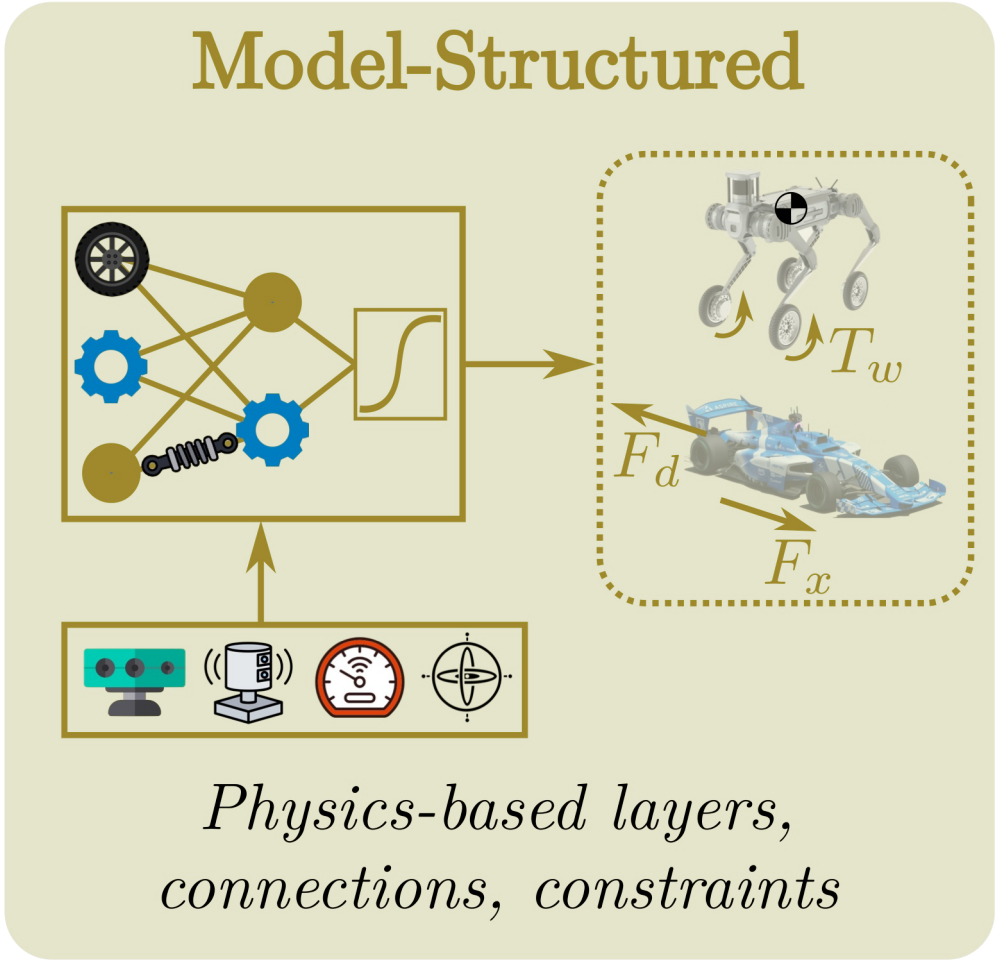
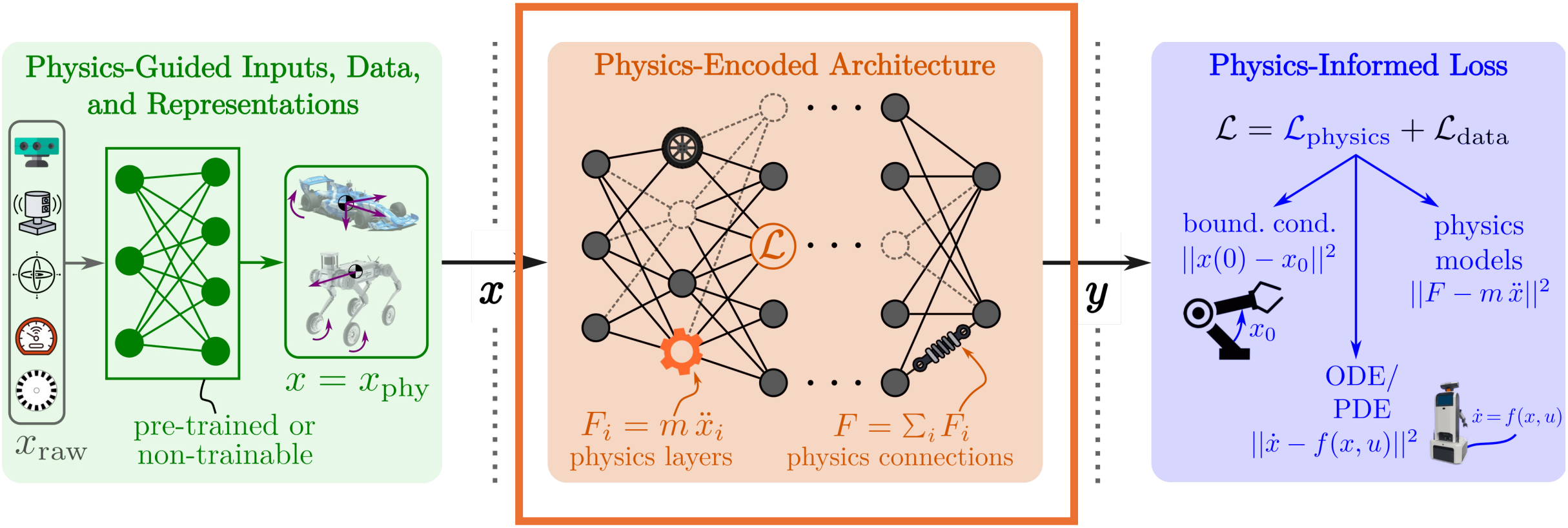
Physics-Encoded Architectures Part III: Model-Structured Learning

Dr. Mattia Piccinini &
Prof. Gastone Pietro Rosati Papini

University of Trento &
Technical University of Munich,
Autonomous Vehicle Systems Lab

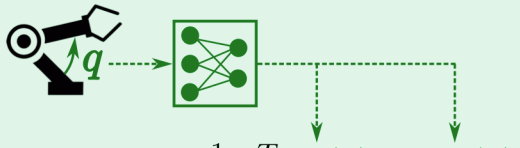


Model Structured Architectures: Positioning



Physics-Encoded Learning Architectures: Multiple Classes

Lagrangian & Hamiltonian Approaches

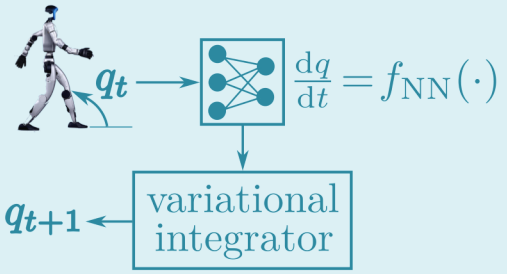


$$\mathcal{L} = \tfrac{1}{2}\dot{q}^T M(q)\dot{q} - \mathcal{V}(q)$$

$$M(q)\ddot{q} + c(q,\dot{q}) + g(q) = \tau$$

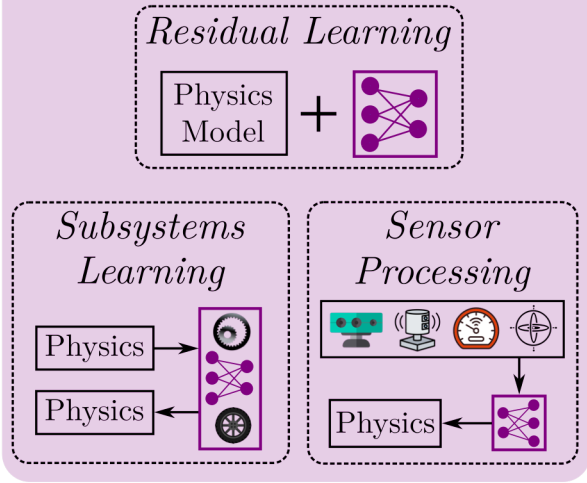
Learning dynamics from variational principles

NODE & Variational Integrator Networks

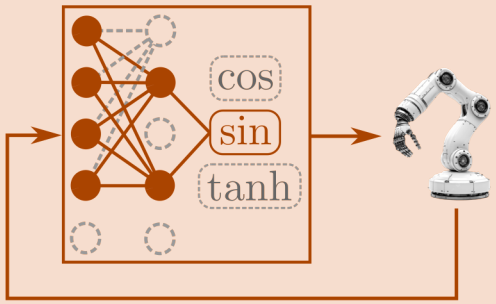


Learning dynamics via structure-preserving integration

Hybrid Physics-Neural

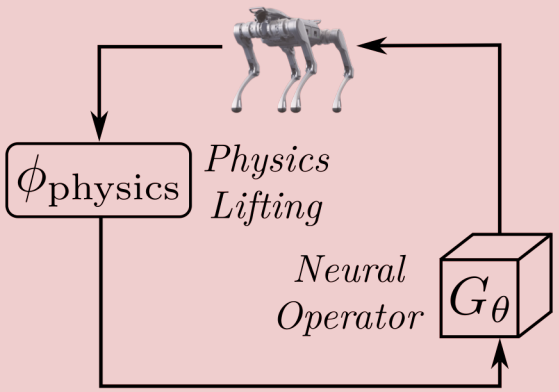


Physics-Encoded Topology Learning



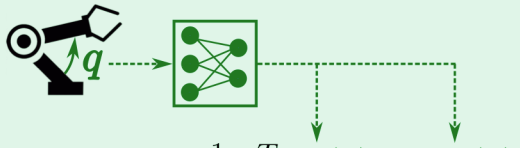
Learning connectivity from physics priors

Physics-Encoded Neural Operators



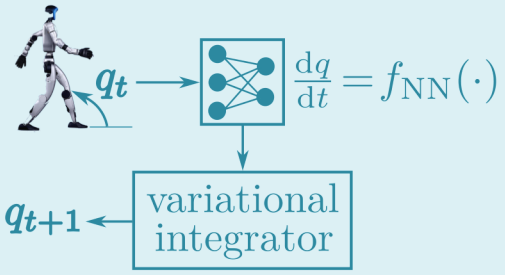
Physics-Encoded Learning Architectures: Multiple Classes

Lagrangian & Hamiltonian Approaches



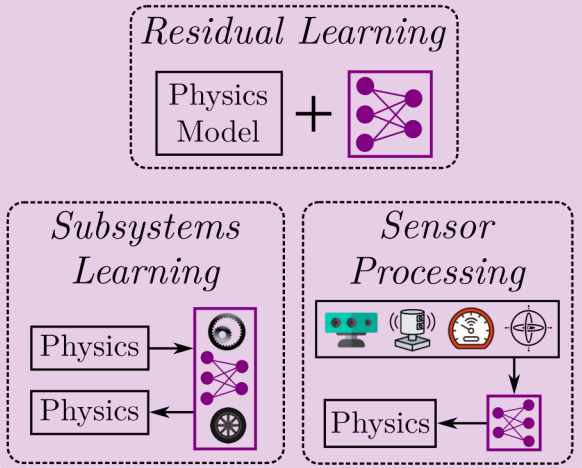
Learning dynamics from variational principles

NODE & Variational Integrator Networks

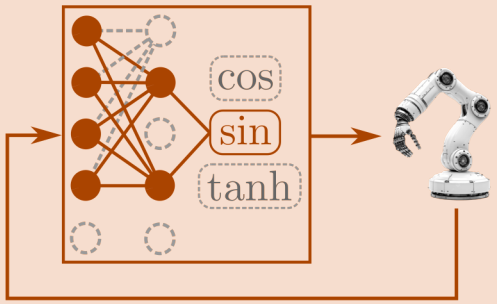


Learning dynamics via structure-preserving integration

Hybrid Physics-Neural

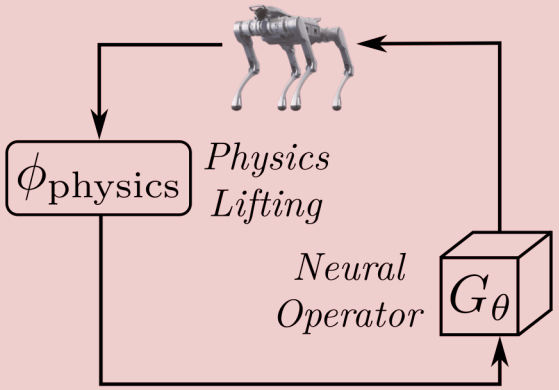


Physics-Encoded Topology Learning

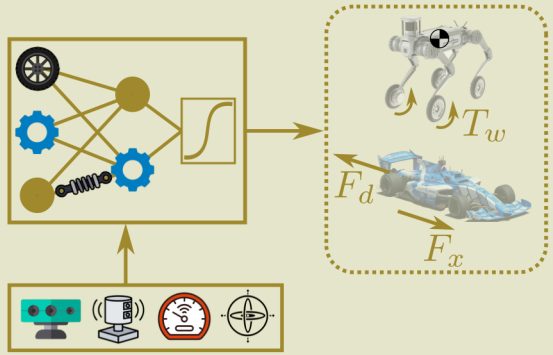


Learning connectivity from physics priors

Physics-Encoded Neural Operators

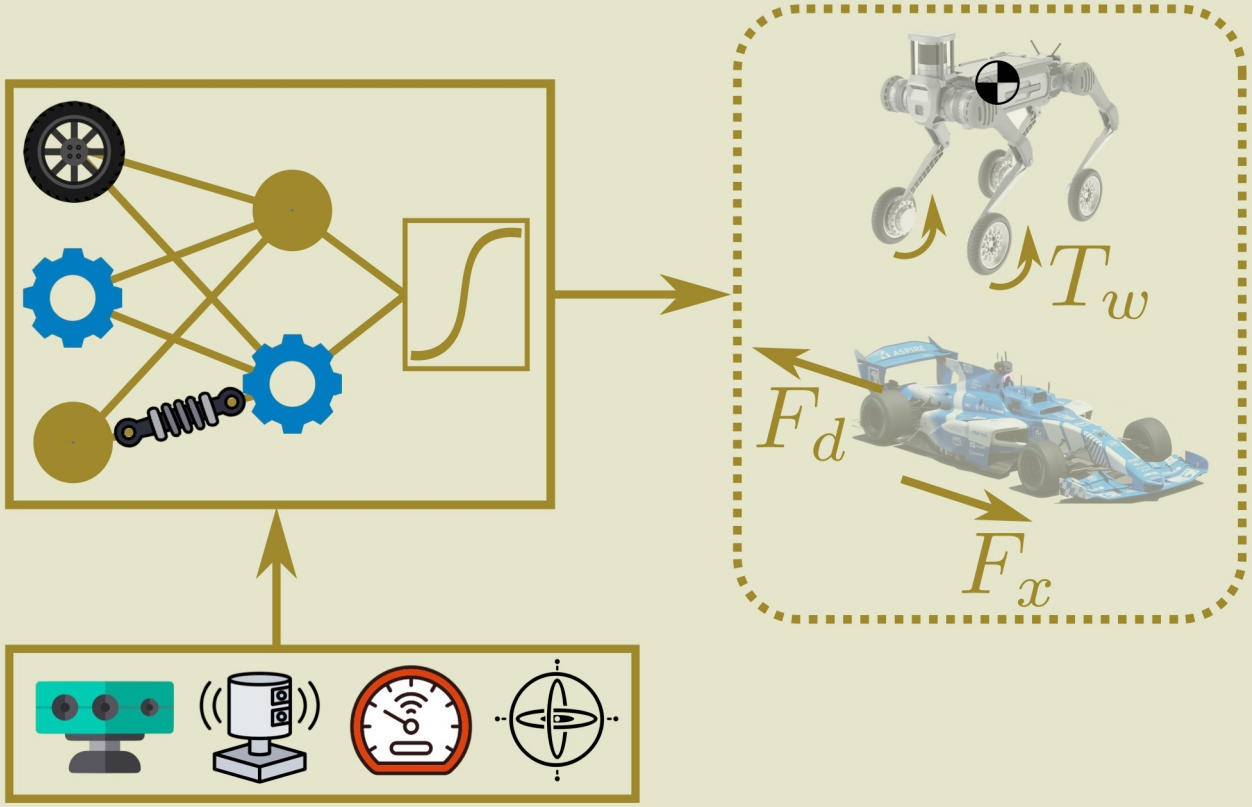


Model-Structured



Physics-based layers, connections, constraints

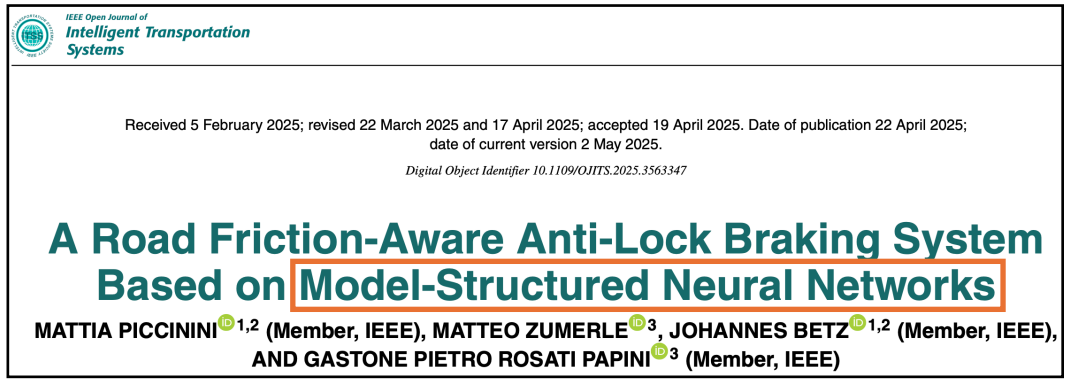
Model-Structured



Physics-based layers, connections, constraints

Taxonomy

The term «***Model-Structured Neural Network***» (***MS-NN***) has recently been introduced by papers of our labs (2025-2026) and aims to cover a broad class of approaches



