Sirpy Problem 2

SIRPY Problems · 002

The Solution You'll Swear Is Fake (Until You Check It)

Part of an ongoing series documenting real problems solved with SIRPY before its public launch. Every paper shows the exact problem, the exact input, the solver's output, and what we learned from it. Nothing is hidden. Nothing is recreated later. Every number comes from a single run.

Last time

Our first problem looked harmless:

$$y'' = 2y^3, \qquad y(0) = -0.1, \qquad y(9) = -1.$$

Its exact solution,

$$y(x) = \frac{1}{x - 10},$$

was perfectly smooth on the interval ([0,9]), but contained a singularity sitting just one unit beyond the boundary at (x=10).

SIRPY not only recovered the exact solution to machine precision, it also identified that unseen singularity without ever being told it existed.

Today we change only one number.

This week's problem

The differential equation is identical.

The left boundary is identical.

Only the value at the right endpoint changes.

$$y'' = 2y^3, \qquad y(0) = -0.1, \qquad y(11) = 1.$$

Before reading further, stop for a moment.

Last week the singularity was located at (x=10).

Now the interval extends to (x=11).

What do you expect to happen?

Exactly what we handed SIRPY

```
from sirpy import solve_bvp, plot_verification

r = solve_bvp(
    order=2,
    f_str="2*y**3",
    x0=0,
    x1=11,
    left_bc={0: -0.1, 1: "gamma"},
    right_bc={0: 1},
    iterations=20,
    verbose=True,
)

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

That is the complete input.

No conversion to a first-order system.

No mesh to design.

No initial guess.

We specify the equation exactly as written, tell SIRPY that the initial slope is unknown by labeling it `"gamma"`, and let it recover the missing initial condition automatically.

The second line, `plot_verification`, is built into SIRPY.

It substitutes the returned solution directly back into the original differential equation and plots the two sides together so they can be compared visually.

The first surprise

The solver returned a perfectly smooth solution over the entire interval.

Then it reported something unexpected.

```
exact first-integral test - the solution's singularity is at
x = 11.9999, OUTSIDE [0,11] (margin 0.9999).
```

The singularity moved.

Last week it was at (x=10).

This week, changing only the boundary condition moved it to approximately (x=12).

The equation never changed.

Only the information we required the solution to satisfy changed.

That is the important mathematical point.

For nonlinear boundary value problems, the boundary conditions are not merely values attached to the solution—they determine which solution exists, and with it the location of features such as singularities.

The cliff from last week's problem never disappeared.

It simply moved.

But is the solution actually correct?

This is the question that matters.

Finding a smooth curve satisfying the boundary values is not enough.

The curve must satisfy the differential equation everywhere between those boundaries.

Because SIRPY returns its answer as a polynomial rather than a table of sampled values, the solution can be substituted directly back into the original equation.

The solver reported

```
verification by substitution - VERIFIED:
satisfies the equation to 8.31e-10 on [0,11]
and the boundary data to 0.00e+00.

