Design Document

1. Program Structure

Directory Layout

- **b_form/**: Contains implementation files for B-Spline operations.
- **curve_fitting/**: Handles curve fitting functionality for plane and spherical curves.
- **json/**: Parses configuration files and manages JSON input.
- **pp_form/**: Implements piecewise polynomial splines (pp-Form).
- **utils/**: Provides utilities, such as tridiagonal matrix solvers and derivative calculations.

Each directory corresponds to a key aspect of the system, encapsulating related functionalities.

2. Design Ideas

The program adheres to the following design principles:

- **Modularity**: Classes and files are designed to perform specific tasks. For example, BSpline focuses on B-spline operations, and CurveFitting manages curve fitting.
- **Reusability**: Common utility functions, such as solving tridiagonal systems and computing chordal lengths, are centralized in the utils module for reuse.
- **Flexibility**: Constructors and methods are overloaded to handle multiple scenarios, such as natural, clamped, or periodic boundary conditions.
- **Extensibility**: The design allows easy extension to support higher-order splines or additional boundary conditions.

3. Key Classes and Functionalities

3.1 B-Spline (b_form/b_spline.h, b_spline.cpp)

Responsibilities:

- Implements B-spline interpolation.

- Supports different boundary conditions (natural, clamped, periodic).
- Provides methods for knot vector computation, curve fitting, and spline evaluation.

Key Methods:

- evaluate(double t): Evaluates the B-spline at a given parameter value.
- fit(const std::vector<double>& x, const std::vector<double>& y): Fits the spline to given data points.
- computeKnots(): Computes uniform or customized knot vectors.
- evaluateBasis(int i, int k, double t): Evaluates the basis function recursively.

3.2 Piecewise Polynomial (pp_form/piecewise_polynomial.h, piecewise_polynomial.cpp)

Responsibilities:

- Represents piecewise polynomials for spline interpolation.
- Allows evaluation and printing of polynomials.

Key Methods:

- evaluate(double x): Evaluates the polynomial at a given x.
- print(std::ostream& os): Outputs polynomial details for debugging.

3.3 PPSpline (pp_form/pp_spline.h, pp_spline.cpp)

Responsibilities:

- Manages piecewise polynomial splines in pp-form.
- Handles linear and cubic splines with boundary conditions.

Key Methods:

- computeLinearSpline(const std::vector<double>& values, int dim): Constructs linear splines.
- computeCubicSpline(const std::vector<double>& values, int dim, SplineBoundaryCondition bc): Constructs cubic splines with boundary conditions.
- evaluate(double x): Evaluates the spline at a given x.

3.4 Curve Fitting (curve_fitting/curve_fitting.h, curve_fitting.cpp)

Responsibilities:

- Fits splines to planar or spherical data points.
- Supports boundary conditions.

Key Methods:

- fitCurve(const std::vector<double>& x, const std::vector<double>& y, SplineBoundaryCondition bc): Fits a curve in 2D.
- fitSphericalCurve(const std::vector<std::pair<double, double>>& sphericalPoints, SplineBoundaryCondition bc): Fits a curve on a sphere.

3.5 Cubic Spline (pp_form/cubic_spline.h, cubic_spline.cpp)

Responsibilities:

- Implements cubic spline interpolation.
- Supports natural, clamped, and periodic boundary conditions.

Key Methods:

- fitNatural(): Fits a natural cubic spline.
- fitClamped(): Fits a clamped cubic spline.
- fitPeriodic(): Fits a periodic cubic spline.

3.6 Linear Spline (pp_form/linear_spline.h, linear_spline.cpp)

Responsibilities:

- Implements linear spline interpolation.
- Provides basic linear interpolation methods.

Key Methods:

- fit(const std::vector<double>& x_points, const std::vector<double>& y_points): Fits a linear spline to given points.
- evaluate(double t): Evaluates the linear spline at a given point.

3.7 Polynomial (pp_form/polynomial.h, polynomial.cpp)

Responsibilities:

- Represents individual polynomials.

- Supports operations like addition, subtraction, and multiplication.

Key Methods:

- evaluate(double x): Evaluates the polynomial at x.
- derivative(): Computes the polynomial's derivative.
- operator+, operator-, operator*: Enables polynomial arithmetic.

3.8 JSON Configuration Handler (json/json_config_handler.h, json_config_handler.cpp)

Responsibilities:

- Parses JSON configuration files for spline parameters.

Key Methods:

- parseConfig(): Parses raw JSON content into structured data.
- getConfig(): Retrieves parsed data.

3.9 Utility Functions (utils/spline_utils.h, spline_utils.cpp)

Responsibilities:

- Provides reusable mathematical utilities, including:
    - Solving tridiagonal matrix systems.
    - Computing chordal lengths.
    - Generating uniform nodes.

Key Methods:

- solveTridiagonalMatrix(): Solves tridiagonal linear systems using the Thomas algorithm.
- computeChordalLength(): Computes cumulative chordal lengths.
- generateUniformNodes(): Generates evenly spaced nodes in a given interval.

3.10 MathFunction (utils/math_function.h, math_function.cpp)

Responsibilities:

- Encapsulates mathematical functions.

Key Methods:

- evaluate(double x): Evaluates the function at x.

4. Class Relationships

- **PPSpline** and **BSpline** depend on utility methods from spline_utils.
- **CurveFitting** uses PPSpline and BSpline for spline fitting.
- **PiecewisePolynomial** aggregates multiple Polynomial objects.
- **JSONConfigHandler** integrates with other classes by providing configuration data.