Integrating the Robertson Stiff Chemical-Kinetics System: Adaptive BDF Integration, Mass Conservation, and a Quantitative Demonstration of Stiffness

July 12, 2026

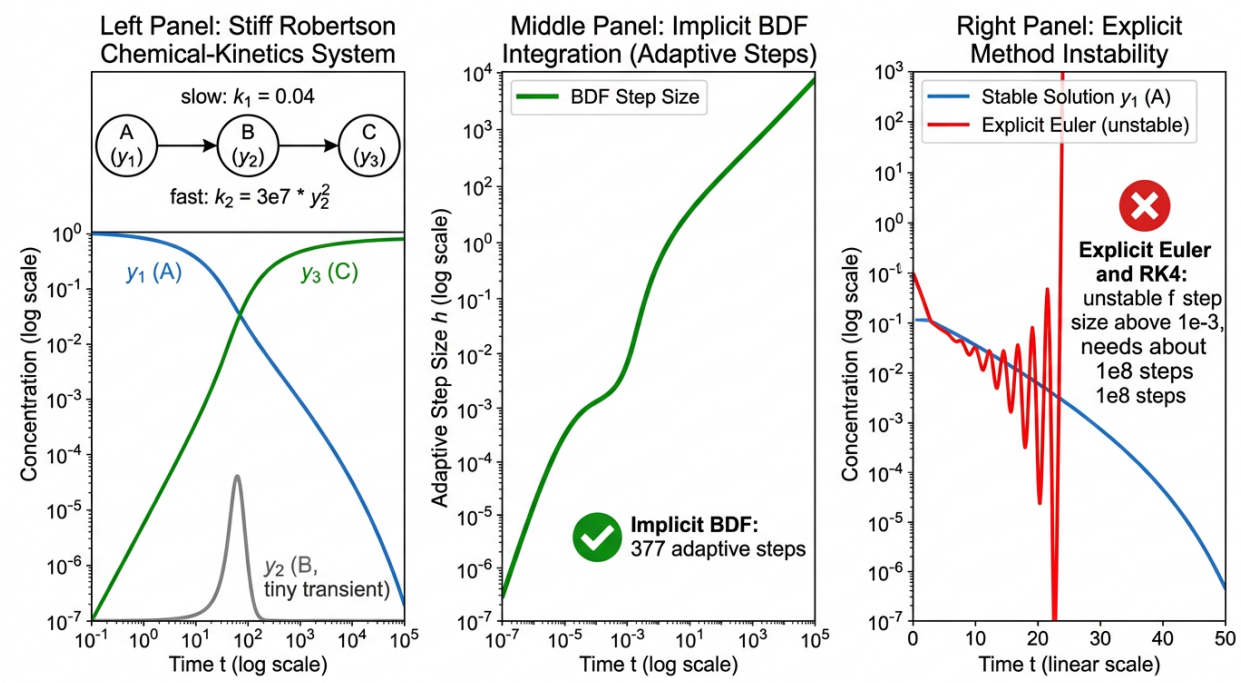

Figure 1: Graphical abstract. The Robertson system evolves three chemical species over eleven decades of time: y_1 decays from 1 to ≈ 0.018, y_3 grows from 0 to ≈ 0.982, and the reactive intermediate y_2 never exceeds $\approx 3.6 \times 10^{-5}$ (left). An implicit backward differentiation formula (BDF) solver with adaptive step-size control reaches $t = 10^5$ in only 377 accepted steps, its step size growing from $\sim 10^{-7}$ to $\sim 3.6 \times 10^3$ (centre). Explicit integrators (forward Euler, RK4) are stability-limited to $h \lesssim 10^{-3}$: for larger steps the solution blows up (right), and reaching $t = 10^5$ would require $\sim 10^8$ fixed steps.

Abstract

The Robertson problem—a three-species autocatalytic reaction network with rate constants spanning nine orders of magnitude—is the canonical benchmark for *stiff* systems of ordinary differential equations. We integrate it from $t = 0$ to $t = 10^5$ using an adaptive, variable-order implicit BDF method (`SciPy solve_ivp`, `method='BDF'`) with an analytic Jacobian, a relative tolerance of 10^{-6}, and per-component absolute tolerances atol $= [10^{-6}, 10^{-14}, 10^{-6}]$ chosen to resolve the disparate species magnitudes—in particular the trace intermediate y_2, whose peak value is $\approx 3.6 \times 10^{-5}$. The integration succeeds in 377 accepted internal steps (1068 right-hand-side evaluations, 13 Jacobian evaluations). We report the state at $t = 1$, 10^2, 10^4, and 10^5: the slow reactant y_1 takes the values **0.96646**, **0.61724**, **0.10730**, and **0.017866** respectively.

The linear conservation invariant $y_1 + y_2 + y_3 = 1$ holds to a maximum absolute deviation of 1.33×10^{-15}—machine precision. To quantify the stiffness we implement custom fixed-step forward-Euler and classical RK4 integrators and analyse the spectrum of the Jacobian along the solution. Although the system is *not* stiff at $t = 0$ (its eigenvalues are $\{-0.04, 0, 0\}$), stiffness emerges as the fast modes activate, the dominant eigenvalue reaching $|\lambda|_{\max} \approx 9.8 \times 10^3$ near $t = 10^5$. The empirically measured stable step sizes, $h^{\mathrm{FE}}_{\mathrm{stable}} \approx 7.7 \times 10^{-4}$ and $h^{\mathrm{RK4}}_{\mathrm{stable}} \approx 1.2 \times 10^{-3}$, match the eigenvalue prediction $h \leq C/|\lambda|_{\max}$ to within 12–21%. Reaching $t = 10^5$ with an explicit method would require ~ 3.5–4.9×10^8 steps (an estimated 2–3.4 CPU-hours), versus 377 BDF steps—a factor of $\sim 10^6$. We deliver the log-spaced solution trajectory and the full adaptive step-size history so that every result can be independently reproduced.

