Michael E. Aiello

Described in the paper Implementation of an Advanced AC BrushlessMotor Controller for Use in High Reliability a model for a basic caged induction motor was presented. A custom simulator was used to run a simple trajectory open loop using V/f (constant volt seconds) command reference.

In this section we add to the simulator, a simple closed loop control model that demonstrates field oriented control for an induction motor.

In keeping with that presented in the previous sections, we set the motor simulator to run in the *Rotating Frame* (physical model with no D/Q transformations). We also enable space vector PWM (simulations in the paper cited above used analog voltages applied to the motor terminals).

Other changes to the simulator:

- DC bus voltage is changed from 600 VDC to 100 VDC.
- The trajectory command of 10 seconds is maintained but the maximum speed command is changed from 50 radians/sec to 1 radians/sec (approx 2 comutation cycles over 10 sec).

The motor simulator runs entirely in floating point double precision format. However, I thought it may be useful to demonstrate the control portion of the simulator running in both double precision floating point (64 bit) and fixed precision (32 bit) formats.

In line with experimenting with the fixed 32 bit format, it made sense to me to look at the control solutions for induction motors provided by the TI C2000 motor control SDK.

TI C2000 SOC's are designed specifically for motor control applications. These chips are designed to implement control algorithms written entirely using fixed point arithmetic. Current C2000 (C28) devices employ 32 bit fixed math operations. Newer versions of these devices (C29) employ 32 and 64 bit fixed math operations. Both device types can also perform single and double precision floating operations, but with less efficiency then cpu cores such as the ARM R5 or A72.

For the C2000 family of devices, TI includes examples for basic implementations of field oriented control of an induction motor without any feedforward elements added to the control.

For the purposes of the following demonstration, we use one example provided by TI for sensored field oriented induction motor control that fits very nicely with the induction motor simulation model provided in the paper cited above.

This example is outlined in the following TI application note included with this paper
sprabp8.pdf

Specifically, the simple control scheme oulined in Figure 9 of **sprabp8.pdf** is what will be implemented in the simulator. This block diagram is shown below.

