SHARC: Simulator for Hardware Architecture and Real-time Control

Paul K. Wintz[1] Yasin Sonmez[2] Paul Griffioen[2] Mingsheng Xu[1] Surim Oh[1]
Heiner Litz[1] Ricardo G. Sanfelice[1] Murat Arcak[2]

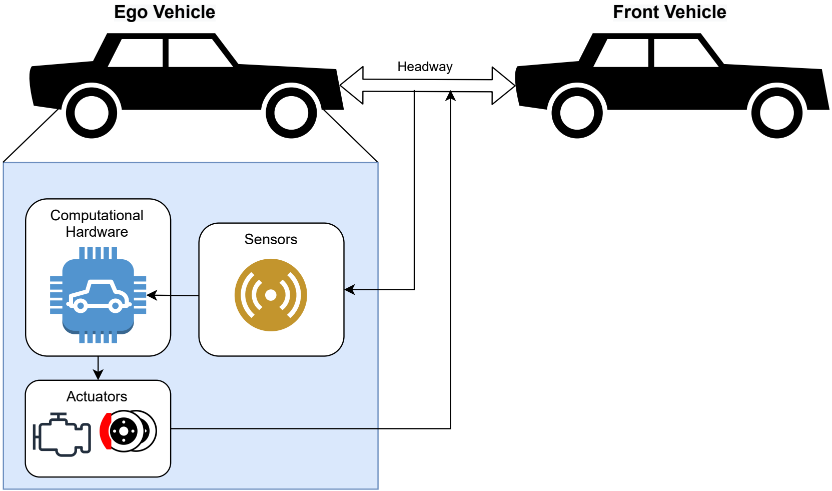
[1]University of California, Santa Cruz
[2]University of California, Berkeley

May, 2025

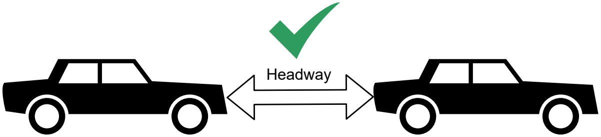
UC SANTA CRUZ | Baskin Engineering

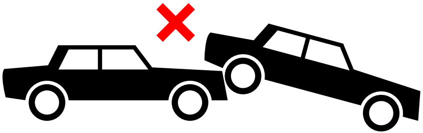
Berkeley

Motivating Example: Adaptive Cruise Control (ACC)



Motivating Example: Adaptive Cruise Control (ACC)





If no computational delays:

⟹ Guaranteed minimum headway

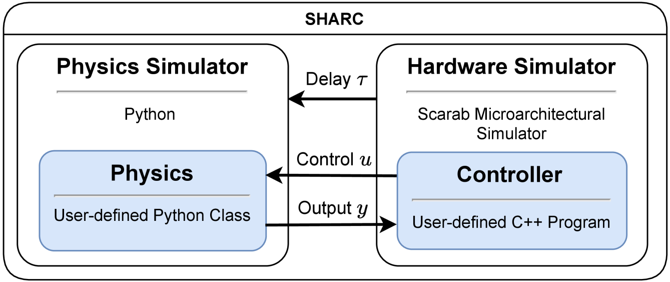
If computational delays:

⟹ ???

Computational delays depend on

- Control Algorithm, implementation, and parameters
- Computational hardware
- Current state and measurements
- Recent computations

