^2 u''(x)}{2} + \frac{h^3 u^{(3)}(x)}{3!} + \frac{h^4 u^{(4)}(\xi_1)}{4!}$$
$$u(x-h) = u(x) - hu'(x) + \frac{h^2 u''(x)}{2} - \frac{h^3 u^{(3)}(x)}{3!} + \frac{h^4 u^{(4)}(\xi_2)}{4!}$$

where $x \leq \xi_1 \leq x+h$ and $x-h \leq \xi_2 \leq x$. Adding these two expressions and solving for $u''(x)$ shows that

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} - \frac{h^2(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}. \tag{7}$$

This suggests that we should define our numerical scheme by replacing $u''(x)$ in $Lu = f$ with the approximation

$$u''(x) \approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}.$$

Therefore, we define the numerical scheme as

$$L_h u^{(h)} = f^{(h)} := \begin{cases} \frac{-u_{n+1}+2u_n-u_{n-1}}{h^2} + cu_n = f_n \quad \text{for } n = 1, \ldots, N-1 \\ u_0 = \epsilon \\ u_N = \delta \end{cases}. \tag{8}$$

For $n = 1, \ldots, N-1$, the scheme presents us with the recurrence relation

$$-u_{n-1} + (2 + ch^2)u_n - u_{n+1} = h^2 f_n$$

with initial conditions $u_0 = \epsilon$ and $u_N = \delta$. This recurrence relation is represented by the following system of equations

$$\begin{aligned} (2+ch^2)u_1 - u_2 &= h^2 f_1 + u_0 \\ -u_1 + (2+ch^2)u_2 - u_3 &= h^2 f_2 \\ -u_2 + (2+ch^2)u_3 - u_4 &= h^2 f_3 \\ &\vdots \\ -u_{N-2} + (2+ch^2)u_{N-1} &= h^2 f_{N-1} + u_N. \end{aligned}$$

In matrix form, this system of equations becomes

$$\begin{bmatrix} 2+ch^2 & -1 & 0 & \ldots & 0 \\ -1 & 2+ch^2 & -1 & \ldots & 0 \\ 0 & -1 & 2+ch^2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 2+ch^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} h^2 f_1 + u_0 \\ h^2 f_2 \\ h^2 f_3 \\ \vdots \\ h^2 f_{N-1} + u_N \end{bmatrix} \tag{9}$$

The solution to this system of equations paired with the initial conditions allows us to explicitly find $u^{(h)}$, our scheme's solution.

In section 4 we examine the convergence, consistency, and stability of this scheme in order to determine its usefulness in approximating the analytical solution to the problem $Lu = f$.

3.2. Implementation

In order to efficiently use the numerical scheme just described we will need to implement the scheme using computational software.

3.2.1. Discretized Solution

We have implemented the numerical scheme described above in MATLAB which can be used by calling the m-function `numerical_scheme.m`. We will now describe the parameters necessary to call the function, how the function computes the solution, and the results outputted by the function. Please refer to appendix B for the function definition in MATLAB.

This m-function requires the following parameters to compute the numerical solution:

- `f` - MATLAB function that represents the function f in the differential equation $Lu = f$.
- `c` - A real number that represents the constant c in the differential equation $Lu = f$.
- `initials` - An array with two elements representing the initial conditions in the problem $Lu = f$. The first element of the array is ϵ and the second element of the array is δ.
- `interval` - An array with two elements represent the endpoints of the interval of definition in the problem $Lu = f$.
- `subintervals` - An integer that represents the number of subintervals with which to construct the uniform nodes on the interval of definition. This corresponds to an h-value of $1/$`subintervals` on the grid D_h.

Calling the function as follows

```
numerical_scheme(f, c, initials, interval, subintervals)
```

returns the array `[x, u]` where `x` is an array whose elements are the nodes on the grid D_h and `u` is an array whose elements are the numerical solution obtained by the scheme $L_h u^{(h)} = f^{(h)}$ evaluated on the nodes of the grid.

From these parameters, after verifying that c is a positive real number, the function creates and assigns to `x` the uniform nodes equally spaced on the interval with width $1/$`subintervals`. We then construct the coefficient matrix `A` and the right-hand side vector `b` of the equation in (9). Finally, we then assign to `u` the solution vector which is given by `initials(1), inv(A) * b, initials(2)`.