1 Introduction

The numerical integration of chemical-kinetics equations routinely confronts a pathology known as *stiffness*: the presence, within a single dynamical system, of physical processes whose characteristic time scales differ by many orders of magnitude. When a reaction network couples very fast relaxation modes to slowly evolving bulk concentrations, an explicit time integrator must resolve the fastest mode—not because that mode carries interesting dynamics on the time scale of interest, but because numerical *stability* forbids larger steps. The result is a catastrophic mismatch between the accuracy the user requires and the step size the method is permitted to take, rendering explicit methods effectively useless even long after the fast transients have died away. The phenomenon was first identified and named by Curtiss and Hirschfelder [1], who observed that certain differential equations arising in reaction kinetics are "stiff" in the sense that stability, rather than accuracy, dictates the step size. Its resolution—implicit methods with unconditional or near-unconditional stability—was placed on a rigorous footing by the stability theory of Dahlquist [2] and the practical backward-differentiation-formula (BDF) integrators of Gear [3, 4].

The system studied here was introduced by Robertson [5] as a compact model of an autocatalytic reaction and has since become the single most widely used benchmark for stiff ODE solvers, appearing in the classic method comparisons of Enright et al. [6], the definitive monograph of Hairer and Wanner [7], and virtually every modern IVP test set, including that of Mazzia et al. [8]. Its enduring appeal lies in its economy: only three species and three rate constants, yet rate constants (0.04, 10^4, and 3×10^7) spanning nine decades produce genuinely stiff behaviour, an almost-conserved trace intermediate that stresses absolute-tolerance handling, and a linear mass-conservation invariant that provides an independent check on solution quality. These same features make it an ideal vehicle for *demonstrating* stiffness pedagogically, which is the dual purpose of this report.

Stiff chemical kinetics is not merely a textbook curiosity. Detailed combustion mechanisms, atmospheric photochemistry, and biochemical reaction networks all generate stiff systems in which robust implicit integration is a computational bottleneck; specialised multiscale strategies continue to be developed to accelerate stiff-chemistry integration in reacting-flow simulations [9]. The methods we exercise here—adaptive variable-order BDF with a modified-Newton inner solve—are precisely those embodied in production solvers such as LSODE/VODE [10, 11], SUNDIALS [12], the MATLAB `ode15s` integrator [13], and the `BDF` option of SciPy's `solve_ivp` [14] that we employ.

This report has two objectives. First, we compute a high-quality reference solution of the Robertson problem over the full interval $[0, 10^5]$, extract the species concentrations at four target times, and verify mass conservation to machine precision. Second, and more instructively, we *quantify* the stiffness: we implement explicit forward-Euler and RK4 integrators from first principles [15, 16], measure the largest step size for which they remain stable, connect that limit to the

eigenvalue spectrum of the system's Jacobian, and extrapolate the prohibitive cost an explicit method would incur to reach $t = 10^5$. In doing so we correct a common misconception—that the Robertson problem is "most stiff at $t = 0$"—and show instead that the binding stability constraint arises only after the fast modes activate.

2 Mathematical Formulation

2.1 The Robertson reaction network

The Robertson mechanism describes three chemical species A, B, C with molar concentrations y_1, y_2, y_3 undergoing the reactions

$$A \xrightarrow{k_1} B, \qquad k_1 = 0.04, \tag{1}$$

$$B + B \xrightarrow{k_3} C + B, \qquad k_3 = 3 \times 10^7, \tag{2}$$

$$B + C \xrightarrow{k_2} A + C, \qquad k_2 = 10^4. \tag{3}$$

The first reaction is slow, the second (an autocatalytic self-reaction of B) is extremely fast, and the third is fast. Applying the law of mass action yields the initial value problem

$$\begin{aligned} y_1' &= -0.04\, y_1 + 10^4\, y_2\, y_3, \\ y_2' &= 0.04\, y_1 - 10^4\, y_2\, y_3 - 3 \times 10^7\, y_2^2, \\ y_3' &= 3 \times 10^7\, y_2^2, \end{aligned} \qquad \mathbf{y}(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{4}$$

integrated over $t \in [0, 10^5]$. Physically, A is slowly converted to the reactive intermediate B, which is immediately consumed—partly regenerating A, partly forming the terminal product C—so that B is held at a tiny quasi-steady concentration while A drains into C over five decades of time.

2.2 The conservation invariant

Summing the three equations in (4) gives $y_1' + y_2' + y_3' = 0$ identically, so the total concentration is a conserved quantity fixed by the initial data:

$$y_1(t) + y_2(t) + y_3(t) = y_1(0) + y_2(0) + y_3(0) = 1 \qquad \forall\, t \geq 0. \tag{5}$$

This linear first integral is not imposed on the solver; it emerges from the structure of the equations, and the numerical deviation $|y_1 + y_2 + y_3 - 1|$ therefore serves as an *independent* diagnostic of round-off and truncation error. In the terminology of Hairer and Wanner [7], the Robertson system sits at the boundary between a stiff ODE and a differential-algebraic equation (DAE): as $t \to \infty$ the differential equation for y_2 degenerates toward the algebraic constraint $0.04\, y_1 \approx 3 \times 10^7\, y_2^2$, a quasi-steady-state relation that is the microscopic origin of the stiffness.

2.3 Jacobian and local time scales

The stability behaviour of any integrator applied to (4) is governed by the Jacobian $J(\mathbf{y}) = \partial \mathbf{f} / \partial \mathbf{y}$,

$$J(\mathbf{y}) = \begin{bmatrix} -0.04 & 10^4\, y_3 & 10^4\, y_2 \\ 0.04 & -10^4\, y_3 - 6 \times 10^7\, y_2 & -10^4\, y_2 \\ 0 & 6 \times 10^7\, y_2 & 0 \end{bmatrix}, \tag{6}$$

which we supply to the implicit solver in closed form. The local relaxation time scales are the reciprocals of the moduli of the nonzero eigenvalues of J. At the initial condition $\mathbf{y}(0) = [1, 0, 0]^\top$ the Jacobian collapses to

$$J(\mathbf{y}(0)) = \begin{bmatrix} -0.04 & 0 & 0 \\ 0.04 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \text{spec} = \{-0.04,\, 0,\, 0\}, \tag{7}$$

so that at $t = 0$ the system is *not* stiff at all: the only active time scale is the slow one, $1/0.04 = 25$. Stiffness is a property that *develops* as y_2 grows away from zero and the $-6 \times 10^7\, y_2$ and $-10^4\, y_3$ terms come to dominate the spectrum. This observation, developed quantitatively in Section 4.4, is central to understanding why explicit integrators fail on this problem, and it corrects the intuitive but incorrect expectation that the difficulty is concentrated in the initial layer.

