Astrodynamics Software and Science Enabling Toolkit (ASSET) Training

NESC Training – Flight Mechanics Tech. Discipline Team

Astrodynamics and Space Research Laboratory

Presenter: Aaron Houin
PI: Dr. Rohan Sood

The University of Alabama, Tuscaloosa AL

Training Overview

- Day 1: Introduction to ASSET's core functionality
  - Vector Functions - "Basic building blocks used in nearly all ASSET operations"
  - ODEs and Integrators - "Defining solution spaces and integrating the dynamics"
  - Phases - "Setting up optimization problems"
  - Optimal Control Problems (OCPs) - "Configuring complex, multi-phase optimization problems"
- Day 2: Using the "Astro Library" for quick astrodynamics modeling
  - Two-Body Problem Example
  - N-Body Problem Example
  - CR3BP Example
  - Ephemeris Pulsing Rotating Example

Phases

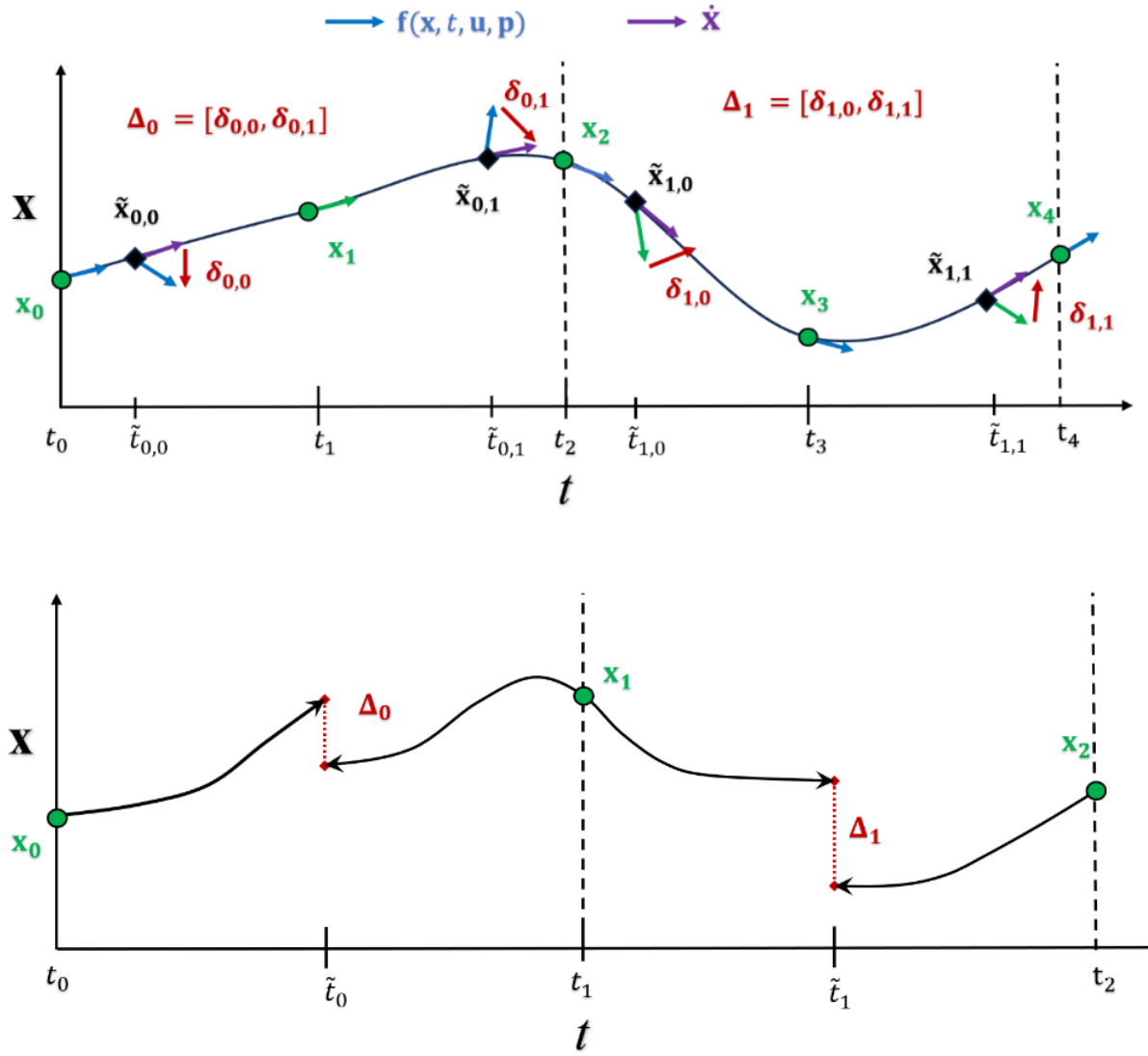
Phases: Overview

- Interface for solving optimal control and boundary value problems
  - Its extensive, please read tutorial
- Employs a **transcription** to turn dynamics into equality constraints
- Apply additional constraints and objectives to specify problem
- Add additional phase static parameters (e.g. $\mathbf{s}$)

$$\begin{aligned}
\text{minimize:} \quad & J = \phi(\mathbf{y}_{0,f}, \mathbf{s}) + \int_{t_0}^{t_f} \psi(\mathbf{y}, \mathbf{s}) \\
\text{subject to:} & \\
\text{dynamics:} \quad & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t, \mathbf{u}, \mathbf{p}) = \mathbf{f}(\mathbf{y}) \\
\text{boundary:} \quad & \mathbf{c_b}(\mathbf{x}_{0,f}, t_{0,f}, \mathbf{u}_{0,f}, \mathbf{p}, \mathbf{s}) = \mathbf{c_b}(\mathbf{y}_{0,f}, \mathbf{s}) = \mathbf{0} \\
& \mathbf{g_b}(\mathbf{x}_{0,f}, t_{0,f}, \mathbf{u}_{0,f}, \mathbf{p}, \mathbf{s}) = \mathbf{g_b}(\mathbf{y}_{0,f}, \mathbf{s}) \leq \mathbf{0} \\
\text{path:} \quad & \mathbf{c_p}(\mathbf{x}, t, \mathbf{u}, \mathbf{p}) = \mathbf{c_p}(\mathbf{y}, \mathbf{s}) = \mathbf{0} \\
& \mathbf{g_p}(\mathbf{x}, t, \mathbf{u}, \mathbf{p}, \mathbf{s}) = \mathbf{g_p}(\mathbf{y}, \mathbf{s}) \leq \mathbf{0}
\end{aligned}$$

Phases: Transcriptions

- Turns dynamics constraint in equality constraints
- Legendre Gauss Lobatto (LGL) Collocation
  - Orders: 3,5,7
  - **Most robust option, use for most problems**
- Adaptive central shooting scheme
  - Dormand Prince 8(7) method (same as integrator)
  - Error controlled

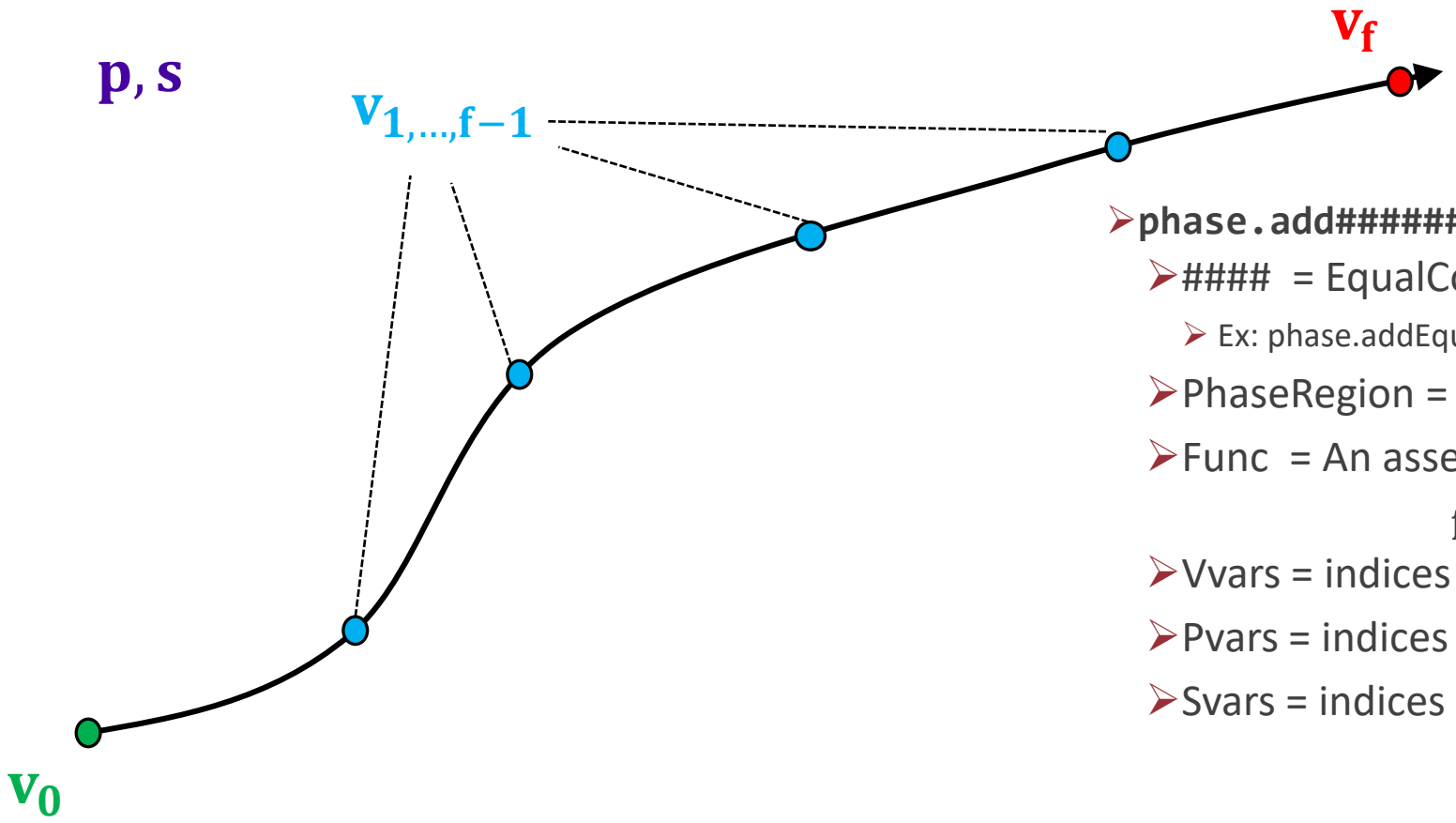


Phases: Initialization

- Must specify the transcription, initial guess for trajectory, and number of segments in the phase
- Internal initial guess will be interpolated
- By default, segments will be evenly spaced in time
- Specify phase static params (if any) by setting their initial values

```
transcription = "LGL3"
nsegments = 256

phase = ltode.phase(transcription,traj,nsegments)

phase.setStaticParams([1.0,3.14])#Phase now has 2 static params
```

Adding General Constraints & Objectives



- **`phase.add######(PhaseRegion,Func,Vvars,Pvars,Svars)`**
  - `####` = EqualCon, InequalCon, StateObjective
    - Ex: phase.addEqualCon
  - PhaseRegion = Source location of Vvars in trajectory
  - Func = An asset vector function defining constraint/objective