Using this function then allows us to compute the numerical solution to the problem $Lu = f$.

3.2.2. Plotting

4. Numerical Scheme Properties

There are three main properties of the numerical scheme presented in section 3 that are important to the validity of the numerical solution, namely the con-

vergence, consistency, and stability of the numerical scheme. We will now investigate these properties in detail to determine how well the numerical scheme approximates the analytical solution to our differential equation.

4.1. Convergence

The single most important property of the scheme presented in (8) is its convergence to the analytical solution. We will now rigorously define this notion of convergence.

But first, we must define the measure of the deviation between two solutions. Let U_h be the normed linear space of all functions defined on the grid D_h as presented in (6). For $u^h \in U_h$, the equipped norm is given as

$$||u^{(h)}|| = \sup_n |u_n| = \max_n |u_n|. \tag{10}$$

With this definition, we can then precisely define the measure of deviation between two solutions $a^{(h)}$ and $b^{(h)}$ as $||a^{(h)} - n^{(h)}||$.

Thus, we say that the solution $u^{(h)}$ given to us by the scheme $L_h u^{(h)} = f^{(h)}$ `converges` to the discretized analytical solution $[u]_h$ to the problem $Lu = f$ if

$$||[u]_h - u^{(h)}|| \to 0 \quad \text{as } h \to 0. \tag{11}$$

We now present strong numerical evidence that our scheme $L_h u^{(h)} = f^{(h)}$ converges to the discretized analytical solution to the problem $Lu = f$. The program used to create the following tables can be found in appendix C.

The following table shows the values of the computed norm $||[u]_h - u^{(h)}||$ for increasingly smaller values of h for $c = 1$ on the interval $[0, 1]$ with the initials conditions $u(0) = 1$ and $u(1) = 0.5$ with various definitions of the function $f(x)$.

h	x	x^2	x^3	x^4	x^5	$e^{0.5x}$	$\sin(0.1x)$
10^{-1}	`0.2456e-4`						
10^{-2}							
10^{-3}							
10^{-4}							

Table 1: Values of $||[u]_h - u^{(h)}||$ for various functions $f(x)$ with $c = 1$, $u(0) = 1$, and $u(1) = 0.5$.

The above table suggests that the difference between the approximate solution and the exact solution does tend toward zero as we refine the grid value h and that this happens regardless of the choice of $f(x)$.

A natural question would then be if the convergence is dependent upon the choice of c. The following table shows the values of $||[u]_h - u^h||$ for various increasing values of c with $f(x) = e^{0.5x}$ and $u(0) = 1$, $u(1) = 0.5$.

h	1	17	59	119	409	1307	14639
10^{-1}							
10^{-2}							
10^{-3}							
10^{-4}							

Table 2: Values of $||[u]_h - u^{(h)}||$ for various values of c for the function $f(x) = e^{0.5x}$ with $u(0) = 1$ and $u(1) = 0.5$.

The above table suggests that the convergence is not dependent on c for $f(x) = e^{0.5x}$.

The last aspect of convergence to investigate is the impact of the initial conditions on the scheme's convergence convergence. The following table shows the values of $||[u]_h - u^h||$ for various initial conditions with $f(x) = e^{0.5x}$ and $c = 1$.

h	$u(0) = 0$ $u(1) = 0$	$u(0) = 0$ $u(1) = 1$	$u(0) = -3$ $u(1) = 5$	$u(0) = 10$ $u(1) = -4$	$u(0) = -7$ $u(1) = -2$
10^{-1}					
10^{-2}					
10^{-3}					
10^{-4}					

Table 3: Values of $||[u]_h - u^{(h)}||$ for various initial values for the function $f(x) = e^{0.5x}$ with $c = 1$.

Again, from the table we conclude that the approximate solution converges to the exact solution regardless of the choice in initial conditions for the function $f(x) = e^{0.5x}$.

4.2. Consistency

The consistency of a numerical scheme is a measure of how well the solution obtained by the scheme approximates the analytical solution as the grid the scheme is defined on becomes more refined. Formally, for a scheme $L_h u^{(h)} = f^{(h)}$ for the problem $Lu = f$, we say the scheme is *consistent* if

$$||L_h[u]_h - L_h u^{(h)}|| \to 0 \text{ as } h \to 0 \tag{12}$$

where $||u^{(h)}||$ is the norm defined on the normed linear space U_h as presented in (10) in section 4.1.