SHARC: Simulator for Hardware Architecture and Real-time Control



SHARC

Physics Simulator

Python

Physics

User-defined Python Class

Delay τ

Control u

Output y

Hardware Simulator

Scarab Microarchitectural Simulator

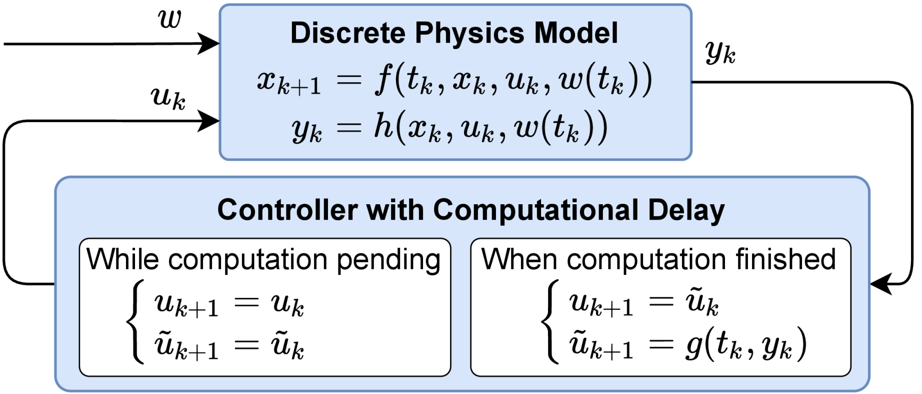
Controller

User-defined C++ Program

Features

- Uses same executable as would be deployed.
- Parallelized to shorten run times.
- Easy configuration via JSON files.
- Dockerized for easy setup.

Mathematical Model of Delayed Computations



Discrete Physics Model

$$x_{k+1} = f(t_k, x_k, u_k, w(t_k))$$
$$y_k = h(x_k, u_k, w(t_k))$$

Controller with Computational Delay

While computation pending

$$\begin{cases} u_{k+1} = u_k \\ \tilde{u}_{k+1} = \tilde{u}_k \end{cases}$$

When computation finished

$$\begin{cases} u_{k+1} = \tilde{u}_k \\ \tilde{u}_{k+1} = g(t_k, y_k) \end{cases}$$

Controller Execution Simulation

To estimate controller run time, we use the Scarab Microarchitectural Simulator.

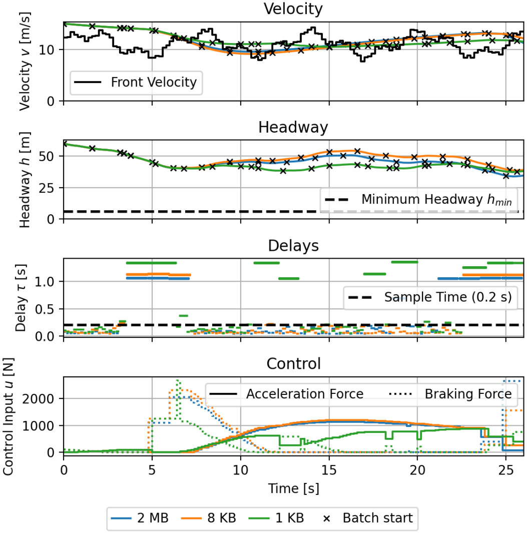
- Low level simulation of controller binary on CPU
- Simulates caching, branch prediction, pipelining, etc.
- Customizable processor parameters
  - Cache size
  - Clock speed
  - Architecture
- Provides detailed statistics.

ACC Example: Instruction Cache Size Comparison

Problem 1 (Linear MPC)

$$\begin{aligned} \text{minimize} \quad & |\text{velocity error}|^2 \\ & + |\text{control effort}|^2 \\ \text{subject to} \quad & \\ & \text{Linear System Dynamics} \\ & \text{Linear Safety Constraints} \end{aligned}$$

$\Longrightarrow$ Performance degrades if instruction cache is only 1 KB.



