Assignment 5 Report

Akshit Vakati Venkata MM23B009

May 2, 2024

1 Transform your chances:

The convolution theorem states that the Fourier transform of the convolution of two functions is equal to the pointwise product of their Fourier transforms.

Let $f(x)$ and $g(x)$ be two continuous functions with Fourier transforms $F(k)$ and $G(k)$ respectively. Then, the convolution of $f(x)$ and $g(x)$, denoted as $f * g$, is defined as:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

The Fourier transform of a function $f(x)$ is defined as:

$$\mathcal{F}(f(x)) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ikx}dx$$

Similarly, the inverse Fourier transform of $F(k)$ is given by:

$$\mathcal{F}^{-1}(F(k)) = \int_{-\infty}^{\infty} F(k)e^{2\pi ikx}dk$$

The Fourier transform of a function $f(x)$ gives its frequency domain representation, where k represents frequency. It is used to analyze the frequency content of a signal or a function.

The Fourier transform of $f * g$ is given by:

$$\mathcal{F}(f * g) = F(k) \cdot G(k)$$

Properties of the Fourier transform include linearity, time shifting, frequency shifting, and convolution. The convolution theorem states that the Fourier transform of the convolution of two functions is equal to the pointwise product of their Fourier transforms.

Proof

Let's consider the convolution of $f(x)$ and $g(x)$, denoted as $h(x)$:

$$h(x) = f * g = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

Now, let's find the Fourier transform of $h(x)$:

$$\begin{aligned}
\mathcal{F}(h(x)) &= \int_{-\infty}^{\infty} h(x)e^{-2\pi ikx}dx \\
&= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} f(t)g(x-t)dt\right) e^{-2\pi ikx}dx \\
&= \int_{-\infty}^{\infty} f(t) \left(\int_{-\infty}^{\infty} g(x-t)e^{-2\pi ikx}dx\right) dt
\end{aligned}$$

Now, let's define a new variable $u = x - t$, so $x = u + t$ and $dx = du$:

$$\begin{aligned}
\mathcal{F}(h(x)) &= \int_{-\infty}^{\infty} f(t) \left(\int_{-\infty}^{\infty} g(u)e^{-2\pi ik(u+t)}du\right) dt \\
&= \int_{-\infty}^{\infty} f(t) \left(\int_{-\infty}^{\infty} g(u)e^{-2\pi iku}du\right) e^{-2\pi ikt}dt \\
&= \int_{-\infty}^{\infty} f(t)e^{-2\pi ikt} \left(\int_{-\infty}^{\infty} g(u)e^{-2\pi iku}du\right) dt \\
&= \left(\int_{-\infty}^{\infty} f(t)e^{-2\pi ikt}dt\right) \left(\int_{-\infty}^{\infty} g(u)e^{-2\pi iku}du\right) \\
&= F(k) \cdot G(k)
\end{aligned}$$

Therefore, the Fourier transform of the convolution of $f(x)$ and $g(x)$ is equal to the pointwise product of their Fourier transforms.

Properties of Convolution

The convolution defines a product on the linear space of integrable functions. This product satisfies the following algebraic properties:

Commutativity

$$f * g = g * f$$

Proof: By definition:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t)g(t-\tau)d\tau$$

Changing the variable of integration to $u = t - \tau$ yields the result.

Associativity

$$f * (g * h) = (f * g) * h$$

Proof: This follows from using Fubini's theorem (i.e., double integrals can be evaluated as iterated integrals in either order).

In mathematical analysis, Fubini's theorem is a result that gives conditions under which it is possible to compute a double integral by using an iterated integral, introduced by Guido Fubini in 1907.
It states that if a function is integrable on a rectangle $X \times Y$, then one can evaluate the double integral as an iterated integral:

$$\iint_{X\times Y} f(x,y)\, d(x,y) = \int_X \left(\int_Y f(x,y)\, dy \right) dx = \int_Y \left(\int_X f(x,y)\, dx \right) dy$$

This means that the order of integration can be interchanged, and the value of the double integral remains the same.

