DANTE VILLARREAL

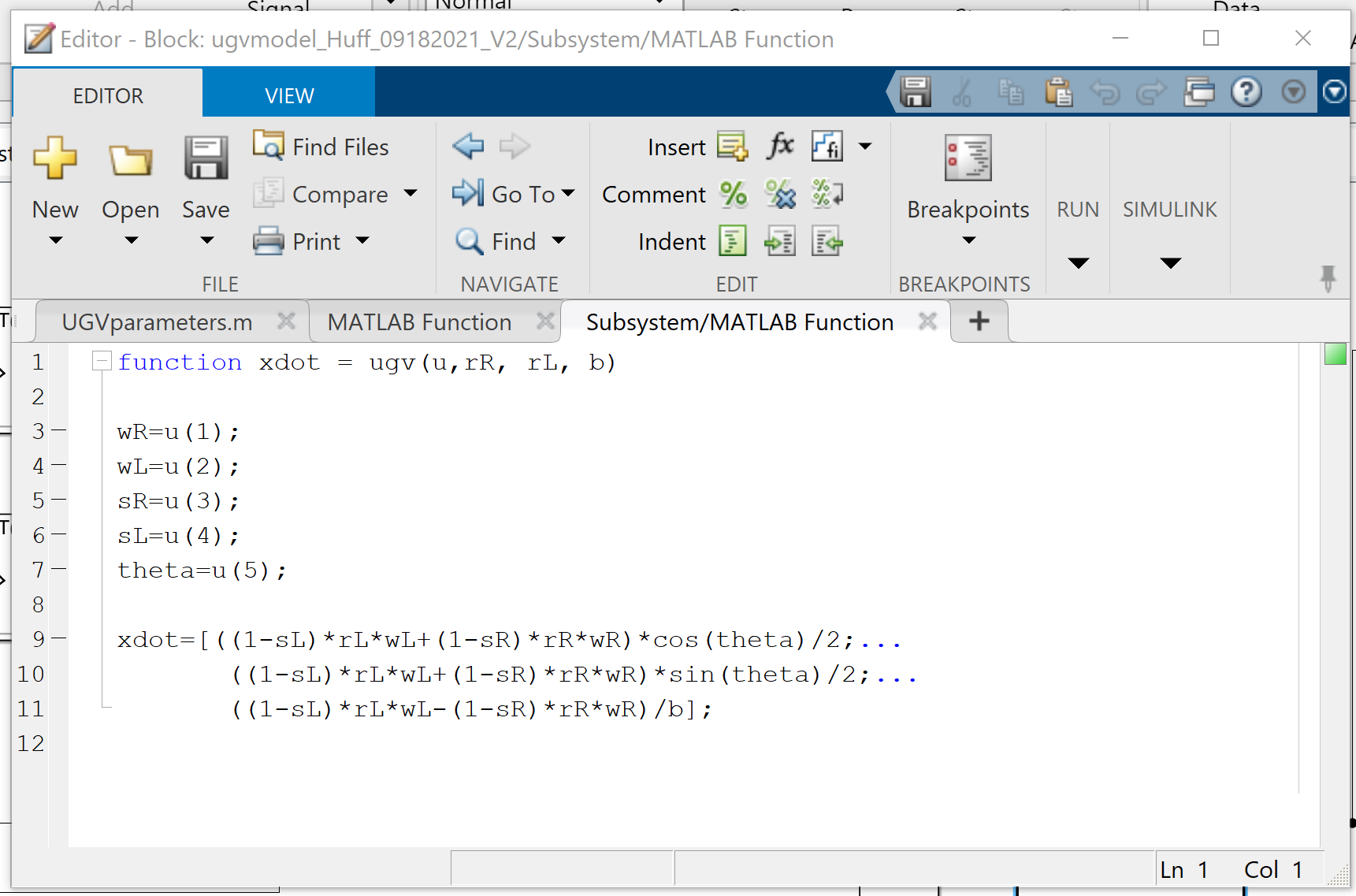
ID: 1001580411

Course: Intro to Unmanned Vehicles

Section: 5383-001

Answer Example:

This part of the model (and your knowledge of how it works) contains the information needed to answer the following example questions:



Example Question 0.1 Where in the UGV model are the Kinematic Equations of Motion that define how the UGV moves in its environment

Ans: These equations are located in the Block: ugvmodel\_Huff\_09182021\_V2/Subsystem/MATLAB Function/UGV

Example Question 0.2 What are the equations of motion?

Ans: xdot=[((1-sL)\*rL\*wL+(1-sR)\*rR\*wR)\*cos(theta)/2;...

((1-sL)\*rL\*wL+(1-sR)\*rR\*wR)\*sin(theta)/2;...

