JSLAB v1.0.0
DOCUMENTATION

github.com/PR-DC/JSLAB

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Contents

JSLAB v1.0.0

DOCUMENTATION

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

1   About JSLAB

Welcome to the JSLAB Documentation! This guide provides comprehensive information for users and contributors, detailing the features, usage, and contribution guidelines to ensure a consistent and high-quality codebase.

Figure 1 - JSLAB logo

The **JavaScript Laboratory (JSLAB)** is an open-source environment designed for scientific computing, data visualization, and various other computer operations. Inspired by *GNU Octave* and *Matlab*, JSLAB leverages the advantages of JavaScript, including its blazing speed, extensive examples, backing by some of the largest software companies globally, and the vast community of active programmers and software engineers.

The program was developed to fulfill the need for performing calculations in a programming language that allows for code reuse in later project stages. JavaScript was chosen for its speed, dynamic nature, interpretability, extensive library support, large existing codebase, backing by major software companies, and the ability to create both desktop and mobile applications.

1.1   Why Choose JSLAB?

- **Backed by Leading Investments**

  JavaScript is supported by major industry investments, ensuring continuous innovation and robust development. Our commitment to excellence makes JSLAB a trusted choice for professionals and organizations worldwide.

- **Powered by JavaScript, Trusted by Giants**

  Join the ranks of top companies who leverage JavaScript for their mission-critical applications. With JSLAB, you benefit from the same reliable and scalable technology that powers some of the most advanced projects on Earth and beyond.

- **Thriving Community and Massive User Base**

  Become part of a vibrant and growing community of JavaScript developer. Extensive support network and active forums ensure you always have the resources and assistance you need to succeed.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- **Comprehensive Functionality Comparable to Leading Tools**

  JSLAB bridges the gap between JavaScript and specialized scientific tools. Enjoy functionalities equivalent to MATLAB, GNU Octave, Python, R, and Julia, all within a single, unified platform. Perform data analysis, machine learning, numerical computations, and more with ease.

- **Seamless and Native GUI with HTML, CSS, and SVG**

  Design intuitive and visually appealing graphical user interfaces using native HTML, CSS, and SVG. Create interactive dashboards, custom visualizations, and responsive layouts without the need for additional frameworks.

- **Extend with Native Modules via NPM and C++/C**

  Enhance JSLAB's capabilities by integrating native modules from npm, built with C++ and C. Tap into a vast ecosystem of extensions and customize your environment to meet your specific needs, ensuring maximum performance and flexibility.

- **Join the JSLAB Revolution Today!**

  Experience the seamless integration of powerful scientific computing and the flexibility of JavaScript. Whether you're developing complex algorithms, analyzing vast datasets, or creating innovative applications, JSLAB empowers you to achieve more.

1.2 License

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see `https://www.gnu.org/licenses/`.

Copyright (C) 2024 PR-DC info@pr-dc.com

2 About documentation

This documentation serves as a comprehensive guide for both users and contributors of the JSLAB Library. It covers an introduction to the project, detailed features, installation and setup instructions, user interface overview, practical examples, coding standards, build instructions, contribution guidelines, and mechanisms for providing feedback.

The documentation is structured to provide clear and detailed information, ensuring that both new users and seasoned contributors can effectively utilize and contribute to the JSLAB project.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

3   Coding style

3.1   Documentation and Comments

Clear documentation is essential for maintaining and understanding the codebase. We utilize JSDoc for structured documentation and inline comments to clarify complex logic.

- **JSDoc:** Use JSDoc comments to document files, classes, methods, parameters, and return values.

- **Inline Comments:** Add comments to explain non-trivial code segments.

```
1  /**
2   * Constructs the JSLAB library environment.
3   * @param {Object} config Configuration options.
4   */
5  constructor(config) {
6    // Initialize properties
7    this.config = config;
8    // ... other initializations
9  }
```

3.2   Naming Conventions

- **Classes and Constants:** Use uppercase letters with underscores (e.g., `PRDC_JSLAB_LIB`).

- **Functions and Methods:** Use camelCase (e.g., `getFullFilePath`).

- **Variables:** Use lowercase letters with underscores (e.g., `new_file_path`).

- **Meaningful Names:** Choose self-explanatory names that convey the purpose clearly.

3.3   Code Structure and Modularization

- **Separation of Concerns:** Organize code into separate modules/files based on functionality.

- **Object-Oriented Design:** Use ES6 classes to encapsulate related properties and methods.

```
1  const { PRDC_JSLAB_EVAL } = require('./jslab-eval');
2
3  class PRDC_JSLAB_LIB {
4    constructor(config) {
5    this.eval = new PRDC_JSLAB_EVAL(this);
6    // ... other initializations
7    }
8  }
```

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

3.4   Error Handling

- **Try-Catch Blocks:** Use try-catch to handle potential errors gracefully.

- **Custom Errors:** Throw custom error objects for specific error scenarios.

```
1  try {
2    return this.resolve(file_path);
3  } catch {
4    this.jsl.env.error('Path resolution failed.');
5  }
```

3.5   State Management and Cleanup

- **Resource Tracking:** Maintain arrays and objects to track active asynchronous operations.

- **Cleanup Methods:** Implement methods to clear resources and reset the environment state.

3.6   Configuration Management

- **Centralized Configuration:** Use a dedicated class (`PRDC_APP_CONFIG`) to manage all configuration settings.

- **Conditional Configurations:** Adjust settings based on the runtime environment or command-line arguments.

3.7   Best Practices

- **Consistent Formatting:** Use a consistent code formatter to maintain uniform code style.

- **Meaningful Commit Messages:** Write clear and descriptive commit messages that explain the purpose of the changes.

- **Modular Code:** Write reusable and modular code to enhance maintainability and scalability.

- **Comprehensive Testing:** Implement thorough tests to ensure the reliability of your contributions.

4   Installation

You can install JSLAB by either downloading the latest stable release from GitHub or by building it from source. Choose the method that best fits your needs.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

4.1 Download the Latest Stable Release

- Visit the JSLAB Releases Page on GitHub Repository:

  `https://github.com/PR-DC/JSLAB/releases`

- Download the appropriate installer and install program.

- Try examples from:

  `https://github.com/PR-DC/JSLAB/tree/master/examples`

4.2 Build from Source

If you prefer to build JSLAB from source, follow the detailed build instructions available in this documentation.

5 Build instructions

5.1 Prerequisites

In order to download necessary tools, clone the repository, and install dependencies via npm, you need network access.

- **Node.js:** Ensure that Node.js is installed on your system. You can download it from the official website:

  `https://nodejs.org/`

- **npm:** npm is typically installed alongside Node.js.

- **node-gyp:** node-gyp is installed alongside with application but it requires additional tools and libraries depending on your operating system. Follow the instructions for your specific OS from:

  `https://github.com/nodejs/node-gyp`

- **Git:** Suggested for cloning the repository. Download it from the official website:

  `https://git-scm.com/`

5.2 Installation Steps

1. Clone the JSLAB repository:

   ```
   git clone git clone https://github.com/PR-DC/JSLAB.git
   ```

2. Navigate to the project directory:

   ```
   cd JSLAB
   ```

3. Install the necessary dependencies:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

```
npm install
```

4. Start the application:

```
npm start
```

5. Check examples from:
   https://github.com/PR-DC/JSLAB/tree/master/examples

6   Contributing

6.1   Setting Up the Development Environment

Follow the detailed build instructions available in this documentation.

6.2   Making Changes

Follow the coding style and best practices available in this documentation.

6.3   Submitting Changes

1. Create a new branch for your feature or bugfix:

```
git checkout -b feature/your-feature-name
```

2. Make your changes and commit them with clear messages:

```
git commit -m "Add feature X to improve Y"
```

3. Push your branch to your forked repository:

```
git push origin feature/your-feature-name
```

4. Submit a Pull Request (PR) detailing your changes.

6.4   Testing

Before submitting a PR, ensure that all tests pass and add new tests for any new functionality you introduce.

6.5   Reviewing Process

All PRs are subject to review by the maintainers. Be prepared to make revisions based on feedback to align with project standards.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

6.6   Best Practices

- **Consistent Formatting:** Use a consistent code formatter (e.g., Prettier) to maintain uniform code style.

- **Meaningful Commit Messages:** Write clear and descriptive commit messages that explain the purpose of the changes.

- **Modular Code:** Write reusable and modular code to enhance maintainability and scalability.

- **Comprehensive Testing:** Implement thorough tests to ensure the reliability of your contributions.

7   Feedback

Your feedback is invaluable in improving the JSLAB Library. Whether you encounter bugs, have feature requests, or need assistance, please reach out through the following channels:

- **GitHub Issues:** Report bugs or suggest features by opening an issue in the GitHub repository.

- **Email**: Contact us directly at info@pr-dc.com or main author at mpetrasinovic@pr-dc.com.

We encourage active participation and appreciate all forms of feedback that help us enhance the functionality and usability of JSLAB.

8   Code references

8.1   basic

`ans`

Type: `<Unknown>`
Stores the result of the last evaluated expression.

`clear()`

Clears all defined variables in the current context.

`clc()`

Clears the console screen.

`cls()`

---

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Clears the console screen. Alias for `clc`.

`version`

Type: `<Unknown>`
Returns the current version of the JSLAB.

`platform`

Type: `<Unknown>`
Returns the platform on which JSLAB is running.

`jsl_file_name`

Type: `<Unknown>`
Returns the file name of the current JSLAB script.

`info()`

Provides information about the current environment.

`settings()`

Accesses or modifies the user settings for JSLAB.

`cmd_help()`

Provides help information for JSLAB commands.

`editor()`

Accesses the code editor interface within JSLAB.

`pwd`

Type: `<Unknown>`
Returns the current working directory.

`breakpoint()`

Sets a breakpoint in the code for debugging.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

`debug_flag`

Type: `<Unknown>`
Returns the current debug flag status.

`debug`

Type: `<Unknown>`
Enables or disables debug mode.

`pause()`

Pauses the execution of the current script.

`stoppoint()`

Sets a stop point in the script execution.

`logpoint()`

Sets a log point to record information during execution.

`updatepoint()`

Updates specific points in the script during execution.

`checkStop()`

Checks if the execution should stop based on conditions.

`endPoint()`

Marks the endpoint of a script or process.

`run(script_path, lines, [silent], [force_run])`

Parameters:

- `script_path <string>`: The path to the script to run.

- `lines <Array.<number>>`: An array of line numbers to run or focus on within the script.

- `silent <boolean>`: Whether to suppress output from the script execution.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `force_run` `<Boolean>`: If true, forces the script to run even if stop conditions are met.

Runs a script from a specified path, optionally focusing on specific lines and controlling output visibility.

`helpToJSON([name], [type])`

Parameters:

- `name` `<string>`: The name of the documentation item.

- `type` `<string>`: The type of the documentation (e.g., 'category').

Returns: `<string>`: The JSON string of the documentation or undefined if not found. Retrieves documentation in JSON format based on the provided name and type.

`help(name, type)`

Parameters:

- `name` `<string>`: The name of the documentation item.

- `type` `<string>`: The type of the documentation.

Returns: `<string>`: The JSON string of the documentation or undefined if not found. Retrieves documentation based on the provided name and type.

`jslFileName()`

Returns: `<string>`: The file name of the JSL script.
Retrieves the file name of the currently active JSL script.

`clearStorage()`

Clears the application's local storage.

`savePath(new_path)`

Parameters:

- `new_path` `<string>`: The path to save.

Saves a path to the application's list of saved paths.

`removePath(saved_path)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `saved_path <string>`: The path to remove.

Removes a previously saved path from the application's list of saved paths.

`cd(new_path)`

Parameters:

- `new_path <string>`: The new path to set as the current working directory.

Changes the current working directory to the specified path.

`workspace()`

Returns: `<Object>`: The current workspace object.
Retrieves the current workspace.

`updateWorkspace()`

Updates the workspace display based on the current state.

`updateFileBrowser()`

Updates the file browser display based on the current state.

`error(msg)`

Parameters:

- `msg <string>`: The error message to display.

Displays an error message.

`disp(msg)`

Parameters:

- `msg <string>`: The message to display.

Displays a general message.

`warn(msg)`

Parameters:

- `msg <string>`: The warning message to display.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Displays a warning message.

`checkStopLoop()`

Verifies if a loop within the script execution should be terminated, typically used to avoid infinite or lengthy unnecessary execution.

`edit([filepath])`

Parameters:

- `filepath <string>`: Path to the file to be opened in the editor.

Opens a specified file in an editor or opens the editor to a default or previously specified file.

`getExamples()`

Returns: `<Array.<string>>`: An array of paths to the example scripts.
Returns a list of all example scripts available within a predefined directory.

`openExample(filename)`

Parameters:

- `filename <string>`: Name of the example file to open.

Opens a specified example script in the editor window.

`showMessageBox(options)`

Parameters:

- `options <Object>`: Configuration options for the message box.

Returns: `<number>`: The index of the button clicked by the user.
Displays a synchronous message box to the user and waits for their response.

`save(file_path, args)`

Parameters:

- `file_path <string>`: Path where the JSON file will be saved.

- `args <string>`: Variables to save. If 'all' is specified, saves all available variables.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Saves specified variables to a JSON file.

`load(args)`

Parameters:

- `args <*>`: A single filename or a scope and filename to specify where to load the variables.

Loads variables from a specified JSON file into the specified scope or the default script context. If an error occurs during file reading or parsing, it logs an error message.

`system(arg)`

Parameters:

- `arg <*>`: The command and its arguments to be executed.

Returns: `<string>`: The output of the executed command.
Executes a system shell command.

`getCompletions(data)`

Parameters:

- `data <Array>`: Data containing the start of the string to complete, context, and keywords.

Returns: `<Array.<string>>`: An array of completion suggestions.
Retrieves completion suggestions based on the current context and input.

`getObjectByProp(obj, prop, val)`

Parameters:

- `obj <Object>`: The object to search through.

- `prop <string>`: The property name to match.

- `val <*>`: The value to match against the property.

Returns: `<Object>`: The found object with key and value, or null if not found.
Retrieves an object by matching a specific property value.

`getObjectsByProp(obj, prop, val)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- obj `<Object>`: The parent object to search through.

- prop `<string>`: The property name to match.

- val `<*>`: The value to match against the property.

Returns: `<Object>`: An object containing all matched key-value pairs.
Retrieves multiple objects from a parent object by matching a specific property value.

`strcmp(x, y)`

Parameters:

- x `<string>`: The first string.

- y `<string>`: The second string.

Returns: `<number>`: The result of the comparison.
Compares two strings lexicographically.

`unrequire(module)`

Parameters:

- module `<string>`: The module to unrequire.

Unloads a previously required module from the cache.

`resetSandbox()`

Resets the sandbox environment to its initial state.

`compileNapi(path, [show_output])`

Parameters:

- path `<string>`: The path to the N-API module.

- show_output `<boolean>`: Whether to show output in the command window.

Returns: `<Array>`: An array containing the result of the compilation and targets.
Compiles a N-API module located at the specified path.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.2   math

`Pi`

Type: `<Unknown>`
Pi number.

`d2r`

Type: `<Unknown>`
Coefficient for converting degrees to radians.

`r2d`

Type: `<Unknown>`
Coefficient for converting radians to degrees.

`seedRandom(args)`

Parameters:

- `args <any>`: Arguments used to seed the random generator.

Returns: `<any>`: The result from the seeded random generator.
Seeds the random number generator with the provided arguments.

`interp(x, y, xq)`

Parameters:

- `x <Array>`: The x-values of the data points.

- `y <Array>`: The y-values of the data points, corresponding to each x-value.

- `xq <Number>`: The x-value for which to interpolate a y-value.

Returns: `<Number>`: The interpolated y-value at xq.
Performs linear interpolation on a set of data points.

`bilinearFunction(x, midPoint, midValue)`

Parameters:

- `x <number>`: The input value for the function.

- `midPoint <number>`: The midpoint of the function where the slope changes.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- midValue <number>: The value of the function at the midpoint.

Returns: <number>: The output value of the bilinear function.
Calculates the output of a bilinear function based on input value, midpoint, and mid-value.

`random([min], [max])`

Parameters:

- min <Number>: The lower bound of the range.

- max <Number>: The upper bound of the range.

Returns: <Number>: A random number within the specified range.
Generates a random number between a specified range.

`randInt([min], [max])`

Parameters:

- min <Number>: The lower bound of the range.

- max <Number>: The upper bound of the range.

Returns: <Number>: A random integer within the specified range.
Generates a random integer within a specified range.

`acosd(x)`

Parameters:

- x <Number>: The value to compute the arc cosine for.

Returns: <Number>: The arc cosine of x in degrees.
Computes the arc cosine of x, with the result in degrees.

`acotd(x)`

Parameters:

- x <Number>: The value to compute the arc cotangent for.

Returns: <Number>: The arc cotangent of x in degrees.
Computes the arc cotangent of x, with the result in degrees.

`acscd(x)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- x <Number>: The value to compute the arc cosecant for.

Returns: <Number>: The arc cosecant of x in degrees.
Computes the arc cosecant of x, with the result in degrees.

`asecd(x)`

Parameters:

- x <Number>: The value to compute the arc secant for.

Returns: <Number>: The arc secant of x in degrees.
Computes the arc secant of x, with the result in degrees.

`asind(x)`

Parameters:

- x <Number>: The value to compute the arc sine for.

Returns: <Number>: The arc sine of x in degrees.
Computes the arc sine of x, with the result in degrees.

`atand(x)`

Parameters:

- x <Number>: The value to compute the arc tangent for.

Returns: <Number>: The arc tangent of x in degrees.
Computes the arc tangent of x, with the result in degrees.

`atan2d(y, x)`

Parameters:

- y <Number>: The y coordinate.

- x <Number>: The x coordinate.

Returns: <Number>: The arc tangent of y/x in degrees.
Computes the arc tangent of the quotient of its arguments, with the result in degrees.

`cosd(x)`

Parameters:

- x <Number>: The angle in degrees.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Number>`: The cosine of x in degrees.
Computes the cosine of x, where x is in degrees.

`cotd(x)`

Parameters:

- `x` `<Number>`: The angle in degrees.

Returns: `<Number>`: The cotangent of x.
Computes the cotangent of x, where x is in degrees.

`cscd(x)`

Parameters:

- `x` `<Number>`: The angle in degrees.

Returns: `<Number>`: The cosecant of x.
Computes the cosecant of x, where x is in degrees.

`secd(x)`

Parameters:

- `x` `<Number>`: The angle in degrees.

Returns: `<Number>`: The secant of x.
Computes the secant of x, where x is in degrees.

`sind(x)`

Parameters:

- `x` `<Number>`: The angle in degrees.

Returns: `<Number>`: The sine of x.
Computes the sine of x, where x is in degrees.

`tand(x)`

Parameters:

- `x` `<Number>`: The angle in degrees.

Returns: `<Number>`: The tangent of x.
Computes the tangent of x, where x is in degrees.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

`polyfit(x, y, n)`

Parameters:

- x `<Array.<number>>`: The array of x-values.

- y `<Array.<number>>`: The array of y-values corresponding to each x-value.

- n `<number>`: The degree of the polynomial to fit.

Returns: `<Array.<number>>`: The coefficients of the fitted polynomial in descending order.
Fits a polynomial of degree n to the given data points and returns the coefficients (highest degree first).

`polyval(p, x_in)`

Parameters:

- p `<Array.<number>>`: The coefficients of the polynomial in descending order.

- x_in `<Array.<number>>`: The array of x-values at which to evaluate the polynomial.

Returns: `<Array.<number>>`: The resulting y-values after evaluating the polynomial at x_in.
Evaluates a polynomial with given coefficients at specified x-values.

`roots(p)`

Parameters:

- p `<Array.<number>>`: Array of polynomial coefficients, ordered from highest degree to constant term.

Returns: `<Array.<number>>`: Array of roots (real or complex) of the polynomial.
Computes the roots of a polynomial with the given coefficients.

`polystr(p, [opts], [opts.x_symbol], [opts.y_symbol], [opts.precision], [opts.lang])`

Parameters:

- p `<Array.<number>>`: An array of polynomial coefficients, ordered from highest degree to constant term.

- opts `<Object>`: Optional settings for the polynomial string.

- opts.x_symbol `<string>`: The symbol to use for the variable x.

- opts.y_symbol `<string>`: The symbol to use for the variable y.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- opts.precision <number>: The number of decimal places for coefficients.

- opts.lang <string>: The language format for the output ('tex', 'c', etc.).

Returns: <string>: The formatted polynomial string.
Generates a string representation of a polynomial based on the provided coefficients and options.

```
polystrc(p, [opts])
```

Parameters:

- p <Array.<number>>: An array of polynomial coefficients, ordered from highest degree to constant term.

- opts <Object>: Optional settings for the polynomial string.

Returns: <string>: The polynomial string formatted for C language.
Generates a C language formatted string representation of a polynomial.

```
polystrtex(p, [opts])
```

Parameters:

- p <Array.<number>>: An array of polynomial coefficients, ordered from highest degree to constant term.

- opts <Object>: Optional settings for the polynomial string.

Returns: <string>: The polynomial string formatted for LaTeX.
Generates a LaTeX formatted string representation of a polynomial.

```
spikeFilter(x, dx_max, n)
```

Parameters:

- x <Array.<number>>: The input sequence of numbers.

- dx_max <number>: The maximum allowed difference between consecutive elements before considering it a spike.

- n <number>: The maximum number of consecutive spikes to correct.

Returns: <Array.<number>>: The filtered sequence with spikes removed.
Filters out spikes in a sequence by replacing sudden large changes with the previous value.

```
fminbnd(func, a, b, [tol])
```

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `func <function>`: The function to minimize. Should accept a single number and return a number.

- `a <number>`: The lower bound of the interval.

- `b <number>`: The upper bound of the interval.

- `tol <number>`: The tolerance for convergence (optional).

Returns: `<number>`: The x-value where the function attains its minimum within [a, b].
Finds the minimum of a univariate function within a specified interval using a bracketing method.

`magnitude(num)`

Parameters:

- `num <number>`: A real number or an object with 'real' and 'imag' properties.

