

# 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](mailto:fuxingang@gmail.com)

# 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

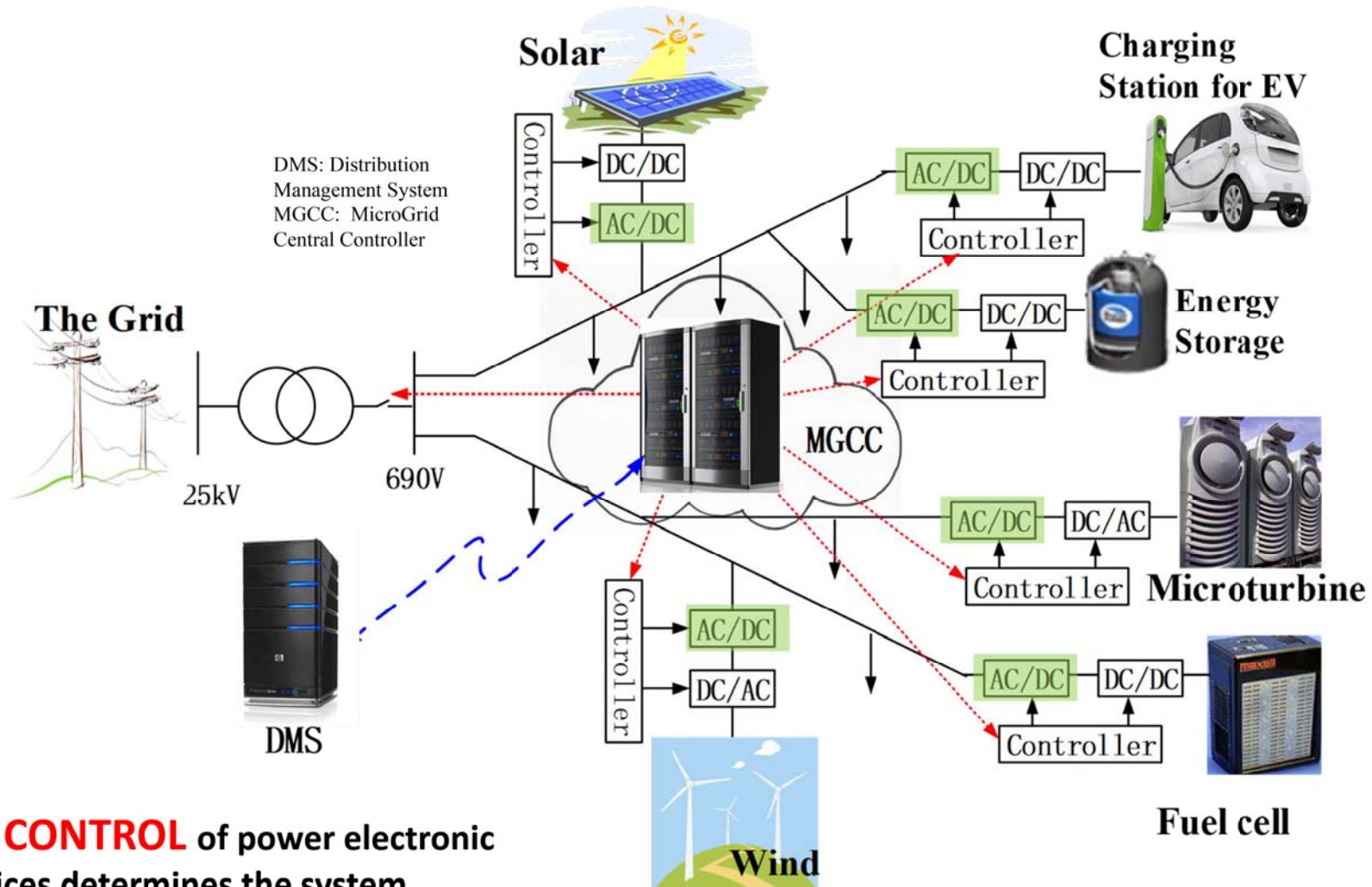
III. Trajectory based Training

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.

# 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.*

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

2/18/2021 2:04 PM

353 | [f](#) [t](#) [in](#) [e](#) [m](#)

