

The U.S. Navy Seabase Project



Architectural Design (Crane Controller Software)

MTU Computer Science Division:

Johnathan Even

Harriet King

Anthony Oliver

MichiganTech.



Table of Contents

<u>Section</u>	<u>Page #</u>
Overview	3
Platform team	3
Crane team	3
Software team	3
History	4
Goals of project	4
Requirements	5
Simulink Overview.....	6
Crane physical design overview.....	7
Platform physical design overview.....	8
Risk analysis	9
Answers on how to port existing code.....	12
Linking	13
Differences between our code and existing code	13
List of functionality from Albuquerque	14
Questions	17
Summary	18
Glossary	19

Overview

The US Navy Seabase Senior Design project concerns itself with the design and fabrication of a scale version of a TG3637 ship crane used by the US Navy for at-sea logistics operations. The following will be a discussion of the project specifications and an explanation of the task to be completed by the Computer Science division of the project, the Platform division of the project, and the Scaled Crane division of the project.

Platform Team (Six Degrees of Freedom)

The platforms team's contribution to this project is to design a platform to simulate the wave motion of the ocean. The wave motion will fall within certain guidelines and the team's project is meant to be disassembled and reassembled quickly for demonstrations. The motion platform is meant to hold the scaled crane. The design uses a Stewart platform to simulate the motion.

Crane Team (Scaled Crane)

The crane team's contribution to the project is to design a scaled crane for use on a ship. The crane is meant to load and unload cargo from the ship while maintaining as much stability as physically possible.

Software Team

The software team's contribution to this project is to port some existing controller code from the C language into a software product called MATLAB. The code will be used in a portion of the MATLAB software called Simulink. In simulink, simulations can be run without the hardware actually being attached. When the hardware is attached it has 6 inputs from a controller called the d-Space controller. The software is meant to dampen the swing of the payload. The code will handle the motion simulated by the Platform team and stabilize the payload on the scaled crane. The software is also designed to be scaled for the large crane.

History

- A working crane model already existed at Sandia National Labs in Albuquerque. The controller for the crane was a CC2000 (which was used due to a legal binding contract). A group of files, written in C language, did the payload damping for the crane as well as helped dictate the action the CC2000 would take.
- Gordon Parker issued a simulation in Simulink of a model crane but it is not using the same type of solution that is needed in this model crane. The Albuquerque crane has the correct type of solution as we need in this project.

Goals of Project:

- To create a controller in Simulink that incorporates inverse kinematics to dampen the swing of a model crane payload on a platform modeling ship motion.
- To port existing Albuquerque code to Simulink to use in new model crane
- The servo controls for the actuators of the crane and platform are to be done by the ME/EE teams.
- The real-time control code design (CS) must be done simultaneously with the instrumentation design (EE/ME)
- A rapid-prototyping control computer system, based on MATLAB's Real-Time Workshop (e.g. DSPACE) should be used to record sensor data and send commands to the crane drive system and the motion platform drive system. Evaluation of a hardened computer control system, suitable for mounting on a ship, should be performed.
- Development of a consistent framework for the real-time control code so that it can be used directly in existing simulation codes

Requirements

Listed below are the requirements issued by the project sponsor. These are the requirements for the Computer Science division of the Seabase project. These are to be taken into consideration when porting the existing code.

Sponsor Requirements

1. The crane controller will take six platform movement inputs and two operator inputs and one crane tip sensor input and apply inverse kinematics to dictate the amount and direction of crane movement.
2. The inverse kinematics will calculate the correct amount and direction of crane movement to maintain damped swing on crane load.
3. The crane boom can only be directed to move from horizontal (zero degrees) to 85 degrees up (luff).
4. The crane boom can only be directed to move from pointing toward the ship's bow (zero degrees) to 360 degrees around in either direction and back (slew).
5. The crane boom can only be directed to hoist the load from the tip to a certain distance below the ships deck. (hoist)
6. The crane controller will port the existing code with inverse kinematics to the new controller system in D-space with simulink.

