A New Type System for Automatic Computing with Operator Objects

Thorsten Dahmen

University of Konstanz, Universittsstrae 10, 78457 Konstanz, Germany

Floating point numbers are the standard representation of approximations of real numbers in computers. For the last decade, the idea of introducing classes representing discrete approximations of real-valued functions $\mathbb{L}^2(\mathbb{R})$ and linear operators $\mathbb{T}(R_1, R_2) : R_1 \to R_2$ with $R_1, R_2 \in \{\mathbb{R}, \mathbb{L}^2(\mathbb{R})\}$, embedded into an object-oriented framework, led to a remarkable advancement in first-solve-then-discretize numerical algorithms for general boundary value problems, promoted by the Matlab package "Chebfun" [2].

However, *non-linear* operators $\mathbf{P} \in \mathbb{O}(Y)$ refer to non-linear mappings $X_1 \times \cdots \times X_n \to Y : \boldsymbol{x} \mapsto \mathrm{f}(\boldsymbol{x})$, with $X_{1,\ldots,n}, Y \in \{\mathbb{R}, \mathbb{L}^2(\mathbb{R})\}$ for which there is no approximate discretization. Automatically computed Frchet derivatives $\mathrm{d}\mathrm{f}/\mathrm{d}x \in \mathbb{T}$ may be used to obtain an affine discrete approximation $\boldsymbol{x} \mapsto \mathrm{d}\mathrm{f}/\mathrm{d}x(\boldsymbol{x_0})(\boldsymbol{x} - \boldsymbol{x_0}) + \mathrm{f}(\boldsymbol{x_0})$ in the neighborhood of an operating point $\boldsymbol{x_0}$ with $\mathrm{d}\mathrm{f}/\mathrm{d}x(\boldsymbol{x_0}) \in \mathbb{T}(R_1, R_2)$ and $\mathrm{f}(\boldsymbol{x_0}) \in R_2$. In computer programs, it seems natural to represent nonlinear operators as (anonymous) functions. E.g., in Chebfun `P = chebop(@(x,u,p) diff(u,2)+x.*u.*p)` constructs an object representing the nonlinear operator $\mathbb{L}^2(\mathbb{R}) \times \mathbb{R} \to \mathbb{L}^2(\mathbb{R}) : \ (u, p) \mapsto u'' + \mathrm{x}up$ on the interval [-1, 1]. However, `P` does not know about its arguments since in the anonymous function `u` and `p` are only placeholders to define the mapping f.

We introduce the types `Lfun{V<:Bool,Int,Float}` for $\mathbb{L}^2(V)$ `with` $V \in \{\mathbb{B}, \mathbb{Z}, \mathbb{R}\}$, `Linop{R1,R2}` for $\mathbb{T}(R_1, R_2)$ and `Affop{R1,R2} for` $\mathbb{A}(R_1, R_2)$ and use the Julia Programming Language [1] since for our approach many requirements concerning the programming language are not met by Matlab. `Lfun{Float}` and `Linop{R1,R2}` originate from the Julia package "Approxfun" [3].

Furthermore, we introduce the type `Optor{Y}` to represent non-linear operators where each $X_{1,\ldots,n}, Y \in \{V, \mathbb{L}^2(V), \mathbb{A}(R_1, R_2)\}$. The type `Optor{Y}` represents non-linear operators exactly since besides the field for the function defining the mapping f, an object of type `Optor{Y}` manages a tuple of references to its arguments, which are themselves of type `Optor{Y}`. The arguments refer to those leaves of the evaluation tree of the operator that are variables alias seeds. The evaluation tree may also have constants as leaves, which, in contrast to variables, are of type `Y` only.

A value $\mathrm{x} \in V$ can always be represented as a non-varying function $\mathbb{L}^2(V)$ and a constant $\mathrm{x} \in Y$ can always be represented by a variable $\mathbb{O}(Y)$, which is an operator with an empty argument list. Therefore, in addition to the standard promotion rules $\mathbb{B} \prec \mathbb{Z} \prec \mathbb{R}$, which apply for example when adding an integer and a float, we enable $V \prec \mathbb{L}(V)$, $Y \prec \mathbb{O}(Y)$, and promotion from operators to operators of greater type parameter: $\mathbb{O}(\mathbb{B}) \prec \mathbb{O}(\mathbb{Z}) \prec \mathbb{O}(\mathbb{R})$, $\mathbb{O}(V) \prec \mathbb{O}(\mathbb{L}^2(V))$ and $\mathbb{O}(\mathbb{L}^2(\mathbb{B})) \prec \mathbb{O}(\mathbb{L}^2(\mathbb{Z})) \prec \mathbb{O}(\mathbb{L}^2(\mathbb{R}))$.

since overloaded methods, in this case of the .^-function, ensure that arithmetic rules are applied automatically.

Like in Chebfun, automatic Frchet differentiation is enabled by overloading functions for `Optor{Affop}` objects according to the chain rule. Using our new type system simplifies the implementation of automatic Frchet differentiation and linearity detection, compared to that in Chebfun. Linearity of each variable is automatically detected since in this case, $\mathrm{d}\mathrm{f}/\mathrm{d}x$ is independent of the corresponding variable.

The new type system heavily benefits from many of Julia's typical features such as parametric types, multiple dispatch, promotion and metaprogramming. We believe that this concept promotes our vision to implement a fully automatic first-solve-then-discretize solver that can handle hybrid optimal control problems, boundary value problems, NLP problems and equation systems in a unified manner.

References

[1] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Julia: A Fast Dynamic Language for Technical Computing. *arXiv:1209.5145 [cs]*, September 2012. 00066.

[2] T. A Driscoll, N. Hale, and L. N. Trefethen. *Chebfun Guide*. Pafnuty Publications, 2014. 00008.

[3] Sheehan Olver and Alex Townsend. A practical framework for infinite-dimensional linear algebra. In *Proceedings of the 1st First Workshop for High Performance Technical Computing in Dynamic Languages*, pages 57–62. IEEE Press, 2014. 00001.