**GTM Has Hybrid Energy Covered**

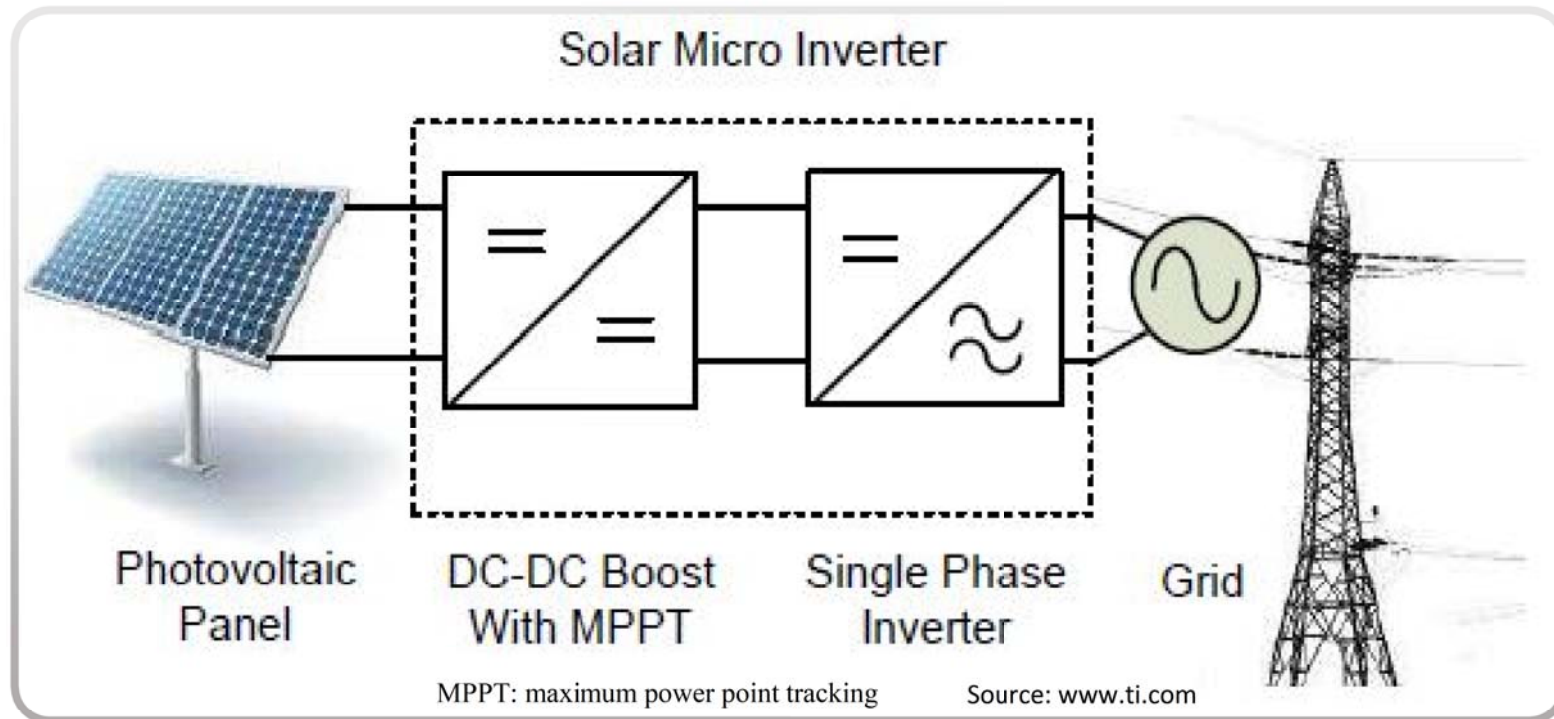
READ MORE

gtrm

#### TOP ARTICLES

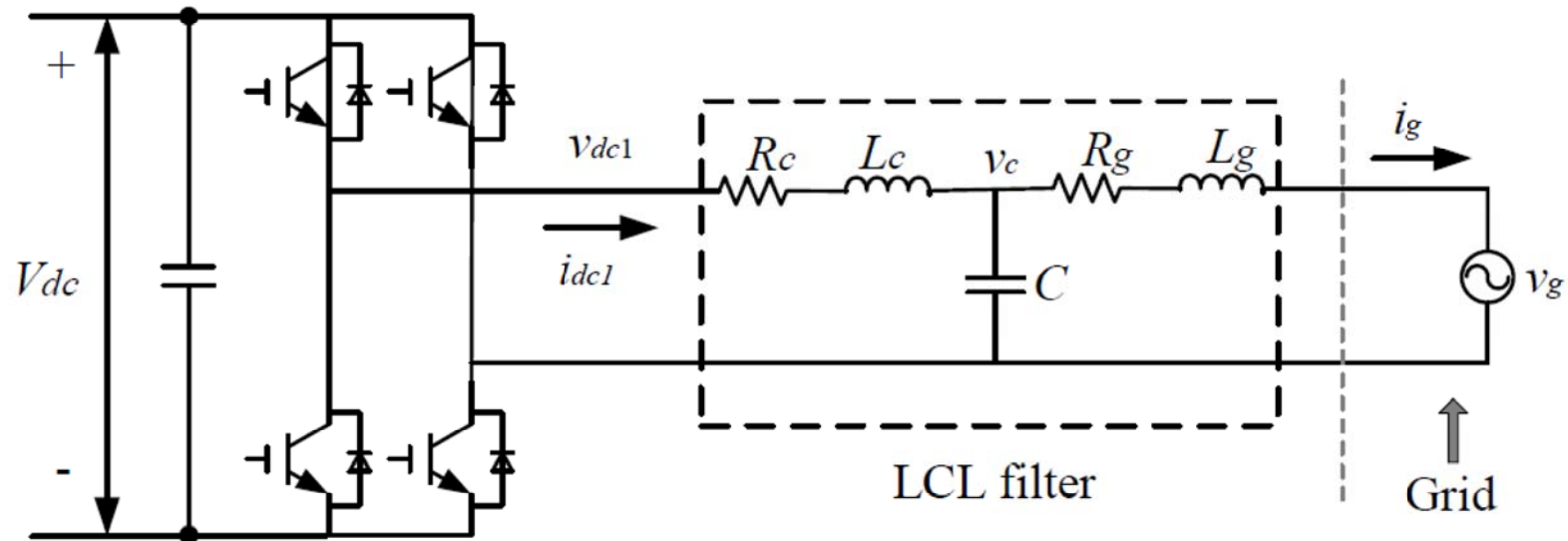
MOST POPULAR MOST COMMENTS

## *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.

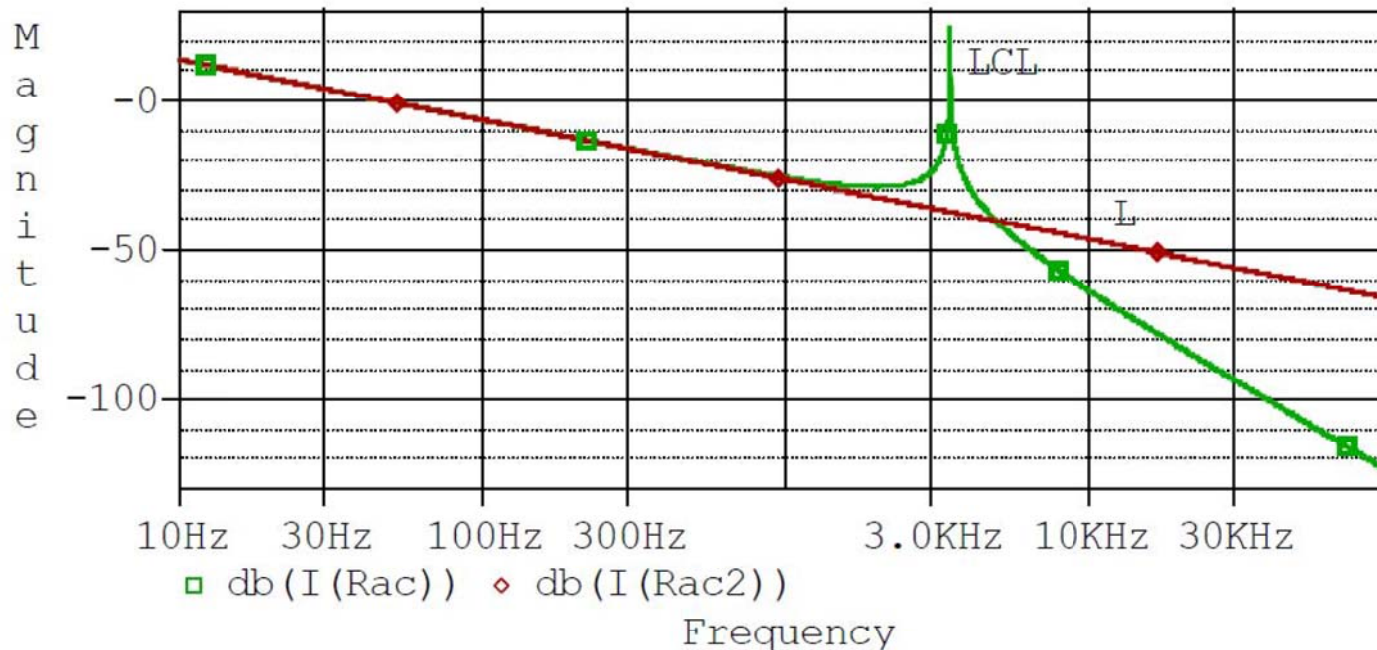
## Single-phase Solar Grid-tied Inverter/ Converter



