Verry: an open-source package for verified computation written in Python 3

Ryoga Iwanami

Waseda University

September 23, 2025 @Oldenburg

Outline

Introduction: What is Verry

Verry is a verified computation library written in Python 3.

Current features include:

- Affine arithmetic,
- Automatic differentiation,
- Interval arithmetic,
- ODE solver,
- Quadrature, etc.

Repo: `https://github.com/python-verry/verry`
Docs: `https://python-verry.github.io/verry/`

Introduction: Getting started

You can install Verry via PyPI: `pip install verry`.

Example (interval arithmetic)

```
from verry import FloatInterval as FI

print(sum(FI("0.1") for _ in range(10)))
# output: [inf=0.99999, sup=1.00001]
```

Remark

- Verry requires Python 3.13, and sometimes this is not pre-installed.
- Rounding errors are controlled by C++ extensions. However, you do not need to build C++; pre-built binaries are available in most cases.

Introduction: Solving an IVP of ODEs

The next example is solving an initial value problem (IVP) of ODEs:

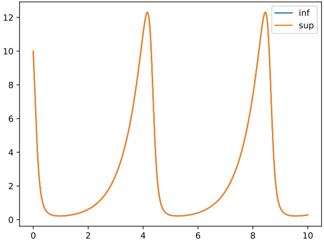
$$\frac{\mathrm{d}x}{\mathrm{d}t} = 1.5x + xy, \quad \frac{\mathrm{d}y}{\mathrm{d}t} = -3y + xy \quad \text{(Lotka–Volterra equations).}$$

```
from verry import FloatInterval as FI
from verry.integrate import C0Solver, doubleton, eilo

def fun(t, x, y):
    return (1.5 * x - x * y, -3 * y + x * y)

solver = C0Solver(integrator=eilo, tracker=doubleton)
r = solver.solve(fun, t0=FI(0), y0=[FI(10), FI(5)], t_bound=FI(10))
assert r.status == "SUCCESS"
print(r.content.y[0])  # output: [inf=0.286776, sup=0.287650]
```

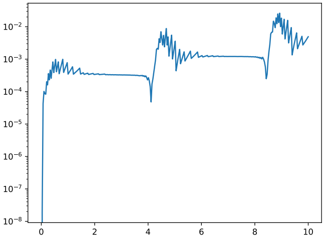
Introduction: Solving an IVP of ODEs




```
import matplotlib.pyplot as plt
import numpy as np

ts = np.linspace(0, 10, 300)
xs = [r.content.sol(t)[0] for t in ts]
plt.plot(ts, [x.inf for x in xs], label="inf")
plt.plot(ts, [x.sup for x in xs], label="sup")
plt.legend()
plt.show()

plt.clf()
plt.semilogy(
    ts, [x.diam() / x.mid() for x in xs]
)
plt.show()
```



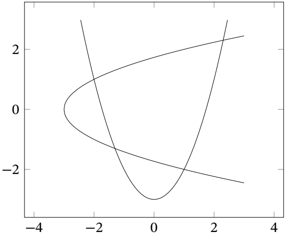
Introduction: Solving nonlinear equations

The existence (and uniqueness) of solutions to nonlinear equations can be verified using `allroot`:

```
from verry.linalg import FloatIntervalMatrix as FIM

def fun(x, y):
    return (x**2 - y - 3, -x + y**2 - 3)

dom = FIM(inf=[-3, -3], sup=[3, 3])
r = allroot(fun, dom, unique=True)
print(len(r.unique))  # output: 4
```



Remark

One may solve a boundary value problem of ODEs using `allroot` and `C1Solver`.

Introduction: Solver options

The ODE solver `C0Solver` has two options: *integrator* and *tracker*. The basic usage is like this:

```
# 1. define a solver.
solver = C0Solver(integrator, tracker)

# 2. apply the solver to the IVP.
result = solver.solve(fun, t0, y0, t_bound)
```

These options correspond to the algorithms consisting the solver, and their settings significantly impact accuracy and computational time.

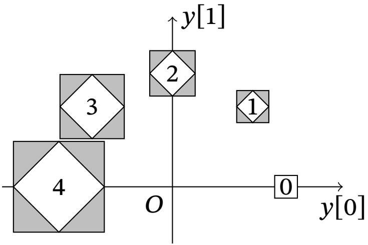
Remark

User-defined algorithms can also be passed as options. The requirements for the implementation are documented.

Introduction: Wrapping effect

Let $\Phi(t, t_0, y_0)$ be a solution of the IVP:

$$\begin{cases} \mathrm{d}y/\mathrm{d}t = f(t, y) & \text{if } t \neq t_0, \\ y = y_0 & \text{if } t = t_0. \end{cases}$$



We can find $[y_k]$, an enclosure of $\Phi(t_k, t_0, y_0)$, by solving the following problem for each $k = 0, 1, 2, \ldots$.

$$\text{Find } [y_{k+1}] \text{ s.t. } [y_{k+1}] \supseteq \{\Phi(t_{k+1}, t_k, y) \mid y \in [y_k]\}.$$

Problem (wrapping effect)

Usually the diameter of $[y_k]$ rapidly increases.