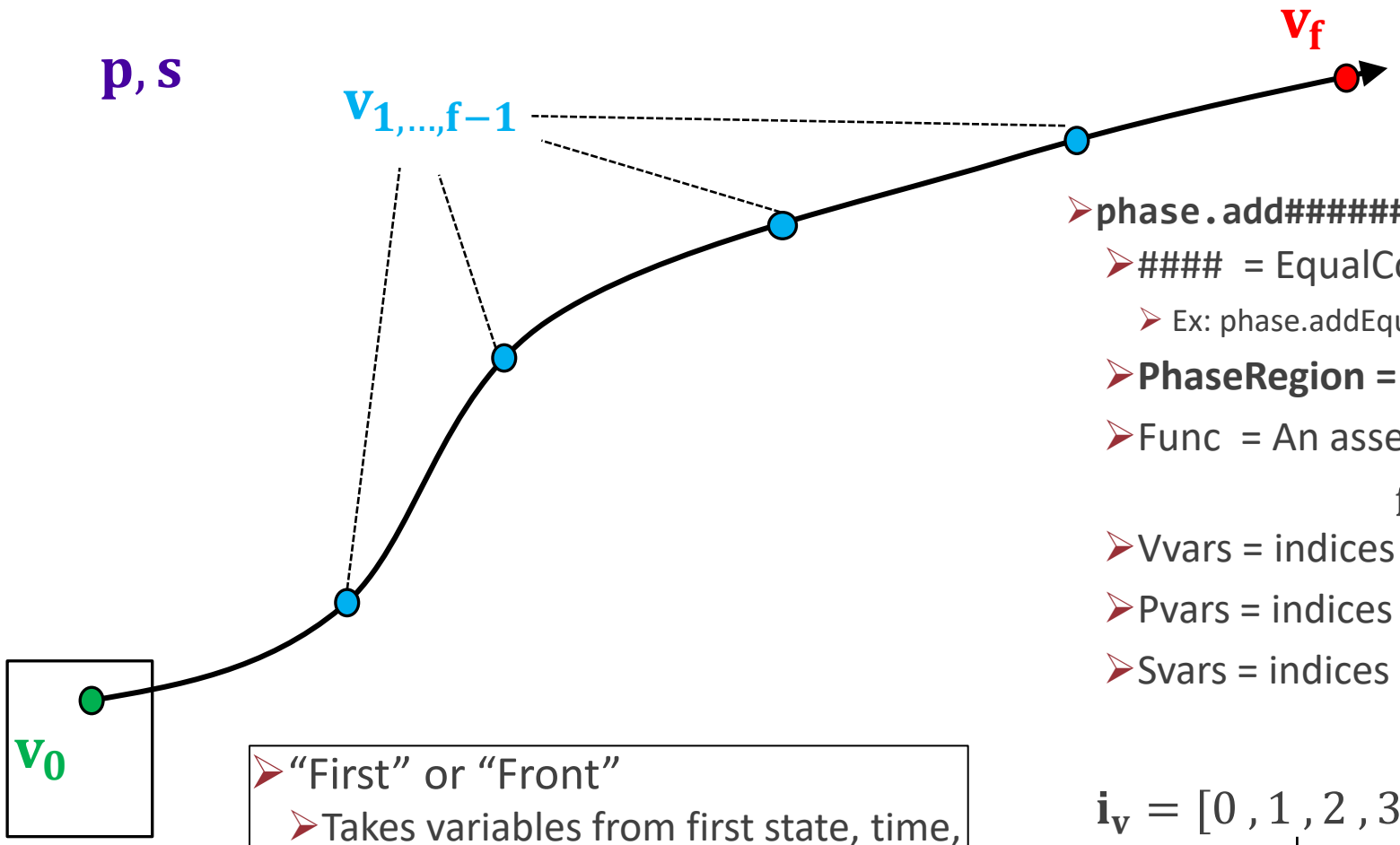
$$\mathbf{f}([\mathbf{v}_i, \mathbf{p}, \mathbf{s}])$$

  - Vvars = indices of input state, time, control variables to input
  - Pvars = indices of ODE parameter variables to input
  - Svars = indices of phase static parameter variables to input

$$\mathbf{i_v} = [0\,, 1\,, 2\,, 3\ ] \qquad \mathbf{i_p} = [0] \qquad \mathbf{i_s} = [0]$$

$$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}] \qquad \mathbf{p} = [p_0\ ] \qquad \mathbf{s} = [s_0\ ]$$

Adding General Constraints & Objectives



- `phase.add######(PhaseRegion,Func,Vvars,Pvars,Svars)`
  - `####` = EqualCon, InequalCon, StateObjective
    - Ex: phase.addEqualCon
  - **PhaseRegion = Source location of Vvars in trajectory**
  - Func = An asset vector function defining constraint/objective

$$\mathbf{f}([\mathbf{v}_0, \mathbf{p}, \mathbf{s}])$$

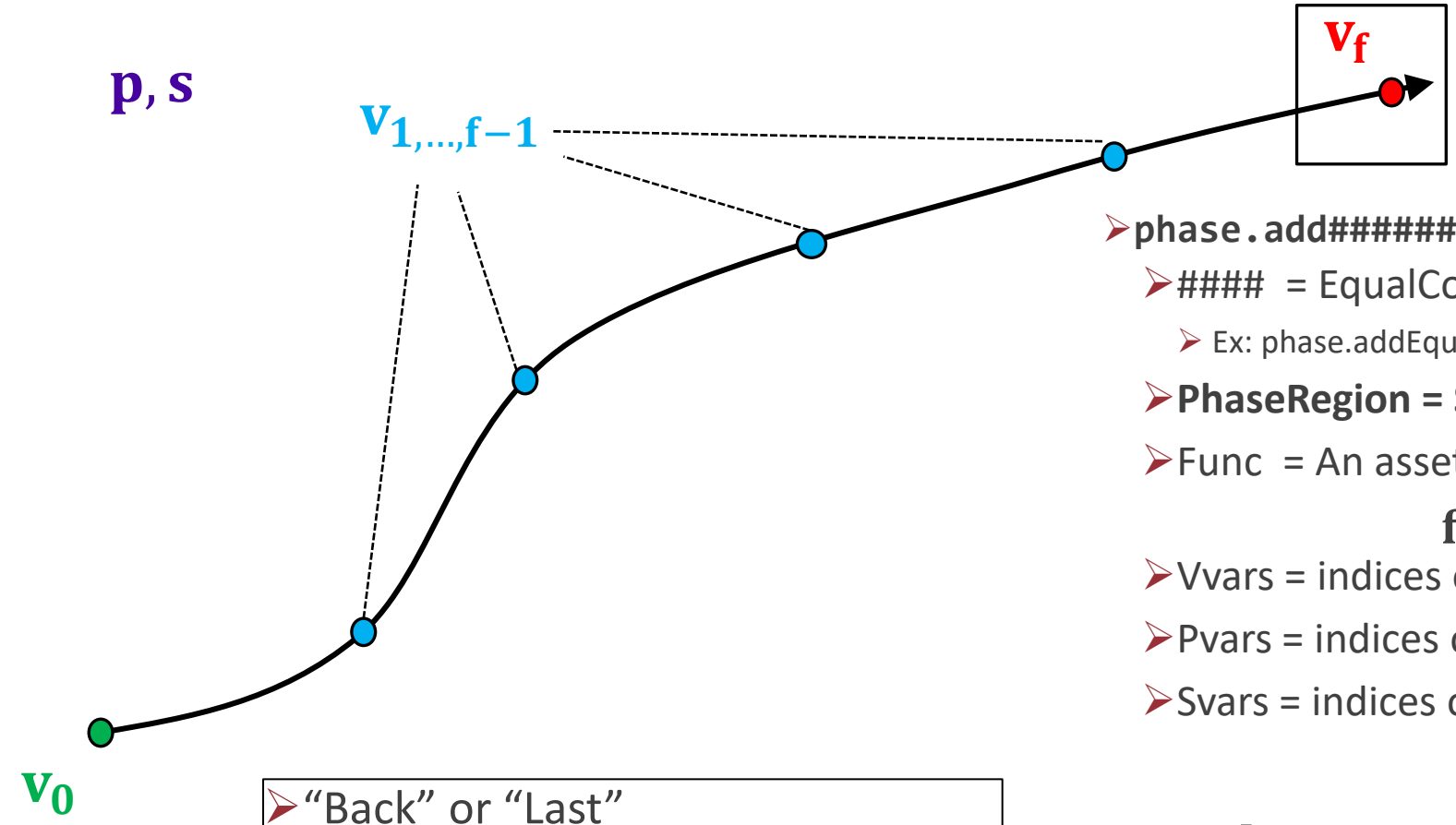
  - Vvars = indices of input state, time, control variables to input
  - Pvars = indices of ODE parameter variables to input
  - Svars = indices of phase static parameter variables to input

- "First" or "Front"
  - Takes variables from first state, time, controls in trajectory
  - Ex: Initial conditions

$$\mathbf{i_v} = [0\,, 1\,, 2\,, 3\ ] \qquad \mathbf{i_p} = [0] \qquad \mathbf{i_s} = [0]$$

$$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}] \qquad \mathbf{p} = [p_0\ ] \qquad \mathbf{s} = [s_0\ ]$$

Adding General Constraints & Objectives



- `phase.add######(PhaseRegion,Func,Vvars,Pvars,Svars)`
  - `####` = EqualCon, InequalCon, StateObjective
    - Ex: phase.addEqualCon
  - **PhaseRegion = Source location of Vvars in trajectory**
  - Func = An asset vector function defining constraint/objective

$$\mathbf{f}([\mathbf{v}_f, \mathbf{p}, \mathbf{s}])$$

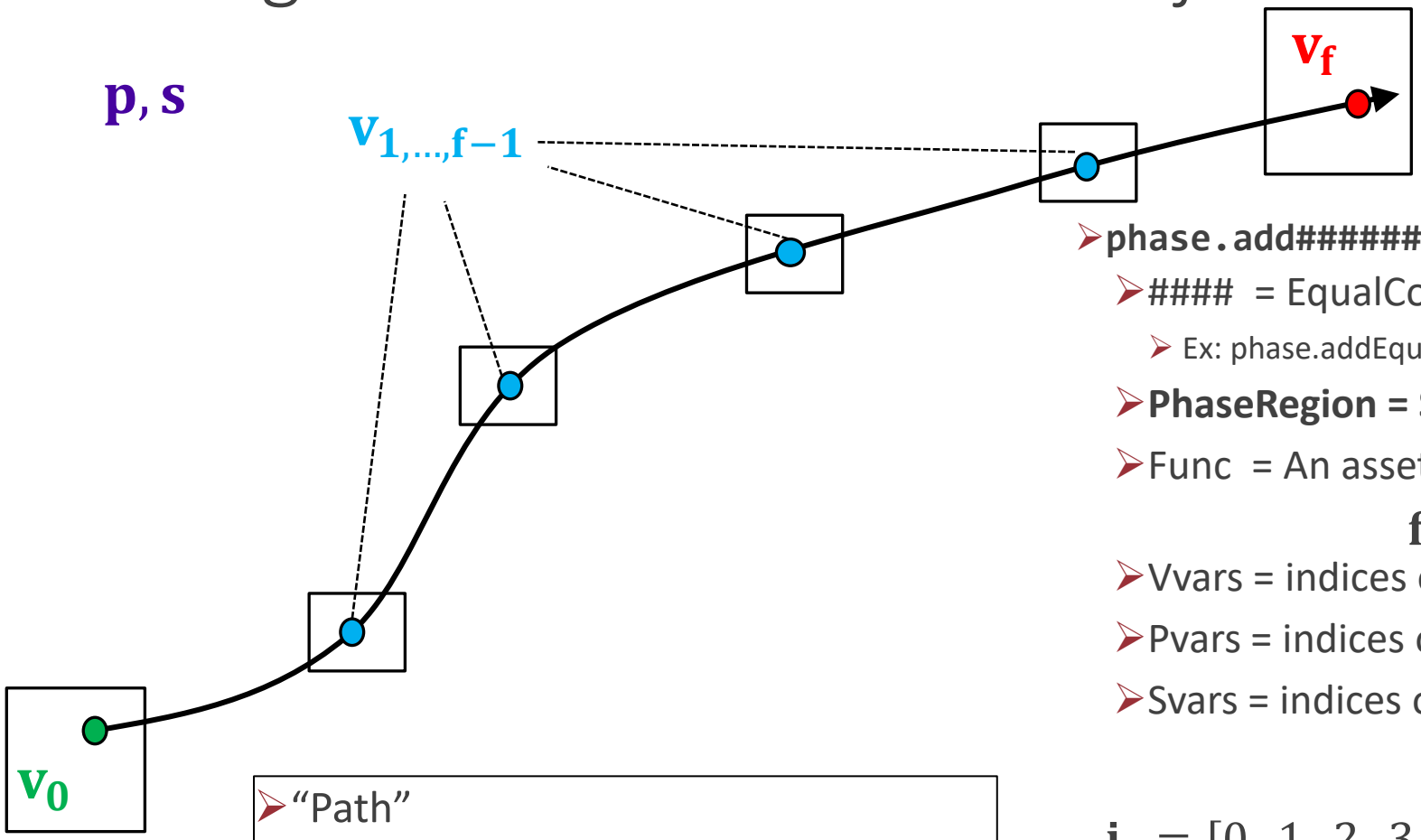
  - Vvars = indices of input state, time, control variables to input
  - Pvars = indices of ODE parameter variables to input
  - Svars = indices of phase static parameter variables to input

