Deep Recurrent Neural Network Control Applied in Solar Inverters for Grid Integration

Dr. Xingang Fu

Department of Electrical Engineering and Computer Science
Texas A&M University-Kingsville, USA
Email : fuxingang@gmail.com

TEXAS A&M UNIVERSITY KINGSVILLE

Outline

I. Background
II. Deep Recurrent Neural Network Controller
III. Trajectory based Training
IV. Local Stability and Local Convergence
V. Hardware-in-the-loop Experiment
VI. Discussions about RNN Control

Outline

I. **Background**

II. Deep Recurrent Neural Network Controller

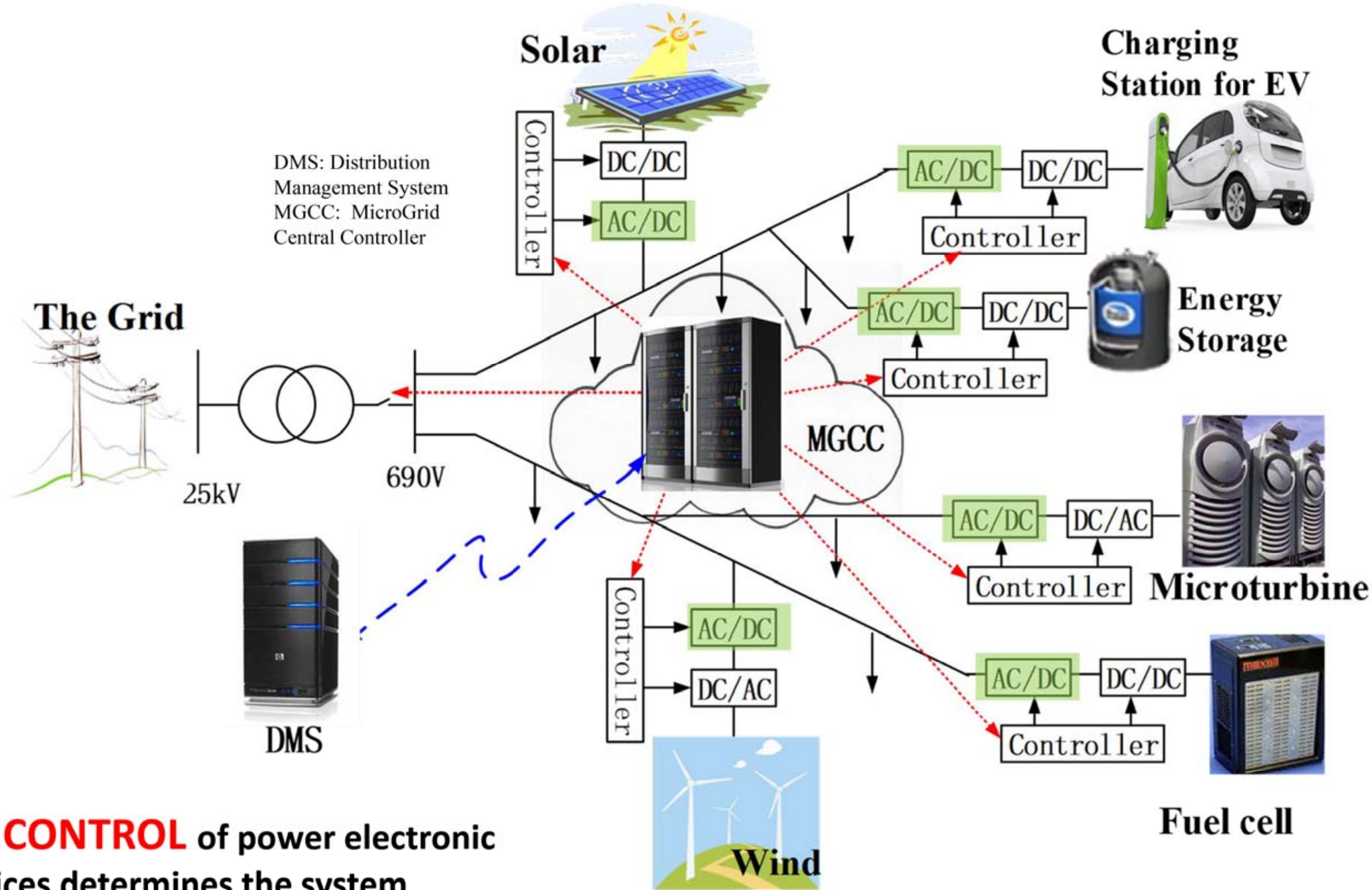
III. Trajectory based Training

IV. Local Stability and Local Convergence

V. Hardware-in-the-loop Experiment

VI. Discussions about RNN Control

Smart Grid



The **CONTROL** of power electronic devices determines the system performance to a **GREAT** degree.

TEXAS A&M UNIVERSITY KINGSVILLE

California's Rooftop Solar Mandate Wins Final Approval

The California Building Standards Commission has signed off on the residential solar mandate—a first of its kind for the nation.

JULIA PYPER | DECEMBER 05, 2018

The solar mandate marks a groundbreaking development for the clean energy sector.

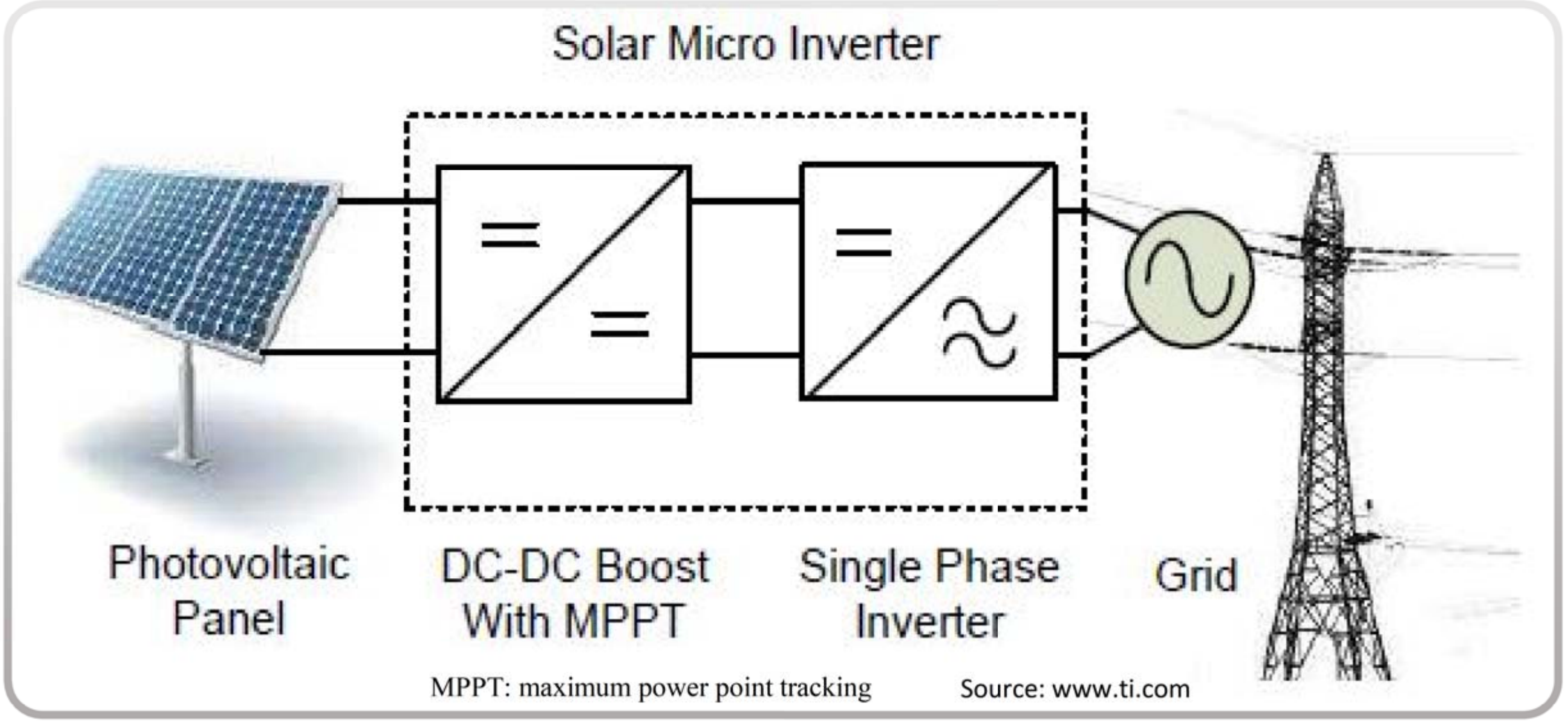
353

It's official. All new homes in California must incorporate solar power starting in 2020.