- **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

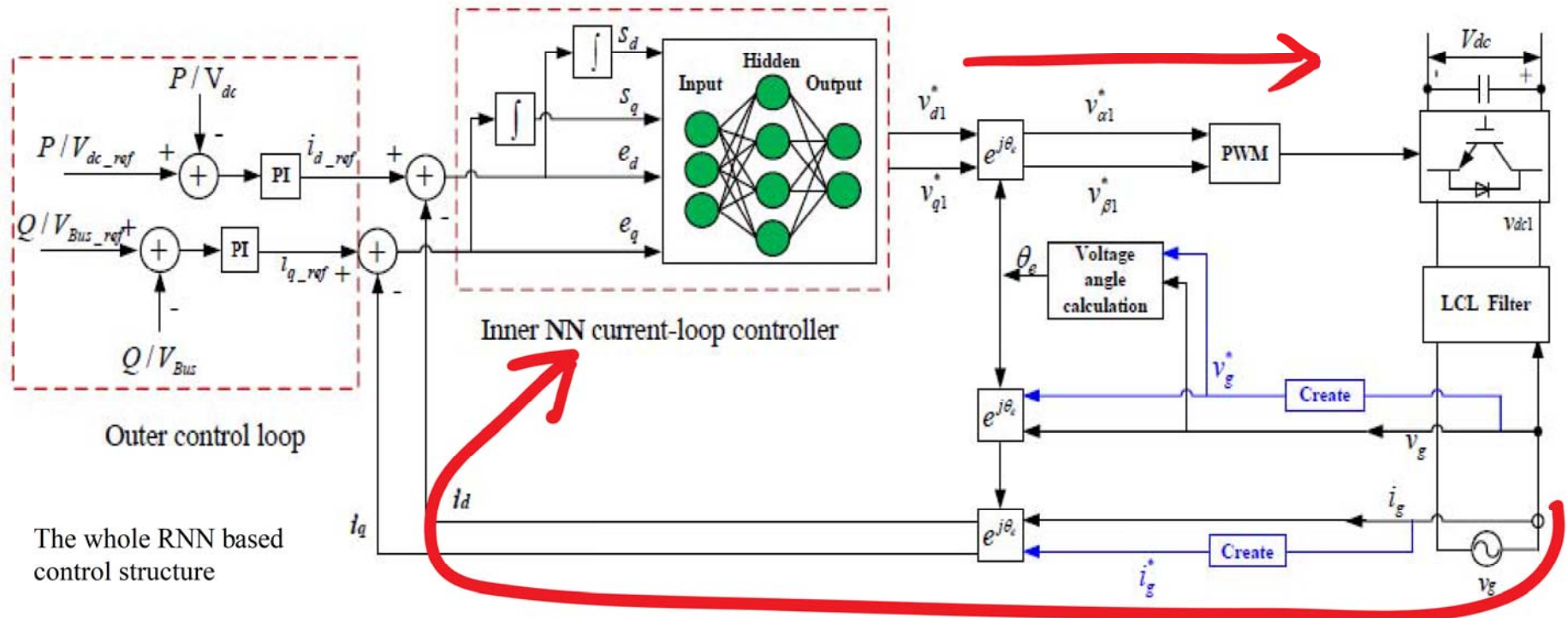
VI. Discussions about RNN Control

# Deep Learning

The screenshot shows a Google search for "deep learning". The search results include sponsored products such as "Dihuni OptiReady Supermicro...", "Supermicro 4023S-TRT 4U/Tower AM...", "Intel Deep Learning Inference...", "Velocity Micro ProMagix...", "Deep Learning (Adaptive...) \$24.95", and "Deep Learning (Adaptive...) \$48.43". Below the products, there are several educational diagrams and articles. One diagram shows a neural network structure with layers labeled  $Z_1$ ,  $Z_2$ ,  $Z_3$ ,  $Z_4$ , and  $Z_5$ . Another diagram compares "Machine Learning" (input, feature extraction, classification, output) with "Deep Learning" (input, feature extraction + classification, output). A graph titled "why deep learning" shows "Performance" vs "Amount of data", comparing "Deep learning" (blue line) and "Other learning algorithms" (red line). Other diagrams include "Simple Neural Network" and "Deep Learning Neural Network" with layers labeled "Input Layer", "Hidden Layer", and "Output Layer".

Mainly in Image/video process, natural language process

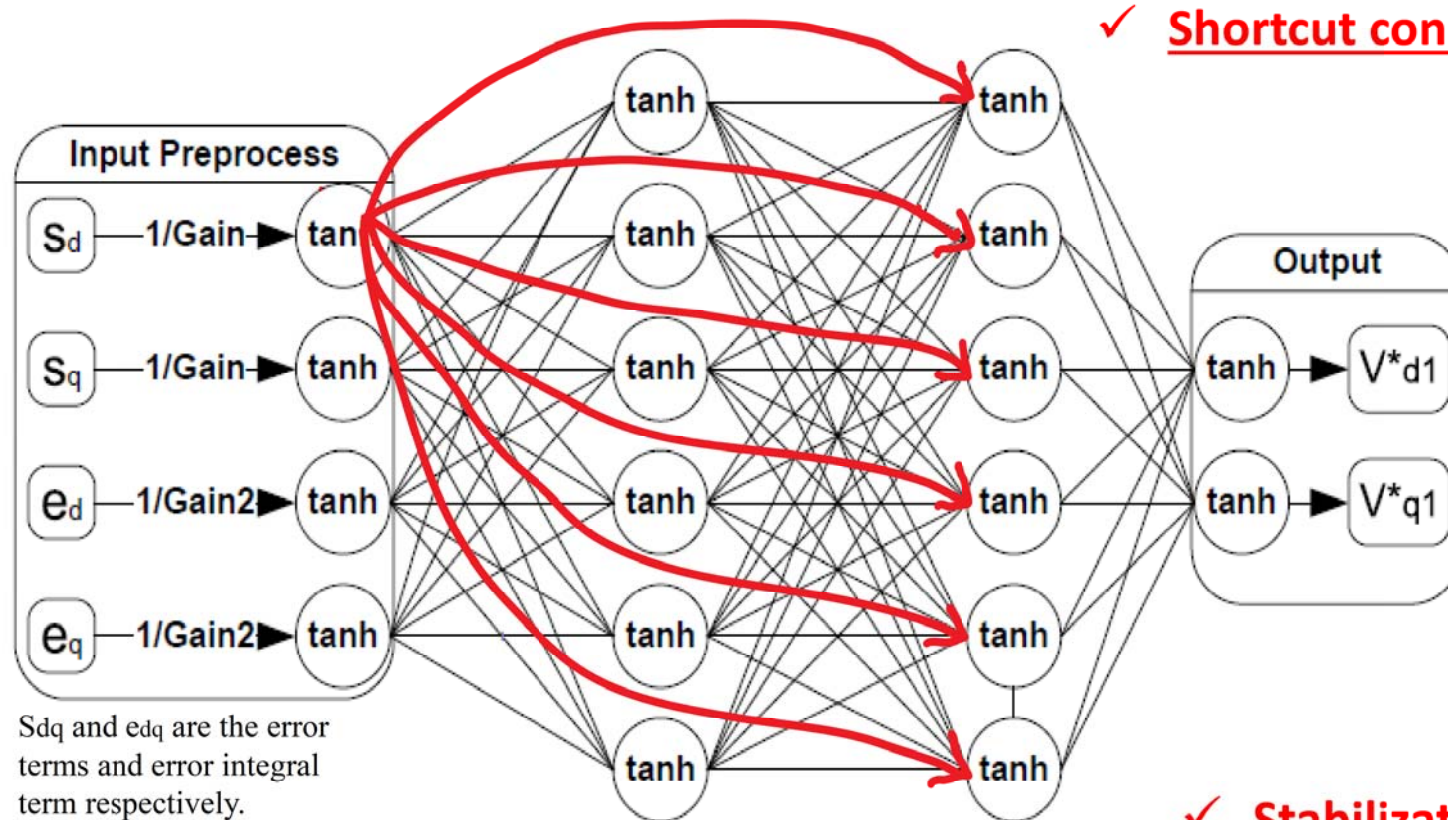
# Proposed Neural Network in Feedback Control Loops