- "Back" or "Last"
  - Takes variables from last state, time, controls in trajectory
  - Ex: Terminal conditions, objectives

$$\mathbf{i_v} = [0\,, 1\,, 2\,, 3\ ] \qquad \mathbf{i_p} = [0] \qquad \mathbf{i_s} = [0]$$

$$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}] \qquad \mathbf{p} = [p_0\ ] \qquad \mathbf{s} = [s_0\ ]$$

Adding General Constraints & Objectives



- **phase.add######(PhaseRegion,Func,Vvars,Pvars,Svars)**
  - #### = EqualCon, InequalCon, StateObjective
    - Ex: phase.addEqualCon
  - **PhaseRegion = Source location of Vvars in trajectory**
  - Func = An asset vector function defining constraint/objective

$$\mathbf{f}([\mathbf{v}_i, \mathbf{p}, \mathbf{s}]) \quad i = 0 \ \ldots f$$

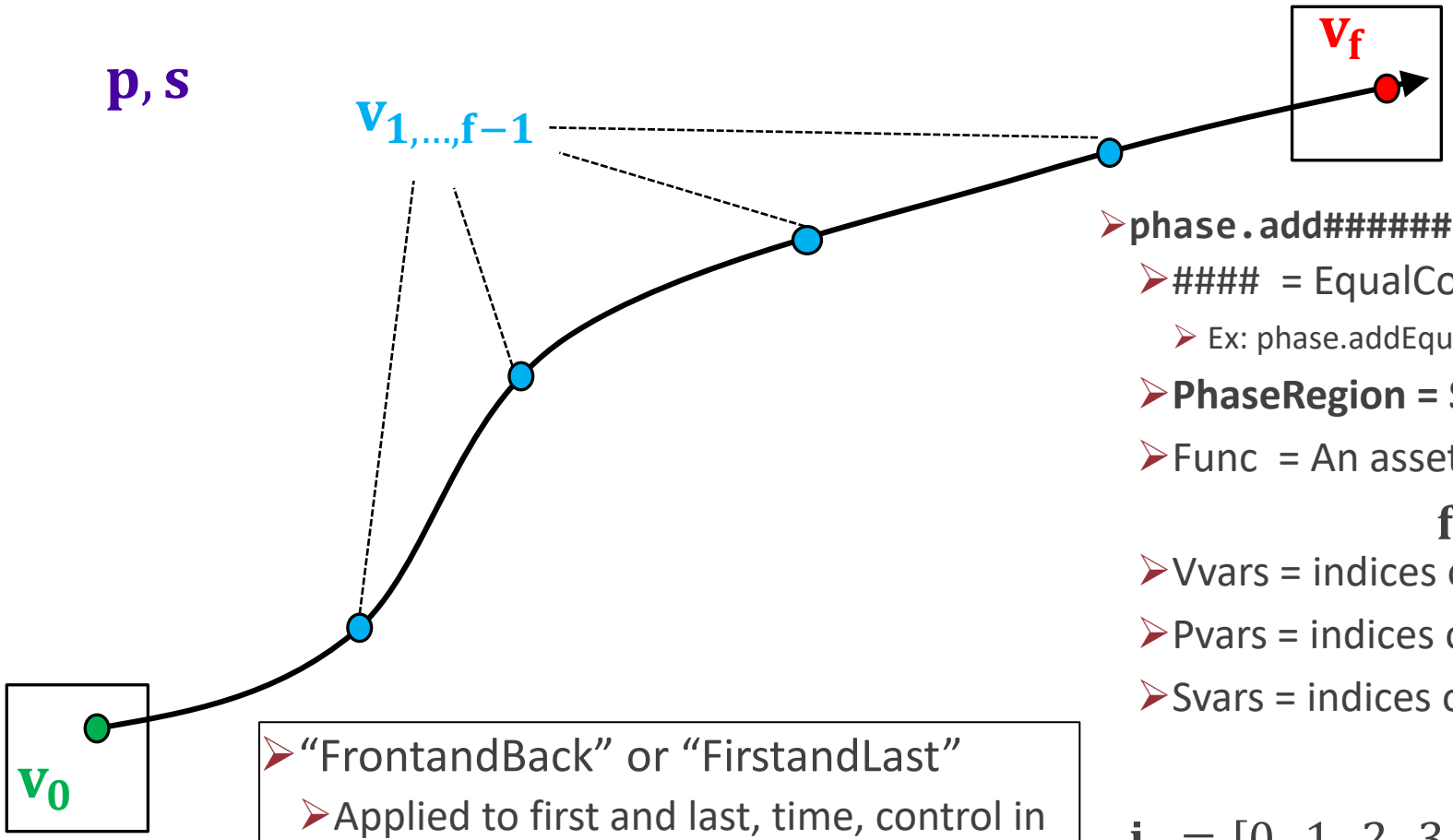
  - Vvars = indices of input state, time, control variables to input
  - Pvars = indices of ODE parameter variables to input
  - Svars = indices of phase static parameter variables to input

- "Path"
  - Applied to every state, time, control in trajectory independently
  - Ex: Control Bounds, Keep out constraint

$$\mathbf{i_v} = [0\,, 1\,, 2\,, 3\ ] \qquad \mathbf{i_p} = [0] \qquad \mathbf{i_s} = [0]$$

$$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}] \qquad \mathbf{p} = [p_0\ ] \qquad \mathbf{s} = [s_0\ ]$$

Adding General Constraints & Objectives



- `phase.add######(PhaseRegion,Func,Vvars,Pvars,Svars)`
  - `####` = EqualCon, InequalCon, StateObjective
    - Ex: phase.addEqualCon
  - **PhaseRegion = Source location of Vvars in trajectory**
  - Func = An asset vector function defining constraint/objective

$$\mathbf{f}([\mathbf{v}_0, \mathbf{v}_f, \mathbf{p}, \mathbf{s}])$$

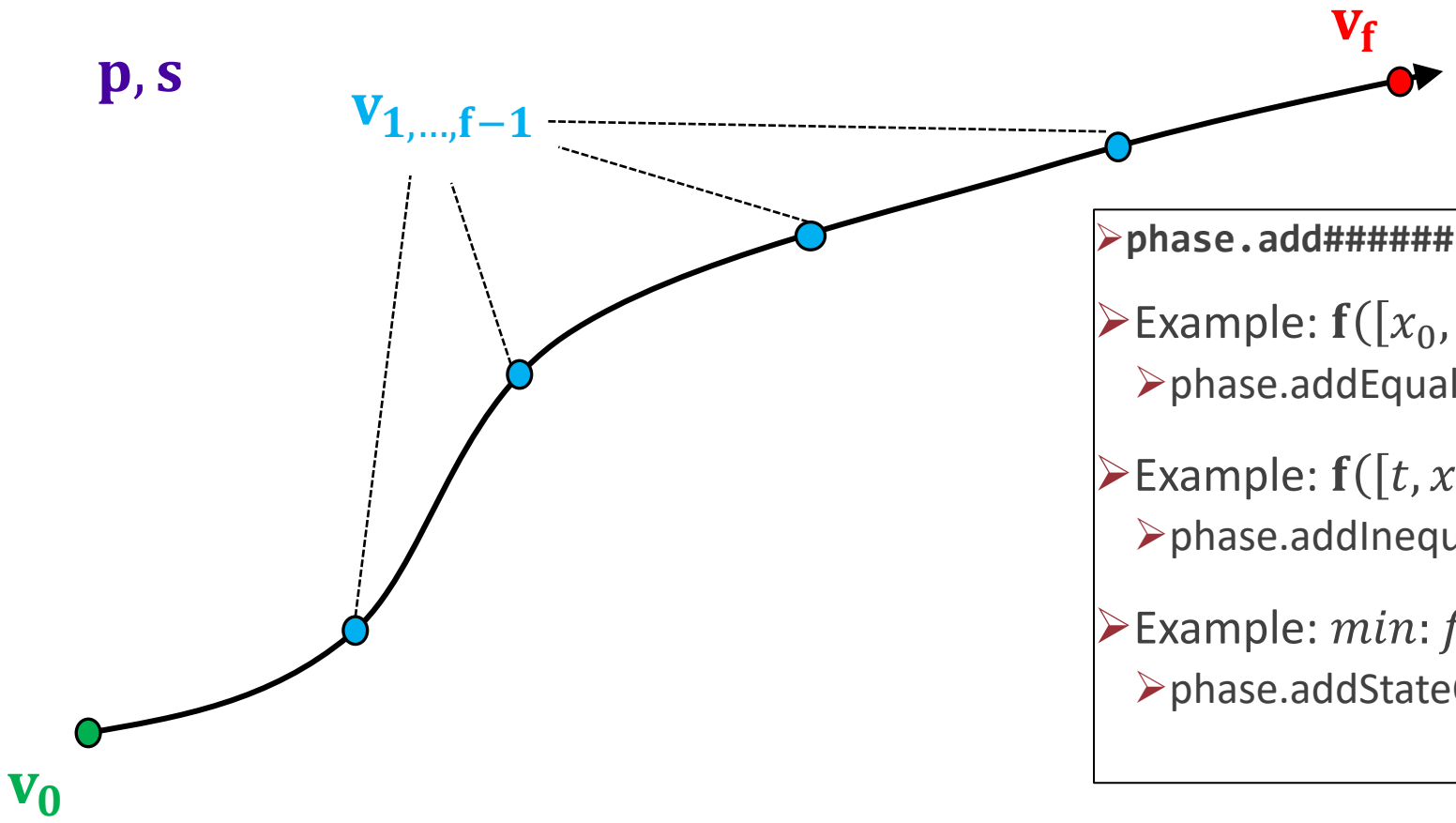
  - Vvars = indices of input state, time, control variables to input
  - Pvars = indices of ODE parameter variables to input
  - Svars = indices of phase static parameter variables to input

- "FrontandBack" or "FirstandLast"
  - Applied to first and last, time, control in trajectory **together**
  - Ex: Periodicity constraint, delta time objective

$$\mathbf{i_v} = [0\,, 1\,, 2\,, 3\ ] \qquad \mathbf{i_p} = [0] \qquad \mathbf{i_s} = [0]$$

$$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}] \qquad \mathbf{p} = [p_0\ ] \qquad \mathbf{s} = [s_0\ ]$$