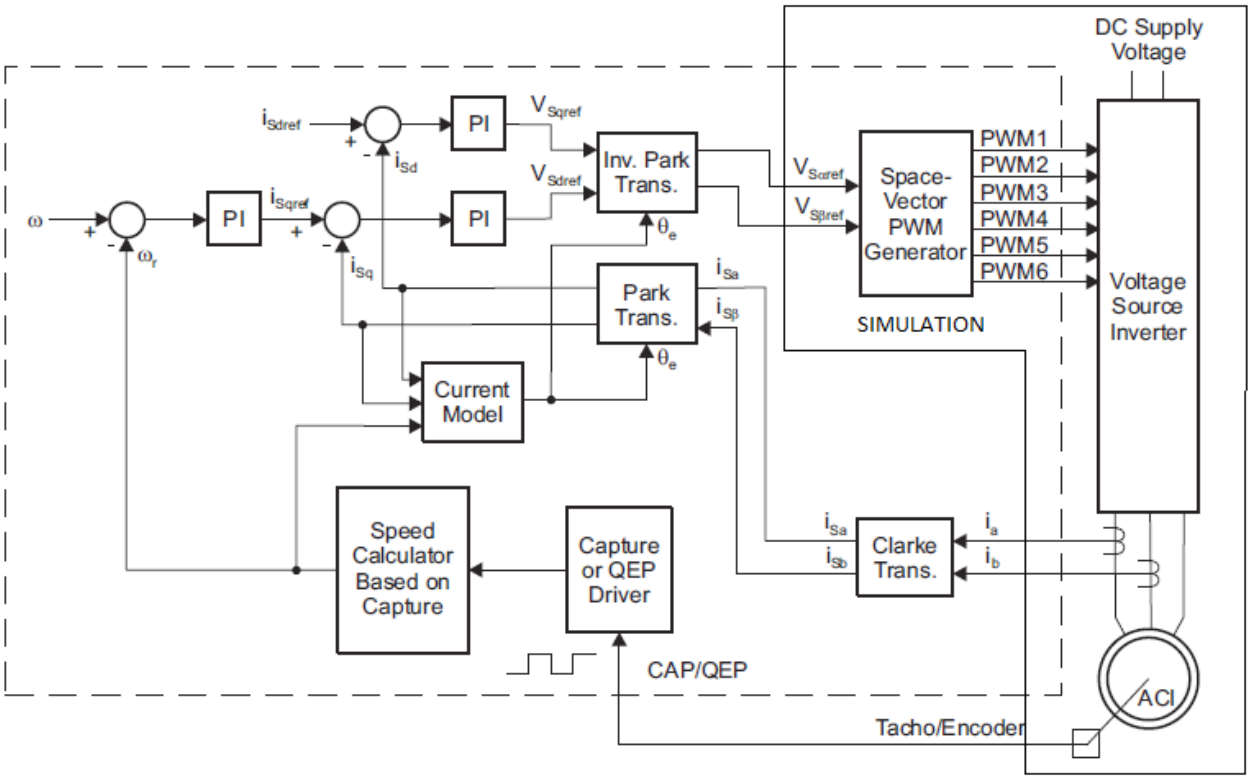


Figure 9. Overall Block Diagram of Indirect Rotor Flux Oriented Control

In the actual C2000 code development environment (SDK), the control shown in the block diagram above is described by C code in the following document (specifically the code starting at "BUILDLEVEL==LEVEL5" on page 17 of this document).
See included document **HVACI_Sensored.c.pdf**

Using references from "HVACI_Sensored.c" above, two versions of this control are added to the simulator, labed *fixed* and *double*.

Modifying TI's original macros allowed a direct transition to 32 bit fixed point simulation and 64 bit double simulation.
The original IQMath 32 bit fixed point is shown in the included document
FOC_tests_fixed_sim.hpp.pdf

Using full double precision math, the following algorithm was used with the simulator.
See included document **controlSuite_double_sim.hpp.pdf**

For the in which this implementation runs using 32 bit fixed point math, this algorithm was used See included document **controlSuite_fixed_sim.hpp.pdf**

The code in both files above use macros provided by the original TI C2000 SDK.

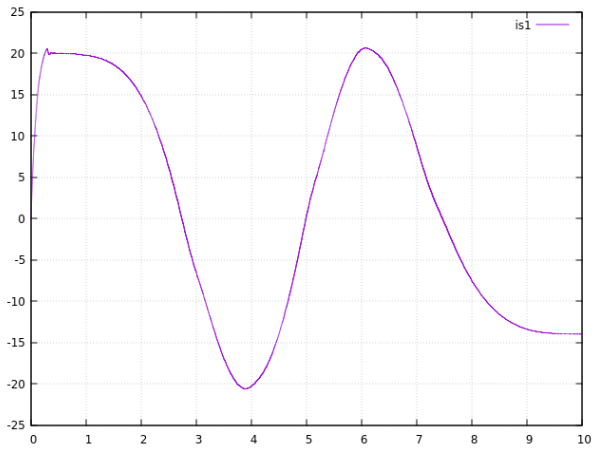
For the double precision floating point case it was a relatively simple process to translate from the TI Fixed Point format (IQMath) to double precision floating point.

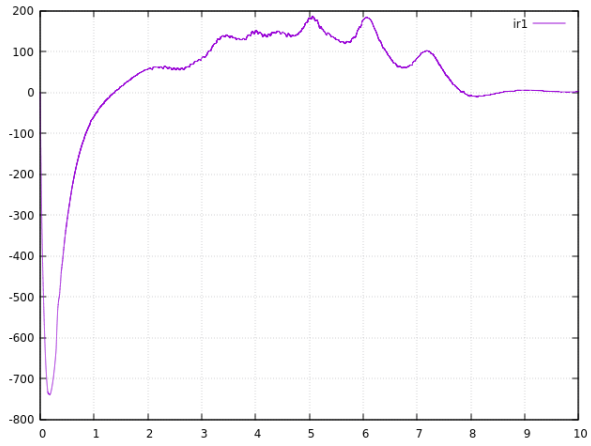
However, for the fixed point test, a little work was involved in the translation because TI does not provide the IQMath implementation in the form of simulation code.
Fortunately, there happens to be a 32 bit fixed point execution model written in C provided by Michal Getka, Ivan Voras and Tim Harkrick that could be used as a drop-in for TI IQMath.
See included document **fptc.h.pdf**