Checked by exact coefficient differentiation
at interior points,
not at the interface nodes.
```

Read that carefully.

SIRPY differentiated its own polynomial exactly.

It substituted that derivative back into

$$y'' = 2y^3,$$

and compared the two sides throughout the interval.

The agreement was approximately

$$8.3 \times 10^{-10},$$

while the boundary conditions were satisfied exactly to machine precision.

Even more importantly, the comparison was made at interior points rather than at the interfaces where polynomial pieces meet.

Matching at interfaces would have been the easy test.

Matching away from them is the meaningful one.

Figure 1

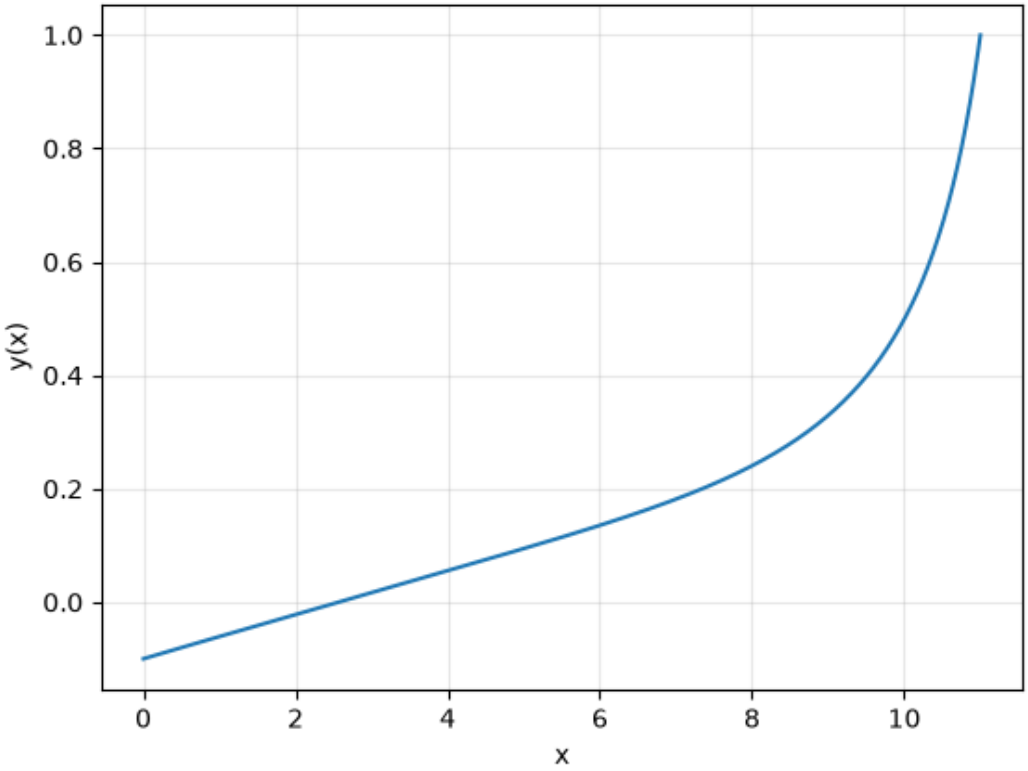

Figure 1: *Solution on* ([0,11])

Figure 2

The verification plot overlays the left-hand side,

$$y'',$$

and the right-hand side,

$$2y^3,$$

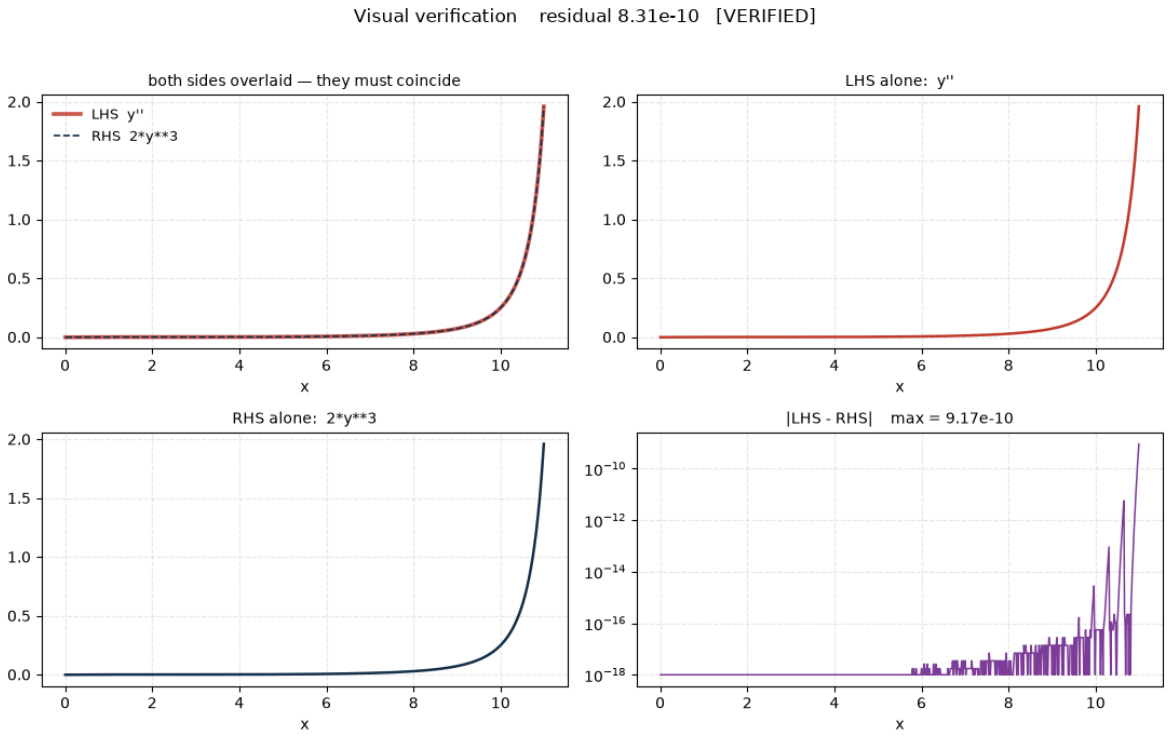

Figure 2: Verification plot produced by `plot_verification()`

throughout the interval.

The two curves are visually indistinguishable.

Rather than asking the user to trust the solver, SIRPY simply plots both sides of the original equation.

What SIRPY also reported

One design goal of SIRPY is that it should never advertise a more impressive accuracy than it has actually earned.

During this solve it printed

```
node matching (5.6e-17) measures interface agreement only;

the estimated truncation error of the series is ~5.3e-12.

The true accuracy is limited by the LARGER of the two.

Raise iterations to reduce it.
```

The interface mismatch is almost zero.

That sounds impressive.

SIRPY immediately explains why that number is **not** the one users should trust.

Instead, it reports the larger truncation estimate as the meaningful accuracy.

That correction was not requested.

The solver volunteered it.

One more detail

The equation is nonlinear.

That means several different solutions may satisfy the same boundary conditions.

SIRPY reports this explicitly.

```
this equation is nonlinear in y,

so several solutions may satisfy the same boundary data.

This is the branch reached from the starting slope;

pass gamma0=<value> to search for a different branch,
then verify() it.
```

Many solvers quietly return one solution.

SIRPY returns one verified solution and reminds the user that others may exist.

Those are different things.

What this problem teaches

- Changing the boundary data changed the location of the singularity even though the differential equation itself never changed.
- A solution represented as a polynomial can be substituted directly back into the original equation for verification.
- SIRPY verified the differential equation throughout the interval instead of checking only the boundary conditions.
- The solver distinguishes between interface agreement and actual solution accuracy rather than reporting whichever number looks better.
- Nonlinear boundary value problems may have multiple solution branches, and SIRPY says so explicitly.

Another problem is already waiting.

Not because it is more difficult.

Because it asks a different question.

Sometimes computing a solution is the easy part.

Understanding what that solution means is considerably harder.

SIRPY is an ordinary differential equation solver currently in testing. It is being developed by Luminary Quantum Institute, LLC (LQI), operating through the Mitra Institute of Education (MIE). Every value quoted above comes directly from a single execution of the solver. Nothing has been recreated or edited after the fact.