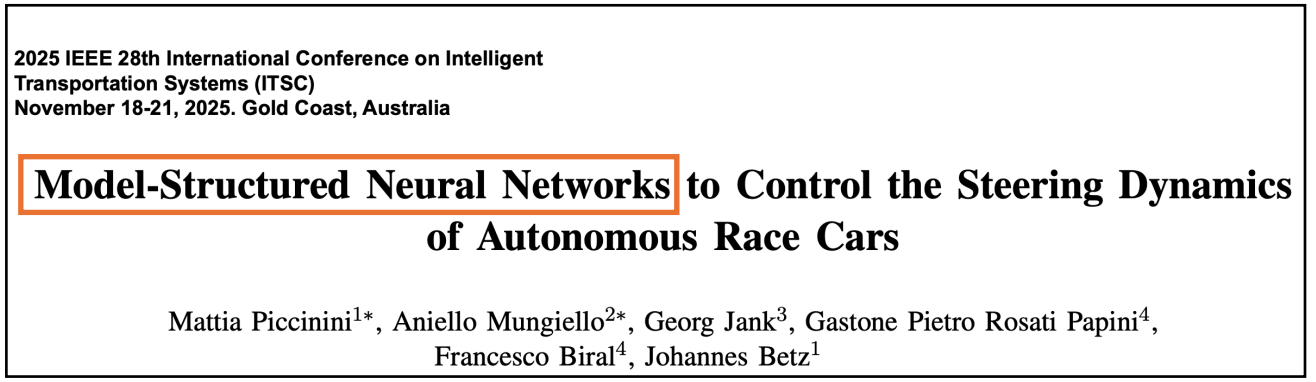
IEEE Open Journal of Intelligent Transportation Systems

Received 5 February 2025; revised 22 March 2025 and 17 April 2025; accepted 19 April 2025. Date of publication 22 April 2025; date of current version 2 May 2025.

Digital Object Identifier 10.1109/OJITS.2025.3563347

A Road Friction-Aware Anti-Lock Braking System Based on Model-Structured Neural Networks

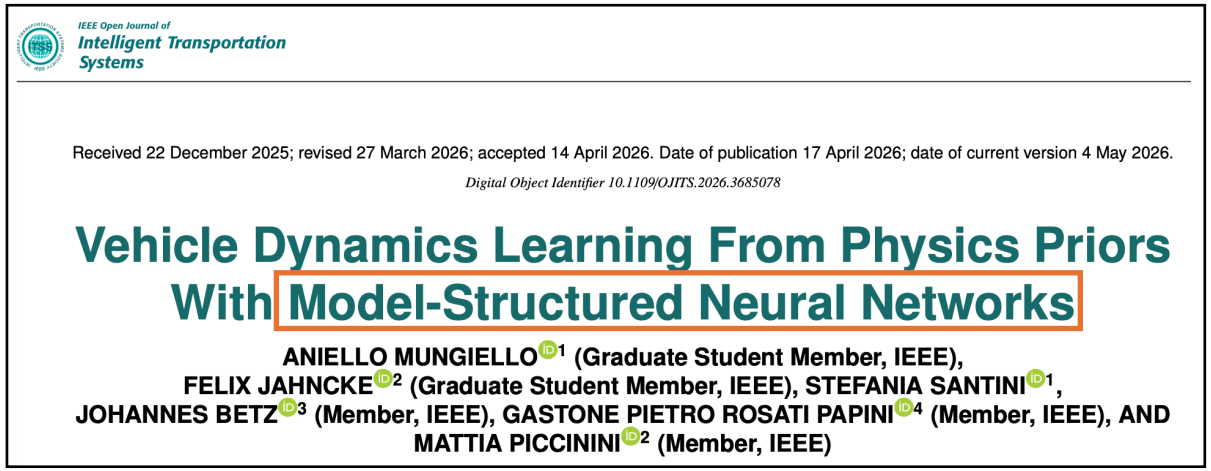
MATTIA PICCININI [1,2] (Member, IEEE), MATTEO ZUMERLE [3], JOHANNES BETZ [1,2] (Member, IEEE), AND GASTONE PIETRO ROSATI PAPINI [3] (Member, IEEE)



2025 IEEE 28th International Conference on Intelligent Transportation Systems (ITSC)
November 18-21, 2025. Gold Coast, Australia

Model-Structured Neural Networks to Control the Steering Dynamics of Autonomous Race Cars

Mattia Piccinini[1*], Aniello Mungiello[2*], Georg Jank[3], Gastone Pietro Rosati Papini[4], Francesco Biral[4], Johannes Betz[1]



IEEE Open Journal of Intelligent Transportation Systems

Received 22 December 2025; revised 27 March 2026; accepted 14 April 2026. Date of publication 17 April 2026; date of current version 4 May 2026.

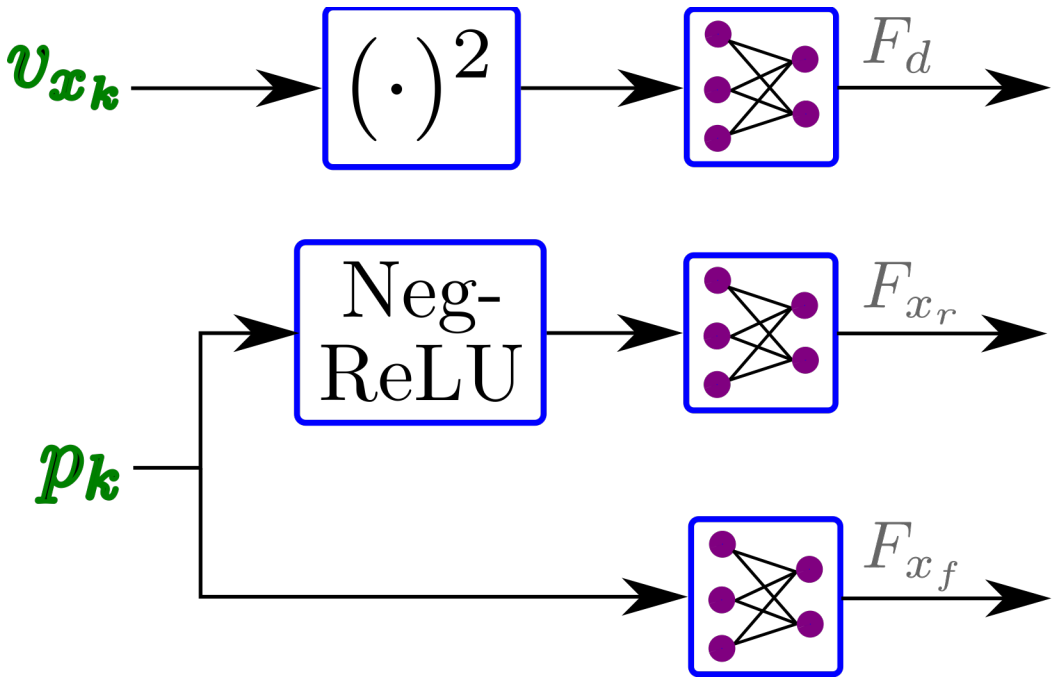
Digital Object Identifier 10.1109/OJITS.2026.3685078

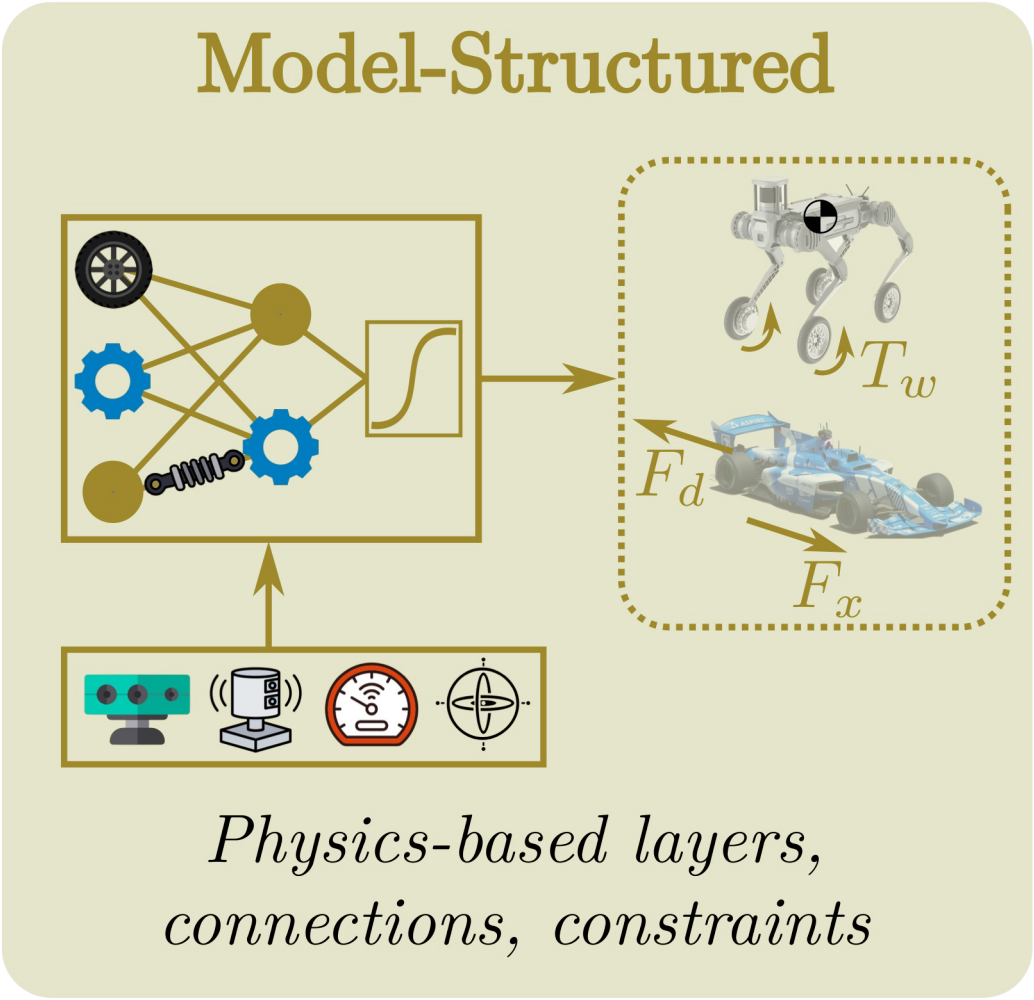
Vehicle Dynamics Learning From Physics Priors With Model-Structured Neural Networks

ANIELLO MUNGIELLO [1] (Graduate Student Member, IEEE), FELIX JAHNCKE [2] (Graduate Student Member, IEEE), STEFANIA SANTINI [1], JOHANNES BETZ [3] (Member, IEEE), GASTONE PIETRO ROSATI PAPINI [4] (Member, IEEE), AND MATTIA PICCININI [2] (Member, IEEE)

Model Structured Architectures: General Principles

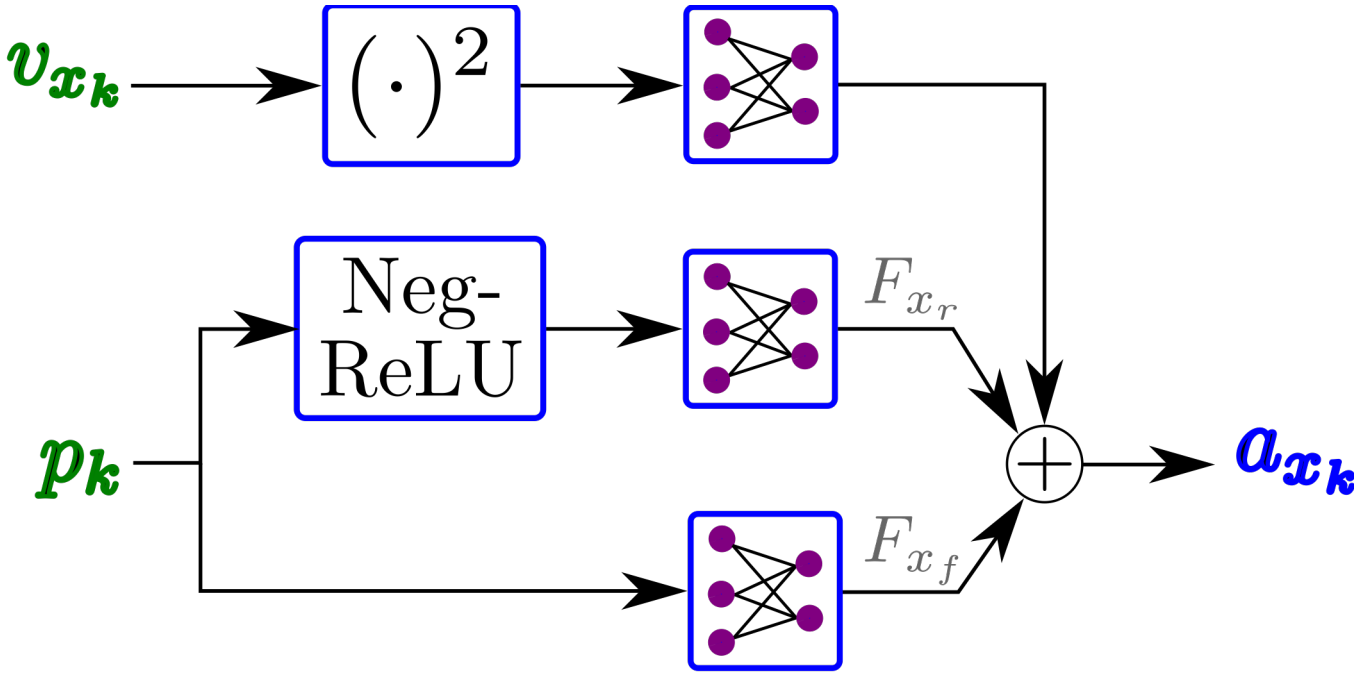
- ***Physics-based layers***: encoding custom physics equations / principles