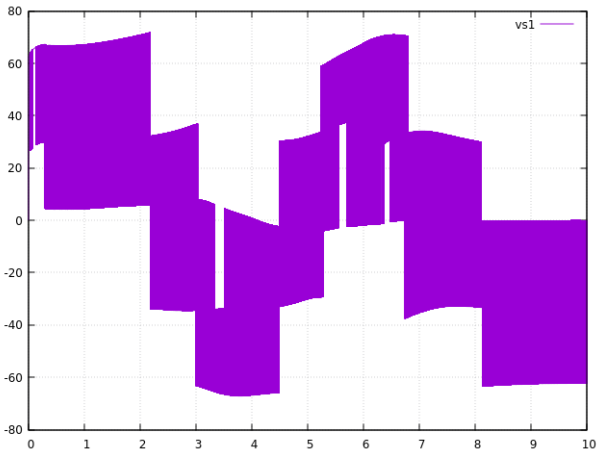
To run the simulation for the induction motor using the control defined in **controlSuite_double_sim.hpp** or **controlSuite_fixed_sim.hpp**, some parameters were changed relative to the open loop simulation cited in the original paper.

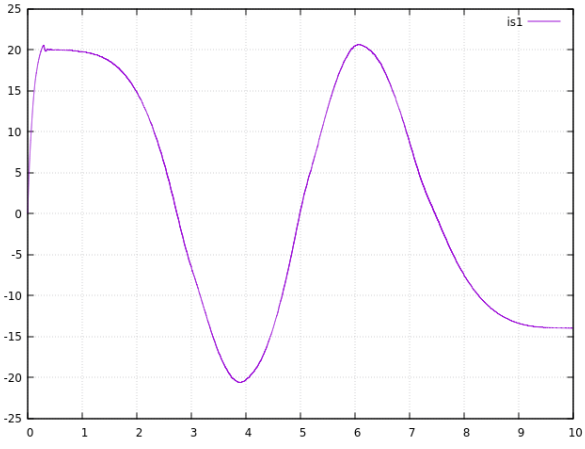
First, the maximum speed (electrical frequency) for the 10 second test trajectory was reduced to 1 radian/sec. The DC bus voltage was reduced from 600 VDC to 100 VDC. Also, the portion of the simulator dealing with voltage control to the induction motor was changed from analog to PWM running at 20 KHz.

Plots of is1 (phase 1 stator current), ir1 (phase 1 rotor current), and vs1 (line to neutral voltage of phase 1 stator) and the command velocity omegad are shown below (both double and fixed simulations show similar results).