TOP ARTICLES

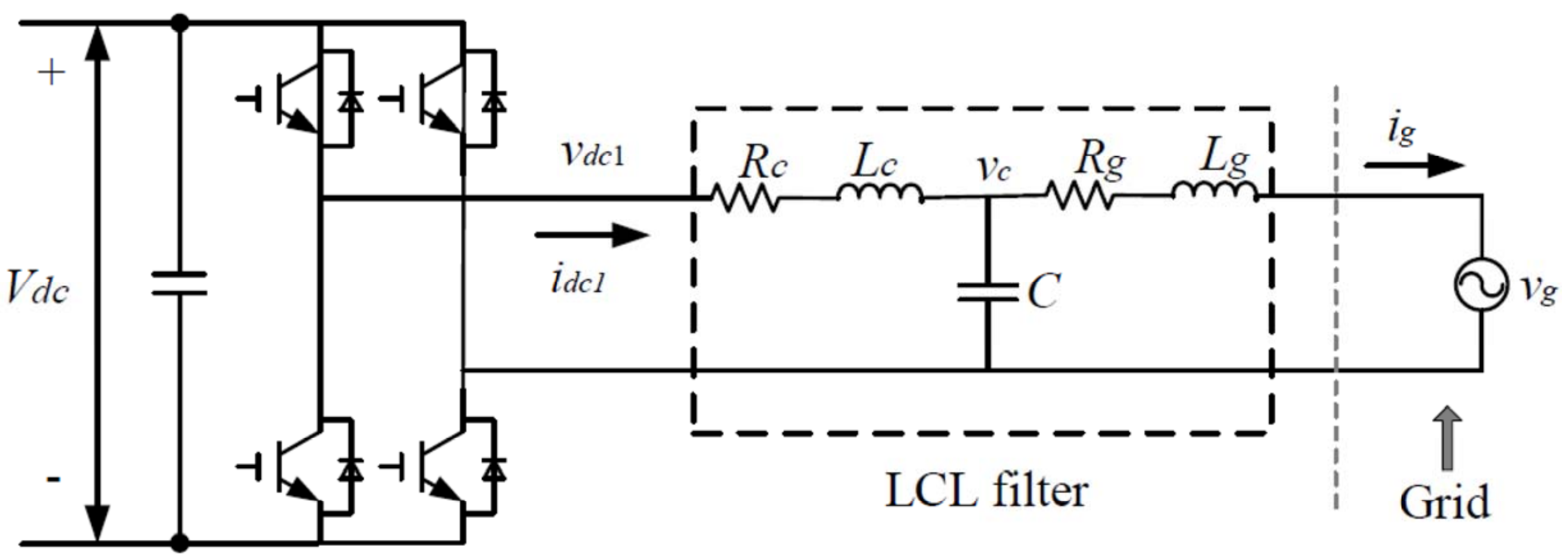
MOST POPULAR MOST COMMENTS

Solar Microinverter



- A **Solar Microinverter** converts DC generated by the single solar module to AC, and generally is connected to the grid. It allows two-way power flow.

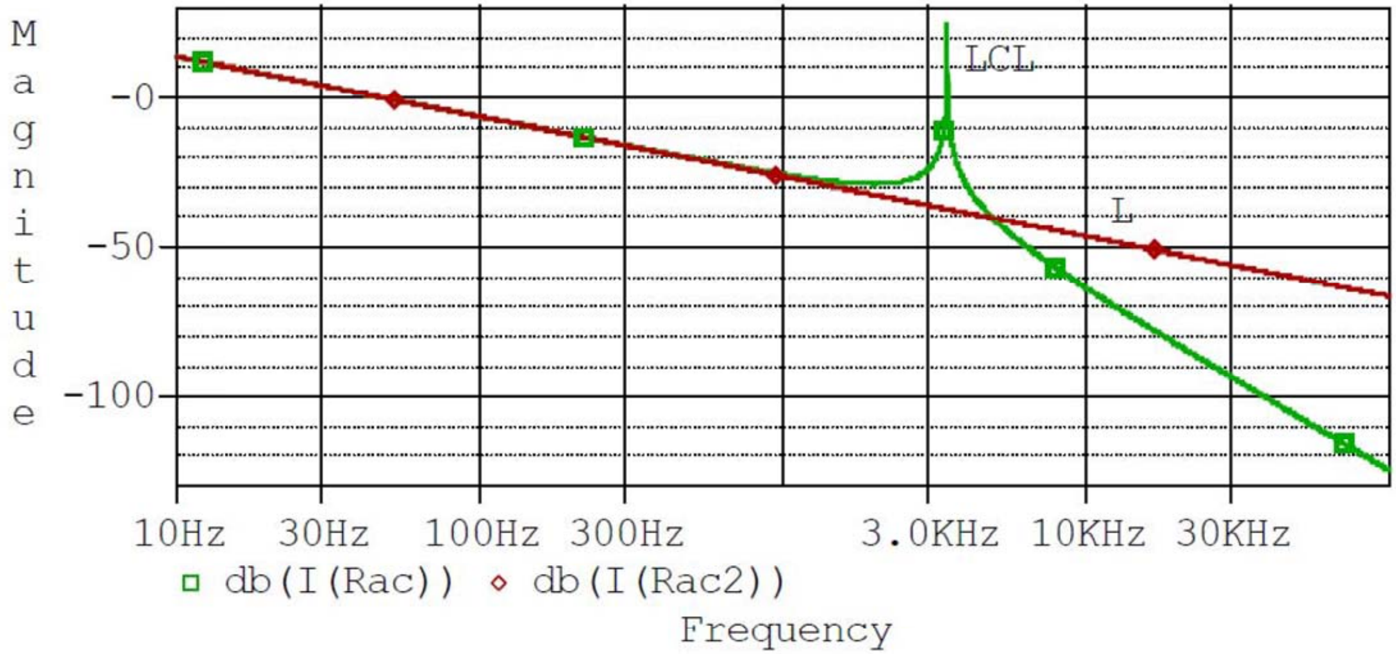
Single-phase Solar Grid-tied Inverter/ Converter



- Single phase inverter is for home application.
- To reduce harmonics, the LCL filter is a wise choice and generally used.

Resonant Problem

❖ However, as the LCL filter has a **Resonant Problem** and tends to be unstable.



*That **Peak** point (resonance frequency) causes the unstable problem and control difficulty.*

❖ ***The control of the LCL filter is a challenge!***

✓ Many damping / control methods have been proposed to handle this.