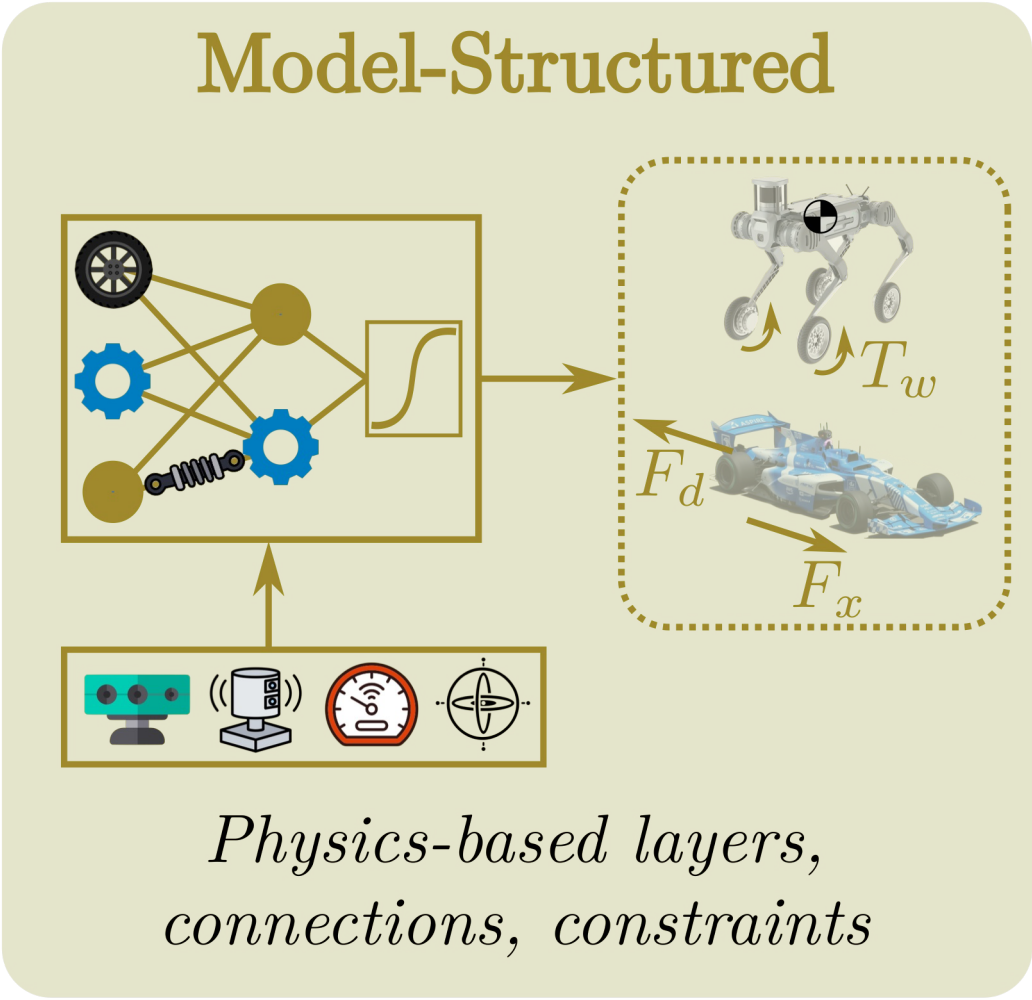




Model Structured Architectures: General Principles

- ***Physics-based layers***: encoding custom physics equations / principles
- ***Physics-based connections***: superposition of forces, symmetries, …



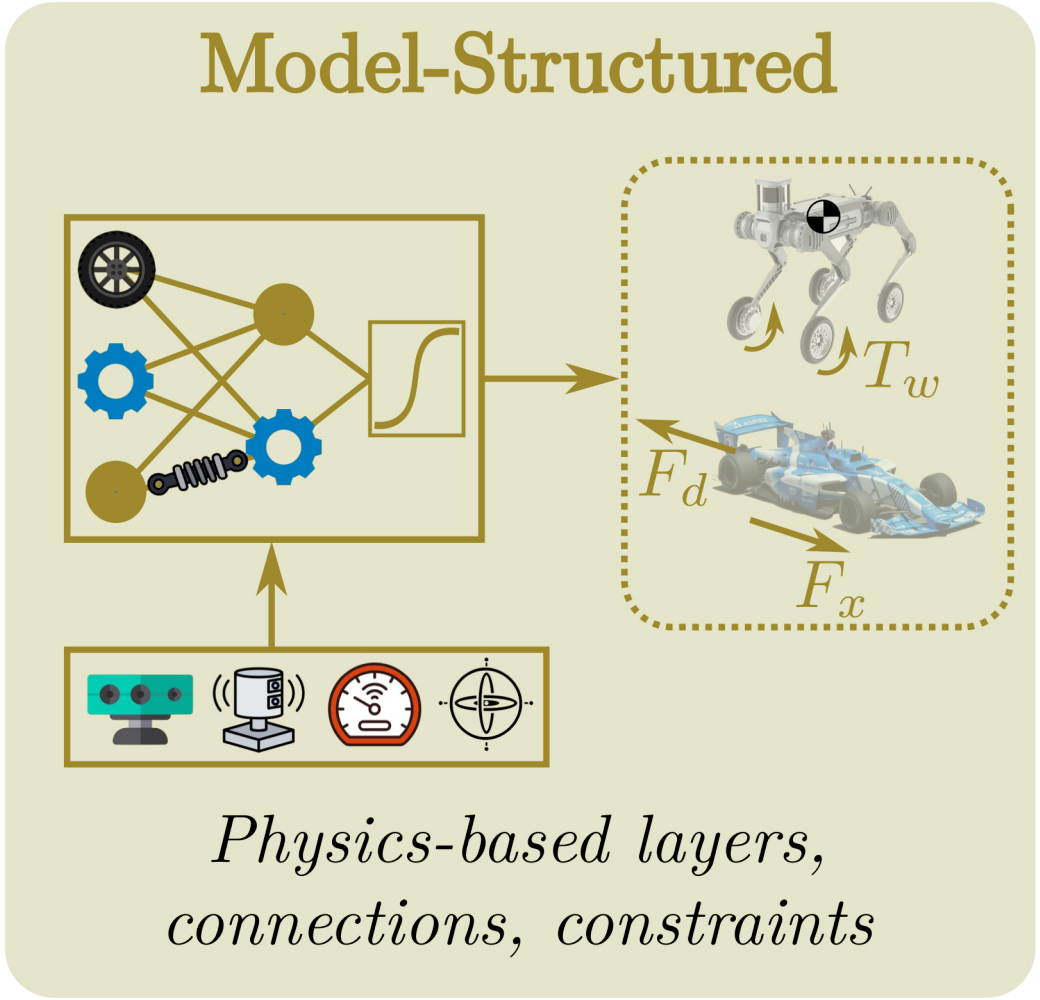


Model Structured Architectures: General Principles

- ***Physics-based layers***: encoding custom physics equations / principles
- ***Physics-based connections***: superposition of forces, symmetries, …
- ***Physics-based constraints***: Constrain parameters to have physically meaningful values
  - Note: Only „**architectural**“ constraints, no loss function regularization terms!
  - Example from Chrosniak et al., 2024

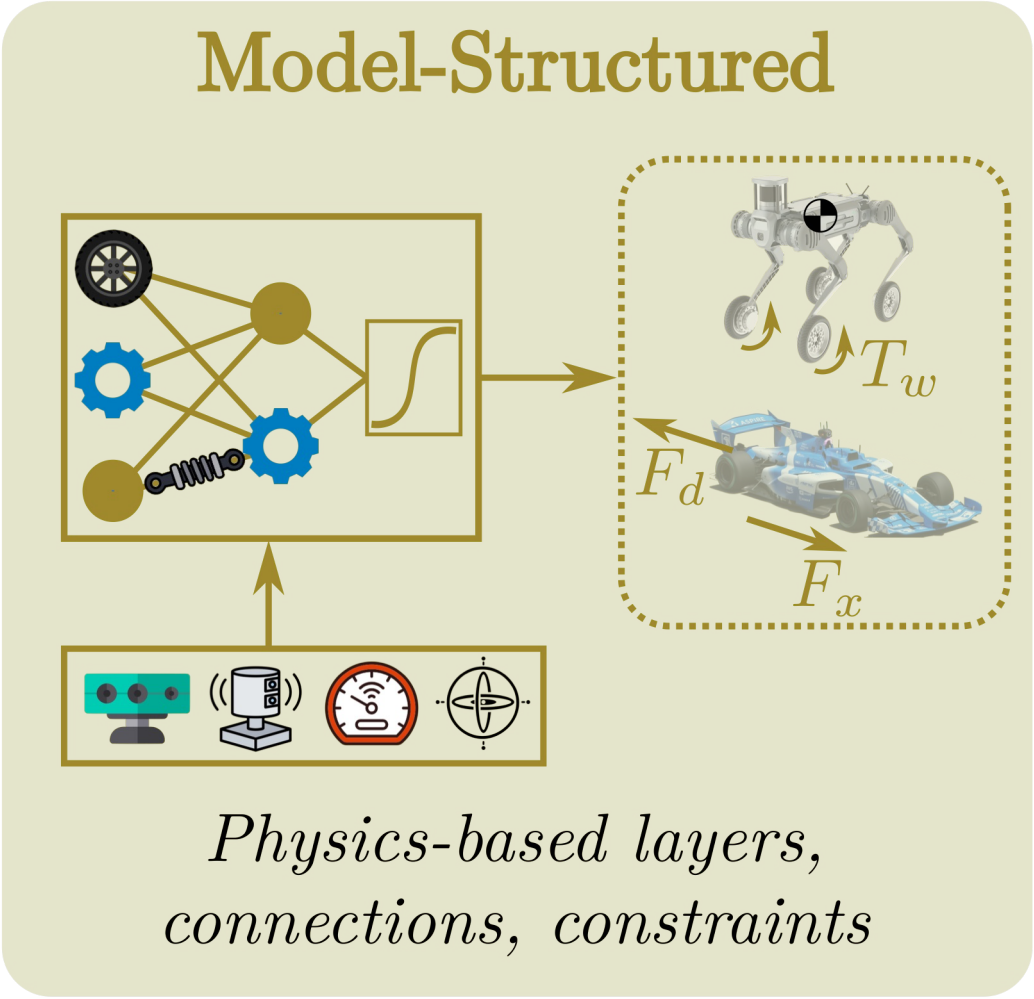
$$\hat{\Phi}_u = \sigma(z) \cdot (\bar{\Phi}_u - \underline{\Phi}_u) + \underline{\Phi}_u$$

σ = sigmoid fun, $\bar{\Phi}_u$ = upper bound, $\underline{\Phi}_u$ = lower bound



Model Structured Architectures: General Principles

- ***Physics-based layers***: encoding custom physics equations / principles
- ***Physics-based connections***: superposition of forces, symmetries, …
- ***Physics-based constraints***: Constrain parameters to have physically meaningful values

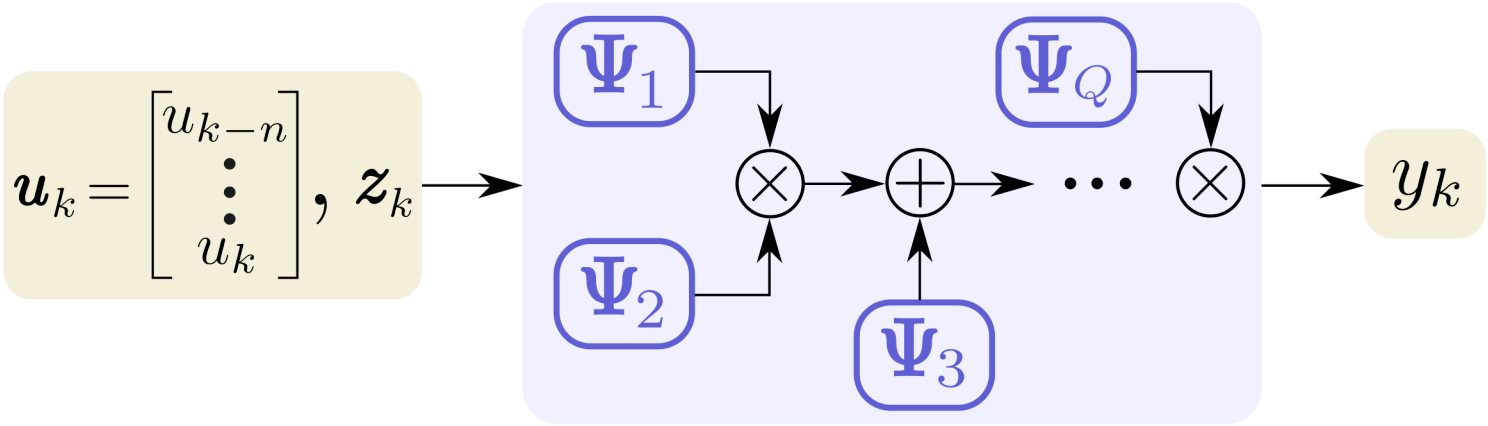


Practical guideline: start from an (imperfect) physical model and *neuralize* its architecture

Model Structured Architectures: Proposed Design

General approach for **dynamics learning of physical systems** via

Neural Compositional Modules (NCMs) Ψ



y_k = system output, $\boldsymbol{u}_k = [u_{k-n}, \dots, u_k]$ = history of inputs, $\boldsymbol{z}_k$ = scheduling vars

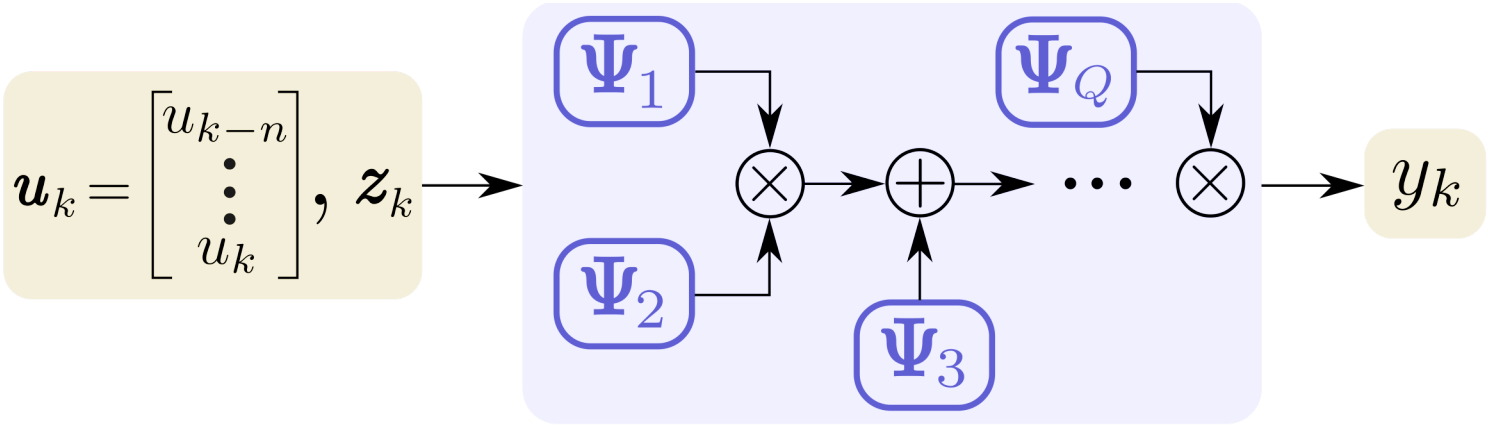
$$y_k = \mathcal{F}\left(\Psi_1, \Psi_2, \dots, \Psi_Q\right), \quad \Psi_i = \Psi_i(\boldsymbol{u}_k, \boldsymbol{z}_k), \quad i \in \{1, \dots, Q\}$$

