A Numerical Solution to a Second Order Ordinary Differential Equation

Hunter, Marcus   Tiger, Matthew   Wigfield, Jacob

November 25, 2015

Contents

1. Introduction

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

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

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 $\lambda_h(x) = c_3 e^{-2\omega x}$ 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

$$u(x) = u_h(x) + u_p(x)$$
$$= c_1 e^{\omega x} + c_2 e^{-\omega x} - e^{\omega x}\int_0^x e^{-2\omega s}\left[\int_0^s f(r)e^{\omega r} dr\right] ds. \tag{5}$$

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

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

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

$$D_h = \{x_n | x_n = hn \text{ for } 0 \le n \le N\}.$$

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

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} - \frac{h^2(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}. \tag{6}$$

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

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

For $n = 1, \dots, N-1$, 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

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

In matrix form, this system of equations becomes

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

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

We will now discuss the implementation of the above numerical scheme and how to use the implementation to numerically approximate the solution to the problem $Lu = f$.

3.2.1. Discretized Solution

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 convergence, 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

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

$$||L_h[u]_h - L_h u^{(h)}|| \to 0 \text{ as } h \to 0 \tag{7}$$

where $||u^{(h)}|| = \max_n |u_n|$.

Using the expression for the second derivative in (6) 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.

$$L_h[u]_h = f_n - \frac{h^2(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}. \tag{8}$$

7

Combining the expression in (8) with the fact that $L_h u^{(h)} = f_n$, we see that

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

$$= \left|\left|\frac{(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}\right|\right| h^2$$

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 (7), 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

$$||L_h[u]_h - L_h u^{(h)}|| = \left|\left|\frac{(u^{(4)}(\xi_1) + u^{(4)}(\xi_2))}{4!}\right|\right| h^2 \leq \frac{2M}{4!} h^2. \tag{9}$$

Moreover, from the inequality in (9), we see that

$$||L_h[u]_h - L_h u^{(h)}|| \leq \frac{2M}{4!} h^2 = Ch^2 \tag{10}$$

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

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.

```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 $Lu = f$.

```matlab
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 / (length(x) - 1);

% 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 = 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 = main_diag + sub_diag + sup_diag;

v = inv(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`.

```matlab
function t = uniform_nodes(interval, n)
% Create uniform nodes on the given interval with n subintervals.
%
```

```matlab
% 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
```