Returns: `<number>`: The magnitude of the number.
Calculates the magnitude (absolute value) of a complex number or a real number.

`compareComplex(a, b)`

Parameters:

- `a <number>`: First number to compare.

- `b <number>`: Second number to compare.

Returns: `<number>`: -1 if a ¡ b, 1 if a ¿ b, 0 if equal.
Compares two numbers (real or complex) according to Octave's rules.

`min(arr)`

Parameters:

- `arr <Array.<(number|Object)>>`: An array of numbers or complex number objects.

Returns: `<number>`: The minimum value found in the array.
Finds the minimum value in an array of numbers, which can include both real numbers and complex numbers. Complex numbers are represented as objects with 'real' and 'imag' properties. The comparison follows Octave's rules: 1. Compare magnitudes (absolute values) of the numbers. 2. If magnitudes are equal, compare real parts. 3. If real parts are equal, compare imaginary parts.

`max(arr)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `arr <Array.<(number|Object)>>`: An array of numbers or complex number objects.

Returns: `<number>`: The maximum value found in the array.
Finds the maximum value in an array of numbers, which can include both real numbers and complex numbers. Complex numbers are represented as objects with 'real' and 'imag' properties. The comparison follows Octave's rules: 1. Compare magnitudes (absolute values) of the numbers. 2. If magnitudes are equal, compare real parts. 3. If real parts are equal, compare imaginary parts.

`real(input)`

Parameters:

- `input <number>`: A number, complex number object, or array thereof.

Returns: `<number>`: The real part(s) of the input.
Extracts the real part of a number or an array of numbers. Handles mixed inputs containing both real numbers and complex numbers.

`imag(input)`

Parameters:

- `input <number>`: A number, complex number object, or array thereof.

Returns: `<number>`: The imaginary part(s) of the input.
Extracts the imaginary part of a number or an array of numbers. Handles mixed inputs containing both real numbers and complex numbers.

`cumtrapz(args)`

Parameters:

- `args <any>`: Arguments required for cumulative trapezoidal integration.

Returns: `<any>`: The result of the cumulative trapezoidal integration.
Performs cumulative trapezoidal integration on the provided data.

`trapz(args)`

Parameters:

- `args <any>`: Arguments required for trapezoidal integration.

Returns: `<any>`: The result of the trapezoidal integration.
Performs trapezoidal integration on the provided data.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.3   non_blocking

`nbwhile(fn)`

Parameters:

- `fn <function>`: A function that returns a boolean value; when false, the loop exits.

Executes a given function in a non-blocking while loop.

`nbrun(fn)`

Parameters:

- `fn <function>`: The function to be executed.

Executes a given function once in a non-blocking manner.

`nbnext(fn)`

Parameters:

- `fn <function>`: The function to execute next.

Schedules the next block of code to be executed in a non-blocking manner.

`waitMSeconds(ms)`

Parameters:

- `ms <Number>`: The number of milliseconds to wait.

Returns: `<Promise.<void>>`: A promise that resolves after the specified time has elapsed.
Waits for a specified number of milliseconds in a non-blocking manner.

`waitSeconds(s)`

Parameters:

- `s <Number>`: The number of seconds to wait.

Returns: `<Promise.<void>>`: A promise that resolves after the specified time has elapsed.
Waits for a specified number of seconds in a non-blocking manner.

`waitMinutes(min)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- min <Number>: The number of minutes to wait.

Returns: <Promise.<void>>: A promise that resolves after the specified time has elapsed. Waits for a specified number of minutes in a non-blocking manner.

initWorker(path)

Parameters:

- path <string>: The path to the module to configure the worker.

Returns: <Worker>: The initialized Worker instance.
Initializes a new worker with the specified module path.

8.4 path

getUniquePath(path)

Parameters:

- path <String>: The base path for which a unique version is required.

Returns: <String>: A unique filesystem path based on the input path.
Generates a unique filesystem path by appending a number to the input path if the original path exists.

getDir(path)

Parameters:

- path <String>: The filesystem path from which to extract the directory.

Returns: <String>: The directory from the given path.
Extracts the directory of file.

getDirName(path)

Parameters:

- path <String>: The filesystem path from which to extract the directory name.

Returns: <String>: The name of the directory from the given path.
Extracts the name of the directory from a given filesystem path.

pathSep()

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<String>`: The path separator character used by the system.
Retrieves the platform-specific path separator character.

`pathFileName(path)`

Parameters:

- `path` `<string>`: The complete file path.

Returns: `<string>`: The file name extracted from the path.
Retrieves the file name from the provided file path.

`pathFileExt(path)`

Parameters:

- `path` `<string>`: The complete file path.

Returns: `<string>`: The file extension extracted from the path.
Retrieves the file extension from the provided file path.

8.5   windows

`active_window`

Type: `<Unknown>`
Current active window ID.

`open_windows`

Type: `<Unknown>`
Array of open windows.

`openWindow(file)`

Parameters:

- `file` `<string>`: The path to the HTML file to open in the new window.

Returns: `<number>`: The identifier (wid) of the newly opened window.
Opens a new window with the specified file.

`closeWindows(wid)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `wid <number>`: Identifier for the window to close.

Closes the specified window.

`openWindow3D([imports])`

Parameters:

- `imports <Array.<Object>>`: An array of import objects specifying modules to import.

Returns: `<Promise.<Object>>`: A promise that resolves to the window object once imports are ready.
Opens a new 3D window and imports specified modules.

`openPlotlyjs()`

Returns: `<Promise.<Window>>`: The window object where Plotly.js is loaded and the plot container is available.
Opens a Plotly.js window and initializes the plot container.

`openWindowBlank()`

Returns: `<Promise.<Object>>`: A promise that resolves to the window object once it is ready.
Opens a new blank window.

8.6 figures

`open_figures`

Type: `<Unknown>`
Array of open figures.

`active_figure`

Type: `<Unknown>`
Current active figure ID.

`figure(id)`

Parameters:

- `id <Number>`: Identifier for the figure.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Number>`: The identifier of the opened or updated figure.
Opens or updates a figure with specified options.

`getFigure(fid)`

Parameters:

- `fid <string>`: The identifier of the figure to retrieve.

Returns: `<Object>`: The figure object if found, otherwise `false`.
Retrieves the figure object associated with the specified figure ID.

`getPlot(fid)`

Parameters:

- `fid <string>`: The identifier of the figure to retrieve the plot for.

Returns: `<Object>`: The plot object if it exists, otherwise `false`.
Retrieves the plot object for a specified figure ID.

`close(id, [type])`

Parameters:

- `id <number>`: The identifier of the figure or window to close. Use "all" to close all.

- `type <string>`: The type of object to close ('figure' or 'window').

Closes a figure or window by its identifier.

`saveFigureDialog(fid)`

Parameters:

- `fid <String>`: The figure identifier.

Opens a dialog for saving a figure in various formats.

`saveFigure(fid, figure_path, size)`

Parameters:

- `fid <String>`: The figure identifier.

- `figure_path <String>`: The path where the figure should be saved.

- `size <Array>`: Optional dimensions [width, height] to use if saving as a PDF.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Saves a figure to a specified path in various formats.

`legend(label)`

Parameters:

- `label <String>`: The label for the x-axis.

Sets the label for the x-axis of the active figure.

`xlabel(label)`

Parameters:

- `label <String>`: The label for the x-axis.

Sets the label for the x-axis of the active figure.

`ylabel(label)`

Parameters:

- `label <String>`: The label for the y-axis.

Sets the label for the y-axis of the active figure.

`zlabel(label)`

Parameters:

- `label <String>`: The label for the z-axis.

Sets the label for the z-axis of the active figure.

`title(label)`

Parameters:

- `label <String>`: The title text.

Sets the title of the active figure.

`xlim(lim)`

Parameters:

- `lim <String>`: x limits.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Sets the xlim of the active figure.

`ylim(lim)`

Parameters:

- `lim <String>`: y limits.

Sets the ylim of the active figure.

`zlim(lim)`

Parameters:

- `lim <String>`: z limits.

Sets the zlim of the active figure.

`view(azimuth, elevation)`

Parameters:

- `azimuth <number>`: The azimuth angle.

- `elevation <number>`: The elevation angle.

Adjusts the view based on azimuth and elevation angles.

`axis(style)`

Parameters:

- `style <Object>`: The style configuration to apply to the axis.

Applies the specified style to the active figure's plot axis.

`printFigure(filename, options)`

Parameters:

- `filename <String>`: The name of the file where the figure should be printed.

- `options <Object>`: Printing options.

Prints the currently active figure to a file.

`plot(traces, options)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `traces <Array>`: Data traces to plot.

- `options <Object>`: Configuration options for plotting.

Returns: `<Number>`: The plot identifier.
Plots data on the active figure.

8.7   time

`timezone`

Type: `<Unknown>`
Current timezone string.

`last_tic`

Type: `<Unknown>`
Last tic timestamp.

`tic()`

Starts a timer for measuring elapsed time. To be used with `toc` to measure time intervals.

`toc([tic])`

Parameters:

- `tic <number>`: The start time in milliseconds from which to calculate elapsed time. If omitted, uses the last time recorded by `tic()`.

Returns: `<number>`: The elapsed time in seconds.
Calculates the wall-clock time elapsed since a specified start time or the last call to `tic()`, in seconds.

`tocms([tic])`

Parameters:

- `tic <number>`: The start time in milliseconds from which to calculate elapsed time. If omitted, uses the last time recorded by `tic()`.

Returns: `<number>`: The elapsed time in milliseconds.
Calculates the wall-clock time elapsed since a specified start time or the last call to `tic()`, in seconds.

`getTimestamp()`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: The current Unix timestamp as an integer.
Gets the current Unix timestamp adjusted for a specified timezone.

`getTime()`

Returns: `<string>`: The current time in 'HH:mm:ss' format.
Gets the current time as a string adjusted for a specified timezone.

`getFullTime()`

Returns: `<string>`: The current time in 'HH:mm:ss.SSS' format.
Gets the current time with milliseconds as a string adjusted for a specified timezone.

`getDate()`

Returns: `<string>`: The current date in 'dd.MM.yyyy.' format.
Gets the current date as a string adjusted for a specified timezone.

`getDateTime()`

Returns: `<string>`: The current date and time in 'dd.MM.yyyy. HH:mm:ss' format.
Gets the current date and time as a string adjusted for a specified timezone.

`getDateTimeFull()`

Returns: `<string>`: The current date and time in 'dd.MM.yyyy. HH:mm:ss.SSS' format.
Gets the current date and time with milliseconds as a string adjusted for a specified timezone.

`getDateTimeStr()`

Returns: `<string>`: The current date and time in 'ddMMyyyy_HHmmss' format for use in filenames.
Gets the current date and time as a string suitable for filenames, adjusted for a specified timezone.

`setTimezone(tz)`

Parameters:

- `tz` `<String>`: The timezone identifier (e.g., "America/New_York", "Europe/Paris") to be set for all time-related operations.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Sets the timezone to be used for time calculations and formatting. This method allows the application to adjust displayed times according to a specific timezone.

8.8   array

`array(A)`

Parameters:

- `A <Iterable>`: The iterable or array-like object to convert.

Returns: `<Array>`: A new array containing the elements from the input.
Converts an iterable or array-like object into a standard array.

`end(A)`

Parameters:

- `A <Array>`: The array from which the last element is to be retrieved.

Returns: `<*>`: The last element of the array. If the array is empty, returns undefined.
Retrieves the last element from the provided array.

`endi(A)`

Parameters:

- `A <Array>`: The array to evaluate.

Returns: `<number>`: The index of the last element.
Returns the index of the last element in the array.

`column(A, col)`

Parameters:

- `A <Array>`: The matrix array.
- `col <number>`: The column index to retrieve.

Returns: `<Array>`: The specified column as an array.
Retrieves a specific column from the provided matrix.

`row(A, row)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `A <Array>`: The matrix array.

- `row <number>`: The row index to retrieve.

Returns: `<Array>`: The specified row as an array.
Retrieves a specific row from the provided matrix.

`index(rows, cols, rows_max)`

Parameters:

- `rows <number>`: The row index or array of row indices.

- `cols <number>`: The column index or array of column indices.

- `rows_max <number>`: The maximum number of rows.

Returns: `<Array.<number>>`: An array of calculated indices.
Generates an array of indices based on rows and columns.

`indexOfAll(A, value)`

Parameters:

- `A <Array>`: The array to search.

- `value <*>`: The value to find.

Returns: `<Array.<number>>`: An array of indices where the value is found.
Finds all indices of a specified value in the array.

`indexOfMulti(A, search_elements, [from_index])`

Parameters:

- `A <Array>`: The array to search.

- `search_elements <Array>`: The sequence of elements to find.

- `from_index <number>`: The index to start the search from.

Returns: `<number>`: The starting index of the found sequence, or -1 if not found.
Finds the index of a sequence of elements in the array.

`setSub(A, indices, B)`

Parameters:

- `A <Array>`: The target array.

---

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- indices `<Array.<number>>`: The indices at which to set values.

- B `<Array>`: The array of values to set.

Sets a subset of array elements based on provided indices.

`getSub(A, indices)`

Parameters:

- A `<Array>`: The source array.

- indices `<Array.<number>>`: The indices of elements to retrieve.

Returns: `<Array>`: An array containing the retrieved elements.
Retrieves a subset of array elements based on provided indices.

`moveElement(A, from_index, to_index)`

Parameters:

- A `<Array>`: The array to modify.

- from_index `<number>`: The index of the element to move.

- to_index `<number>`: The target index where the element should be moved.

Returns: `<Array>`: The modified array with the element moved.
Moves an element within the array from one index to another.

`removeElement(A, index)`

Parameters:

- A `<Array>`: The array to modify.

- index `<number>`: The index of the element to remove.

Returns: `<Array>`: The array after the element has been removed.
Removes an element from the array at the specified index.

`removeElementByValue(A, value)`

Parameters:

- A `<Array>`: The array to modify.

- value `<*>`: The value to remove.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array>`: The array after the value has been removed.
Removes the first occurrence of a specified value from the array.

`removeElementProp(A, B, [prop])`

Parameters:

- `A <Array.<Object>>`: The array to filter.

- `B <Array>`: The array of properties or values to remove.

- `prop <string>`: The property name to check in objects within A.

Returns: `<Array>`: The filtered array.
Removes elements from array A that have properties listed in array B.

`findIndexProp(A, property, value)`

Parameters:

- `A <Array.<Object>>`: The array to search.

- `property <string>`: The property name to compare.

- `value <*>`: The value to match.

Returns: `<number>`: The index of the matching object, or -1 if not found.
Finds the index of the first object in the array where the specified property matches the given value.

`setValueAt(A, indices, value)`

Parameters:

- `A <Array>`: The target array.

- `indices <Array.<number>>`: An array of indices representing the position.

- `value <*>`: The value to set.

Returns: `<*>`: The value that was set.
Sets a value at the specified multi-dimensional indices in the array.

`getValueAt(A, indices)`

Parameters:

- `A <Array>`: The source array.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- indices <Array.<number>>: An array of indices representing the position.

Returns: <*>: The value at the specified indices, or undefined if out of bounds.
Retrieves a value from the array at the specified multi-dimensional indices.

`setVal(A, indices, value)`

Parameters:

- `A <Array>`: The target array.

- `indices <Array.<number>>`: An array of indices representing the position.

- `value <*>`: The value to set.

Returns: <*>: The value that was set.
Alias for setValueAt.

`getVal(A, indices)`

Parameters:

- `A <Array>`: The source array.

- `indices <Array.<number>>`: An array of indices representing the position.

Returns: <*>: The value at the specified indices, or undefined if out of bounds.
Alias for getValueAt.

`createArray(length, [dimensions])`

Parameters:

- `length <number>`: The size of the first dimension.

- `dimensions <number>`: Sizes of subsequent dimensions.

Returns: <Array>: The newly created n-dimensional array.
Creates an n-dimensional array.

`createFilledArray(length, val, [dimensions])`

Parameters:

- `length <number>`: The size of the first dimension.

- `val <*>`: The value to fill the array with.

- `dimensions <number>`: Sizes of subsequent dimensions.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array>`: The filled n-dimensional array.
Creates an n-dimensional array filled with a specific value.

`fill(val, A, length)`

Parameters:

- `val <*>`: The value to fill the array with.

- `A <Array>`: The array to fill.

- `length <number>`: The number of elements to fill.

Fills the array with a specific value.

`NaNs(size)`

Parameters:

- `size <number>`: The size of each dimension.

Returns: `<Array>`: The NaN-filled n-dimensional array.
Creates an n-dimensional array filled with NaN.

`zeros(size)`

Parameters:

- `size <number>`: The size of each dimension.

Returns: `<Array>`: The zero-filled n-dimensional array.
Creates an n-dimensional array filled with zeros.

`ones(size)`

Parameters:

- `size <number>`: The size of each dimension.

Returns: `<Array>`: The one-filled n-dimensional array.
Creates an n-dimensional array filled with ones.

`eye(size)`

Parameters:

- `size <number>`: The size of the identity matrix.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array>`: The identity matrix as a 2D array.
Creates a 2-dimensional identity matrix.

`scale(A, s)`

Parameters:

- `A <Array.<number>>`: The array to scale.

- `s <number>`: The scalar value.

Returns: `<Array.<number>>`: The scaled array.
Scales an array by a scalar.

`linspace(x1, x2, [N])`

Parameters:

- `x1 <number>`: The start value.

- `x2 <number>`: The end value.

- `N <number>`: The number of points.

Returns: `<Array.<number>>`: The linearly spaced vector.
Creates a linearly spaced vector.

`range(args)`

Parameters:

- `args <*>`: The start, end, and optional step for the range.

Returns: `<Array.<number>>`: An array containing numbers within the specified range.
Generates an array of numbers within a specified range.

`colon(x1, x2, dx)`

Parameters:

- `x1 <number>`: The starting value of the sequence.

- `x2 <number>`: The ending value of the sequence.

- `dx <number>`: The increment between values in the sequence.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array.<number>>`: An array of numbers from `x1` to `x2` with step size `dx`.
Generates a sequence of numbers from `x1` to `x2` with increments of `dx`.

`elementWise(func, arrays)`

Parameters:

- `func <function>`: The function to apply to the elements.

- `arrays <Array>`: One or more arrays to process.

Returns: `<Array>`: A new array with the function applied to each corresponding element.
Applies a function to corresponding elements of one or more arrays.

`arrayfun(A, dim, fun)`

Parameters:

- `A <Array>`: The array or matrix to process.

- `dim <number>`: The dimension along which to apply the function.

- `fun <function>`: The function to apply to each element.

Returns: `<Array>`: The result of applying the function to A.
Applies a function to each element of an array or matrix along a specified dimension.

`divideEl(x, y)`

Parameters:

- `x <Array>`: The numerator array or matrix.

- `y <Array>`: The denominator array or matrix.

Returns: `<Array>`: The result of the element-wise division.
Performs element-wise division on two arrays or matrices.

`multiplyEl(x, y)`

Parameters:

- `x <Array>`: The first array or matrix.

- `y <Array>`: The second array or matrix.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array>`: The result of the element-wise multiplication.
Performs element-wise multiplication on two arrays or matrices.

`powEl(x, y)`

Parameters:

- x `<Array>`: The base array or matrix.

- y `<Array>`: The exponent array, matrix, or scalar.

Returns: `<Array>`: The result of the element-wise exponentiation.
Raises elements of an array or matrix to the power of elements in another array or matrix, element-wise.

`dot(x, y)`

Parameters:

- x `<Array>`: The first array or matrix.

- y `<Array>`: The second array or matrix.

Returns: `<Array>`: The result of the element-wise multiplication.
Performs element-wise multiplication on two arrays or matrices.

`lZero(e)`

Parameters:

- e `<number>`: The element to check.

Returns: `<boolean>`: True if greater than zero, otherwise false.
Determines if the element is greater than zero.

`average(arr)`

Parameters:

- arr `<Array.<number>>`: The array to average.

Returns: `<number>`: The average value.
Calculates the average value of an array.

`averageEM(data, alpha)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- data `<Array.<number>>`: The data array.

- alpha `<number>`: The smoothing factor between 0 and 1.

Returns: `<Array.<number>>`: The Exponential Moving Average of the data.
Calculates the Exponential Moving Average of the data.

`isequal(A1, A2)`

Parameters:

- A1 `<Array>`: The first array.

- A2 `<Array>`: The second array.

Returns: `<boolean>`: True if arrays are equal, otherwise false.
Determines if two arrays are equal.

`neg(A)`

Parameters:

- A `<Array.<boolean>>`: The array to negate.

Returns: `<Array.<boolean>>`: The negated array.
Negates the boolean values in an array.

`all(A)`

Parameters:

- A `<Array.<boolean>>`: The array to check.

Returns: `<boolean>`: True if all elements are true, otherwise false.
Determines if all elements in an array evaluate to true.

`any(A)`

Parameters:

- A `<Array.<boolean>>`: The array to check.

Returns: `<boolean>`: True if any element is true, otherwise false.
Determines if any element in an array evaluates to true.

`arrayContains(arr, item)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- arr `<Array>`: The array to search.

- item `<*>`: The item to search for.

Returns: `<boolean>`: True if the item is found, otherwise false.
Checks if the array contains a specified item.

`hasDuplicates(array)`

Parameters:

- array `<Array>`: The array to check for duplicates.

Returns: `<boolean>`: True if duplicates are found, false otherwise.
Checks if an array contains any duplicate elements.

`removeDuplicates(arr)`

Parameters:

- arr `<Array>`: The array from which duplicates are to be removed.

Returns: `<Array>`: An array containing only unique elements from the original array.
Removes duplicate values from an array.

`fliplr(array)`

Parameters:

- array `<Array>`: The matrix array to flip horizontally.

Returns: `<Array>`: The horizontally flipped matrix.
Reverses the order of elements in each row of a matrix.

`movelr(array, n)`

Parameters:

- array `<Array>`: The array to be modified.

- n `<number>`: The number of elements to move from the start to the end.

Returns: `<Array>`: The modified array with the first `n` elements moved to the end.
Moves the first `n` elements of the array to the end.