Distributivity

$$f * (g + h) = (f * g) + (f * h)$$

Proof: This follows from linearity of the integral.

Associativity with Scalar Multiplication

$$a(f * g) = (af) * g$$

for any real (or complex) number a.

Procedure

Take File as input:

- Read all the lines from the file taken as argument.
- Strip leading and trailing whitespaces from each line.

Read the Lines:

- Read all the lines from the file.
- Strip leading and trailing whitespaces from each line.

Convert Equations to SymPy Expressions:

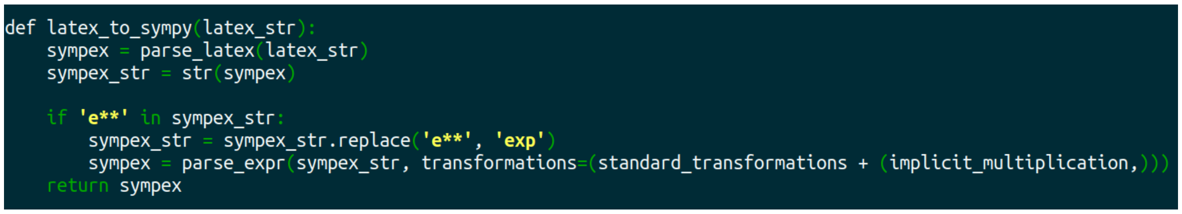
- Convert the first equation string (`equation1_str`) to a SymPy expression (`expr1`):

  `expr1 = latex_to_sympy(equation1_str)`

- Convert the second equation string (`equation2_str`) to a SymPy expression (`expr2`):

  `expr2 = latex_to_sympy(equation2_str)`

Explanation of `latex_to_sympy` Function



```python
def latex_to_sympy(latex_str):
    sympex = parse_latex(latex_str)
    sympex_str = str(sympex)

    if 'e**' in sympex_str:
        sympex_str = sympex_str.replace('e**', 'exp')
        sympex = parse_expr(sympex_str, transformations=(standard_transformations + (implicit_multiplication,)))
    return sympex
```