Adding General Constraints & Objectives

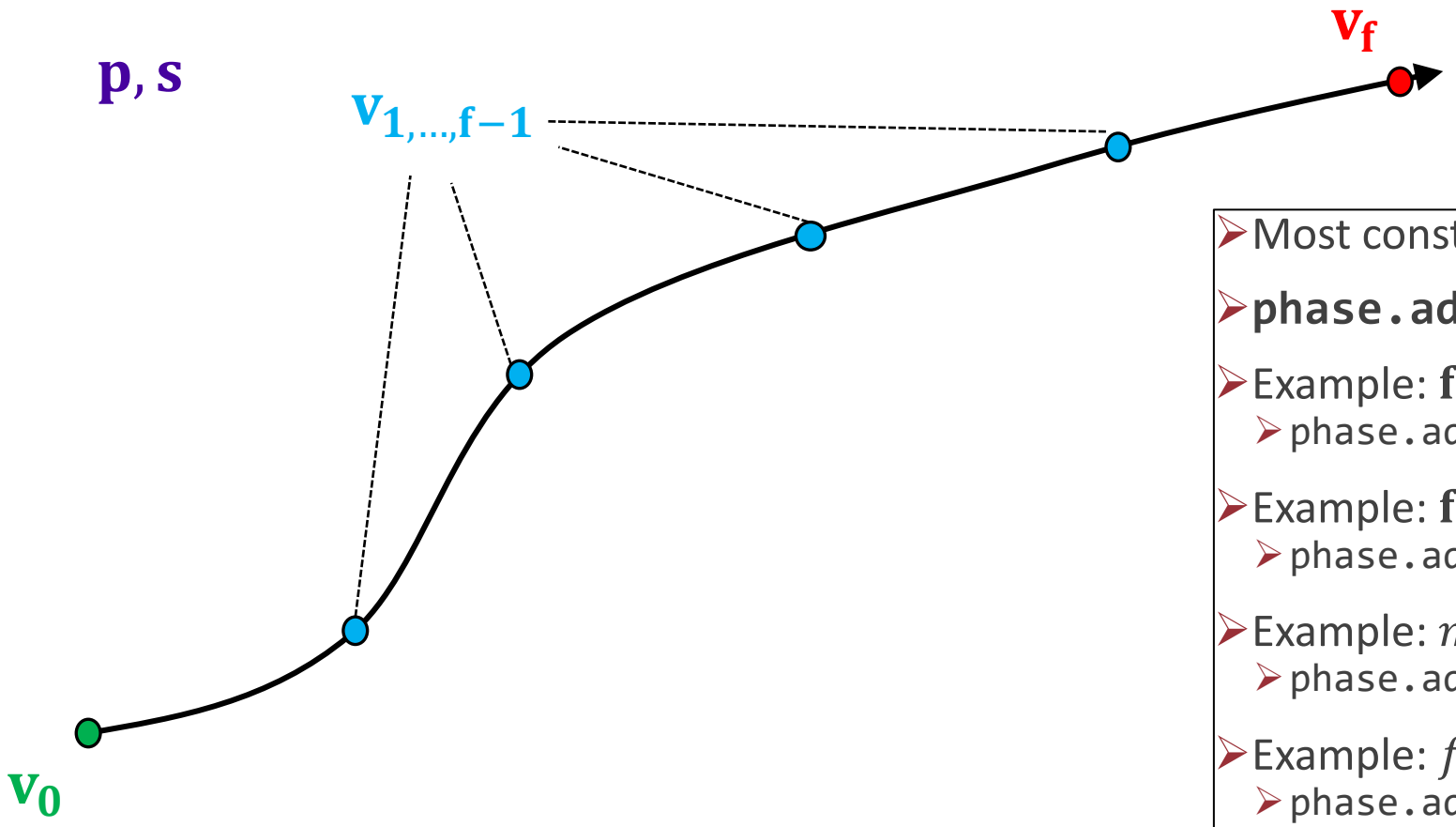


- `phase.add######(PhaseRegion,Func,Vvars,Pvars,Svars)`
- Example: $\mathbf{f}([x_0, t, u_0, p_0]) = \mathbf{0}$ at first state in trajectory
  - phase.addEqualCon("First", f, [0,2,3] , [0], [])
- Example: $\mathbf{f}([t, x_0, s_0]) < \mathbf{0}$ at last state in trajectory
  - phase.addInequalCon("Last", f, [2,0] , [], [0])
- Example: min: $f([t])$ at last state in trajectory
  - phase.addStateObjective("Last", f, [2] , [], [])

$\mathbf{i_v} = [0\,, 1\,, 2\,, 3\ ]$ $\quad \mathbf{i_p} = [0]$ $\quad \mathbf{i_s} = [0]$

$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}]$ $\quad \mathbf{p} = [p_0\ ]$ $\quad \mathbf{s} = [s_0\ ]$

Adding General Constraints & Objectives

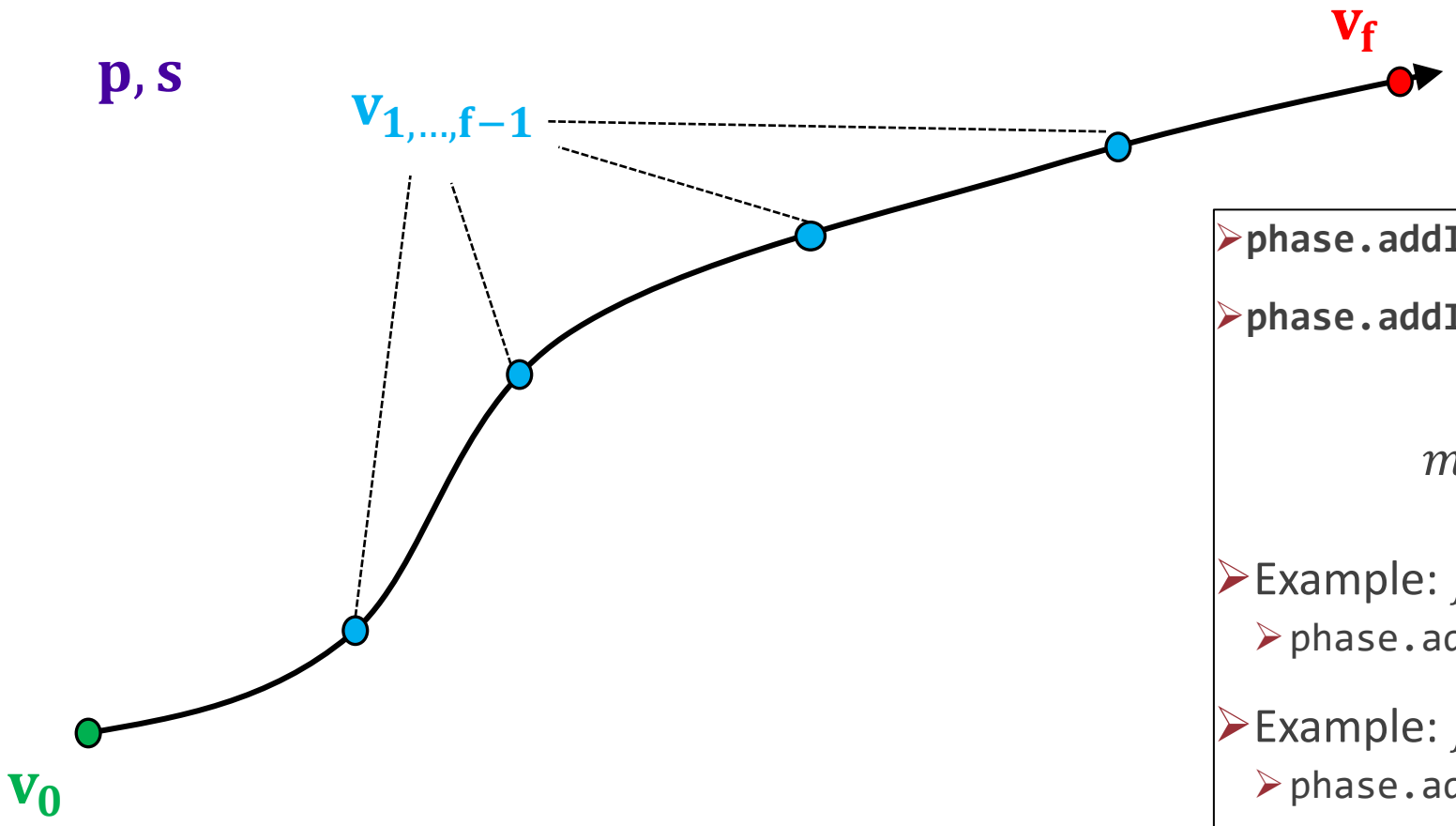


$\mathbf{i_v} = [0\,, 1\,, 2\,, 3\,]$ $\quad \mathbf{i_p} = [0]$ $\quad \mathbf{i_s} = [0]$

$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}]$ $\quad \mathbf{p} = [p_0\,]$ $\quad \mathbf{s} = [s_0\,]$

- Most constraints only need variables from one group
- **`phase.add#######(PhaseRegion,Func,Vvars)`**
- Example: $\mathbf{f}([x_0, t, u_0]) = \mathbf{0}$ at first state in trajectory
  - `phase.addEqualCon("First", f, [0,2,3])`
- Example: $\mathbf{f}([t, x_0\,]) < \mathbf{0}$ at last state in trajectory
  - `phase.addInequalCon("Last", f, [2,0])`
- Example: min: $f([t])$ at last state in trajectory
  - `phase.addStateObjective("Last", f, [2])`
- Example: $f([p_0]) = 0$
  - `phase.addEqualCon("ODEParams",f,[0])`
- Example: $f([s_0]) < 0$
  - `phase.addEqualCon("StaticParams",f,[0])`

Adding Integral Objectives



$\mathbf{i_v} = [0\,, 1\,, 2\,, 3\,]$ $\quad \mathbf{i_p} = [0]$ $\quad \mathbf{i_s} = [0]$

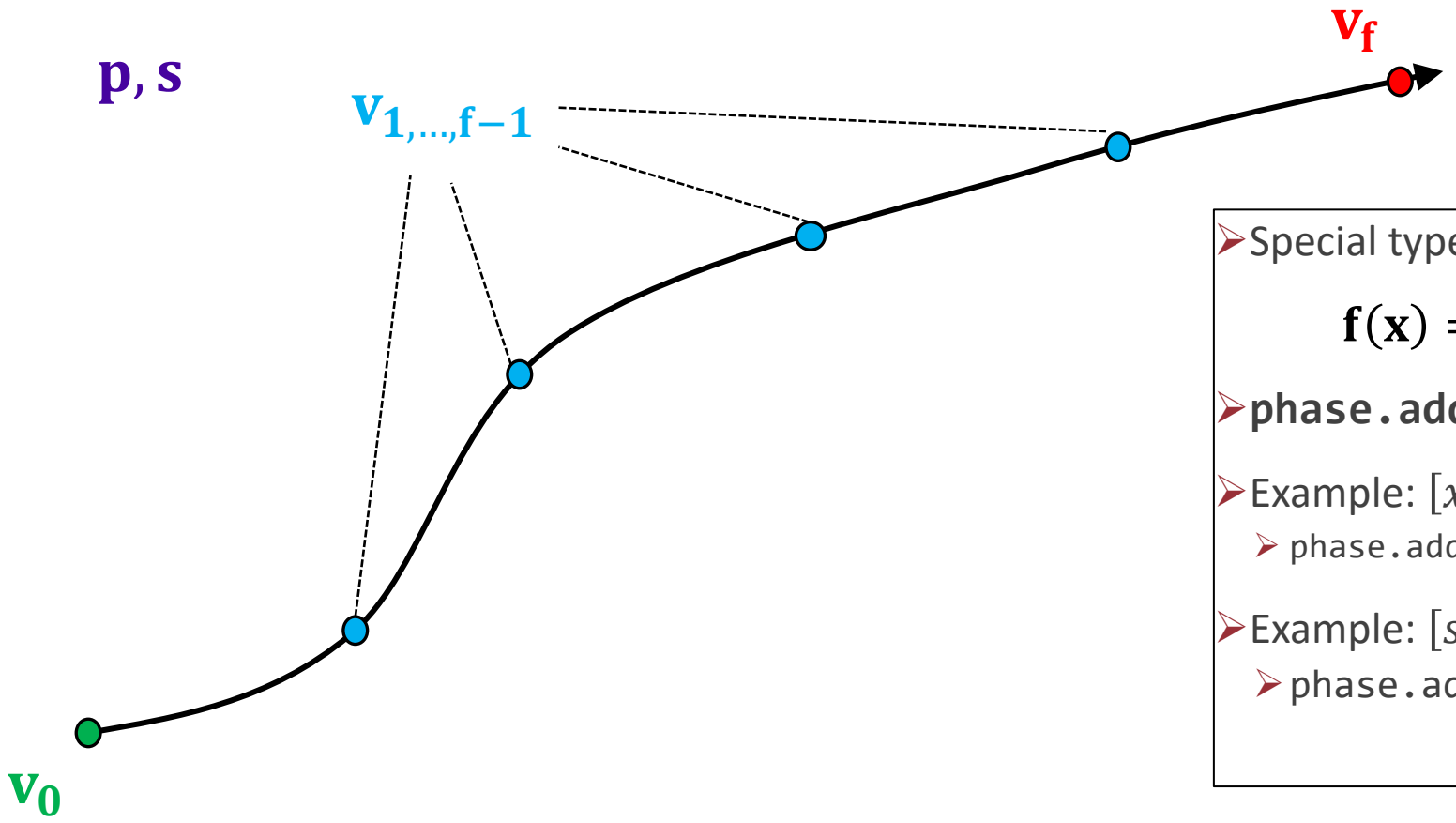
$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}]$ $\quad \mathbf{p} = [p_0\,]$ $\quad \mathbf{s} = [s_0\,]$

