Writing Control Law Code for the UAV platform

UAV research group
University of Minnesota, Twin Cities

The control laws for the UAV platform are implemented in C code. The same control code is used by the Software in the Loop simulation (SIL), Processor in the Loop simulation (PIL) and flight computer of the UAV. The control code must implement the function prototypes defined in control_law_interface.h, which can be found in the folder /Software/FlightCode/control/. The functions are listed as follows:

extern double *get_control_outputs(double * feedback, **double** *reference , **double** time, **double** sample_time)

This function returns a double array containing the outputs of the controller. The array should contain the actuator commands [aileron, elevator, rudder, throttle]. See UAV_controllaw_ICD.pdf for a full description of the control law I/O.

double *feedback : Double array that contains the feedback signals necessary for the controller. In the UAV platform, it contains the body frame angles, angular rates, and accelerations. See UAV_controllaw_ICD.pdf for a full description of the control law I/O.

double *reference: Double array containing the reference signals needed by the controller. For the current UAV platform it contains the pitch and roll reference angles.

double time: Current time

double sample_time: Controller sample time.

extern void reset_control()

This function resets the controller's internal variables to their initial values, such as an integrator state. The exact form of this function will vary depending on the controller.

Example Control Codes

Examples and baseline code can be found in the folder Software/FlightCode/control/; a brief explanation about these implementations is given as follows:

empty_control.c: Provides a basic layout for the control code. The position where the control code needs to be written is indicated in comments.

baseline_control.c: Baseline PI controller which uses angles and angular rates feedback. It has an integral action on the position errors (angles) and an integrator anti-windup strategy to handle actuator saturations. Has a simple yaw damper for the directional axis.

lqr_control.c: LQR controller that uses angles and angular rates feedback and an integrator anti-windup strategy to handle actuator saturation. Lateral-direction axes (aileron-rudder) are coupled.

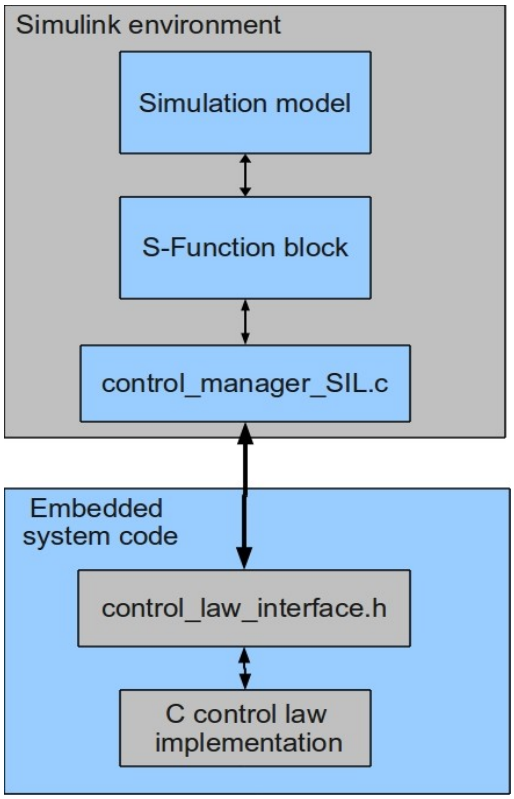
Compiling a Control Law Code

The compilation process for software in the loop and hardware are explained in the following sections.

SIL simulation

Requirements: Matlab - mex utility linked to a C compiler. Type `mex -setup` in the Matlab command line to configure the mex utility.

A schematic of the Software in the Loop Simulation interaction with the C code implementation is presented in Figure 1.



SIL and C code schematic

The Simulink model runs the UAV simulation while a S-Function block linked to a binary Mex file runs the compiled C code located in control_manager_SIL.c . Function *mdlOutputs* in *c*ontrol_manager_SIL.c receives the sensor signals and control references each simulation time step, these signals are transfered to the control_law_implementation.h control routine (***get_control_outputs***) implemented by the desired C controller. The controller returns to control_manager_SIL.c the actuator commands [aileron, elevator, rudder, throttle] that are also transferred from the *mdlOutputs* routine in control_manager_SIL.c to the Simulink model through the S-Function block. These commands are set to the actuator inputs of the UAV simulation. This cycle is repeated every time step until the end of the simulation.

control_manager_SIL.c includes the header file *control_law_interface.h* which serves as interface connecting the function definitions and the control implementation (C controller) specified in the

compilation process. The source code and header files included in *control_manager_SIL.c* are located in the *Software/FlightCode/* directory and are used by the hardware implementation as well.

The C code utilized by the SIL simulation is compiled from the Matlab environment by running the setup.m script where the desired controller must be selected. The portion of the script where this selection is done is shown as follows:

```
 %% Set controller mode
% Use this variable to quickly change what controller is used in the
% simulation.
%
% 1 = baseline controller (C implementation)
% 2 = baseline controller (Simulink)
% 3 = LQR controller (C implementation)
% 4 = LQR controller (Simulink)
% 5 = Student controller (C implementation)
% 6 = Student controller (Simulink)
controller_mode = 2;

% Load controller parameters or compile flight code
switch controller_mode
   case 1 % Baseline controller in C
      % Compile Flight Software:
          . . .
          . . .
case 5 % Student controller in C
 % Compile Flight Software:
  control_code_path ='../../Software/FlightCode/control/student_control.c'; % Specify your control code file name here
  eval(['mex -g -DSIL_MODE_  -I../../Software/FlightCode/control/ control_manager_SIL.c ' control_code_path]);

case 6  % Student controller in Simulink
        % Specify your Simulink controller parameters (gains, etc) here
```

By selecting an option in the controller_mode variable the desired controller is loaded. Two control laws (LQR and baseline) are implemented in C and Simulink which correspond to options 1,2,3 and 4. Option 5 is the student controller that provides a basic layout in C of the controller structure. The path and name of the student controller can be changed from the default by setting the path (as a string) in the variable control_path located in case 5 of the portion of code displayed above. Option 6 provides the same layout of option 5 but in a Simulink block. Note that control parameters should be specified in case 6 of the shown code.

The setup.m file is located in the SIL simulation folder : Simulation/SIL_Sim/. If a C controller is selected, errors and warnings of the compilation are displayed in the Matlab console after running the setup.m script. To use the same C controller code for both hardware implementation and Software in the Loop Simulation it is recommended to save the C control code in the folder: Software/FlightCode/control/.

Hardware implementation:

To compile the Flight computer software the make tool is used. A makefile written for the UAV platform compiles the source code. To change the selected flight controller, the makefile must be edited to specify which controller is compiled. Currently this is Line 40 of the makefile, which is as follows:

```
OBJ = L_avionics.c control/lqr_control.c
```

With this code, the controller in “control/lqr_control.c” will be compiled. The software requirements for compiling the flight computer code for the MPC5200b-tiny system are explained in the PIL simulation guide.