Simulink - Overview

Below is an outline of information to get a brief idea of what simulink is and the purpose it serves in this project. Simulink will function as an interface between the dspace controller and the crane. It will do all the calculations, filtering, kinematics, etc. needed for the project's completion.

1. Will need to use s-functions in simulink
2. Flag in C code (ex. Flag1, flag2) sets what function in template is called
3. Dspace has plugs that bring in the inputs and take out the outputs.
4. The point of the simulation was to show us how C code fits into the s-functions and help us quickly become expert at s-functions
5. To make something in simulink, do program, it's not click and drag/drop
6. There can be only six digital signals input to the crane controller in dspace and dspace representative is hedging on having RS232 interprocessor communication working
7. Eleven total control inputs and dspace only has room for six digitally
8. Swing sensor input could be analog maybe
9. In matlab/simulink, no good way to have a call to a subroutine, there's some kind of performance hit when go to a subroutine, so can't standardize the subroutines. Instead have to copy each subroutine into the front of each s-function so each s-function has those subroutines (example: multiplication of matrices)
10. Can make simulink model with just control and test it with seven “pretend” inputs to see if it's workings
11. After we identify how many inputs and outputs need for each s-function, then putting s-functions on program without any code yet, but correct number of inputs and outputs and then test by dragging in a sin wave from matlab, this is way to work up a framework.
12. Most important is to understand the parameters to solution() function, then can replace the matlab sin wave input with the dsapce input once connected to the sensors
13. Simulink has generic blocks called s-functions and can write C code in a file, compile it and will run within the s-function
14. Dspace system will control the crane
15. Our code will be in matlab
16. Analog/digital chip converter inputs to function which outputs to other functions
17. Dspace system monitors the legs of the Stuart platform
18. Dspace hardware has drivers that work on matlab/simulink hardware
19. Looked at matlab/simulink in lab: once matlab running, get prompt>> type “helpdesk” and get directory down left side, go to simulink | writing s-functions | overview or simulation stages to understand how it works, then can understand the C code

Crane Physical Design – Overview

Below is what is needed to get caught up to par with what is going on with the crane team's design. It is more important to focus on this group because this group is what the code is being ported to.

1. Rider block moving does not control damping, but might have effect on inverse kinematics. The rider block is for “shortening” the arm of the crane so it can deposit/pick up loads closer under the crane arm
2. crane tip sensor is reading the angle of hoist wire from vertical and is input to the inverse kinematics
3. two joysticks are planned for crane operator (so two inputs)
4. INPUTS:
 - sensors on crane tip for swing angle
 - (two directions, so probably two inputs)
 - sensor measuring force on cable (to read information, not related to controller)
 - one motor for rider block raise and lower
 - two motors for rider block movement (operator controlled)
 - one motor for hoist
 - one motor for luff
 - one motor for slew
 - ELEVEN INPUTS: 6 for platform, 2 for swing, 3 for operator
5. experimenter might command two degrees of roll, one inch of heave (up/down) and control strategy needs to know that
6. the crane controller needs to make the crane move in a pattern that cancels out any motion of the ship so the load doesn't swing
7. rider block has tag lines and currently are on a single winch so sideways distance is always equal to each other, in future might have separate winches so can pull load out of plane as a way of damping the load swing, but that's not part of this project
8. operator commands such as “go forward” but if ship is moving so crane needs to go backward, then those two are “added on” so the actual movement is corrected
9. crane will have both input and output from computer to machinery
10. Computer to control the cables and some sensors on crane gather info and act on it.
11. The “act on” part is algorithms that probably will be added much later, just need a way to add. Central part of this project is to leave it open to customization later

Platform Physical Design

Below is the basic design of the platform used in the crane simulation. The purpose of the platform is to simulate wave motion that the software will counteract with inverse kinematics, filtering, etc.