- `phase.addIntegralObjective(Func,Vvars,Pvars,Svars)`
- `phase.addIntegralObjective(Func,Vvars)`

$$min: \int_{t_0}^{t_f} f([\mathbf{v}, \mathbf{s}, \mathbf{p}])\, dt$$

- Example: $f([x_0, x_1, u_0, s_0])$
  - `phase.addIntegralObjective(f,[0,1,3],[],[0])`
- Example: $f([x_0, u_0\,])$
  - `phase.addIntegralObjective(f,[0,3])`

Adding Boundary Value Constraints



- Special type of equality constraint

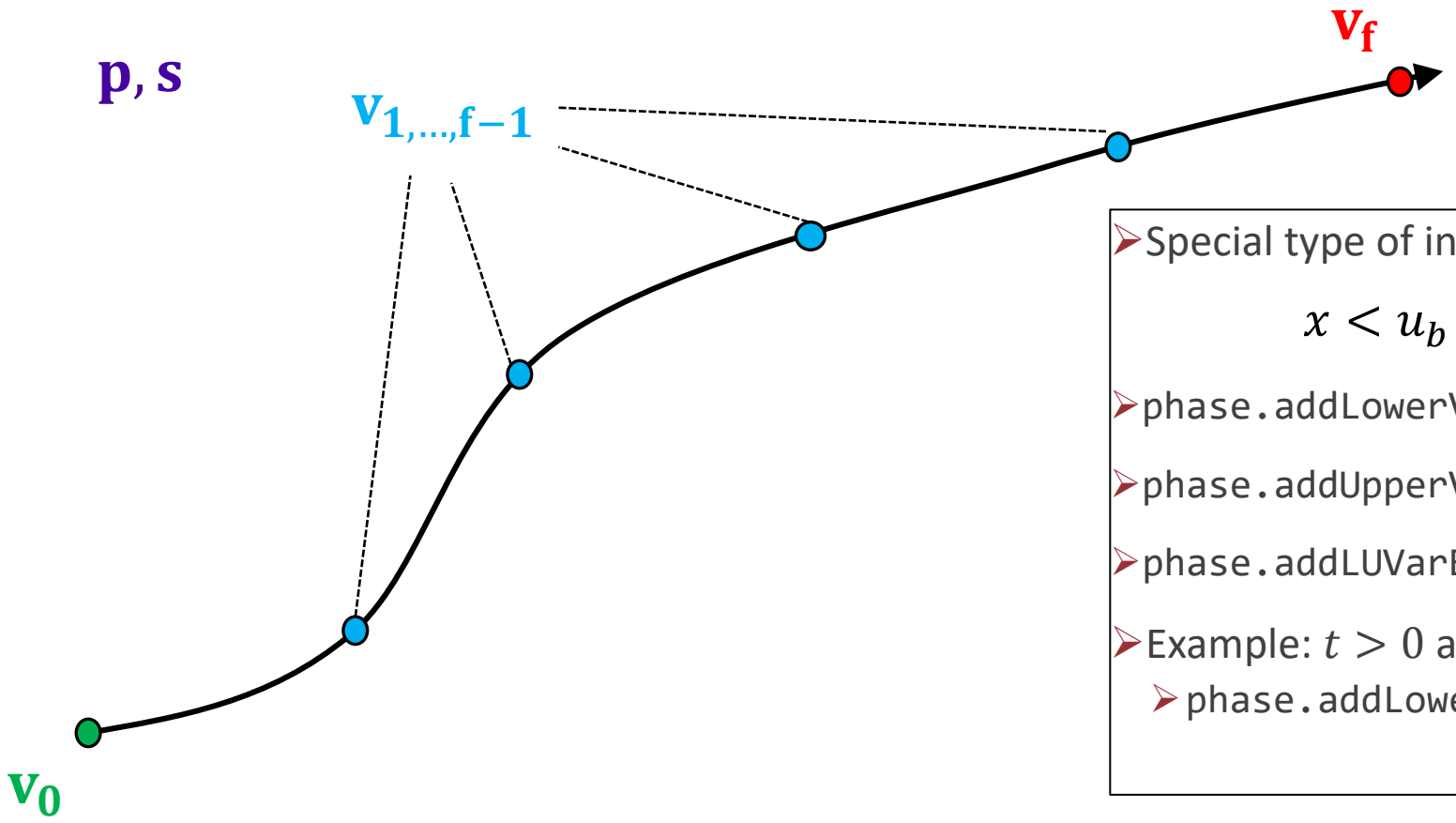
$$\mathbf{f(x) = x - x_c = 0}$$

- `phase.addBoundaryValue(PhaseRegion,Vars,Vals)`
- Example: $[x_0, x_1, t] = [1.5, 2.3, 0.0]$ at first state
  - `phase.addBoundaryValue("First",[0,1,2] , [1.5,2.3,0.0])`
- Example: $[s_0\ ] = [1.0]$
  - `phase.addBoundaryValue("StaticParams",[0],[1.0])`

$\mathbf{i_v} = [0\ , 1\ , 2\ , 3\ ]$ $\quad$ $\mathbf{i_p} = [0]$ $\quad$ $\mathbf{i_s} = [0]$

$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}]$ $\quad$ $\mathbf{p} = [p_0\ ]$ $\quad$ $\mathbf{s} = [s_0\ ]$

Adding Variable Bounds



- Special type of inequality constraints

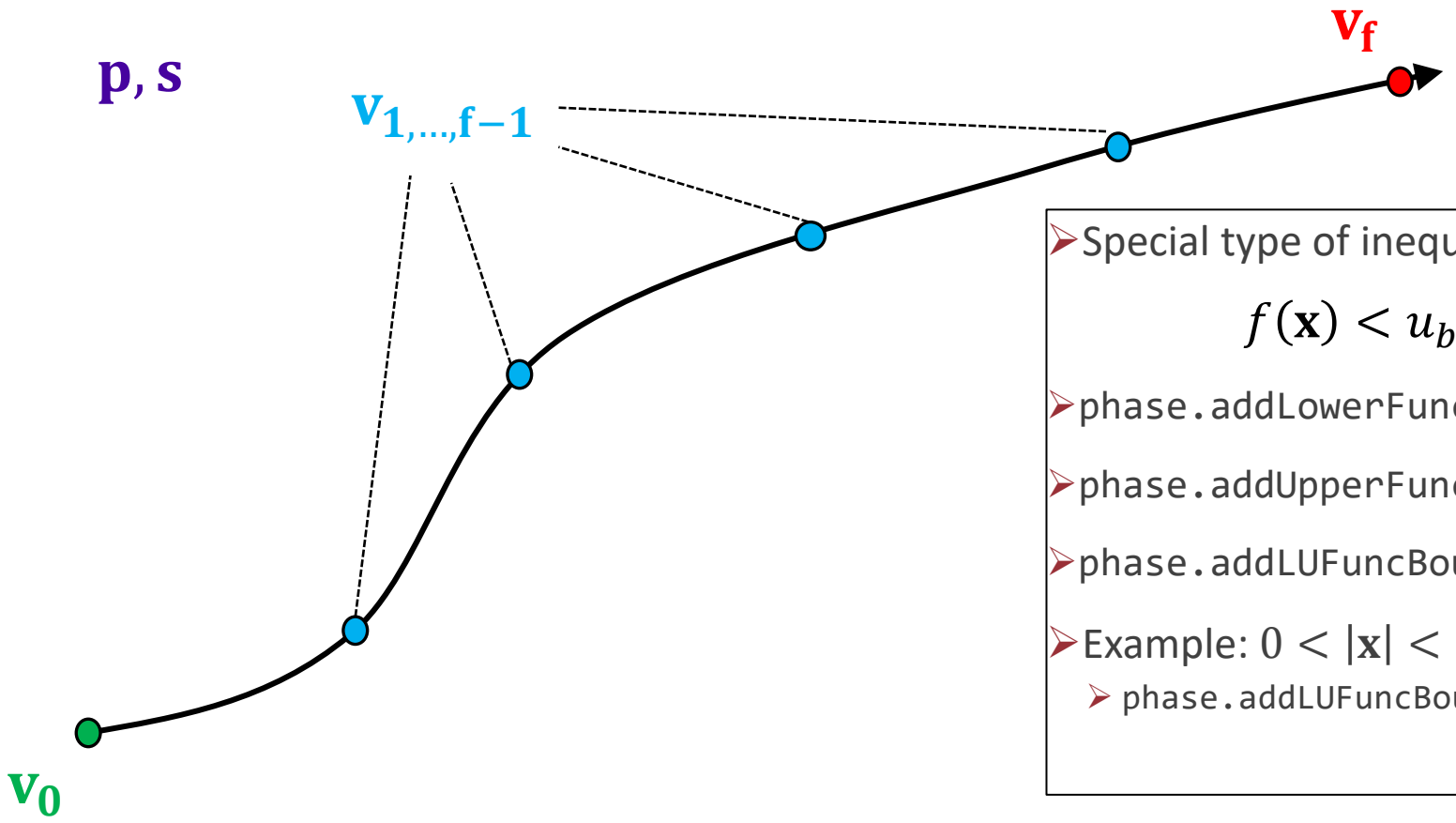
$$x < u_b \ , \qquad x > l_b \ , \qquad l_b < x < u_b$$

- `phase.addLowerVarBound(PhaseRegion,var,lbound)`
- `phase.addUpperVarBound(PhaseRegion,var,bound)`
- `phase.addLUVarBound(PhaseRegion,var,lbound,ubound)`
- Example: $t > 0$ at first state
  - `phase.addLowerVarBound("Front",2,0.0)`

$$\mathbf{i_v} = [0\ , 1\ , 2\ , 3\ ] \qquad \mathbf{i_p} = [0] \qquad \mathbf{i_s} = [0]$$

$$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}] \qquad \mathbf{p} = [p_0\ ] \qquad \mathbf{s} = [s_0\ ]$$