Using the expression for the second derivative in (7) and replacing it in our problem $Lu = f$, we see that after some rearranging

$$\frac{-u(x+h) + 2u(x) - u(x-h)}{h^2} + cu(x) = f(x) - \frac{h^2(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}.$$

For the discretized analytical solution $[u]_h$ to our problem $Lu = f$ on the grid D_h, this equation then becomes

$$\frac{-u(x_{n+1}) + 2u(x_n) - u(x_{n-1})}{h^2} + cu(x_n) = f(x_n) - \frac{h^2(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}.$$

From this equation we notice that the left side is precisely the evaluation of our numerical scheme for the discretized analytical solution, i.e.

$$L_h[u]_h = f_n - \frac{h^2(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}. \tag{13}$$

Combining the expression in (13) with the fact that $L_h u^{(h)} = f_n$, we see that

$$\begin{aligned} ||L_h[u]_h - L_h u^{(h)}|| &= \left|\left|\left(f_n - \frac{h^2(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}\right) - f_n\right|\right| \\ &= \left|\left|\frac{(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}\right|\right| h^2 \end{aligned}$$

From the above equation it is clear that $||L_h[u]_h - L_h u^{(h)}|| \to 0$ as $h \to 0$. Therefore, according to the definition in (12), we see that our scheme $L_h[u]_h = f^{(h)}$ is consistent.

If we make the assumption that the analytical solution's fourth derivative is bounded, i.e. $|u^{(4)}(x)| \leq M$ for all $0 \leq x \leq 1$, then

$$||L_h[u]_h - L_h u^{(h)}|| = \left|\left|\frac{(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}\right|\right| h^2 \leq \frac{2M}{4!} h^2. \tag{14}$$

Moreover, from the inequality in (14), we see that

$$||L_h[u]_h - L_h u^{(h)}|| \leq \frac{2M}{4!} h^2 = Ch^2 \tag{15}$$

where the constant C does not depend on h. In this case, we then say that the scheme $L_h u^{(h)} = f^{(h)}$ has *order of consistency* 2.

We thus conclude that as our scheme is consistent, it does in fact approximate the analytical solution to the problem $Lu = f$ and approaches the analytical solution as we refine the grid D_h.

4.3. Stability

The last property we wish to investigate for our numerical scheme is the stability of the scheme. The stability of the scheme will ensure that small changes in the

right hand side of the scheme will only cause small changes in the solution derived from the scheme. Formally, the scheme $L_h u^{(h)} = f^{(h)}$ will be called *stable* if for any chosen $h_0 > 0$ and $\delta > 0$ such that for any $h < h_0$ and for any $\varepsilon^{(h)}$ with $\left|\left|\varepsilon^{(h)}\right|\right| < \delta$, the solution for the finite-difference problem

$$L_h z^{(h)} = f^{(h)} + \varepsilon^{(h)}$$

is unique and the deviation from the solution of the unperturbed problem satisfies the estimate

$$\left|\left|z^{(h)} - u^{(h)}\right|\right| \leq k \left|\left|\varepsilon^{(h)}\right|\right| \tag{16}$$

where k does not depend on h or $\varepsilon^{(h)}$.

We now present strong numerical evidence that our scheme satisfies the above inequality. We begin by investigating this condition for our scheme $L_h u^{(h)} = f^{(h)}$ with $f(x) = e^{0.5x}$, $c = 1$, and initial conditions $u(0) = 0$ and $u(1)$. The following table shows the values of $\left|\left|z^{(h)} - u^{(h)}\right|\right|$ for the solutions to $L_h u^{(h)} = f^{(h)}$ and $L_h z^{(h)} = f^{(h)} + \varepsilon^{(h)}$ for increasing values of $\varepsilon^{(h)}$ up to $\varepsilon^{(h)} = 0.5$. The program used to create the following tables can be found in appendix D.

h	0.1	0.2	0.3	0.4
`1.0e-3`	0.011318111	0.022636221	0.033954332	0.045272443
`3.0e-3`	0.011318112	0.022636223	0.033954335	0.045272446
`5.0e-3`	0.011318112	0.022636223	0.033954335	0.045272446
`7.0e-3`	0.011318112	0.022636223	0.033954335	0.045272446
`9.0e-3`	0.011318112	0.022636223	0.033954335	0.045272446
`1.1e-4`	0.011318112	0.022636223	0.033954335	0.045272446
`1.3e-4`	0.011318112	0.022636223	0.033954335	0.045272446
`1.5e-4`	0.011318112	0.022636223	0.033954335	0.045272446

Table 4: Values of $\left|\left|z^{(h)} - u^{(h)}\right|\right|$ for various perturbations $\varepsilon^{(h)}$for the function $f(x) = e^{0.5x}$ with $c = 1$ and initial values $u(0) = 0$ and $u(1) = 0$.