Example Pseudocode

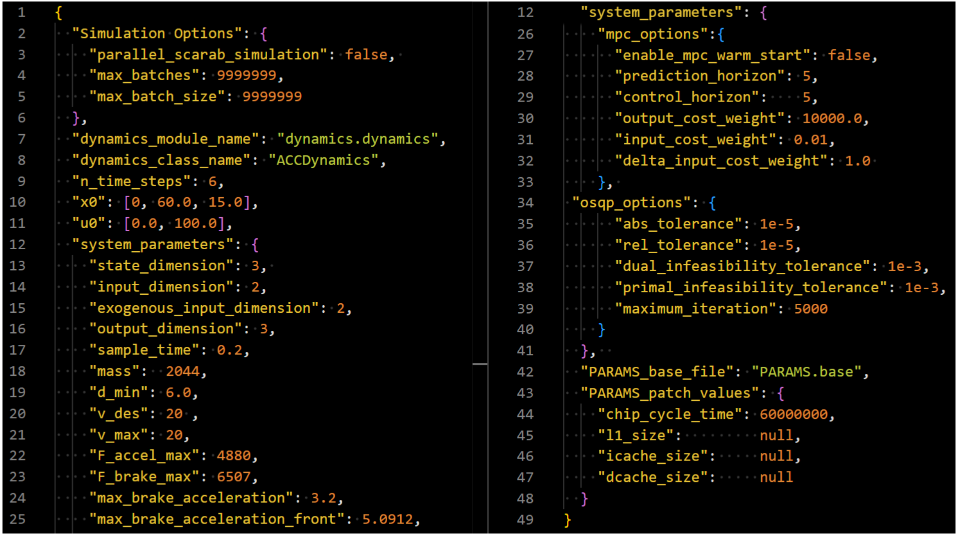
Physics Dynamics (Python interface)

```
class MyDynamics(Dynamics):
  def evolve_state(self, t0, x0, u, tf):
    return xf # Final state

  def get_output(self, x, u, w):
    return y

  def get_exogenous_input(self, t):
    return w
```

Controller (C++)

```
class MyController : Controller {
    void calculateControl(double t, Vec &y){
        return u;
    }
};
```

Configuration



```
{
  "Simulation Options": {
    "parallel_scarab_simulation": false, 
    "max_batches": 9999999,
    "max_batch_size": 9999999
  },
  "dynamics_module_name": "dynamics.dynamics",
  "dynamics_class_name": "ACCDynamics",
  "n_time_steps": 6,
  "x0": [0, 60.0, 15.0],
  "u0": [0.0, 100.0],
  "system_parameters": {
    "state_dimension": 3,
    "input_dimension": 2,
    "exogenous_input_dimension": 2,
    "output_dimension": 3,
    "sample_time": 0.2,
    "mass":  2044,
    "d_min": 6.0,
    "v_des": 20 ,
    "v_max": 20,
    "F_accel_max": 4880,
    "F_brake_max": 6507,
    "max_brake_acceleration": 3.2,
    "max_brake_acceleration_front": 5.0912,
```

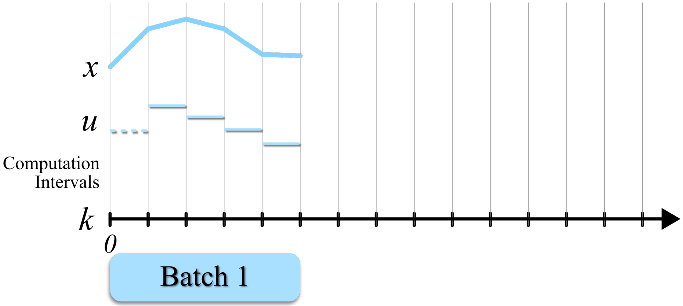
```
  "system_parameters": {
    "mpc_options":{
      "enable_mpc_warm_start": false,
      "prediction_horizon": 5,
      "control_horizon":    5,
      "output_cost_weight": 10000.0,
      "input_cost_weight": 0.01,
      "delta_input_cost_weight": 1.0
    },
 "osqp_options": {
      "abs_tolerance": 1e-5,
      "rel_tolerance": 1e-5,
      "dual_infeasibility_tolerance": 1e-3,
      "primal_infeasibility_tolerance": 1e-3,
      "maximum_iteration": 5000
    }
  },
  "PARAMS_base_file": "PARAMS.base",
  "PARAMS_patch_values": {
    "chip_cycle_time": 60000000,
    "l1_size":         null,
    "icache_size":     null,
    "dcache_size":     null
  }
}
```

SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

$\Longrightarrow$ Simulating slow controllers on a long time horizon can require several days

$\Longrightarrow$ We designed a parallelization scheme that allows many time steps to be run in parallel