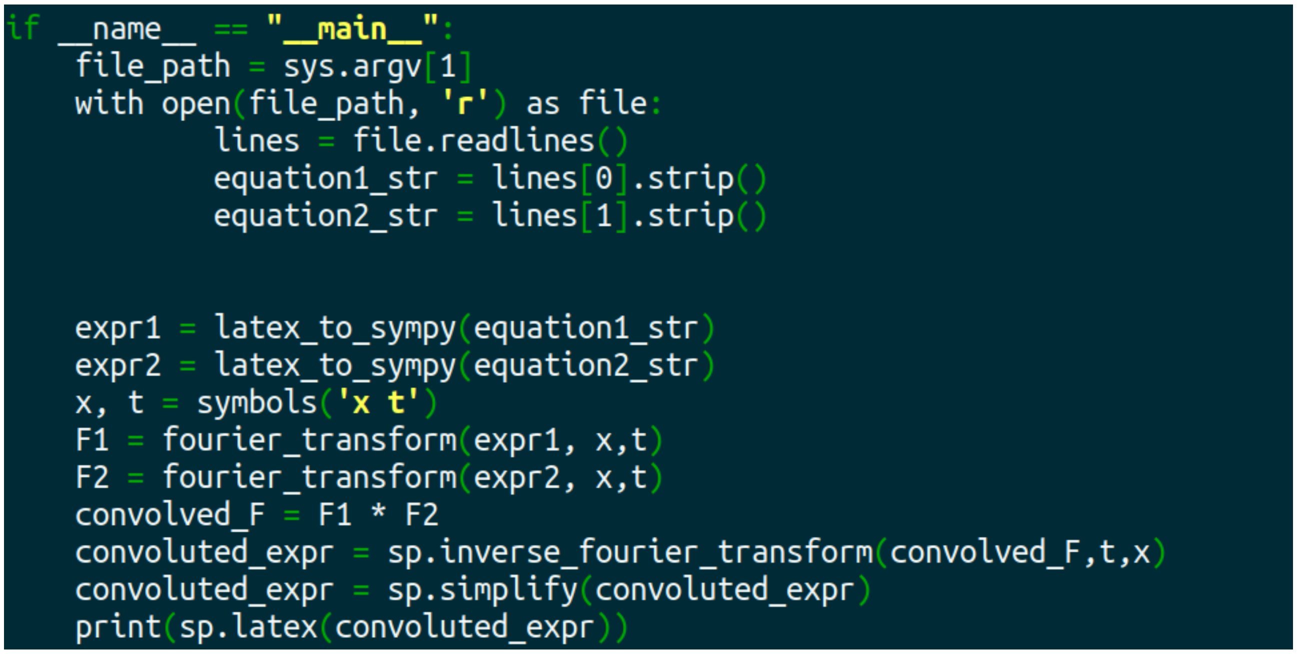
Figure 1: latex to sympy

The `latex_to_sympy` function takes a string representing a mathematical expression in LaTeX format as input and returns the corresponding SymPy expression.

- **Input:** The input `latex_str` is a string representing a mathematical expression in LaTeX format.
- **Output:** The output is a SymPy expression equivalent to the input LaTeX expression.

1. **Parse LaTeX String:**
   - Parse the LaTeX string using `parse_latex` function from SymPy. This function converts the LaTeX string into a SymPy expression object.
   - Store the resulting SymPy expression in the variable `sympex`.
2. **Check for Exponential Terms:**
   - Convert the SymPy expression object to a string representation using `str()`.
   - Check if the string contains the substring `'e**'` which represents exponential terms.
   - If exponential terms are found:
     - Replace `'e**'` with `'exp'` to match SymPy's convention for exponential functions.
     - Parse the modified expression string again using `parse_expr` from SymPy, ensuring that the transformations applied include the replacement of implicit multiplication.

This function is useful for converting mathematical expressions from LaTeX format to SymPy format, allowing for further symbolic manipulation and computation.



```python
if __name__ == "__main__":
    file_path = sys.argv[1]
    with open(file_path, 'r') as file:
            lines = file.readlines()
            equation1_str = lines[0].strip()
            equation2_str = lines[1].strip()


    expr1 = latex_to_sympy(equation1_str)
    expr2 = latex_to_sympy(equation2_str)
    x, t = symbols('x t')
    F1 = fourier_transform(expr1, x,t)
    F2 = fourier_transform(expr2, x,t)
    convolved_F = F1 * F2
    convoluted_expr = sp.inverse_fourier_transform(convolved_F,t,x)
    convoluted_expr = sp.simplify(convoluted_expr)
    print(sp.latex(convoluted_expr))
```

Figure 2: Code block

Define Symbols:

- Define symbols x and t using `symbols()` function from SymPy:

$$x, t = \texttt{symbols('x t')}$$

Compute Fourier Transforms:

- Compute the Fourier transform of `expr1` with respect to x and t, store the result in `F1`:

$$\texttt{F1} = \texttt{fourier\_transform(expr1, x, t)}$$

- Compute the Fourier transform of `expr2` with respect to x and t, store the result in `F2`:

$$\texttt{F2} = \texttt{fourier\_transform(expr2, x, t)}$$

Perform Convolution:

- Compute the convolution of `F1` and `F2`, store the result in `convolved_F`:

$$\texttt{convolved\_F} = \texttt{F1} \times \texttt{F2}$$

Inverse Fourier Transform:

- Compute the inverse Fourier transform of `convolved_F` with respect to t and x, store the result in `convoluted_expr`:

$$\texttt{convoluted\_expr} = \texttt{inverse\_fourier\_transform(convolved\_F, t, x)}$$

Importance and Procedure of Inverse Fourier Transform

In the task at hand, we are performing operations in the frequency domain using Fourier transforms. However, to interpret our results and apply them in practical scenarios, we need to transform the computed frequency-domain representations back to the time or spatial domain. This is where the inverse Fourier transform becomes crucial.