Where $\mathcal{F} \in \mathcal{C}(O)$, $O = \{+,\times\}$ denotes the set of all compositions generated by sums and multiplications

Model Structured Architectures: Proposed Design

General approach for **dynamics learning of physical systems** via

Neural Compositional Modules (NCMs) Ψ

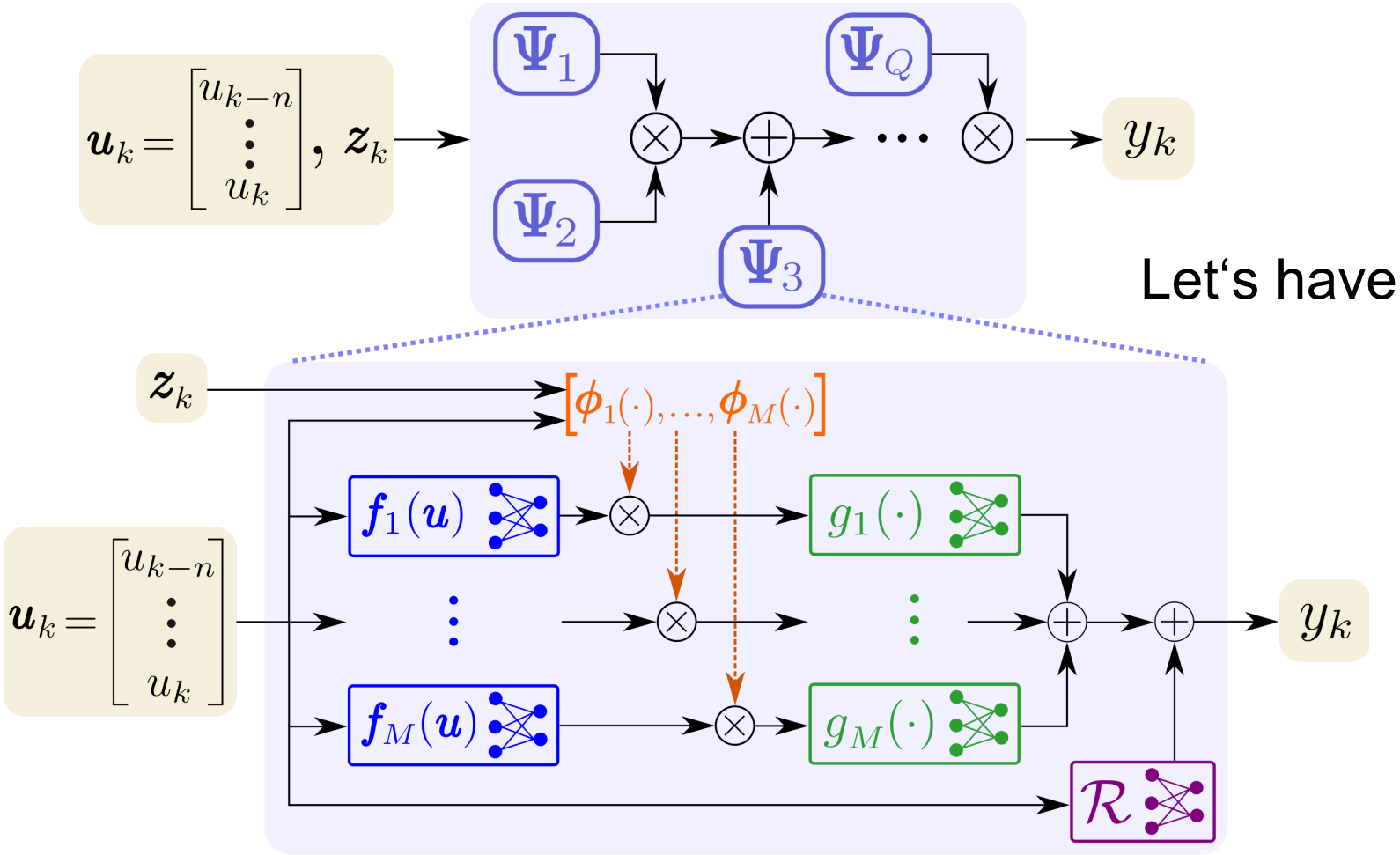


How to design an **NCM Ψ** ?

Model Structured Architectures: Proposed Design

General approach for **dynamics learning of physical systems** via

Neural Compositional Modules (NCMs) Ψ



Let‘s have a closer look!

Neural Compositional Modules: Input / Output Representation

Consider a physical system described by a nonlinear time-invariant state-space model

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

Substitute x_k from the first into the second eq.

$$y_k = h(x_k, u_k) = h(f(x_{k-1}, u_{k-1}), u_k)$$
$$= \cdots = \mathcal{G}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}}, x_{k-\mathrm{n}})$$

$$y_k = \mathcal{G}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}}, \cancel{x_{k-\mathrm{n}}})$$

If the past history of inputs (n) is „*long enough*“, the effect of the initial state $x_{k-\mathrm{n}}$ is negligible (the free response vanishes, and the input-forced response dominates)

Neural Compositional Modules: Input / Output Representation

Consider a physical system described by a nonlinear time-invariant state-space model

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

Substitute x_k from the first into the second eq.

$$y_k = h(x_k, u_k) = h(f(x_{k-1}, u_{k-1}), u_k)$$
$$= \cdots = \mathcal{G}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}}, x_{k-\mathrm{n}})$$

$$\boxed{y_k = \mathcal{G}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}})}$$

If the past history of inputs (n) is „*long enough*“, the effect of the initial state $x_{k-\mathrm{n}}$ is negligible (the free response vanishes, and the input-forced response dominates)

Neural Compositional Modules: Input / Output Representation

$$y_k = \mathcal{G}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}})$$

Also assume that we can have different **parametrizations** of $\mathcal{G}(\cdot)$ in **different operating regimes** (with partial or no overlap among the regimes), with **scheduling variables** z_k

Full NCM: $y_k = \Psi(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}}, z_k)$

output

history of inputs

scheduling vars

Questions & Discussion

Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***linear*** : $y_k = (\boldsymbol{\Gamma} * \boldsymbol{u})_k = \sum_{\mathrm{i}=0}^{\infty} \Gamma_{\mathrm{i}}\, u_{k-\mathrm{i}}$

- $\boldsymbol{\Gamma}$ is the **discrete-time transfer function** (= impulse response) of the system

Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***linear*** : $y_k = (\boldsymbol{\Gamma} * \boldsymbol{u})_k = \sum_{\mathrm{i}=0}^{\infty} \Gamma_{\mathrm{i}}\, u_{k-\mathrm{i}}$

- $\boldsymbol{\Gamma}$ is the **discrete-time transfer function** (= impulse response) of the system
- **Infinite Impulse Response (IIR)** system

Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***linear*** : $y_k = (\boldsymbol{\Gamma} * \boldsymbol{u})_k \approx \sum_{\mathrm{i}=0}^{\mathrm{n}} \Gamma_{\mathrm{i}}\, u_{k-\mathrm{i}}$

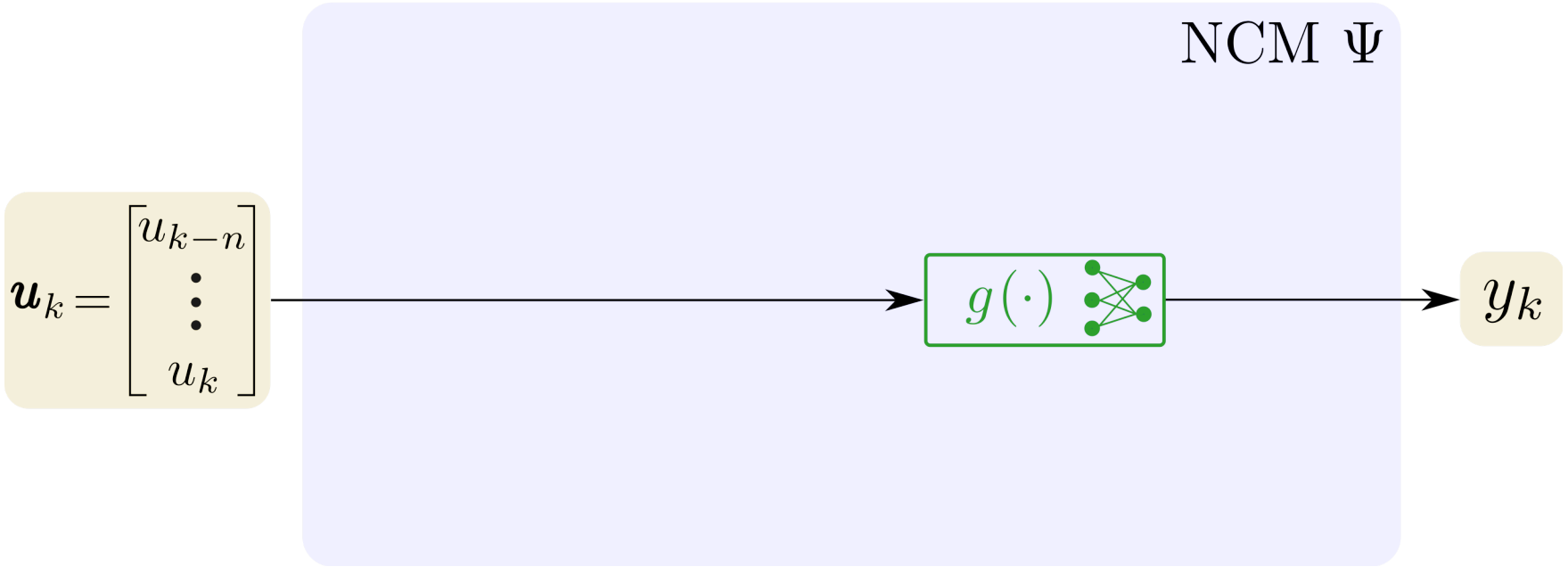
- $\boldsymbol{\Gamma}$ is the **discrete-time transfer function** (= impulse response) of the system
- **Finite Impulse Response (FIR)** approximation

Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***linear*** : $y_k = (\boldsymbol{\Gamma} * \boldsymbol{u})_k \approx \sum_{\mathrm{i}=0}^{\mathrm{n}} \Gamma_{\mathrm{i}}\, u_{k-\mathrm{i}} = g(\boldsymbol{u}_k)$

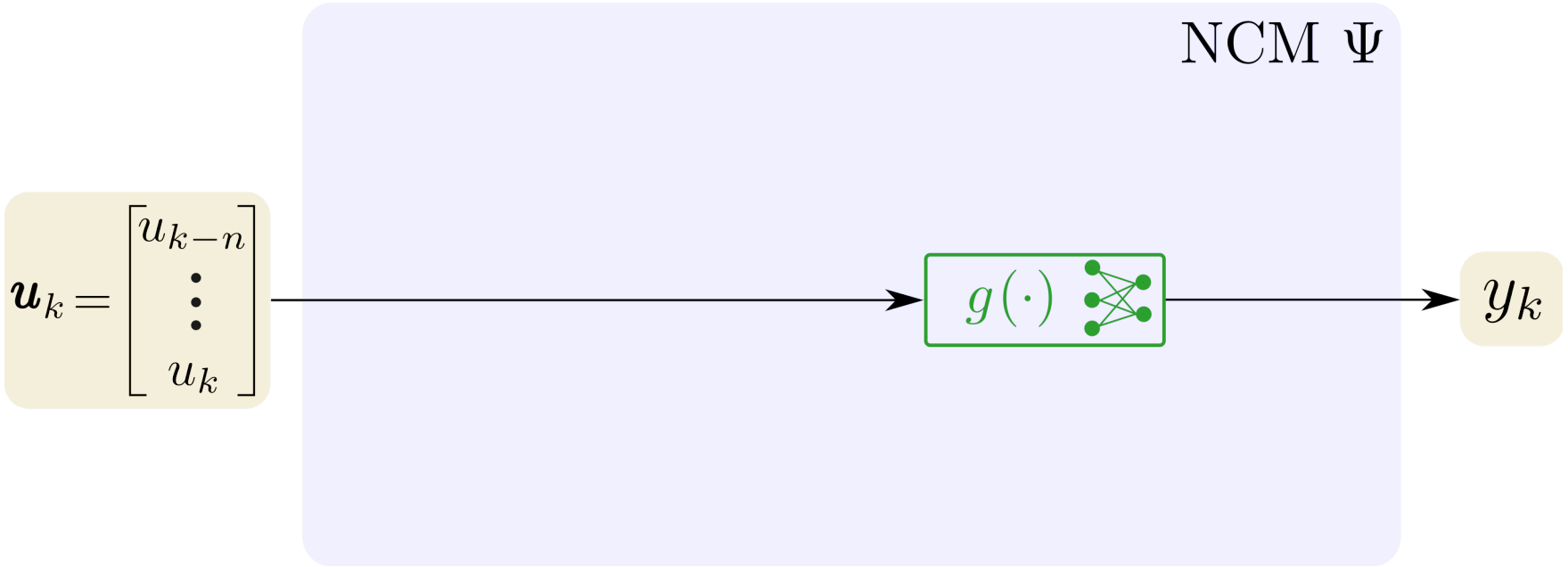
- $\boldsymbol{\Gamma}$ is the **discrete-time transfer function** (= impulse response) of the system
- **Finite Impulse Response (FIR)** approximation = **1 linear fully connected layer** g whose weights are the impulse response coefficients



Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***linear*** : $y_k = \sum_{\mathrm{i}=0}^{\mathrm{n}} \Gamma_{\mathrm{i}} \qquad u_{k-\mathrm{i}} = g(\boldsymbol{u}_k \quad)$

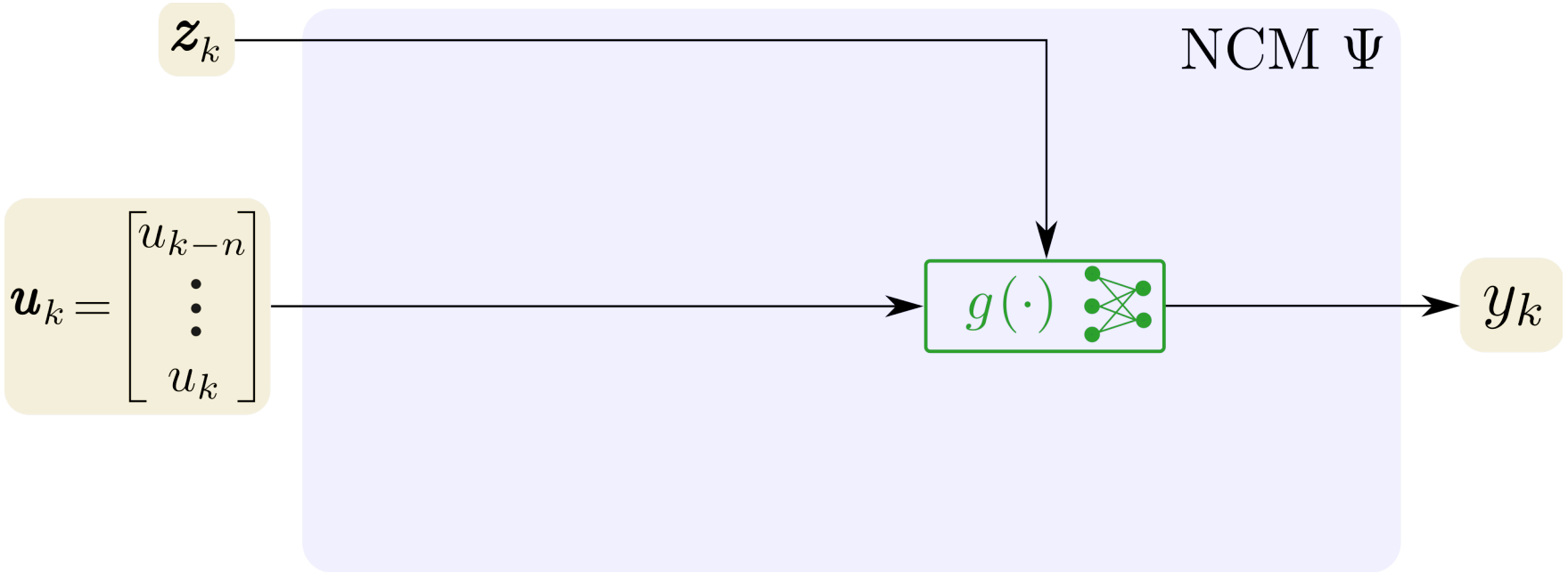


Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \dots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***locally linear*** : $y_k = \sum_{\mathrm{i}=0}^{\mathrm{n}} \Gamma_{\mathrm{i}}(\boldsymbol{z}_k)\, u_{k-\mathrm{i}} = g(\boldsymbol{u}_k, \boldsymbol{z}_k)$