Outline

I. Background

II. Deep Recurrent Neural Network Controller

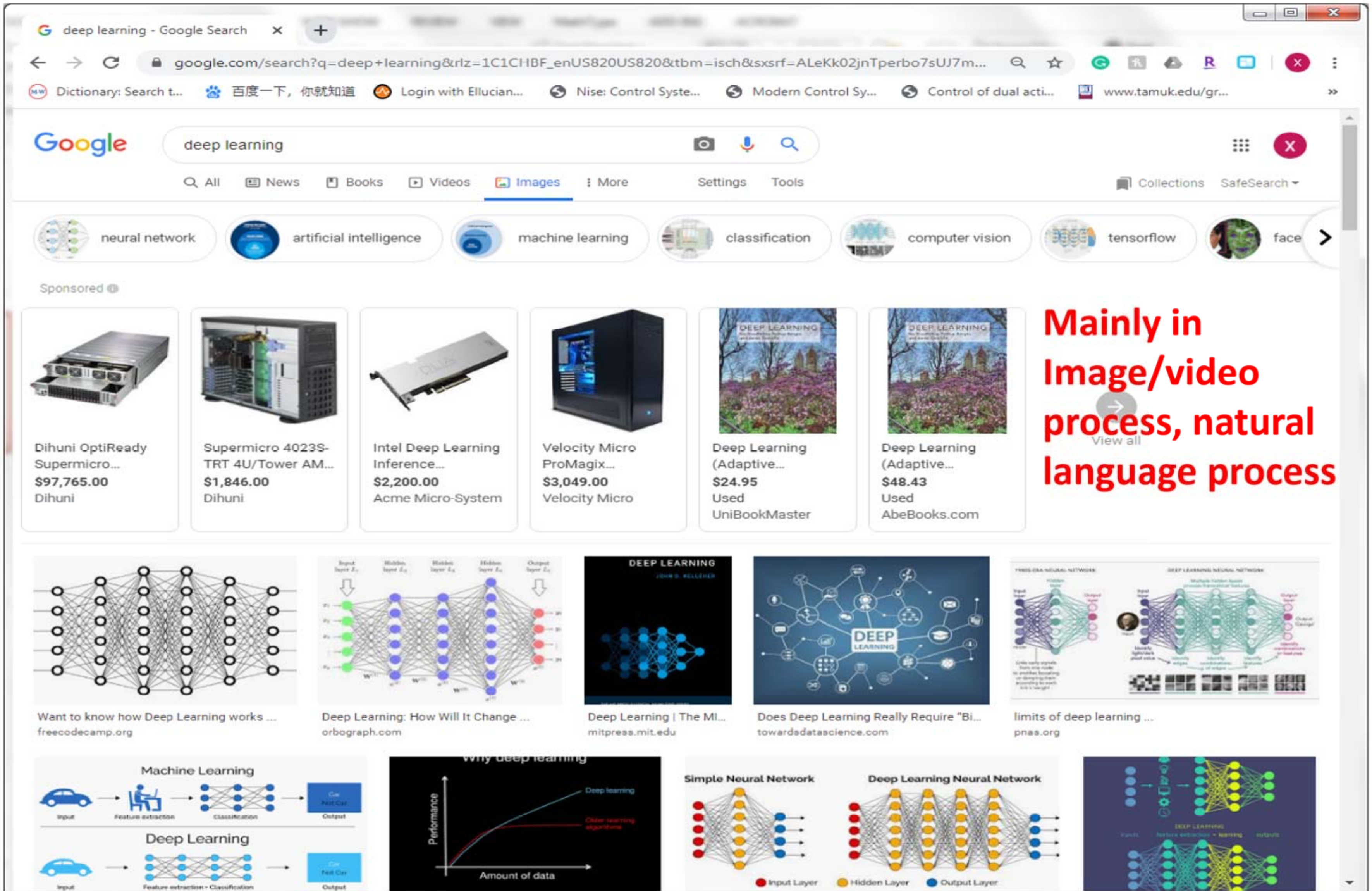
III. Trajectory based Training

IV. Local Stability and Local Convergence

V. Hardware-in-the-loop Experiment

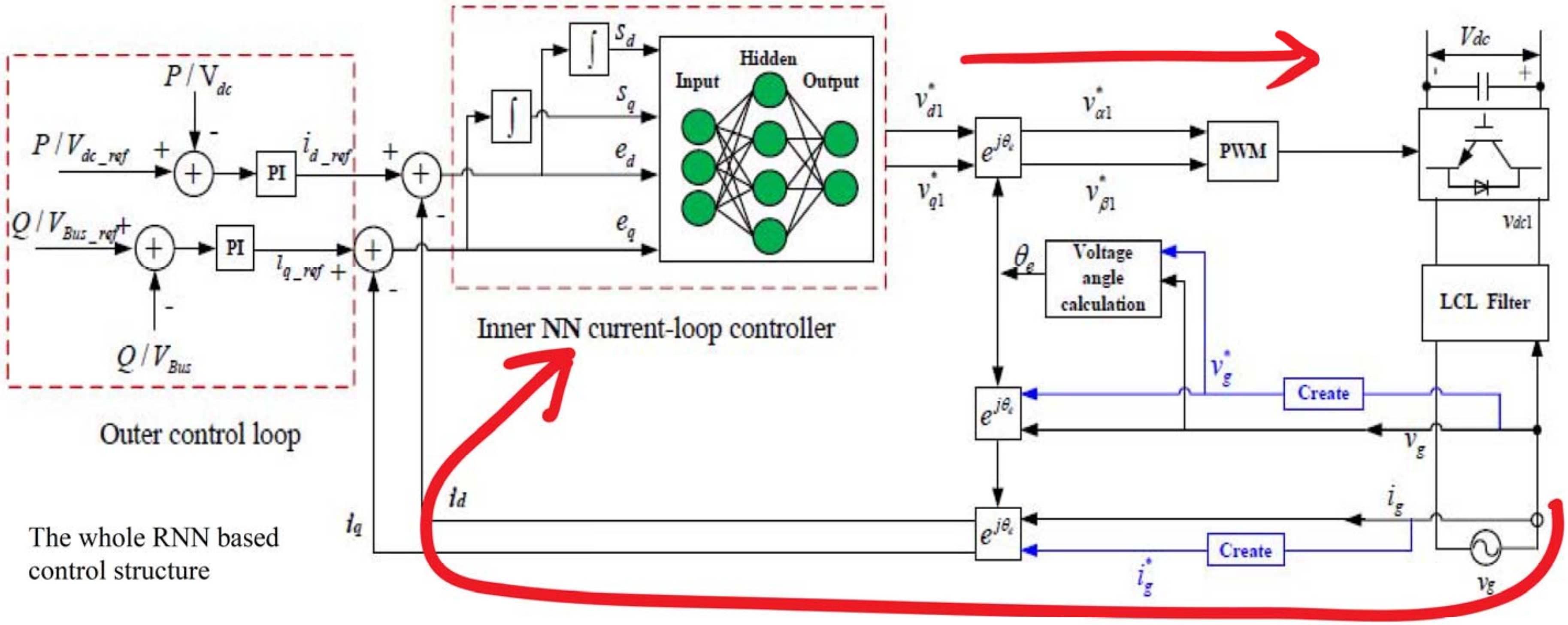
VI. Discussions about RNN Control

Deep Learning



Mainly in Image/video process, natural language process

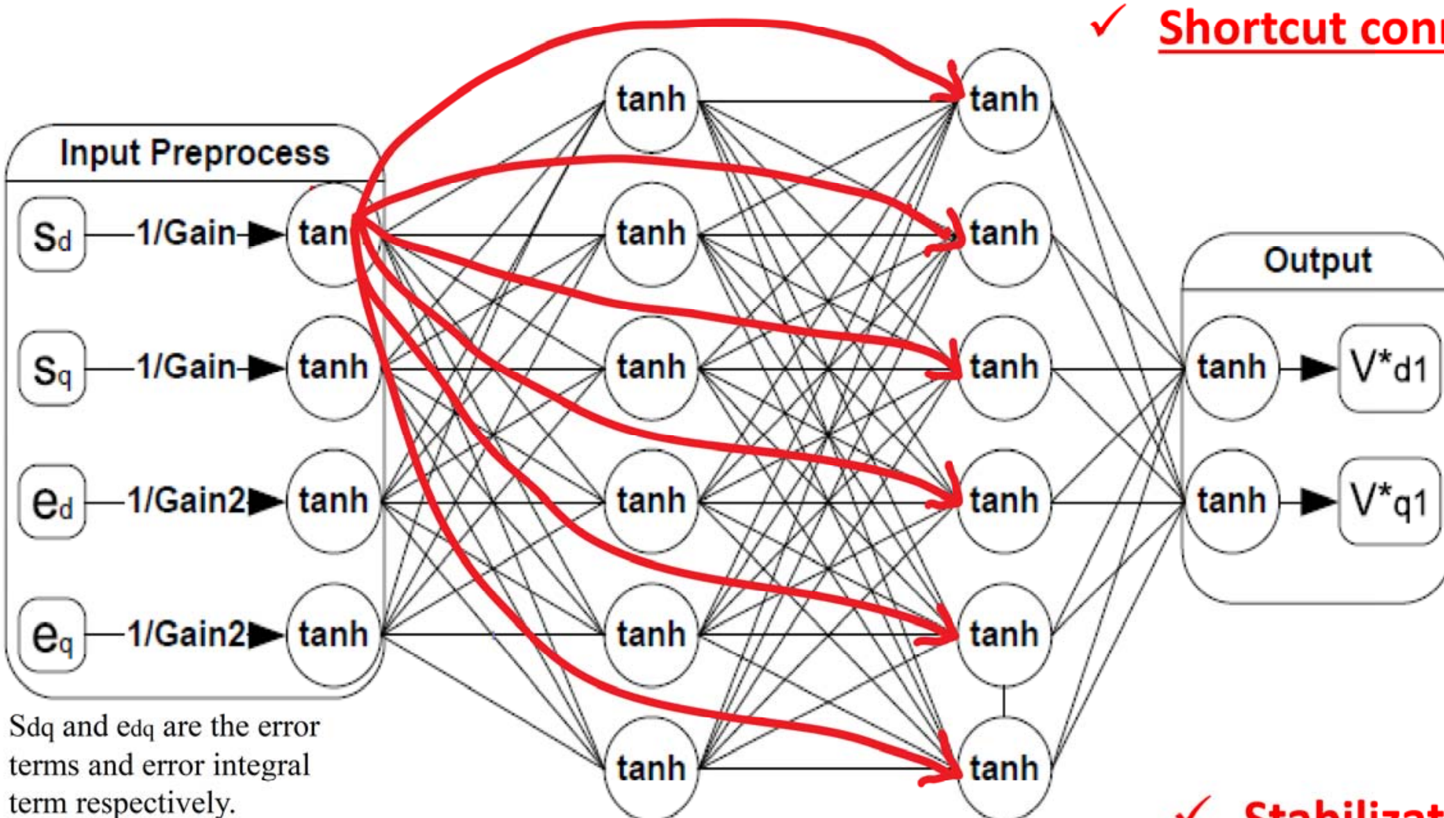
Proposed *Neural Network* in *Feedback Control Loops*



Recurrent Neural Network

Feature - *Error Integral*

✓ Shortcut connections



Sdq and edq are the error terms and error integral term respectively.

✓ Stabilization matrix added to the output

✓ No steady-state tracking error.