From this table it is clear that for the choice of $h_0 =$ `1.0e-3` and $\delta = 0.4 + 0.00001$, that stability is confirmed for $f(x) = e^{0.5x}$ with $c = 1$ and initial conditions $u(0) = 0$ and $u(1) = 0$. That is, making the conservative choice of $k = 0.115$ this table shows that $\left|\left|z^{(h)} - u^{(h)}\right|\right| \leq k||\varepsilon^{(h)}||$ for $h < h_0$ and $\varepsilon^{(h)} < \delta$.

5. Worked Example

Specify a polynomial for f and set c and show how to implement solution and then plot it.

A. Analytical Solution Program

The following is the m-function `analytical_solution.m` for use in MATLAB to compute the analytical solution to the problem $Lu = f$. Returns a function handle to evaluate the analytical solution at any point on the interval [0, 1]. This function requires the symbolic toolbox in MATLAB.

```
function u = analytical_solution(f, c, initials)
% For the given function f, find the analytical solution to the second
% order differential equation -u''(x) + c * u(x) = f(x) where the passed
% parameter c is positive and subject to the initial conditions u(0) =
% initials(1) and u(1) = initials(2).
%
% Returns a function representing the analytical solution to the above
% differential equation.

if c <= 0
    error('c must be positive.')
end

epsilon = initials(1);
delta = initials(2);
omega = sqrt(c);

syms r;
syms s;

% Define function to help compute particular solution and constants.
kappa = @(x) int(exp(-2.*omega*s)*int(f(r)*exp(omega.*r), 0, s), 0, x);

% Specify the constants unique to choice of initial conditions.
den = exp(-omega) - exp(omega);
c_1 = (epsilon.*exp(-omega) - delta - exp(omega) .* kappa(1)) ./ den;
c_2 = (-epsilon.*exp(omega) + delta + exp(omega) .* kappa(1)) ./ den;

% Define the homogeneous solution.
u_h = @(x) c_1 * exp(omega .* x) + c_2 .* exp(-omega .* x);

% Define the particular solution.
u_p = @(x) -exp(omega .* x) .* kappa(x);

% The solution to the differential equation is the homogeneous solution
% plus the particular solution
u = @(x) u_h(x) + u_p(x);

end
```

B. Numerical Scheme Program

The following is the m-function `numerical_scheme.m` for use in MATLAB to compute the numerical solution to the problem $Lu = f$.

```
function [x, u] = numerical_scheme(f, c, initials, interval, subintervals)
% Implements the numerical scheme to solve the second order differential
% equation -u''(x) + c * u(x) = f(x). f is the handle to the specified
% function f(x) and c is the unknown constant present in the equation which
% must be positive. The array initials are the initial conditions of the
% boundary problem and the first element is the condition at the left
% endpoint of the interval and the second element is the condition at
% right endpoint of the interval passed. subintervals is the number of
% subintervals from which to construct the nodes for the scheme to use.
%
% Returns the nodes x and the solution u(x) at those nodes.

if c <= 0
    error('c must be positive.')
end

% Create uniform nodes on interval with specified number of subintervals.
if subintervals > 1
    x = uniform_nodes(interval, subintervals)';
    v = x(2:subintervals);
else
    error('Number of subintervals must be greater than 1.')
end

u_0 = initials(1);
u_N = initials(2);
h = 1 / subintervals;

% We wish to find v = [u_1, ..., u_subintervals-1] such that Av = b.

% Define b as the evaluation of f at the nodes x_1, ..., x_subintervals-1
% with the exception of the first and last element which is
% f(x(1)) + initials(1) and f(x(subintervals-1)) + initials(2) respectively.
b = h^2 * f(v);
b(1) = b(1) + u_0;
b(length(b)) = b(length(b)) + u_N;

% We need to construct the coefficient matrix A.
main_diag = diag(repelem([2 + c * h^2], length(v)), 0);
sub_diag = diag(repelem([-1], length(v) - 1), -1);
sup_diag = diag(repelem([-1], length(v) - 1), 1);

A = sparse(main_diag + sub_diag + sup_diag);

% Solve system of equations Av = b.
v = A\b;

% Our solution is then u = [u_0, v_1, ..., v_N-1, u_N].
u = [u_0; v; u_N];
```

```
end
```