`plus(a, b)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- a `<number>`: The first operand, scalar or array.

- b `<number>`: The second operand, scalar or array.

Returns: `<number>`: The result of adding `a` and `b`.
Adds two operands, which can be either scalars or arrays. If both operands are arrays, they must be of the same length.

`add(a, b)`

Parameters:

- a `<number>`: The first operand, scalar or array.

- b `<number>`: The second operand, scalar or array.

Returns: `<number>`: The result of adding `a` and `b`.
Adds two operands, which can be either scalars or arrays. If both operands are arrays, they must be of the same length.

`minus(a, b)`

Parameters:

- a `<number>`: The first operand, scalar or array.

- b `<number>`: The second operand, scalar or array.

Returns: `<number>`: The result of subtracting `b` from `a`.
Subtracts the second operand from the first, which can be either scalars or arrays. If both operands are arrays, they must be of the same length.

`subtract(a, b)`

Parameters:

- a `<number>`: The first operand, scalar or array.

- b `<number>`: The second operand, scalar or array.

Returns: `<number>`: The result of subtracting `b` from `a`.
Subtracts the second operand from the first, which can be either scalars or arrays. If both operands are arrays, they must be of the same length.

`cross3D(A, B, cols)`

Parameters:

- A `<Array.<number>>`: The first 3D vector.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `B <Array.<number>>`: The second 3D vector.

- `cols <number>`: The number of columns (typically 1 for single vectors).

Returns: `<Array.<number>>`: The cross product vector.
Computes the cross product of two 3D vectors.

`concatRow(cols_C, args)`

Parameters:

- `cols_C <number>`: The number of columns in the concatenated matrix.

- `args <Array>`: The matrices to concatenate.

Returns: `<Array>`: The concatenated matrix.
Concatenates multiple matrices row-wise.

`concatCol(rows_C, args)`

Parameters:

- `rows_C <number>`: The number of rows in the concatenated matrix.

- `args <Array>`: The matrices to concatenate.

Returns: `<Array>`: The concatenated matrix.
Concatenates multiple matrices column-wise.

`repRow(A, rows)`

Parameters:

- `A <Array>`: The row vector to repeat.

- `rows <number>`: The number of times to repeat the row.

Returns: `<Array>`: The repeated row matrix.
Repeats a row vector multiple times.

`repCol(A, cols)`

Parameters:

- `A <Array>`: The column vector to repeat.

- `cols <number>`: The number of times to repeat the column.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array>`: The repeated column matrix.
Repeats a column vector multiple times.

`sumRow(A, rows, cols)`

Parameters:

- `A <Array.<number>>`: The matrix array.

- `rows <number>`: The number of rows in the matrix.

- `cols <number>`: The number of columns in the matrix.

Returns: `<Array.<number>>`: An array containing the sum of each row.
Sums the elements of each row in a matrix.

`sumCol(A, rows, cols)`

Parameters:

- `A <Array.<number>>`: The matrix array.

- `rows <number>`: The number of rows in the matrix.

- `cols <number>`: The number of columns in the matrix.

Returns: `<Array.<number>>`: An array containing the sum of each column.
Sums the elements of each column in a matrix.

`normRow(A, rows, cols)`

Parameters:

- `A <Array.<number>>`: The matrix array.

- `rows <number>`: The number of rows in the matrix.

- `cols <number>`: The number of columns in the matrix.

Returns: `<Array.<number>>`: An array containing the norm of each row.
Calculates the Euclidean norm of each row in a matrix.

`normCol(A, rows, cols)`

Parameters:

- `A <Array.<number>>`: The matrix array.

- `rows <number>`: The number of rows in the matrix.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- cols <number>: The number of columns in the matrix.

Returns: <Array.<number>>: An array containing the norm of each column.
Calculates the Euclidean norm of each column in a matrix.

```
transpose(A, rows, cols)
```

Parameters:

- A <Array.<number>>: The matrix array to transpose.

- rows <number>: The number of rows in the original matrix.

- cols <number>: The number of columns in the original matrix.

Returns: <Array.<number>>: The transposed matrix array.
Transposes a matrix.

```
multiply(A, B, rows_A, cols_A, cols_B)
```

Parameters:

- A <Array.<number>>: The first matrix array.

- B <Array.<number>>: The second matrix array.

- rows_A <number>: The number of rows in matrix A.

- cols_A <number>: The number of columns in matrix A.

- cols_B <number>: The number of columns in matrix B.

Returns: <Array.<number>>: The resulting matrix array after multiplication.
Multiplies two matrices.

```
dotColumn(A, B, rows, cols)
```

Parameters:

- A <Array.<number>>: The first matrix array.

- B <Array.<number>>: The second matrix array.

- rows <number>: The number of rows in each matrix.

- cols <number>: The number of columns in each matrix.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array.<number>>`: An array containing the dot product of each column.
Performs element-wise dot product on columns of two matrices.

`diag(A, length)`

Parameters:

- `A <Array.<number>>`: The array to form the diagonal.

- `length <number>`: The size of the square matrix.

Returns: `<Array.<number>>`: The diagonal matrix as a 1D array.
Creates a diagonal matrix from a given array.

`linsolve(A, B, N)`

Parameters:

- `A <Array.<number>>`: The coefficient matrix.

- `B <Array.<number>>`: The constant terms.

- `N <number>`: The size of the matrix (NxN).

Returns: `<Array.<number>>`: The solution vector.
Solves a linear system of equations using LU decomposition.

`reciprocal(A, length)`

Parameters:

- `A <Array.<number>>`: The array of numbers.

- `length <number>`: The number of elements in the array.

Returns: `<Array.<number>>`: An array containing the reciprocals of the original elements.
Computes the reciprocal of each element in the array.

`reshape(A, rows, cols)`

Parameters:

- `A <Array>`: The array to reshape.

- `rows <number>`: The number of rows in the new shape.

- `cols <number>`: The number of columns in the new shape.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array>`: The reshaped array.
Reshapes an array into a new dimension.

`maxi(A)`

Parameters:

- `A <Array.<number>>`: The array to search.

Returns: `<Array>`: An array containing the max value and its index.
Finds the maximum element in the array and its index, excluding NaN values.

`mini(A)`

Parameters:

- `A <Array.<number>>`: The array to search.

Returns: `<Array>`: An array containing the min value and its index.
Finds the minimum element in the array and its index, excluding NaN values.

`sorti(A)`

Parameters:

- `A <Array.<number>>`: The array to sort.

Returns: `<Array>`: An array containing the sorted values and their original indices.
Sorts the array in ascending order with indices, excluding NaN values.

`arrayRand(l, u, rows, cols, [randFun])`

Parameters:

- `l <number>`: The lower bound of the range.

- `u <number>`: The upper bound of the range.

- `rows <number>`: The number of rows.

- `cols <number>`: The number of columns.

- `randFun <function>`: The random function to use.

Returns: `<Array.<number>>`: An array filled with random numbers within the specified range.
Generates an array with random floating-point numbers within a specified range.

`arrayRandi(lu, rows, cols, [randFun])`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `lu <Array.<number>>`: An array containing the lower and upper bounds [lower, upper].

- `rows <number>`: The number of rows.

- `cols <number>`: The number of columns.

- `randFun <function>`: The random function to use.

Returns: `<Array.<number>>`: An array filled with random integers within the specified range.

Generates an array with random integer numbers within a specified range.

`normalizeVector(v)`

Parameters:

- `v <Array.<number>>`: The vector to normalize.

Returns: `<Array.<number>>`: The normalized vector.

Normalizes a 3D vector.

`dotVector(a, b)`

Parameters:

- `a <Array.<number>>`: The first vector.

- `b <Array.<number>>`: The second vector.

Returns: `<number>`: The dot product of the vectors.

Computes the dot product of two 3D vectors.

`skewVector(v)`

Parameters:

- `v <Array.<number>>`: The vector to skew.

Returns: `<Array.<number>>`: The skew-symmetric matrix as a 1D array.

Creates a skew-symmetric matrix from a 3D vector.

`meshgrid(x, y)`

Parameters:

- `x <Array.<number>>`: The x-coordinates.

- `y <Array.<number>>`: The y-coordinates.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array>`: The meshgrid arrays [X, Y].
Generates coordinate matrices from coordinate vectors.

`dispMatrix(varname, A, rows, cols)`

Parameters:

- `varname <string>`: The name of the variable.

- `A <Array.<number>>`: The matrix array.

- `rows <number>`: The number of rows in the matrix.

- `cols <number>`: The number of columns in the matrix.

Returns: `<string>`: The string representation of the matrix.
Displays a matrix with a variable name.

`dispRowVector(varname, A, length)`

Parameters:

- `varname <string>`: The name of the variable.

- `A <Array.<number>>`: The row vector array.

- `length <number>`: The length of the row vector.

Returns: `<string>`: The string representation of the row vector.
Displays a row vector with a variable name.

`dispColumnVector(varname, A, length)`

Parameters:

- `varname <string>`: The name of the variable.

- `A <Array.<number>>`: The column vector array.

- `length <number>`: The length of the column vector.

Returns: `<string>`: The string representation of the column vector.
Displays a column vector with a variable name.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.9 color

`colororder`

Type: `<Unknown>`
Color order

`color(id)`

Parameters:

- `id <number>`: Identifier for the color. Can be a numeric index, a predefined color name, or an RGB array.

Returns: `<string>`: The hex code of the color.
Returns a color code based on the provided identifier.

`getColorG2R(value, [k])`

Parameters:

- `value <number>`: A number between 0 and 1 indicating position on the gradient.

- `k <number>`: A scaling factor to adjust the gradient effect.

Returns: `<string>`: The hsl color string.
Calculates a color on a gradient from green to red based on a value.

`colourGradientor(p, rgb_beginning, rgb_end)`

Parameters:

- `p <number>`: The percentage (0 to 1) between the two colors.

- `rgb_beginning <Array.<number>>`: The RGB values of the start color.

- `rgb_end <Array.<number>>`: The RGB values of the end color.

Returns: `<Array.<number>>`: The RGB values of the calculated gradient color.
Calculates the gradient color between two colors based on a percentage.

`rgb2hex(r, g, b)`

Parameters:

- `r <number>`: The red color value (0-255).

- `g <number>`: The green color value (0-255).

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- b `<number>`: The blue color value (0-255).

Returns: `<Array.<number>>`: The HEX representation of the color.
Converts RGB color values to HEX.

`rgbToHsl(r, g, b)`

Parameters:

- r `<number>`: The red color value (0-255).

- g `<number>`: The green color value (0-255).

- b `<number>`: The blue color value (0-255).

Returns: `<Array.<number>>`: The HSL representation of the color.
Converts RGB color values to HSL (Hue, Saturation, Lightness).

`hslToRgb(h, s, l)`

Parameters:

- h `<number>`: The hue value (0-360).

- s `<number>`: The saturation value (0-100).

- l `<number>`: The lightness value (0-100).

Returns: `<Array.<number>>`: The RGB representation of the color.
Converts HSL color values to RGB.

`hueToRgb(v1, v2, vh)`

Parameters:

- v1 `<number>`: Helper value 1.

- v2 `<number>`: Helper value 2.

- vh `<number>`: The hue value to convert.

Returns: `<number>`: The RGB value for the hue.
Helper function for converting a hue to RGB.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.10 conversion

`speak(msg)`

Parameters:

- `msg <string>`: The text message to be spoken.

Converts text to speech using the Web Speech API.

`num2str(num, precision)`

Parameters:

- `num <number>`: The number to convert.

- `precision <number>`: The number of digits after the decimal point.

Returns: `<string>`: The formatted string.
Converts a number to a string with specified precision.

`changeExtension(path, ext_new)`

Parameters:

- `path <string>`: The original file path.

- `ext_new <string>`: The new extension without the dot.

Returns: `<string>`: The file path with the new extension.
Changes the extension of a file path.

`removeExtension(path)`

Parameters:

- `path <string>`: The original file path.

Returns: `<string>`: The file path without extension.
Remove the extension of a file path.

`ned2rpy(roll, pitch, yaw, [A])`

Parameters:

- `roll <number>`: The roll angle in radians.

- `pitch <number>`: The pitch angle in radians.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- yaw `<number>`: The yaw angle in radians.

- A `<Array.<number>>`: Optional matrix to apply transformation to.

Returns: `<Array.<Array.<number>>>`: The transformation matrix.
Transforms coordinates from NED (North, East, Down) frame to RPY (Roll, Pitch, Yaw) frame.

`uint8ToString(data)`

Parameters:

- data `<Uint8Array>`: The array of uint8 numbers.

Returns: `<string>`: The converted string.
Converts an array of uint8 numbers to a string.

`hex2dec(hex)`

Parameters:

- hex `<Array.<string>>`: The array of hexadecimal strings.

Returns: `<Array.<number>>`: The array of decimal numbers.
Converts an array of hexadecimal strings to an array of decimal numbers.

`numToASCII(num)`

Parameters:

- num `<number>`: The number to convert.

Returns: `<string>`: The ASCII character.
Converts a number to its ASCII character equivalent.

`numToHexStr(num, dig, prefix)`

Parameters:

- num `<number>`: The number to convert.

- dig `<number>`: The number of digits in the resulting string.

- prefix `<boolean>`: Whether to add a '0x' prefix.

Returns: `<string>`: The hexadecimal string.
Converts a number to a hexadecimal string with a fixed number of digits.

`int8To2ASCII(num)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- num <number>: The int8 number.

Returns: <string>: The two ASCII characters.
Converts an int8 number to two ASCII characters.

```
int16To4ASCII(num)
```

Parameters:

- num <number>: The int16 number to convert.

Returns: <string>: A string of four ASCII characters.
Converts an int16 number to four ASCII characters representing its hexadecimal value.

```
uint8sToInt16(part1, part2)
```

Parameters:

- part1 <number>: The first uint8 value.

- part2 <number>: The second uint8 value.

Returns: <number>: The combined int16 value.
Combines two uint8 values into an int16 value.

```
uint8sToInt32(part1, part2, part3, part4)
```

Parameters:

- part1 <number>: The first uint8 value.

- part2 <number>: The second uint8 value.

- part3 <number>: The third uint8 value.

- part4 <number>: The fourth uint8 value.

Returns: <number>: The combined int32 value.
Combines four uint8 values into an int32 value.

```
uint8sToFloat(part1, part2, part3, part4)
```

Parameters:

- part1 <number>: The first uint8 value.

- part2 <number>: The second uint8 value.

- part3 <number>: The third uint8 value.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `part4 <number>`: The fourth uint8 value.

Returns: `<number>`: The floating-point number.
Converts four uint8 values into a floating-point number.

uint16ToInt16(num)

Parameters:

- `num <number>`: The uint16 value to convert.

Returns: `<number>`: The converted int16 value.
Converts a uint16 value to an int16 value.

uint8To2ASCII(num)

Parameters:

- `num <number>`: The uint8 number to convert.

Returns: `<string>`: A string containing two ASCII characters representing the hexadecimal value.
Converts a uint8 number to two ASCII characters.

ms2time(ms)

Parameters:

- `ms <number>`: The time in milliseconds.

Returns: `<string>`: The time string.
Converts milliseconds to a time string in mm:ss format.

dec2bin(x)

Parameters:

- `x <number>`: The decimal number.

Returns: `<string>`: The binary string.
Converts a decimal number to a binary string.

dec2hex(x)

Parameters:

- `x <number>`: The decimal number.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<string>`: The hexadecimal string.
Converts a decimal number to a hexadecimal string.

`dec2oct(x)`

Parameters:

- `x <number>`: The decimal number.

Returns: `<string>`: The octal string.
Converts a decimal number to an octal string.

`round(number, [decimals], [string])`

Parameters:

- `number <number>`: The number to round.

- `decimals <number>`: The number of decimal places.

- `string <boolean>`: Whether to return the result as a string.

Returns: `<number>`: The rounded number.
Rounds a number to a specified number of decimal places.

`roundIf(value, p)`

Parameters:

- `value <number>`: The value to round.

- `p <number>`: The number of digits after the decimal point.

Returns: `<number>`: The rounded number with a fixed number of decimal places, or the original value if it is not a number.
Rounds a number to a fixed number of decimal places if it is a number.

`bitString(n)`

Parameters:

- `n <number>`: The number to convert.

Returns: `<string>`: A bit string representing the number.
Converts a uint8_t number to a bit string.

`getBitFlags(map, name_column, val)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- map `<Object>`: The mapping of bit positions to flag names.

- name_column `<string>`: The column name in the mapping that contains the flag names.

- val `<number>`: The bitfield value.

Returns: `<Object>`: An object with keys as flag names and values indicating the presence (1) or absence (0) of each flag.
Generates a set of flags from a bitfield based on a mapping.

`getEnumVal(enum_object, prop, val)`

Parameters:

- enum_object `<Object>`: The enumeration object to search.

- prop `<string>`: The property name to match.

- val `<*>`: The property value to match.

Returns: `<number>`: The enumeration key as a number, or the index if not found.
Retrieves the enumeration value based on a property match.

`invertEnum(enum_object, [prop])`

Parameters:

- enum_object `<Object>`: The enumeration object to invert.

- prop `<string>`: An optional property name to use from the enumeration values.

Returns: `<Object>`: The inverted enumeration object.
Inverts an enumeration, swapping keys and values, optionally based on a specific property of the enumeration values.

`arrayToHexStr(A, [prefix])`

Parameters:

- A `<Array.<number>>`: The array to convert.

- prefix `<boolean>`: Whether to add a "0x" prefix to each hex value.

Returns: `<string>`: A string of hexadecimal values.
Converts an array of numbers to a string of hexadecimal values, optionally prefixed with "0x".

`arrayToASCII(array)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- array <Array.<number>>: The array of numbers to convert.

Returns: <string>: The ASCII string representation of the array.
Converts an array of numbers to an ASCII string.

`extend(objects)`

Parameters:

- objects <Object>: The objects to merge into the target object.

Returns: <Object>: The extended object.
Extends an object with properties from additional objects.

`normalizeRC(rc, [deadzone])`

Parameters:

- rc <number>: The RC input value.

- deadzone <number>: The deadzone value below which the output is set to zero.

Returns: <number>: The normalized value.
Normalizes the value of a radio control (RC) input.

`checkValueUpdate(data)`

Parameters:

- data <Object>: An object containing the current value and the last value.

Returns: <boolean>: True if the value has been updated; false otherwise.
Checks if a value has been updated and updates the last value if it has.

`resetValue(data)`

Parameters:

- data <Object>: The object whose value and last value properties will be reset.

Resets the value and last value properties of an object.

8.11 device

`checkDeviceUSB(VID, PID)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- VID `<string>`: Vendor ID of the USB device.

- PID `<string>`: Product ID of the USB device.

Returns: `<boolean>`: True if the device is found, false otherwise.
Checks if there is a USB device connected with the specified Vendor ID and Product ID.

`checkDeviceSTM([PID])`

Parameters:

- PID `<string>`: Product ID of the USB device, default is for Virtual COM Port.

Returns: `<boolean>`: True if the device is found, false otherwise.
Checks for a connected USB device by STM and an optional Product ID.

`checkDeviceCH340()`

Returns: `<boolean>`: True if the device is found, false otherwise.
Checks if there is a USB device connected using a CH340 chip.

`checkDriver(driver_name)`

Parameters:

- driver_name `<string>`: Name of the driver to check.

Returns: `<boolean>`: True if the driver is found, false otherwise.
Checks if a specific driver is installed on the system.

`checkDriverFTDI()`

Returns: `<boolean>`: True if the drivers are found, false otherwise.
Checks if the drivers for FTDI devices are installed.

`checkDriverCP210x()`

Returns: `<boolean>`: True if the drivers are found, false otherwise.
Checks if the drivers for Silicon Labs CP210x USB to UART bridge are installed.

`checkDriverCH340()`

Returns: `<boolean>`: True if the drivers are found, false otherwise.
Checks if the drivers for CH340 USB to serial converter are installed.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.12   file_system

`readFile(file_path)`

Parameters:

- `file_path <string>`: Path to the file.

Returns: `<Buffer>`: The content of the file or false in case of an error.
Reads the content of a file at the specified path.

`writeFile(file_path, data)`

Parameters:

- `file_path <string>`: The path to the file where data will be written.

- `data <Buffer>`: The data to write to the file.

Returns: `<boolean>`: Returns true if the file was written successfully, false if an error occurred.
Writes data to a specified file synchronously. This method should overwrite the file if it already exists.

`deleteFile(file_path)`

Parameters:

- `file_path <string>`: The path to the file that should be deleted.

Returns: `<boolean>`: Returns true if the file was deleted successfully, false if an error occurred.
Deletes a specified file synchronously.

`deleteDir(file_path)`

Parameters:

- `file_path <string>`: The path to the file that should be deleted.

Returns: `<boolean>`: Returns true if the file was deleted successfully, false if an error occurred.
Deletes a specified file synchronously.

`filesInFolder(folder, ext)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `folder <string>`: Path to the folder.

- `ext <string>`: File extension filter.

Returns: `<Array.<string>>`: Array of file paths matching the extension in the specified folder.
Lists files in a specified folder, optionally filtering by extension.

`chooseFile(options)`

Parameters:

- `options <Object>`: Configuration options for the dialog.

Returns: `<string>`: The selected file path(s) or an empty array if canceled.
Opens a dialog for the user to choose a file, synchronously.

`getDefaultPath(type)`

Parameters:

- `type <string>`: Type of the default path (e.g., 'root', 'documents').

Returns: `<string>`: The default path for the specified type.
Retrieves a default path based on a specified type.

`openFolder(filepath)`

Parameters:

- `filepath <string>`: Path to the folder.

Opens the specified folder in the system's file manager.

`makeDirectory(directory)`

Parameters:

- `directory <string>`: The path where the directory will be created.

Returns: `<boolean>`: True if the directory was successfully created or already exists, false if an error occurred.
Creates a directory at the specified path if it does not already exist. This method delegates the directory creation task to the environment's makeDirectory function.

`mkdir(directory)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- directory <string>: The path where the directory will be created.

Returns: <boolean>: True if the directory was successfully created or already exists, false if an error occurred.
Alias for makeDirectory. Creates a directory at the specified path if it does not already exist. This method delegates the directory creation task to the environment's makeDirectory function.