✓ This structure succeeded in the hardware-in-the-loop experiments

Outline

I. Background

II. Deep Recurrent Neural Network Controller

III. Trajectory based Training

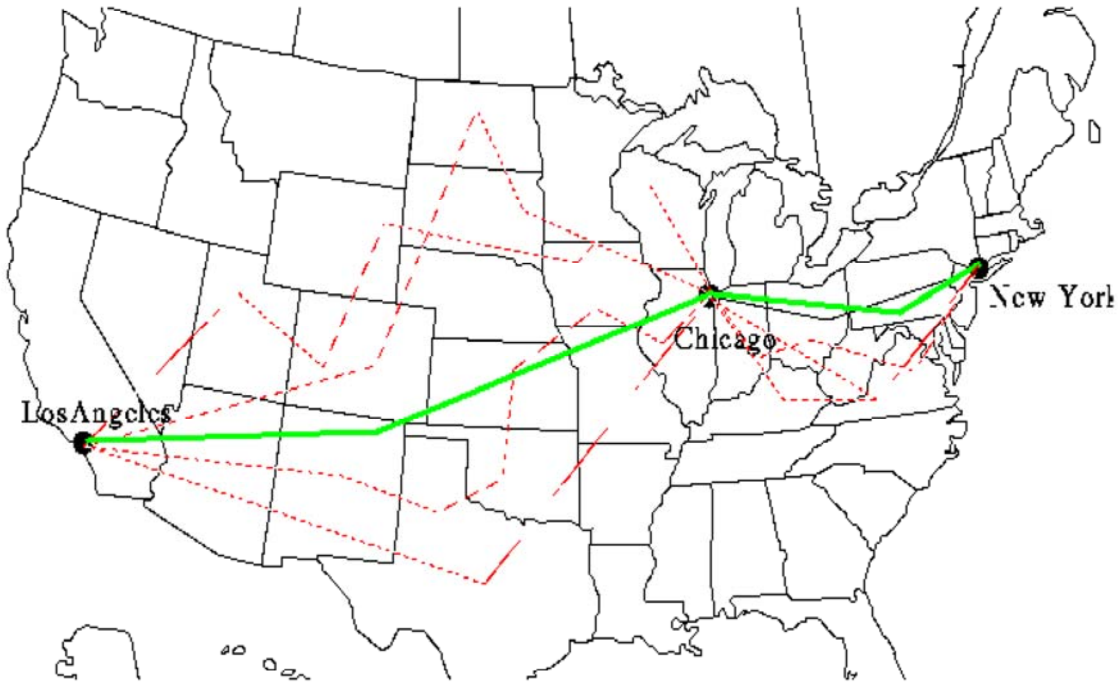
IV. Local Stability and Local Convergence

V. Hardware-in-the-loop Experiment

VI. Discussions about RNN Control

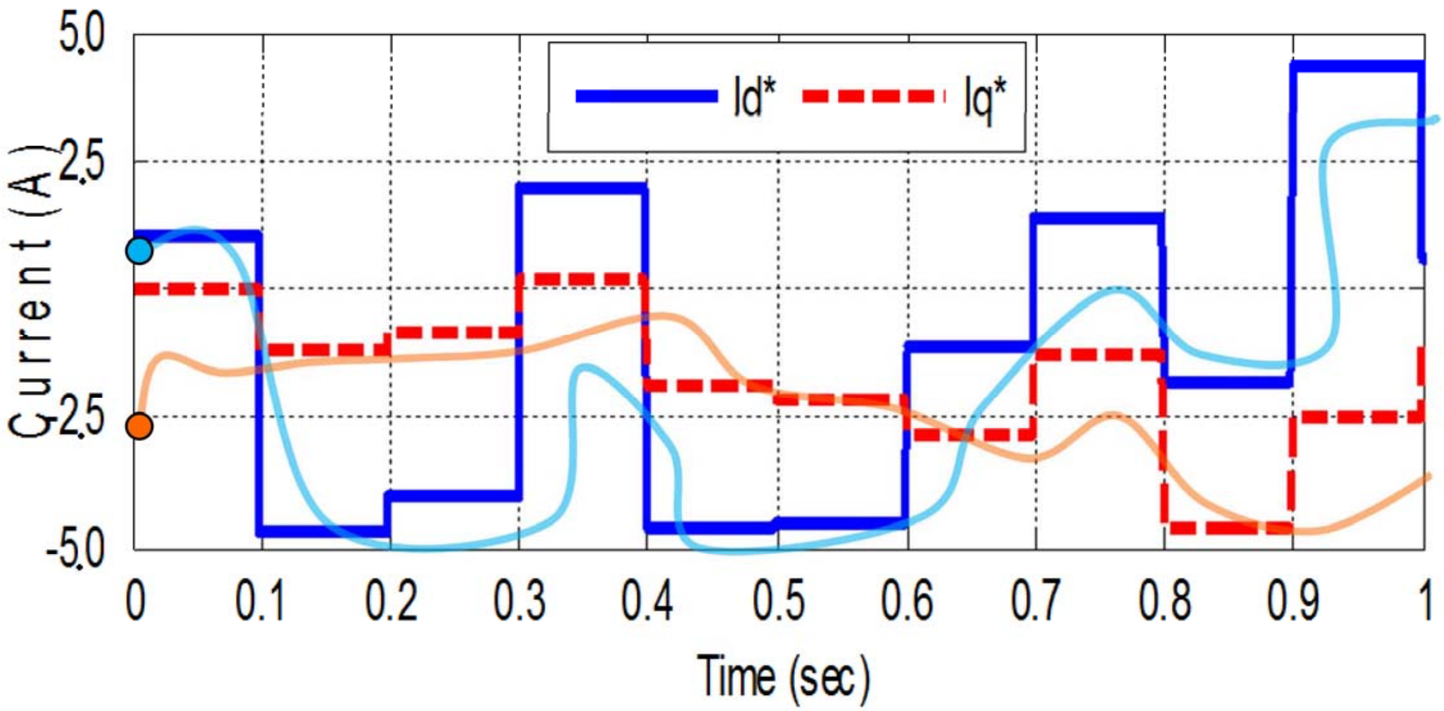
Approximate Dynamic Programming

- **Dynamic programming** (DP) employs Bellman's optimality principle and provides powerful tool for solving optimization,



- DP can is also powerful for **OPTIMAL CONTROL** problems, e.g. Adaptive Critic Designs (ACDs) is one class of **Approximate Dynamic Programming** to approximate optimal control.

Approximate Dynamic Programming Based Trajectory Training

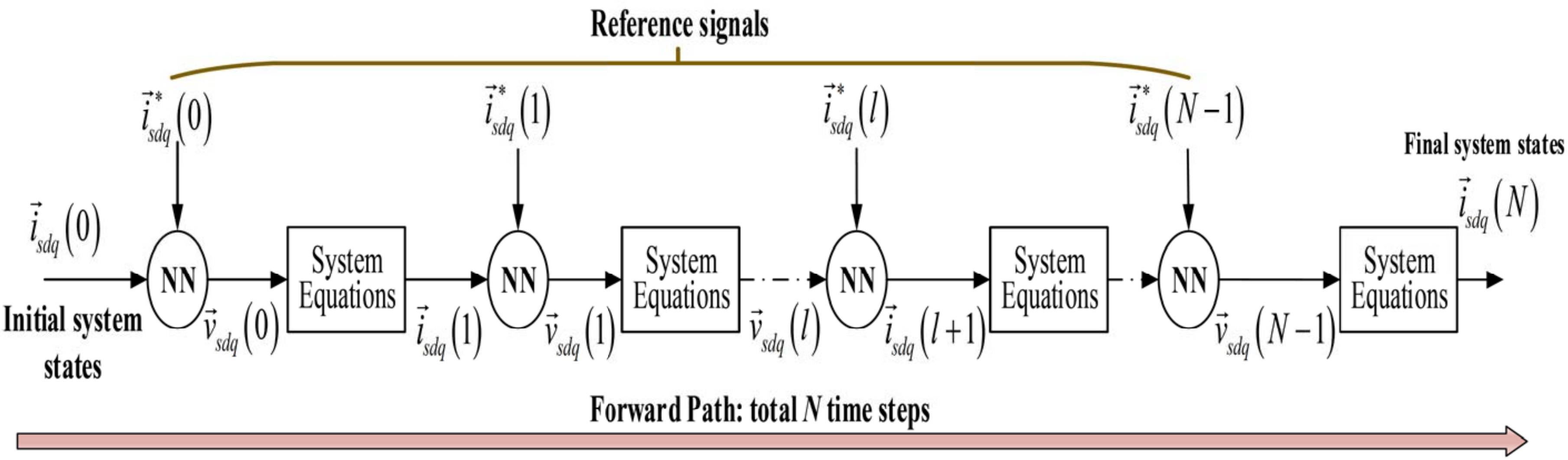


Note that: before training, the NN controller cannot adjust the currents tc follow the reference.

