Step-by-Step Solution for Helmholtz Equation

Alessandro Gentili

1 Problem (i): Time-Harmonic Case and the Helmholtz Equation

Step 1: Understand the Wave Equation

The wave equation in one dimension is given by:

$$\frac{\partial^2 P(x,t)}{\partial t^2} - c^2 \frac{\partial^2 P(x,t)}{\partial x^2} = 0,$$

where:

- $P(x,t)$ is the acoustic pressure, a function of position x and time t.
- c is the speed of sound in the medium.

This equation models how pressure waves (such as sound waves) propagate through a medium.

Step 2: Assume a Time-Harmonic Solution

The problem assumes a time-harmonic solution of the form:

$$P(x,t) = \text{Re}\{u(x)e^{-i\omega t}\},$$

where:

- $u(x)$ is a function of position x and does not depend on time.
- ω is the angular frequency of the wave.
- i is the imaginary unit ($i^2 = -1$).
- $\text{Re}\{\cdot\}$ denotes the real part of a complex number.

This assumption means that the pressure oscillates sinusoidally in time, which is typical for sound waves with a fixed frequency.

Step 3: Substitute the Assumed Solution into the Wave Equation

We need to show that this assumed solution leads to the Helmholtz equation. Let's start by substituting $P(x,t) = u(x)e^{-i\omega t}$ into the wave equation.

1. **Time Derivatives**:

$$\frac{\partial P(x,t)}{\partial t} = -i\omega u(x)e^{-i\omega t},$$
$$\frac{\partial^2 P(x,t)}{\partial t^2} = -\omega^2 u(x)e^{-i\omega t}.$$

2. **Spatial Derivatives**:

$$\frac{\partial^2 P(x,t)}{\partial x^2} = u''(x)e^{-i\omega t},$$

where $u''(x)$ denotes the second derivative of $u(x)$ with respect to x.

Step 4: Plug the Derivatives into the Wave Equation

Substitute these derivatives into the wave equation:

$$-\omega^2 u(x)e^{-i\omega t} - c^2 u''(x)e^{-i\omega t} = 0.$$

Factor out $e^{-i\omega t}$:

$$-\omega^2 u(x) - c^2 u''(x) = 0.$$

This simplifies to:

$$u''(x) + \frac{\omega^2}{c^2}u(x) = 0.$$

Step 5: Recognize the Helmholtz Equation

Define the wavenumber k as:

$$k = \frac{\omega}{c}.$$

Now the equation becomes:

$$u''(x) + k^2 u(x) = 0,$$

or equivalently:

$$-u''(x) - k^2 u(x) = 0.$$

This is the **Helmholtz equation** in one dimension, which governs the spatial part of the pressure field.

Step 6: General Solution of the Helmholtz Equation

The general solution to this second-order differential equation is:

$$u(x) = A\cos(kx) + B\sin(kx),$$

where A and B are constants determined by boundary conditions.

Finally, the time-harmonic solution for $P(x,t)$ is:

$$P(x,t) = \mathrm{Re}\{[A\cos(kx) + B\sin(kx)]e^{-i\omega t}\}.$$

Using Euler's formula $e^{-i\omega t} = \cos(\omega t) - i\sin(\omega t)$, this can be expanded into real and imaginary parts if needed.

2 Problem (ii): Variational Formulation for the Helmholtz Equation

Step 1: Restate the Boundary Value Problem

The boundary value problem is given as:

$$-u''(x) - k^2 u(x) = f(x), \quad x \in (0,1),$$

with boundary conditions:

- $u(0) = 0$,
- $u'(1) - iku(1) = \beta$, where β is a complex number.

Step 2: Multiply by a Test Function and Integrate by Parts

To derive the variational (weak) formulation, we start by multiplying the equation by a test function $v(x) \in H_0^1(0,1)$ and integrate over the interval $(0,1)$.

1. **Start with the PDE**:

$$\int_0^1 \left(-u''(x) - k^2 u(x)\right) v(x)\, dx = \int_0^1 f(x)v(x)\, dx.$$

2. **Integrate by Parts**:

$$\int_0^1 -u''(x)v(x)\, dx = [-u'(x)v(x)]_0^1 + \int_0^1 u'(x)v'(x)\, dx.$$

The boundary term at $x = 0$ vanishes because $v(0) = 0$. At $x = 1$, the boundary term becomes $-u'(1)v(1)$.