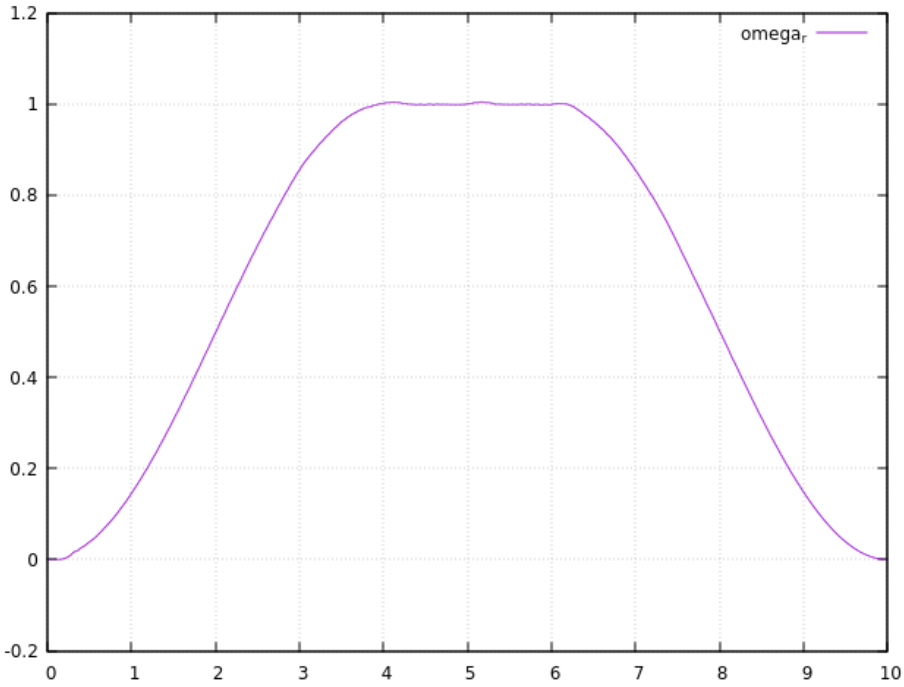




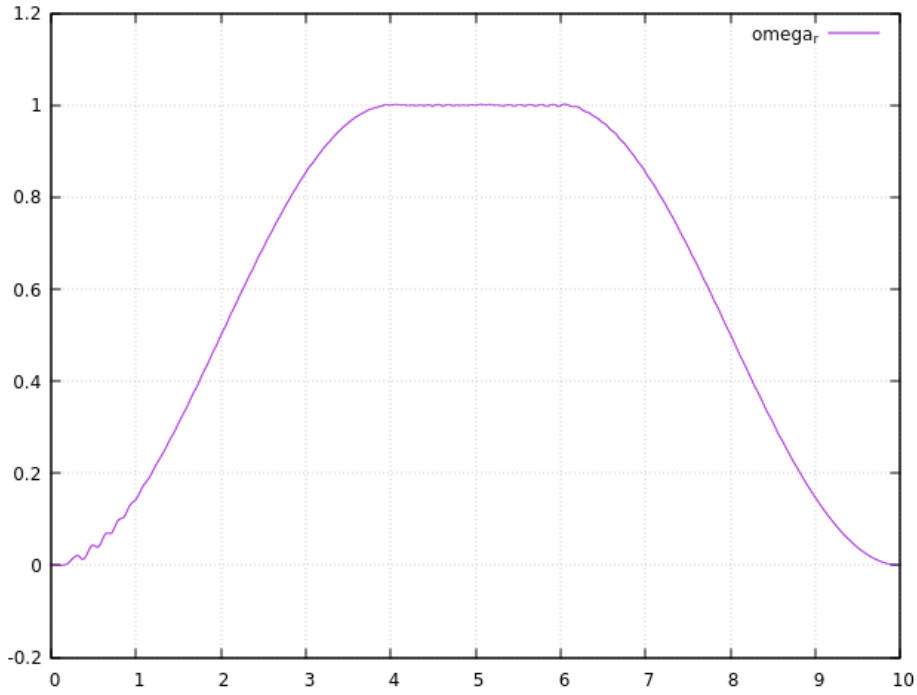




Finally, a comparison of the actual velocity omega_r for the simulation when run using "double precision" control (**FOC_tests_double_sim.hpp.pdf**) and "fixed point" control (**FOC_tests_fixed_sim.hpp.pdf**)



Double precision test run:



32 bit fixed point test run:

These results show equivalence using a control implemented in double precision floating point math and control implemented in fixed point math.

However, this is not to say that both double and fixed implementations are interchangable.
For example, if an application requires a high degree of feedback resolution, the double implementation will provide better performance.

However, these results do suggest future tests using 64 bit fixed point math which can be provided by TI's new C29 family of devices (ref F29H859TU-Q1).