# Documentación del Proyecto

Daniel Fernando Correa Carreño
Issac David Durango Martinez
Jonatan David Rodas Piedrahita
Santiago Cano Vasquez

March 7, 2025

## Contents

**1 Introducción** **3**

**2 Resumen de Módulos** **3**
2.1 gui.latex_util . . . 3
2.1.1 Propósito . . . 3
2.1.2 Clase Principal: `LatexLabel` . . . 3
2.2 gui.home_page . . . 3
2.2.1 Propósito . . . 3
2.2.2 Clase: `HomePage` . . . 4
2.3 gui.form_page . . . 4
2.3.1 Propósito . . . 4
2.3.2 Clase: `FormPage` . . . 4
2.4 gui.results_page . . . 4
2.4.1 Propósito . . . 4
2.4.2 Clase: `ResultsPage` . . . 4
2.5 gui.metodos_acoplados_page . . . 5
2.5.1 Propósito . . . 5
2.5.2 Clase: `MetodosAcopladosPage` . . . 5
2.6 gui.F_results_page . . . 5
2.6.1 Propósito . . . 5
2.6.2 Clase: `FResultsPage` . . . 5
2.7 methods.FGraphicResults . . . 6
2.7.1 Propósito . . . 6
2.7.2 Clase: `FGraphicResults` . . . 6
2.8 methods.field_functions . . . 6
2.8.1 Propósito . . . 6
2.8.2 Funciones Principales . . . 6
2.9 methods.metodo_ondulatorio . . . 6
2.9.1 Propósito . . . 6

2.9.2 Funciones Clave . . . . . 7
2.10 methods.GraphicResults.py . . . . . 7
2.10.1 Propósito . . . . . 7
2.10.2 Clase: `GraphicResults` . . . . . 7
2.11 Pruebas . . . . . 7

**3 Conclusión** **7**

# 1 Introducción

Este proyecto demuestra una aplicación GUI de PySide6 con múltiples vistas para analizar y mostrar modos de una guía de onda. Consta de varios módulos en Python, cada uno enfocado en un aspecto específico de la aplicación:

- **Módulos GUI**: Proporcionan las distintas páginas (Inicio, Formulario, Resultados, etc.) en una interfaz basada en QStackedWidget.
- **Módulos de Métodos**: Implementan el análisis de modos de la guía de onda, incluyendo los enfoques "ondulatorio" y "rayos".
- **Utilidades**: Contienen clases y funciones auxiliares (p. ej., para renderizado LaTeX o para calcular perfiles de campo).
- **Pruebas**: Verifican la corrección contra datos de referencia.

# 2 Resumen de Módulos

## 2.1 gui.latex_util

### 2.1.1 Propósito

Define la clase `LatexLabel`, que renderiza expresiones LaTeX en un QPixmap para su despliegue en una GUI PySide6.

### 2.1.2 Clase Principal: `LatexLabel`

- **Uso:**

```
from latex_util import LatexLabel
label = LatexLabel(r"$\alpha = \beta + \gamma$")
layout.addWidget(label)
```

- **Inicialización**:

  **latex_text (str)** Cadena LaTeX a renderizar.

  **fontsize (int)** Tamaño de fuente (por defecto 10).

- **Método Estático:** `latex_to_pixmap(latex_text, fontsize)`
  Convierte la cadena LaTeX en un QPixmap usando Matplotlib.

## 2.2 gui.home_page

### 2.2.1 Propósito

Implementa `HomePage`, la página inicial de la aplicación, mostrando un título, descripción y un botón para ir a la página de formulario.

#### 2.2.2 Clase: `HomePage`

- **Constructor**:

  **parent (QWidget)** Widget padre.

  **stack (QStackedWidget)** El QStackedWidget para navegar.

- **Métodos Clave**:

  `setup_ui()` Crea la interfaz con título, descripción y botón.

  `go_to_form()` Navega a la `FormPage`.

### 2.3 gui.form_page

#### 2.3.1 Propósito

Implementa `FormPage`, donde el usuario ingresa parámetros numéricos de la guía (ej. $n_{co}$, $n_{cl}$, altura, etc.). Al pulsar "Submit", navega a `ResultsPage`.

#### 2.3.2 Clase: `FormPage`

- **Constructor**:

  **parent (QWidget)** Widget padre.

  **stack (QStackedWidget)** El QStackedWidget para navegar.

- **Métodos Clave**:

  `setup_ui()` Construye el formulario con `QFormLayout` y botones "Back" y "Submit".

  `go_to_results()` Valida entradas y crea `ResultsPage`.

  `go_to_homepage()` Elimina la página del stack, regresando a `HomePage`.

### 2.4 gui.results_page

#### 2.4.1 Propósito

Muestra resultados para dos métodos: "rayos" y "ondulatorio". Cada método tiene su propia tabla con datos TE y TM. Incluye ecuaciones en LaTeX y un `QScrollArea` para desplazar el contenido.

#### 2.4.2 Clase: `ResultsPage`

- **Constructor**:

  **parent, stack** Widgets de PySide6 y la pila para navegar.

  **n_co, n_cl, h, lambd** Parámetros de la guía.

- **Métodos Clave**:

`setup_ui()` Dispone las tablas, el área de scroll y los botones.