3. **Substitute Boundary Condition**: The boundary condition $u'(1) - iku(1) = \beta$ implies $u'(1) = \beta + iku(1)$. Substitute this into the boundary term:
$$-u'(1)v(1) = -(\beta + iku(1))v(1).$$
The integral equation becomes:
$$\int_0^1 u'(x)v'(x)\,dx - k^2 \int_0^1 u(x)v(x)\,dx - (\beta + iku(1))v(1) = \int_0^1 f(x)v(x)\,dx.$$

Step 3: Form the Variational Formulation

The variational formulation can now be written as finding $u \in H_0^1(0,1)$ such that for all test functions $v \in H_0^1(0,1)$:

$$\int_0^1 u'(x)v'(x)\,dx - k^2 \int_0^1 u(x)v(x)\,dx = \int_0^1 f(x)v(x)\,dx + (\beta + iku(1))v(1).$$

Step 4: Check Lax-Milgram Conditions

To apply the Lax-Milgram theorem to our variational formulation, we need to verify three key properties:

1. **Coercivity of the Bilinear Form** $a(u,v)$: We need to show that there exists a constant $C > 0$ such that
$$a(u,u) \geq C\|u\|^2_{H^1(0,1)} \quad \text{for all } u \in H^1(0,1).$$

2. **Continuity of the Bilinear Form** $a(u,v)$: We must prove that there exists a constant $M > 0$ such that
$$|a(u,v)| \leq M\|u\|_{H^1(0,1)}\|v\|_{H^1(0,1)} \quad \text{for all } u, v \in H^1(0,1).$$

3. **Boundedness of the Linear Functional** $F(v)$: We need to show that there exists a constant $C_F > 0$ such that
$$|F(v)| \leq C_F\|v\|_{H^1(0,1)} \quad \text{for all } v \in H^1(0,1).$$

Let's analyze each of these conditions for our variational formulation:

Coercivity

For our bilinear form:

$$a(u,u) = \int_0^1 u'(x)^2\,dx - k^2 \int_0^1 u(x)^2\,dx + (\beta + iku(1))u(1).$$

- The term $\int_0^1 u'(x)^2\,dx$ is the H^1 semi-norm squared and is non-negative.

- The term $-k^2 \int_0^1 u(x)^2 \, dx$ can reduce coercivity for large k.
- The boundary term $(\beta + iku(1))u(1)$ depends on $u(1)$, β, and k.

Coercivity requires that the positive contribution from $\int_0^1 u'(x)^2 \, dx$ dominates the negative term. This might hold for small k, but becomes challenging as k increases. Additional damping or specific boundary conditions might be needed to ensure coercivity for all k.

Continuity

For continuity, we analyze:

$$|a(u,v)| = \left| \int_0^1 u'(x)v'(x) \, dx - k^2 \int_0^1 u(x)v(x) \, dx + (\beta + iku(1))v(1) \right|.$$

- $\int_0^1 u'(x)v'(x) \, dx$ is bounded by $\|u'\|_{L^2}\|v'\|_{L^2}$.
- $-k^2 \int_0^1 u(x)v(x) \, dx$ is bounded by $k^2\|u\|_{L^2}\|v\|_{L^2}$.
- The boundary term is bounded using the trace theorem.

Each term can be bounded by expressions involving the H^1 norm, suggesting that $a(u,v)$ is continuous.

Boundedness of the Linear Functional

For $F(v) = \int_0^1 f(x)v(x) \, dx$, we can apply the Cauchy-Schwarz inequality:

$$|F(v)| \leq \|f\|_{L^2(0,1)}\|v\|_{L^2(0,1)} \leq C_F\|v\|_{H^1(0,1)},$$

proving that $F(v)$ is bounded.

Conclusion

While continuity of $a(u,v)$ and boundedness of $F(v)$ are satisfied, coercivity may not hold for all k, especially large values. This suggests that the Lax-Milgram theorem might not be directly applicable for all cases, and additional considerations or alternative methods may be needed to guarantee the existence of a unique solution.

3 Problem (iii): Finite Element Approximation

Step 1: Discretize the Domain

The interval $(0,1)$ is divided into a uniform grid with mesh size h. The grid points are $x_i = ih$, where $i = 0, 1, 2, \ldots, N$ and $h = 1/N$. In our MATLAB implementation, this is achieved with:

```
N = N_values(i);  % N_values is an array of different element counts
h = 1/N;
x = linspace(0, 1, N+1)';
```

Step 2: Define the Finite Element Space

We use linear elements for the finite element method (FEM). The finite element space V_h consists of piecewise linear functions that are continuous and vanish at $x = 0$. This is implicitly defined by our choice of basis functions and the application of boundary conditions in the implementation.