`openDir(filepath)`

Parameters:

- filepath <string>: Path to the directory.

Opens the specified directory in the system's file manager. Alias for \`openFolder\`.

`showFolder(filepath)`

Parameters:

- filepath <string>: Path to the folder.

Shows the specified folder in the system's file manager. Alias for \`openFolder\`.

`showDir(filepath)`

Parameters:

- filepath <string>: Path to the directory.

Shows the specified directory in the system's file manager. Alias for \`openDir\`.

`openProgramFolder()`

Opens the program's root folder in the system's file manager.

`showFileInFolder(filepath)`

Parameters:

- filepath <string>: Path to the file.

Shows the specified file in its containing folder within the system's file manager.

`showFileInDir(filepath)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `filepath <string>`: Path to the file.

Shows the specified file in its containing directory within the system's file manager. Alias for `showFileInFolder`.

`readcsv(filePath, delimiter)`

Parameters:

- `filePath <string>`: Path to the CSV file.

- `delimiter <string>`: Delimiter used in the CSV file (e.g., ',', ';', '\{}t').

Returns: `<Array.<Object>>`: - Parsed CSV data as an array of objects.
Reads a CSV file and returns a promise that resolves with the parsed data.

`checkDirectory(directory)`

Parameters:

- `directory <string>`: The path to the directory to check.

Checks if the specified directory exists.

`copyDir(src, dest)`

Parameters:

- `src <string>`: The source directory path.

- `dest <string>`: The destination directory path.

Recursively copies a directory from the source path to the destination path.

`copyFolder(src, dest)`

Parameters:

- `src <string>`: The source folder path.

- `dest <string>`: The destination folder path.

Copies a folder from the source path to the destination path.

`cp(src, dest)`

Parameters:

- `src <string>`: The source directory path.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `dest <string>`: The destination directory path.

Copies a directory from the source path to the destination path.

```
copyDir7z(src, dest)
```

Parameters:

- `src <string>`: The source 7z archive path.

- `dest <string>`: The destination directory path.

Copies a 7z archive from the source path to the destination path, extracts it, and removes the archive.

8.13   system

```
system(args)
```

Parameters:

- `args <*>`: Command arguments.

Returns: `<string>`: The output of the command as a string, or false if an error occurred. Executes a system command and returns the output.

```
exec(args)
```

Parameters:

- `args <*>`: Command arguments.

Executes a system command.

```
spawn(args)
```

Parameters:

- `args <*>`: Command arguments.

Executes a system command.

```
getTaskList()
```

Returns: `<Array>`: An array containing the tasklist output or [false, []] if an error occurred. Retrieves a list of all running tasks on the system.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

`isProgramRunning(program_name)`

Parameters:

- `program_name <string>`: The name of the program to check.

Returns: `<Array>`: An array where the first element is a boolean indicating if the program is running, and the second element is an array of process IDs if the program is running. Checks if a specific program is running and retrieves its process IDs.

`killProcess(pid)`

Parameters:

- `pid <number>`: The process ID of the process to kill.

Returns: `<string>`: The output of the kill command as a string, or false if an error occurred.
Attempts to kill a process by its process ID.

8.14 geographic

`calculateTilesBoundingBox(tile_coordinates)`

Parameters:

- `tile_coordinates <Array.<Object>>`: An array of objects with tile coordinates, each having properties `x`, `y`, and `z` for tile X and Y coordinates and zoom level, respectively.

Returns: `<Object>`: An object containing the bounds as an array of `[min_lat, min_lng]` and `[max_lat, max_lng]`, and the center as `[latitude, longitude]`.
Calculates the bounding box and center from an array of tile coordinates.

`offsetLatLon(lat, lon, distance, bearing)`

Parameters:

- `lat <number>`: The latitude of the starting point.

- `lon <number>`: The longitude of the starting point.

- `distance <number>`: The distance from the starting point in meters.

- `bearing <number>`: The bearing in degrees from north.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array.<number>>`: An array containing the latitude and longitude of the calculated point.
Calculates a new latitude and longitude based on a starting point, distance, and bearing using the Haversine formula.

`latLonDistance(lat1, lon1, lat2, lon2)`

Parameters:

- `lat1 <number>`: The latitude of the first point.

- `lon1 <number>`: The longitude of the first point.

- `lat2 <number>`: The latitude of the second point.

- `lon2 <number>`: The longitude of the second point.

Returns: `<number>`: The distance between the two points in meters.
Calculates the distance between two points on Earth using the Haversine formula.

`latLonAltDistance(lat1, lon1, alt1, lat2, lon2, alt2)`

Parameters:

- `lat1 <number>`: The latitude of the first point.

- `lon1 <number>`: The longitude of the first point.

- `alt1 <number>`: The altitude of the first point in meters.

- `lat2 <number>`: The latitude of the second point.

- `lon2 <number>`: The longitude of the second point.

- `alt2 <number>`: The altitude of the second point in meters.

Returns: `<number>`: The 3D distance between the two points in meters.
Calculates the distance between two points on Earth including altitude difference using the Haversine formula.

`checkNewLatLon(latlon)`

Parameters:

- `latlon <Object>`: An object containing the current and last values of latitude and longitude.

Returns: `<boolean>`: True if the latitude and/or longitude values have been updated; false otherwise.
Checks if the latitude and longitude values have been updated.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.15 networking

`CRC_TABLE_XMODEM`

Type: `<Unknown>`
XMODEM CRC table

`crc16xmodem(byte_array)`

Parameters:

- `byte_array <Uint8Array>`: The input data as a byte array.

Returns: `<number>`: The CRC-16/XMODEM checksum as a numeric value.
Calculates the CRC-16/XMODEM checksum of a byte array.

`getIP()`

Returns: `<string>`: The IP address if found, otherwise an empty string.
Retrieves the primary IPv4 address of the current machine.

`pingAddressTCP(host, port, [timeout])`

Parameters:

- `host <string>`: The IP address or hostname to ping.

- `port <number>`: The port number to use for the connection.

- `timeout <number>`: The timeout in milliseconds before the attempt is considered failed.

Returns: `<Promise.<boolean>>`: A promise that resolves to `true` if the connection is successful, `false` otherwise.
Attempts to establish a TCP connection to the specified host and port to check reachability.

`pingAddress(host, timeout)`

Parameters:

- `host <string>`: The IP address or hostname to ping.

- `timeout <number>`: The timeout in milliseconds for the ping command.

Executes a ping command to check if an IP address is reachable.

`pingAddressSync(host, timeout)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- host <string>: The IP address or hostname to ping.

- timeout <number>: The timeout in milliseconds for the ping command.

Executes a ping command synchronized to check if an IP address is reachable.

`findFirstUnusedPort(port, min_port, max_port)`

Parameters:

- port <number>: The starting port number to check.

- min_port <number>: The minimum port number in the range.

- max_port <number>: The maximum port number in the range.

Finds the first unused port within a specified range, checking sequentially from `port` to `max_port`.

`ip2dec(ip, [subnets])`

Parameters:

- ip <string>: The IPv4 address in dot-decimal notation.

- subnets <number>: The number of subnets in the IP address, default is 4.

Returns: <number>: The decimal equivalent of the IPv4 address.
Converts an IPv4 address to its decimal equivalent.

`tcp(host, port, onConnectCallback)`

Parameters:

- host <string>: The hostname or IP address to connect to.

- port <number>: The port number on the host to connect to.

- onConnectCallback <function>: A callback function that is called when the connection is successfully established.

Returns: <PRDC_JSLAB_TCP_CLIENT>: An instance of the TCP client with event handlers set up.
Creates a TCP client for communication with a specified host and port.

`udp(host, port)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- host <string>: The hostname or IP address to connect to.

- port <number>: The port number to connect to.

Returns: <PRDC_JSLAB_UDP>: An instance of the UDP client.
Creates a UDP client for sending data to a specified host and port.

```
udpServer(port)
```

Parameters:

- port <number>: The port number to listen on.

Returns: <PRDC_JSLAB_UDP_SERVER>: An instance of the UDP server.
Creates a UDP server to listen on a specified port.

8.16   format

```
formatBytes(bytes, [decimals])
```

Parameters:

- bytes <Number>: Number of bytes.

- decimals <Number>: Number of decimal places to include in the formatted string.

Returns: <String>: Formatted bytes string with appropriate unit.
Formats the given byte count into a readable string.

```
formatBPS(bps, [decimals])
```

Parameters:

- bps <Number>: Number of bits per second.

- decimals <Number>: Number of decimal places to include in the formatted string.

Returns: <String>: Formatted bits per second string with appropriate unit.
Formats the given bits per second (bps) into a readable string.

```
formatPrefix(number, [decimals])
```

Parameters:

- number <Number>: The number to format.

- decimals <Number>: Number of decimal places to include in the formatted string.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<String>`: Formatted number string with metric prefix.
Formats a number with metric prefixes (k, M, G, etc.) based on its value.

`formatNum(number, [decimals])`

Parameters:

- `number <Number>`: The number to format.

- `decimals <Number>`: The number of decimal places.

Returns: `<String>`: The formatted number as a string.
Formats a number to a specified number of decimal places.

`formatFloatDigits(number, [digits])`

Parameters:

- `number <number>`: The number to format.

- `digits <number>`: The number of significant digits.

Returns: `<string>`: The formatted number as a string.
Formats a floating-point number to have a fixed number of significant digits.

`clip(number, control_number, clip_value, [direction])`

Parameters:

- `number <number>`: The number to clip.

- `control_number <number>`: The reference number for the clipping condition.

- `clip_value <number>`: The value to clip to.

- `direction <boolean>`: The direction of clipping (true for max, false for min).

Returns: `<number>`: The clipped number.
Clips a number to a specified value based on a condition.

`clipHeight(value, [min], [max])`

Parameters:

- `value <number>`: The value to clip.

- `min <number>`: The minimum value of the range.

- `max <number>`: The maximum value of the range.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: The clipped height value.
Clips a height value to a specified range.

`fieldnames(obj)`

Parameters:

- `obj <Object>`: The object from which to extract keys.

Returns: `<Array.<string>>`: An array containing the keys of the object.
Retrieves the field names (keys) of the given object.

`strrep(str, old_string, new_string)`

Parameters:

- `str <string>`: The original string.

- `old_string <string>`: The substring to be replaced.

- `new_string <string>`: The substring to replace with.

Returns: `<string>`: The resulting string after replacements.
Replaces all occurrences of a specified substring within a string.

`clipVal(value, [min], [max])`

Parameters:

- `value <number>`: The value to clip.

- `min <number>`: The minimum value of the range.

- `max <number>`: The maximum value of the range.

Returns: `<number>`: The clipped value.
Clips a value to a specified range.

`toFixedIf(value, p)`

Parameters:

- `value <Number>`: The value to round.

- `p <Number>`: The number of decimal places to round to.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Number>`: The rounded number, or the original number if it wasn't a float.
Rounds a number to a fixed number of decimal places, but only if it's a floating point number.

`getCircularReplacer()`

Returns: `<function>`: A replacer function that can be used with JSON.stringify to handle circular references.
Provides a replacer function for JSON.stringify() to prevent circular references.

`isNaN(value)`

Parameters:

- `value <*>`: The value to check.

Returns: `<boolean>`: True if the value is NaN, false otherwise.
Determines if the provided value is NaN.

`isObject(value, [ignore_array])`

Parameters:

- `value <*>`: The value to check.

- `ignore_array <boolean>`: Whether to consider arrays as not objects.

Returns: `<boolean>`: True if the value is an object, false otherwise.
Determines if the provided value is an object.

`isNumber(value)`

Parameters:

- `value <*>`: The value to check.

Returns: `<boolean>`: True if the value is a number, false otherwise.
Determines if the provided value is a number.

`isString(value)`

Parameters:

- `value <*>`: The value to check.

Returns: `<boolean>`: True if the value is a string, false otherwise.
Determines if the provided value is a string.

`isFunction(value)`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Parameters:

- `value <*>`: The value to check

Returns: `<boolean>`: True if the value is a function, false otherwise.
Determines if the provided value is a function.

`isArray(value)`

Parameters:

- `value <*>`: The value to check.

Returns: `<boolean>`: True if the value is an array, false otherwise.
Determines if the provided value is an array.

`isNull(value)`

Parameters:

- `value <*>`: The value to check.

Returns: `<boolean>`: True if the value is null, false otherwise.
Determines if the provided value is null.

`isEmpty(array)`

Parameters:

- `array <Array>`: The array to check.

Returns: `<boolean>`: True if the array is empty, false otherwise.
Checks if an array is empty.

`isInfinity(value)`

Parameters:

- `value <number>`: The value or array of values to check.

Returns: `<boolean>`: - Returns true if infinite, otherwise false. Returns an array of booleans if input is an array.
Checks if the given value(s) are infinite.

`isNumeric(variable)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `variable <*>`: The variable to check.

Returns: `<boolean>`: True if the variable is numeric, false otherwise.
Checks if a variable is numeric.

`hasKey(object, key)`

Parameters:

- `object <Object>`: The object to check for the presence of the key.

- `key <string>`: The key to check for in the object.

Returns: `<boolean>`: - True if the object has the key, false otherwise.
Checks if the specified object has the given key.

`isUndefined(value)`

Parameters:

- `value <*>`: The value to check.

Returns: `<boolean>`: True if the value is undefined, false otherwise.
Determines if the provided value is undefined.

`normalizeAngle(angle)`

Parameters:

- `angle <number>`: The angle to normalize.

Returns: `<number>`: The normalized angle.
Normalizes an angle to the range of -180 to 180 degrees.

`normalizeAngle360(angle)`

Parameters:

- `angle <number>`: The angle to normalize.

Returns: `<number>`: The normalized angle.
Normalizes an angle to the range of 0 to 360 degrees.

`normalizeLat(latitude, precision)`

Parameters:

- `latitude <number>`: The latitude to normalize.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- precision `<number>`: The precision of the normalization.

Returns: `<number>`: The normalized latitude.
Normalizes latitude to the range of -90 to 90 degrees with specified precision.

`normalizeLon(longitude, precision)`

Parameters:

- longitude `<number>`: The longitude to normalize.

- precision `<number>`: The precision of the normalization.

Returns: `<number>`: The normalized longitude.
Normalizes longitude to the range of -180 to 180 degrees with specified precision.

`checkUndefined(val)`

Parameters:

- val `<*>`: The value to check.

Returns: `<*>`: The original value if not undefined, otherwise an empty string.
Checks a value for undefined and returns an empty string if it is undefined, otherwise returns the value.

`prettyPrint(data)`

Parameters:

- data `<*>`: The data to pretty-print.

Returns: `<Array>`: An array containing the pretty-printed data as a string and a boolean indicating if the data was an object.
Pretty-prints data, converting it into a more readable format for display. Handles strings, objects, and other data types.

`stringify(object)`

Parameters:

- object `<*>`: The object to stringify.

Returns: `<string>`: The JSON string representation of the object or the object itself if not an object.
Converts an object to a JSON string if it is an object, otherwise returns the object as is.

`safeStringify(data, [depth_limit])`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- data <Object>: The object to stringify.

- depth_limit <number>: The maximum depth to traverse in the object, beyond which the traversal is stopped.

Returns: <string>: A JSON string representation of the object, with special handling for deep objects, circular references, and HTML escaping of strings.

Safely serializes an object into a JSON string, handling circular references and deep structures, with depth control. It also escapes strings to prevent HTML injection.

8.17 render

`debounceIn(func, wait)`

Parameters:

- func <function>: The function to debounce.

- wait <number>: The period to wait before allowing another call, in milliseconds.

Returns: <function>: The debounced function.

Debounces a function, ensuring it's only invoked once at the beginning of consecutive calls during the wait period.

`debounceInOut(func, wait)`

Parameters:

- func <function>: The function to debounce.

- wait <number>: The period to wait before allowing another call, in milliseconds.

Returns: <function>: The debounced function.

Debounces a function, calling it at the first and last of consecutive calls during the wait period.

`debounceOut(func, wait)`

Parameters:

- func <function>: The function to debounce.

- wait <number>: The period to wait before allowing another call, in milliseconds.

Returns: <function>: The debounced function.

Debounces a function, ensuring it's only invoked once at the end of consecutive calls during the wait period.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.18 geometry

`getrotation_matrix(a, b)`

Parameters:

- `a <Array.<number>>`: The initial unit vector.

- `b <Array.<number>>`: The target unit vector.

Returns: `<Array.<Array.<number>>>`: - The resulting rotation matrix.
Generates a rotation matrix to rotate from vector a to vector b.

`transform(coordinates, scale_factor, rotation_matrix, translation)`

Parameters:

- `coordinates <Array.<Array.<number>>>`: Array of coordinate points.

- `scale_factor <number>`: Factor by which to scale the coordinates.

- `rotation_matrix <Array.<Array.<number>>>`: Matrix used to rotate the coordinates.

- `translation <Array.<number>>`: Vector used to translate the coordinates.

Returns: `<Array.<Array.<number>>>`: - The transformed coordinates.
Transforms coordinates by scaling, rotating, and translating them.

`createVectors3D(xi, yi, zi, ui, vi, wi, scale, angle_factor, opts)`

Parameters:

- `xi <Array.<number>>`: X coordinates of vector origins.

- `yi <Array.<number>>`: Y coordinates of vector origins.

- `zi <Array.<number>>`: Z coordinates of vector origins.

- `ui <Array.<number>>`: X components of vectors.

- `vi <Array.<number>>`: Y components of vectors.

- `wi <Array.<number>>`: Z components of vectors.

- `scale <number>`: Scale factor for the vectors.

- `angle_factor <number>`: Angle factor for arrowheads.

- `opts <Object>`: Additional plotting options.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Object>`: - An object containing line and head trace data for plotting.
Creates 3D vectors for plotting based on provided parameters.

`symRectangle(W, H, [Z])`

Parameters:

- `W <number>`: Width of the rectangle.

- `H <number>`: Height of the rectangle.

- `Z <number>`: Z-coordinate for the rectangle plane.

Returns: `<Array.<number>>`: - Array of vertex coordinates for the rectangle.
Creates a symmetrical rectangle in 3D space.

`boundary3D(points, [shrink])`

Parameters:

- `points <Array.<Array.<number>>>`: Array of points defining the shape.

- `shrink <number>`: Factor by which to shrink the boundary.

Returns: `<Array>`: - An array containing boundary facets and the volume.
Generates the boundary of a 3D shape based on points and a shrink factor.

`writeOff(filename, vertices, faces)`

Parameters:

- `filename <string>`: The path to the OFF file.

- `vertices <Array.<Array.<number>>>`: Array of vertex coordinates.

- `faces <Array.<Array.<number>>>`: Array of face indices.

Writes geometry data to an OFF file.

`readOff(filename)`

Parameters:

- `filename <string>`: The path to the OFF file.

Returns: `<Object>`: - An object containing vertices and faces arrays.
Reads an OFF file and returns the vertices and faces.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.19 Matrix

`column(index)`

Parameters:

- `index <number>`: The index of the column to extract.

Returns: `<Array>`: The extracted column as an array.
Extracts a specific column from a matrix.

`row(index)`

Parameters:

- `index <number>`: The index of the row to extract.

Returns: `<Array>`: The extracted row as an array.
Extracts a specific row from a matrix.

`length(dim)`

Parameters:

- `dim <number>`: The dimension (0 for rows, 1 for columns).

Returns: `<number>`: The length along the specified dimension.
Gets the length of the matrix along a specified dimension.

`numel()`

Returns: `<number>`: The number of elements.
Gets the number of elements in the matrix.

`size(dim)`

Parameters:

- `dim <number>`: The dimension (0 for rows, 1 for columns).

Returns: `<number>`: The size along the specified dimension.
Gets the size of the matrix along a specified dimension.

`reshape(rows, cols)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- rows <number>: New number of rows.

- cols <number>: New number of columns.

Returns: <PRDC_JSLAB_MATRIX>: The reshaped matrix.
Reshapes the matrix to the specified dimensions.

`repmat(rowReps, colReps)`

Parameters:

- rowReps <number>: Number of row repetitions.

- colReps <number>: Number of column repetitions.

Returns: <PRDC_JSLAB_MATRIX>: The replicated matrix.
Replicates the matrix a specified number of times.

`transpose()`

Returns: <PRDC_JSLAB_MATRIX>: The transposed matrix.
Transposes the matrix.

`inv()`

Returns: <PRDC_JSLAB_MATRIX>: The inverse matrix.
Computes the inverse of the matrix.

`det()`

Returns: <number>: The determinant.
Computes the determinant of the matrix.

`trace()`

Returns: <number>: The trace of the matrix.
Computes the trace of the matrix (sum of diagonal elements).

`norm()`

Returns: <number>: The Frobenius norm.
Computes the Frobenius norm of the matrix.

`powm(p)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `p <number>`: The exponent.

Returns: `<PRDC_JSLAB_MATRIX>`: The resulting matrix.
Raises the matrix to a power.

`expm()`

Returns: `<PRDC_JSLAB_MATRIX>`: The exponential matrix.
Computes the matrix exponential.

`add(A)`

Parameters:

- `A <PRDC_JSLAB_MATRIX>`: The matrix to add.

Returns: `<PRDC_JSLAB_MATRIX>`: The resulting matrix.
Adds two matrices.

`plus(A)`

Parameters:

- `A <PRDC_JSLAB_MATRIX>`: The matrix to add.

Returns: `<PRDC_JSLAB_MATRIX>`: The resulting matrix.
Adds two matrices (alias for add).

`subtract(A)`

Parameters:

- `A <PRDC_JSLAB_MATRIX>`: The matrix to subtract.

Returns: `<PRDC_JSLAB_MATRIX>`: The resulting matrix.
Subtracts matrix A from the current matrix.

`minus(A)`

Parameters:

- `A <PRDC_JSLAB_MATRIX>`: The matrix to subtract.

Returns: `<PRDC_JSLAB_MATRIX>`: The resulting matrix.
Subtracts matrix A from the current matrix (alias for subtract).

`multiply(A)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- A `<PRDC_JSLAB_MATRIX>`: The matrix to multiply with.