((1-sL)\*rL\*wL-(1-sR)\*rR\*wR)/b];

Example Question 0.3 What are the inputs to this Model Component?

Ans: A vector generated by a Mux Block that contains five values:

1. The angular velocity of the right track drive wheel - wR=u(1)
2. The angular velocity of the left track drive wheel - wL=u(2)
3. The percent slip of the right track – sR=u(3)
4. The percent slip of the left track – sL=u(4)
5. The heading angle of the UGV expressed in radians clockwise from the X axis for the vehicle’s navigational frame – theta=u(5)

Example Question 0.4 What is output/returned from this Model Component?

Ans: The UGV MATLAB function returns the vector “xdot” containing three caluclated values:

1. The vehicle’s instantaneous velocity relative to the Navigational Frame’s X-axis.
2. The vehicle’s instantaneous velocity relative to the Navigational Frame’s Y-axis.
3. The vehicle’s instantaneous angular velocity

Now it is your Turn.

Questions:

1.0 This set of questions will help you explain how the model simulates the forward velocity of the UGV, its position with respect to the Navigational Frame, and its current heading with respect to the Navigational Frame.

1.1 **Where in the UGV model is the above information generated? (Tell me location within the Model structure and provide a “picture” of the simulation blocks that produce these values).**

Information Generated:

This is asking about question 1. Where is the forward velocity, position and heading generated.

This image below is located in ugvmodel->Subsystem

The matlab function in blue generates 3 values: “x-dot”, “y-dot”, and “theta-dot”

Aka instantaneous velocity in x-direction, instantaneous velocity in y-direction, and turn-rate

This is in ugvmodel->Subsystem->MATLAB Function

A screenshot of a computer program

AI-generated content may be incorrect.

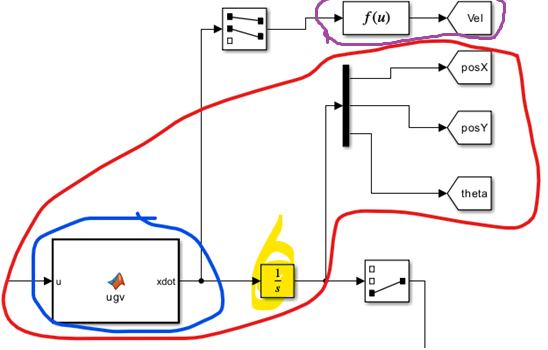
The integral of each “x-dot”, “y-dot”, and “theta-dot”, respectively, gives you “x-position”, “y-position”, and “theta” aka heading angle in radians

This integral calculation is done by the block in yellow

The “Fcn” block circled in purple generates the forward velocity by taking in “x-dot” and “y-dot”, and doing the Pythagorean theorem to find the velocity.

A screenshot of a computer

AI-generated content may be incorrect.



1.2 **What Simulink Blocks are used to calculate the UGV’s instantaneous forward velocity?**

A screenshot of a computer program

AI-generated content may be incorrect.

The ugvmodel->Subsystem->MATLAB function block. Its location is in blue and its contents are in the image directly above this. This calculates the instantaneous x and y velocity and turn rate.

Then, the “Fcn” block in ugvmodel->Subsystem takes the instantaneous x and y velocity, does the Pythagorean theorem and generates the instantaneous forward velocity

A screenshot of a computer

AI-generated content may be incorrect.

1.3 **In your own words (and pictures if that helps) explain how this velocity is calculated.**

A screenshot of a computer program

AI-generated content may be incorrect.

The ugvmodel->Subsystem->MATLAB function block. Its location is in blue and its contents are in the image directly above this. This calculates the instantaneous x and y velocity and turn rate.

The instantaneous x and y velocity are calculated using left/right slippage, left/right wheel angular velocity, left/right wheel radius

Then, the “Fcn” block in ugvmodel->Subsystem takes the instantaneous x and y velocity, does the Pythagorean theorem and generates the instantaneous forward velocity. Its contents are below

A screenshot of a computer

AI-generated content may be incorrect.

1.4 **In your own words (and pictures if that helps) explain how we calculate the simulated location and heading of the UGV with respect to the Navigational Frame**

In ugvmodel->Subsystem:  
A diagram of a machine

AI-generated content may be incorrect.

In blue, we generate the x-dot, y-dot, and theta-dot. We then put it through the yellow block, which integrates each. This gives us the x-pos, y-pos, theta. This then gives us a vector of 3 values. We then put it into the demux vertical black bar and that splits it into 3 values: x-position, y-position, theta.

1.5 **What does this Block do?**

Diagram

Description automatically generated

So the output of the ugvmodel->Subsystem->MATLAB function block is a vector of 3 values: x-dot, y-dot, theta-dot. The block above, the selector blocks “selects” just the first 2 values of that vector, aka the x-dot and y-dot. Then it feeds it into the function which does the Pythagorean theorem and obtains the instantaneous forward velocity.

