Sirpy Problem 3

SIRPY · Problem 3 — The Equation That Depends on Its Own Answer

An ongoing series. We're building a public library of problems SIRPY has solved — each post is one problem, the exact input we handed the solver, the exact output it returned, and an honest read of what happened. No slides. No hand-waving. Numbers you can check.

—

A different kind of hard

The first two problems were about singularities — where a solution blows up, and how the boundary data moves that point. This one is a different kind of difficulty, and it's one you can feel before you know any of the mathematics.

Here is the problem:

$$y'' = -y \int_0^1 y^2 dt, \quad y(0) = 1,\ y(1) = 0$$

Look at the right-hand side. The equation for y depends on $\int_0^1 y^2 dt$ — an integral of y over the *entire* interval. To evaluate the equation at any single point, you already need to know the whole solution, because the integral sees all of it at once.

That's the trap. Ordinary solvers march along building the solution point by point, using the equation to take the next step. But here the equation can't tell you the next step until the whole curve already exists. You need the answer to compute the answer.

This is a **nonlocal** problem — an integro-differential boundary value problem. It shows up in real places: quantum mechanics (the Gross–Pitaevskii and Hartree equations), population models where growth depends on total population, beam and membrane problems where local curvature depends on total tension. The nonlocal coupling is not exotic; it's just awkward for tools built to march.

What we handed the solver

Here is the entire program. Read the equation off the code:

```
import numpy as np, matplotlib.pyplot as plt
from sirpy import solve_bvp, plot_verification

r = solve_bvp(order=2, f_str="-M*y", x0=0, x1=1,
              left_bc={0: 1, 1: "gamma"}, right_bc={0: 0},
              nonlocal_terms={"M": {"integrand": "y**2",
                                    "from": 0, "to": 1}},
              iterations=20, verbose=True)

plot_verification(r, save="verify.png", show=True)
```

Notice how the nonlocal part is written. We give the integral a name — `M` — use it in the equation as `-M*y`, and then, in one `nonlocal_terms` block, we tell SIRPY what `M` actually is: the integral of y^2 from 0 to 1. That's it. **M *is* $\int_0^1 y^2 dt$.** We wrote the problem the way it reads on paper: name the quantity you don't yet know, then say what it is.

This is the same move we saw earlier, promoted one level. In Problem 2 we named an unknown *number* — the initial slope, "gamma" — and let SIRPY find it. Here we name an unknown *functional* — an integral of the solution — and let SIRPY find that. Same philosophy: you name what you don't know; the solver resolves it.

How it escapes the trap

The chicken-and-egg problem — needing the answer to compute the answer — is resolved by self-consistency: SIRPY refines M and the solution together until they stop moving and agree.

Its own report shows this happening:

```
Nonlocal BVP solved: M = 0.349689348703, gamma = -0.880625711864
nonlocal self-consistency loop converged in 35 outer sweeps
(last change 9.86e-11).
```

Thirty-five outer sweeps, converging until `M` changed by less than 10^{-10}. The final self-consistent value is $M = 0.34969$, with recovered initial slope $y'(0) = -0.88063$.

Is the answer real? Check it.

The same honest question as before. The equation is nonlinear and nonlocal; SIRPY hands back a curve and a number `M` that are consistent with *each other*. How do we know the curve actually solves the original equation, rather than just being internally self-consistent?

Because SIRPY returns the solution as a polynomial, we can substitute it straight back into the equation and measure the defect. The solver did exactly that:

```
verification by substitution - VERIFIED: satisfies the equation
to 4.16e-17 on [0, 1]. Checked by exact coefficient differentiation
at interior points, not at the interface nodes.
```

That is 4×10^{-17} — machine precision. SIRPY took its own solution, differentiated it exactly, formed $y'' + M \cdot y$, and confirmed it vanishes across the whole interval to the last bit a 64-bit machine can represent. And the boundary conditions were met: $y(0) = 1$ exactly, and the check at the far end came in at 4.29×10^{-14}.