Returns: `<PRDC_JSLAB_MATRIX>`: The resulting matrix.
Multiplies two matrices.

`linsolve(B)`

Parameters:

- B `<PRDC_JSLAB_MATRIX>`: The right-hand side matrix.

Returns: `<PRDC_JSLAB_MATRIX>`: The solution matrix.
Solves a linear system.

`divideEl(A)`

Parameters:

- A `<PRDC_JSLAB_MATRIX>`: The matrix or scalar to divide by.

Returns: `<PRDC_JSLAB_MATRIX>`: The resulting matrix.
Divides each element by another matrix or scalar.

`multiplyEl(A)`

Parameters:

- A `<PRDC_JSLAB_MATRIX>`: The matrix or scalar to multiply by.

Returns: `<PRDC_JSLAB_MATRIX>`: The resulting matrix.
Multiplies each element by another matrix or scalar.

`powEl(p)`

Parameters:

- p `<number>`: The exponent.

Returns: `<PRDC_JSLAB_MATRIX>`: The resulting matrix.
Raises each element to a power.

`elementWise(func)`

Parameters:

- func `<function>`: The function to apply.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<PRDC_JSLAB_MATRIX>`: The resulting matrix.
Applies a function to each element of the matrix.

`reciprocal()`

Returns: `<PRDC_JSLAB_MATRIX>`: The matrix with reciprocals.
Computes the reciprocal of each element in the matrix.

`sum()`

Returns: `<number>`: The sum of all elements.
Computes the sum of all elements in the matrix.

`sort([order])`

Parameters:

- `order <string>`: The order of sorting ('asc' or 'desc').

Returns: `<PRDC_JSLAB_MATRIX>`: The sorted matrix.
Sorts the elements of the matrix.

`min()`

Returns: `<number>`: The minimum value.
Finds the minimum element in the matrix.

`max()`

Returns: `<number>`: The maximum value.
Finds the maximum element in the matrix.

`clone()`

Returns: `<PRDC_JSLAB_MATRIX>`: A cloned matrix instance.
Creates a clone of the current matrix.

`index(args)`

Parameters:

- `args <number>`: Row and column indices.

Returns: `<Array>`: The selected elements.
Retrieves elements based on row and column indices.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

`setSub(args)`

Parameters:

- `args <*>`: Indices and values to set.

Sets a subset of the matrix elements.

`getSub(args)`

Parameters:

- `args <*>`: Indices to retrieve.

Returns: `<PRDC_JSLAB_MATRIX>`: The subset matrix.
Gets a subset of the matrix elements.

`toArray()`

Returns: `<Array>`: The matrix data as a two-dimensional array.
Converts the matrix to a two-dimensional array.

`toFlatArray()`

Returns: `<Array>`: The matrix data as a one-dimensional array.
Converts the matrix to a one-dimensional array.

`toString()`

Returns: `<string>`: The string representation of the matrix.
Returns a string representation of the matrix.

`toJSON()`

Returns: `<Array>`: The matrix data as a two-dimensional array.
Returns a JSON representation of the matrix.

`toSafeJSON()`

Returns: `<Array>`: The matrix data as a two-dimensional array.
Returns a safe JSON representation of the matrix.

`toPrettyString()`

Returns: `<string>`: The pretty string representation of the matrix.
Returns a pretty string representation of the matrix.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.20 Vector

`length()`

Returns: `<number>`: The length of the vector.
Calculates the length (magnitude) of the vector.

`norm()`

Returns: `<number>`: The length of the vector.
Calculates the length (magnitude) of the vector.

`add(v)`

Parameters:

- v `<PRDC_JSLAB_VECTOR>`: The second vector.

Returns: `<PRDC_JSLAB_VECTOR>`: The resulting vector after addition.
Adds two vectors and returns the result.

`plus(v)`

Parameters:

- v `<PRDC_JSLAB_VECTOR>`: The second vector.

Returns: `<PRDC_JSLAB_VECTOR>`: The resulting vector after addition.
Adds two vectors and returns the result.

`subtract(v)`

Parameters:

- v `<PRDC_JSLAB_VECTOR>`: The vector to subtract.

Returns: `<PRDC_JSLAB_VECTOR>`: The resulting vector after subtraction.
Subtracts the second vector from the first and returns the result.

`minus(v)`

Parameters:

- v `<PRDC_JSLAB_VECTOR>`: The vector to subtract.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<PRDC_JSLAB_VECTOR>`: The resulting vector after subtraction.
Subtracts the second vector from the first and returns the result.

`scale(scale_x, [scale_y], [scale_z])`

Parameters:

- `scale_x <number>`: The scale factor for the x-component or an object with x, y, z properties.

- `scale_y <number>`: The scale factor for the y-component.

- `scale_z <number>`: The scale factor for the z-component.

Returns: `<PRDC_JSLAB_VECTOR>`: The scaled vector.
Scales a vector by the given factors.

`multiply(s)`

Parameters:

- `s <number>`: The scale factor.

Returns: `<PRDC_JSLAB_VECTOR>`: The scaled vector.
Scales a vector by the given factor.

`divide(s)`

Parameters:

- `s <number>`: The scale factor.

Returns: `<PRDC_JSLAB_VECTOR>`: The scaled vector.
Scales a vector by dividing each element by given factor.

`equals(v)`

Parameters:

- `v <PRDC_JSLAB_VECTOR>`: The second vector.

Returns: `<boolean>`: True if vectors are equal, false otherwise.
Checks if two vectors are equal.

`angleTo(v)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- v <PRDC_JSLAB_VECTOR>: The vector.

Returns: <number>: The angle in degrees.
Calculates the angle of a vector.

`projectTo(v)`

Parameters:

- v <PRDC_JSLAB_VECTOR>: The vector.

Returns: <PRDC_JSLAB_VECTOR>: The projected vector.
Projects vector to given vector.

`angles(v)`

Parameters:

- v <PRDC_JSLAB_VECTOR>: The vector.

Returns: <Array>: The angles azimuth and elevation in degrees.
Calculates the angles of a vector.

`distance(v)`

Parameters:

- v <PRDC_JSLAB_VECTOR>: The second vector.

Returns: <number>: The distance between the two vectors.
Calculates the distance between two vectors.

`dot(v)`

Parameters:

- v <PRDC_JSLAB_VECTOR>: The second vector.

Returns: <number>: The dot product.
Calculates the dot product of two vectors.

`cross(v)`

Parameters:

- v <PRDC_JSLAB_VECTOR>: The second vector.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<PRDC_JSLAB_VECTOR>`: The cross product vector.
Calculates the cross product of two vectors.

`interpolate(v, f)`

Parameters:

- `v <PRDC_JSLAB_VECTOR>`: The ending vector.

- `f <number>`: The interpolation factor.

Returns: `<PRDC_JSLAB_VECTOR>`: The interpolated vector.
Interpolates between two vectors by a factor.

`offset(x, [y], [z])`

Parameters:

- `x <number>`: The amount to offset in the x-direction or an object with x, y, z properties.

- `y <number>`: The amount to offset in the y-direction.

- `z <number>`: The amount to offset in the z-direction.

Returns: `<PRDC_JSLAB_VECTOR>`: The updated vector instance.
Offsets the vector by the given amounts.

`normalize()`

Returns: `<PRDC_JSLAB_VECTOR>`: The normalized vector.
Normalizes the vector to have a length of 1.

`negate()`

Returns: `<PRDC_JSLAB_VECTOR>`: The negated vector.
Negates the vector components.

`clone()`

Returns: `<PRDC_JSLAB_VECTOR>`: A new vector instance with the same components.
Creates a clone of this vector.

`toArray()`

Returns: `<Array.<number>>`: An array containing the x, y, z components of the vector.
Converts the vector to an array.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

`toMatrix()`

Returns: `<Object>`: A matrix representation of the vector.
Converts the vector to a matrix.

`toColMatrix()`

Returns: `<Object>`: A column matrix representation of the vector.
Converts the vector to a column matrix.

`toRowMatrix()`

Returns: `<Object>`: A row matrix representation of the vector.
Converts the vector to a row matrix.

`toString()`

Returns: `<string>`: The string representation in the format 'Vector(x:, y:, z:)'.
Returns a string representation of the vector.

`toJSON()`

Returns: `<string>`: The string representation in the format 'Vector(x:, y:, z:)'.
Returns a string representation of the vector.

`toSafeJSON()`

Returns: `<string>`: The string representation in the format 'Vector(x:, y:, z:)'.
Returns a string representation of the vector.

`toPrettyString()`

Returns: `<string>`: The pretty string representation of the object.
Converts the object to a pretty string representation.

8.21 Symbolic

`setValue(value)`

Parameters:

- `value <*>`: The new value to assign to the symbolic variable.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Sets the value of the symbolic variable.

`toString()`

Returns: `<string>`: The string representation in the format 'Vector(x:, y:, z:)'.
Returns a string representation of the vector.

`toNumeric()`

Returns: `<*>`: The numeric representation of the symbolic value.
Converts the symbolic value to a numeric value.

`toJSON()`

Returns: `<string>`: The JSON string representation of the symbolic value.
Converts the symbolic value to a JSON string.

`toSafeJSON()`

Returns: `<Object>`: The safe JSON representation of the object.
Converts the object to a safe JSON representation.

`toPrettyString()`

Returns: `<string>`: The pretty string representation of the object.
Converts the object to a pretty string representation.

8.22   Window

`open(file)`

Parameters:

- `file` `<string>`: The path to the HTML file to open in the window.

Returns: `<Promise.<void>>`: A promise that resolves when the window is opened and ready.
Opens the window with the specified file.

`focus()`

Returns: `<void>`:
Brings focus to the window.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.23   Figure

`open()`

Opens the figure window if it is not already open.

`focus()`

Brings the figure window to the foreground.

8.24   Plot

`legend(label)`

Parameters:

- label <String>: Label text for the x-axis.

Sets the label for the x-axis of the plot.

`xlabel(label)`

Parameters:

- label <String>: Label text for the x-axis.

Sets the label for the x-axis of the plot.

`ylabel(label)`

Parameters:

- label <String>: Label text for the y-axis.

Sets the label for the y-axis of the plot.

`zlabel(label)`

Parameters:

- label <String>: Label text for the z-axis.

Sets the label for the z-axis of the plot.

`title(label)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `label <String>`: Title text for the plot.

Sets the title of the plot.

`xlim(lim)`

Parameters:

- `lim <String>`: Limits for the x-axis.

Sets the limits for the x-axis of the plot.

`ylim(lim)`

Parameters:

- `lim <String>`: Limits for the y-axis.

Sets the limits for the y-axis of the plot.

`zlim(lim)`

Parameters:

- `lim <String>`: Limits for the z-axis.

Sets the limits for the z-axis of the plot.

`view(azimuth, elevation)`

Parameters:

- `azimuth <number>`: The azimuth angle.

- `elevation <number>`: The elevation angle.

Adjusts the view based on azimuth and elevation angles.

`axis(style)`

Parameters:

- `style <Object>`: The style configuration to set for the axis.

Sets the axis style value and updates the plot layout if the plot is ready.

`print(filename, options)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- filename <String>: The filename for the print job.

- options <Object>: Options for the print job.

Adds a print job to the queue and prints it if the system is ready.

`update(traces, N)`

Parameters:

- traces <Object>: The trace data to be updated in the plot.

- N <number>: The data length or index for updating the plot.

Updates plot data by delegating to the `updateData` method.

`remove()`

Removes the plot from the figure, cleaning up any resources associated with it.

8.25 freecad_link

`start(exe, options)`

Parameters:

- exe <string>: The executable path of FreeCAD.

- options <Object>: Configuration options such as port and host.

Starts the FreeCAD application and establishes a TCP connection for remote procedure calls. Attempts to start FreeCAD if it's not running and connects to its TCP server.

`findServer()`

Attempts to locate the FreeCAD TCP server within the network, respecting the startup timeout. Checks if the TCP server is reachable by sending a 'PING' command.

`send(message, timeout)`

Parameters:

- message <string>: The message to send.

- timeout <number>: Timeout in milliseconds to wait for a response.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Buffer>`: - The response from the server or false if the request times out.
Sends a message to the FreeCAD TCP server and waits for a response. Manages the TCP
communication by ensuring message integrity and handling timeouts.

`inputPraser(message)`

Parameters:

- `message <string>`: The message received from FreeCAD.

Returns: `<Array>`: - An array of parsed message components.
Parses the input from FreeCAD responses to identify errors and data. Splits the message by
'—' and checks for error or data messages.

`showMessage(message)`

Parameters:

- `message <string>`: The message received from FreeCAD.

Displays a message from FreeCAD in the JSLAB interface. Parses and displays messages
specifically tagged as 'MSG' from FreeCAD.

`quit()`

Closes the FreeCAD application gracefully. Sends a quit command and handles the
termination of the TCP connection.

`open(filePath, timeout)`

Parameters:

- `filePath <string>`: The path to the file to be opened.

- `timeout <number>`: Timeout in milliseconds to wait for a response.

Opens a specified file in FreeCAD. Sends a command to open a file and handles responses
to confirm file access.

`importFile(filePath, timeout)`

Parameters:

- `filePath <string>`: The path of the file to be imported.

- `timeout <number>`: Timeout in milliseconds to wait for a response.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Imports a file into the current FreeCAD document. Sends an import command and handles responses to confirm the import operation.

`newDocument(filename, timeout)`

Parameters:

- `filename <string>`: Optional filename for the new document.

- `timeout <number>`: Timeout in milliseconds to wait for a response.

Creates a new document in FreeCAD, optionally specifying a filename. Sends a command to create a new document and handles the document creation response.

`save(timeout)`

Parameters:

- `timeout <number>`: Timeout in milliseconds to wait for a response.

Saves the current document in FreeCAD. Sends a save command and handles responses to confirm the save operation.

`saveAs(filePath, timeout)`

Parameters:

- `filePath <string>`: The new file path for the document.

- `timeout <number>`: Timeout in milliseconds to wait for a response.

Saves the current document in FreeCAD under a new filename. Sends a save as command and handles responses to confirm the operation.

`close(timeout)`

Parameters:

- `timeout <number>`: Timeout in milliseconds to wait for a response.

Closes the current document in FreeCAD. Sends a close command and handles responses to confirm the document closure.

`evaluate(command, timeout)`

Parameters:

- `command <string>`: The command to be evaluated in FreeCAD.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `timeout <number>`: Timeout in milliseconds to wait for a response.

Executes a command in FreeCAD and returns the evaluation result. Sends an evaluate command with the specified command string.

`callScript(script, param, timeout)`

Parameters:

- `script <string>`: The script to run.

- `param <string>`: Parameters to pass to the script.

- `timeout <number>`: Timeout in milliseconds to wait for a response.

Runs a script in FreeCAD with optional parameters. Sends a script command along with parameters and handles the script execution response.

`getArea(timeout)`

Parameters:

- `timeout <number>`: Timeout in milliseconds to wait for a response.

Retrieves the area of the selected object in FreeCAD. Sends a measure area command and parses the response to extract the area value.

`getVolume(timeout)`

Parameters:

- `timeout <number>`: Timeout in milliseconds to wait for a response.

Retrieves the volume of the selected object in FreeCAD. Sends a measure volume command and parses the response to extract the volume value.

8.26   om_link

`start(exe)`

Parameters:

- `exe <string>`: Path to the executable to be run, defaults to the OpenModelica compiler if not provided.

	web: pr-dc.com	**JSLAB v1.0.0**
JSL PR-DC	email: info@pr-dc.com	DOCUMENTATION
	github: github.com/PR-DC	

Starts the interaction with an external executable by initializing the necessary environment and parameters. Launches the executable with the appropriate command-line arguments for interaction via ZMQ. Waits for a port file to be created to establish the ZMQ communication.

`sendExpression(expr)`

Parameters:

- `expr <string>`: The expression to be evaluated.

Returns: `<Promise.<any>>`: - A promise that resolves with the parsed result of the expression evaluation.
Sends an expression to be evaluated by the external executable through the ZMQ connection and waits for the result. Parses the response using a dedicated expression parser.

`ModelicaSystem(filename, modelname, [libraries], [command_line_options])`

Parameters:

- `filename <string>`: The path to the Modelica file.

- `modelname <string>`: The name of the Modelica model.

- `libraries <Array.<string>>`: An array of library paths to load.

- `command_line_options <string>`: Additional command-line options for the simulation.

Initializes and configures a Modelica system with the specified parameters and libraries. Loads necessary files and prepares the environment for simulation.

`BuildModelicaModel()`

Builds the Modelica model by sending the appropriate build command and parsing the resulting XML file.

`getWorkDirectory()`

Returns: `<string>`: - The path to the working directory.
Retrieves the working directory used for temporary files and simulations.

`xmlparse()`

Parses the XML file generated by the Modelica compiler to extract simulation parameters and variables.

`processVariable(scalar)`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Parameters:

- `scalar <Object>`: The scalar variable to process.

Processes a scalar variable from the XML file and categorizes it based on its properties.

`getQuantities([args])`

Parameters:

- `args <Array.<string>>`: An array of quantity names to retrieve. If omitted, returns all quantities.

Returns: `<Array.<Object>>`: - An array of quantity objects.
Retrieves a list of quantities based on the provided arguments.

`getLinearQuantities([args])`

Parameters:

- `args <Array.<string>>`: An array of linear quantity names to retrieve. If omitted, returns all linear quantities.

Returns: `<Array.<Object>>`: - An array of linear quantity objects.
Retrieves a list of linearized quantities based on the provided arguments.

`getParameters([args])`

Parameters:

- `args <string>`: A single parameter name or an array of parameter names to retrieve. If omitted, returns all parameters.

Returns: `<Object>`: - An object containing the requested parameters or a single parameter value.
Retrieves simulation parameters based on the provided arguments.

`getInputs([args])`

Parameters:

- `args <string>`: A single input name or an array of input names to retrieve. If omitted, returns all inputs.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Object>`: - An object containing the requested inputs or a single input value.
Retrieves input variables based on the provided arguments.

`getOutputs([args])`

Parameters:

- `args` `<string>`: A single output name or an array of output names to retrieve. If omitted, returns all outputs.

Returns: `<Object>`: - An object containing the requested outputs or a single output value.
Retrieves output variables based on the provided arguments.

`getContinuous([args])`

Parameters:

- `args` `<string>`: A single continuous variable name or an array of names to retrieve. If omitted, returns all continuous variables.

Returns: `<Object>`: - An object containing the requested continuous variables or a single value.
Retrieves continuous variables based on the provided arguments.

`getSimulationOptions([args])`

Parameters:

- `args` `<string>`: A single simulation option name or an array of names to retrieve. If omitted, returns all simulation options.

Returns: `<Object>`: - An object containing the requested simulation options or a single option value.
Retrieves simulation options based on the provided arguments.

`getLinearizationOptions([args])`

Parameters:

- `args` `<string>`: A single linearization option name or an array of names to retrieve. If omitted, returns all linearization options.

Returns: `<Object>`: - An object containing the requested linearization options or a single option value.
Retrieves linearization options based on the provided arguments.

`setParameters(args)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `args <string>`: A single parameter assignment (e.g., "param=5") or an array of assignments.

Sets simulation parameters based on the provided arguments.

`setSimulationOptions(args)`

Parameters:

- `args <string>`: A single simulation option assignment (e.g., "stepSize=0.01") or an array of assignments.

Sets simulation options based on the provided arguments.

`setLinearizationOptions(args)`

Parameters:

- `args <string>`: A single linearization option assignment or an array of assignments.

Sets linearization options based on the provided arguments.

`setInputs(args)`

Parameters:

- `args <string>`: A single input assignment (e.g., "input1=10") or an array of assignments.

Sets input variables based on the provided arguments.

`createcsvData()`

Creates a CSV file containing input data for the simulation.

`simulate([resultfile], [sim_flags])`

Parameters:

- `resultfile <string>`: The name of the result file to generate.

- `sim_flags <string>`: Additional simulation flags.

Runs the simulation with optional result file and simulation flags.

`linearize()`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array.<Object>>`: - An array containing the A, B, C, and D matrices.
Performs linearization of the model and retrieves the linear matrices.

`getLinearMatrix()`

Returns: `<Array.<Object>>`: - An array containing the A, B, C, and D matrices.
Retrieves the linear A, B, C, and D matrices.

`getLinearMatrixValues(matrix_name)`

Parameters:

- `matrix_name <Object>`: The linear matrix object to convert.

Returns: `<Array.<Array.<number>>>`: - The converted matrix as a 2D array or 0 if empty.
Converts linear matrix data into a two-dimensional array format.

`getlinear_inputs()`

Returns: `<string>`: - The linear input variables or false if the model is not linearized.
Retrieves the linear input variables.

`getlinear_outputs()`

Returns: `<string>`: - The linear output variables or false if the model is not linearized.
Retrieves the linear output variables.

`getlinear_states()`

Returns: `<string>`: - The linear state variables or false if the model is not linearized.
Retrieves the linear state variables.

`getSolutions([args], [resultfile])`

Parameters:

- `args <string>`: A single variable name or an array of names to retrieve solutions for.
  If omitted, retrieves all variables.

- `resultfile <string>`: The path to the result file.

Returns: `<Promise.<any>>`: - A promise that resolves with the simulation results or an
error message.
Retrieves simulation solutions based on the provided arguments and result file.

`createValidNames(name, value, structname)`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Parameters:

- `name <string>`: The original variable name.

- `value <any>`: The value of the variable.

- `structname <string>`: The structure name (e.g., 'continuous', 'parameter').

Creates valid variable names by replacing invalid characters and categorizes them based on the structure name.

`parseExpression(args)`

Parameters:

- `args <string>`: The expression string to parse.

Returns: `<Array>`: - The parsed data which could be an array, an object, or a string. Parses a given expression string into structured data based on predefined formats. Handles various formats including single and nested lists, records, and single elements.

`close()`

Closes the current session safely by cleaning up resources such as temporary files and network connections. Terminates any active processes and removes temporary port files.

8.27 tcp_client

`onConnect()`

Handles successful connection establishment by setting the client's active status to true.

`onError(err)`

Parameters:

- `err <Error>`: The error object that was thrown.