Feedback connection in closed-loop control

## Recurrent Neural Network

## Feature - *Error Integral*



✓ Shortcut connections

✓ 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**

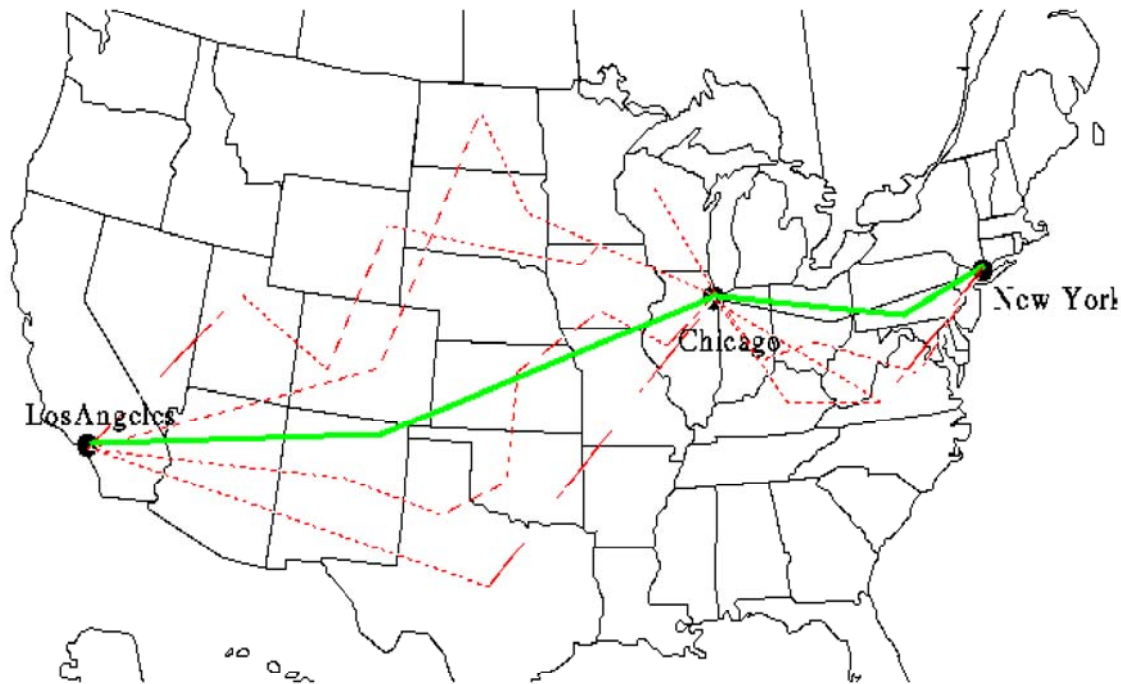
IV. Local Stability and Local Convergence

V. Hardware-in-the-loop Experiment

VI. Discussions about RNN Control

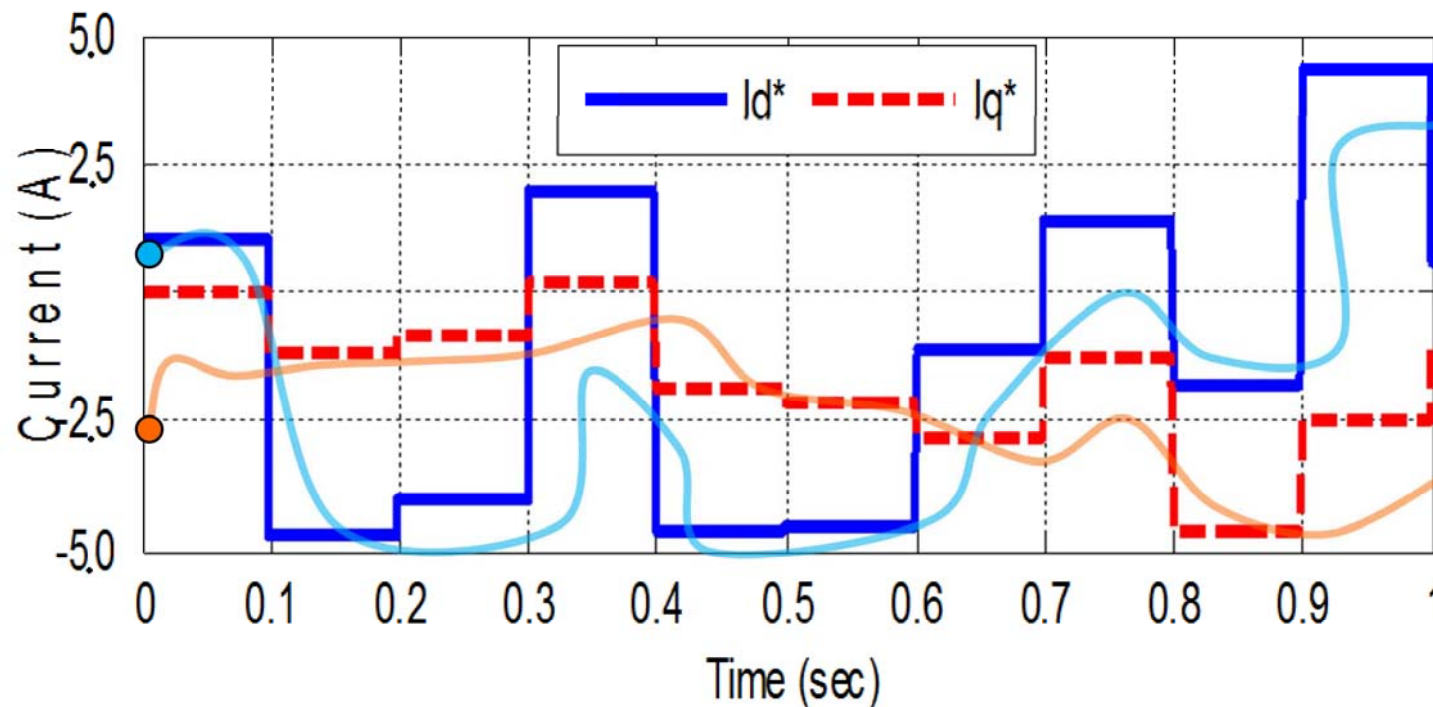
## Approximate Dynamic Programming

- **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

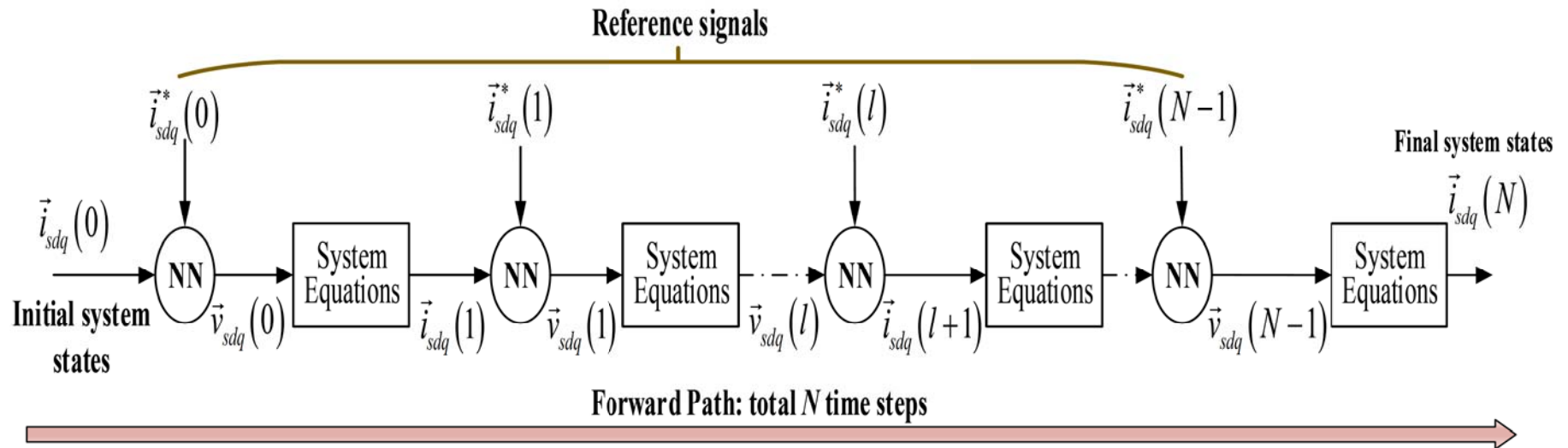


Note that: before training, the NN controller **cannot** adjust the currents to follow the reference.

