This documentation is mostly generated by ChatGPT.

Symars Documentation

1. Enum: `DType`

1.1. Description

The floating point precision you are using in computation.

1.2. Variants

- `DType.F32`
- `DType.F64`

2. Class: `SymarsUni`

2.1. Description

The `SymarsUni` class generates Rust code for a **scalar** in SymPy.

2.2. Constructor

```
SymarsUni(dtype: DType, tol: float = 1e-9, debug: bool = False)
```

- **Parameters:**
  - `dtype` (`DType`): Specifies the data type (`F32` or `F64`) for Rust code generation.
  - `tol` (`float`): Tolerance for floating-point equality checks (default: `1e-9`).
  - `debug` (`bool`): Enables debug mode for verbose output (default: `False`).

2.3. Methods

1. `generate_func(self, name: str, expr)`
   - Purpose: Generates Rust function code for a given SymPy expression.
   - Parameters:
     - `name` (`str`): Name of the function to generate.
     - `expr`: A SymPy expression to translate into Rust code.
   - Returns: `str` containing the generated Rust function code.

2. `generate_func_given_params(self, name: str, expr, params)`
   - Purpose: Generates Rust function code for a SymPy expression with a specified parameter list.
   - Parameters:
     - `name` (`str`): Name of the function.
     - `expr`: A SymPy expression to translate.
     - `params` (`list[str]`): List of parameter names for the function.
   - Returns: `str` containing the generated Rust function code.
   - Notes: The user is responsible for ensuring the correctness of the parameter list.

3. Class: `SymarsDense`

3.1. Description

The `SymarsDense` generates Rust code for dense matrices in SymPy, serving as backend to interface multiple rust crates, such as `nalgebra` documented below.

3.2. Constructor

```
SymarsDense(dtype: DType, tol: float = 1e-9, debug: bool = False)
```

- Parameters:
  - `dtype` (`DType`): Specifies the data type (`F32` or `F64`) for Rust code generation.
  - `tol` (`float`): Tolerance for floating-point equality checks (default: `1e-9`).
  - `debug` (`bool`): Enables debug mode for verbose output (default: `False`).

3.3. Methods

1. `generate(self, mat: sp.Matrix, func_name: str) -> dict`
   - Purpose: Generates Rust function implementations for each element of the matrix.
   - Parameters:
     - `mat` (`sp.Matrix`): A SymPy matrix whose elements will be converted to Rust code.
     - `func_name` (`str`): The base name for the functions generated for each matrix element.

- Returns: `dict` mapping `(row, col)` indices to their respective Rust function implementation strings.

4. Class: `SymarsNalgebra`

4.1. Description

The `SymarsNalgebra` class interfaces `SymarsDense` in format compatible with `nalgebra`, the Rust linear algebra crate.

4.2. Constructor

```
SymarsNalgebra(dtype: DType, tol: float = 1e-9, debug: bool = False)
```

- Parameters:
  - `dtype` (`DType`): Specifies the data type (`F32` or `F64`) for Rust code generation.
  - `tol` (`float`): Tolerance for floating-point equality checks (default: `1e-9`).
  - `debug` (`bool`): Enables debug mode for verbose output (default: `False`).

4.3. Methods

1. `generate(self, mat: sp.Matrix, func_name: str) -> str`
   - Purpose: Generates Rust code for a `nalgebra::SMatrix`.
   - Parameters:
     - `mat` (`sp.Matrix`): A SymPy matrix whose elements will be converted to Rust code.
     - `func_name` (`str`): The base name for the Rust matrix function.
   - Returns: `str` containing the complete Rust code for the matrix operation, including individual element functions and the matrix assembly function.

5. Notes

1. Remember to pass **legal identifier** (in Rust) to the `name` parameter.