The inverse Fourier transform allows us to recover the original signal or function from its frequency-domain representation, enabling us to analyze, manipulate, and interpret the results effectively.

Procedure

The procedure for performing the inverse Fourier transform in this task can be outlined as follows:

1. **Compute Fourier Transforms**: Compute the Fourier transforms of the given functions using the Fourier transform function provided. For a function $f(x)$, its Fourier transform $F(k)$ is defined as:

$$\mathcal{F}(f(x)) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ikx}dx$$

2. **Define Symbols**: Define the necessary symbols, such as x and t, using the SymPy library.

3. **Perform Convolution**: Compute the convolution of the Fourier transforms obtained in the previous step. The convolution of two functions $F(k)$ and $G(k)$, denoted as $F(k) * G(k)$, is given by:

$$(F * G)(k) = \int_{-\infty}^{\infty} F(k')G(k - k')dk'$$

4. **Apply Inverse Fourier Transform**: Use the inverse Fourier transform function to transform the convolved Fourier transforms back to the time or spatial domain. The inverse Fourier transform of a function $F(k)$ is defined as:

$$\mathcal{F}^{-1}(F(k)) = \int_{-\infty}^{\infty} F(k)e^{2\pi ikx}dk$$

5. **Simplify**: Optionally, simplify the resulting expressions using SymPy's simplification functions.

6. **Output**: Print or display the final expressions, both in their symbolic form and LaTeX representation.

Following this procedure ensures that we can effectively interpret and utilize the results obtained from the Fourier domain operations in our task.

2 Poised for Poiseuille flow

According to Navier stokes equation[1], when applied on a Poiseuille flow in a circular pipe with radius R is subject to the action of an imposed pressure gradient in direction z. Given equations are :

Equation (1) also called as equation of continuity:

$$\frac{\partial \rho}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(\rho r v_r) + \frac{1}{r}\frac{\partial(\rho v_z)}{\partial z} + \frac{1}{r}\frac{\partial(\rho v_\theta)}{\partial \theta} = 0 \tag{1}$$

Equation (2) Navier Stokes Equation:

$$\frac{\partial(\rho \vec{v})}{\partial t} + \nabla \cdot (\rho \vec{v}\vec{v}) = \rho \vec{g} - \nabla P + \mu \nabla^2 \vec{v} \tag{2}$$

Governing equations in many cases are intricate and interdependent, often comprising coupled partial differential terms. Solving these equations directly is often impractical, necessitating logical simplifications. Common simplifications involve:

1. **θ-symmetric flow:**
   θ components and their changes can be neglected, which means $v_\theta = 0$ and $\frac{\partial}{\partial \theta}\vec{v} = 0$.

2. **Fully Developed Flow (z-velocity not dependent on z):**
   $\frac{\partial}{\partial z} v_z = 0$.

3. **Incompressible flow (ρ is constant)**

4. **Impenetrable wall:**
   Zero radial velocity at a radius equal to pipe radius. $\implies$

   $$v_r(R) = 0$$

   where R is the radius of the pipe.

5. **Continuous Flow Profile**:

   - **Mathematical Meaning**: A flow profile is continuous if there are no breaks or jumps in the velocity distribution along any direction.
   - **Mathematical Representation**:
     - For radial direction $(v_r(r))$: $v_r(r)$ is a continuous function of r.
     - For axial direction $(v_z(z))$: $v_z(z)$ is a continuous function of z.

[1] Deville, "Exact Solutions of the Navier–Stokes Equations".