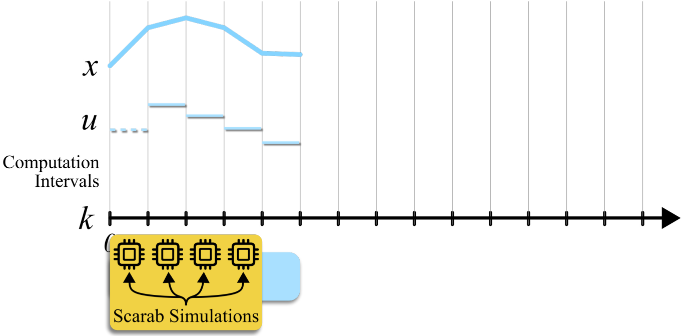


SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

$\Longrightarrow$ Simulating slow controllers on a long time horizon can require several days

$\Longrightarrow$ We designed a parallelization scheme that allows many time steps to be run in parallel

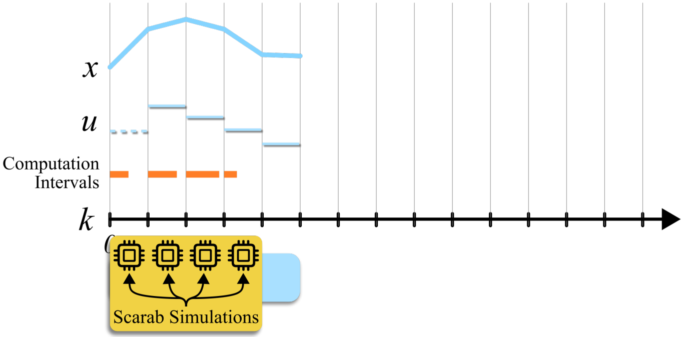


SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

⟹ Simulating slow controllers on a long time horizon can require several days

⟹ We designed a parallelization scheme that allows many time steps to be run in parallel

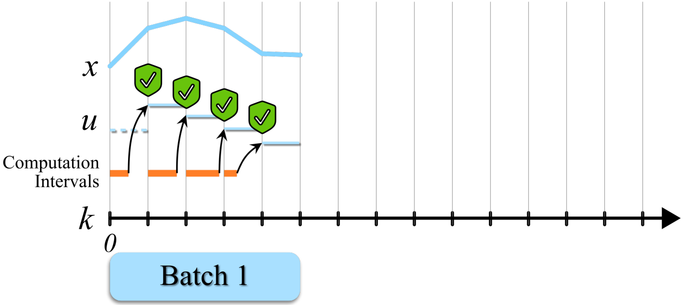


SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

$\Longrightarrow$ Simulating slow controllers on a long time horizon can require several days

$\Longrightarrow$ We designed a parallelization scheme that allows many time steps to be run in parallel

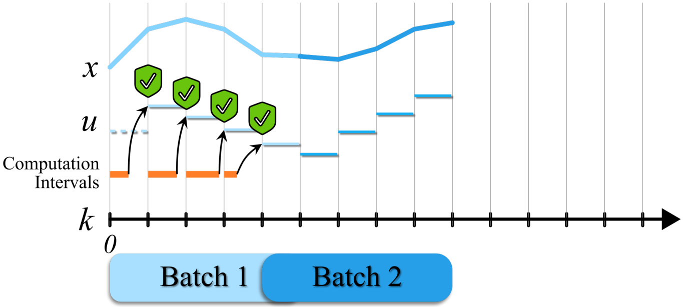


SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

$\Longrightarrow$ Simulating slow controllers on a long time horizon can require several days

$\Longrightarrow$ We designed a parallelization scheme that allows many time steps to be run in parallel

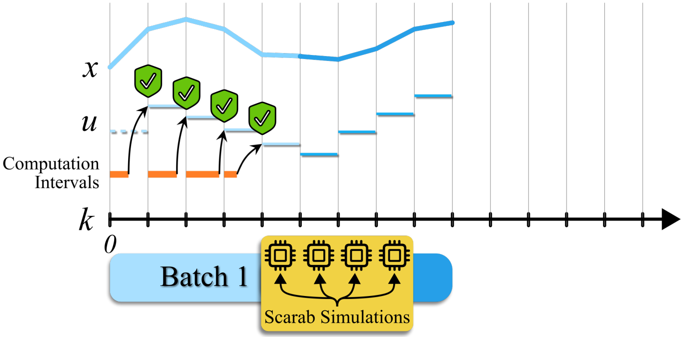


SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

$\implies$ Simulating slow controllers on a long time horizon can require several days