- platform team has goal of sending reading of actual platform movement to the crane controller, but if at last minute that's not possible, then can switch to platform operator commands for input about what platform is doing (used in inverse kinematics). Gordon wants it to be actual movement of platform because it could be different from platform operator commands.
- platform motion is input to crane control and crane control does all inverse kinematics to damp the load swing
- Gordon expects each platform leg's actuators to have a sensor. On the actual ship there are six sensors that synchronize with Global Positioning System (GPS)
- encoder is a measurement on legs that knows the position of the leg, or can measure the movement with some LDBTs

Risk Analysis

A basic risk analysis was performed at the beginning project, from a broad range of risk to sever risk that could alter the projects completion. The idea of this was to try and make predictions for scenarios that might occur and cause altercations within the project. As seen in the following pages many of the risks assessed were seen within the course of the research. The risk matrix prepared below to demonstrate what risk were analyzed and which ones that were confirmed.

Legend

Impact scale:

- L = Life threatening
- P = Project threatening
- E = Expensive in cost or time
- S = Some cost or time penalty
- N = Negligible cost or time penalty

Seabase Project – Computer Science Division

Risk #	Risk Description	Causes	Management	Risk Reduce or Respond	Source of Uncertainty	Nature	Probability	Impact
1	Personality conflicts	3,4	Improve communication and address problems	Reduce	Estimate	Members of our group don't get along or we don't get along with individuals in the other senior design groups	Likely	S
2	Poor Communication	1,5	Communicate more often and/or more effectively	Both	Estimate	Our group is unable to properly communicate information to each other or the other senior design groups	Unlikely	P
3	Project/course expectation conflicts	2,6	Find middle ground and agree to meet standards	Reduce	Estimate	Team members expect different things out of the course/project	Likely	S
4	Work ethic conflicts	3	Improve communication and address problems	Respond	Estimate	Team members expect more work/progress out of each other or other senior design groups	Likely	S
5	Poor organization/work distribution	2,3,4	Reevaluate plan and make new schedule	Both	Estimate	Planning and scheduling is done poorly	Likely	S
6	Don't understand project/topic	2,5,9,12, 13,14	Improve communication with other groups/team members	Both	Estimate	Individuals don't understand goals of project or specific technical aspects of the project	Unlikely	S
7	Scope too big or keeps changing	2,3,5,12, 10	Set deadline for requirements signed off and document decisions	Both	Estimate	Requirements of the project grow too large or change too often, preventing us from making scheduling realistic goals	Unlikely	S
8	Project prematurely ends before completion	15	Finish course requirements with provided information	Out of our control	Event	US Navy stops funding this project or MTU decides not to further work on project	Very unlikely	P
9	Resources (physical and leadership)	8,2,6,7,1 5	Team initiates more communication with project leaders and/or other groups	Respond	Event/ Estimate	There is insufficient equipment/tools available or there is lacking leadership from supervisors	Unlikely	S

Seabase Project – Computer Science Division

Risk #	Risk Description	Causes	Management	Risk Reduce or Respond	Source of Uncertainty	Nature	Probability	Impact
10	Poor planning/organization	2,4,6	Discuss general schedule with groups	Reduce	Estimate	Project roles are not clear to each group	Unlikely	S
11	Missed deadlines	1-7,9,10	Proper scheduling and goal assessment	Both	Estimate	Group doesn't meet project deadlines, may delay other work	Unlikely	S
12	Incorrect requirements	2,6,7,9,10	Have requirements review and signoff	Reduce	Estimate	Team misunderstands project	Unlikely	P
13	Requirements not met	2,6,7,9,12	High level of communication between groups/leaders. Design review and signoff	Reduce	Estimate	Design doesn't fulfill requirements	Unlikely	P
14	Bad design	6,11,12,13	Creative and effective solutions	Reduce	Estimate	Hasty, uninformed decisions	Unlikely	P
15	Lose team member	1,3	Reschedule project and remove least useful requirements	Respond	Event	For whatever reason, team member leaves group	Very Unlikely	L
16	Individuals don't know how to use equipment		Ask questions and receive training	Respond	Estimate	Team members have no training/experience with technologies	Likely	S
17	ME's don't finish specs required for sponsor payment for laptop			Respond	Event	ME's don't understand/can't complete	Unlikely	S
18	Payment for laptop delayed or denied	17		Respond	Event	Insufficient resources to complete project	Unlikely	S
19	Disorderly/unproductive meetings	2,5,6	Plan ahead with goals for meetings	Reduce	Estimate	Team disorganized and unprepared	Unlikely	S