- $\boldsymbol{z}_k$ is a (set of) discrete or continuous scheduling variables. For example, $\boldsymbol{u}_k \subseteq \boldsymbol{z}_k$
- We propose to have different modules $g_{\mathrm{j}}(\boldsymbol{u}_k)$ activated locally by $\boldsymbol{\phi}_{\mathrm{j}}(\boldsymbol{z}_k)$, $\mathrm{j} \in \{1, \dots, M\}$

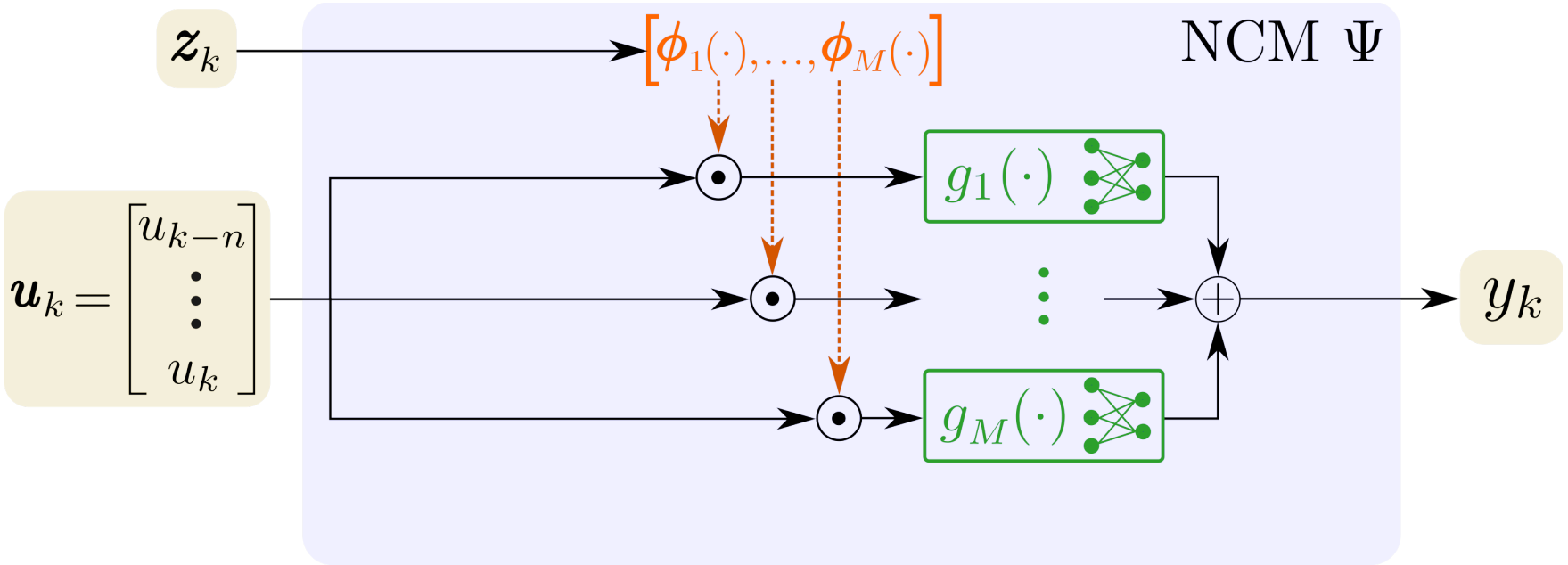


Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \dots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***locally linear*** : $y_k = \sum_{j=1}^{M} [\boldsymbol{u}_k \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$

- $\boldsymbol{z}_k$ is a (set of) discrete or continuous scheduling variables. For example, $\boldsymbol{u}_k \subseteq \boldsymbol{z}_k$
- We propose to have different modules $g_{\mathrm{j}}(\boldsymbol{u}_k)$ activated locally by $\boldsymbol{\phi}_{\mathrm{j}}(\boldsymbol{z}_k)$, j $\in \{1, \dots, M\}$
- $\odot$ is the element-wise product

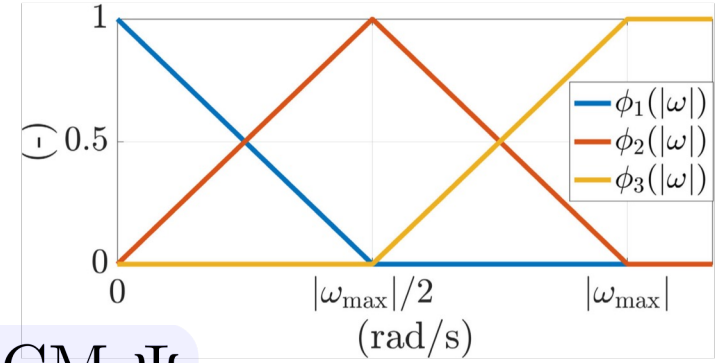


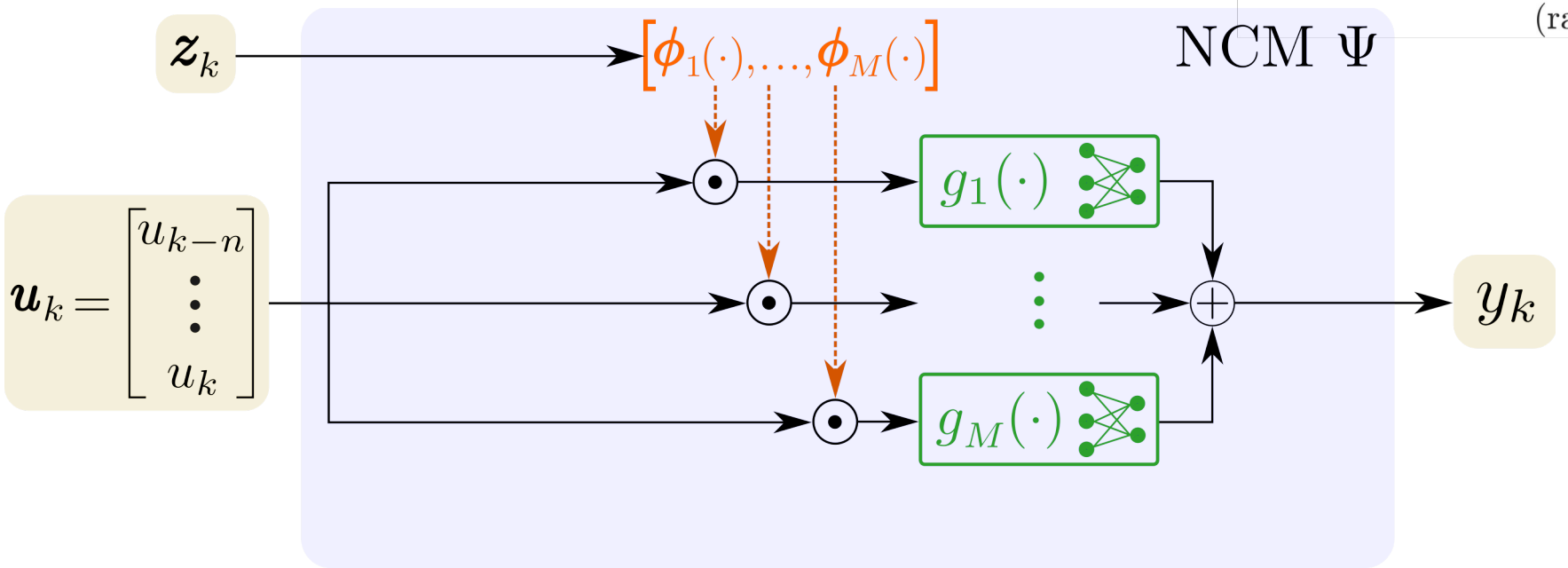
Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***locally linear*** : $y_k = \sum_{j=1}^{M} [\boldsymbol{u}_k \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$

- $\boldsymbol{\phi}_{\mathrm{j}}(\boldsymbol{z}_k)$ may be trainable or frozen
- Common choices: triangular, rectangular, Gaussian
- $\sum_{\mathrm{j}=1}^{M} \boldsymbol{\phi}_{\mathrm{j}}(\boldsymbol{z}_k) = 1, \quad \boldsymbol{\phi}_{\mathrm{j}}(\boldsymbol{z}_k) \geq 0 \ \ \forall \boldsymbol{z}_k$





Neural Compositional Modules: Internal Architecture

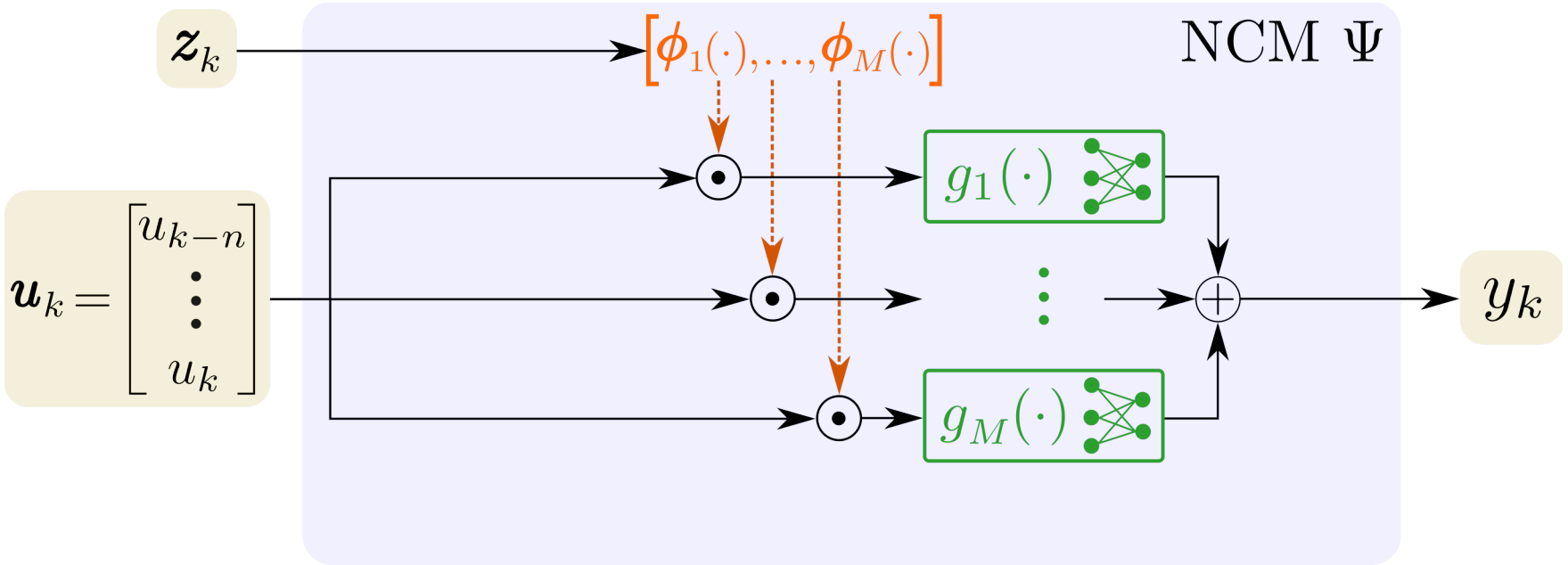
$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***locally linear*** : $y_k = \sum_{j=1}^{M} [\boldsymbol{u}_k \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$

$\boldsymbol{u}_k = [u_{k-n}, \ldots, u_k]$: past history of inputs

$\boldsymbol{\phi}_j = [\phi_{j_1}, \ldots, \phi_{j_{n+1}}]$: regime-dependent activation functions

$\boldsymbol{g}_j = [g_{j_1}, \ldots, g_{j_{n+1}}]^T$: neural FIR layer (fully connected layer)

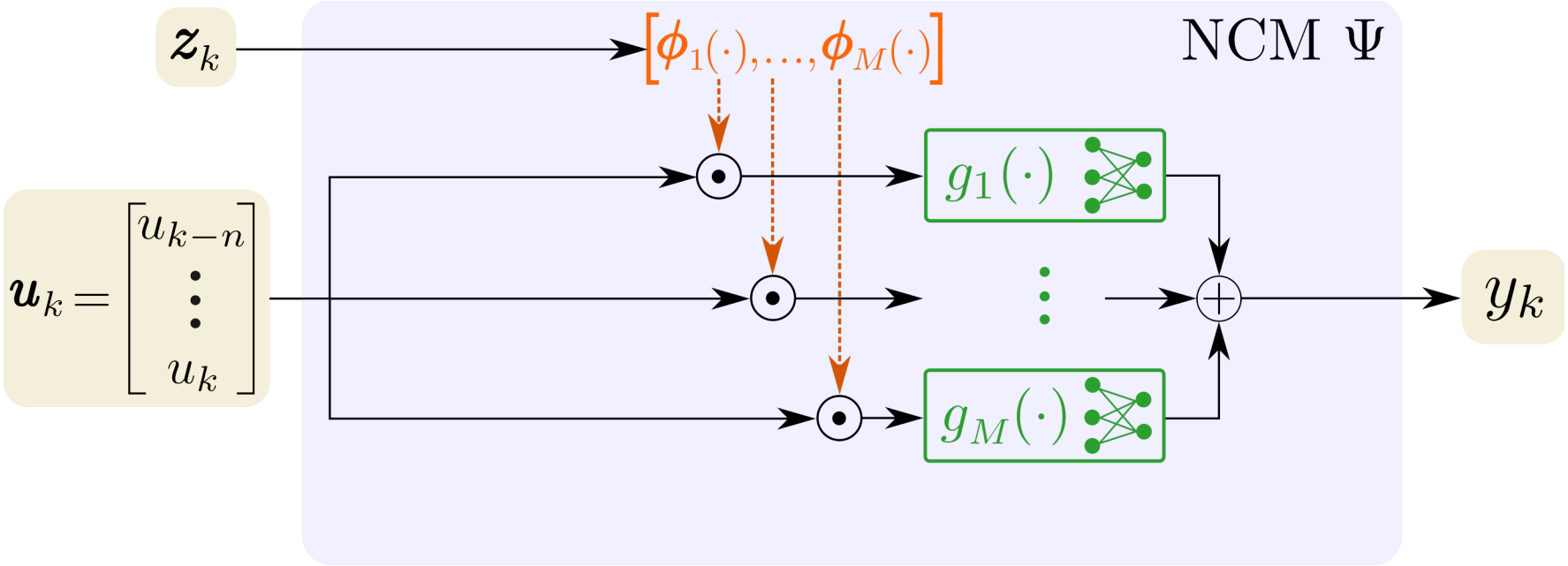


Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***locally linear*** :

$$y_k = \sum_{j=1}^{M} [\ \boldsymbol{u}_k\ \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$$