Introduction: Integrator and tracker

The common way to reduce W.E. is appending the extra step:

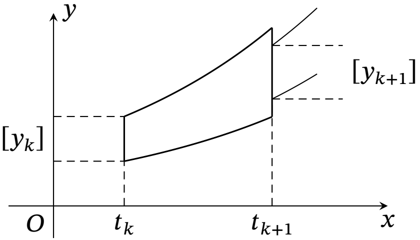
I. Find $[p_k(t)]$ s.t. $\forall t \in (t_k, t_{k+1}), \forall y \in [y_k], \Phi(t, t_k, y) \in [p_k(t)]$,
II. Find $[y_{k+1}]$ s.t. $\forall y \in [y_0], \Phi(t_{k+1}, t_0, y) \in [y_{k+1}]$.

We refer to the implementation of step I as the *integrator* and the implementation of step II as the *tracker*.

Introduction: Integrator and tracker

The common way to reduce W.E. is appending the extra step:

- I Find $[p_k(t)]$ s.t. $\forall t \in (t_k, t_{k+1}), \forall y \in [y_k], \Phi(t, t_k, y) \in [p_k(t)]$,
- II Find $[y_{k+1}]$ s.t. $\forall y \in [y_0], \Phi(t_{k+1}, t_0, y) \in [y_{k+1}]$.



We refer to the implementation of step I as the *integrator* and the implementation of step II as the *tracker*.

Background: Delay differential equations

Delay differential equations (DDEs): Differential equations whose right-hand side contains values of the solution at a previous time.

This talk focuses on IVPs of DDEs with single constant delay, such as

$$\begin{cases} \dot{y}(t) = f(t, y(t), y(t-\tau)) & \text{if } t > t_0, \\ y(t) = y_0(t) & \text{if } t_0 - \tau < t \leq t_0, \end{cases} \quad \text{where } \tau > 0.$$

Example

Constant coefficient linear DDEs $\dot{y}(t) = Ay(t) + By(t-\tau)$

Wright's equation $\dot{y}(t) = -\alpha y(t-\tau)(1 + y(t))$

Mackey–Glass equations $\dot{y}(t) = \beta y(t-\tau)/(1 + y(t-\tau)^n) - \gamma y(t)$

Background: The relationship between ODEs and DDEs

Let $y_1(t)$ be a solution of the initial value problem of ODEs

$$\begin{cases} \dot{y}(t) = f(t, y(t), y_0(t-\tau)) & \text{if } t_0 < t < t_0 + \tau, \\ y(t) = y_0(t_0) & \text{if } t = t_0. \end{cases}$$

Then the solution of the following initial value problem of ODEs

$$\begin{cases} \dot{y}(t) = f(t, y(t), y_1(t-\tau)) & \text{if } t_0' < t < t_0' + \tau, \\ y(t) = y_1(t_0') & \text{if } t = t_0' \end{cases}$$

is a solution of DDEs in $[t_0', t_0' + \tau]$, where $t_0' = t_0 + \tau$.

$\rightsquigarrow$ We can find a solution of DDEs by solving ODEs step by step.

Question

There are a lot of methods for solving IVPs of ODEs.

Problem

Which method is suitable for solving IVPs of DDEs?

Experiment: Compared methods

We conducted experiments for IVPs of two DDEs. Compared integrators and trackers are listed below.

1. Integrator: constant enclosure[2] (eilo) / polynomial enclosure[4] (kashi)
2. Tracker: QR decomposition[5] (qr) / affine arithmetic[1] (affine)

Remark

- (eilo) & (qr) is equivalent to the routine implemented in AWA.
- (kashi) & (affine) is equivalent to the routine implemented in kv.

---

[2] P. Eijgenraam, *The solution of initial value problems using interval arithmetic: formulation and analysis of an algorithm*, Centrum Voor Wiskunde en Informatica, Amsterdam, 1981.

[4] M. Kashiwagi, *Power series arithmetic and its application to numerical validation*, in Proceedings of the 1995 Symposium on Nonlinear Theory and its Applications, Las Vegas, NV, 1995, pp. 251–254.

[5] R. J. Lohner, *Enclosing the Solutions of Ordinary Initial and Boundary Value Problem*, in Computerarithmetic, E. Kaucher, U. Kulisch, and Ch. Ullrich, eds., B. G. Teubner, Stuttgart, 1987, pp. 225–286.

[1] J. L. D. Comba and J. Stolfi, *Affine Arithmetic and its Applications to Computer Graphics*, in Proceedings of the VI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI ’93), Recife, PE, 1993, pp. 9–18.

Experiment: Test problems

We employed two equations for testing:

1. Wright's equation: $\dot{y}(t) = -2y(t)(1 + y(t-1))$ $(y(t) \in \mathbb{R})$,
2. Linear equations: $\dot{y}(t) = Ay(t) - y(t-1)$ $(y(t) \in \mathbb{R}^3)$.