$\implies$ We designed a parallelization scheme that allows many time steps to be run in parallel

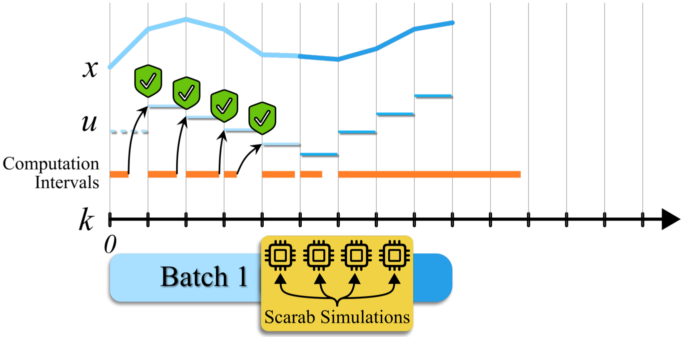


SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

⟹ Simulating slow controllers on a long time horizon can require several days

⟹ We designed a parallelization scheme that allows many time steps to be run in parallel

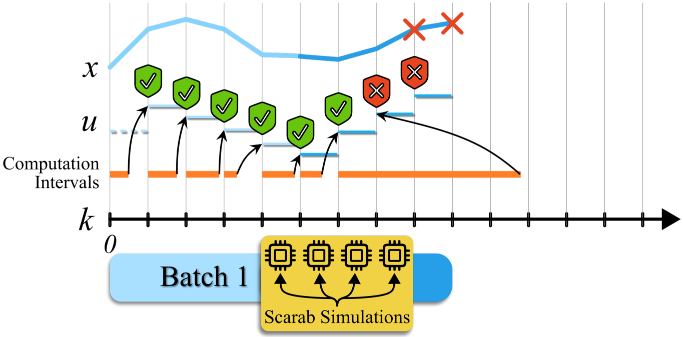


SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

$\Longrightarrow$ Simulating slow controllers on a long time horizon can require several days

$\Longrightarrow$ We designed a parallelization scheme that allows many time steps to be run in parallel

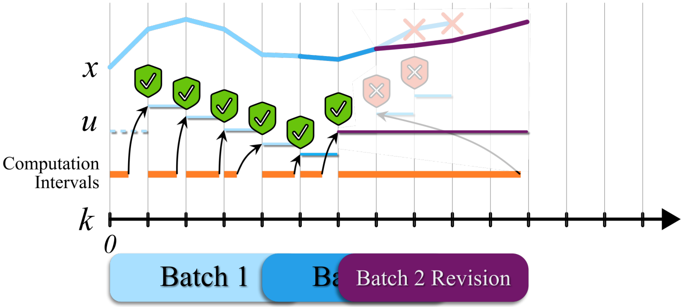


SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

$\Longrightarrow$ Simulating slow controllers on a long time horizon can require several days

$\Longrightarrow$ We designed a parallelization scheme that allows many time steps to be run in parallel

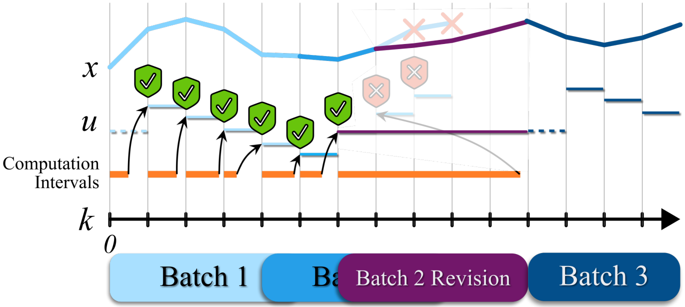


SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

⟹ Simulating slow controllers on a long time horizon can require several days

⟹ We designed a parallelization scheme that allows many time steps to be run in parallel