1.6 **What does this Block do?**

Chart

Description automatically generated with medium confidence

So the output of the ugvmodel/Subsystem/MATLAB function block is a vector of 3 values: x-dot, y-dot, and theta-dot. The integral block then receives that vector and does the integral for each. This demux block then just splits the vector into its individual values: x, y, theta, so we can see them separately.

1.7 **What does this Block do as it specifically relates to determining the position and heading of the UGV?**

Diagram, schematic

Description automatically generatedThis is the integral block. It receives the vector containing the x-dot, y-dot, and theta-dot from the ugvmodel/Subsystem/MATLAB function block. In order to find the actual x-position, y-position, and theta, you integrate each. So the integral of x-dot is x-position, and same for y-dot, and the integral of theta-dot, aka turn rate, is the angle, theta.

2.0 **The following set of questions are designed to help you think about and explain how we “drive” our simulated UGV and how we specify environmental conditions that effect its performance.**

ok

2.1 **Where in the UGV model are the inputs to our UGV generated? (Tell me location within the Model structure and provide a “picture” of the simulation blocks that produce these values).**

Before it reaches the subsystem, the inputs are generated in ugvmodel. As seen below.

A diagram of a computer

AI-generated content may be incorrect.

In ugvmodel->MATLAB function (the input one):  
A screenshot of a computer

AI-generated content may be incorrect.

In ugvmodel->MATLAB function1 (it’s called slip):  
A screenshot of a computer program

AI-generated content may be incorrect.

2.2 **In your own words, explain what inputs are being generated to drive the UGV. What would the “Real-World” inputs be (i.e. how would a person drive the “real-world” physical manifestation of this UGV model). How do we replicate/generate these “Real-World” inputs in the simulation?**

These inputs are the left/right duty cycle and left/right slippage. The real-world inputs would be the “signal” to go a certain speed, which the UGV model would then translate to DCR and DCL. The slippage signal could come from a sensor in the left and right tracks that would detect if computed angular speed matches the actual angular speed. In this simulation, these are just generated based on the time.

2.3 **What do the following two blocks do in the simulation?**

A picture containing graphical user interface

Description automatically generatedThe “12:34” clock gives you the “time” for the ugvmodel/MATLAB function called “input” that generates the DCR and DCL based on the time, aka time since simulation began. This generates an output at a specified rate. We have it set to 1. So every ‘1’, it will output the next number. The “analog clock” outputs the current simulation time with “decimation.” So, right now the decimation is 10, so every ‘10’ time, it will generate an output.

2.4 **How do I logically specify the simulated human inputs to this manually controlled UGV? Tell me the location and show me the logic used to specify the simulated commanded inputs to our simulated vehicle. (A picture is worth 1,000 words, are at least several lines of code)**

In the ugvmodel->MATLAB Function (name=input) and ugvmodel->MATLAB Function1 (name=slip) you can specify simulated commanded inputs. You can just completely remove the “if” block, and instead of “DCR=0;” you can put “DCR=1;” for example. Same thing for the “slip” function. You can completely remove the “if” block, and instead of 0 for sR and sL, you can specify any number you would like to simulate.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

3.0 **In your own words explain the role or function of the following simulation blocks in the Model.**

Diagram

Description automatically generated

Ok so the “n-D T(u)” block is the lookup table. So this is what “translates” the DC R&L , duty cycle right and left, to actual right and left angular speeds. So from DCR->wR . This lookup table alone would give us that speed, BUT motors don’t instantaneously go to that speed, so the last block “simulates” the motor “ramping up” speed to get to that specified angular speed form the lookup table. The output of the “n-D T(u)” block is a discrete value. The middle block, the “rate transition” block converts the discrete value into a *continuous* value. Why? Because the “transfer function” block needs a continuous value. So 1st block goes from DCR to wR. 2nd block takes the discrete value from lookup table and transforms it into a continuous value. 3rd block simulates the “ramping” up of the speed.

4.0 **The following set of questions are intended to have you think about how the information stored in the MATLAB workspace relates to the UGV Simulink model.**

ok

4.1 **In your own words, explain what is meant by the phrase “a parametric model”. How is the provided UGV model “parametric”?**

This is parametric because the parameters are adjustable

4.2 **Where within our MATLAB/Simulink environment is the information that defines the physical characteristics of the vehicle and the waypoint features of the vehicle’s working environment?**

This is in the UGVparameters.m file

A screenshot of a computer program

AI-generated content may be incorrect.

4.3 **What does “b” Workspace variable represent. What role does it play in the UGV Simulation Model?**

This is the width of the UGV. We need this to calculate “theta-dot”, aka turn rate.