- ✓ 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*.

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

## 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  $4 \times 1000 = 4000$ .
  - ✓ 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.

$$1: C \leftarrow 0, \vec{e}_{dq}(0) \leftarrow 0, \vec{s}_{dq}(0) \leftarrow 0, \frac{\partial \vec{i}_{dq}(0)}{\partial \vec{w}} \leftarrow 0, \frac{\partial \phi_{dq}(0)}{\partial \vec{w}} \leftarrow 0$$

2: {Calculate the Jacobian matrix  $J(\vec{w})$ }

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

$$4: \vec{u}_{dq}(k) \leftarrow k_{PWM} R(\vec{e}_{dq}(k), \vec{s}_{dq}(k), \vec{w}) - \vec{v}_{dq}$$

$$5: \frac{\partial \vec{s}_{dq}(k)}{\partial \vec{w}} \leftarrow T_s \left[ \frac{\partial \phi_{dq}(k)}{\partial \vec{w}} - \frac{1}{2} \frac{\partial \vec{i}_{dq}(k)}{\partial \vec{w}} \right]$$

$$6: \frac{\partial \vec{u}_{dq}(k)}{\partial \vec{w}} \leftarrow k_{PWM} \left[ \frac{\partial R(k)}{\partial \vec{w}} + \frac{\partial R(k)}{\partial \vec{e}_{dq}(k)} \frac{\partial \vec{i}_{dq}(k)}{\partial \vec{w}} + \frac{\partial R(k)}{\partial \vec{s}_{dq}(k)} \frac{\partial \vec{s}_{dq}(k)}{\partial \vec{w}} \right]$$

$$7: \frac{\partial \vec{i}_{dq}(k+1)}{\partial \vec{w}} \leftarrow A \frac{\partial \vec{i}_{dq}(k)}{\partial \vec{w}} + B \frac{\partial \vec{u}_{dq}(k+1)}{\partial \vec{w}}$$

$$8: \vec{i}_{dq}(k+1) \leftarrow A \vec{i}_{dq}(k) + B \vec{u}_{dq}(k)$$

$$9: \vec{e}_{dq}(k+1) \leftarrow \vec{i}_{dq}(k+1) - \vec{i}_{dq\_ref}(k+1)$$

$$10: \vec{s}_{dq}(k+1) \leftarrow \vec{s}_{dq}(k) + \frac{T_s}{2} [\vec{e}_{dq}(k) + \vec{e}_{dq}(k+1)]$$

11:  $C \leftarrow C + U(\vec{e}_{dq}(k+1))$  {accumulate DP cost}

$$12: \frac{\partial \phi(k+1)}{\partial \vec{w}} \leftarrow \frac{\partial \phi(k)}{\partial \vec{w}} + \frac{\partial \vec{i}_{dq}(k+1)}{\partial \vec{w}}$$

$$13: \frac{\partial V(k+1)}{\partial \vec{w}} \leftarrow \frac{\partial V(k+1)}{\partial \vec{e}_{dq}(k+1)} \frac{\partial \vec{i}_{dq}(k+1)}{\partial \vec{w}}$$

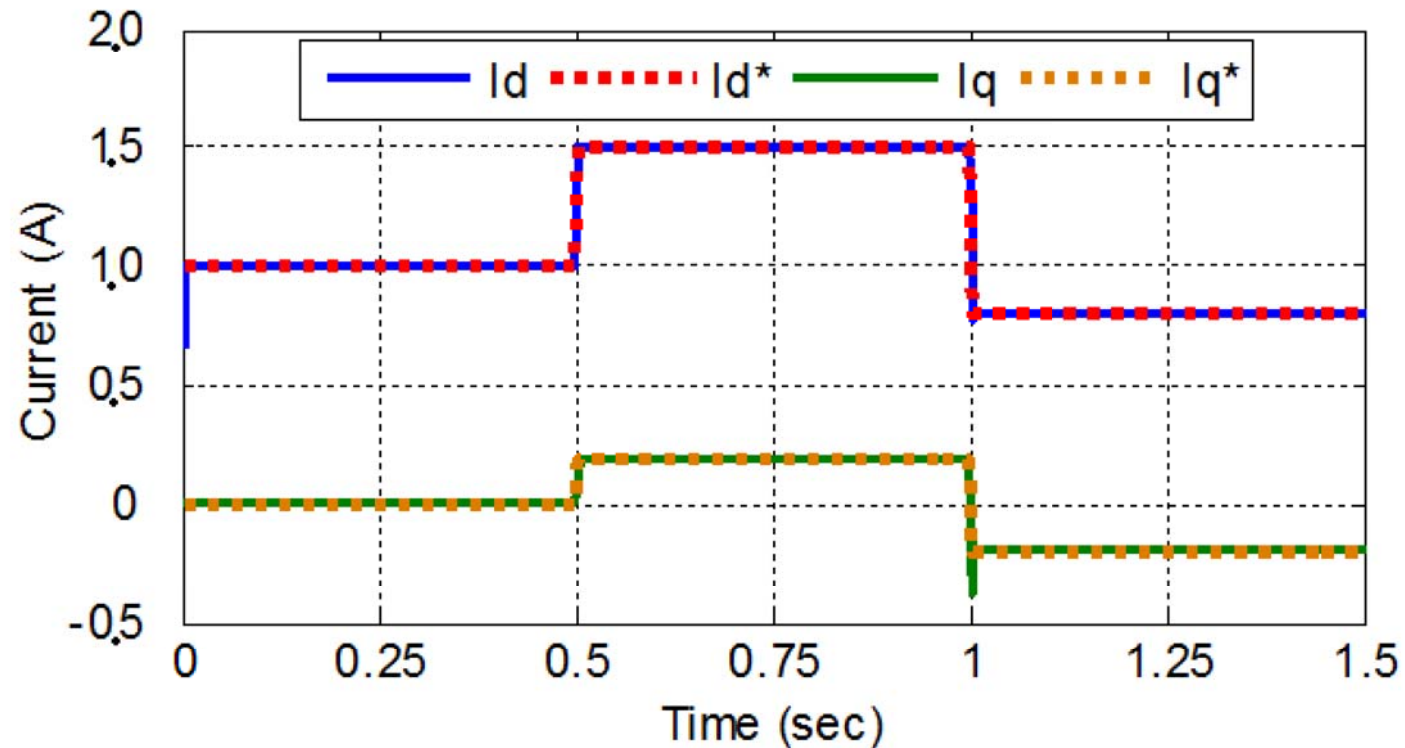
$$14: \text{the } (k+1)\text{th row of } J(\vec{w}) \leftarrow \frac{\partial V(k+1)}{\partial \vec{w}}$$

15: **end for**

2/1: 16: {On exit, the Jacobian matrix  $J(\vec{w})$  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)