`fillTableRayos()` / `fillTableOndulatorio()` Usa `metodo_rayo` o `metodo_ondulatorio` para llenar la tabla.

`handle_ondulatorio_cell_clicked()` Genera gráficas emergentes usando `GraphicResults`.

## 2.5 gui.metodos_acoplados_page

### 2.5.1 Propósito

Muestra una descripción del fenómeno de guías acopladas y un conjunto de ecuaciones LaTeX. También ofrece navegación a la página `FResultsPage` para manejar gráficas basadas en $F$.

### 2.5.2 Clase: `MetodosAcopladosPage`

- **Constructor**:

  **parent, stack** Widgets de PySide6 y la pila para navegar.

  **n_eff_TE, n_eff_TM, lambd** Parámetros adicionales para el acoplamiento.

- **Métodos Clave**:

  `setup_ui()` Muestra múltiples ecuaciones en dos columnas, y botones "Back" y "Graphics".

  `show_graphics()` Navega a `FResultsPage`.

## 2.6 gui.F_results_page

### 2.6.1 Propósito

Despliega una tabla de resultados basados en $F$. Cada fila corresponde a una comparación de modos (por ejemplo, modo 0 y 1), y cada columna a un valor de $F$. Al hacer clic en una celda, se muestra la gráfica de $P_a$ y $P_b$ usando `FGraphicResults`.

### 2.6.2 Clase: `FResultsPage`

- **Constructor**:

  **parent, stack** Widgets PySide6 y pila para navegación.

  **n_eff_TE, n_eff_TM, lambd** Índices y longitud de onda para trazar $F$.

- **Métodos Clave**:

  `handle_F_cell_clicked()` Determina qué valor de $F$ y qué comparación de modos se selecciona, luego crea `FGraphicResults` para mostrar la gráfica.

## 2.7 methods.FGraphicResults

### 2.7.1 Propósito

Proporciona la lógica para calcular las curvas $P_a$ y $P_b$ basadas en un parámetro $F$. Traza estas curvas con un etiquetado personalizado del eje x en términos de múltiplos de $\pi$.

### 2.7.2 Clase: `FGraphicResults`

- **Constructor**:

  **n_eff1, n_eff2** Dos índices efectivos.

  **lambd** Longitud de onda.

- **Métodos**:

  `get_beta()`, `get_delta()`, `get_kappa()`, `get_psi()` Calculan parámetros intermedios para el acoplamiento.

  `plot_F_graphs()` Traza $P_a(z)$ y $P_b(z)$ en una sola figura, etiquetando el eje x en múltiplos de $\pi/2$ o $\pi$.

## 2.8 methods.field_functions

### 2.8.1 Propósito

Contiene funciones para generar perfiles de campo TE/TM (even/odd). Cada función devuelve un lambda que calcula el valor del campo en función de $x$.

### 2.8.2 Funciones Principales

- `get_E_y_even()`, `get_E_y_odd()`, `get_H_z_even()`, `get_H_z_odd()` (modos TE).
- `get_H_y_even()`, `get_H_y_odd()`, `get_E_z_even()`, `get_E_z_odd()` (modos TM).

## 2.9 methods.metodo_ondulatorio

### 2.9.1 Propósito

Implementa el enfoque ondulatorio para resolver las ecuaciones de modo en una guía plana. Utiliza funciones trascendentes en términos de $U$ y el método de bisección para encontrar soluciones válidas.

#### 2.9.2 Funciones Clave

- `get_W()`: Devuelve una función $W(U)$ que depende de $m$ y si es TE o TM.
- `funcion_ondulatoria()`: Construye una función cuyas raíces son valores de $U$ válidos.
- `metodo_ondulatorio()`: Itera sobre índices de modo, encuentra $U$ por bisección y los convierte a ángulos $\theta$ en grados (TE/TM).

### 2.10 methods.GraphicResults.py

#### 2.10.1 Propósito

Similar a `FGraphicResults`, pero para modos TE/TM en una guía plana. Genera gráficas de campo E y H usando `field_functions.py`.

#### 2.10.2 Clase: `GraphicResults`

- **Constructor**: Recibe $n_{co}$, $n_{cl}$, $h$, $\lambda$ y una lista opcional de modos. Precalcula las soluciones para cada modo.
- **plot_fields()**: Traza las distribuciones de campo eléctrico y magnético en dos subgráficas.

### 2.11 Pruebas

Existen varias pruebas para validar la corrección del análisis:

- `tests.metodo_ondulatorio_test`:
  - **test_TE_function()**: Compara la función TE con referencias.
  - **test_TM_function()**: Compara la función TM con referencias.
  - **test_result()**: Verifica ángulos calculados con valores de referencia para los primeros modos.

## 3 Conclusión

Esta aplicación integra una GUI de PySide6 con métodos de análisis de guías de onda ("rayos" y "ondulatorio") para calcular y mostrar diversos parámetros de modos. La documentación anterior describe la responsabilidad y uso de cada módulo, con ejemplos de implementación y referencias a las pruebas. Para más detalles, consultar los docstrings dentro del código y los archivos de prueba.