Adding Function Bounds



- Special type of inequality constraints

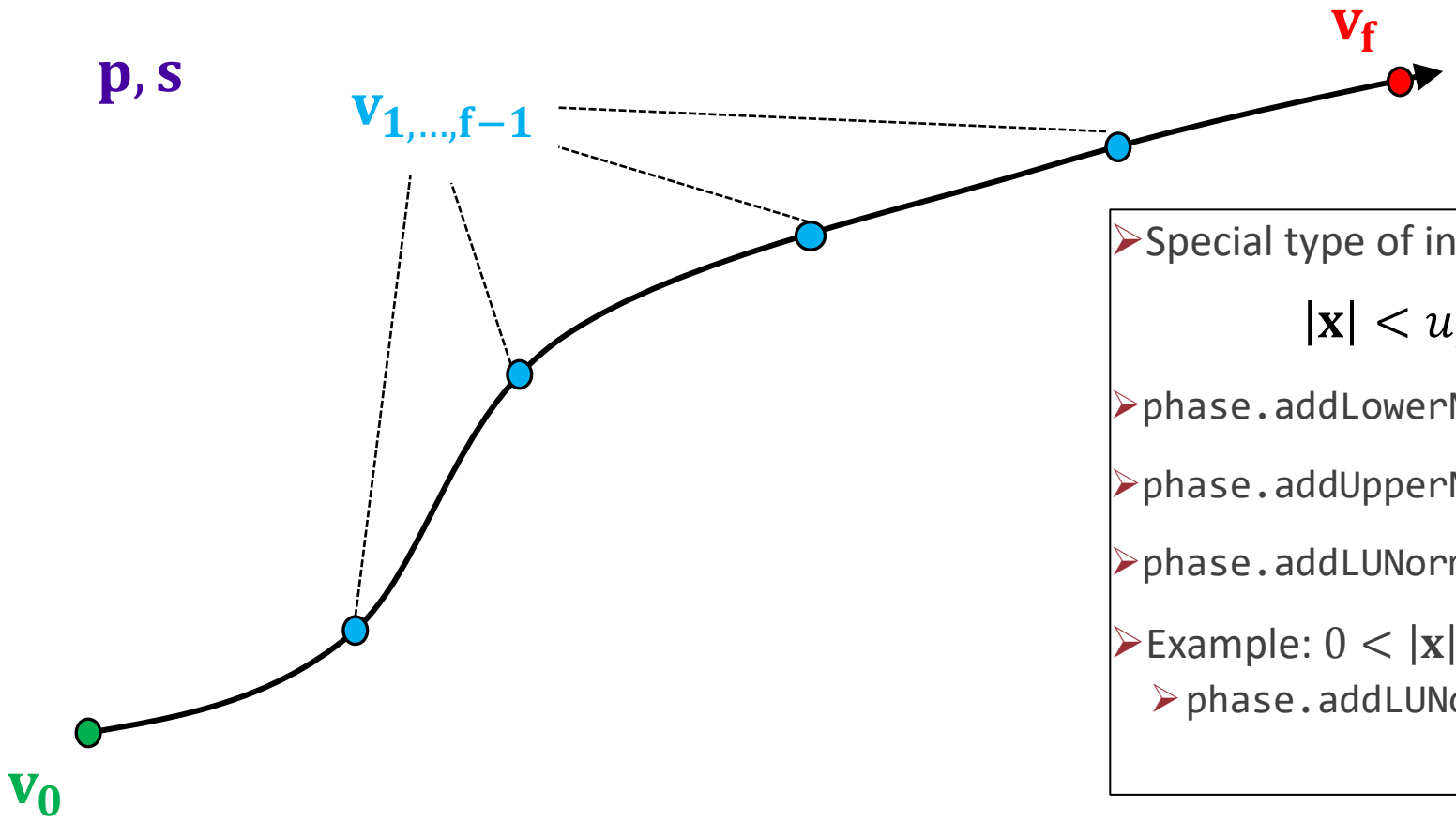
$$f(\mathbf{x}) < u_b \ , \qquad f(\mathbf{x}) > l_b \ , \qquad l_b < f(\mathbf{x}) < u_b$$

- `phase.addLowerFuncBound(PhaseRegion,Func, Vars,lbound)`
- `phase.addUpperFuncBound(PhaseRegion,Func, Vars,ubound)`
- `phase.addLUFuncBound(PhaseRegion,Func,Vars,lbound,ubound)`
- Example: $0 < |\mathbf{x}| < 1$ at all states
  - `phase.addLUFuncBound("Path",Args(2).norm(),[0,1],0.0,1.0)`

$$\mathbf{i_v} = [0\,, 1\,, 2\,, 3\ ] \qquad \mathbf{i_p} = [0] \qquad \mathbf{i_s} = [0]$$

$$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}] \qquad \mathbf{p} = [p_0\ ] \qquad \mathbf{s} = [s_0\ ]$$

Adding Norm Bounds



- Special type of inequality constraints

$$|\mathbf{x}| < u_b \ , \qquad |\mathbf{x}| > l_b \ , \qquad l_b < |\mathbf{x}| < u_b$$

- `phase.addLowerNormBound(PhaseRegion, Vars,lbound)`
- `phase.addUpperNormBound(PhaseRegion,Vars,ubound)`
- `phase.addLUNormBound(PhaseRegion, Vars,lbound, ubound)`
- Example: $0 < |\mathbf{x}| < 1$ at all states
  - `phase.addLUNormBound("Path",[0,1], 0.0,1.0)`

$$\mathbf{i_v} = [0\ , 1\ , 2\ , 3\ ] \qquad \mathbf{i_p} = [0] \qquad \mathbf{i_s} = [0]$$

$$\mathbf{v} = [x_0, x_1, t, u_0] = [\mathbf{x}, t, \mathbf{u}] \qquad \mathbf{p} = [p_0\ ] \qquad \mathbf{s} = [s_0\ ]$$

Solving and Optimizing

- Invoke optimizer using phase/ocp object
  - `phase.optimize()`
    - Optimize objective subject to constraints
  - `phase.solve()`
    - Solve just the constraints
  - `phase.solve_optimize()`
    - Calls .solve then .optimize
  - `phase.solve_optimize_solve()`
    - Same as above but calls solve if optimize fails
  - `phase.optimize_solve()`
    - Call optimize then solve if optimize fails
- Retrieve trajectory from phase
  - `phase.returnTraj()`
- All calls return a convergence flag

```
ode = ShuttleReentry()

phase = ode.phase("LGL3",TrajIG,40)

phase.addBoundaryValue("Front",range(0,6),TrajIG[0][0:6])
phase.addLUVarBound("Path",6,np.deg2rad(-90.0),np.deg2rad(90.0))
phase.addLUVarBound("Path",7,np.deg2rad(-90.0),np.deg2rad(1.0))
phase.addUpperDeltaTimeBound(tmax,1.0)
phase.addBoundaryValue("Back" ,[0,2,3],[htf,vtf,gammatf])
phase.addDeltaVarObjective(1,-1.0)

## IG is bad, solve first before optimize
flag1 = phase.solve_optimize()
Traj1 = phase.returnTraj()

## Add in Heating Rate Constraint, scale so rhs is order 1
phase.addUpperFuncBound("Path",QFunc(),[0,2,6],Qlimit)
flag2 = phase.optimize()
Traj2 = phase.returnTraj()

phase.addUpperFuncBound("Path",QFunc(),[0,2,6],WayToLow)
flag3 = phase.optimize()
Traj3 = phase.returnTraj()

print(flag1)
print(flag2)
print(flag3)
```

```
ConvergenceFlags.CONVERGED
ConvergenceFlags.CONVERGED
ConvergenceFlags.NOTCONVERGED
```

Optimizer Scroll

- Prints optimization progress
- Color coded convergence criteria

```
================================================================
 ____  ____  ____  ____  ____  ____  ____
||P ||||S ||||I ||||O ||||P ||||T ||||  ||
||__||||__||||__||||__||||__||||__||||__||
|/__\||/__\||/__\||/__\||/__\||/__\||/__\|

      Parallel Sparse Interior-point Optimizer
================================================================
Problem Statistics

 Primal Variables         : 123
 Equality Constraints     : 101
 Inequality Constraints   : 0

 KKT-Matrix DIM (P+E+2*I) : 224
 KKT-Matrix NNZs          : 1863
 KKT-Matrix NNZ%          : 3.712930%

================================================================
Beginning: PSIOPT
Beginning: KKT-Matrix Analysis
 LDLT Factor NNZs      : 3793
 LDLT Factor FLOPs     : 0 MFLOPs
 Analysis/Reorder Time : 3.209 ms
Finished : KKT-Matrix Analysis
Beginning: Optimization Algorithm
================================================================================================================================
|Iter| Mu Val | Prim Obj |  Bar Obj |  KKT Inf |  Bar Inf | ECons Inf| ICons Inf| AlphaP | AlphaD |LS| PPS |HF| HPert |
|0   |1.00e-03| 1.250e+00| 0.000e+00|6.9469e-02|0.0000e+00|1.3265e-02|0.0000e+00|1.00e+00|1.00e+00| 0|    0| 0|0.0e+00|
|1   |1.00e-03| 7.616e-01| 0.000e+00|1.6594e+00|0.0000e+00|4.2633e-14|0.0000e+00|1.00e+00|1.00e+00| 0|    0| 0|0.0e+00|
|2   |1.00e-03| 7.616e-01| 0.000e+00|4.4409e-16|0.0000e+00|1.9895e-13|0.0000e+00|1.00e+00|1.00e+00| 0|    0| 0|0.0e+00|

Optimal Solution Found
 Iterations : 3
 Prim Obj   : 7.61594166e-01
 KKT Inf    : 4.44089210e-16
 Bar Inf    : 0.00000000e+00
 ECons Inf  : 1.98951966e-13
 ICons Inf  : 0.00000000e+00

 NLP Function Evaluation Time :      1.025 ms      0.342 ms/iter
 KKT Matrix Factor/Solve Time :      0.343 ms      0.114 ms/iter
 Console Print Time           :      0.851 ms      0.284 ms/iter
 Total Time (NLP+KKT+Print)   :      2.219 ms      0.740 ms/iter

Finished : Optimization Algorithm
 PSIOPT Total Time : 9.027 ms
Finished : PSIOPT
================================================================
```

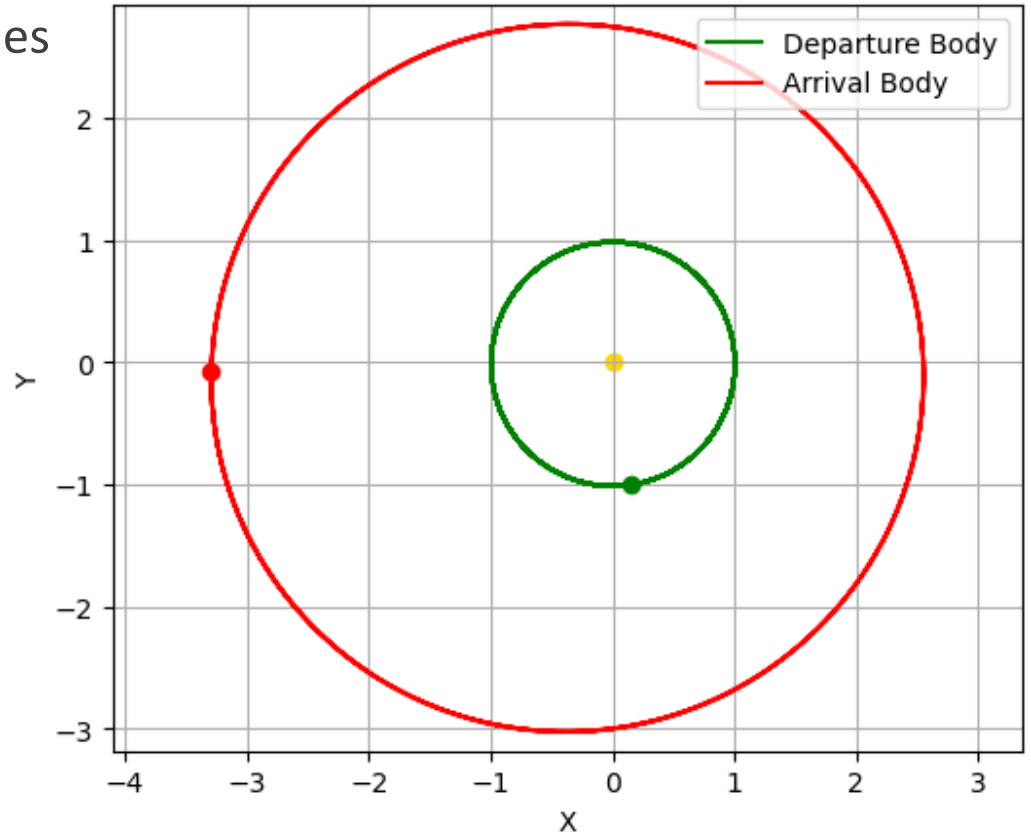
Optimizer Settings

- Each phase has an optimizer instance as member variable
  - Use to modify optimizer related settings
  - See PSIOPT tutorial for more details on settings

```
phase = ode.phase("LGL5",IG,32)
phase.addBoundaryValue("Front",range(0,3),[0,1,0])
phase.addUpperVarBound("Path",0,1/9)
phase.addIntegralObjective((Args(1)[0]**2)/2,[3])
phase.addBoundaryValue("Back",range(0,3) ,[0,-1,1])

phase.optimizer.set_OptLSMode("L1")
phase.optimizer.set_KKTtol(1.0e-10)
phase.optimizer.set_PrintLevel(0)

phase.optimize()

Traj = phase.returnTraj()
```