Handles errors by setting the client's active status to false and possibly logging the error.

`onData(data)`

Parameters:

- `data <Buffer>`: The incoming data buffer.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Handles incoming data by appending it to the buffer.

`read([N])`

Parameters:

- `N <number>`: The number of bytes to read. Reads all available bytes if not specified.

Returns: `<Buffer>`: The data read from the buffer.
Reads a specified number of bytes from the buffer.

`write(data)`

Parameters:

- `data <Buffer>`: The data to send over the TCP connection.

Writes data to the TCP connection if the client is active.

`availableBytes()`

Returns: `<number>`: The number of available bytes.
Returns the number of bytes available in the buffer.

`close()`

Closes the TCP connection and cleans up resources.

8.28 udp_client

`onConnect()`

Handles successful connection establishment by setting the client's active status to true.

`onError(err)`

Parameters:

- `err <Error>`: The error object that was thrown.

Handles errors by setting the client's active status to false and possibly logging the error.

`write(data)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- data `<Buffer>`: The data to send over the UDP connection.

Sends data over the UDP connection if the client is active.

`close()`

Closes the UDP connection and cleans up resources.

8.29   udp_server

`onData(data)`

Parameters:

- data `<Buffer>`: The received data buffer.

Called when data is received. Buffers the incoming data for later retrieval.

`read([N])`

Parameters:

- N `<number>`: The maximum number of bytes to read. Reads all available bytes by default.

Returns: `<Array>`: An array containing the first N bytes of buffered data.
Reads a specified number of bytes from the buffer.

`availableBytes()`

Returns: `<number>`: The number of bytes currently stored in the buffer.
Returns the number of bytes available in the buffer.

`close()`

Closes the UDP server and releases any resources.

8.30   mathjs

`isComplex(x)`

Parameters:

- x `<*>`: The value to test.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<boolean>`: Returns true if `x` is a Complex number, false otherwise.
Test whether a value is a Complex number.

isBigNumber(x)

Parameters:

- x `<*>`: The value to test.

Returns: `<boolean>`: Returns true if `x` is a BigNumber, false otherwise.
Test whether a value is a BigNumber.

isFraction(x)

Parameters:

- x `<*>`: The value to test.

Returns: `<boolean>`: Returns true if `x` is a Fraction, false otherwise.
Test whether a value is a Fraction.

isUnit(x)

Parameters:

- x `<*>`: The value to test.

Returns: `<boolean>`: Returns true if `x` is a Unit, false otherwise.
Test whether a value is a Unit.

isMatrix(x)

Parameters:

- x `<*>`: The value to test.

Returns: `<boolean>`: Returns true if `x` is a Matrix, false otherwise.
Test whether a value is a Matrix.

isCollection(x)

Parameters:

- x `<*>`: The value to test.

Returns: `<boolean>`: Returns true if `x` is an Array or Matrix, false otherwise.
Test whether a value is a collection (Array or Matrix).

isDenseMatrix(x)

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a DenseMatrix, false otherwise.
Test whether a value is a DenseMatrix.

```
isSparseMatrix(x)
```

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a SparseMatrix, false otherwise.
Test whether a value is a SparseMatrix.

```
isRange(x)
```

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a Range, false otherwise.
Test whether a value is a Range.

```
isIndex(x)
```

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is an Index, false otherwise.
Test whether a value is an Index.

```
isBoolean(x)
```

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a boolean, false otherwise.
Test whether a value is a boolean.

```
isResultSet(x)
```

Parameters:

- x <*>: The value to test.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: <boolean>: Returns true if `x` is a ResultSet, false otherwise.
Test whether a value is a ResultSet.

isHelp(x)

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a Help object, false otherwise.
Test whether a value is a Help object.

isDate(x)

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a Date object, false otherwise.
Test whether a value is a Date.

isRegExp(x)

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a RegExp object, false otherwise.
Test whether a value is a RegExp.

isAccessorNode(x)

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is an AccessorNode, false otherwise.
Test whether a value is an AccessorNode.

isArrayNode(x)

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is an ArrayNode, false otherwise.
Test whether a value is an ArrayNode.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

`isAssignmentNode(x)`

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is an AssignmentNode, false otherwise.
Test whether a value is an AssignmentNode.

`isBlockNode(x)`

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a BlockNode, false otherwise.
Test whether a value is a BlockNode.

`isConditionalNode(x)`

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a ConditionalNode, false otherwise.
Test whether a value is a ConditionalNode.

`isConstantNode(x)`

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a ConstantNode, false otherwise.
Test whether a value is a ConstantNode.

`isFunctionAssignmentNode(x)`

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a FunctionAssignmentNode, false otherwise.
Test whether a value is a FunctionAssignmentNode.

`isFunctionNode(x)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a FunctionNode, false otherwise.
Test whether a value is a FunctionNode.

`isIndexNode(x)`

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is an IndexNode, false otherwise.
Test whether a value is an IndexNode.

`isNode(x)`

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is a Node, false otherwise.
Test whether a value is a Node.

`isObjectNode(x)`

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is an ObjectNode, false otherwise.
Test whether a value is an ObjectNode.

`isOperatorNode(x)`

Parameters:

- x <*>: The value to test.

Returns: <boolean>: Returns true if `x` is an OperatorNode, false otherwise.
Test whether a value is an OperatorNode.

`isParenthesisNode(x)`

Parameters:

- x <*>: The value to test.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<boolean>`: Returns true if `x` is a ParenthesisNode, false otherwise.
Test whether a value is a ParenthesisNode.

`isRangeNode(x)`

Parameters:

- x `<*>`: The value to test.

Returns: `<boolean>`: Returns true if `x` is a RangeNode, false otherwise.
Test whether a value is a RangeNode.

`isRelationalNode(x)`

Parameters:

- x `<*>`: The value to test.

Returns: `<boolean>`: Returns true if `x` is a RelationalNode, false otherwise.
Test whether a value is a RelationalNode.

`isSymbolNode(x)`

Parameters:

- x `<*>`: The value to test.

Returns: `<boolean>`: Returns true if `x` is a SymbolNode, false otherwise.
Test whether a value is a SymbolNode.

`isChain(x)`

Parameters:

- x `<*>`: The value to test.

Returns: `<boolean>`: Returns true if `x` is a Chain, false otherwise.
Test whether a value is a Chain.

`on(event, callback)`

Parameters:

- event `<string>`: The event name to subscribe to.

- callback `<function>`: The callback function to execute when the event occurs.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<void>`:
Subscribe to an event.

`off(event, [callback])`

Parameters:

- `event <string>`: The event name to unsubscribe from.

- `callback <function>`: The callback function to remove.

Returns: `<void>`:
Unsubscribe from an event.

`once(event, callback)`

Parameters:

- `event <string>`: The event name to subscribe to.

- `callback <function>`: The callback function to execute when the event occurs.

Returns: `<void>`:
Subscribe to an event once.

`emit(event, [data])`

Parameters:

- `event <string>`: The event name to emit.

- `data <*>`: The data to pass to the event handlers.

Returns: `<void>`:
Emit an event, triggering all bound callbacks.

`expression`

Type: `<Unknown>`
An object containing functions for parsing and evaluating expressions.

`import(object, [options])`

Parameters:

- `object <Object>`: An object or array of objects with functions or constants to import.

- `options <Object>`: Optional import options.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<void>`:
Import functions or constants into math.js.

`create([config], [factories])`

Parameters:

- `config <Object>`: Optional configuration options.

- `factories <Array.<function()>>`: Optional list of factories to include.

Returns: `<PRDC_JSLAB_MATHJS_DOC>`: Returns a new instance of math.js.
Create a new, isolated math.js instance.

`factory(name, dependencies, create)`

Parameters:

- `name <string>`: The name of the function.

- `dependencies <Array.<string>>`: Array of dependency names.

- `create <function>`: Function to create the new function.

Returns: `<function>`: Returns the created function.
Factory function to create new functions.

`abs(x)`

Parameters:

- `x <number>`: A number or array with numbers.

Returns: `<number>`: Returns the absolute value of `x`.
Calculate the absolute value of a number.

`AccessorNode(object, index)`

Parameters:

- `object <Node>`: The object being accessed.

- `index <IndexNode>`: The index used to access the object.

Returns: `<AccessorNode>`: Returns a new AccessorNode.
A node representing access to a property or index.

`acos(x)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- x <number>: Function input.

Returns: <number>: The arc cosine of `x`.
Calculate the inverse cosine of a value.

```
acosh(x)
```

Parameters:

- x <number>: Function input.

Returns: <number>: The inverse hyperbolic cosine of `x`.
Calculate the inverse hyperbolic cosine of a value.

```
acot(x)
```

Parameters:

- x <number>: Function input.

Returns: <number>: The inverse cotangent of `x`.
Calculate the inverse cotangent of a value.

```
acoth(x)
```

Parameters:

- x <number>: Function input.

Returns: <number>: The inverse hyperbolic cotangent of `x`.
Calculate the inverse hyperbolic cotangent of a value.

```
acsc(x)
```

Parameters:

- x <number>: Function input.

Returns: <number>: The inverse cosecant of `x`.
Calculate the inverse cosecant of a value.

```
acsch(x)
```

Parameters:

- x <number>: Function input.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: The inverse hyperbolic cosecant of `x`.
Calculate the inverse hyperbolic cosecant of a value.

`addScalar(x, y)`

Parameters:

- x `<number>`: First value to add.

- y `<number>`: Second value to add.

Returns: `<number>`: Sum of `x` and `y`.
Add two scalar values, `x + y`.

`and(x, y)`

Parameters:

- x `<boolean>`: First input.

- y `<boolean>`: Second input.

Returns: `<boolean>`: Returns true when both `x` and `y` are true.
Logical AND. Returns true if both inputs are true.

`apply(x, dim, callback)`

Parameters:

- x `<Array>`: The input array or matrix.

- dim `<number>`: The dimension along which to apply the function.

- callback `<function>`: The function to apply.

Returns: `<Array>`: The result of applying the function along the specified dimension.
Apply a function to each entry in a matrix or array.

`arg(x)`

Parameters:

- x `<number>`: Function input.

Returns: `<number>`: The argument of `x`.
Calculate the argument of a complex number.

`ArrayNode(items)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- items `<Array.<Node>>`: An array of nodes.

Returns: `<ArrayNode>`: Returns a new ArrayNode.
An array node representing an array in the expression tree.

`asec(x)`

Parameters:

- x `<number>`: Function input.

Returns: `<number>`: The inverse secant of `x`.
Calculate the inverse secant of a value.

`asech(x)`

Parameters:

- x `<number>`: Function input.

Returns: `<number>`: The inverse hyperbolic secant of `x`.
Calculate the inverse hyperbolic secant of a value.

`asin(x)`

Parameters:

- x `<number>`: Function input.

Returns: `<number>`: The inverse sine of `x`.
Calculate the inverse sine of a value.

`asinh(x)`

Parameters:

- x `<number>`: Function input.

Returns: `<number>`: The inverse hyperbolic sine of `x`.
Calculate the inverse hyperbolic sine of a value.

`AssignmentNode(object, value)`

Parameters:

- object `<Node>`: The symbol or AccessorNode to assign to.

- value `<Node>`: The value to assign.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<AssignmentNode>`: Returns a new AssignmentNode.
An assignment node representing variable assignment.

`atan(x)`

Parameters:

- x `<number>`: Function input.

Returns: `<number>`: The inverse tangent of `x`.
Calculate the inverse tangent of a value.

`atan2(y, x)`

Parameters:

- y `<number>`: Dividend.

- x `<number>`: Divisor.

Returns: `<number>`: The inverse tangent of `y/x`.
Calculate the inverse tangent of `y/x`.

`atanh(x)`

Parameters:

- x `<number>`: Function input.

Returns: `<number>`: The inverse hyperbolic tangent of `x`.
Calculate the inverse hyperbolic tangent of a value.

`atomicMass`

Type: `<Unknown>`
Atomic mass constant, expressed in kg.

`avogadro`

Type: `<Unknown>`
Avogadro's number, approximately `6.022e23` mol¡sup¿-1¡/sup¿.

`bellNumbers(n)`

Parameters:

- n `<number>`: The input value.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: Returns the nth Bell number.
Compute the Bell Numbers, `B(n)`.

`BigNumber(value)`

Parameters:

- `value <number>`: The numeric value.

Returns: `<BigNumber>`: Returns a new BigNumber instance.
BigNumber constructor.

`bignumber(value)`

Parameters:

- `value <number>`: The numeric value.

Returns: `<BigNumber>`: Returns a BigNumber instance.
Create a BigNumber with arbitrary precision.

`bin(n)`

Parameters:

- `n <number>`: The number to format.

Returns: `<string>`: The binary representation of `n`.
Format a number as binary.

`bitAnd(x, y)`

Parameters:

- `x <number>`: First value.

- `y <number>`: Second value.

Returns: `<number>`: Result of `x AND y`.
Bitwise AND operation.

`bitNot(x)`

Parameters:

- `x <number>`: Input value.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: Result of `NOT x`.
Bitwise NOT operation.

`bitOr(x, y)`

Parameters:

- x `<number>`: First value.

- y `<number>`: Second value.

Returns: `<number>`: Result of `x OR y`.
Bitwise OR operation.

`bitXor(x, y)`

Parameters:

- x `<number>`: First value.

- y `<number>`: Second value.

Returns: `<number>`: Result of `x XOR y`.
Bitwise XOR operation.

`BlockNode(blocks)`

Parameters:

- blocks `<Array.<Object>>`: An array of statements.

Returns: `<BlockNode>`: Returns a new BlockNode.
A node representing a block of statements.

`bohrMagneton`

Type: `<Unknown>`
The Bohr magneton in units of `J/T`.

`bohrRadius`

Type: `<Unknown>`
The Bohr radius in meters.

`boltzmann`

Type: `<Unknown>`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

The Boltzmann constant in `J/K`.

boolean(x)

Parameters:

- x <*>: The value to parse.

Returns: <boolean>: The parsed boolean value.
Parse a value into a boolean.

catalan

Type: <Unknown>
The Catalan's constant.

cbrt(x)

Parameters:

- x <number>: Function input.

Returns: <number>: The cube root of `x`.
Calculate the cube root of a value.

ceil(x)

Parameters:

- x <number>: Input value.

Returns: <number>: The rounded value.
Round a value towards plus infinity.

chain(value)

Parameters:

- value <*>: The initial value of the chain.

Returns: <Chain>: A chain object.
Create a chained operation, allowing to chain methods.

Chain(value)

Parameters:

- value <*>: The initial value.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Chain>`: Returns a new Chain instance.
Chain constructor.

`classicalElectronRadius`

Type: `<Unknown>`
Classical electron radius in meters.

`combinations(n, k)`

Parameters:

- n `<number>`: Total number of items.

- k `<number>`: Number of items to choose.

Returns: `<number>`: Number of possible combinations.
Calculate the number of combinations of n items taken k at a time.

`combinationsWithRep(n, k)`

Parameters:

- n `<number>`: Total number of items.

- k `<number>`: Number of items to choose.

Returns: `<number>`: Number of possible combinations with replacement.
Calculate the number of combinations with replacement of n items taken k at a time.

`compare(x, y)`

Parameters:

- x `<number>`: First value to compare.

- y `<number>`: Second value to compare.

Returns: `<number>`: Returns 1 when x ¿ y, -1 when x ¡ y, and 0 when x == y.
Compare two values numerically.

`compareNatural(x, y)`

Parameters:

- x `<string>`: First string to compare.

- y `<string>`: Second string to compare.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: Returns 1 when x ¿ y, -1 when x ¡ y, and 0 when x == y.
Compare two strings using natural order.

```
compareText(x, y)
```

Parameters:

- x `<string>`: First string to compare.

- y `<string>`: Second string to compare.

Returns: `<number>`: Returns 1 when x ¿ y, -1 when x ¡ y, and 0 when x == y.
Compare two strings lexicographically.

```
compile(expr)
```

Parameters:

- expr `<string>`: The expression to compile.

Returns: `<Object>`: A compiled expression that can be evaluated with `eval`.
Compile an expression into a compiled function for faster evaluation.

```
complex([re], [im])
```

Parameters:

- re `<number>`: Real part or a string representation.

- im `<number>`: Imaginary part.

Returns: `<Complex>`: Returns a Complex number.
Create a complex number.

```
Complex([re], [im])
```

Parameters:

- re `<number>`: Real part or a string representation.

- im `<number>`: Imaginary part.

Returns: `<Complex>`: A new Complex number.
Complex number constructor.

```
composition(n, k)
```

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- n <number>: Total number of items.

- k <number>: Number of parts.

Returns: <number>: Number of compositions.
Calculate the composition count of n into k parts.

`concat(a, b, [dim])`

Parameters:

- a <Array>: First array or matrix.

- b <Array>: Other arrays or matrices.

- dim <number>: Dimension along which to concatenate.

Returns: <Array>: Concatenated array or matrix.
Concatenate matrices or arrays along a specified dimension.

`ConditionalNode(condition, trueExpr, falseExpr)`

Parameters:

- condition <Node>: The condition expression.

- trueExpr <Node>: Expression to evaluate when condition is true.

- falseExpr <Node>: Expression to evaluate when condition is false.

Returns: <ConditionalNode>: A new ConditionalNode instance.
A node representing a conditional expression.

`conductanceQuantum`

Type: <Unknown>
Conductance quantum in Siemens.

`conj(x)`

Parameters:

- x <number>: Input value.

Returns: <number>: The complex conjugate of x.
Compute the complex conjugate of a complex number.

`ConstantNode(value)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `value <number>`: The constant value.

Returns: `<ConstantNode>`: A new ConstantNode instance.
A node representing a constant value.

`cos(x)`

Parameters:

- `x <number>`: Function input.

Returns: `<number>`: The cosine of x.
Calculate the cosine of a value.

`cosh(x)`

Parameters:

- `x <number>`: Function input.

Returns: `<number>`: The hyperbolic cosine of x.
Calculate the hyperbolic cosine of a value.

`cot(x)`

Parameters:

- `x <number>`: Function input.

Returns: `<number>`: The cotangent of x.
Calculate the cotangent of a value.

`coth(x)`

Parameters:

- `x <number>`: Function input.

Returns: `<number>`: The hyperbolic cotangent of x.
Calculate the hyperbolic cotangent of a value.

`coulomb`

Type: `<Unknown>`
Coulomb's constant in Nm^2/C^2.

`count(x)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- x <Array>: The input array or matrix.

Returns: <number>: The number of elements.
Count the number of elements in a matrix or array.

`createUnit(name, definition, [options])`

Parameters:

- name <string>: The name of the new unit.

- definition <string>: Definition of the unit.

- options <Object>: Configuration options.

Returns: <Unit>: The created unit.
Create a user-defined unit and register it with the Unit system.

`csc(x)`

Parameters:

- x <number>: Function input.

Returns: <number>: The cosecant of x.
Calculate the cosecant of a value.

`csch(x)`

Parameters:

- x <number>: Function input.

Returns: <number>: The hyperbolic cosecant of x.
Calculate the hyperbolic cosecant of a value.

`ctranspose(x)`

Parameters:

- x <Array>: The matrix to transpose.

Returns: <Array>: The conjugate transpose of x.
Compute the conjugate transpose of a matrix.

`cube(x)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- x `<number>`: Input value.

Returns: `<number>`: The cube of x.
Compute the cube of a value.

`cumsum(x, [dim])`

Parameters:

- x `<Array>`: The input array or matrix.

- dim `<number>`: Dimension along which to calculate.

Returns: `<Array>`: The cumulative sum.
Compute the cumulative sum of a matrix or array.

`deepEqual(x, y)`

Parameters:

- x `<*>`: First value to compare.

- y `<*>`: Second value to compare.

Returns: `<boolean>`: Returns true if x and y are deep equal.
Test element-wise whether two values are equal.

`DenseMatrix(data)`

Parameters:

- data `<Array>`: The data for the matrix.

Returns: `<DenseMatrix>`: A new DenseMatrix instance.
Dense matrix constructor.

`derivative(expr, variable, [options])`

Parameters:

- expr `<string>`: The expression to differentiate.

- variable `<string>`: The variable with respect to which to differentiate.

- options `<Object>`: Optional options object.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Node>`: The derivative of the expression.
Take the derivative of an expression.

`deuteronMass`

Type: `<Unknown>`
Deuteron mass in kilograms.

`diff(x, [dim])`

Parameters:

- x `<Array>`: Input array or matrix.

- dim `<number>`: Dimension along which to calculate the difference.

Returns: `<Array>`: The differences.
Calculate the differences between adjacent values in a matrix or array.

`divideScalar(x, y)`

Parameters:

- x `<number>`: Numerator.

- y `<number>`: Denominator.

Returns: `<number>`: The result of division.
Divide two scalar values, x / y.

`dotDivide(x, y)`

Parameters:

- x `<Array>`: Numerator matrix.

- y `<Array>`: Denominator matrix.

Returns: `<Array>`: The element-wise division.
Divide two matrices element-wise.

`dotMultiply(x, y)`

Parameters:

- x `<Array>`: First matrix.

- y `<Array>`: Second matrix.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array>`: The element-wise multiplication.
Multiply two matrices element-wise.

`dotPow(x, y)`

Parameters:

- x `<Array>`: Base matrix.

- y `<Array>`: Exponent matrix.

Returns: `<Array>`: The element-wise exponentiation.
Exponentiate two matrices element-wise.

`e`

Type: `<Unknown>`
Euler's number, the base of natural logarithms.

`efimovFactor`

Type: `<Unknown>`
Efimov factor.

`eigs(x)`

Parameters:

- x `<Array>`: A square matrix.

Returns: `<Object>`: An object containing eigenvalues and eigenvectors.
Calculate eigenvalues and eigenvectors of a matrix.

`electricConstant`

Type: `<Unknown>`
Electric constant (vacuum permittivity) in F/m.

`electronMass`

Type: `<Unknown>`
Electron mass in kilograms.

`elementaryCharge`

Type: `<Unknown>`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Elementary charge in coulombs.

equal(x, y)

Parameters:

- x <*>: First value to compare.

- y <*>: Second value to compare.