- ✓ *The goal of training is to adjust the actual currents to follow the reference current trajectories through minimizing the cost function based on the* ***Approximate Dynamic Programming principle.***

$$C_{dp}=\sum_{k=j}^{\infty}\gamma^{k-j}U(\overrightarrow{e_{dq}}(k)) \ =\sum_{k=j}^{\infty}\gamma^{k-j}\sqrt{[i_d(k)-i_{d\_ref}(k)]^2+[i_q(k)-i_{q\_ref}(k)]^2}$$

Deep *Network*



- In the training, the NN needs to be expanded along the trajectory.
- Think of this process as a big network, for a 1000-step calculation, the total number of NN layers will be $4\times1000= 4000$.
  - ✓ it is a large and deep network, which requires lots of calculation.

Forward Accumulation Through Time Algorithm

Algorithm 3.2 FATT algorithm to calculate the Jacobian matrix.

$C \leftarrow 0, \overrightarrow{e_{dq}}(0) \leftarrow 0, \overrightarrow{s_{dq}}(0) \leftarrow 0, \frac{\partial \overrightarrow{i_{dq}}(0)}{\partial \overrightarrow{w}} \leftarrow 0, \frac{\partial \overrightarrow{\phi_{dq}}(0)}{\partial \overrightarrow{w}} \leftarrow 0$

{Calculate the Jacobian matrix $J(\overrightarrow{w})$}

for $k = 0$ to $N-1$ **do**

$\quad \overrightarrow{u_{dq}}(k) \leftarrow k_{PWM} R(\overrightarrow{e_{dq}}(k), \overrightarrow{s_{dq}}(k), \overrightarrow{w}) - \overrightarrow{v_{dq}}$

$\quad \frac{\partial \overrightarrow{s_{dq}}(k)}{\partial \overrightarrow{w}} \leftarrow T_s \left[\frac{\partial \overrightarrow{\phi_{dq}}(k)}{\partial \overrightarrow{w}} - \frac{1}{2} \frac{\partial \overrightarrow{i_{dq}}(k)}{\partial \overrightarrow{w}} \right]$

$\quad \frac{\partial \overrightarrow{u_{dq}}(k)}{\partial \overrightarrow{w}} \leftarrow k_{PWM} \left[\frac{\partial R(k)}{\partial \overrightarrow{w}} + \frac{\partial R(k)}{\partial \overrightarrow{e_{dq}}(k)} \frac{\partial i_{dq}(k)}{\partial \overrightarrow{w}} + \frac{\partial R(k)}{\partial \overrightarrow{s_{dq}}(k)} \frac{\partial \overrightarrow{s_{dq}}(k)}{\partial \overrightarrow{w}} \right]$

$\quad \frac{\partial \overrightarrow{i_{dq}}(k+1)}{\partial \overrightarrow{w}} \leftarrow A \frac{\partial \overrightarrow{i_{dq}}(k)}{\partial \overrightarrow{w}} + B \frac{\partial \overrightarrow{u_{dq}}(k+1)}{\partial \overrightarrow{w}}$

$\quad \overrightarrow{i_{dq}}(k+1) \leftarrow A \overrightarrow{i_{dq}}(k) + B \overrightarrow{u_{dq}}(k)$

$\quad \overrightarrow{e_{dq}}(k+1) \leftarrow \overrightarrow{i_{dq}}(k+1) - \overrightarrow{i_{dq\_ref}}(k+1)$

$\quad \overrightarrow{s_{dq}}(k+1) \leftarrow \overrightarrow{s_{dq}}(k) + \frac{T_s}{2} [\overrightarrow{e_{dq}}(k) + \overrightarrow{e_{dq}}(k+1)]$

$\quad C \leftarrow C + U(\overrightarrow{e_{dq}}(k+1))$ {accumulate DP cost}

$\quad \frac{\partial \overrightarrow{\phi}(k+1)}{\partial \overrightarrow{w}} \leftarrow \frac{\partial \overrightarrow{\phi}(k)}{\partial \overrightarrow{w}} + \frac{\partial \overrightarrow{i_{dq}}(k+1)}{\partial \overrightarrow{w}}$

$\quad \frac{\partial V(k+1)}{\partial \overrightarrow{w}} \leftarrow \frac{\partial V(k+1)}{\partial \overrightarrow{e_{dq}}(k+1)} \frac{\partial \overrightarrow{i_{dq}}(k+1)}{\partial \overrightarrow{w}}$

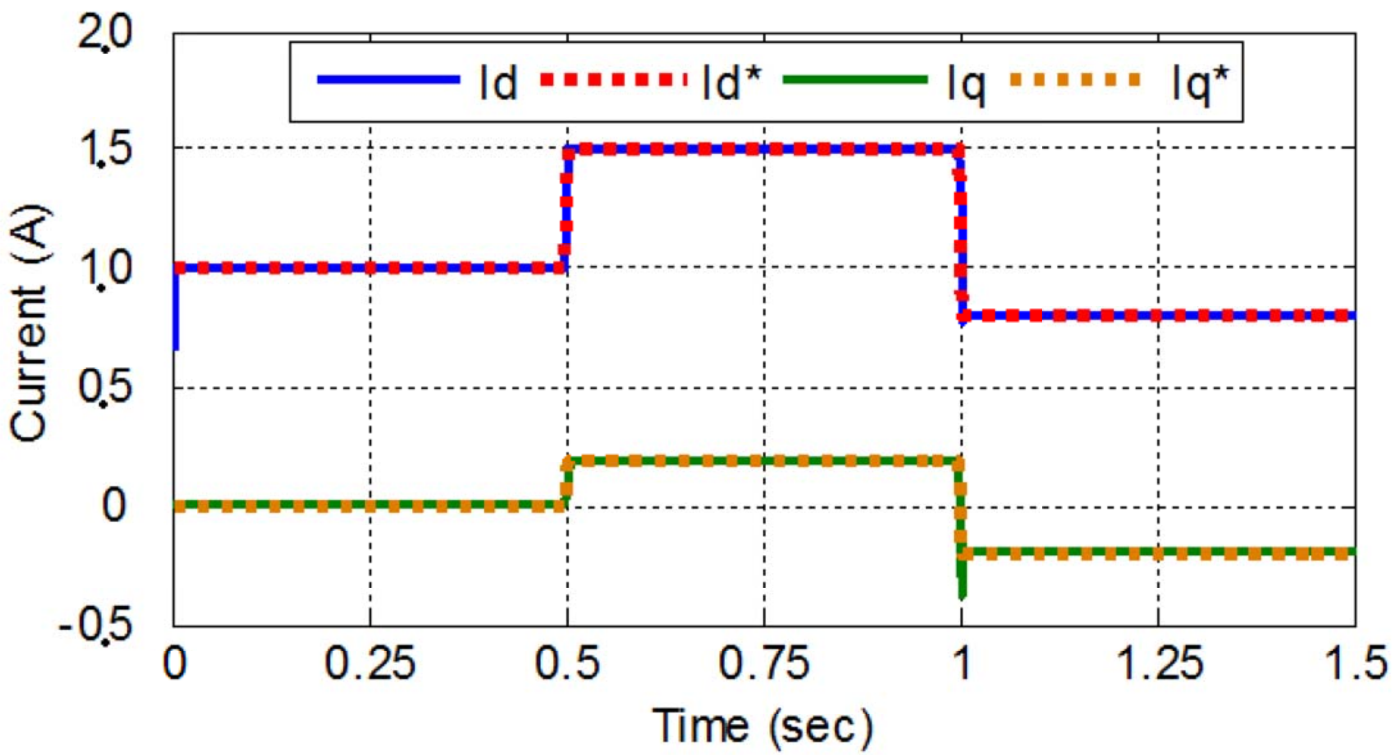
$\quad$ the $(k+1)th$ row of $J(\overrightarrow{w}) \leftarrow \frac{\partial V(k+1)}{\partial \overrightarrow{w}}$

end for

{On exit, the Jacobian matrix $J(\overrightarrow{w})$ is finished for the whole trajectory.}

Introducing of error integral terms brings difficulties in calculating gradients

Well-trained ***RNN Controller***



The actual currents can ***follow the reference trajectories*** *quite well!*

The detailed training algorithm: **LM** (Levenberg–Marquardt) + **FATT** (Forward Accumulation Through Time)

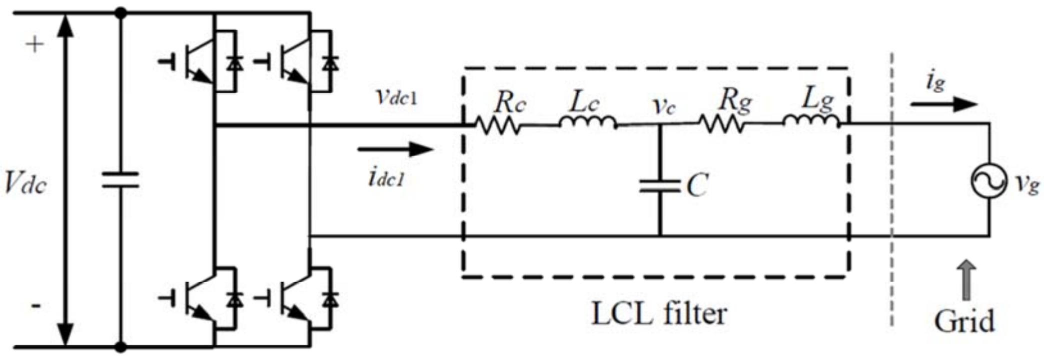
Outline

IV. Local Stability and Local Convergence

System Dynamic Equation

- **State-space model**

Ignoring the capacitor, the **two-dimensional** system equation will be simplified as



$$\frac{d}{dt}\underbrace{\begin{bmatrix} i_d \\ i_q \end{bmatrix}}_{i_{dq}} = A\begin{bmatrix} i_d \\ i_q \end{bmatrix} + B\underbrace{\begin{bmatrix} u_d \\ u_q \end{bmatrix}}_{u_{dq}}$$

where $A = -\begin{bmatrix} \frac{R_g + R_c}{L_g + L_c} & -\omega_s \\ -\omega_s & \frac{R_g + R_c}{L_g + L_c} \end{bmatrix}$,