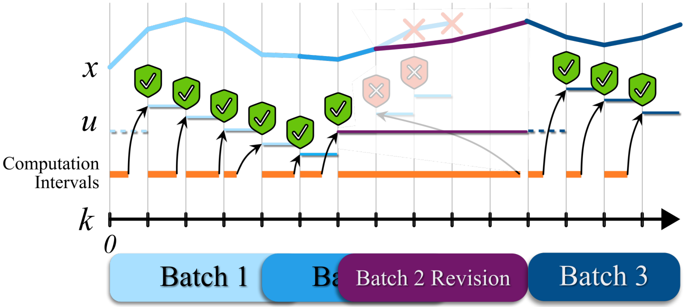


SHARC Parallelization

Simulation in Scarab is 10,000x slower than executing directly on the host processor.

$\Longrightarrow$ Simulating slow controllers on a long time horizon can require several days

$\Longrightarrow$ We designed a parallelization scheme that allows many time steps to be run in parallel

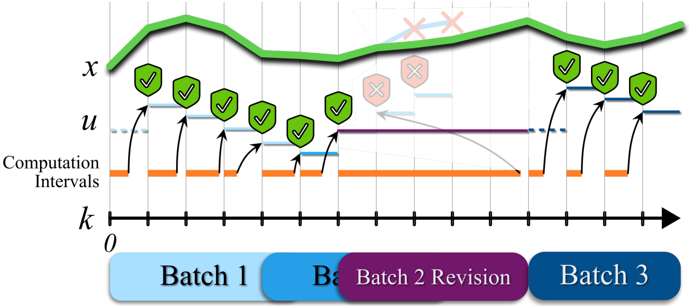


SHARC Parallelization

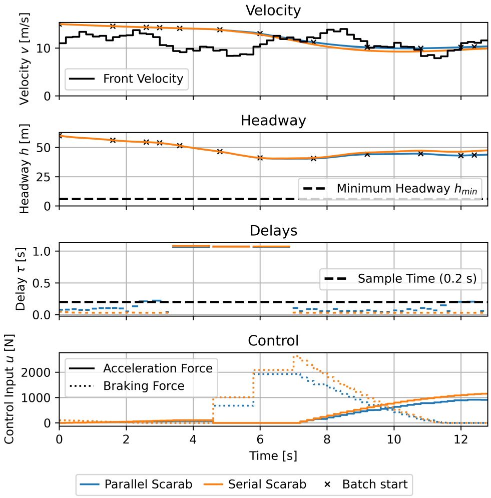
Simulation in Scarab is 10,000x slower than executing directly on the host processor.

⟹ Simulating slow controllers on a long time horizon can require several days

⟹ We designed a parallelization scheme that allows many time steps to be run in parallel



Comparison: Serial vs. Parallel

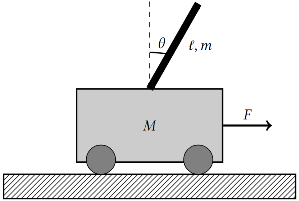


Simulation Time

- Serial: 1 hour, 20 minutes
- Parallel: 40 minutes

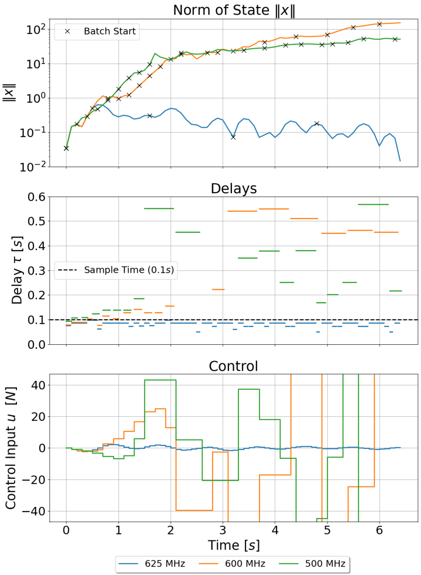
Fidelity loss due to discarding memory effects between time steps.