3 Methodology

3.1 Implicit BDF integration

We integrate (4) with the variable-order (orders 1–5), variable-step backward differentiation formula implemented in `scipy.integrate.solve_ivp` [14], which is a Python descendant of the classical VODE/LSODE family [10, 11]. A k-step BDF approximates the derivative at the new time level by a backward finite difference of the computed history and solves the resulting implicit algebraic system,

$$\sum_{j=0}^{k} \alpha_j\, \mathbf{y}_{n+1-j} = h\, \beta_0\, \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}), \tag{8}$$

for $\mathbf{y}_{n+1}$ at each step. Because BDF methods of order ≤ 6 possess large stability regions that enclose the entire negative real axis (they are $A(\alpha)$-stable, and A-stable for orders 1 and 2), the admissible step size is limited by *accuracy*, not stability—exactly the property a stiff problem demands [2, 7, 17]. The implicit system is solved by a modified Newton iteration; supplying the analytic Jacobian (6) makes each Newton solve exact to first order and minimises the number of Jacobian evaluations and LU factorisations.

Adaptive step-size and order selection are driven by a local error estimate $\mathbf{e}_n$ that the solver keeps below a user weighted norm; the step-size controller itself can be understood as a feedback control loop acting on this error signal [18]. Following the standard convention [7, 13], the per-component error weight combines a relative and an absolute tolerance,

$$\|\mathbf{e}_n\|_{\text{wrms}} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left(\frac{e_{n,i}}{\text{rtol}\, |y_{n,i}| + \text{atol}_i} \right)^2} \;\leq\; 1, \tag{9}$$

where $m = 3$. We set $\text{rtol} = 10^{-6}$ throughout.

3.2 Justification of the per-component absolute tolerances

The choice of atol is the single most consequential numerical decision for this problem, because the absolute tolerance sets the error floor whenever a component approaches zero and the relative term $\text{rtol}\, |y_i|$ becomes negligible. The three species occupy vastly different dynamic ranges: y_1 and y_3 are $\mathcal{O}(1)$ quantities, whereas the intermediate y_2 never exceeds $\approx 3.6 \times 10^{-5}$ and decays to $\approx 7 \times 10^{-8}$

by $t = 10^5$. A single scalar tolerance is therefore inappropriate. We adopt the per-component vector

$$\text{atol} = [\,10^{-6},\ 10^{-14},\ 10^{-6}\,]. \tag{10}$$

The values 10^{-6} on y_1 and y_3 request roughly six significant digits on the two $\mathcal{O}(1)$ species. The very tight 10^{-14} on y_2 is chosen so that the absolute floor lies about eight orders of magnitude below y_2's peak value; this guarantees that the intermediate is controlled by its *relative* error over essentially its entire lifetime, so that the $\mathcal{O}(10^{-5})$ concentration is resolved to full relative accuracy rather than being swamped by an absolute floor comparable to the value itself. Choosing atol_2 too loosely (e.g. 10^{-6}, larger than y_2 itself) would let the solver treat y_2 as numerical noise, corrupt the tightly coupled fast balance $0.04\,y_1 \approx 3 \times 10^7\,y_2^2$, and degrade the conservation identity. This tolerance strategy is exactly the one recommended for the Robertson problem in the standard references [7, 13].

3.3 Solution products

The solver is run with `dense_output=True`, yielding a continuous interpolant used to (i) evaluate the state at the four target times $t \in \{1, 10^2, 10^4, 10^5\}$; (ii) sample a 200-point trajectory logarithmically spaced over $t \in [10^{-6}, 10^5]$; and (iii) verify conservation. The adaptive step-size history is reconstructed from the solver's internal accepted step points $\{t_n\}$ as $h_n = t_{n+1} - t_n$. Both the trajectory and the step-size history are written to CSV so the integration can be re-run and checked against our numbers.

3.4 Explicit integrators and the stability experiment

To demonstrate stiffness we implement, from first principles in NumPy [19], two fixed-step explicit integrators: forward Euler (FE), $\mathbf{y}_{n+1} = \mathbf{y}_n + h\,\mathbf{f}(t_n, \mathbf{y}_n)$, and the classical four-stage Runge–Kutta method (RK4). Each integrator terminates early and flags divergence if $\|\mathbf{y}\|_\infty > 10$ or a non-finite value appears—symptoms of the exponential error growth characteristic of an unstable explicit scheme on a stiff problem.

Our analysis combines three complementary probes. *(i) Eigenvalue theory.* For a linearised mode with eigenvalue $\lambda < 0$, an explicit method is stable only when $h\lambda$ lies inside its stability region; on the negative real axis this gives $h \le C/|\lambda|_{\max}$, with the constant $C = 2$ for forward Euler (whose real-axis stability interval is $(-2, 0)$) and $C \approx 2.785$ for RK4. We evaluate the Jacobian (6) along the reference BDF trajectory and track $|\lambda|_{\max}(t)$, the modulus of its dominant eigenvalue, at $t = 0$, at the y_2 peak, over a short horizon $[0, 20]$, and globally over $[0, 10^5]$. *(ii) Empirical h_{stable}.* We search for the largest stable step directly, by integrating over $[0, 20]$ with a coarse descending scan followed by bisection between the smallest diverging and the largest converging step. *(iii) Blow-up demonstration.* Over $[0, 100]$ we integrate at a step just below and just above the horizon stability limit, recording $\|\mathbf{y}(t)\|_\infty$ to exhibit the transition from a bounded, physical solution to unbounded growth. Finally, we run SciPy's adaptive explicit solvers `RK45` and `DOP853` in a child process under a 30-second wall-clock guard, to confirm that even sophisticated error-controlled explicit codes cannot traverse $[0, 10^5]$. All computation used Python 3.12 with NumPy, SciPy $\ge$ 1.14 [14, 19] and Matplotlib for figures [20].

4 Results

4.1 Reference solution and target states

The BDF integration reached $t = 10^5$ successfully, reporting "*The solver successfully reached the end of the integration interval*," in 377 accepted internal steps with 1068 right-hand-side evaluations, 13 Jacobian evaluations, and 81 LU factorisations. Table 1 lists the species concentrations at the four requested output times, with the slow reactant y_1 highlighted in bold as requested.

Table 1: Robertson solution components at the four target times, from the adaptive BDF integration (rtol $= 10^{-6}$, atol $= [10^{-6}, 10^{-14}, 10^{-6}]$). The slow reactant y_1 is highlighted. Values are reported to twelve significant figures as delivered by the dense interpolant.

t	$\boldsymbol{y_1}$ (reactant A)	y_2 (intermediate B)	y_3 (product C)
10^0	$\mathbf{9.664594432359 \times 10^{-1}}$	$3.074621762662 \times 10^{-5}$	$3.350981054646 \times 10^{-2}$
10^2	$\mathbf{6.172351219993 \times 10^{-1}}$	$6.153597237021 \times 10^{-6}$	$3.827587244034 \times 10^{-1}$
10^4	$\mathbf{1.073006695372 \times 10^{-1}}$	$4.800179038577 \times 10^{-7}$	$8.926988504449 \times 10^{-1}$
10^5	$\mathbf{1.786599825820 \times 10^{-2}}$	$7.274783433414 \times 10^{-8}$	$9.821339289940 \times 10^{-1}$

Explicitly, the highlighted reactant values are

$$y_1(1) = 0.9664594432359, \qquad y_1(10^2) = 0.6172351219993,$$
$$y_1(10^4) = 0.1073006695372, \qquad y_1(10^5) = 0.0178659982582.$$

The physical picture is unambiguous: y_1 decays monotonically from 1 toward zero as it is consumed, y_3 rises monotonically toward 1 as the terminal product accumulates, and y_2 remains a trace intermediate throughout, peaking near 3.6×10^{-5} around $t \approx 10^{-2}$ (Fig. 2) and thereafter decaying slowly as the reservoir of A that feeds it is depleted.

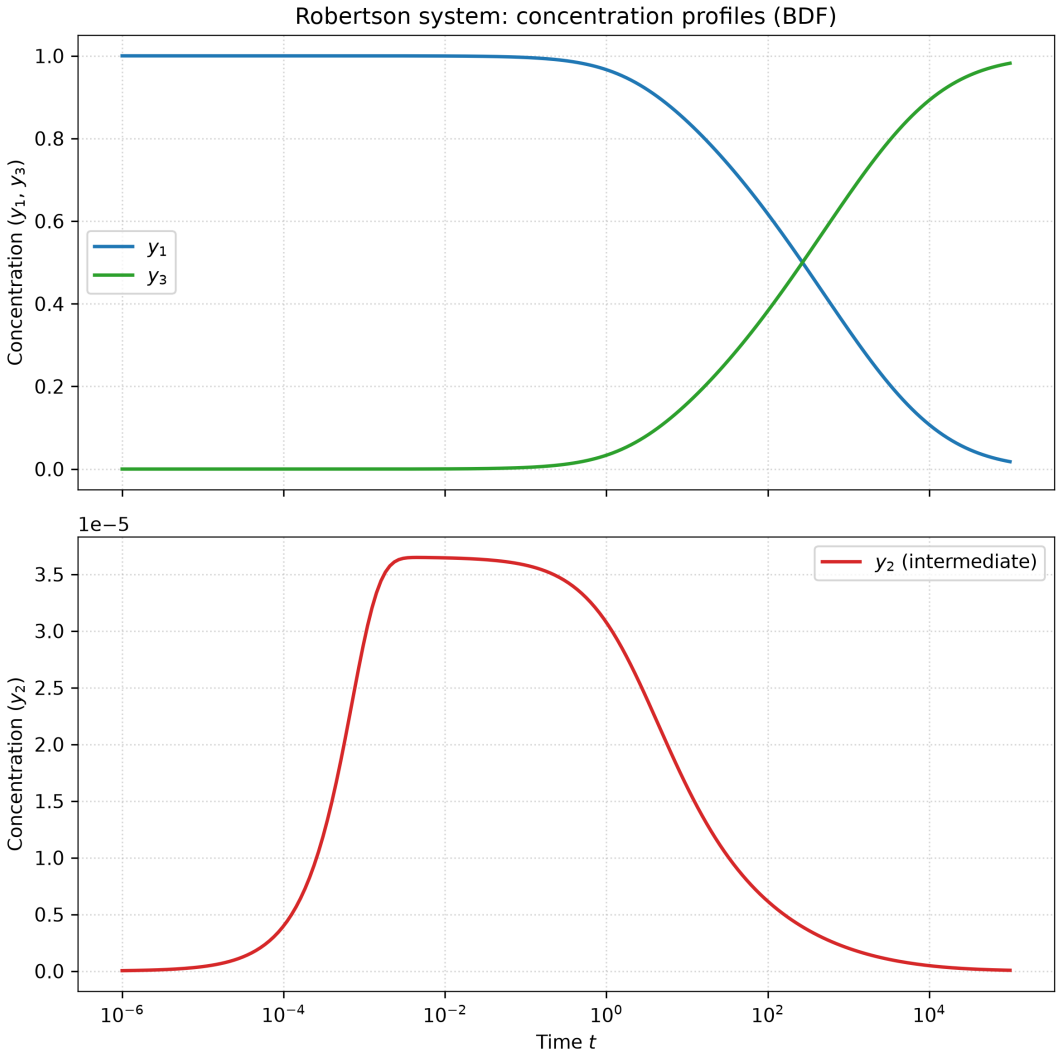

Figure 2: Concentration profiles of the Robertson system from the BDF reference solution, plotted against $\log t$. *Top:* the $\mathcal{O}(1)$ species y_1 (blue, reactant) and y_3 (green, product) exchange population over roughly five decades of time, crossing near $t \approx 30$. *Bottom:* the intermediate y_2 (red), shown on its own linear 10^{-5}-scaled axis, rises to a plateau of $\approx 3.6 \times 10^{-5}$ during the fast transient and then decays. The tiny magnitude of y_2 is precisely what necessitates the tight per-component absolute tolerance $\mathrm{atol}_2 = 10^{-14}$.

4.2 Adaptive step-size history

Figure 3 shows the sequence of accepted step sizes as a function of time on log–log axes. The adaptive controller begins with a minute step of $h \approx 6.6 \times 10^{-8}$ to resolve the initial fast transient during which y_2 is being established, then, as the solution settles onto its slow manifold and the local error estimate relaxes, it increases the step by more than eleven orders of magnitude, reaching $h \approx 3.6 \times 10^{3}$ near the end of the interval. The step size grows essentially in proportion to t once the fast modes are quiescent—the hallmark of a stiff solver operating in its intended regime, taking arbitrarily large steps across the slow dynamics because stability no longer constrains it. It is this multi-decade growth of the step size, from $\sim 10^{-7}$ to $\sim 10^{3}$, that allows the entire interval to be spanned in only 377 steps. The complete step-size sequence is delivered as `robertson_step_sizes.csv` for independent verification.

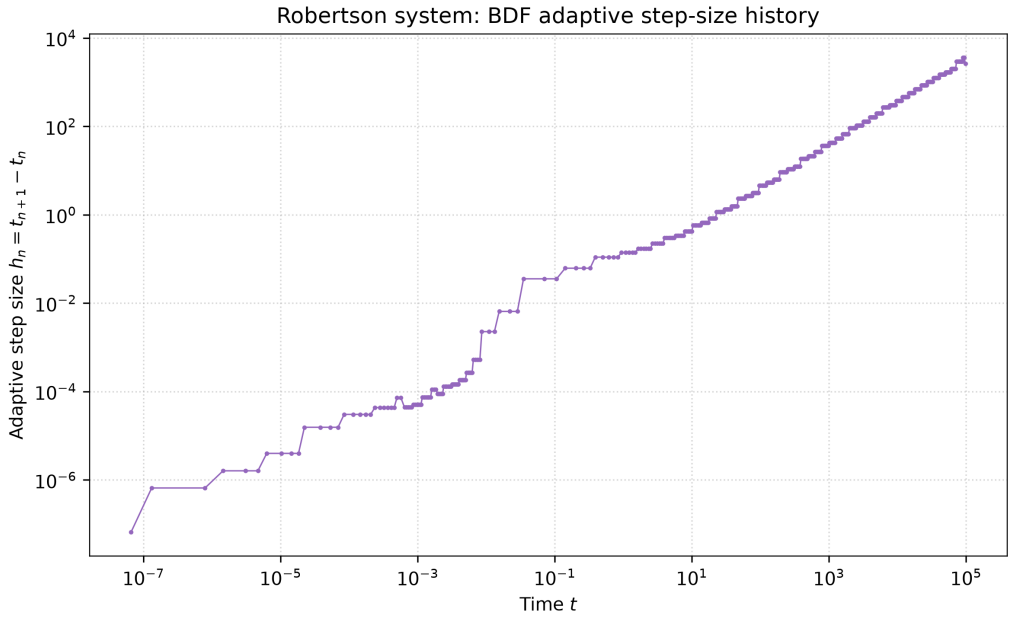

Figure 3: Adaptive step-size history of the BDF solver ($h_n = t_{n+1} - t_n$ versus t_n, log–log). The step size spans more than eleven orders of magnitude: tiny steps ($\sim 6.6 \times 10^{-8}$) resolve the initial stiff transient, growing to $\sim 3.6 \times 10^{3}$ once the solution reaches its slow manifold. The near-linear growth of h with t at late times is the signature of an implicit solver whose step is limited by accuracy rather than stability.

4.3 Mass conservation

Evaluating the invariant (5) at every one of the 377 accepted step points, we find the total concentration is conserved to

$$\max_n |y_1(t_n) + y_2(t_n) + y_3(t_n) - 1| = 1.33 \times 10^{-15}, \tag{11}$$

attained at $t \approx 543.4$, with a mean absolute deviation of 7.33×10^{-16}. Both figures are at the level of double-precision unit round-off ($\varepsilon_{\text{mach}} \approx 2.2 \times 10^{-16}$), confirming that the deviation from conservation is dominated by floating-point round-off rather than by integration error, and that the tolerance choices of Section 3.2 preserve the physical structure of the problem essentially exactly. This near-perfect conservation is a strong, independent endorsement of the reference solution's quality, since the invariant is never imposed on the solver.

4.4 The emergence of stiffness: eigenvalue analysis

Table 2 tracks the dominant Jacobian eigenvalue along the solution. The result overturns the naive expectation that a stiff problem is hardest at its initial instant. At $t = 0$ the spectrum is $\{-0.04, 0, 0\}$ and the only time scale is the slow one ($1/0.04 = 25$); a forward-Euler step as large as $h \lesssim 50$ would be stable there. Stiffness *emerges* only once the intermediate y_2 grows: through the $-6 \times 10^{7}\, y_2$ entry of the Jacobian, the dominant eigenvalue modulus climbs to $\approx 2.19 \times 10^{3}$ at the y_2 peak, and continues to rise—now driven by the $-10^{4}\, y_3$ term as $y_3 \to 0.98$—to a global maximum of $|\lambda|_{\max} \approx 9.83 \times 10^{3}$ near $t = 10^{5}$. The effective stiffness ratio between the fastest and slowest active time scales thus exceeds $\sim 2 \times 10^{5}$ by the end of the integration.

Table 2: Modulus of the dominant Jacobian eigenvalue $|\lambda|_{\max}$ evaluated along the reference solution, and the corresponding forward-Euler stability limit $h = 2/|\lambda|_{\max}$. The system is *not* stiff at $t = 0$; the binding constraint develops later and is largest at the end of the interval.

Evaluation point	$\|\lambda\|_{\max}$	FE limit $2/\|\lambda\|_{\max}$	Character
$t = 0$ (initial condition)	0.04	50	non-stiff
y_2 peak ($t \approx 10^{-2}$)	2.19×10^3	9.1×10^{-4}	stiff
Short horizon $[0, 20]$ (max)	2.91×10^3	6.9×10^{-4}	stiff
Global $[0, 10^5]$ (max, at $t \approx 10^5$)	9.83×10^3	2.0×10^{-4}	most stiff

4.5 Explicit stability limits and empirical h_{stable}

Table 3 compares the empirically measured maximum stable step sizes against the eigenvalue prediction. Searching directly over the horizon $[0, 20]$, we find forward Euler is stable up to $h^{\text{FE}}_{\text{stable}} = 7.70 \times 10^{-4}$ and RK4 up to $h^{\text{RK4}}_{\text{stable}} = 1.16 \times 10^{-3}$. These agree with the theoretical limits $C/|\lambda|_{\max}$ computed from the horizon eigenvalue ($|\lambda|_{\max} = 2.91 \times 10^3$) to within 12% (FE) and 21% (RK4). The modest excess of the empirical value over the theoretical bound is expected and physically meaningful: $|\lambda|_{\max}$ is an instantaneous upper bound, and the stiffest mode is not active over the whole short horizon, so the effective constraint is slightly looser than the worst-case estimate. The close agreement confirms that the observed instability is genuinely the eigenvalue-driven stability barrier of the explicit methods, not an artefact.

Table 3: Empirical versus theoretical maximum stable step size for the explicit integrators. "Empirical" values are measured by bisection over $[0, 20]$; "theory (horizon)" uses $C/|\lambda|_{\max}$ with the $[0, 20]$ eigenvalue 2.91×10^3; "theory (global)" uses the global maximum 9.83×10^3 that binds a full run to $t = 10^5$.

Method	C	h_{stable} (empirical)	h theory (horizon)	ratio emp./thy.	h theory (global)
Forward Euler	2.000	7.70×10^{-4}	6.86×10^{-4}	1.12	2.04×10^{-4}
RK4	2.785	1.16×10^{-3}	9.56×10^{-4}	1.21	2.83×10^{-4}

4.6 Numerical blow-up

Figure 4 demonstrates the instability directly. For step sizes below the (horizon) stability limit, both explicit methods produce a bounded, physical solution that tracks the BDF reference with $\|\mathbf{y}\|_\infty = 1.0$. A modest increase past the threshold is catastrophic: forward Euler with $h = 7.15 \times 10^{-4}$ diverges near $t \approx 25.2$, its infinity-norm exploding to $\approx 3.9 \times 10^2$, while RK4 with $h = 9.96 \times 10^{-4}$ diverges near $t \approx 31.6$, reaching $\approx 4.4 \times 10^5$. The blow-up is abrupt—the solution accumulates error silently and then, once the amplification factor $|1 + h\lambda|$ exceeds unity for the stiffest mode, grows exponentially over a handful of steps. This razor's-edge behaviour, stable one moment and unbounded the next, is the practical face of stiffness.

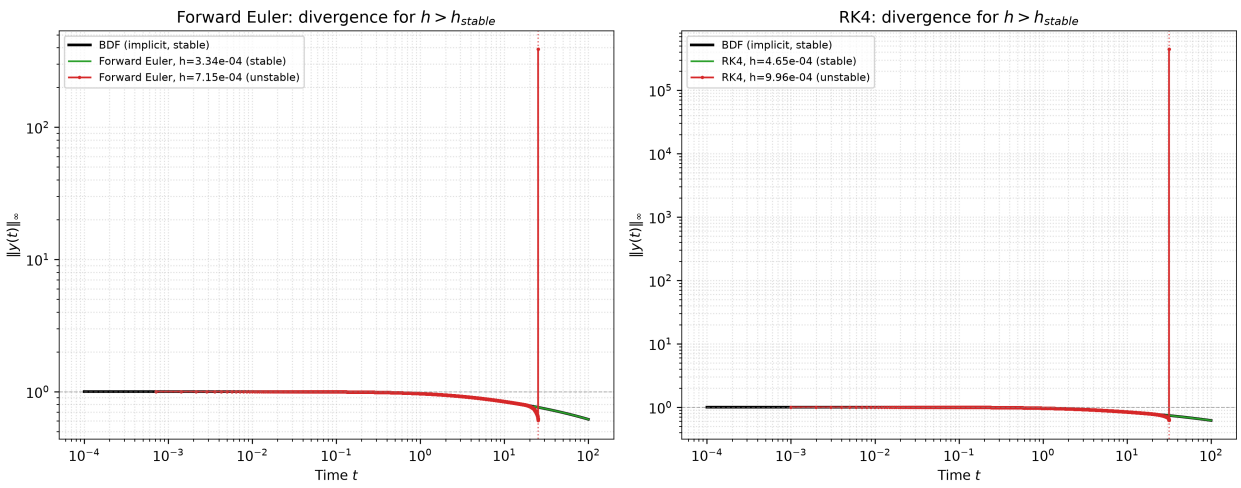

Figure 4: Explicit-method instability versus the stable implicit reference. Infinity-norm $\|\mathbf{y}(t)\|_\infty$ for forward Euler (left) and RK4 (right) at a step *below* (green) and *above* (red) the stability limit, overlaid on the BDF solution (black), which remains at 1.0. Below the limit the explicit solution is bounded and physical; a small step increase past h_{stable} triggers exponential blow-up near $t \approx 25$ (FE) and $t \approx 32$ (RK4). Note the logarithmic vertical scale.

4.7 Cost of reaching $t = 10^5$ with an explicit method

Because the binding stability constraint over the *full* interval is set by the global eigenvalue maximum $|\lambda|_{\max} \approx 9.83 \times 10^3$, an explicit method must use a fixed step no larger than $h_{\text{stable}}^{\text{global}}$ (2.04×10^{-4} for FE, 2.83×10^{-4} for RK4) for the entire run—even though the fast dynamics are long finished. The resulting step counts and extrapolated CPU costs are given in Table 4. Reaching $t = 10^5$ would require $\approx 4.9 \times 10^8$ forward-Euler steps or $\approx 3.5 \times 10^8$ RK4 steps, corresponding to an estimated 2.0 and 3.4 CPU-hours respectively (extrapolated from measured per-step timings). Against the 377 accepted steps of the adaptive BDF solver, this is a step-count penalty of roughly 10^6: the implicit method is about a million times more efficient in steps, and correspondingly faster in wall-clock time, despite each BDF step costing more through its Newton solve and LU factorisation.

Table 4: Extrapolated cost for a fixed-step explicit method to reach $t = 10^5$ using the globally stable step size, compared with the adaptive BDF solver. CPU estimates are extrapolated from measured per-step wall-clock times on the test machine.

Method	Stable step h	Steps to $t = 10^5$	Est. CPU time
Forward Euler (fixed step)	2.04×10^{-4}	4.91×10^8	≈ 2.0 h
RK4 (fixed step)	2.83×10^{-4}	3.53×10^8	≈ 3.4 h
Adaptive BDF (implicit)	**adaptive** ($6.6 \times 10^{-8} \to 3.6 \times 10^3$)	**377**	< 1 **s**
	Explicit/implicit step ratio:	$\sim 10^6$	

4.8 Adaptive explicit solvers also fail

One might hope that an error-controlled *adaptive* explicit method would escape this fate. It does not. SciPy's `RK45` (Dormand–Prince) and `DOP853` solvers reach the modest target $t = 1$ without

difficulty—in 4772 and 4274 right-hand-side evaluations respectively, in well under a tenth of a second—because adaptivity lets them exploit the small step the stiff transient demands. But when asked to reach $t = 10^5$, both were terminated by a 30-second wall-clock guard, having failed to complete: their adaptive step-size controllers are forced back to the stability-limited step $\sim 10^{-4}$ for the entire slow phase, so they too must take $\mathcal{O}(10^8)$ steps. This is exactly the situation that motivated automatic stiffness detection and method-switching in general-purpose solvers [21]: adaptivity controls *accuracy*, but it cannot lift the *stability* ceiling that defines stiffness. Only an implicit method, with a stability region that contains the stiff eigenvalues, can take the large steps the slow dynamics permit [7, 17].

5 Discussion

The results assembled above form a coherent, quantitative picture of stiffness on the Robertson benchmark. The implicit BDF solver crosses eleven decades of time in 377 steps while conserving mass to machine precision, whereas an explicit method of comparable per-step cost would need on the order of 10^8 steps and hours of CPU time to cover the same interval—if it completed at all. The factor of $\sim 10^6$ separating the two is not a matter of tuning or implementation quality; it is the intrinsic signature of a stiff problem, in which the ratio of the fastest to the slowest active time scale is enormous and an explicit method is forced to resolve the former even when only the latter is of interest.

Two aspects of the analysis deserve emphasis. First, the agreement between the empirically measured stable step sizes and the eigenvalue prediction $h \le C/|\lambda|_{\max}$—to within 12–21%—confirms that the observed explicit failure is precisely the linear-stability barrier of the methods, and validates the eigenvalue spectrum of the Jacobian as the correct diagnostic of stiffness. Second, and more subtly, the stiffness of the Robertson problem is *not* present at $t = 0$, where the Jacobian eigenvalues are $\{-0.04, 0, 0\}$ and even a large explicit step would be stable. Stiffness is an *emergent* property here: it appears only as the trace intermediate y_2 activates the fast $-6 \times 10^7\, y_2$ coupling, and it becomes most severe as $y_3 \to 0.98$ near the end of the run. This is why a fixed-step explicit integrator must honour the global eigenvalue maximum for the whole interval, and why the naive intuition that "the hard part is the initial layer" is misleading for this system.

The dual role of the absolute-tolerance vector is also worth restating. The tight $\mathrm{atol}_2 = 10^{-14}$ is what allows the $\mathcal{O}(10^{-5})$ intermediate to be resolved by its relative rather than its absolute error, keeping the fast quasi-steady balance $0.04\, y_1 \approx 3 \times 10^7\, y_2^2$ accurate and the conservation identity intact at the 10^{-15} level. Per-component tolerance scaling of this kind is essential whenever a system's components differ by many orders of magnitude, and the Robertson problem is the archetypal illustration [7, 13]. Our findings reproduce, on a modern open-source software stack [14], the classical conclusions of Robertson [5], Gear [3], Enright et al. [6], and Hairer and Wanner [7] regarding the necessity of implicit methods for stiff kinetics.

A few limitations should be acknowledged. The CPU-time figures for the explicit methods are extrapolations from measured per-step costs on short runs rather than timings of complete 10^8-step integrations, which would themselves take hours; they should be read as order-of-magnitude estimates. The empirical h_{stable} is measured over $[0, 20]$ for tractability, and thus reflects the horizon eigenvalue rather than the slightly stricter global maximum used for the cost extrapolation—a distinction we have been careful to make explicit. For production stiff chemistry at scale, further gains are available from Rosenbrock-type linearly-implicit methods [22], Krylov-based implicit solvers [11, 12], and specialised multiscale integration strategies [9]; the plain adaptive BDF used here is already more than sufficient for the three-species benchmark.

6 Conclusion

We have integrated the Robertson stiff chemical-kinetics system over $t \in [0, 10^5]$ with an adaptive, variable-order implicit BDF method using rtol $= 10^{-6}$ and per-component absolute tolerances atol $= [10^{-6}, 10^{-14}, 10^{-6}]$, the tight middle value being required to resolve the trace intermediate y_2. The integration succeeded in 377 accepted steps, delivering the reactant concentrations $y_1(1) = 0.96646$, $y_1(10^2) = 0.61724$, $y_1(10^4) = 0.10730$, and $y_1(10^5) = 0.017866$, and conserving total mass to a maximum deviation of 1.33×10^{-15}—machine precision. Custom explicit integrators exposed the stiffness quantitatively: their empirically measured stable steps (7.7×10^{-4} for forward Euler, 1.2×10^{-3} for RK4) match the eigenvalue bound $C/|\lambda|_{\max}$ to within 12–21%; they blow up abruptly past that limit; and reaching $t = 10^5$ would cost them ~ 3.5–4.9×10^8 steps and hours of CPU time, versus 377 BDF steps. Contrary to a common assumption, the system is not stiff at $t = 0$ (eigenvalues $\{-0.04, 0, 0\}$); stiffness emerges as the fast modes activate and is most severe near $t = 10^5$. The log-spaced solution trajectory and the complete adaptive step-size history are delivered as CSV files, so that every number in this report can be independently regenerated and checked.

A Reproducibility

All results were produced with Python 3.12, NumPy $\geq$ 2.1 [19], SciPy $\geq$ 1.14 [14], and Matplotlib $\geq$ 3.10 [20]. The implicit reference solution used the following invocation.

```
import numpy as np
from scipy.integrate import solve_ivp

def robertson(t, y):
    y1, y2, y3 = y
    return [-0.04*y1 + 1e4*y2*y3,
             0.04*y1 - 1e4*y2*y3 - 3e7*y2*y2,
             3e7*y2*y2]

def jac(t, y):
    y1, y2, y3 = y
    return [[-0.04,        1e4*y3,              1e4*y2],
            [ 0.04, -1e4*y3 - 6e7*y2,          -1e4*y2],
            [ 0.0,          6e7*y2,              0.0 ]]

sol = solve_ivp(robertson, (0.0, 1e5), [1.0, 0.0, 0.0],
                method="BDF", jac=jac,
                rtol=1e-6, atol=[1e-6, 1e-14, 1e-6],
                dense_output=True)
# Target states:
sol.sol([1.0, 1e2, 1e4, 1e5])
# Step-size history:  h = np.diff(sol.t)
```

Delivered data files. The following CSV files accompany this report and together permit the integration to be re-run and every reported quantity checked:

- `robertson_trajectory.csv` — 200 log-spaced points, $t \in [10^{-6}, 10^5]$; columns t, y_1, y_2, y_3.
- `robertson_step_sizes.csv` — adaptive step history; columns t, h.
- `robertson_target_states.csv` — states at $t = 1, 10^2, 10^4, 10^5$.
- `robertson_conservation.csv` — conservation summary.

- `robertson_stiffness_comparison.csv`, `robertson_blowup_summary.csv`, `robertson_standard_solvers.csv`, and `robertson_unstable_trajectory.csv` — stiffness experiments.

References

[1] C. F. Curtiss and J. O. Hirschfelder. Integration of stiff equations. *Proceedings of the National Academy of Sciences*, 38(3):235–243, 1952. doi: 10.1073/pnas.38.3.235.

[2] Germund G. Dahlquist. A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 3(1):27–43, 1963. doi: 10.1007/BF01963532.

[3] C. William Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[4] C. W. Gear. The automatic integration of ordinary differential equations. *Communications of the ACM*, 14(3):176–179, 1971. doi: 10.1145/362566.362571.

[5] H. H. Robertson. The solution of a set of reaction rate equations. In J. Walsh, editor, *Numerical Analysis: An Introduction*, pages 178–182. Academic Press, London, 1966.

[6] W. H. Enright, T. E. Hull, and B. Lindberg. Comparing numerical methods for stiff systems of O.D.E:s. *BIT Numerical Mathematics*, 15(1):10–48, 1975. doi: 10.1007/BF01932994.

[7] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, Heidelberg, 2nd edition, 1996. doi: 10.1007/978-3-642-05221-7.

[8] Francesca Mazzia, Jeff R. Cash, and Karline Soetaert. A test set for stiff initial value problem solvers in the open source software R: Package deTestSet. *Journal of Computational and Applied Mathematics*, 236(16):4119–4131, 2012. doi: 10.1016/j.cam.2012.03.014.

[9] Dezhi Zhou and Wenming Yang. A heterogeneous multiscale method for stiff combustion chemistry integration in reactive flows. *Combustion and Flame*, 188:428–439, 2018. doi: 10.1016/j.combustflame.2017.09.039.

[10] George D. Byrne and Alan C. Hindmarsh. A polyalgorithm for the numerical solution of ordinary differential equations. *ACM Transactions on Mathematical Software*, 1(1):71–96, 1975. doi: 10.1145/355626.355636.

[11] Peter N. Brown, George D. Byrne, and Alan C. Hindmarsh. VODE: A variable-coefficient ODE solver. *SIAM Journal on Scientific and Statistical Computing*, 10(5):1038–1051, 1989. doi: 10.1137/0910062.

[12] Alan C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, 31(3):363–396, 2005. doi: 10.1145/1089014.1089020.

[13] Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997. doi: 10.1137/S1064827594276424.

[14] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[15] Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, Heidelberg, 2nd edition, 1993. doi: 10.1007/978-3-540-78862-1.

[16] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley & Sons, Chichester, 1991.

[17] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. doi: 10.1137/1.9781611971392.

[18] Gustaf Söderlind. Automatic control and adaptive time-stepping. *Numerical Algorithms*, 31 (1–4):281–310, 2002. doi: 10.1023/A:1021160023092.

[19] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2.

[20] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[21] Linda Petzold. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM Journal on Scientific and Statistical Computing*, 4(1):136–148, 1983. doi: 10.1137/0904010.

[22] H. H. Rosenbrock. Some general implicit processes for the numerical solution of differential equations. *The Computer Journal*, 5(4):329–330, 1963. doi: 10.1093/comjnl/5.4.329.