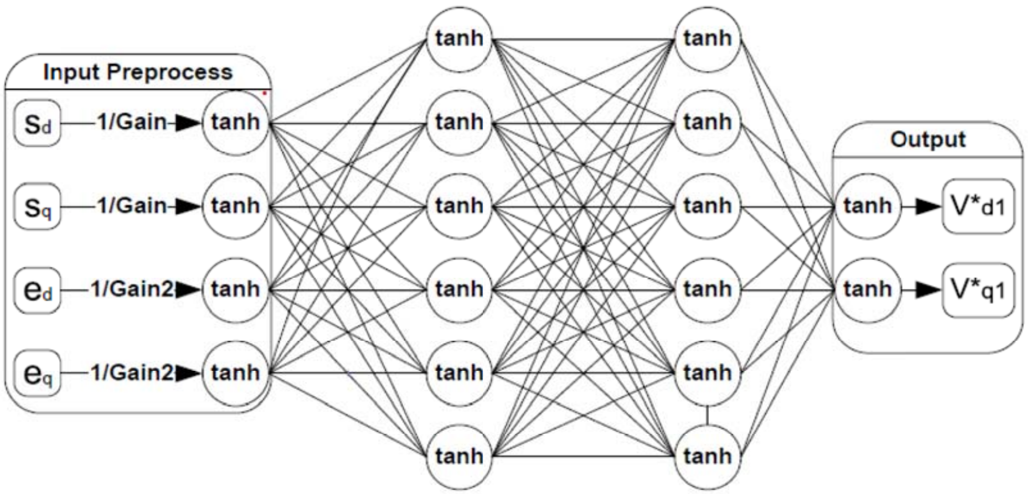
$$B = -\begin{bmatrix} \frac{1}{L_g + L_c} & \\ & \frac{1}{L_g + L_c} \end{bmatrix}$$

Control vector:

$$u_{dq} = R(e_{dq}, s_{dq}) = K_{pwm}N(e_{dq}, s_{dq}, w) - v_{dq}$$

Function format of RNN controller

- **NN function representation**



$$N\left(e_{dq}, s_{dq}, w\right) = \tanh\left(w_3\begin{bmatrix}\tanh\left(w_2\begin{bmatrix}\tanh\left(w_1\begin{bmatrix}\tanh\begin{bmatrix}\frac{e_{dq}}{Gain}\\ \frac{s_{dq}}{Gain2}\end{bmatrix}\\ -1\end{bmatrix}\right)\\ -1\end{bmatrix}\right)\\ -1\end{bmatrix}\right)$$

Augmented System Equations

- **System Equation Transformation**

From the definition of error and error integral

$$e_{dq} = \begin{bmatrix} e_d \\ e_q \end{bmatrix} = i_{dq\_ref} - i_{dq}, \qquad s_{dq} = \int_0^t e_{dq} dt$$

$$\Rightarrow \dot{e}_{dq} = -\frac{d}{dt} i_{dq}, \qquad \dot{s}_{dq} = e_{dq}$$

The original **Two-dimensional** system equations can be augmented into the following **Four-dimensional** one:

$$\begin{cases} \dot{e}_{dq} = A(e_{dq} - i_{dq\_ref}) - BR(e_{dq}, s_{dq}) \\ \dot{s}_{dq} = e_{dq} \end{cases}$$

✓ The equilibrium point of the system can be noted as (0, s*), which means e_{dq}=0 and s_{dq}=s*.

Local Stability and Local Convergence

- To guarantee **local asymptotic stability** and **local exponential convergence**, the weight matrix and bias vector of the NN need to satisfy the following condition:

$$\mathrm{Re}\left\{\mathrm{eig}\left(\begin{bmatrix} G_{11} & G_{12} \\ I & 0 \end{bmatrix}\right)\right\} < 0$$

where

$$G_{11} = A - B\left.\frac{\partial R(e_{dq}, s_{dq})}{\partial e_{dq}}\right|_{e_{dq}=0, s_{dq}=s^*}$$

$$G_{12} = -B\left.\frac{\partial R(e_{dq}, s_{dq})}{\partial s_{dq}}\right|_{e_{dq}=0, s_{dq}=s^*}$$

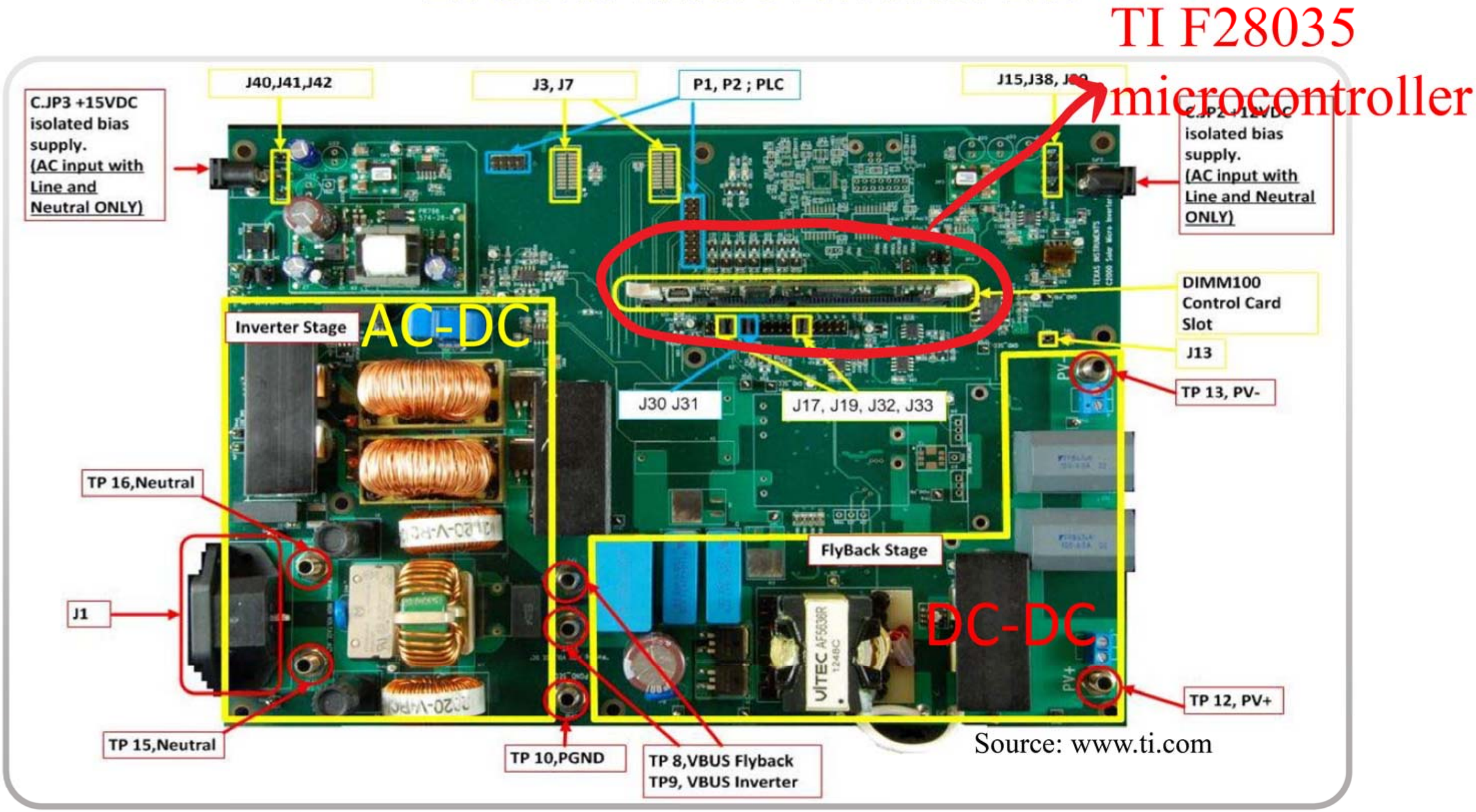
Linearizing the RNN controller

✓ Which means **all the eigenvalues have the negative real parts**.