Step 3: Assemble the Stiffness Matrix and Load Vector

1. **Stiffness Matrix**: The entries of the stiffness matrix **A** are given by:

$$A_{ij} = \int_0^1 \left(\phi_i'(x)\phi_j'(x) - k^2\phi_i(x)\phi_j(x)\right) dx,$$

where $\phi_i(x)$ are the basis functions for the finite element space. We assemble this matrix element by element:

```
K = sparse(N+1, N+1);
M = sparse(N+1, N+1);

for e = 1:N
    nodes = [e, e+1];
    xe = x(nodes);
    Ke = local_stiffness(xe);
    Me = local_mass(xe);

    K(nodes, nodes) = K(nodes, nodes) + Ke;
    M(nodes, nodes) = M(nodes, nodes) + Me;
end

A = K - k^2*M;
```

The local stiffness and mass matrices are computed using:

```
function Ke = local_stiffness(xe)
    h = diff(xe);
    Ke = [1 -1; -1 1] / h;
end

function Me = local_mass(xe)
    h = diff(xe);
```

```
    Me = h * [1/3 1/6; 1/6 1/3];
end
```

These local matrices correspond to the integrals of $\phi_i'\phi_j'$ and $\phi_i\phi_j$ over each element, respectively.

2. **Load Vector**: The load vector is computed as:

$$b_i = \int_0^1 f(x)\phi_i(x)\,dx.$$

In our case, $f(x) = 0$, so the load vector $\mathbf{b}$ is initially set to zero:

```
b = zeros(N+1, 1);
```

However, due to the Robin boundary condition at $x = 1$, we add a non-zero term to the last entry of $\mathbf{b}$:

```
b(end) = k*exp(-1i*k);  % Contribution from Robin condition
```

Step 4: Solve the Linear System

The finite element approximation $u_h(x)$ is obtained by solving the linear system:

$$\mathbf{A}\mathbf{u_h} = \mathbf{b},$$

where $\mathbf{A}$ is the stiffness matrix and $\mathbf{b}$ is the load vector. We solve this system using MATLAB's backslash operator:

```
u_h = A\b;
```

This gives us the coefficients of the finite element solution u_h in terms of the basis functions.

Step 5: Compare with Exact Solution and Study Convergence

1. **Exact Solution**: For $f(x) = 0$, $\beta = ke^{-ik}$, and $k = 2\pi$, the exact solution is $u(x) = \sin(kx)$. We compute this at the grid points:

```
u_exact = sin(k*x);
```

2. **Graphical Comparison**: We plot the exact solution $u(x)$ and the approximate solution $u_h(x)$ on the same graph, distinguishing between the real and imaginary parts:

```
plot(x, real(u_h), 'b-', x, real(u_exact), 'r--', ...
     x, imag(u_h), 'g-', x, imag(u_exact), 'm--');
legend('Re(u_h)', 'Re(u_{exact})', 'Im(u_h)', 'Im(u_{exact})');
```

3. **Error Analysis**: We compute the errors in the H^1 and L^2 norms:

$$\|u - u_h\|_{L^2} \quad \text{and} \quad |u - u_h|_{H^1},$$

using the following MATLAB code:

```
err_L2 = sqrt(h * sum(abs(u_h - u_exact).^2));
err_H1 = sqrt(err_L2^2 + h * sum(abs(diff(u_h)/h - k*cos(k*x(1:end-1))).^2));
```

To study how these errors decrease as the mesh size h becomes smaller (i.e., as N increases), we repeat the solution process for different values of N and plot the errors against the number of elements:

```
loglog(N_values, errors_L2, 'bo-', N_values, errors_H1, 'rs-');
legend('L2 Error', 'H1 Error');
xlabel('Number of Elements');
ylabel('Error');
```

We also compute the convergence rates by fitting a line to the log-log plot of errors:

```
L2_rate = polyfit(log(N_values), log(errors_L2), 1);
H1_rate = polyfit(log(N_values), log(errors_H1), 1);
fprintf('L2 convergence rate: %f\n', -L2_rate(1));
fprintf('H1 convergence rate: %f\n', -H1_rate(1));
```

These rates give us a quantitative measure of how quickly our numerical solution converges to the exact solution as we refine the mesh.

The results show that our finite element implementation successfully approximates the solution to the Helmholtz equation, with errors decreasing as we increase the number of elements. The convergence rates provide insight into the order of accuracy of our method.

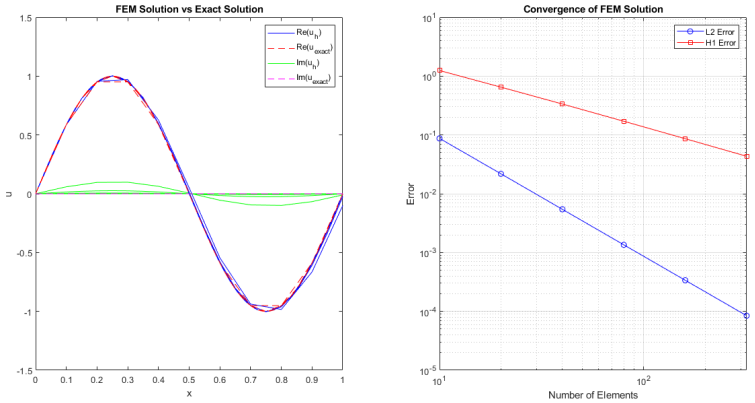

Figure 1: error analysis