Measured indices are computational time T and precision P, where

$$P := -\log_{10}\left(\max_{1\le i\le n} \frac{\mathrm{diam}[y_{k\,i}]}{|\mathrm{mid}[y_{k\,i}]|}\right) \quad (t_k = t_{\mathrm{bound}}).$$

A is generated[3] to hold $\lim_{t\to\infty} |y(t)| = 0$. Rest values are defined as follows.

	Wright's equation	linear equations
t_{bound}	20	5
$y_0(t)$	t	$(1, 1, 1)$

[3] I. Fukuda, Y. Kiri, W. Saito, and Y. Ueda, *Stability Criteria for the System of Delay Differential Equations and its Applications*, Osaka J. Math., 59 (2022), pp. 235–251, doi:10.18910/86342.

Result: Wright's equation

Note: $P = -\log_{10}\left(\max_{1\le i\le n} \frac{\mathrm{diam}[y_{k\,i}]}{|\mathrm{mid}[y_{k\,i}]|}\right)$

We omit so the comparison with Tracker since W.E. does not contribute significantly in one-dimensional systems.

k	10	20	30	40	50	60	70	80
(eilo)	5.89	8.02	8.06	8.08	8.06	8.02	8.00	7.98
(kashi)	7.57	8.14	8.17	8.20	8.17	8.12	8.10	8.07

Table: the relationship between k and P, where $k = \tau/(t_{k+1} - t_k)$ $(\in \mathbb{Z})$.

Observation

- (kashi) always produces better accuracy, especially for $k = 10$.
- Both methods show no further improvement in accuracy beyond $k = 40$.

Result: Linear equations

Note: $P = -\log_{10}\left(\max_{1\le i\le n} \frac{\operatorname{diam}[y_{k\,i}]}{|\operatorname{mid}[y_{k\,i}]|}\right)$

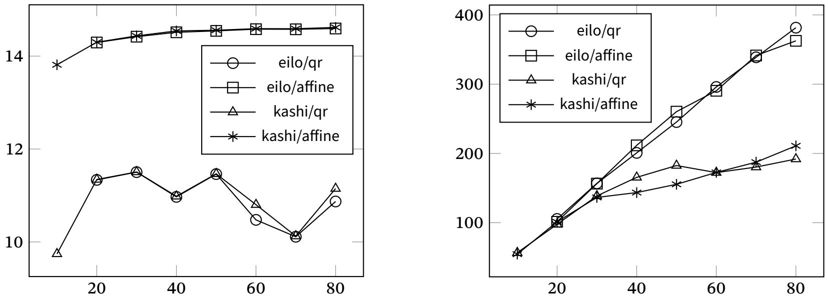


Figure: the left describes k–P relationship, and the right describes k–T relationship.

Observation

P depends on the choice of the tracker, and T depends on the choice of the integrator.

Conclusion

Conclusion

- The first result shows that integrators determine the accuracy if W.E. does not affect.
- The second result shows that the choice of trackers is highly important if the system is multi-dimensional.

Future work

- Reducing a computational time. We expect that this can be achieved by the use of more C/C++ pre-built binaries.
- Implementing solvers specialized in DDEs[6].

[6] R. Szczelina and P. Zgliczyński, *Algorithm for Rigorous Integration of Delay Differential Equations and the Computer-Assisted Proof of Periodic Orbits in the Mackey–Glass Equation*, Found. Comput. Math., 18 (2018), pp. 1299–1332, doi:10.1007/s10208-017-9369-5.

References

[1] J. L. D. Comba and J. Stolfi, *Affine Arithmetic and its Applications to Computer Graphics*, in Proceedings of the VI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI '93), Recife, PE, 1993, pp. 9–18.

[2] P. Eijgenraam, *The solution of initial value problems using interval arithmetic: formulation and analysis of an algorithm*, Centrum Voor Wiskunde en Informatica, Amsterdam, 1981.

[3] I. Fukuda, Y. Kiri, W. Saito, and Y. Ueda, *Stability Criteria for the System of Delay Differential Equations and its Applications*, Osaka J. Math., 59 (2022), pp. 235–251, doi:10.18910/86342.

[4] M. Kashiwagi, *Power series arithmetic and its application to numerical validation*, in Proceedings of the 1995 Symposium on Nonlinear Theory and its Applications, Las Vegas, NV, 1995, pp. 251–254.

[5] R. J. Lohner, *Enclosing the Solutions of Ordinary Initial and Boundary Value Problem*, in Computerarithmetic, E. Kaucher, U. Kulisch, and Ch. Ullrich, eds., B. G. Teubner, Stuttgart, 1987, pp. 225–286.

[6] R. Szczelina and P. Zgliczyński, *Algorithm for Rigorous Integration of Delay Differential Equations and the Computer-Assisted Proof of Periodic Orbits in the Mackey–Glass Equation*, Found. Comput. Math., 18 (2018), pp. 1299–1332, doi:10.1007/s10208-017-9369-5.

Appendix: Computational setup

We conducted experiments under the following environment:

CPU Intel Xeon Platinum 8380H (2.90 GHz) x 4

RAM 3 TB

OS Ubuntu 20.04.6