Example: Nonlinear Inverted Pendulum Example



Problem 2 (Nonlinear MPC)

$$\begin{aligned} \text{minimize} \quad & |\text{angle error}|^2 \\ & + |\text{control effort}|^2 \\ \text{subject to} \quad & \text{Nonlinear Dynamics} \end{aligned}$$



Conclusion

Future Work

- Expand systems simulated in SHARC.
- Generate models of computation time conditioned on state, controller parameters, and hardware configuration.
- Use models of computation time to accelerate parallelization.
- Use SHARC to establish guarantees on system performance.
- Use SHARC for co-design of hardware and controllers by joint optimization.

Questions?

Funding

NSF

AFRL
AIR FORCE RESEARCH LABORATORY

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
UNITED STATES AIR FORCE

DEVCOM
ARMY RESEARCH
LABORATORY

Slides and paper available at
paulwintz.com/publications.

Code at github.com/pwintz/sharc

Linear MPC Problem Formulation for ACC Example

minimize

$$J\big(x_{(\cdot)|k_0}, u_{(\cdot)|k_0}\big) := \sum_{k=k_0}^{k_0+N_p} \big(v_{k|k_0} - v_{\text{des}}\big)^2 + \sum_{k=k_0}^{k_0+N_p-1} u_{k|k_0}^\top R u_{k|k_0} + \alpha \sum_{k=k_0}^{k_0+N_p-2} \big|u_{k+1|k_0} - u_{k|k_0}\big|^2$$

with respect to

$$x_{k_0|k_0}, x_{(k_0+1)|k_0}, \ldots, x_{(k_0+N_p)|k_0} \in \mathbb{R}^2, \quad u_{k_0|k_0}, u_{(k_0+1)|k_0}, \ldots, u_{(k_0+N_p-1)|k_0} \in \mathbb{R}^2$$

subject to

$$x_{k_0|k_0} = \hat{x}_{k_0},$$

and for each $k = k_0,\ k_0+1,\ \ldots,\ k_0+N_p-1$,

$$x_{k+1|k_0} = A(\hat{v}_0)x_{k|k_0} + B(\hat{v}_0)u_{k|k_0} + B_d(\hat{v}_0)\hat{w}(k|k_0),$$

and for each $k = k_0,\ k_0+1,\ \ldots,\ k_0+N_p$,

$$0 \leq v_{k|k_0} \leq v_{\max}, \quad 0 \leq u^{\mathsf{a}}_{k|k_0} \leq u^{\mathsf{a}}_{\max}, \quad 0 \leq u^{\mathsf{b}}_{k|k_0} \leq u^{\mathsf{b}}_{\max}, \quad h_{\min} \leq h_{k|k_0},$$

and for $k = k_0 + N_p$,

$$h_{k|k_0} \geq (v_{\max}/2|a|)v_{k|k_0} - \hat{v}_{\text{F}}^2(k|k_0)/2|a_{\text{F}}| + h_{\min}.$$

Nonlinear MPC Problem Formulation

minimize

$$J\big(x_{(\cdot)|k_0}, u_{(\cdot)|k_0}\big) := \sum_{k=k_0}^{k_0+N_c-1} C\big(x_{k|k_0}, u_{k|k_0}\big) + \sum_{k=k_0+N_c}^{k_0+N_p-1} C\big(x_{k|k_0}, u_{(k_0+N_c-1)|k_0}\big)$$

with respect to

$$x_{k_0|k_0}, x_{(k_0+1)|k_0}, \ldots, x_{(k_0+N_p)|k_0} \in \mathbb{R}^{n_x}, \quad u_{k_0|k_0}, u_{(k_0+1)|k_0}, \ldots, u_{(k_0+N_c-1)|k_0} \in \mathbb{R}^{n_u}$$

subject to

$$x_{k_0|k_0} = \hat{x}_{k_0},$$

and for each $k = k_0,\ k_0+1,\ \ldots,\ k_0+N_p-1,$

$$x_{k+1|k_0} = f(x_{k|k_0}, u_{k|k_0}),$$

and for each $k = k_0,\ k_0+1,\ \ldots,\ k_0+N_p,$

$$\ell_{\mathrm{i}}(x_{k|k_0}, y_{k|k_0}, u_{k|k_0}) \leq 0, \quad \ell_{\mathrm{e}}(x_{k|k_0}, u_{k|k_0}) = 0.$$