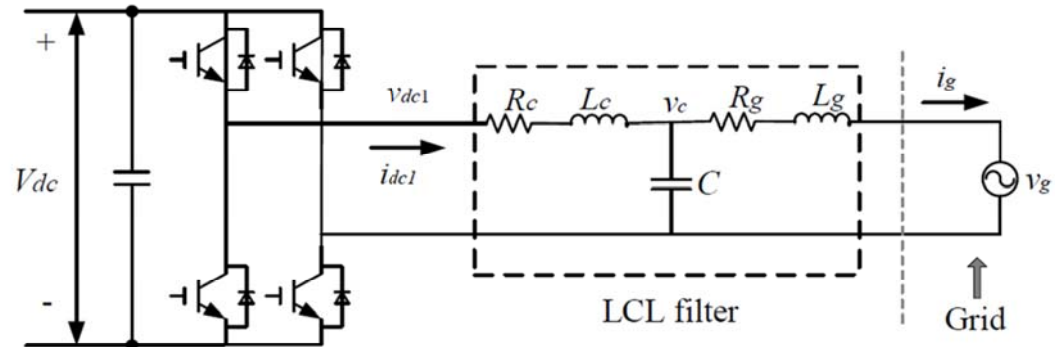
# Outline

## IV. Local Stability and Local Convergence

## System Dynamic Equation

### ➤ State-space model

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



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

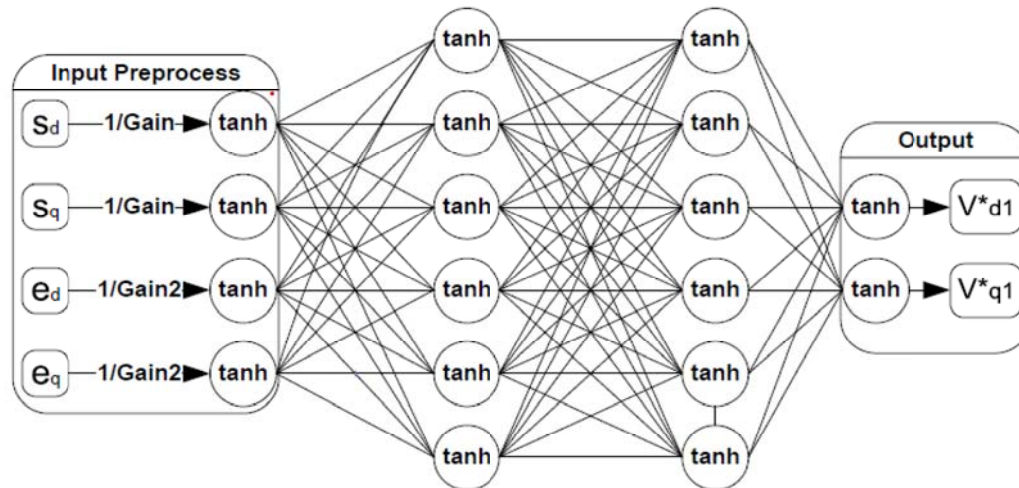
where  $A = - \begin{bmatrix} \frac{R_g + R_c}{L_g + L_c} & -\omega_s \\ -\omega_s & \frac{R_g + R_c}{L_g + L_c} \end{bmatrix}$ ,

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

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

## Function format of RNN controller

### ➤ NN function representation



$$N(e_{dq}, s_{dq}, w) = \tanh \left( w_3 \left[ \tanh \left( w_2 \left[ \tanh \left( w_1 \left[ \begin{array}{c} \tanh \left[ \frac{e_{dq}}{Gain} \right] \\ \frac{s_{dq}}{Gain2} \end{array} \right] \right) \right] \right) \right] \right) \right]$$

## Augmented System Equations

### ➤ System Equation Transformation

From the definition of error and error integral

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

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

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:

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

where

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

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

Linearizing the RNN controller



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

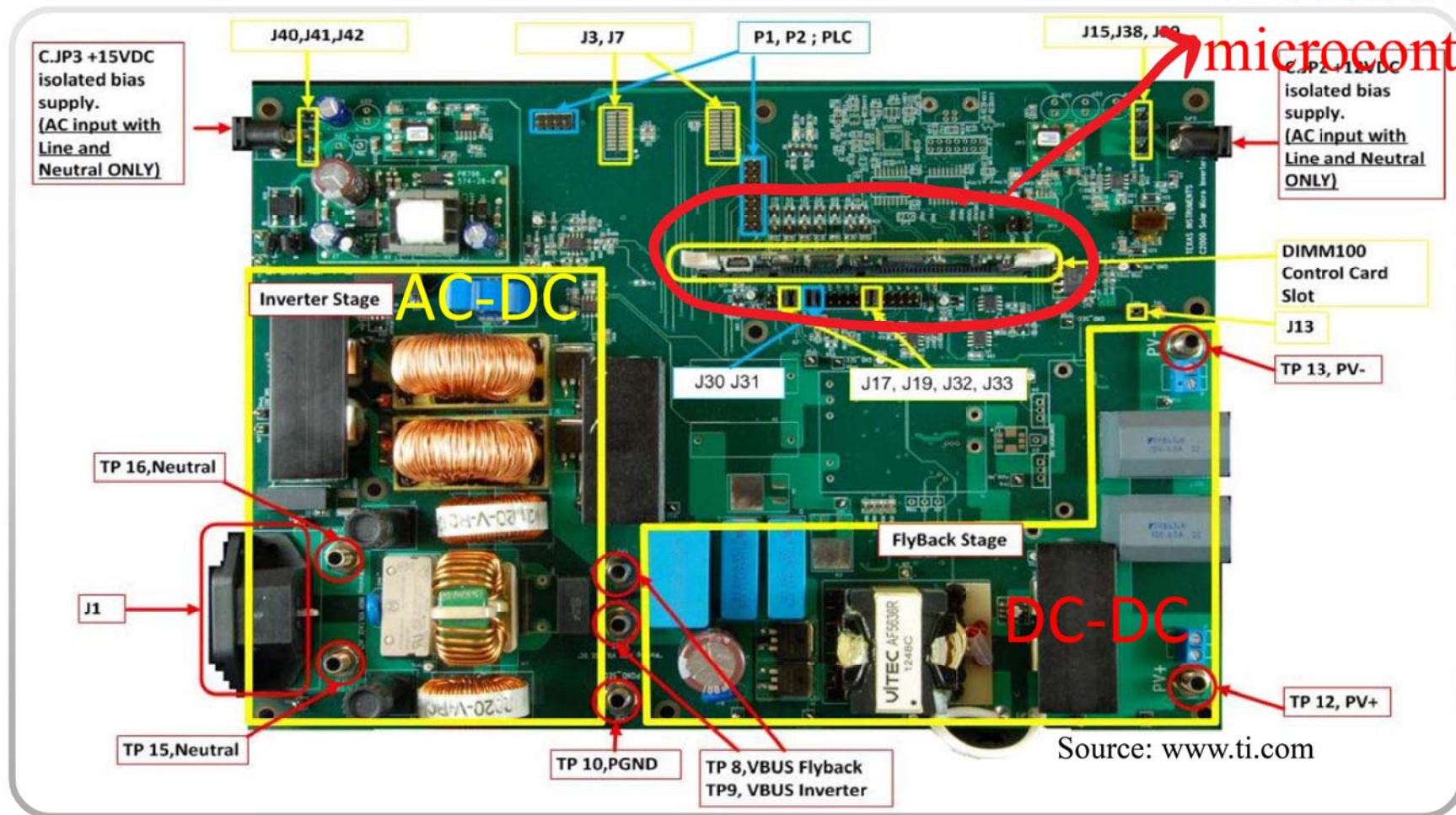
# Outline

## V. Hardware-in-the-loop Experiment



# TI Solar Micro Inverter Kit

TI F28035  
microcontroller

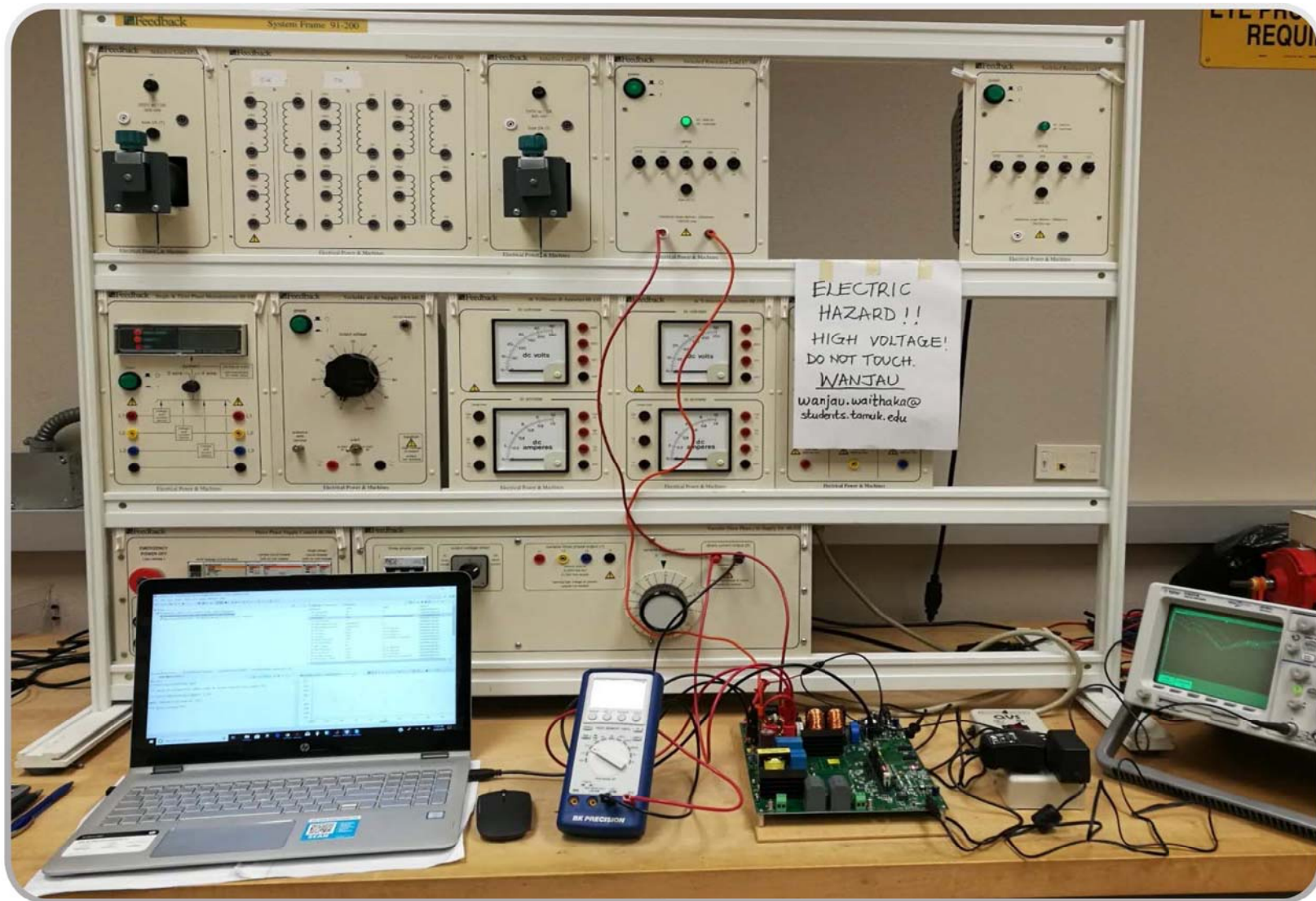


Source: www.ti.com

### 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*

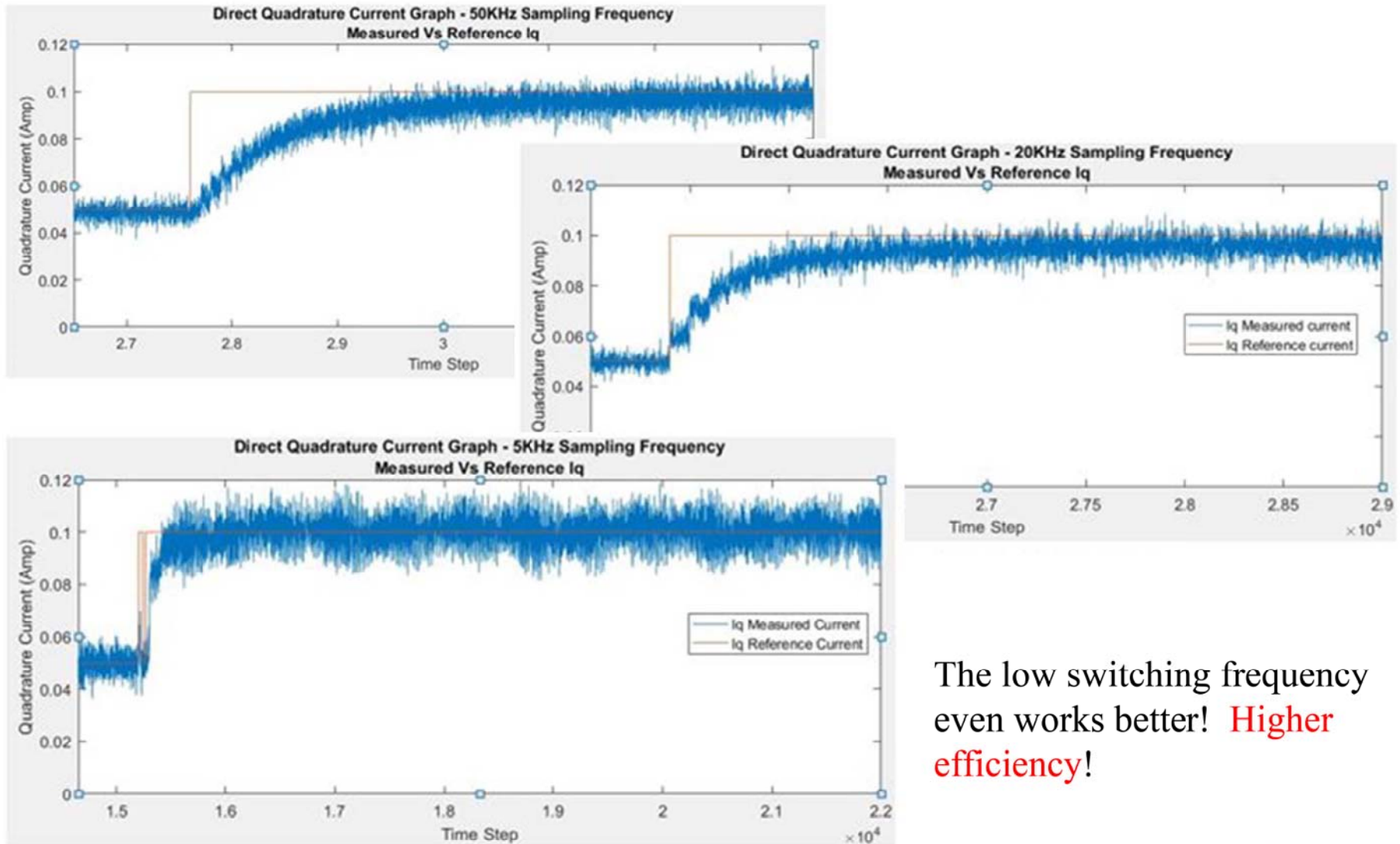


# C Code Implementation

```
workspace_v7 - CCS Edit - SolarMicroInv_F2803x/SolarMicroInv-Main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
*SolarMicro... CNTL_2P2Z_IQ.h CNTL_3P3Z_IQ.h SolarMicroIn... »_2
304 iq24 Ts= _IQ24(0.0002);
305 iq24 Vmax=_IQ24(50);
306 iq24 IAr2=_IQ24(0.0);
307 // Neural network weights:
308 iq24 W1[6][5] =
309 {
310     {_IQ24(1.258341248310924), _IQ24(-0.562469688455717), _IQ24(-0.613060164426299),
311     {_IQ24(0.444881377398476), _IQ24(-0.273665881975338), _IQ24(-0.806833146111455),
312     {_IQ24(-0.076782721698411), _IQ24(0.027988244473527), _IQ24(1.833571759736579),
313     {_IQ24(-0.329307907969426), _IQ24(0.055869759136574), _IQ24(-2.334858504823933),
314     {_IQ24(0.975449301480634), _IQ24(-0.693194087506702), _IQ24(-0.974186473503870),
315     {_IQ24(-1.022063703261071), _IQ24(0.652089780990077), _IQ24(-2.406673122213240),
316     };
317
318