4.4 **What do the rp1 and rp2 variables represent?**

These give us the proximity circles. Rp1 is where you should start slowing down. Rp2 is the circle to tell you at what point to switch to the next waypoint

**4.5 How do I define the Waypoints that exist in the UGV’s simulated environment?**

You can define them in the UGVparameters.m file, in the “Way Points” section.

5.0 The following are a series of experiments or questions what we would like you to perform/answer with the model.

5.1 How/Where do you define the Simulation Run Length for a run?

5.2 Setup Your Simulation so that it has a Run Time of 500 seconds. Make the needed changes within the “Input” and “Slip” MATLAB functions to drive the UGV with 50% positive duty cycles for 30 seconds and then stop. Please remove any slip from the tracks for this experiment.

5.2.1 What is the final location of your vehicle after this simulation run? Please give me the platforms X and Y coordinates relative to the Navigation Frame.

5.2.1.1 What Simulink Block can you add to your model that will tell you your exact X coordinate value?

5.3 Change your simulation inputs so that your simulated vehicle drives to about a meter way from the waypoint defined as X = 5, Y = -3, then does a full circle around this waypoint and then drives back to the original start position located at X = 0, Y = 0. Try to stop the UGV within the outside proximity circle associated with waypoint. (I plan to try and do this by trial and error) Good luck.

Diagram

Description automatically generated

This is what I got.

5.3.1 Provide a picture of the path your UGV took.

5.3.2 Provide a picture of the “Input” MATLAB function logic you used to generate this path.

5.4 The following set of tasks/questions will allow you to explore the affects of changing the physically characteristics for the virtual UGV to change how the vehicle behaves.

5.4.1 Change you simulation environment so that you have four waypoints located at (0,0), (5,0), (5,5), and (0,5).

5.4.2 Change your “Input” MATLAB function logic so your UGV moves through your four points. Provide a picture of the “Input” MATLAB function logic you used to generate this path. When you are successful, RENAME AND SAVE THIS MODEL!!. You will be using this as your starting point for several of the following activities and questions.

5.4.3 Provide a picture of the path your UGV took through these four points.

Chart, scatter chart

Description automatically generated

This is the path my UGV generated and the four waypoints I placed in the environment. It took me a great deal of trial and error to generate this path. I will use this model and path to see the effects of changing vehicle parameters and of wheel slip. Your path does not have to be perfect but it needs to be good enough to allow you to see the impact of parameter changes and wheel slip. I had to play with both the time intervals my duty cycle setting were active and the values of the percent duty cycles to get exact distances to the waypoints of 90 degree turns. Changing the time intervals by small amounts did not necessarily generate changes in the UGV path. I was able to get much better path control by fine tuning the duty cycle values. For the straight-line segments my duty cycles were close to 100%. To get fine turn control I reduced my duty cycles to approximately 50% or so. Remember, the duty cycle dead band is between – 30% and +30% duty cycle, so do not try to go too slow.

5.4.5 How do Changes in “b” affect the vehicle path?

5.4.5.1 What does the “b” parameter in the vehicle model represent?

5.4.5.2 Decrease the “b” parameter value in your model that drives the vehicle in the 5 meter square by 20%. Run the model and save a picture of the resulting UGV path here. Explain how these results are different from the original square. Do these differences make sense to you.

5.4.5.3 Increase the “b” parameter by 20%. Provide a picture of the resulting path. Explain how these results are different from the original square. Do these differences make sense to you.

5.4.5.4 Cut the “b” to half its original value. Repeat the experiment. Tell me what happens to the path. This will be most clear if your original model accurately traces out the square.

5.4.6 Impact of wheel diameter error on the UGV path.

5.4.6.1 Which value in the UGV parameter file represents the drive wheel radius on the right side of the vehicle?

5.4.6.2 Reduce this value by 2% and re-run your 5 Meter Square Model. Provide a picture of the path generated by the UGV.

5.4.6.3 How is this Wheel Radius Error path different from your Original Square Path and the “20% b Error” Path? Can you tell them apart? Can you explain why they are different?

5.4.7 Impact of wheel/track slip on the UGV path.

5.4.7.1 In your original 5 Meter Square Model modify the “slip” MATLAB Function by adding 10% slippage on the left side of the platform for about 5 seconds of your simulation in the middle of your first straight motion segment. Show the “slip” MATLAB Function changes you have made. Provide a picture of the modified path.

5.4.7.2 Repeat the slip exercise in 5.4.7.1 but change the timing of the slip to occur when the UGV is turning.

5.4.7.3 Discuss how wheel/track slips that occur for a finite period of time during a simulation run affects the UGV path. Can you identify where a wheel slip event has occurred during a run?