Answers on how to port existing code

1. The pcs.c file is the main file for the controller because it has the main function.
2. three major things in code we need to do:
3. inverse kinematics, solution() function, measure platform movement and adjust crane tip
4. active swing damping: boom tip swing sensor (swing of load), need to calculate to damp out and keep boom tip over the payload
5. command filtering: operator's commands smoothed out between joystick and motion of crane (including adding for inverse kinematics)
6. the controll.c file has the math for the inverse kinematics and the control() function that is the main function
7. We need all the *functionality* from controll.c
8. There are many global variables and structs in this group of files that controlled the Albuquerque crane
9. Need to figure out where to put which parts of Albuquerque code to put in our simulink version
10. might be parts of the Albuquerque code that are incorrect or inefficient and we should improve on them
11. the math and inverse kinematics are not scalable, they are dictated by the configuration files (in Albuquerque code)
12. Albuquerque code does not have the rider block in it, but since rider block doesn't effect inverse kinematics, we can copy math from the Albuquerque code
13. some of code deals with the rider block but just to read operator joystick and send commands
14. Gordon says to look for velocity steering law in existing code, which was used to tinker with the speed
15. Gordon recommends figuring out exactly what are the seven things that are sent to controll.c->control() function when pcs calls control() and understanding those seven things.
16. CS controller requirements could include: no bugs, works under xyz conditions, fast enough for real time running
17. good way to work is take existing code, make flow chart of it, then segregate the code into what need and what not need, then with reduced set of code, do flow chart and find bogus paths (bugs)
18. do the code in GUI on laptop and when build it, have to go to control board (connected by Ethernet) (from 9-10-04 minutes)

Linking

- pcs.c->solution() function is the link function between pcs and control1.c code
- Pcs.c->main() function calls solution() function which calls control1.c->control() function. Solution() function is the main thing we need from psc.c.

Difference in Existing Code and Our Code

- The pcs.c file read the sensor and other data, using low level DOS to query the com-ports for input (which dspace will do for us on this controller we're making)
- pcs.c was the link between the control1.c and the machinery in Albuquerque crane, all the parsing and weird techniques to read sensors in pcs.c will be done by dspace for us
- In pcs.c main there's real time stuff where they measured how long each function takes and hard coded time increments into the program (instead of having it event driven)
- the Albuquerque crane was not controlled by simulink or dspace and had nothing to do with either of these off-the-shelf applications

List of functionality from Albuquerque

control.h

```
/* declare subroutines */
void control(int Init,
             int Init2,
             struct crane crane,
             struct command *command_motor,
             struct ship shp_mot_sens,
             struct swing swg_ang,
             float *enc_pos);
void matrixmult(float (*Mt1)[4], float (*Mt2)[4], float (*Mat)[4]); //code is to compute a matrix multiplication op
struct coord vectormult( float M[4][4], struct coord vi); //more math
void printmatrix( float A[4][4]);
void printvector( struct coord v);

void quartic(double a, double b, double c, double d, double (*roots)[2]); //find the roots of a fourth order polynomial
double cubic(double p, double q, double r); //more math
double arccosh(double x); //more math
double arcsinh(double x); //more math
double csqrtr(double x, double y); //more math
double csqrti(double x, double y); //more math
double dot(double *x, double *y, int n); //take the dot product of two arrays
float norm( struct coord v);

struct crane inv_kin(struct coord Pc, struct coord gc); //perform the inverse kinematics to cancel ship sway
struct swing swing_angles(struct crane com_pos, struct coord gc); //when line is along gc
```

```

struct crane input_shape(int Init, struct crane com_vel, float L); // initialize??
struct crane sway_damp(struct swing swing_pos_actual); //damp payload sway by adding damp signal to command signal

void compute_motor_rates( struct crane pd_vel,
                          struct crane act_pos,
                          struct command *command_motor);

float encoder_velocity(int chan, float pos);
float swing_deadzone(float angle);

void homotrans(float dx, float dy, float dz, //Compute the homogeneous transformation matrix
               float rx, float ry, float rz,
               float M[4][4]);
void homoinv( float M1[4][4], //Compute the inverse of a homogeneous transformation matrix
              float M2[4][4]);
void homopedestal(float phi, //compute ... to map from the pedestal frame to the crane frame
                  float dx, float dy, float dz,
                  float M[4][4]);

void vel_steering(int first, //implements a velocity steering law
                  struct crane *des_pos,
                  struct crane *act_pos,
                  struct crane *pd_vel);
void compute_inertial_payload_position(
    struct crane *crane_states,
    struct swing *swing,
    float Tic[4][4],
    struct coord *position);
void limit_joystick( struct crane *joystick, //limits the joystick signals if crane joints are out of range
                    struct crane *act_crane_pos);