Returns: <boolean>: Returns true if x equals y.
Test whether two values are equal.

equalScalar(x, y)

Parameters:

- x <number>: First value.

- y <number>: Second value.

Returns: <boolean>: Returns true if x equals y.
Test whether two scalar values are equal.

equalText(x, y)

Parameters:

- x <string>: First string.

- y <string>: Second string.

Returns: <boolean>: Returns true if x equals y.
Test whether two strings are equal.

erf(x)

Parameters:

- x <number>: Input value.

Returns: <number>: The error function evaluated at x.
Calculate the error function of a value.

exp(x)

Parameters:

- x <number>: Exponent.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: The exponential of x.
Calculate the exponential of a value.

`expm1(x)`

Parameters:

- x `<number>`: Input value.

Returns: `<number>`: The result of exp(x) - 1.
Calculate exp(x) - 1.

`factorial(n)`

Parameters:

- n `<number>`: A non-negative integer.

Returns: `<number>`: The factorial of n.
Calculate the factorial of a value.

`false`

Type: `<Unknown>`
Boolean value false.

`faraday`

Type: `<Unknown>`
Faraday constant in C/mol.

`fermiCoupling`

Type: `<Unknown>`
Fermi coupling constant in GeVˆ-2.

`fft(x)`

Parameters:

- x `<Array>`: Input array or matrix.

Returns: `<Array>`: The FFT of x.
Compute the Fast Fourier Transform of a matrix or array.

`FibonacciHeap([compare])`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Parameters:

- compare `<function>`: Comparison function.

Returns: `<FibonacciHeap>`: A new FibonacciHeap instance.
Fibonacci heap data structure.

`filter(x, test)`

Parameters:

- x `<Array>`: The input array or matrix.

- test `<function>`: The test function.

Returns: `<Array>`: The filtered array or matrix.
Filter the items in an array or matrix.

`fineStructure`

Type: `<Unknown>`
Fine-structure constant.

`firstRadiation`

Type: `<Unknown>`
First radiation constant in Wm^2.

`fix(x)`

Parameters:

- x `<number>`: Input value.

Returns: `<number>`: The rounded value.
Round a value towards zero.

`flatten(x)`

Parameters:

- x `<Array>`: The input array or matrix.

Returns: `<Array>`: The flattened array or matrix.
Flatten a multi-dimensional array or matrix into a single dimension.

`floor(x)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- x `<number>`: Input value.

Returns: `<number>`: The rounded value.
Round a value towards negative infinity.

`forEach(x, callback)`

Parameters:

- x `<Array>`: The input array or matrix.

- callback `<function>`: The function to execute on each element.

Returns: `<void>`:
Iterate over each element of a matrix or array.

`format(value, [options])`

Parameters:

- value `<*>`: The value to format.

- options `<Object>`: Formatting options or custom function.

Returns: `<string>`: The formatted value.
Format a value for display.

`fraction(numerator, [denominator])`

Parameters:

- numerator `<number>`: Numerator.

- denominator `<number>`: Denominator.

Returns: `<Fraction>`: A new Fraction instance.
Create a fraction.

`Fraction(numerator, [denominator])`

Parameters:

- numerator `<number>`: Numerator.

- denominator `<number>`: Denominator.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Fraction>`: A new Fraction instance.
Fraction constructor.

`FunctionAssignmentNode(name, params, expr)`

Parameters:

- `name <string>`: The name of the function being assigned.

- `params <Array.<string>>`: An array of parameter names.

- `expr <Node>`: The function expression.

Returns: `<FunctionAssignmentNode>`: A new FunctionAssignmentNode instance.
A node representing a function assignment in the expression tree.

`FunctionNode(name, args)`

Parameters:

- `name <string>`: The name of the function or a SymbolNode.

- `args <Array.<Node>>`: An array of argument nodes.

Returns: `<FunctionNode>`: A new FunctionNode instance.
A node representing a function call in the expression tree.

`gamma(n)`

Parameters:

- `n <number>`: The input value.

Returns: `<number>`: The gamma of n.
Calculate the gamma function of a value.

`gasConstant`

Type: `<Unknown>`
The molar gas constant, in units of J/(molK).

`gcd(args)`

Parameters:

- `args <number>`: Two or more integer numbers.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: The greatest common divisor.
Compute the greatest common divisor of two or more values.

`getMatrixDataType(matrix)`

Parameters:

- `matrix <Matrix>`: The input matrix.

Returns: `<string>`: The data type of the matrix elements.
Get the data type of a matrix.

`gravitationConstant`

Type: `<Unknown>`
Newtonian constant of gravitation, in m^3/(kg s^2).

`gravity`

Type: `<Unknown>`
Acceleration due to gravity on Earth, in m/s^2.

`hartreeEnergy`

Type: `<Unknown>`
Hartree energy, in joules.

`hasNumericValue(x)`

Parameters:

- `x <*>`: The value to test.

Returns: `<boolean>`: Returns true if x is numeric.
Test whether a value is a numeric value.

`Help(value)`

Parameters:

- `value <*>`: The function or object to get help for.

Returns: `<Help>`: A new Help instance.
Help object constructor.

`hex(value)`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Parameters:

- `value <number>`: The value to format.

Returns: `<string>`: The hexadecimal representation.
Format a number as hexadecimal.

`hypot(args)`

Parameters:

- `args <number>`: The input values.

Returns: `<number>`: The hypotenuse.
Calculate the hypotenuse of a list of values.

`i`

Type: `<Unknown>`
The imaginary unit 'i'.

`identity(size, [format])`

Parameters:

- `size <number>`: The size of the matrix.
- `format <string>`: The matrix format.

Returns: `<Matrix>`: An identity matrix.
Create an identity matrix.

`ifft(x)`

Parameters:

- `x <Array>`: Input array or matrix.

Returns: `<Array>`: The inverse FFT of x.
Compute the inverse Fast Fourier Transform.

`im(x)`

Parameters:

- `x <number>`: Input value.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: The imaginary part of x.
Get the imaginary part of a complex number.

`ImmutableDenseMatrix(data)`

Parameters:

- `data <Array>`: The data for the matrix.

Returns: `<ImmutableDenseMatrix>`: A new ImmutableDenseMatrix instance.
Immutable dense matrix constructor.

`Index(ranges)`

Parameters:

- `ranges <Range>`: Ranges or indices.

Returns: `<Index>`: A new Index instance.
Index constructor for matrices.

`IndexNode(dimensions)`

Parameters:

- `dimensions <Array.<Node>>`: The indices.

Returns: `<IndexNode>`: A new IndexNode instance.
A node representing an index operation in the expression tree.

`intersect(a, b)`

Parameters:

- `a <Array>`: First set.
- `b <Array>`: Second set.

Returns: `<Array>`: The intersection of a and b.
Compute the intersection of two sets.

`inverseConductanceQuantum`

Type: `<Unknown>`
Inverse conductance quantum, in ohms.

`invmod(a, m)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- a <number>: The value.

- m <number>: The modulus.

Returns: <number>: The modular inverse.
Calculate the modular inverse of a value.

```
isInteger(x)
```

Parameters:

- x <number>: The value to test.

Returns: <boolean>: Returns true if x is an integer.
Test whether a value is an integer.

```
isNegative(x)
```

Parameters:

- x <number>: The value to test.

Returns: <boolean>: Returns true if x is negative.
Test whether a value is negative.

```
isPositive(x)
```

Parameters:

- x <number>: The value to test.

Returns: <boolean>: Returns true if x is positive.
Test whether a value is positive.

```
isPrime(x)
```

Parameters:

- x <number>: The value to test.

Returns: <boolean>: Returns true if x is prime.
Test whether a number is prime.

```
isZero(x)
```

Parameters:

- x <number>: The value to test.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<boolean>`: Returns true if x is zero.
Test whether a value is zero.

`kldivergence(p, q)`

Parameters:

- p `<Array>`: First probability distribution.

- q `<Array>`: Second probability distribution.

Returns: `<number>`: The KL divergence D¡sub¿KL¡/sub¿(p —— q).
Calculate the Kullback-Leibler divergence between two distributions.

`klitzing`

Type: `<Unknown>`
Von Klitzing constant, in ohms.

`kron(A, B)`

Parameters:

- A `<Array>`: First matrix.

- B `<Array>`: Second matrix.

Returns: `<Matrix>`: The Kronecker product of A and B.
Compute the Kronecker product of two matrices.

`LN10`

Type: `<Unknown>`
Natural logarithm of 10.

`LN2`

Type: `<Unknown>`
Natural logarithm of 2.

`LOG10E`

Type: `<Unknown>`
Base 10 logarithm of e.

`LOG2E`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Type: `<Unknown>`
Base 2 logarithm of e.

`larger(x, y)`

Parameters:

- x `<number>`: First value.

- y `<number>`: Second value.

Returns: `<boolean>`: Returns true if x ¿ y.
Test whether value x is larger than y.

`largerEq(x, y)`

Parameters:

- x `<number>`: First value.

- y `<number>`: Second value.

Returns: `<boolean>`: Returns true if x ¿= y.
Test whether value x is larger than or equal to y.

`lcm(args)`

Parameters:

- args `<number>`: Two or more integer numbers.

Returns: `<number>`: The least common multiple.
Compute the least common multiple of two or more values.

`leafCount(node)`

Parameters:

- node `<Node>`: The root node of the expression tree.

Returns: `<number>`: The number of leaf nodes.
Count the number of leaf nodes in an expression tree.

`leftShift(x, y)`

Parameters:

- x `<number>`: Value to be shifted.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- y `<number>`: Amount of bits to shift.

Returns: `<number>`: The shifted value.
Bitwise left shift operation.

`lgamma(n)`

Parameters:

- n `<number>`: The input value.

Returns: `<number>`: The natural logarithm of the gamma function at n.
Compute the natural logarithm of the gamma function.

`log(x, [base])`

Parameters:

- x `<number>`: Value for which to calculate the logarithm.

- base `<number>`: Base of the logarithm.

Returns: `<number>`: The logarithm of x.
Calculate the natural logarithm of a value.

`log10(x)`

Parameters:

- x `<number>`: Value for which to calculate the logarithm.

Returns: `<number>`: The base-10 logarithm of x.
Calculate the base-10 logarithm of a value.

`log1p(x)`

Parameters:

- x `<number>`: Input value.

Returns: `<number>`: The result of $\log(1 + x)$.
Calculate the natural logarithm of 1 plus a value.

`log2(x)`

Parameters:

- x `<number>`: Value for which to calculate the logarithm.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: The base-2 logarithm of x.
Calculate the base-2 logarithm of a value.

`loschmidt`

Type: `<Unknown>`
Loschmidt constant at 0°C and 1 atm, in m^-3.

`lsolve(L, b)`

Parameters:

- L `<Matrix>`: A lower triangular matrix.

- b `<Matrix>`: A column vector.

Returns: `<Matrix>`: The solution vector x.
Solve a linear system A * x = b where A is a lower triangular matrix.

`lsolveAll(L, b)`

Parameters:

- L `<Matrix>`: A lower triangular matrix.

- b `<Matrix>`: A column vector.

Returns: `<Array>`: An array of solutions.
Find all solutions of a linear system A * x = b where A is a lower triangular matrix.

`lup(A, [threshold])`

Parameters:

- A `<Matrix>`: A square matrix.

- threshold `<number>`: Tolerance threshold.

Returns: `<Object>`: An object containing L, U, and P matrices.
Compute the LU decomposition with partial pivoting.

`lusolve(A, b, [order])`

Parameters:

- A `<Matrix>`: Coefficient matrix.

- b `<Matrix>`: Right-hand side vector or matrix.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- order <string>: Matrix storage order.

Returns: <Matrix>: The solution vector or matrix.
Solve a linear system using LU decomposition.

`lyap(A, Q)`

Parameters:

- A <Matrix>: A square matrix.

- Q <Matrix>: A square matrix.

Returns: <Matrix>: Solution X of the Lyapunov equation.
Solve the Lyapunov equation A*X + X*A' = Q.

`mad(array, [dim])`

Parameters:

- array <Array>: Input array.

- dim <number>: Dimension along which to compute.

Returns: <number>: The median absolute deviation.
Compute the median absolute deviation of a set of values.

`magneticConstant`

Type: <Unknown>
Magnetic constant (vacuum permeability), in N/Aˆ2.

`magneticFluxQuantum`

Type: <Unknown>
Magnetic flux quantum, in Wb.

`map(x, callback)`

Parameters:

- x <Array>: The input array or matrix.

- callback <function>: The function to apply.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Array>`: The result after applying the callback.
Map a function over the elements of a matrix or array.

`matrix([data], [format])`

Parameters:

- `data <Array>`: The data for the matrix.

- `format <string>`: The matrix format.

Returns: `<Matrix>`: A new Matrix instance.
Create a matrix.

`Matrix([data], [format])`

Parameters:

- `data <Array>`: The data for the matrix.

- `format <string>`: The matrix format.

Returns: `<Matrix>`: A new Matrix instance.
Matrix constructor.

`matrixFromColumns(columns)`

Parameters:

- `columns <Array>`: Columns to construct the matrix.

Returns: `<Matrix>`: The constructed matrix.
Create a matrix from given columns.

`matrixFromFunction(size, callback)`

Parameters:

- `size <Array>`: The size of the matrix.

- `callback <function>`: Function to generate matrix entries.

Returns: `<Matrix>`: The generated matrix.
Create a matrix using a provided function.

`matrixFromRows(rows)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- rows `<Array>`: Rows to construct the matrix.

Returns: `<Matrix>`: The constructed matrix.
Create a matrix from given rows.

```
mean(array, [dim])
```

Parameters:

- array `<Array>`: Input array.

- dim `<number>`: Dimension along which to compute.

Returns: `<number>`: The mean value.
Compute the arithmetic mean of a set of values.

```
median(array, [dim])
```

Parameters:

- array `<Array>`: Input array.

- dim `<number>`: Dimension along which to compute.

Returns: `<number>`: The median value.
Compute the median of a set of values.

```
mod(x, y)
```

Parameters:

- x `<number>`: Dividend.

- y `<number>`: Divisor.

Returns: `<number>`: The remainder after division.
Calculate the modulus of two numbers.

```
mode(array)
```

Parameters:

- array `<Array>`: Input array.

Returns: `<Array>`: The mode(s) of the array.
Compute the mode of a set of values.

```
molarMass
```

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Type: `<Unknown>`
Molar mass constant, in kg/mol.

`molarMassC12`

Type: `<Unknown>`
Molar mass of carbon-12, in kg/mol.

`molarPlanckConstant`

Type: `<Unknown>`
Molar Planck constant, in J·s/mol.

`molarVolume`

Type: `<Unknown>`
Molar volume of an ideal gas at 1 atm and 0℃, in m³/mol.

`multinomial(args)`

Parameters:

- `args <number>`: Integer numbers.

Returns: `<number>`: The multinomial coefficient.
Compute the multinomial coefficient of a list of integers.

`multiplyScalar(x, y)`

Parameters:

- `x <number>`: First value.

- `y <number>`: Second value.

Returns: `<number>`: The product of x and y.
Multiply two scalar values, x * y.

`neutronMass`

Type: `<Unknown>`
Neutron mass, in kilograms.

`not(x)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- x `<boolean>`: Input value.

Returns: `<boolean>`: The logical negation of x.
Logical NOT operation.

`nthRoot(a, [root])`

Parameters:

- a `<number>`: The value.

- root `<number>`: The root.

Returns: `<number>`: The nth root of a.
Calculate the nth root of a value.

`nthRoots(a, n)`

Parameters:

- a `<number>`: The value.

- n `<number>`: The degree of the root.

Returns: `<Array>`: An array of the n roots.
Calculate the nth roots of a complex number.

`nuclearMagneton`

Type: `<Unknown>`
Nuclear magneton, in J/T.

`null`

Type: `<Unknown>`
JavaScript null value.

`number(value)`

Parameters:

- value `<*>`: The value to parse.

Returns: `<number>`: The numeric value.
Parse a value into a number.

`numeric(value)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- value <*>: The value to convert.

Returns: <number>: The numeric representation.
Convert a math.js data type to a numeric type.

`ObjectNode(properties)`

Parameters:

- properties <Object>: Object properties as nodes.

Returns: <ObjectNode>: A new ObjectNode instance.
A node representing an object in the expression tree.

`oct(value)`

Parameters:

- value <number>: The value to format.

Returns: <string>: The octal representation.
Format a number as octal.

`OperatorNode(op, fn, args, [implicit])`

Parameters:

- op <string>: The operator symbol.

- fn <string>: The function name.

- args <Array.<Node>>: An array of argument nodes.

- implicit <boolean>: Is the operator implicit?

Returns: <OperatorNode>: A new OperatorNode instance.
A node representing an operator in the expression tree.

`or(x, y)`

Parameters:

- x <boolean>: First value.

- y <boolean>: Second value.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<boolean>`: The logical OR of x and y.
Logical OR operation.

`ParenthesisNode(content)`

Parameters:

- `content <Node>`: The node encapsulated by the parentheses.

Returns: `<ParenthesisNode>`: A new ParenthesisNode instance.
A node representing parentheses in the expression tree.

`parse(expr)`

Parameters:

- `expr <string>`: The expression to parse.

Returns: `<Node>`: The parsed expression node(s).
Parse and evaluate an expression.

`parser()`

Returns: `<Parser>`: A new Parser instance.
Create a parser with its own scope and functions.

`Parser()`

Returns: `<Parser>`: A new Parser instance.
Parser constructor.

`partitionSelect(x, k, [compare])`

Parameters:

- `x <Array>`: The input array or matrix.

- `k <number>`: The kth smallest value to select.

- `compare <function>`: Optional comparison function.

Returns: `<number>`: The selected element.
Select an element from a matrix or array based on its sorted position.

`permutations(n, [k])`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- n `<number>`: Total number of items.

- k `<number>`: Number of items to choose.

Returns: `<number>`: Number of possible permutations.
Calculate the number of permutations of n items taken k at a time.

`phi`

Type: `<Unknown>`
The golden ratio, approximately 1.618.

`pi`

Type: `<Unknown>`
The mathematical constant pi.

`pickRandom(array, [number])`

Parameters:

- array `<Array>`: Array to pick values from.

- number `<number>`: Number of values to pick.

Returns: `<*>`: Picked value(s).
Randomly pick one or more values from a list.

`pinv(x)`

Parameters:

- x `<Array>`: A matrix.

Returns: `<Matrix>`: The pseudoinverse of x.
Compute the pseudoinverse of a matrix.

`planckCharge`

Type: `<Unknown>`
Planck charge, in coulombs.

`planckConstant`

Type: `<Unknown>`
Planck constant, in joule seconds.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

`planckLength`

Type: `<Unknown>`
Planck length, in meters.

`planckMass`

Type: `<Unknown>`
Planck mass, in kilograms.

`planckTemperature`

Type: `<Unknown>`
Planck temperature, in kelvin.

`planckTime`

Type: `<Unknown>`
Planck time, in seconds.

`polynomialRoot(coefficients)`

Parameters:

- `coefficients <Array>`: Coefficients of the polynomial.

Returns: `<Array>`: An array of roots.
Find roots of a univariate polynomial.

`pow(x, y)`

Parameters:

- `x <number>`: Base.

- `y <number>`: Exponent.

Returns: `<number>`: x raised to the power y.
Calculate the power of x to y, x^y.

`prod(array, [dim])`

Parameters:

- `array <Array>`: Input array.

---

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `dim <number>`: Dimension along which to compute the product.

Returns: `<number>`: The product of all values.
Compute the product of a set of values.

`protonMass`

Type: `<Unknown>`
Proton mass, in kilograms.

`qr(x)`

Parameters:

- `x <Matrix>`: A matrix.

Returns: `<Object>`: An object containing Q and R matrices.
Compute the QR decomposition of a matrix.

`quantileSeq(data, prob, [sorted])`

Parameters:

- `data <Array>`: Input data.

- `prob <number>`: Probability or array of probabilities.

- `sorted <boolean>`: Is data sorted?

Returns: `<number>`: The quantile(s).
Compute the quantile of a sequence.

`quantumOfCirculation`

Type: `<Unknown>`
Quantum of circulation, in m^2/s.

`randomInt([min], max)`

Parameters:

- `min <number>`: Minimum value, inclusive.

- `max <number>`: Maximum value, inclusive.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: A random integer.
Generate a random integer between min and max.

`Range(start, end, [step])`

Parameters:

- `start <*>`: Start of the range.

- `end <*>`: End of the range.

- `step <*>`: Step size.

Returns: `<Range>`: A new Range instance.
Create a range.

`RangeNode(start, end, [step])`

Parameters:

- `start <Node>`: Start node.

- `end <Node>`: End node.

- `step <Node>`: Step node.

Returns: `<RangeNode>`: A new RangeNode instance.
A node representing a range in the expression tree.

`rationalize(expr, [scope], [options])`

Parameters:

- `expr <string>`: The expression to rationalize.

- `scope <Object>`: Scope of variables.

- `options <Object>`: Options object.

Returns: `<Object>`: An object with expression and denominator.
Rationalize an expression.

`re(x)`

Parameters:

- `x <number>`: Input value.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: The real part of x.
Get the real part of a complex number.

`reducedPlanckConstant`

Type: `<Unknown>`
Reduced Planck constant, in joule seconds.

`RelationalNode(condition, params)`

Parameters:

- `condition <string>`: Relational condition.

- `params <Node>`: Parameters.

Returns: `<RelationalNode>`: A new RelationalNode instance.
A node representing a relational operation.

`replacer(key, value)`

Parameters:

- `key <string>`: Property name.

- `value <*>`: Property value.

Returns: `<*>`: The transformed value.
Replacer function for JSON serialization.

`resize(x, size, [defaultValue])`

Parameters:

- `x <Array>`: The input matrix.

- `size <Array>`: The new size.

- `defaultValue <*>`: Default value for new entries.

Returns: `<Array>`: The resized matrix.
Resize a matrix.

`resolve(name)`

Parameters:

- `name <string>`: The name to resolve.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<*>`: The resolved value.
Resolve the value of a symbol or function.

`ResultSet(entries)`

Parameters:

- `entries <Array>`: The entries in the result set.