319 iq24 W2[6][7] =
320 {
321     {_IQ24(0.102843076827275), _IQ24(0.191914110410823), _IQ24(-2.917113422519530),
322     {_IQ24(1.163524648490808), _IQ24(0.378480943282934), _IQ24(0.802529910704117),
323     {_IQ24(1.094273060910287), _IQ24(0.451371345111921), _IQ24(0.837306305585758),
324     {_IQ24(0.162899711030004), _IQ24(0.059501640379028), _IQ24(0.448206668409565),
325     {_IQ24(0.472669054198912), _IQ24(0.321434544605847), _IQ24(-1.245561464449516),
326     {_IQ24(0.307093012425141), _IQ24(0.041308699600691), _IQ24(-0.886175489415640),
327     };
328
329
330
331 iq24 W3[2][7] =
332 {
333     {_IQ24(-0.159228774748952), _IQ24(-2.041438799465066), _IQ24(-2.427765035859346),
334     {_IQ24(-2.937394966280304), _IQ24(0.609082562915282), _IQ24(0.384233735866230),
335     };
336
337
338
339 iq24 inputlayer[5]={_IQ24(0), _IQ24(0), _IQ24(0), _IQ24(0), _IQ24(-1)};
340 iq24 hiddenlayer1[7]={_IQ24(0), _IQ24(0), _IQ24(0), _IQ24(0), _IQ24(0), _IQ24(0),
341 iq24 hiddenlayer2[7]={_IQ24(0), _IQ24(0), _IQ24(0), _IQ24(0), _IQ24(0), _IQ24(0),
```

RNN parameters

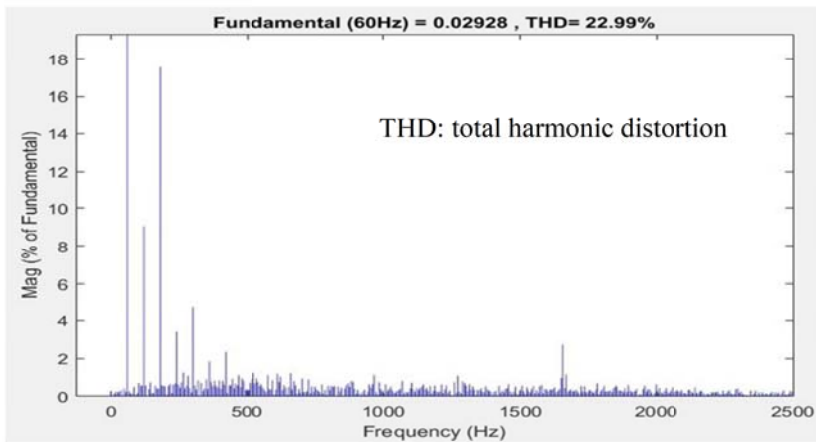
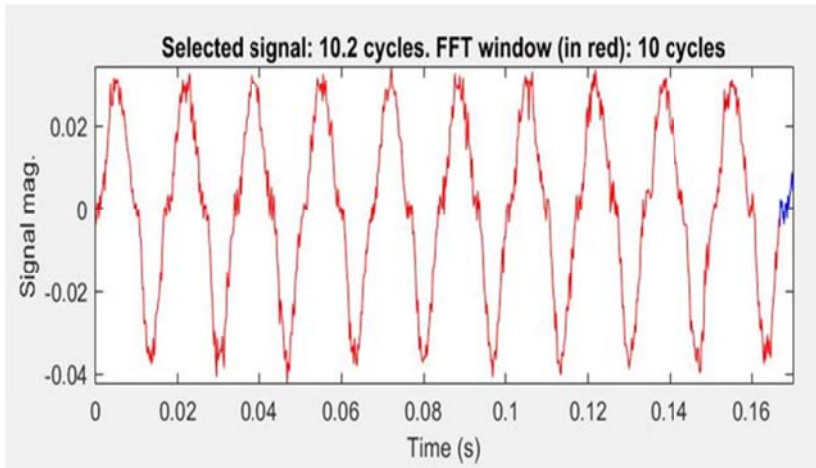
# Current Tracking



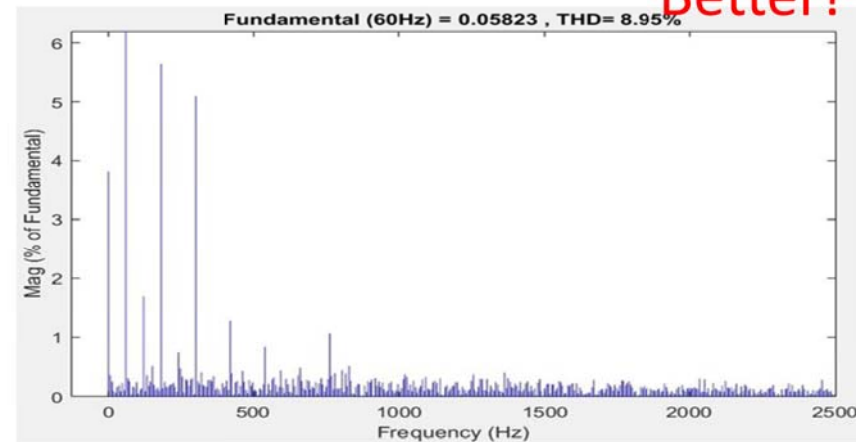
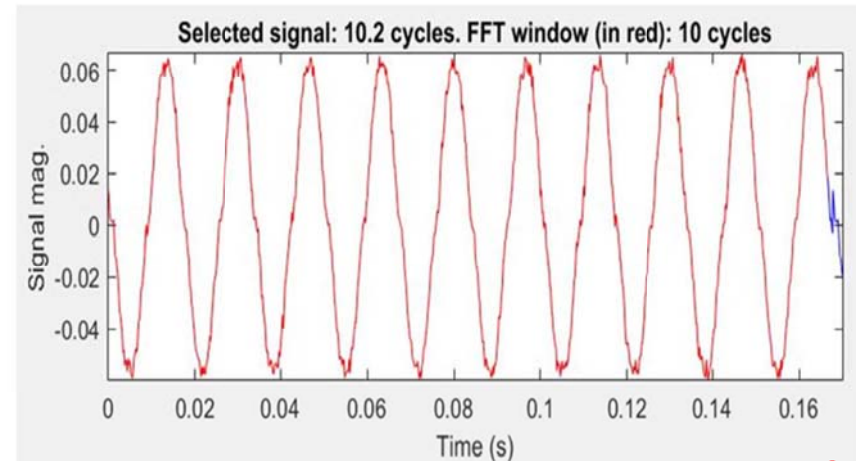
The low switching frequency even works better! **Higher efficiency!**

# THD Comparison

- 5KHz sampling and switching frequency



3P3Z controller by TI in CCS



Better!

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