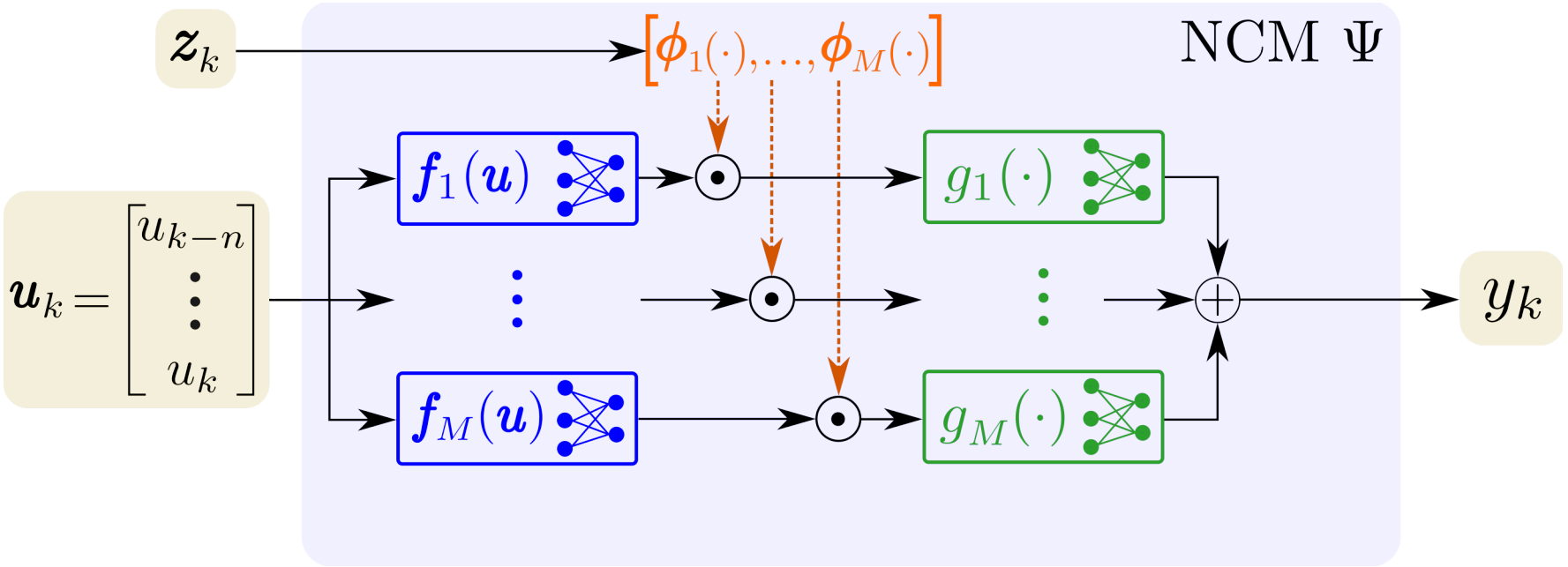
Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***locally linear + input nonlinearities*** :

$$y_k = \sum_{j=1}^{M} [\boldsymbol{f}_j(\boldsymbol{u}_k) \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$$

How to design $\boldsymbol{f}_j(\cdot)$?

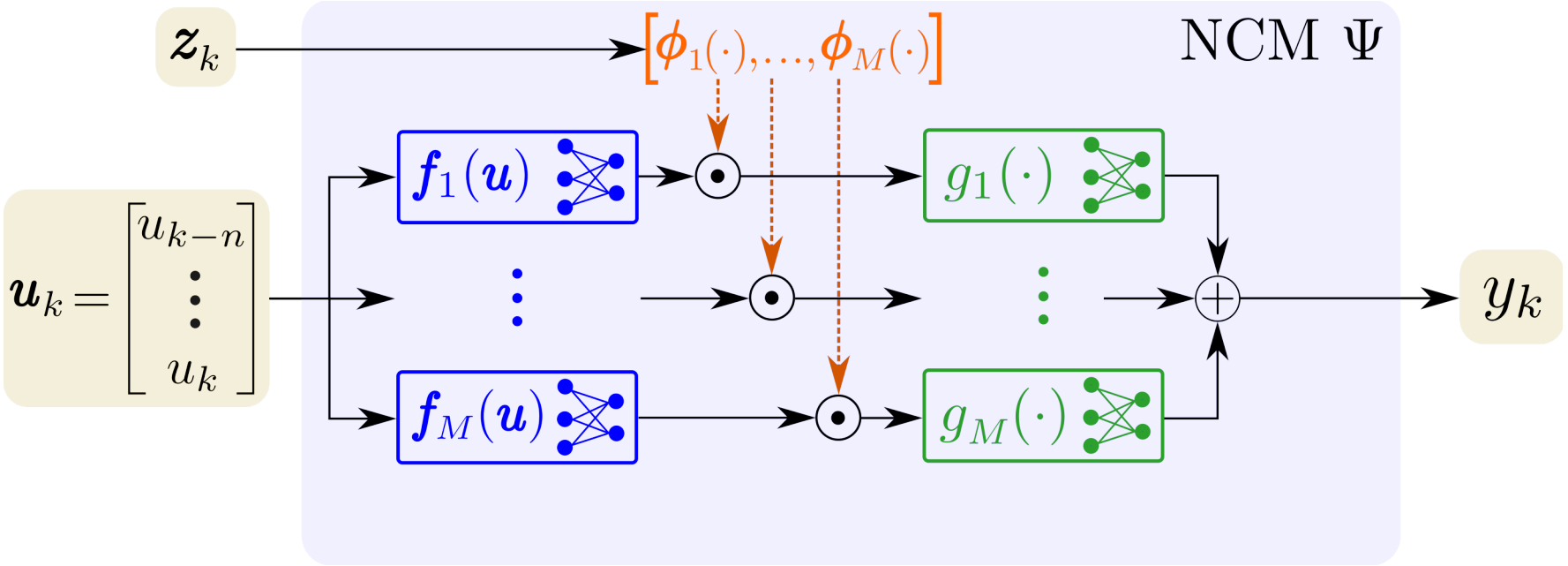


Neural Compositional Modules: Internal Architecture

$$y_k = \sum_{j=1}^{M} [\boldsymbol{f}_j(\boldsymbol{u}_k) \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$$

$f(\boldsymbol{u}_k)$ can encode **prior physics** / domain knowledge (if available):

- Nonlinear input mappings or transformations
- May contain analytical + neural relations → designer / engineer expertise required



Neural Compositional Modules: Internal Architecture

$$y_k = \sum_{j=1}^{M} [\boldsymbol{f}_j(\boldsymbol{u}_k) \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$$

What if only **limited physics priors** are available?

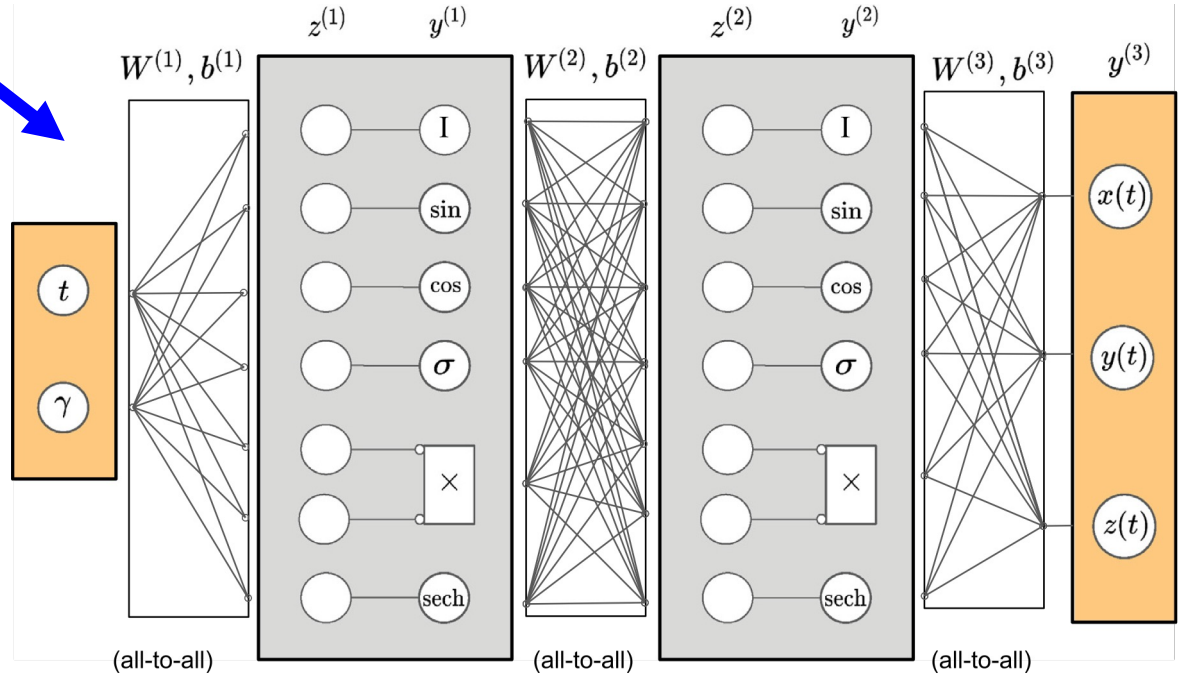
- For example, the dynamics of a robot manipulator are a **combination** of **trigonometric** functions, but the exact form is **unknown**

Neural Compositional Modules: Internal Architecture

$$y_k = \sum_{j=1}^{M} [\boldsymbol{f}_j(\boldsymbol{u}_k) \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$$

What if only **limited physics priors** are available?

- In this case, $\boldsymbol{f}(\boldsymbol{u}_k)$ can learn to combine trigonometric functions with **Equation Learning**



H. Perez-Villeda, J. Piater, M. Saveriano, "Learning and extrapolation of robotic skills using task-parameterized equation learner networks", *Robotics and Autonomous Systems*, 2023

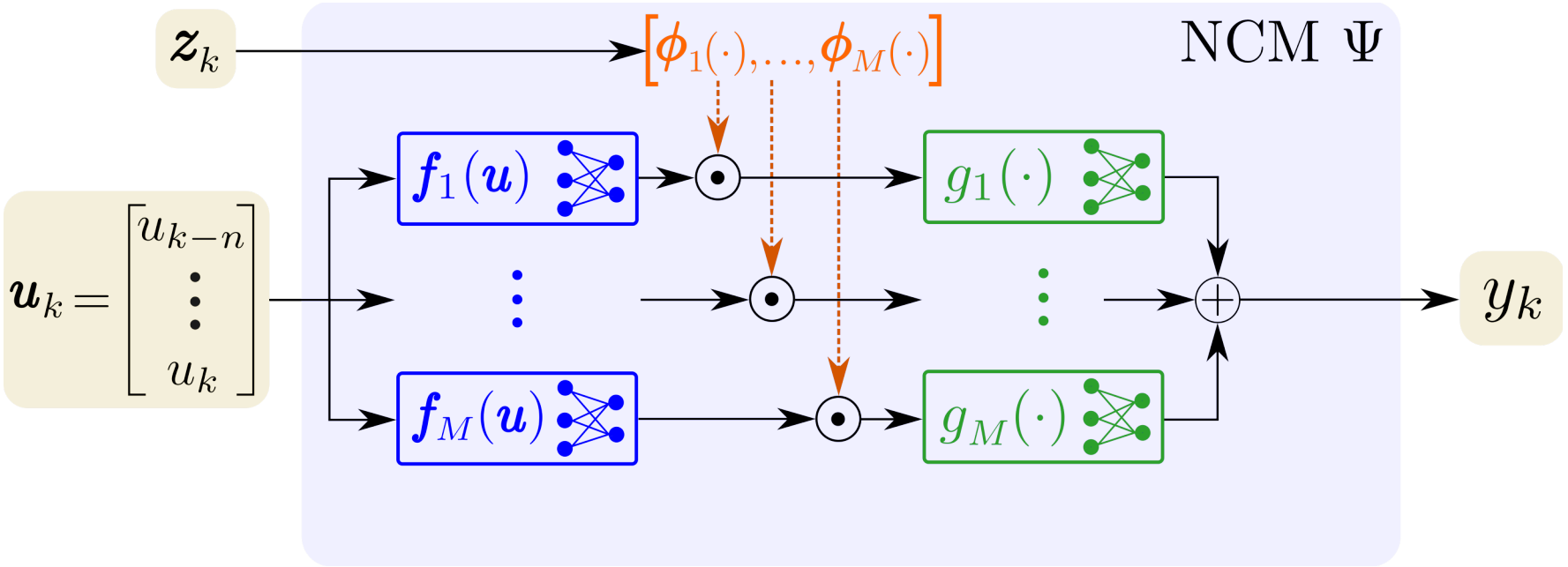
Neural Compositional Modules: Internal Architecture

$$y_k = \sum_{j=1}^{M} [\boldsymbol{f}_j(\boldsymbol{u}_k) \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$$

What if **no physics priors** are available?

- $\boldsymbol{f}_j(\boldsymbol{u}_k)$ can be a **learnable Taylor expansion** of the unknown nonlinear relation

$$\boldsymbol{f}_j(\boldsymbol{u}_k) = a_0 + a_1 \boldsymbol{u}_k + a_2 \boldsymbol{u}_k^2 \quad \text{(2nd order Taylor with learnable } a_0, a_1, a_2\text{)}$$

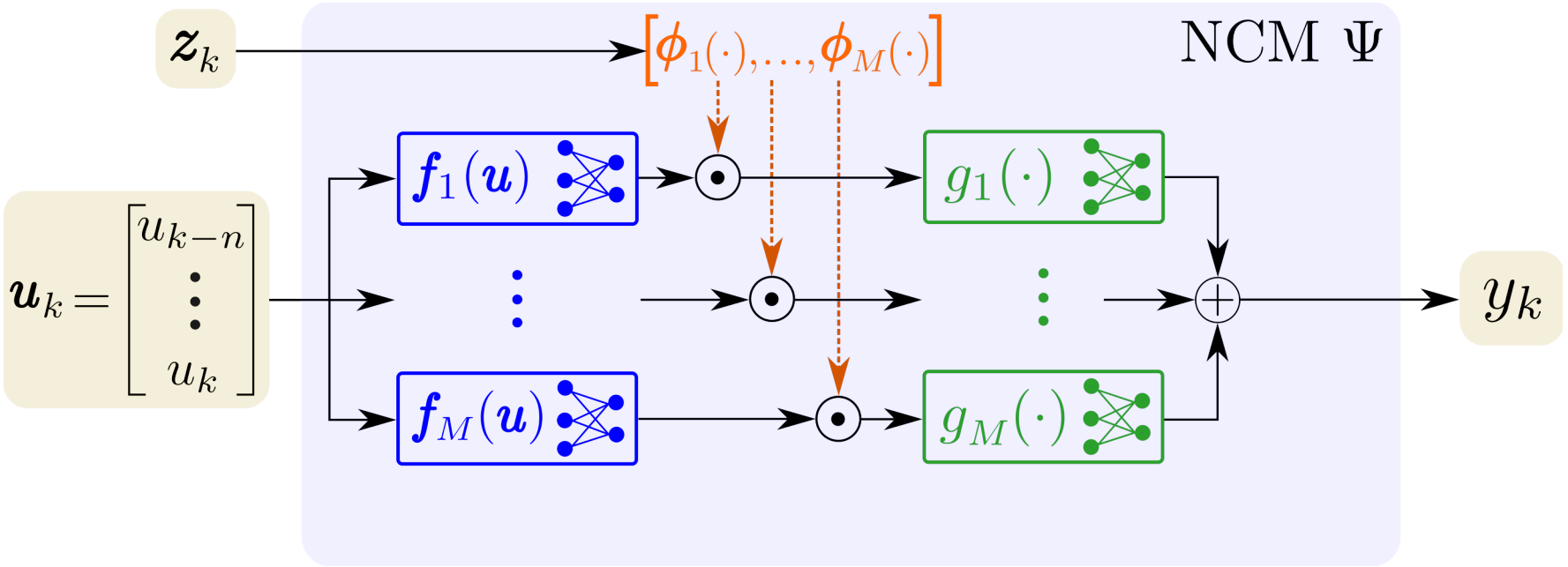


Neural Compositional Modules: Internal Architecture

$$y_k = \sum_{j=1}^{M} [\boldsymbol{f}_j(\boldsymbol{u}_k) \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$$

What if **no physics priors** are available?