```

```

double input_shape_hoist(int init, double hoist);
struct coord input_shape_cart(int Init, struct coord Pn, float L);
struct swing InertialSwingAngles(float NC[4][4], //Computes the inertial swing angles from the boom
    struct swing swing_boom,
    struct crane crane_pos);
void DirCos321_Control(float yaw, //mapping ship frame components to inertial frame components
    float pitch,
    float roll,
    float NS[3][3]);
struct coord CartesianDamping(int Init, struct swing *swg_inert, double Lh); //inertial payload correction to dampen swing
struct coord input_shape_deck(int Init, struct coord dP, double ahat, float L);
double findRoot(double L); //Finds the optimum input shaping gain assuming that...
double find_ahat(int init, double L); //Finds the optimum input shaping gain assuming that...
void des_pos_predict(int first, //predicts the current servo command rate, then compensate for lag
    struct crane *despos);
struct swing Nominal_inertial_swing( //Computes the nominal inertial swing angles
    int Init,
    struct coord Pn,
    double hoist);
void SwingFilterBP(int Init, //first order bandpass filter on the swing states
    double rad,
    double lat,
    struct swing *ans,
    double L);
void limit_vel_steering(struct crane *error);

```

Questions

1. In pcs.c main there's real time stuff where they measured how long each function takes and hard coded time increments into the program (instead of having it event driven)
2. Contoll.c takes readings of ship motion and operator commands and has all the info about the ship.
3. There are many global variables and structs in this group of files that controlled the Albuquerque crane
4. experimenter might command two degrees of roll, one inch of heave (up/down) and control strategy needs to know that
5. Gordon says to look for velocity steering law in existing code, which was used to tinker with the speed
6. do the code in GUI on laptop and when build it, have to go to control board (connected by Ethernet) (from 9-10-04 minutes)
7. flag in C code (ex. Flag1, flag2) sets what function in template is called (#2 unders "simulink")
8. if math and inverse kinematics are dictated by the configuration files, then how are they dictated in simulink version?
9. Number 10 under "Answer to how to port" , some code deals with rider block? In existing Albuquerque code?? or in what we need to make

Summary

Overall communication in this project went very poorly the first semester. It was not concrete how exactly the Computer Science division was part of the project. After learning that designing was not part of the project and rather porting the code was what needed to be accomplished. This project has shifted directions very often and caused many of the risk to be confirmed. The scope of the project is now clear. The goals and proper planning to be successful next semester are understood.

Glossary

- pcs - pendulum control system
- posmv - measured ship motion and we can read motion straight from Stuart platform, so everywhere in code see posmv, can skip it
- slew (alpha) - rotation around a vertical axis
- luff (beta) - is angle of drop in the tip of the crane arm (crane pinned at platform end and no movement there)
- length hoist - vertical drop of cable with loads
- bandwidth – reaction timing of the control motots.
- Ahat - the command shaping changes in the length of the crane cable