Practical Example

- Compute Low-thrust minimum mass transfer between two bodies
  - Bounded departure v_∞: $\mathbf{r} - \mathbf{r_d}(t) = \mathbf{0}\,,\ |\mathbf{v} - \mathbf{v_d}(t)| < v_\infty$
  - Rendezvous with target: $\mathbf{r} - \mathbf{r_a}(t) = \mathbf{0}, \mathbf{v} - \mathbf{v_a}(t) = \mathbf{0}$
  - Bound Throttle Vector: $0.001 < |\mathbf{u}| < 1$
  - Mass optimal: $\min: \int_{t_0}^{t_f} |\mathbf{u}| dt$
- Requires:
  - TwoBody
  - TwoBodyLTODE



Practical Example

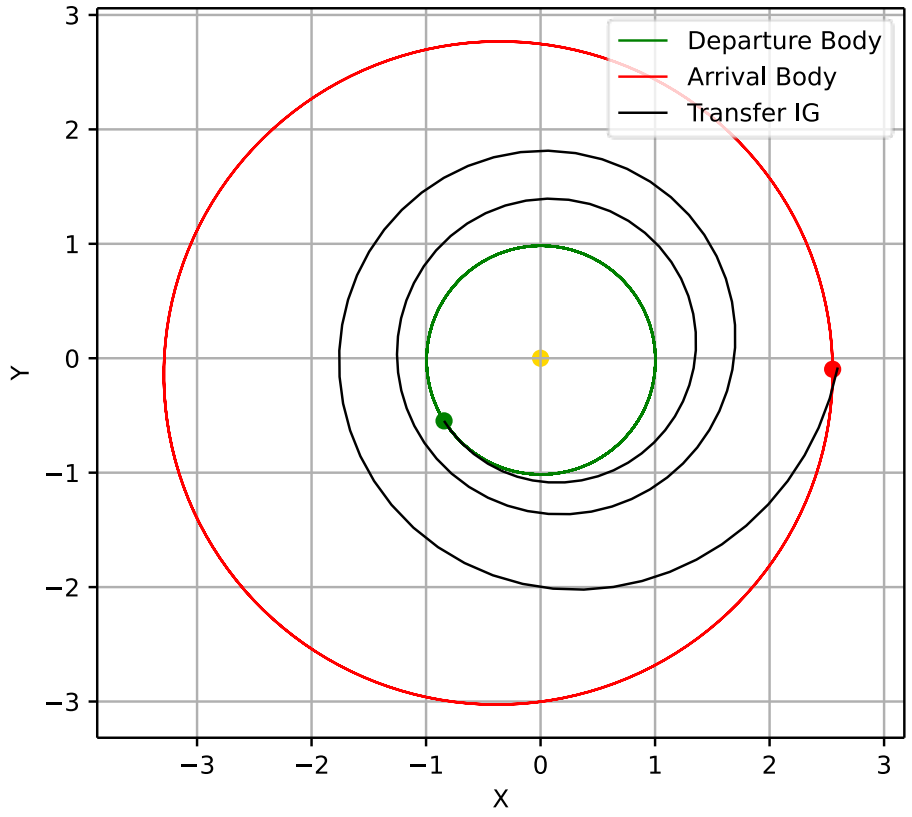
- Setup almost same as integrator example

```
tmax = 100.0 ## Max epoch time
mu = 1.0
alpha = .02  ## low-thrust accelleration
ig_throttle = 0.5  ## throttle level for our ig
vinf = .066

## Initial conditions of departare arrival bodies at initial epoch
dep_ig = [ 1.567e-01, -1.004e+00,  4.690e-05, 9.722e-01,1.505e-01, -2.510e-05,  0]
arr_ig = [-3.28938729, -0.07394068,  0.09236258, -0.01214116, -0.51471216,  0.02445342,0.]



tbode = TwoBody(mu)
tbint = tbode.integrator(.1)

ltode = TwoBodyLT(mu, alpha)
ltig_int = ltode.integrator(.1,TanLaw(ig_throttle),range(0,6))

################################################################################

## Integrate out ics to generate body data
dep_body_traj = tbint.integrate_dense(dep_ig,tmax,20000)
arr_body_traj = tbint.integrate_dense(arr_ig,tmax,20000)

## Load into table for vector functions
dep_body_tab = vf.InterpTable1D(dep_body_traj,tvar = 6)
arr_body_tab = vf.InterpTable1D(arr_body_traj,tvar = 6)
################################################################################
```

Practical Example

- Generate initial guess using modified search

```
t0s = np.linspace(4*np.pi,10*np.pi,1000)

X0t0U0s = []

for t0 in t0s:
    X0t0U0 = np.zeros(10)
    X0t0U0[0:6] = dep_body_tab(t0)

    v0dir =  X0t0U0[3:6]/np.linalg.norm(X0t0U0[3:6])
    X0t0U0[3:6]+=v0dir*vinf  # add vinf to IC

    X0t0U0[6]=t0
    X0t0U0s.append(X0t0U0)

tfs = [tmax]*len(t0s)

nthreads = 8
trajs_events = ltig_int.integrate_dense_parallel(X0t0U0s,tfs,
                                                 [(CrossEvent(arr_body_tab),0,True)],
                                                 nthreads)
trajs = [t[0] for t in trajs_events]

def keyfunc(traj):
    Rf = traj[-1][0:3]
    Rftarg = arr_body_tab(traj[-1][6])[0:3]
    return np.linalg.norm(Rf-Rftarg)

## sort for min distance to target at crossing
trajs.sort(key=keyfunc)

traj = trajs[0]
```



Practical Example

- Setup phase object and add constraints/objectives

```
transcription = "LGL3"
nsegments = 128

phase = ltode.phase(transcription,traj,nsegments)

phase.addEqualCon("First",PosCon(dep_body_tab),[0,1,2,6])
phase.addUpperFuncBound("First",VinfFunc(dep_body_tab),[3,4,5,6],vinf)

phase.addLUNormBound("Path",range(7,10),.001,1.0,1)

phase.addEqualCon("Last",RendCon(arr_body_tab),range(0,7))

phase.addIntegralObjective(Args(3).norm(),range(7,10))

phase.addLowerVarBound("First",6,traj[0][6]-1.1)
phase.addUpperVarBound("First",6,traj[0][6]+1.1)

phase.addUpperDeltaTimeBound((traj[-1][6]-traj[0][6])*1.5)
phase.addLowerDeltaTimeBound(1.0)

phase.addLowerVarBound("Last",6,0)
phase.addUpperVarBound("Last",6,tmax)

phase.optimize_solve()


traj = phase.returnTraj()
```

Practical Example

- Setup phase object and add constraints/objectives

$$\mathbf{r} - \mathbf{r}_d(t) = \mathbf{0}$$

```
def PosCon(rvtab):
    Rt = Args(4)
    R = Rt.head(3)
    t = Rt[3]
    return R-rvtab(t).head(3)
```

$$\mathbf{i_v} = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$$

$$\mathbf{v} = [r_x, r_y, r_z, v_x, v_y, v_z, t, u_x, u_y, u_z] = [\mathbf{x}, t, \mathbf{u}]$$

```
transcription = "LGL3"
nsegments = 128

phase = ltode.phase(transcription,traj,nsegments)

phase.addEqualCon("First",PosCon(dep_body_tab),[0,1,2,6])
phase.addUpperFuncBound("First",VinfFunc(dep_body_tab),[3,4,5,6],vinf)

phase.addLUNormBound("Path",range(7,10),.001,1.0,1)

phase.addEqualCon("Last",RendCon(arr_body_tab),range(0,7))

phase.addIntegralObjective(Args(3).norm(),range(7,10))

phase.addLowerVarBound("First",6,traj[0][6]-1.1)
phase.addUpperVarBound("First",6,traj[0][6]+1.1)

phase.addUpperDeltaTimeBound((traj[-1][6]-traj[0][6])*1.5)
phase.addLowerDeltaTimeBound(1.0)

phase.addLowerVarBound("Last",6,0)
phase.addUpperVarBound("Last",6,tmax)

phase.optimize_solve()


traj = phase.returnTraj()
```

Practical Example

- Setup phase object and add constraints/objectives

$$|\mathbf{v} - \mathbf{v_d}(t)| < v_\infty$$

```
def VinfFunc(rvtab):
    Vt = Args(4)
    V = Vt.head(3)
    t = Vt[3]
    return (V-rvtab(t).segment(3,3)).norm()
```

$$\mathbf{i_v} = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$$

$$\mathbf{v} = [r_x, r_y, r_z, v_x, v_y, v_z, t, u_x, u_y, u_z] = [\mathbf{x}, t, \mathbf{u}]$$

```
transcription = "LGL3"
nsegments = 128

phase = ltode.phase(transcription,traj,nsegments)

phase.addEqualCon("First",PosCon(dep_body_tab),[0,1,2,6])
phase.addUpperFuncBound("First",VinfFunc(dep_body_tab),[3,4,5,6],vinf)

phase.addLUNormBound("Path",range(7,10),.001,1.0,1)

phase.addEqualCon("Last",RendCon(arr_body_tab),range(0,7))

phase.addIntegralObjective(Args(3).norm(),range(7,10))

phase.addLowerVarBound("First",6,traj[0][6]-1.1)
phase.addUpperVarBound("First",6,traj[0][6]+1.1)

phase.addUpperDeltaTimeBound((traj[-1][6]-traj[0][6])*1.5)
phase.addLowerDeltaTimeBound(1.0)

phase.addLowerVarBound("Last",6,0)
phase.addUpperVarBound("Last",6,tmax)

phase.optimize_solve()


traj = phase.returnTraj()
```

Practical Example

- Setup phase object and add constraints/objectives

$$0.001 < |\mathbf{u}| < 1$$

$$\mathbf{i_v} = [0\,,1\,,2\,,3\,,4\,,5\,,6\,,7\,,8\,,9\,]$$

$$\mathbf{v} = \left[r_x, r_y, r_z, v_x, v_y, v_z\,, t, u_x, u_y, u_z\right] = [\mathbf{x}, t, \mathbf{u}]$$

```
transcription = "LGL3"
nsegments = 128

phase = ltode.phase(transcription,traj,nsegments)

phase.addEqualCon("First",PosCon(dep_body_tab),[0,1,2,6])
phase.addUpperFuncBound("First",VinfFunc(dep_body_tab),[3,4,5,6],vinf)

phase.addLUNormBound("Path",range(7,10),.001,1.0,1)

phase.addEqualCon("Last",RendCon(arr_body_tab),range(0,7))

phase.addIntegralObjective(Args(3).norm(),range(7,10))

phase.addLowerVarBound("First",6,traj[0][6]-1.1)
phase.addUpperVarBound("First",6,traj[0][6]+1.1)

phase.addUpperDeltaTimeBound((traj[-1][6]-traj[0][6])*1.5)
phase.addLowerDeltaTimeBound(1.0)

phase.addLowerVarBound("Last",6,0)
phase.addUpperVarBound("Last",6,tmax)

phase.optimize_solve()


traj = phase.returnTraj()
```