- **Figure 1** is the solution SIRPY found on [0, 1] — starting at $y(0) = 1$, landing cleanly on $y(1) = 0$.
- **Figure 2** is the built-in `plot_verification` output: the left and right sides of the equation drawn together, agreeing everywhere. The overlap is the proof.

Why this one is worth your attention

Here is the honest claim, and it is not that other solvers *can't* do this. A capable numericist can solve a nonlocal problem — by hand-building the outer loop themselves: call a BVP solver, extract the solution, numerically integrate y^2 over the interval, feed the new value back, check convergence, repeat, and manage every way that loop can misbehave.

What SIRPY does is absorb that entire loop behind one `nonlocal_terms` block, do the integral exactly rather than by quadrature, and then verify the result by substitution to machine precision — a check the hand-rolled version gives you no easy way to perform. You wrote the equation as it reads. The solver handled the coupling, and proved its own answer.

A note on honesty

This equation is nonlinear, so we make the same disclosure as before: this is *a* self-consistent solution — the one the iteration reached from its starting point. We are not claiming it is the only one. Nonlocal nonlinear problems can admit several solutions, and SIRPY lets you search for others by changing the starting slope. What we are claiming is narrow and verified: this solution satisfies the equation to 4×10^{-17} and the boundary data to 4×10^{-14}. That is the whole claim, and it is checkable.

Next in the series

We'll keep following this thread — problems where the honest, verifiable answer is the hard part, and where seeing the solution as a *function* rather than a table is what makes certainty possible.

—

SIRPY is a differential-equation solver in testing, not yet released — a product of Luminary Quantum Institute, LLC (LQI), developed at the Mitra Institute of Education (MIE). Every line quoted above is verbatim from a single run of the solver; nothing is from memory. Have a problem you think is hard? We'd like to see it.

#SIRPY #MIE #LQI #NumericalAnalysis #DifferentialEquations #AppliedMathematics #IntegroDifferential #ScientificComputing

—

FIGURES — attach in this order: 1. Figure 1 — SIRPY's solution on [0, 1], meeting $y(0) = 1$ and $y(1) = 0$