6. **Smooth (or Differentiable) Flow Profile**:

- **Mathematical Meaning**: A flow profile is smooth if the velocity changes smoothly and continuously as we move along the direction of flow.
- **Mathematical Representation**:
  - For radial direction ($v_r(r)$): $v_r(r)$ is differentiable with respect to r, i.e., $v_r'(r)$ exists and is continuous.
  - For axial direction ($v_z(z)$): $v_z(z)$ is differentiable with respect to z, i.e., $v_z'(z)$ exists and is continuous.

The continuity equation (Eq. 1) is then

$$\frac{1}{r}\frac{\partial}{\partial r}(rv_r) = 0.$$

integration yields :

$$rv_r = f(z)$$

But, since $v_r = 0$ at the wall, $r = R$, we conclude that $f(z) = 0$ and thus that v_r is zero everywhere in the flow. The Navier–Stokes equation for the radial component of velocity reduces to $\frac{\partial p}{\partial r} = 0$. The pressure depends only on z and not on r. The equation for the velocity component v_z (A.23) yields

$$-\frac{dp}{dz} + \mu\left(\frac{\partial^2 v_z}{\partial r^2} + \frac{1}{r}\frac{\partial v_z}{\partial r}\right) = 0$$

or

$$\frac{dp}{dz} = \mu\frac{1}{r}\frac{d}{dr}\left(r\frac{dv_z}{dr}\right).$$

The left-hand side term only depends on z; on the right-hand side there is only dependence on r. Thus the two terms must be equal to a constant. Integrating, we obtain:

$$\frac{dp}{dz} = \mu\frac{1}{r}\frac{d}{dr}\left(r\frac{dv_z}{dr}\right) = C_1$$

where C_1 is a constant. Now, let's solve this differential equation:

$$r\frac{dv_z}{dr} = \frac{1}{\mu}\left(C_1\frac{r^2}{2} + C_2\right)$$

where C_2 is another constant. We can solve this equation for v_z:

$$\frac{dv_z}{dr} = \frac{1}{\mu}\left(C_1\frac{r}{2} + \frac{C_2}{r}\right)$$

Integrating again:

$$v_z = \frac{1}{\mu}\left(C_1\frac{r^2}{4} + C_2 \ln r + C_3\right)$$

where C_3 is yet another constant. Now, let's express this in terms of the given pressure gradient, and we know $\frac{dp}{dz}$ is a constant which indicates that P is only a **linear** function of z:

$$v_z = \frac{dp}{dz}\cdot\frac{1}{\mu}\left(\frac{r^2}{4} + A\ln r + B\right)$$

where A and B are constants.

The velocity must be finite on the axis $r = 0$. This leads to $A \equiv 0$. Taking into account the condition $v_z(R) = 0$, we have

$$v_z = -\frac{dp}{dz}\frac{R^2}{4\mu}\left(1 - \frac{r^2}{R^2}\right) = \frac{GR^2}{4\mu}\left(1 - \frac{r^2}{R^2}\right),$$

with the definition $\frac{dp}{dz} = -G$. In Poiseuille flow, the velocity profile is parabolic. The maximum velocity at the center is

$$v_{\max} = \frac{dp}{dz}\frac{R^2}{4\mu}.$$

in the given assignment μ, R are taken as unity

so

$$v_{\max} = \frac{1}{4}\frac{dp}{dz}.$$

Therefore, the parabolic profile may be written as

$$v_z = v_{\max}\left(1 - r^2\right).$$

or

$$v_z = \frac{1}{4}\frac{dp}{dz}\left(1 - r^2\right).$$

This is the formula used in *vel.cpp* file and which take r as an input and returns the v_z as output.

3 References:

1. convolution-L.pdf
2. Wikipedia contributors, *Convolution theorem — Wikipedia, The Free Encyclopedia*
3. Wikipedia contributors, *Navier–Stokes equations — Wikipedia, The Free Encyclopedia*
4. Deville, “Exact Solutions of the Navier–Stokes Equations”
5. Blackledge, “Chapter 2 - 2D Fourier Theory”
6. Gravemeier, Wall, and Ramm, “- Numerical solution of the incompressible Navier-Stokes equations by a three-level finite element method”