Note that the above m-function is dependent on the m-function `uniform_nodes.m` and that it must be in MATLAB's path when using `numerical_scheme.m`.

```
function t = uniform_nodes(interval, n)
% Create uniform nodes on the given interval with n subintervals.
%
% Returns array of endpoints for each subinterval.

a = interval(1);
b = interval(2);

t = [];

for i=1:n+1
    x_i = a + (b - a)*((i - 1)/n);

    t = [t, x_i];
end

end
```

C. Numerical Scheme Convergence Program

The following is the m-script `convergence.m` for use in MATLAB to produce a table displaying the convergence of the numerical scheme to the analytical solution

```
format long;

addpath(genpath('../'))

c = 1;
initials = [1, 0.5];
interval = [0, 1];
max_order = 3;

functions = {@(x) x, @(x) x.^2, @(x) x.^3, @(x) x.^4, @(x) x.^5, ...
             @(x) exp(0.5*x), @(x) sin(0.1*x)};

body = [];
for f=1:length(functions)
    disp(functions(f))
    e = convergence_test(functions{f}, c, initials, interval, max_order);
    body = [body, e];
end

disp(functions);
fprintf('------------------------------------------------------------\n');
disp(body);
```

This m-script is dependent upon the following m-function `convergence_test.m`

```
function e = convergence_test(f, c, initials, interval, max_order)
% Compute the normed difference between the numerical solution and
% analytical solution to -u''(x) + c*u(x) = f(x) on the passed interval
% with the initial conditions u(interval(1)) = initials(1) and
% u(interval(2)) = initials(2) on nodes of increasing size from 10^1 to
% 10^max_order.
%
% Returns the vector e showing the normed difference for each
% subinterval size. As the subinterval size increases, i.e. as the h-value
% increases, the normed difference should be decreasing.

e = [];
subintervals = feval(@(x) 10.^x, 1:max_order);

solution = analytical_solution(f, c, initials);

for i=1:length(subintervals)
    h = subintervals(i);

    % Compute numerical scheme for given h-value.
    [x, u] = numerical_scheme(f, c, initials, interval, h);

    % Compute analytical solution at given nodes
    u_h = subs(solution, x);
```

```
    % Add to epsilon vector the norm of the difference.
    e = [e; double(norm(u_h - u, inf))];
end
```

D. Numerical Scheme Stability Program

The following is the m-script `stability.m` for use in MATLAB to produce a table displaying the stability of the numerical scheme to the analytical solution

```
format long;

addpath(genpath('../'))

f = @(x) exp(0.5*x);
c = 1;
initials = [0, 0];
interval = [0, 1];

perturbations = 0.1:0.1:0.5;

v = [];
for i=1:length(perturbations)
    perturbation = perturbations(i);
    disp(perturbation)

    diff = stability_test(f, c, initials, interval, perturbation);

    v = [v, diff];
end

v
```

This m-script is dependent upon the following m-function `stability_test.m`

```
function diff = stability_test(f, c, initials, interval, perturbation)
% Compute the normed difference between the numerical solution for the
% unperturbed solution and the perturbed solution
% (adding perturbation to f) for the numerical scheme
% to the problem to -u''(x) + c*u(x) = f(x) on the passed interval
% with the initial conditions u(interval(1)) = initials(1) and
% u(interval(2)) = initials(2).
%
% Returns the vector diff showing the normed difference between the
% solution for the unperturbed problem and the solution for the perturbed
% problem for each of the given h_values.

% Get values of h from h=1000 to h = 21,000.
h_values = feval(@(x) 1000.*x, 1:2:15);

diff = [];
for i=1:length(h_values)
    h = h_values(i);

    % Compute unperturbed solution
    [x, u] = numerical_scheme(f, c, initials, interval, h);

    % Compute perturbed solution
    [x_z, z] = numerical_scheme(@(x) f(x) + perturbation, c, initials, interval, h);
```

```
    diff = [diff; norm(u - z, inf)];
end
```