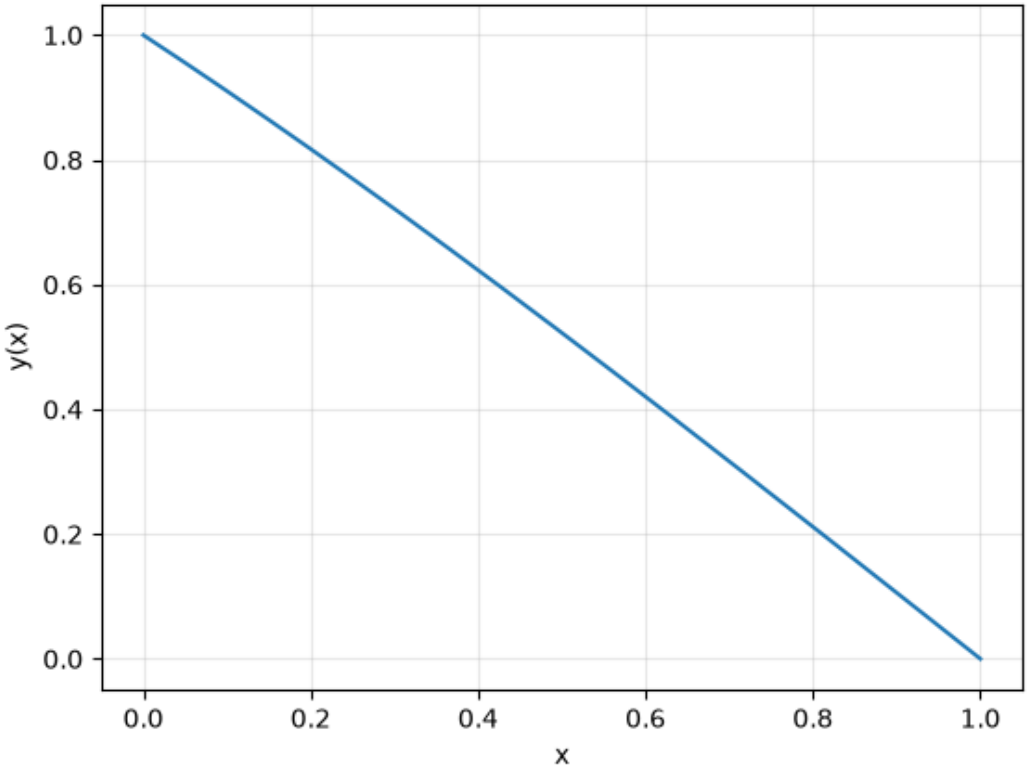

Figure 1: *Solution on* ([0,11])

2. Figure 2 — `plot_verification`: both sides of the equation, agreeing everywhere

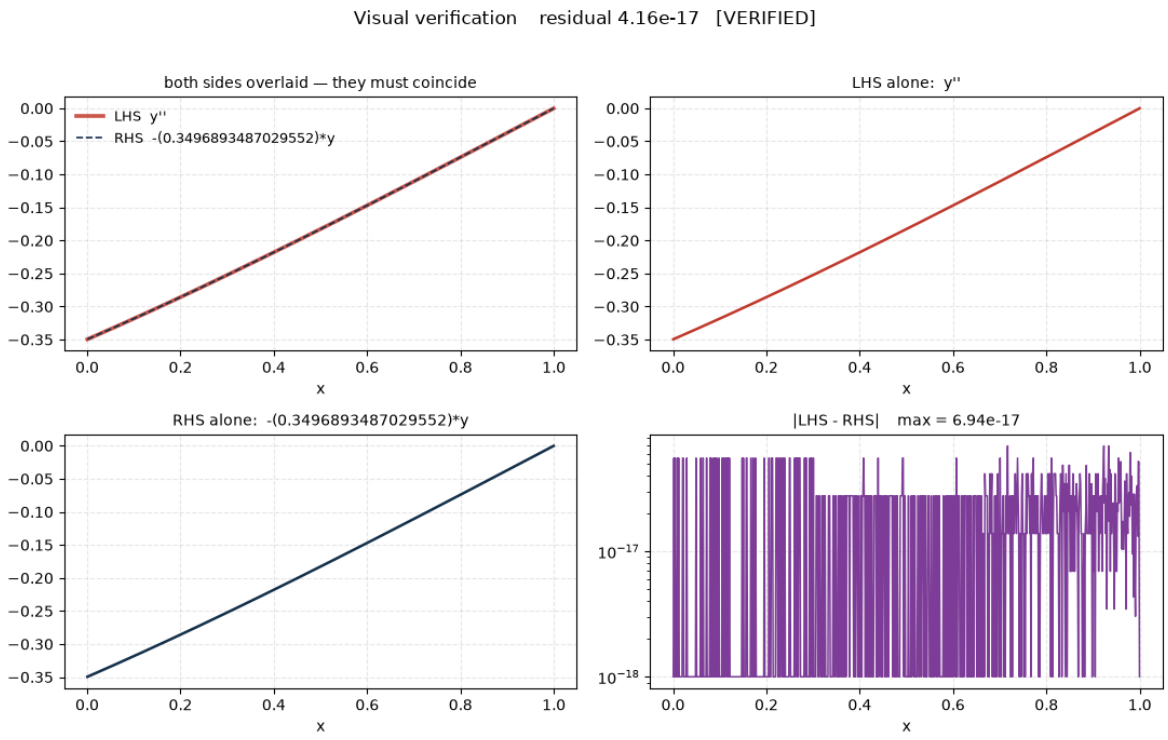

Figure 2: Verification plot produced by `plot_verification()`