- $\boldsymbol{f}_j(\boldsymbol{u}_k)$ can be a **learnable Taylor expansion** of the unknown nonlinear relation
- … Or a more general-purpose machine learning model

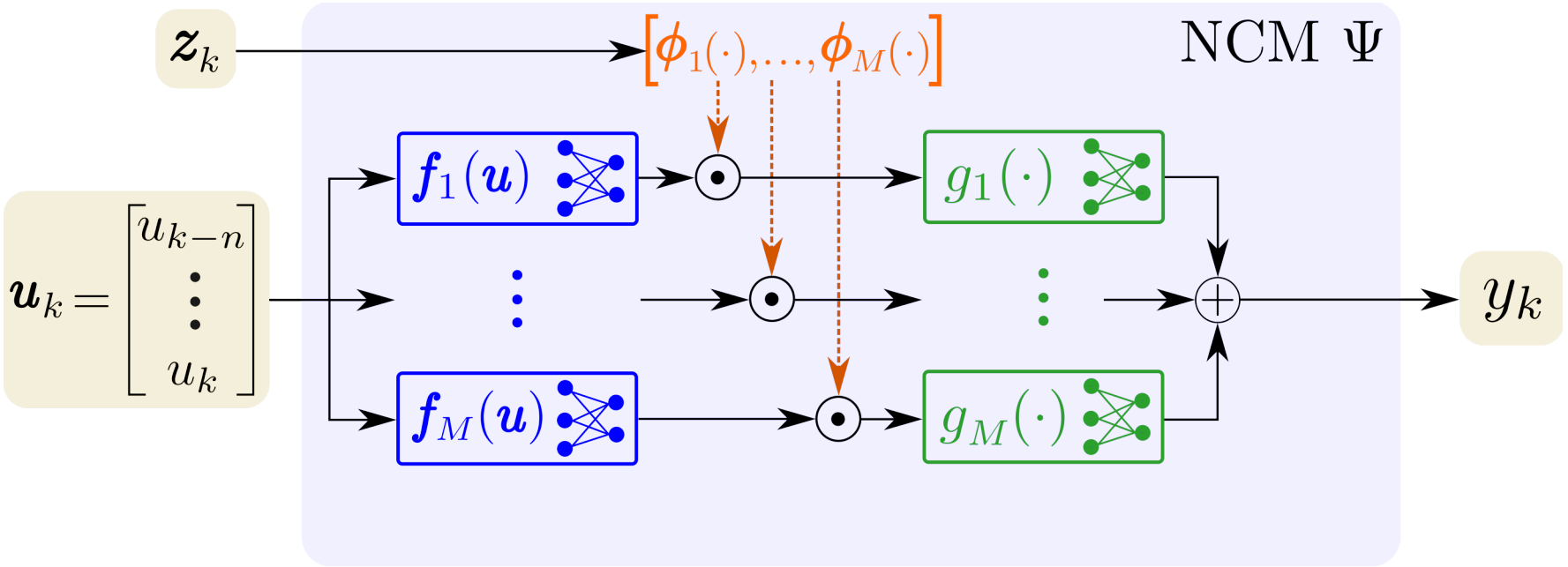


Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***locally linear + input nonlinearities*** :

$$y_k = \sum_{j=1}^{M} [\boldsymbol{f}(\boldsymbol{u}_k) \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j$$



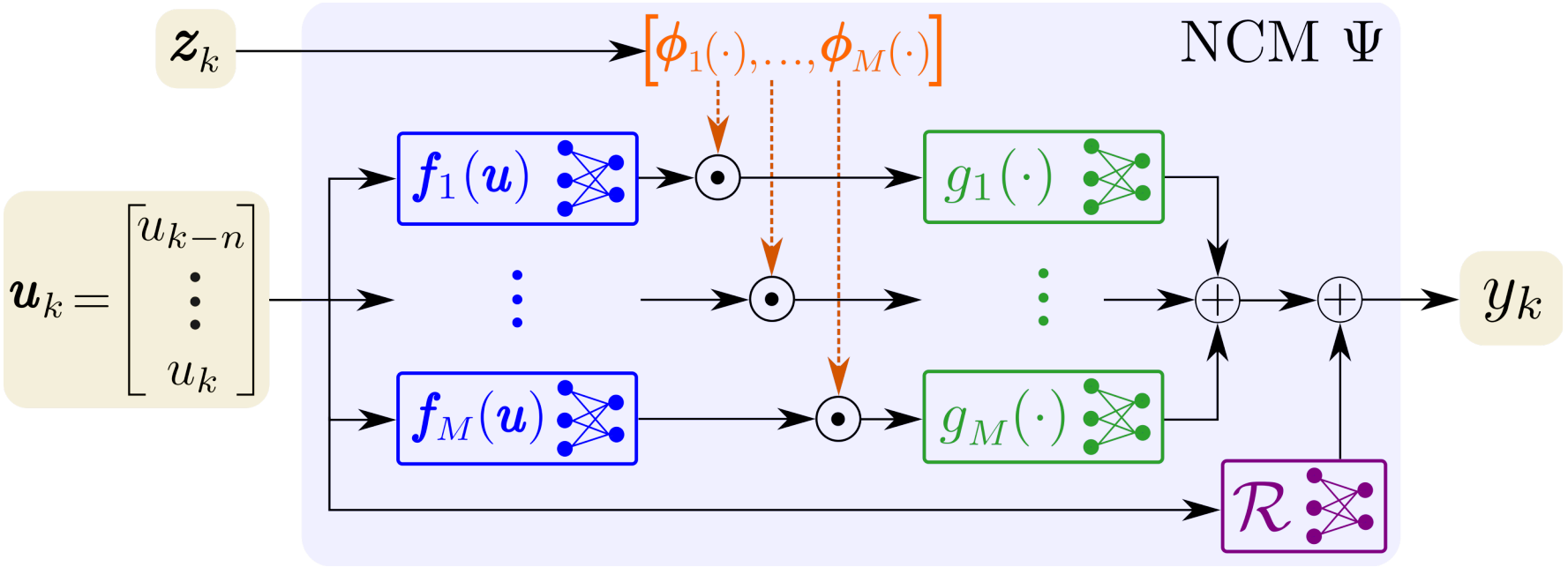
Neural Compositional Modules: Internal Architecture

$$y_k = \boldsymbol{\Psi}(u_k, u_{k-1}, \ldots, u_{k-\mathrm{n}},\ z_k)$$

If the physical system is ***fully nonlinear*** :

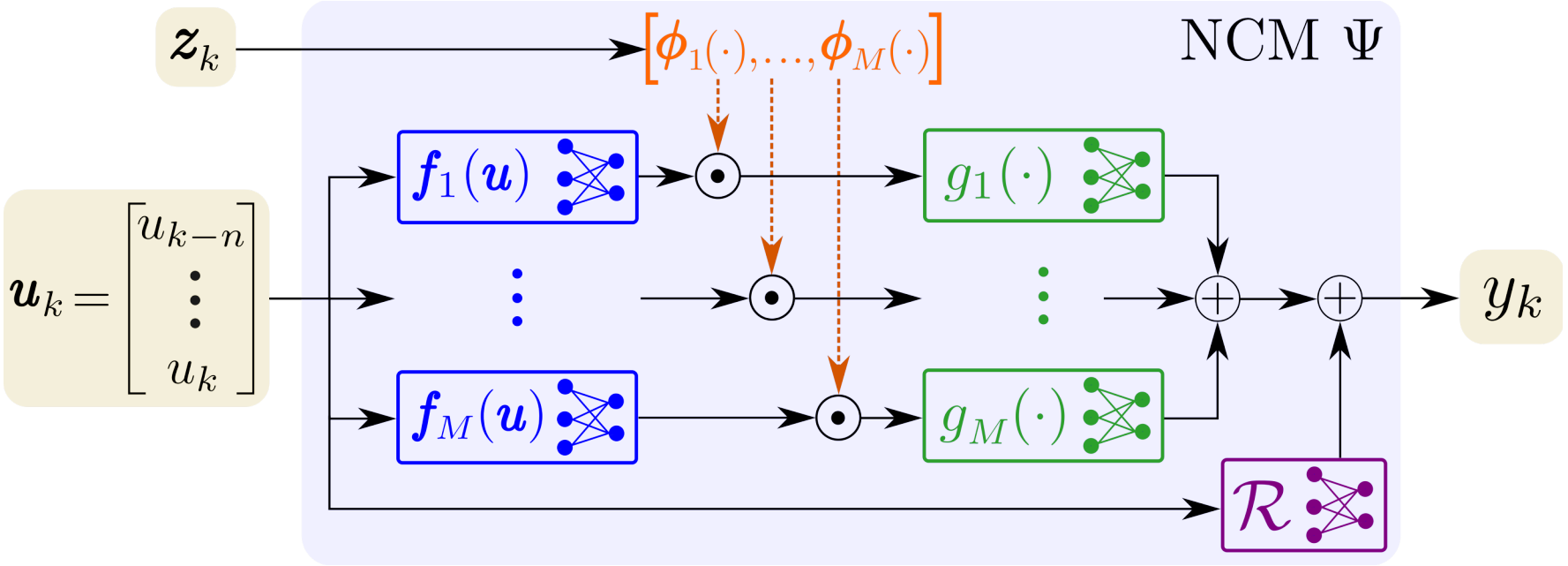
general-purpose NN (residual learning)

$$y_k = \sum_{j=1}^{M} [\boldsymbol{f}(\boldsymbol{u}_k) \odot \boldsymbol{\phi}_j(\boldsymbol{z}_k)] \cdot \boldsymbol{g}_j \quad + \quad \overbrace{\mathcal{R}(\boldsymbol{u}_k)}$$



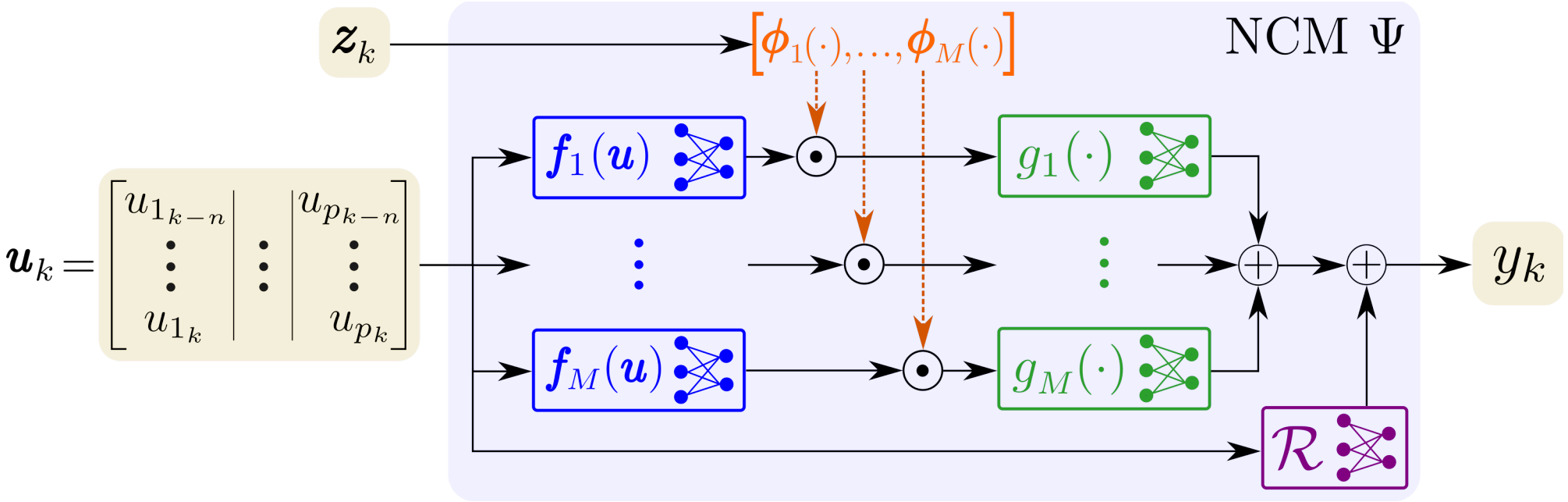
Neural Compositional Modules: Notes and Extensions

- The system can have p **inputs** (e.g., pedals and steering angle for a car)



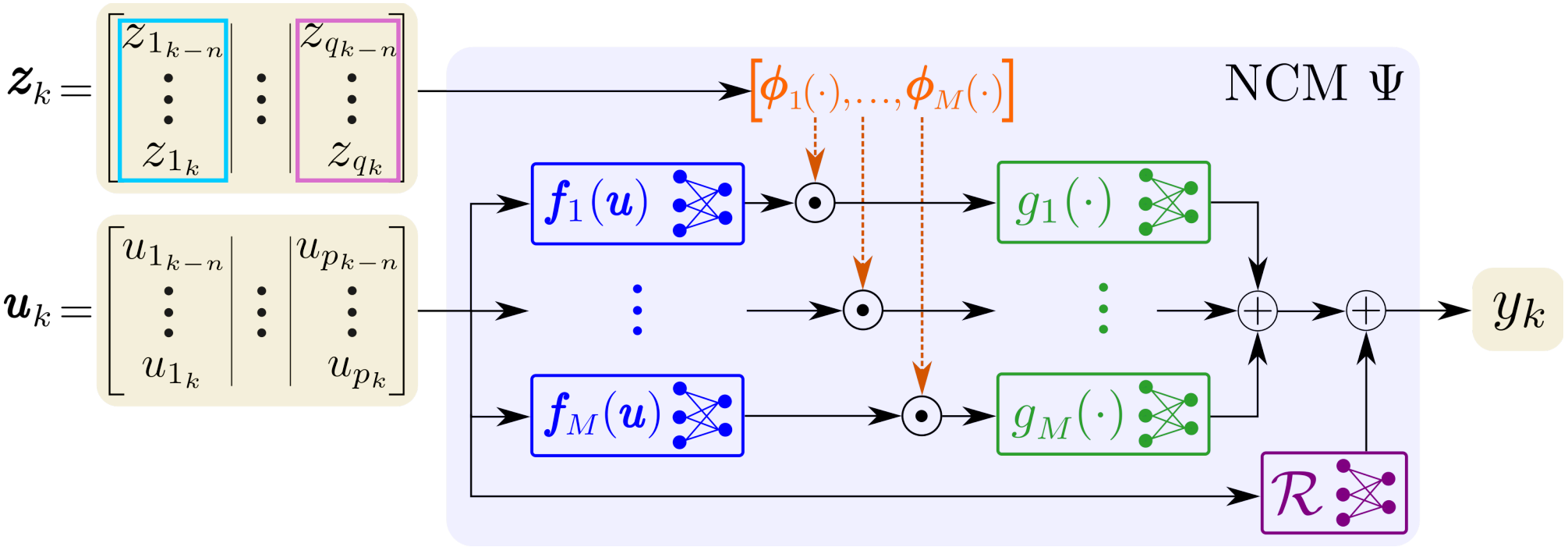
Neural Compositional Modules: Notes and Extensions

