Example usage

You get a global namespace of definitions, into which you can insert things using the `insert` function:

$$x_0 := 4$$

$$y_0 := 12$$

You can then evaluate expressions with these names:

$$3 + x_0 \cdot y_0 = 51$$

You can also define functions:

$$k(x) := \sin(x)$$

$$k(\pi) = 0$$

Real numbers aren't the only things supported! When possible, the evaluation engine tries to keep things rational:

$$\frac{1}{4} + \frac{1}{234} = \frac{119}{468}$$

It also supports complex numbers and decimals (decimals which are too long get turned into floating point numbers, but otherwise are parsed into rationals):

$$1.3 + i = \frac{13}{10} + i$$

$$\overline{3i} = -3i \qquad |3 + 4i| = 5 \qquad \exp(23i) = -0.53283 - 0.84622i$$

Another builtin datatype is matrices (implicitly vectors are matrices of the appropriate size):

$$g(z) := \begin{pmatrix} z \\ \frac{1}{2}z \end{pmatrix} \qquad g(3+i) = \begin{pmatrix} 3+i \\ \frac{3}{2} + \frac{1}{2}i \end{pmatrix} \qquad \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix} g(12) = \begin{pmatrix} 24 \\ 78 \end{pmatrix}$$

$$g(5)^T = \begin{pmatrix} 5 & \frac{5}{2} \end{pmatrix} \qquad \left(g(3+i)^* \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix} \right)^T = \begin{pmatrix} 9 - 3i \\ \frac{27}{2} - \frac{9}{2}i \end{pmatrix} \qquad \det\left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \right) = -2$$

The engine works fine with your builtin functions/macros: (if you're reading the pdf, we'll have just defined an inner product function `ip`):

$$\left\langle \begin{pmatrix} 1 \\ i \end{pmatrix}, \begin{pmatrix} 2 \\ -i \end{pmatrix} \right\rangle = 1$$

The final datatype that's built-in is callable objects. Callable objects are first-class citizens here:

$$f_0(z) := z^2 \qquad f_1(z) := \frac{1}{z} \qquad f_2(z) := iz$$

$$(f_0 + 2)(2) = 6 \qquad (f_0 \cdot f_1)(5) = 5 \qquad \left(\frac{f_1}{f_2} + f_0 \right)(3) = 9 - \frac{1}{9}i$$

You can also define closures. The syntax is as follows:

$$T := x \mapsto 3x + \begin{pmatrix}1\\2\end{pmatrix} \qquad T\left(\begin{pmatrix}2\\4\end{pmatrix}\right) = \begin{pmatrix}7\\14\end{pmatrix}$$

And these can capture variables:

$$T_0(x) := z \mapsto x + z \qquad T_0(2)(6) = 8$$

For the curious, you can now do (some weak form of) lambda calculus:

$$\text{True} := y \mapsto n \mapsto y \qquad \text{False} := y \mapsto n \mapsto n \qquad \text{If} := c \mapsto y \mapsto n \mapsto c(y)(n)$$

$$\text{If}\ (\text{True})(3)(4) = 3$$

Finally, there are three more functions supplied by the API:

- `debug` which will dump the current context. It's hard to parse, so you can ignore it and instead report bugs on the repo.
- `floateval` which will evaluate as usual but turn any rationals into floating point numbers.
- `floatexpr` which lets you evaluate things and get a float directly. Since this library is built on top of `context` expressions, then trying to use `floateval` or `evaluate` will yield a context expression.

Why do we care about not getting a context expression? Because we can then use this to do graphing: