Numerical Techniques PSLTDSim utilizes a variety of numerical techniques to perform integration involved with dynamic agents. Some of the employed methods are coded 'by hand', while others utilize Python packages. This appendix is meant to provide more detail on available integration methods for system frequency, cover how Laplace style block diagrams are computed by PSLTDSim, and explain how agents currently handle the integration of running values.

Combined Swing Equation Solutions The options included in PSLTDSim to solve the combined swing equation for a new system frequency are Euler, Adams-Bashforth, and Runge-Kutta. Each of these methods are numerical approximations that provide an *approximation* to the solution of an initial value problem. Method functions presented below were adapted from XXXrefBoyceDiprima.

Euler Method Of the of integration methods available to solve system frequency, the Euler method is the simplest. In general terms, to find the next y value given some function $f(t, y)$ is

$$y_{n+1} = y_n + f(t_n, y_n)t_s, \tag{1}$$

where t_s is desired time step. The next value of y is simply a projection along a line tangent to f at time t. It should be noted that the accuracy of approximation methods is often related to the time step size.

Runge-Kutta Method Improving on the Euler method, the Runge-Kutta method combines numerous projections as a weighted average to find the next y value. The fourth order four-stage Runge-Kutta method is shown as Equation 2.

$$\begin{aligned} k_1 &= f(t_n,\ y_n) \\ k_2 &= f(t_n + t_s/2,\ y_n + t_s k_1/2) \\ k_3 &= f(t_n + t_s/2,\ y_n + t_s k_2/2) \\ k_4 &= f(t_n + t_s,\ y_n + t_s k_3) \\ y_{n+1} &= y_n + t_s(k_1 + 2k_2 + 2k_3 + k_4)/6 \end{aligned} \tag{2}$$

It can be seen that k_1 and k_4 are on either side of the interval of approximation defined by the time step ts, and k_2 and k_3 represent midpoints.

Adams-Bashforth Method Unlike previously introduced methods, the Adams-Bashforth method requires data from previous time steps. Methods of this nature are sometimes referred to as multistep or predictor-corrector methods. A two-step approximation is described in Equation 3, however, larger step methods do exist.

$$y_{n+1} = y_n + t_s\left(1.5f(t_n, y_n) - 0.5f(t_{n-1}, y_{n-1})\right) \tag{3}$$

Regardless of the number of steps, the Adams-Bashforth methods utilize a weighted combination of values similar to the Runge-Kutta method, but using only previous data.

Scipy solve_ivp The Scipy solve_ipv function is capable of numerically integrating ODEs.

Trapezoidal Integration To integrate known values generated each time step, a trapezoidal integration method is used. Given some value $x(t)$, the trapezoidal method states that

$$\int_{t-t_s}^{t_s} x(t)\mathrm{d}t \approx t_s \left(x(t) + x(t-ts)\right)/2, \tag{4}$$

where t_s is the time step used between calculated values of x. Visually, this method can be thought of connecting the two y values with a straight line, then calculating the area of the trapezoid formed between. This technique is used in a variety of agents, such as the BA agent for IACE, and the window integrator agent.

Python Code Example To compare the resulting approximates from each method a Python script was created.

Approximation Result Comparisons Using the code i

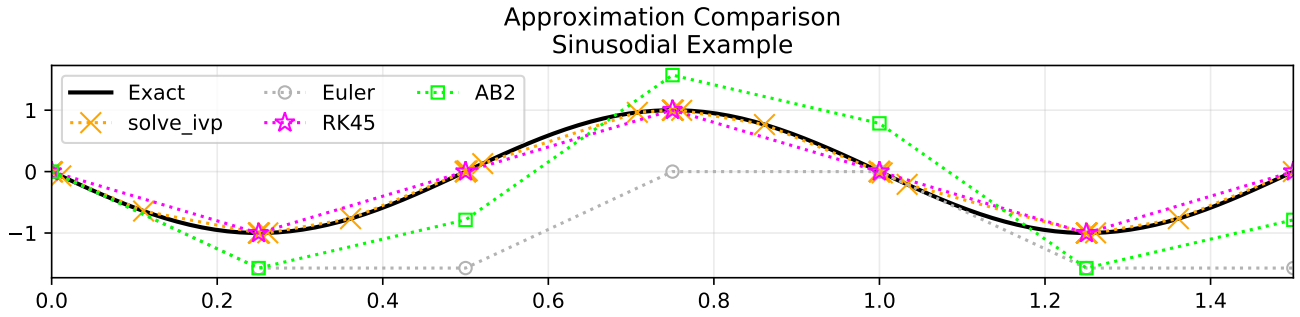

Figure 1: Approximation comparison of a sinusoidal function.

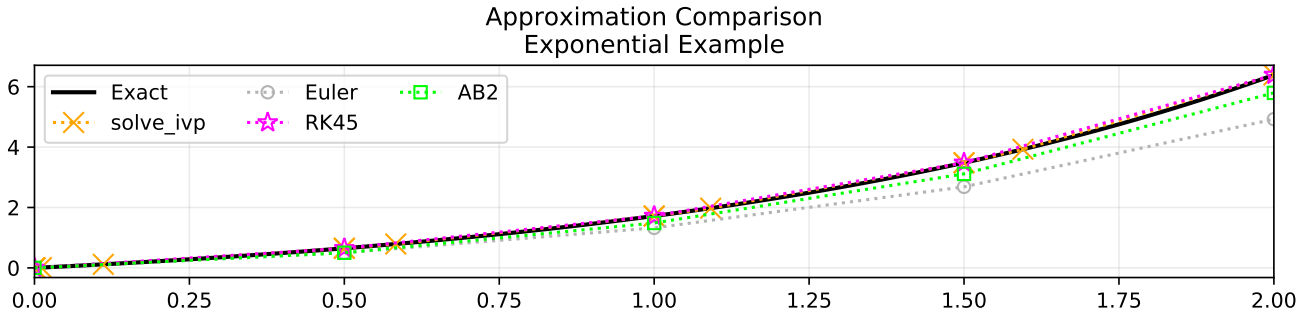

Figure 2: Approximation comparison of an exponential function.

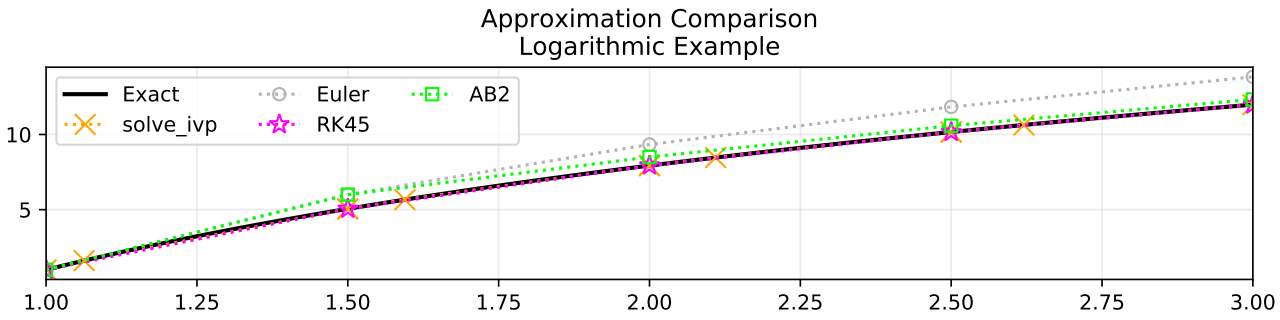

Figure 3: Approximation comparison of a logarithmic function.

sig.lsim To process a signal through a Laplace block, the block is first transformed into state space matrices and then inserted into sig.lsim for a linear simulation. This is used in all dynamic processes of PSLTDSim including governor states, filter and delay blocks.