- The system can have p **inputs** (e.g., pedals and steering angle for a car)



Neural Compositional Modules: Notes and Extensions

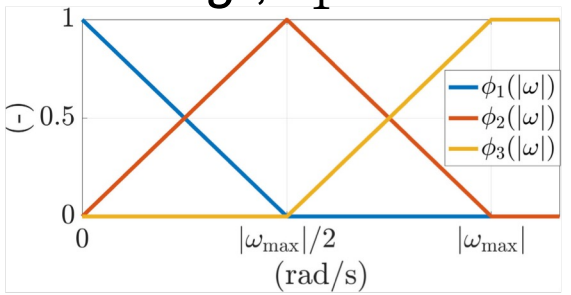
- The system can have p **inputs** (e.g., pedals and steering angle for a car)
- The system can have q **scheduling variables** $\boldsymbol{z}$ (e.g., engine speed and gears)
  - $\boldsymbol{z}_{1_k} = [z_{1_{k-n}}, \dots, z_{1_k}]$, $\boldsymbol{z}_{q_k} = [z_{q_{k-n}}, \dots, z_{q_k}]$, $\boldsymbol{z}_k = [\boldsymbol{z}_{1_k}, \dots, \boldsymbol{z}_{q_k}]$
  - $\boldsymbol{\phi}_1(\boldsymbol{z}_k) = \boldsymbol{\phi}_1(\boldsymbol{z}_{1_k}) \odot \cdots \odot \boldsymbol{\phi}_1(\boldsymbol{z}_{q_k})$ (with $\odot$ denoting the element-wise product)
  - $\boldsymbol{\phi}_M(\boldsymbol{z}_k) = \boldsymbol{\phi}_M(\boldsymbol{z}_{1_k}) \odot \cdots \odot \boldsymbol{\phi}_M(\boldsymbol{z}_{q_k})$ (with M denoting the n. of different operating regimes)

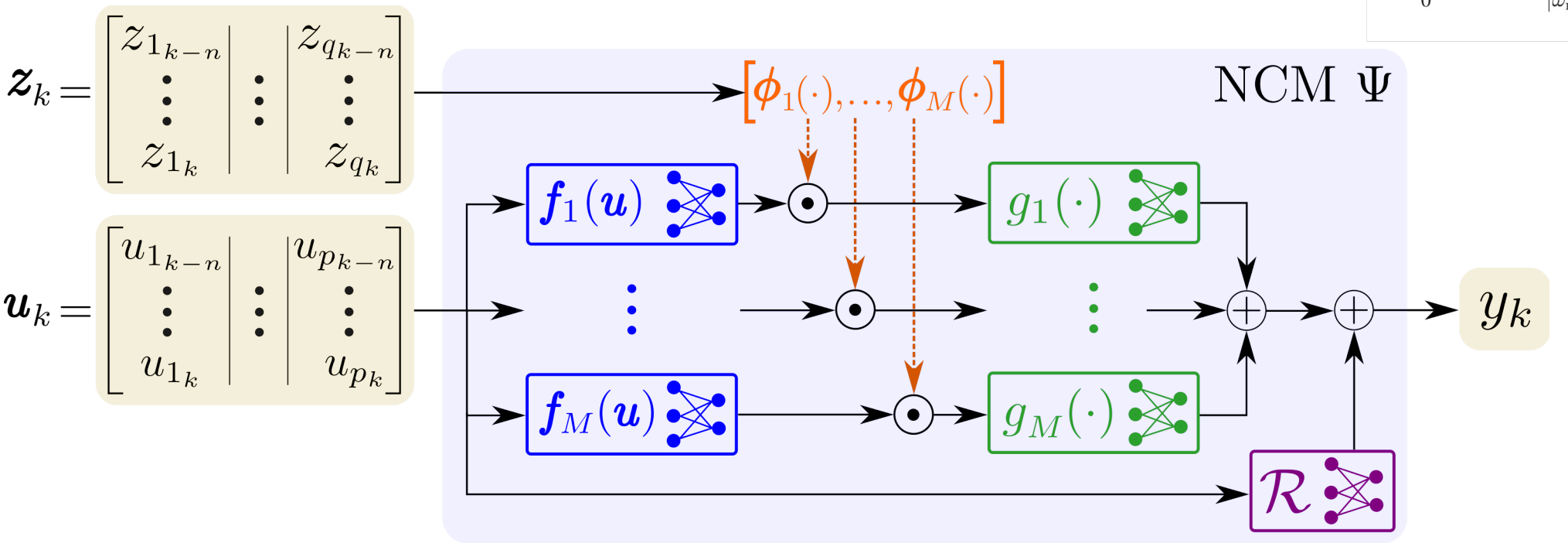


Neural Compositional Modules: Notes and Extensions

- The system can have p **inputs** (e.g., pedals and steering angle for a car)
- The system can have q **scheduling variables** $\boldsymbol{z}$ (e.g., engine speed and gears)
  - $\boldsymbol{z}_{1_k} = [z_{1_{k-n}}, \dots, z_{1_k}]$, $\boldsymbol{z}_{q_k} = [z_{q_{k-n}}, \dots, z_{q_k}]$, $\boldsymbol{z}_k = [\boldsymbol{z}_{1_k}, \dots, \boldsymbol{z}_{q_k}]$
  - $\boldsymbol{\phi}_1(\boldsymbol{z}_k) = \boldsymbol{\phi}_1(\boldsymbol{z}_{1_k}) \odot \cdots \odot \boldsymbol{\phi}_1(\boldsymbol{z}_{q_k})$
  - $\boldsymbol{\phi}_M(\boldsymbol{z}_k) = \boldsymbol{\phi}_M(\boldsymbol{z}_{1_k}) \odot \cdots \odot \boldsymbol{\phi}_M(\boldsymbol{z}_{q_k})$

$e.g.,\ z_1 = \omega$

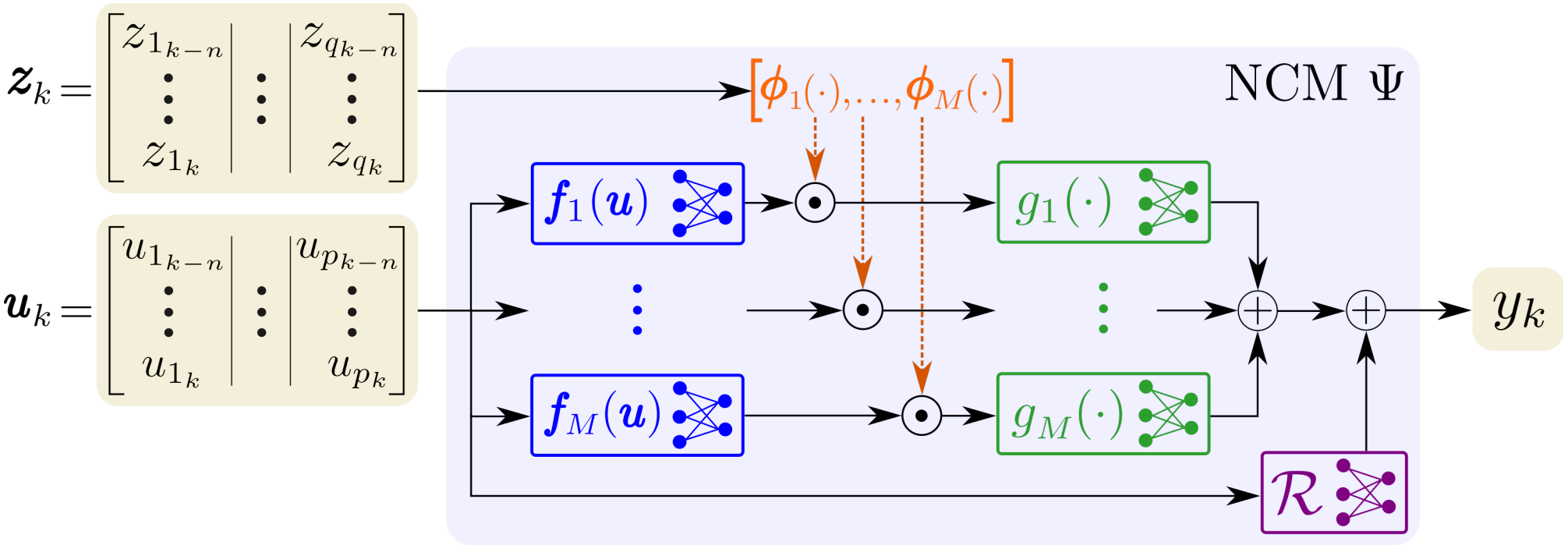




Neural Compositional Modules: Notes and Extensions

- The system can have p **inputs** (e.g., pedals and steering angle for a car)
- The system can have q **scheduling variables** $\boldsymbol{z}$ (e.g., engine speed and gears)
  - $\boldsymbol{z}_{1_k} = [z_{1_{k-n}}, \dots, z_{1_k}]$, $\boldsymbol{z}_{q_k} = [z_{q_{k-n}}, \dots, z_{q_k}]$, $\boldsymbol{z}_k = [\boldsymbol{z}_{1_k}, \dots, \boldsymbol{z}_{q_k}]$
  - $\boldsymbol{\phi}_1(\boldsymbol{z}_k) = \boldsymbol{\phi}_1(\boldsymbol{z}_{1_k}) \odot \cdots \odot \boldsymbol{\phi}_1(\boldsymbol{z}_{q_k})$
  - $\boldsymbol{\phi}_M(\boldsymbol{z}_k) = \boldsymbol{\phi}_M(\boldsymbol{z}_{1_k}) \odot \cdots \odot \boldsymbol{\phi}_M(\boldsymbol{z}_{q_k})$

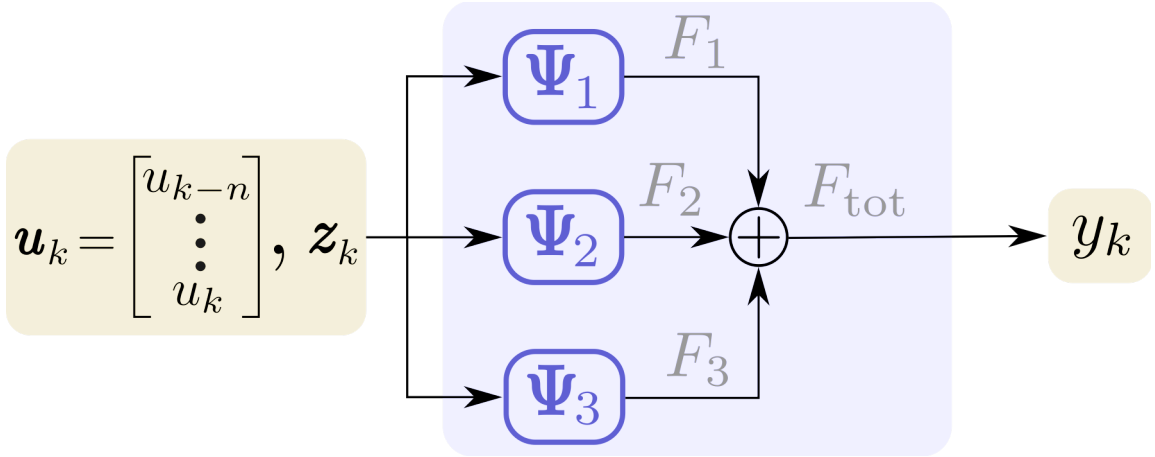
It still holds that $\sum_{j=1}^{M} \boldsymbol{\phi}_j(\boldsymbol{z}_k) = 1$, $\boldsymbol{\phi}_j(\boldsymbol{z}_k) \geq 0 \ \forall \boldsymbol{z}_k$



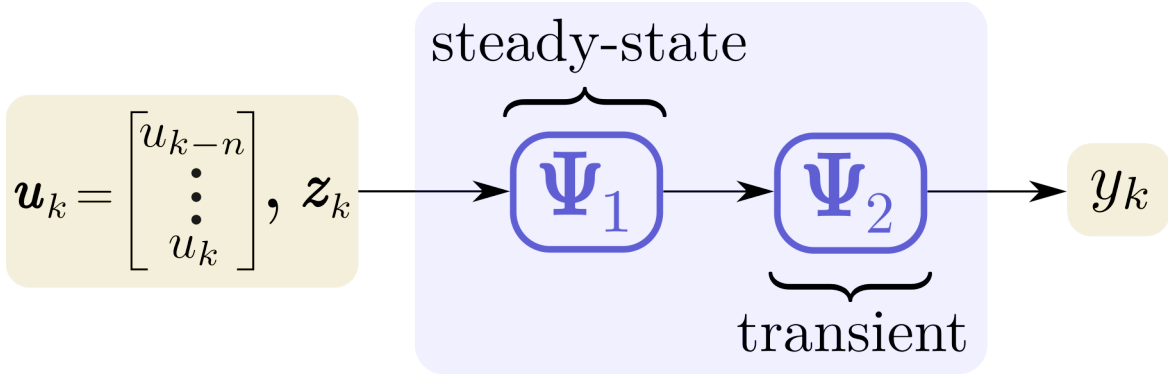
Why Combining Multiple Neural Compositional Modules?

Idea: Each NCM Ψ_i can be ***specialized*** to learn a sub-system / part of the dynamics

Example 1: Learning the effect of independent forces and moments



Example 2: Learning the steady-state and transient dynamics of a system



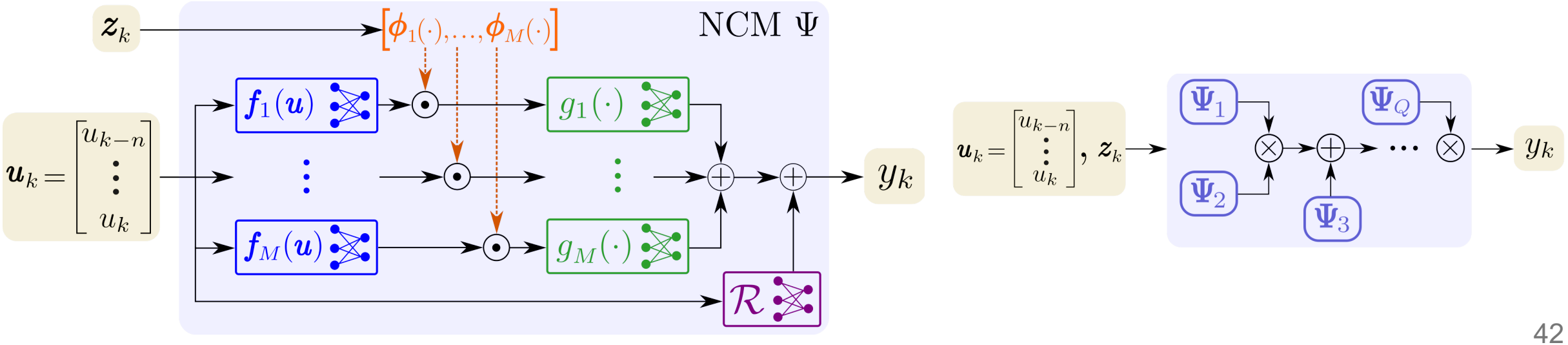
Pros and Cons of Neural Compositional Modules

Pros:

- The trained weights of the $\boldsymbol{f}_j(\cdot)$ and $\boldsymbol{g}_j(\cdot)$ may be physically interpreted
- Split task complexity in different NCMs $\boldsymbol{\Psi}$
- Better generalization and sample efficiency than black-box ML models

Cons:

- Some design effort is still required in the $\boldsymbol{f}_j(\cdot)$ modules



Questions & Discussion

Funded with the contribution of the Ministry of University and Research pursuant to Decree No. 1236 of 01/08/2023 – FIS 2 Call; project FIS-2023-03684 'Neu4mes', CUP: E53C24003800001