OBJECTIVE DIRECTED CONTROL USING LOCAL MINIMISATION FOR AN AUTONOMOUS UNDERWATER VEHICLE

Lars Alminde, Jan D. Bendtsen, Jakob Stoustrup [*]
and Kristin Y. Pettersen [**]

[*] *Department of Electronic Systems, Section of Automation and Control, Aalborg University,*
{alminde, dimon, jakob}@es.aau.dk
[**] *Department of Engineering Cybernetics, Norwegian University of Science and Technology,*
kristin.y.pettersen@itk.ntnu.no

Abstract: This paper presents a new method for control of complex MIMO plants based on a quantised state description. The control algorithm solves a minimisation problem for a set of user defined convex control objective functions with the plant dynamics as a constraint. The solution makes use of local linear models that are effectively calculated by propagating the state on-line using a quantised state description. The method is demonstrated on a model of an autonomous underwater vehicle.

Keywords: non-linear MIMO control, optimisation, quantised state system

1. INTRODUCTION

Models for intelligent autonomous systems, whether land, sea or space based, are in general complex in terms of the dimension of the state-space, the number of in- and outputs, and the non-linearities in the driving equations.

This complexity makes it inherently difficult to design model based controllers for the high-dimensional non-linear plants using analytic methods such as e.g. back-stepping or sliding-mode control, see (Khalil 2000). Therefore, such problems are often attacked by a combination of linearisation and decoupling techniques in which control of the complex plant is made into one or more less complex problems.

For instance in Healey and Lienard (1993) controllers for an Autonomous Underwater Vehicle (AUV) are designed by separating the complex Multiple-Input-Multiple-Output (MIMO) system into three single-input-single-output (SISO) systems for which controllers are designed using a sliding mode approach. In the simplification from the MIMO system to the assumed decoupled SISO systems some actuators are left unused.

This paper presents a control algorithm for MIMO plants where the control algorithm on-line solves a minimisation problem for a set of user specified convex control objective functions with the plant dynamics as a constraint. The approach resembles Model Predictive Control (MPC), but the application of a quantised state systems approach allows the optimisation problem to be efficiently reformulated based on local linear models which can be minimised with a simple algorithm. This help allowing input sequences to be calculated in real-time, which is often not feasible, see e.g. (Oort *et al.* 2006). The approach is demonstrated through simulations of an AUV model based on Healey and Lienard (1993).

The paper is organised as follows; First an introduction to quantised state systems is presented in Section 2, hereafter the proposed control method is developed in Section 3. Section 4 describes the AUV model and describes the control objective functions used and the acquired simulation results. Finally, Section 5 presents conclusions.

2. QUANTISED STATE SYSTEMS

Quantised State Systems (QSS) is a recent approach for propagating ordinary differential equations by decoupling the states and transforming the system into a discrete event system, where event times are dynamically decided using a quantum separation criteria between two models of different order for each state. The approach was developed by Kofman (2002). This paper will make use of the QSS2 algorithm in which the event time is decided using the difference between a first order and second order model. This technique is reviewed in the remainder of this section, and further details can be found in Kofman (2002).

The QSS2 algorithm simulates systems of the following form, with state vector $\mathbf{x}$ of dimension n and input vector $\mathbf{u}$ of dimension m:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{1}$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) \tag{2}$$

The QSS2 algorithm integrates the state and produces the output $\mathbf{g}(\mathbf{x})$ by decoupling the system into event-communicating software entities representing functions and integrators, see fig. 1.

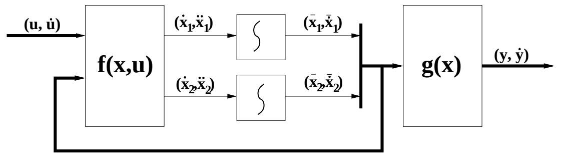


Fig. 1. Structure of a QSS2 simulation. Thick lines represent vector signals. This figure represent a system with two states

The following describes how the integrator blocks and function blocks work, respectively.

2.1 QSS2 Integrators

The integrators maintain a first order and second order model of the state trajectory as a function of time since the last update, $\tau = t - t_k$.

$$\bar{x}_i(\tau) = \bar{x}_i(t_k) + \dot{\bar{x}}_i\tau \tag{3}$$

$$x_i(\tau) = x_i(t_k) + \dot{x}_i\tau + \frac{1}{2}\ddot{x}_i\tau^2 \tag{4}$$

where $\bar{x}_i \in \mathbb{R}$ and $x_i \in \mathbb{R}$ are the i'th states in the first- and second order models, respectively.