Outline

V. Hardware-in-the-loop Experiment

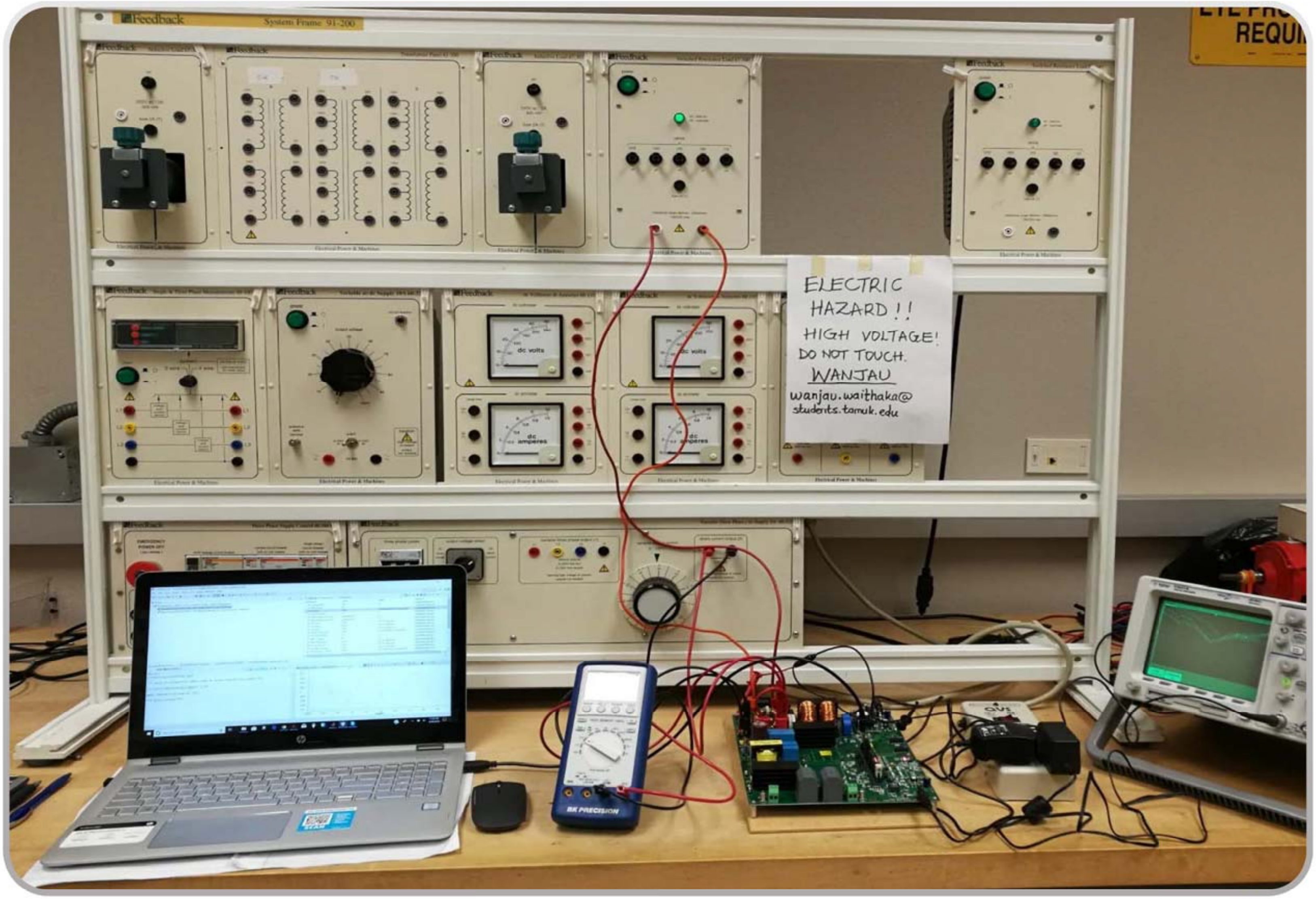
TI Solar Micro Inverter Kit



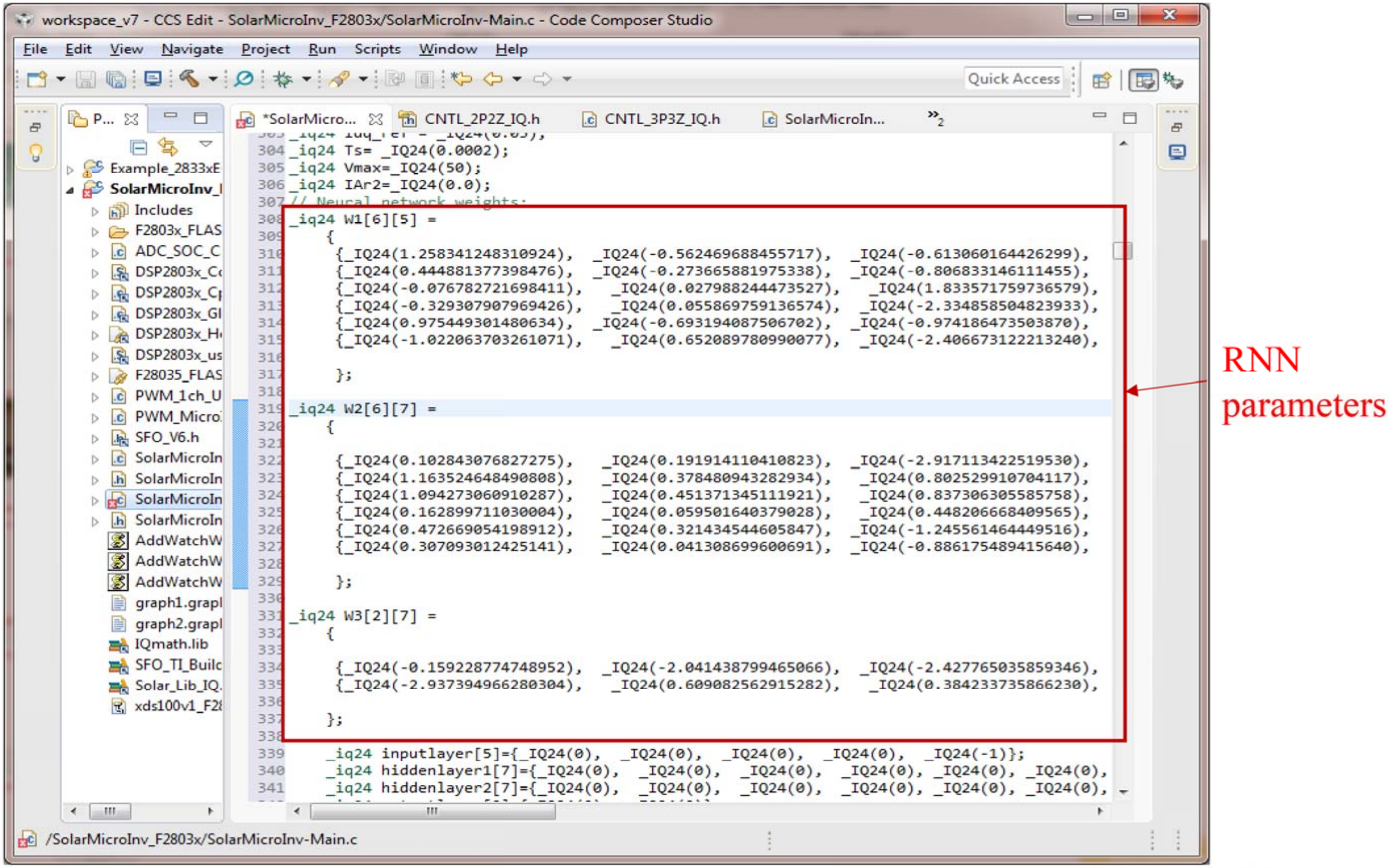
Features:

1) 28 to 45V at input; 280W for 220VAC, 140W for 110VAC;
2) C2000 Piccolo F28035 32-bit 60MHz MCU