Practical Example

- Setup phase object and add constraints/objectives

$$\mathbf{r} - \mathbf{r_a}(t) = \mathbf{0}, \mathbf{v} - \mathbf{v_a}(t) = \mathbf{0}$$

```
def RendCon(rvtab):
    RVt = Args(7)
    RV = RVt.head(6)
    t = RVt[6]
    return RV-rvtab(t)
```

$$\mathbf{i_v} = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$$

$$\mathbf{v} = [r_x, r_y, r_z, v_x, v_y, v_z, t, u_x, u_y, u_z] = [\mathbf{x}, t, \mathbf{u}]$$

```
transcription = "LGL3"
nsegments = 128

phase = ltode.phase(transcription,traj,nsegments)

phase.addEqualCon("First",PosCon(dep_body_tab),[0,1,2,6])
phase.addUpperFuncBound("First",VinfFunc(dep_body_tab),[3,4,5,6],vinf)

phase.addLUNormBound("Path",range(7,10),.001,1.0,1)

phase.addEqualCon("Last",RendCon(arr_body_tab),range(0,7))

phase.addIntegralObjective(Args(3).norm(),range(7,10))

phase.addLowerVarBound("First",6,traj[0][6]-1.1)
phase.addUpperVarBound("First",6,traj[0][6]+1.1)

phase.addUpperDeltaTimeBound((traj[-1][6]-traj[0][6])*1.5)
phase.addLowerDeltaTimeBound(1.0)

phase.addLowerVarBound("Last",6,0)
phase.addUpperVarBound("Last",6,tmax)

phase.optimize_solve()


traj = phase.returnTraj()
```

Practical Example

- Setup phase object and add constraints/objectives

$$\int_{t_0}^{t_f} |\mathbf{u}| dt$$

$$\mathbf{i_v} = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$$

$$\mathbf{v} = [r_x, r_y, r_z, v_x, v_y, v_z, t, u_x, u_y, u_z] = [\mathbf{x}, t, \mathbf{u}]$$

```
transcription = "LGL3"
nsegments = 128

phase = ltode.phase(transcription,traj,nsegments)

phase.addEqualCon("First",PosCon(dep_body_tab),[0,1,2,6])
phase.addUpperFuncBound("First",VinfFunc(dep_body_tab),[3,4,5,6],vinf)

phase.addLUNormBound("Path",range(7,10),.001,1.0,1)

phase.addEqualCon("Last",RendCon(arr_body_tab),range(0,7))

phase.addIntegralObjective(Args(3).norm(),range(7,10))

phase.addLowerVarBound("First",6,traj[0][6]-1.1)
phase.addUpperVarBound("First",6,traj[0][6]+1.1)

phase.addUpperDeltaTimeBound((traj[-1][6]-traj[0][6])*1.5)
phase.addLowerDeltaTimeBound(1.0)

phase.addLowerVarBound("Last",6,0)
phase.addUpperVarBound("Last",6,tmax)

phase.optimize_solve()


traj = phase.returnTraj()
```

Practical Example

- Setup phase object and add constraints/objectives
- Place bounds on time to make problem well posed

$$\mathbf{i_v} = [0\,,1\,,2\,,3\,,4\,,5\,,6\,,7\,,8\,,9\,]$$

$$\mathbf{v} = \left[r_x, r_y, r_z, v_x, v_{y,}\, v_z\,, t, u_x, u_{y,}\, u_z\right] = [\mathbf{x}, t, \mathbf{u}]$$

```
transcription = "LGL3"
nsegments = 128

phase = ltode.phase(transcription,traj,nsegments)

phase.addEqualCon("First",PosCon(dep_body_tab),[0,1,2,6])
phase.addUpperFuncBound("First",VinfFunc(dep_body_tab),[3,4,5,6],vinf)

phase.addLUNormBound("Path",range(7,10),.001,1.0,1)

phase.addEqualCon("Last",RendCon(arr_body_tab),range(0,7))

phase.addIntegralObjective(Args(3).norm(),range(7,10))

phase.addLowerVarBound("First",6,traj[0][6]-1.1)
phase.addUpperVarBound("First",6,traj[0][6]+1.1)

phase.addUpperDeltaTimeBound((traj[-1][6]-traj[0][6])*1.5)
phase.addLowerDeltaTimeBound(1.0)

phase.addLowerVarBound("Last",6,0)
phase.addUpperVarBound("Last",6,tmax)

phase.optimize_solve()


traj = phase.returnTraj()
```

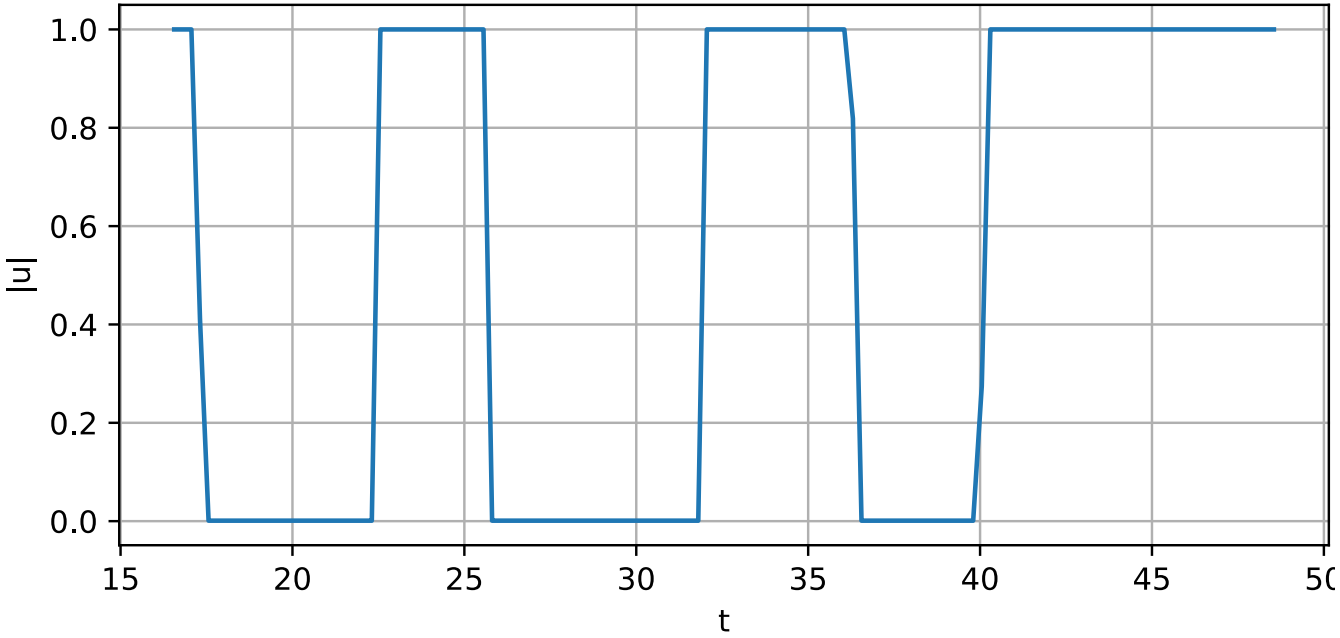
Practical Example

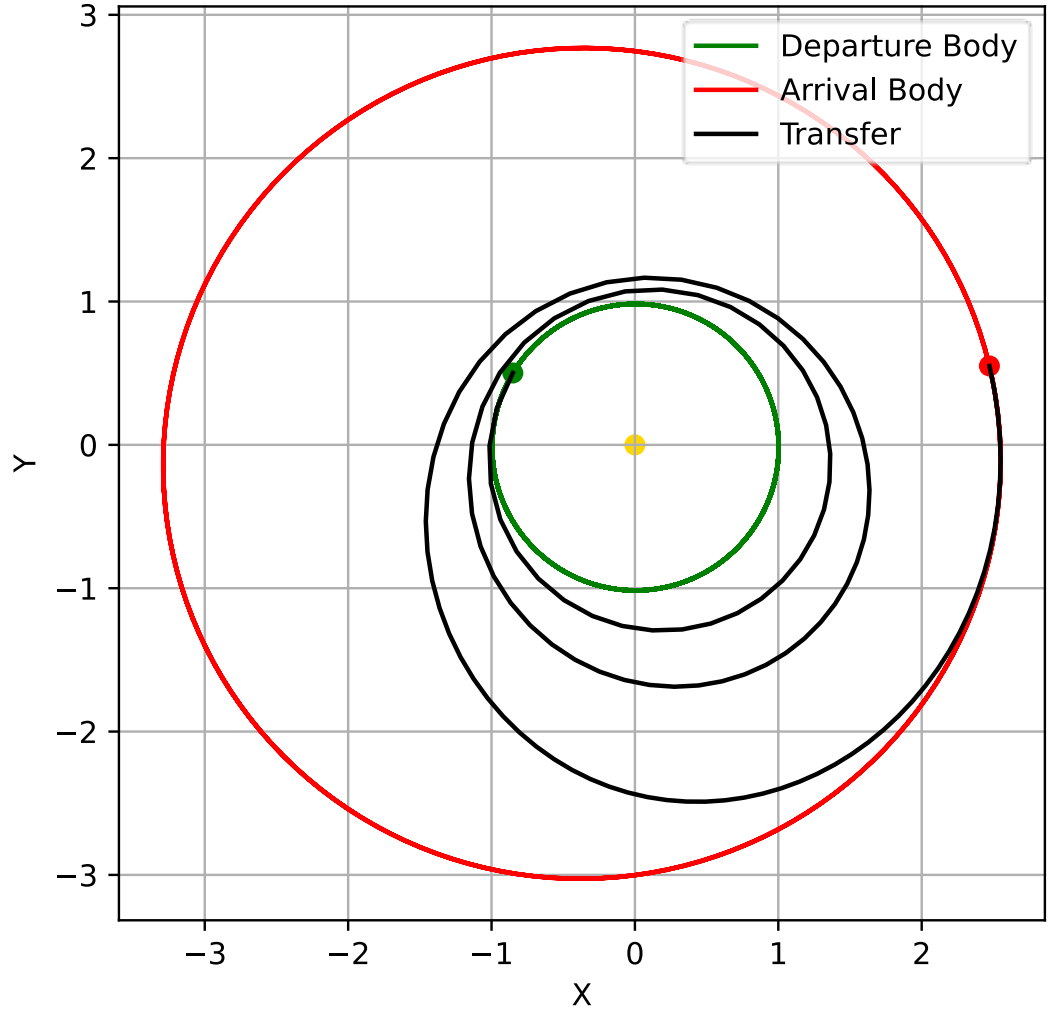
- Setup phase object and add constraints/objectives
- Optimize and retrieve trajectory

```
transcription = "LGL3"
nsegments = 128

phase = ltode.phase(transcription,traj,nsegments)

phase.addEqualCon("First",PosCon(dep_body_tab),[0,1,2,6])
phase.addUpperFuncBound("First",VinfFunc(dep_body_tab),[3,4,5,6],vinf)

phase.addLUNormBound("Path",range(7,10),.001,1.0,1)

phase.addEqualCon("Last",RendCon(arr_body_tab),range(0,7))

phase.addIntegralObjective(Args(3).norm(),range(7,10))

phase.addLowerVarBound("First",6,traj[0][6]-1.1)
phase.addUpperVarBound("First",6,traj[0][6]+1.1)

phase.addUpperDeltaTimeBound((traj[-1][6]-traj[0][6])*1.5)
phase.addLowerDeltaTimeBound(1.0)

phase.addLowerVarBound("Last",6,0)
phase.addUpperVarBound("Last",6,tmax)

phase.optimize_solve()


traj = phase.returnTraj()
```

Practical Example

- Optimizes solution in < 1s
- Get expected bang-off-bang control profile





Conclusion

- See tutorial online for more in-depth details

Tutorials - ASSET 0.4.0 documentation (alabamaasrl.github.io)