The second model is updated whenever an external event occurs, i.e. new information from the block representing $\mathbf{f}(\mathbf{x}, \mathbf{u})$, while the first model $\bar{x}_i(\tau)$, is updated when the difference between the two models exceeds a preset *quantum*, i.e. when:

$$|\bar{x}_i(\tau) - x_i(\tau)| > \Delta Q \tag{5}$$

where ΔQ is the chosen quantum. At each event the next internal event time is calculated by solving for τ in eq. (5).

The above scheme is sketched in fig. 2, which provides an example trajectory. Here, it can be seen how the models for $\bar{x}_i(\tau)$ and $x_i(\tau)$ are allowed to evolve independently whereafter upon reaching the difference ΔQ are reset to the same condition.

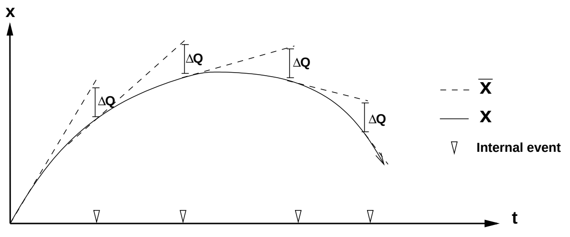


Fig. 2. A sample QSS2 trajectory

With this formulation one can think of $(\bar{\mathbf{x}}, \dot{\bar{\mathbf{x}}})$ as an operating point, or rather an *operating trajectory*, with the guarantee that it is correct to within the chosen quantum within the time interval until the next event.

2.2 State and Output Maps

Whenever an integrator produces a new operating trajectory, $(\bar{\mathbf{x}}, \dot{\bar{\mathbf{x}}})$, this is communicated to the blocks that are connected to the output of the integrator. For fig. 1 this means $\mathbf{f}(\mathbf{x}, \mathbf{u})$ and $\mathbf{g}(\mathbf{x})$. These function blocks then calculate new outputs which depend on the input variable[1].

In order to calculate the second derivative; the block numerically derives a matrix with second order information as in the first order Taylor expansion of $\mathbf{f}(\mathbf{z})$ with $\mathbf{z} = [\mathbf{x}^\mathbf{T}\ \mathbf{u}^\mathbf{T}]^\mathbf{T}$ (equivalently for $\mathbf{g}(\mathbf{x})$):

$$\mathbf{f}(\mathbf{z}(\tau)) = \mathbf{f}(\mathbf{z}(t_k)) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{z}}(\mathbf{z}(t_k))\right]\dot{\mathbf{z}}\tau \tag{6}$$

Each time there is an integrator event or an input event the corresponding column in $\frac{\partial \mathbf{f}}{\partial \mathbf{z}}$ is updated.

[1] At initialisation, a static coupling analysis of the equation set is performed

2.3 QSS2 Properties

The properties of the QSS2 approach to ODE propagation have been established in Kofman (2002). Most importantly it is proven that the QSS2 simulation converges to a region around the equilibrium of the original system, where the size of the region is a function of the selected quantum size. For practical applications, the quantum can be selected small enough to make the region small compared to system noise.

For the application of QSS2 in this paper the benefits are:

- the operating trajectory is updated more often the more non-linear the model is
- local linearised models of the system is efficiently produced by the simulation
- the time interval of the local linearised model is known as the time until the next scheduled integrator event

The algorithm has been implemented in a Java-based library for execution of discrete event models, see (Alminde *et al.* 2006).

3. OBJECTIVE DIRECTED CONTROL

This section develops the control algorithm. First, control using a single control objective will be described and secondly it will be described how this can be extended to multiple control objectives controlled by non-overlapping sets of actuators.