Returns: `<ResultSet>`: A new ResultSet instance.
ResultSet constructor.

`reviver(key, value)`

Parameters:

- `key <string>`: Property name.

- `value <*>`: Property value.

Returns: `<*>`: The transformed value.
Reviver function for JSON deserialization.

`rightArithShift(x, y)`

Parameters:

- `x <number>`: The value to shift.

- `y <number>`: The amount to shift.

Returns: `<number>`: The shifted value.
Bitwise right arithmetic shift operation.

`rightLogShift(x, y)`

Parameters:

- `x <number>`: The value to shift.

- `y <number>`: The amount to shift.

Returns: `<number>`: The shifted value.
Bitwise right logical shift operation.

`rotate(x, [turns])`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- x `<Array>`: The input matrix.

- turns `<number>`: Number of 90-degree rotations.

Returns: `<Array>`: The rotated matrix.
Rotate the elements of a matrix.

`rotationMatrix(theta)`

Parameters:

- theta `<number>`: Rotation angle in radians.

Returns: `<Matrix>`: The rotation matrix.
Create a 2D rotation matrix.

`rydberg`

Type: `<Unknown>`
Rydberg constant, in mˆ-1.

`SQRT1_2`

Type: `<Unknown>`
Square root of 1/2.

`SQRT2`

Type: `<Unknown>`
Square root of 2.

`sackurTetrode`

Type: `<Unknown>`
Sackur-Tetrode constant at 1 atm, in J/K.

`schur(A)`

Parameters:

- A `<Matrix>`: A square matrix.

Returns: `<Object>`: An object containing matrices U and T.
Compute the Schur decomposition of a matrix.

`sec(x)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- x <number>: Function input.

Returns: <number>: The secant of x.
Calculate the secant of a value.

`sech(x)`

Parameters:

- x <number>: Function input.

Returns: <number>: The hyperbolic secant of x.
Calculate the hyperbolic secant of a value.

`secondRadiation`

Type: <Unknown>
Second radiation constant, in m·K.

`setCartesian(sets)`

Parameters:

- sets <Array>: The sets to compute the product of.

Returns: <Array>: The Cartesian product.
Create the Cartesian product of two or more sets.

`setDifference(a, b)`

Parameters:

- a <Array>: First set.
- b <Array>: Second set.

Returns: <Array>: The difference a $\setminus\{\}$ b.
Compute the difference between two sets.

`setDistinct(a)`

Parameters:

- a <Array>: Input set.

Returns: <Array>: A set with distinct elements.
Remove duplicate elements from a set.

`setIntersect(a, b, [others])`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Parameters:

- a `<Array>`: First set.

- b `<Array>`: Second set.

- others `<Array>`: Additional sets.

Returns: `<Array>`: The intersection of the sets.
Compute the intersection of two or more sets.

`setIsSubset(a, b)`

Parameters:

- a `<Array>`: Potential subset.

- b `<Array>`: Superset.

Returns: `<boolean>`: Returns true if a is a subset of b.
Test whether a set is a subset of another set.

`setMultiplicity(e, multiset)`

Parameters:

- e `<*>`: Element to count.

- multiset `<Array>`: The multiset.

Returns: `<number>`: The multiplicity of e.
Count the multiplicity of an element in a multiset.

`setPowerset(set)`

Parameters:

- set `<Array>`: The input set.

Returns: `<Array>`: The power set.
Compute the power set of a set.

`setSize(set)`

Parameters:

- set `<Array>`: The input set.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: The number of elements.
Get the size of a set.

`setSymDifference(a, b)`

Parameters:

- a `<Array>`: First set.

- b `<Array>`: Second set.

Returns: `<Array>`: The symmetric difference.
Compute the symmetric difference of two sets.

`setUnion(a, b, [others])`

Parameters:

- a `<Array>`: First set.

- b `<Array>`: Second set.

- others `<Array>`: Additional sets.

Returns: `<Array>`: The union of the sets.
Compute the union of two or more sets.

`sign(x)`

Parameters:

- x `<number>`: Input value.

Returns: `<number>`: The sign of x.
Compute the sign of a number.

`simplify(expr, [rules], [scope])`

Parameters:

- expr `<string>`: The expression to simplify.

- rules `<Object>`: Simplification rules.

- scope `<Object>`: Scope for variables.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<Node>`: The simplified expression.
Simplify an expression.

`simplifyConstant(expr, [options])`

Parameters:

- `expr <Node>`: The expression to simplify.

- `options <Object>`: Simplification options.

Returns: `<Node>`: The simplified expression.
Simplify an expression containing constants.

`simplifyCore(expr)`

Parameters:

- `expr <Node>`: The expression to simplify.

Returns: `<Node>`: The simplified expression.
Perform core simplifications on an expression.

`sin(x)`

Parameters:

- `x <number>`: Function input.

Returns: `<number>`: The sine of x.
Calculate the sine of a value.

`sinh(x)`

Parameters:

- `x <number>`: Function input.

Returns: `<number>`: The hyperbolic sine of x.
Calculate the hyperbolic sine of a value.

`slu(A, order, threshold)`

Parameters:

- `A <Matrix>`: A sparse matrix.

- `order <number>`: Ordering and analysis control parameter.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- threshold <number>: Partial pivoting threshold.

Returns: <Object>: The LU decomposition.
Sparse Linear Solver using LU decomposition.

`smaller(x, y)`

Parameters:

- x <number>: First value.

- y <number>: Second value.

Returns: <boolean>: Returns true if x ¡ y.
Test whether value x is smaller than y.

`smallerEq(x, y)`

Parameters:

- x <number>: First value.

- y <number>: Second value.

Returns: <boolean>: Returns true if x ¡= y.
Test whether value x is smaller than or equal to y.

`Spa(A)`

Parameters:

- A <Matrix>: A sparse matrix.

Returns: <Object>: The LU decomposition.
Compute the Sparse LU decomposition of a matrix.

`sparse([data], [datatype])`

Parameters:

- data <Array>: The data for the matrix.

- datatype <string>: The data type.

Returns: <SparseMatrix>: A new SparseMatrix instance.
Create a sparse matrix.

`SparseMatrix([data])`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- data `<Object>`: The data for the matrix.

Returns: `<SparseMatrix>`: A new SparseMatrix instance.
Sparse matrix constructor.

`speedOfLight`

Type: `<Unknown>`
Speed of light in vacuum, in m/s.

`splitUnit(unit, parts)`

Parameters:

- unit `<Unit>`: The unit to split.

- parts `<Array>`: Array of units to split into.

Returns: `<Array>`: Array of units.
Split a unit into its numeric value and unit string.

`sqrt(x)`

Parameters:

- x `<number>`: Value for which to calculate the square root.

Returns: `<number>`: The square root of x.
Calculate the square root of a value.

`sqrtm(x)`

Parameters:

- x `<Array>`: A square matrix.

Returns: `<Matrix>`: The principal square root of x.
Calculate the principal square root of a matrix.

`square(x)`

Parameters:

- x `<number>`: Input value.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<number>`: The square of x.
Compute the square of a value.

`squeeze(x)`

Parameters:

- x `<Array>`: The input array or matrix.

Returns: `<Array>`: The squeezed array or matrix.
Remove singleton dimensions from an array or matrix.

`std(array, [normalization], [dim])`

Parameters:

- array `<Array>`: Input array.

- normalization `<string>`: Normalization mode.

- dim `<number>`: Dimension along which to compute.

Returns: `<number>`: The standard deviation.
Compute the standard deviation of a set of values.

`stefanBoltzmann`

Type: `<Unknown>`
Stefan-Boltzmann constant, in W/(m^2K^4).

`stirlingS2(n, k)`

Parameters:

- n `<number>`: Total number of objects.

- k `<number>`: Number of non-empty subsets.

Returns: `<number>`: The Stirling number.
Stirling numbers of the second kind.

`string(value)`

Parameters:

- value `<*>`: The value to convert.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<string>`: The string representation of value.
Parse a value into a string.

`subset(x, index, [replacement], [defaultValue])`

Parameters:

- x `<Array>`: The input matrix.

- index `<Index>`: The index.

- replacement `<*>`: The replacement value.

- defaultValue `<boolean>`: Default value for missing entries.

Returns: `<*>`: The subset or updated matrix.
Get or set a subset of a matrix or array.

`sylvester(A, B, C)`

Parameters:

- A `<Matrix>`: A square matrix.

- B `<Matrix>`: A square matrix.

- C `<Matrix>`: A matrix.

Returns: `<Matrix>`: Solution X of the Sylvester equation.
Solve the Sylvester equation A*X + X*B = C.

`SymbolNode(name)`

Parameters:

- name `<string>`: The symbol name.

Returns: `<SymbolNode>`: A new SymbolNode instance.
A node representing a symbol in the expression tree.

`symbolicEqual(expr1, expr2, [options])`

Parameters:

- expr1 `<Node>`: First expression.

- expr2 `<Node>`: Second expression.

- options `<Object>`: Comparison options.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<boolean>`: Returns true if expressions are symbolically equal.
Test whether two expressions are symbolically equal.

`tan(x)`

Parameters:

- x `<number>`: Function input.

Returns: `<number>`: The tangent of x.
Calculate the tangent of a value.

`tanh(x)`

Parameters:

- x `<number>`: Function input.

Returns: `<number>`: The hyperbolic tangent of x.
Calculate the hyperbolic tangent of a value.

`tau`

Type: `<Unknown>`
The constant tau, equal to 2pi.

`thomsonCrossSection`

Type: `<Unknown>`
Thomson cross section, in m^2.

`to(x, unit)`

Parameters:

- x `<Unit>`: The unit to be converted.

- unit `<Unit>`: Target unit.

Returns: `<Unit>`: The converted unit.
Convert a unit to another unit.

`true`

Type: `<Unknown>`
Boolean value true.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

`typeOf(x)`

Parameters:

- x <*>: The variable.

Returns: <string>: The type of x.
Get the type of a variable.

`typed(name, signatures)`

Parameters:

- name <string>: Function name.

- signatures <Object>: Function signatures.

Returns: <function>: The typed function.
Create a typed-function.

`unaryMinus(x)`

Parameters:

- x <number>: Input value.

Returns: <number>: The negated value.
Invert the sign of a value.

`unaryPlus(x)`

Parameters:

- x <number>: Input value.

Returns: <number>: The input value.
Unary plus operation.

`unequal(x, y)`

Parameters:

- x <*>: First value.

- y <*>: Second value.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<boolean>`: Returns true if x is not equal to y.
Test whether two values are unequal.

`Unit(value, unit)`

Parameters:

- `value <number>`: The numeric value.

- `unit <string>`: The unit string or Unit object.

Returns: `<Unit>`: A new Unit instance.
Unit constructor.

`unit(value, unit)`

Parameters:

- `value <number>`: The numeric value.

- `unit <string>`: The unit string or Unit object.

Returns: `<Unit>`: A new Unit instance.
Create a unit.

`E`

Type: `<Unknown>`
The base of natural logarithms, e.

`PI`

Type: `<Unknown>`
The mathematical constant pi.

`usolve(U, b)`

Parameters:

- `U <Matrix>`: An upper triangular matrix.

- `b <Matrix>`: A column vector.

Returns: `<Matrix>`: The solution vector x.
Solve a linear system A * x = b where A is an upper triangular matrix.

`usolveAll(U, b)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- `U <Matrix>`: An upper triangular matrix.

- `b <Matrix>`: A column vector.

Returns: `<Array>`: An array of solutions.
Find all solutions of a linear system A * x = b where A is an upper triangular matrix.

`vacuumImpedance`

Type: `<Unknown>`
Vacuum impedance, in ohms.

`variance(array, [normalization], [dim])`

Parameters:

- `array <Array>`: Input array.

- `normalization <string>`: Normalization mode.

- `dim <number>`: Dimension along which to compute.

Returns: `<number>`: The variance.
Compute the variance of a set of values.

`weakMixingAngle`

Type: `<Unknown>`
Weak mixing angle.

`wienDisplacement`

Type: `<Unknown>`
Wien displacement constant, in m·K.

`xgcd(a, b)`

Parameters:

- `a <number>`: An integer.

- `b <number>`: An integer.

Returns: `<Array>`: An array [gcd, x, y] satisfying gcd = a*x + b*y.
Extended greatest common divisor for integers.

`xor(x, y)`

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Parameters:

- x <boolean>: First value.

- y <boolean>: Second value.

Returns: <boolean>: The logical XOR of x and y.
Logical XOR operation.

`ArgumentsError(fn, count, min, [max])`

Parameters:

- fn <string>: Function name.

- count <number>: Actual argument count.

- min <number>: Minimum required arguments.

- max <number>: Maximum allowed arguments.

Returns: <ArgumentsError>: A new ArgumentsError instance.
Error thrown when an incorrect number of arguments is passed.

`DimensionError(actual, expected, [relation])`

Parameters:

- actual <number>: Actual dimension.

- expected <number>: Expected dimension.

- relation <string>: Relation between dimensions.

Returns: <DimensionError>: A new DimensionError instance.
Error thrown when matrix dimensions mismatch.

`IndexError(index, min, max)`

Parameters:

- index <number>: The invalid index.

- min <number>: Minimum allowed index.

- max <number>: Maximum allowed index.

Returns: <IndexError>: A new IndexError instance.
Error thrown when an index is out of range.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

8.31 parallel

`getProcessorsNum()`

Returns: `<number>`: Number of processors.
Retrieves the number of logical processors available.

`workerFunction()`

Returns: `<string>`: The worker script as a string.
Generates the worker's internal script.

`init(num_workers)`

Parameters:

- num_workers `<number>`: Number of workers to initialize.

Initializes the worker pool with the specified number of workers.

`assignTasksToWorkers()`

Assigns tasks from the queue to available workers.

`run(context, [setup_function], args, work_function_str)`

Parameters:

- context `<Object>`: Context variables to assign in the worker.

- setup_function `<function>`: Optional setup function to execute before work_function.

- args `<Array>`: Arguments to pass to the work_function.

- work_function_str `<function>`: The work function to execute.

Returns: `<Promise>`: - Resolves with the result of the work_function.
Enqueues a task to be executed by the worker pool.

`parfor(start, end, [step], [num_workers], [context], [setup_function], work_function)`

Parameters:

- start `<number>`: The initial value of the loop counter.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- end `<number>`: The terminating value of the loop counter.

- step `<number>`: The amount by which to increment the loop counter each iteration.

- num_workers `<number>`: The number of workers to use.

- context `<Object>`: Optional context to pass to the work_function.

- setup_function `<function>`: Optional setup function to execute before work_function.

- work_function `<function>`: The function to execute on each iteration.

Returns: `<Promise.<Array>>`: - A promise that resolves to an array of results.
Executes a parallel for loop by dividing the iteration range among workers.

`terminate()`

Terminates all workers and resets the worker pool.

8.32 mat

`new(A, rows, cols)`

Parameters:

- A `<Array>`: The matrix data.

- rows `<number>`: Number of rows.

- cols `<number>`: Number of columns.

Returns: `<PRDC_JSLAB_MATRIX>`: A new matrix instance.
Creates a new matrix.

`fill(v, rows, [cols])`

Parameters:

- v `<number>`: The value to fill the matrix with.

- rows `<number>`: Number of rows.

- cols `<number>`: Number of columns.

Returns: `<PRDC_JSLAB_MATRIX>`: The filled matrix.
Creates a matrix filled with a specific value.

`NaNs(rows, [cols])`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- rows <number>: Number of rows.

- cols <number>: Number of columns.

Returns: <PRDC_JSLAB_MATRIX>: The NaN-filled matrix.
Creates a matrix filled with NaN values.

`ones(rows, [cols])`

Parameters:

- rows <number>: Number of rows.

- cols <number>: Number of columns.

Returns: <PRDC_JSLAB_MATRIX>: The ones-filled matrix.
Creates a matrix filled with ones.

`zeros(rows, [cols])`

Parameters:

- rows <number>: Number of rows.

- cols <number>: Number of columns.

Returns: <PRDC_JSLAB_MATRIX>: The zeros-filled matrix.
Creates a matrix filled with zeros.

`diag(A)`

Parameters:

- A <Array>: The array to create the diagonal matrix from.

Returns: <PRDC_JSLAB_MATRIX>: The diagonal matrix.
Creates a diagonal matrix from an array.

`eye(size)`

Parameters:

- size <number>: The size of the identity matrix.

Returns: <PRDC_JSLAB_MATRIX>: The identity matrix.
Creates an identity matrix of a given size.

`concatRow(args)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- args <PRDC_JSLAB_MATRIX>: Matrices to concatenate.

Returns: <PRDC_JSLAB_MATRIX>: The concatenated matrix.
Concatenates multiple matrices vertically (row-wise).

```
concatCol(args)
```

Parameters:

- args <PRDC_JSLAB_MATRIX>: Matrices to concatenate.

Returns: <PRDC_JSLAB_MATRIX>: The concatenated matrix.
Concatenates multiple matrices horizontally (column-wise).

```
isMatrix(A)
```

Parameters:

- A <Object>: The object to check.

Returns: <boolean>: True if A is a matrix, else false.
Checks if the provided object is a matrix.

8.33   vec

```
new(x, y, z)
```

Parameters:

- x <number>: The x-component of the vector.

- y <number>: The y-component of the vector.

- z <number>: The z-component of the vector.

Returns: <PRDC_JSLAB_VECTOR>: A new vector instance.
Creates a new vector with specified x, y, z components.

```
polar(length, radian)
```

Parameters:

- length <number>: The length of the vector.

- radian <number>: The angle in radians.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<PRDC_JSLAB_VECTOR>`: The resulting vector.
Creates a vector from polar coordinates.

`spherical(length, azimuth, elevation)`

Parameters:

- `length <number>`: The length (magnitude) of the vector.

- `azimuth <number>`: The azimuth angle in radians (angle from the X-axis in the XY plane).

- `elevation <number>`: The elevation angle in radians (angle from the XY plane towards Z).

Returns: `<PRDC_JSLAB_VECTOR>`: The resulting vector.
Creates a vector from spherical coordinates.

8.34 sym

`load()`

Returns: `<Promise.<void>>`: A promise that resolves when the libraries are loaded.
Loads the symbolic math libraries (SymPy and NumPy) using Pyodide. Initializes the Python environment for symbolic computations.

`getSymbolName(symbol)`

Parameters:

- `symbol <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>`: The symbolic variable or its name.

Returns: `<string>`: The name of the symbolic variable.
Retrieves the name of a symbolic variable.

`checkLoaded()`

Checks if the symbolic libraries are loaded. Throws an error if not.

`eval(code)`

Parameters:

- `code <string>`: The Python code to evaluate.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: `<*>`: The result of the evaluated code.
Evaluates a Python code string within the symbolic math environment.

`sym(name)`

Parameters:

- `name <string>`: The name of the symbolic variable.

Returns: `<PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>`: The created symbolic variable.
Creates a single symbolic variable.

`syms(names)`

Parameters:

- `names <Array.<string>>`: An array of names for the symbolic variables.

Returns: `<Array.<PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>>`: An array of created symbolic variables.
Creates multiple symbolic variables.

`mat(expr)`

Parameters:

- `expr <Array.<Array.<(PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string|number)>>>`: The nested array representing the matrix.

Returns: `<PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>`: The symbolic matrix.
Creates a symbolic matrix from a nested array expression.

`mul(args)`

Parameters:

- `args <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>`: The symbolic expressions to multiply.

Returns: `<PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>`: The resulting symbolic expression after multiplication.
Multiplies multiple symbolic expressions.

`div(args)`

Parameters:

- `args <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>`: The symbolic expressions to divide.

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The resulting symbolic expression after division.
Divides multiple symbolic expressions.

`plus(args)`

Parameters:

- args <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The symbolic expressions to add.

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The resulting symbolic expression after addition.
Adds multiple symbolic expressions.

`minus(args)`

Parameters:

- args <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The symbolic expressions to subtract.

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The resulting symbolic expression after subtraction.
Subtracts multiple symbolic expressions.

`pow(expr, n)`

Parameters:

- expr <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The base expression.

- n <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The exponent.

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The resulting symbolic expression after exponentiation.
Raises a symbolic expression to a power.

`transp(expr)`

Parameters:

- expr <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The matrix expression to transpose.

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The transposed matrix expression.
Transposes a symbolic matrix expression.

`inv(expr)`

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- expr <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The matrix expression to invert.

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The inverse of the matrix expression.
Computes the inverse of a symbolic matrix expression.

```
det(expr)
```

Parameters:

- expr <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The matrix expression whose determinant is to be computed.

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The determinant of the matrix expression.
Computes the determinant of a symbolic matrix expression.

```
diff(expr, x, [n])
```

Parameters:

- expr <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The expression to differentiate.

- x <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The variable with respect to which differentiation is performed.

- n <number>: The order of differentiation.

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The differentiated expression.
Differentiates a symbolic expression with respect to a variable.

```
intg(expr, x, lims)
```

Parameters:

- expr <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The expression to integrate.

- x <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The variable with respect to which integration is performed.

- lims <Array.<(PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL|string)>>: The limits of integration as [lower, upper]. If undefined, indefinite integration is performed.

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The integrated expression.
Integrates a symbolic expression with respect to a variable.

```
subs(expr, x, val)
```

Parameters:

web: pr-dc.com
email: info@pr-dc.com
github: github.com/PR-DC

JSLAB v1.0.0
DOCUMENTATION

- expr <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The expression in which to substitute.

- x <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The variable to substitute.

- val <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The value or array of values to substitute.

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The expression after substitution.
Substitutes a value into a symbolic expression.

`simplify(expr)`

Parameters:

- expr <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The expression to simplify.

Returns: <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The simplified expression.
Simplifies a symbolic expression.

`showLatex(expr)`

Parameters:

- expr <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The expression to display in LaTeX format.

Displays the LaTeX representation of a symbolic expression.

`displatex(expr)`

Parameters:

- expr <PRDC_JSLAB_SYMBOLIC_MATH_SYMBOL>: The expression to convert to a LaTeX string.

Displays the LaTeX string of a symbolic expression.

`clear()`

Clears all symbolic variables and resets the symbolic math environment.