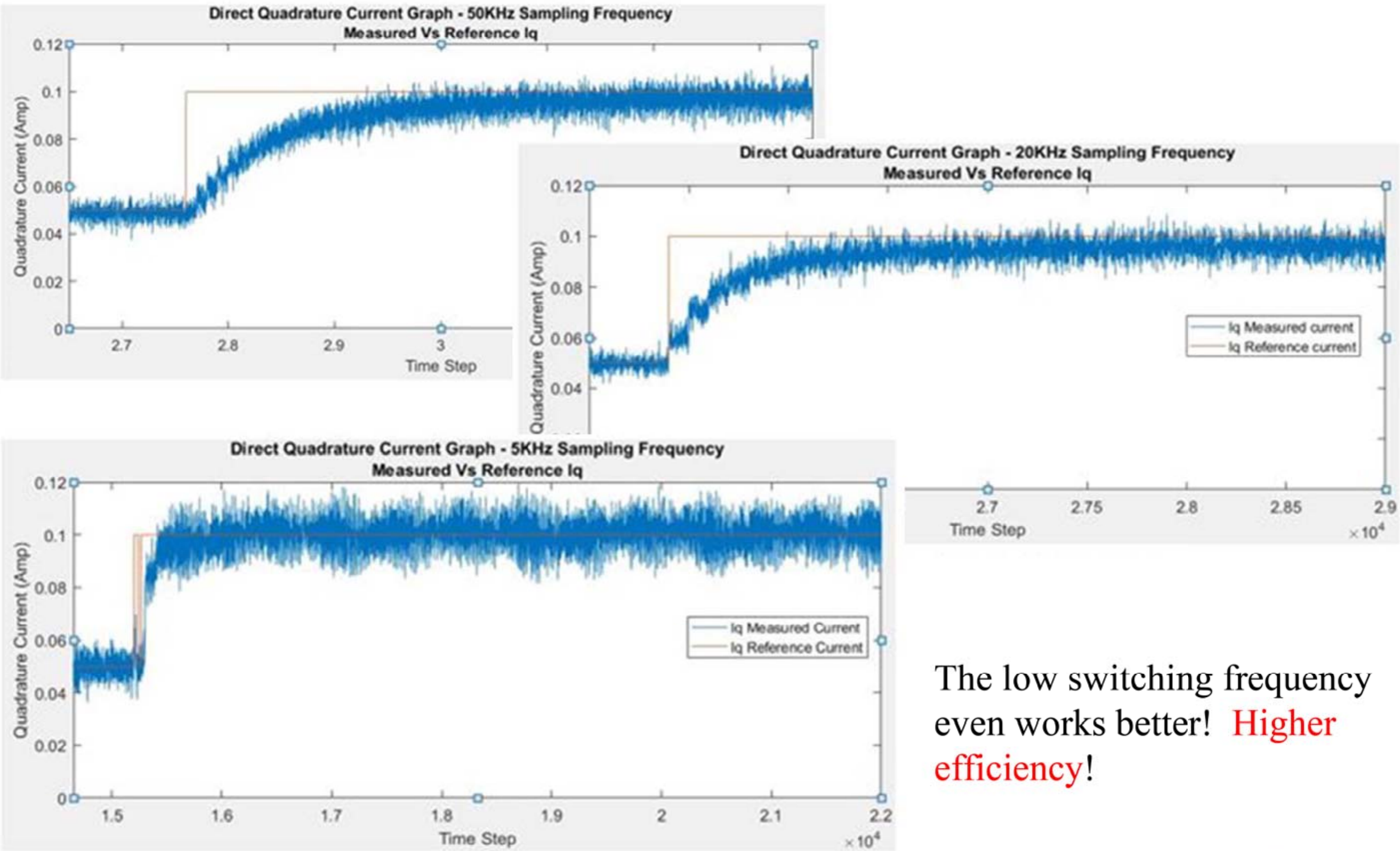
Hardware Validation Laboratory Set-up



C Code Implementation



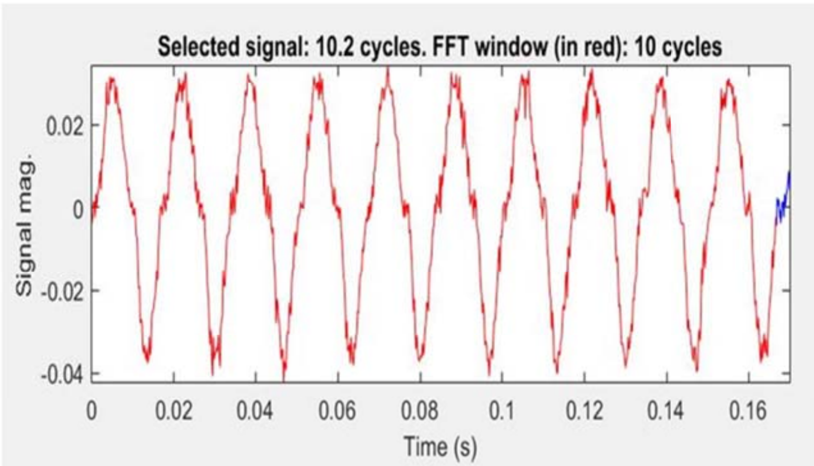
Current Tracking

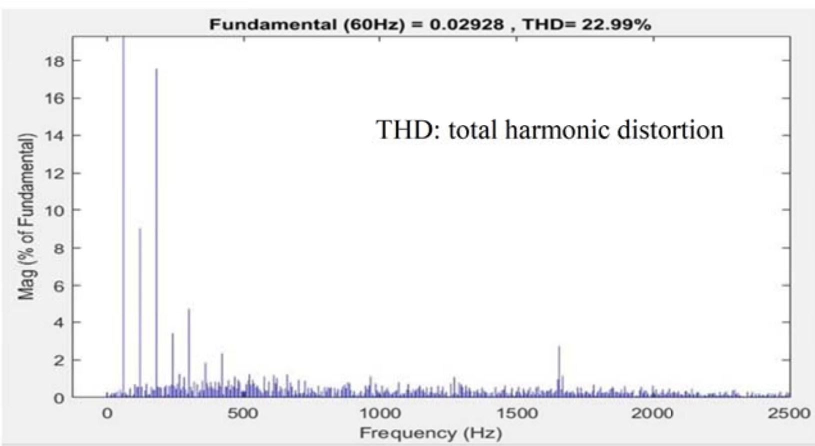


The low switching frequency even works better! Higher efficiency!

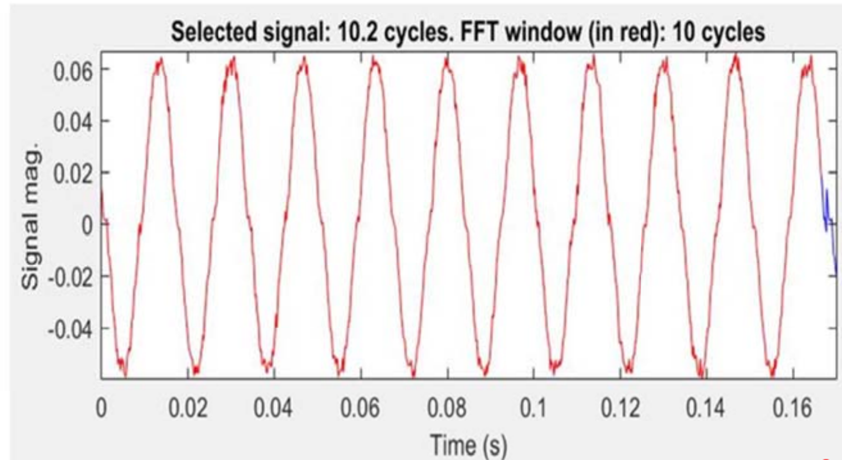
THD Comparison

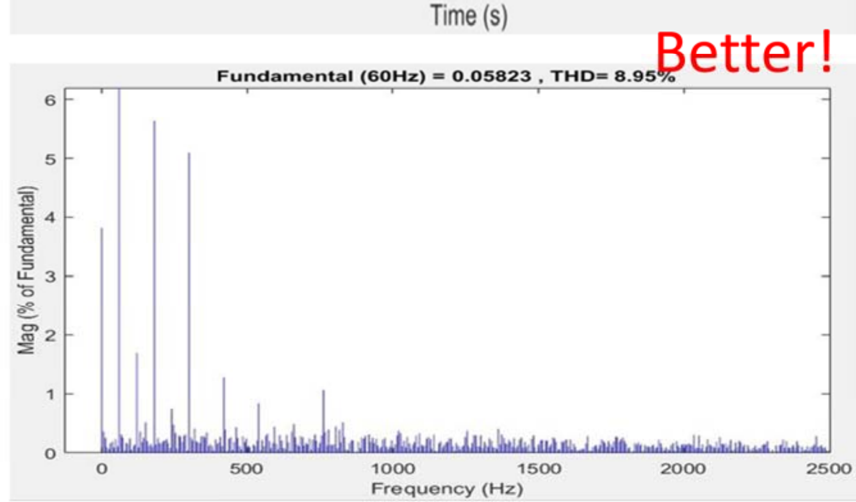
- 5KHz sampling and switching frequency





3P3Z controller by TI in CCS





RNN Controller

Outline

VI. Discussions about RNN Control

Discussions about RNN Control

➢ *Advantages*

- a **damping free** technique;
- allows for **low sampling** & **switching** frequency while maintaining high performance;
- ✓ strong ability of tracking the references, fast response, lower overshoot,
- ✓ **Robust**, **adaptive**, can tolerate a wide range of system parameter changes;
- ✓ approximate optimal control trained by Approximate Dynamic Programming principle
- ✓ …...

➢ *Further Improvements*

- ✓ **DSP+FPGA** implementation or Special Chip to solve calculation problem ? Not Available
- ✓ Refine the structure of RNN to better fit the embedded controller application.
- ✓ **Reducing** training hardware (gpu/cloud, supercomputer)
- ✓ Global stability problem, global convergence problem


Questions


Thank You