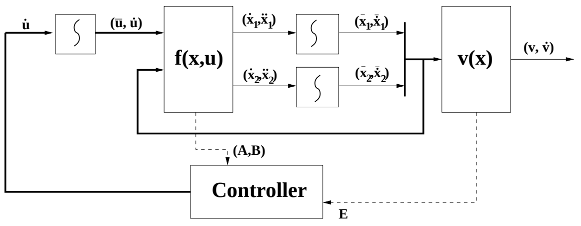
3.1 Single Objective Control

For control purposes we will augment the system with a control objective in the form of a scalar convex control objective function which maps the state to a scalar. The control system looks like ($v(\mathbf{x})$ here takes the place of $\mathbf{y}$ of the previous section):

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{7}$$

$$v = \mathbf{v}(\mathbf{x}) \tag{8}$$

The control problem is to find input signals that minimise $v = v(\mathbf{x})$, where v is an appropriate performance measure, e.g., measuring the weighted distance to a desired output value. The QSS2 algorithm provides the Hessian $\frac{\partial \mathbf{f}}{\partial \mathbf{z}}(\mathbf{z}(t_k))$, see eq. (6), which can be divided into two matrices $\mathbf{A}(\mathbf{x}, \mathbf{u})$ and $\mathbf{B}(\mathbf{x}, \mathbf{u})$ representing state sensitivity and input sensitivity respectively. Similarly for $v(\mathbf{x})$ the QSS2 algorithm provides a state sensitivity matrix $\mathbf{E}(\mathbf{x})$, which is the Jacobian of eq. (8). These matrices are communicated to the controller, and



$\dot{\mathbf{u}}$
$(\bar{\mathbf{u}}, \dot{\mathbf{u}})$
f(x,u)
$(\dot{\mathbf{x}}_1, \ddot{\mathbf{x}}_1)$
$(\dot{\mathbf{x}}_2, \ddot{\mathbf{x}}_2)$
$(\bar{\mathbf{x}}_1, \bar{\dot{\mathbf{x}}}_1)$
$(\bar{\mathbf{x}}_2, \bar{\dot{\mathbf{x}}}_2)$
v(x)
$(\mathbf{v}, \dot{\mathbf{v}})$
(A,B)
Controller
E

Fig. 3. Control structure. Thick lines are vector signals. Dashed lines are matrix signals. This figure has two states and one input

the controller output $\dot{\mathbf{u}}$ is fed to the plant through a set of integrators, see fig. 3.

The choice of letting the controller control the input slopes $\dot{\mathbf{u}}$ rather than $\mathbf{u}$ directly, is due to the fact that the matrices used in the calculation are based on a fixed operating point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$, so any immediate change in the control signal would render the matrices incorrect, whereas by controlling through the slopes the QSS2 mechanism for automatically switching operating points is intact.

Based on the information in the matrices we can formulate the following local model for $\ddot{\mathbf{x}}$ and $\dot{v}$ in the operating point of $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$:

$$\ddot{\mathbf{x}} = \mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}})\dot{\mathbf{x}} + \mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}})\dot{\mathbf{u}} \tag{9}$$

$$\dot{v} = \mathbf{E}(\bar{\mathbf{x}})\dot{\mathbf{x}} \tag{10}$$

The control strategy is to provide an input signal that keeps $\dot{v}$ negative. To this end we need to see how $\dot{v}$ is affected by the control vector $\dot{\mathbf{u}}$ over a time horizon τ. This is achieved by inserting a solution to eq. (9) over the time horizon into the expression for $\dot{v}$, i.e. eq. (10):

$$\dot{v}(\tau, \dot{\mathbf{u}}) = \mathbf{E}(\bar{\mathbf{x}}) \left(\mathbf{\Phi}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \tau)\dot{\mathbf{x}} + \mathbf{\Gamma}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \tau)\dot{\mathbf{u}} \right) \tag{11}$$

where $\mathbf{\Phi}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \tau)$ is the state transition matrix:

$$\mathbf{\Phi}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \tau) = \mathbf{I} + \sum_{k=1}^{\infty} \frac{\mathbf{A}^k(\bar{\mathbf{x}}, \bar{\mathbf{u}})\tau^k}{k!} \tag{12}$$

and $\mathbf{\Gamma}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \tau)$ is the input transition matrix:

$$\mathbf{\Gamma}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \tau) = \sum_{k=0}^{\infty} \frac{\mathbf{A}^k(\bar{\mathbf{x}}, \bar{\mathbf{u}})\tau^{k+1}}{(k+1)!} \mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \tag{13}$$

However, since fig. 3, introduces the new state $\mathbf{u}$, we also need to penalise this state in the control objective function in order to ensure that control signals approach zero as the control objective function is minimised. To this end we use a quadratic cost term: $(\mathbf{u} - \mathbf{u_f})\mathbf{P}(\mathbf{u} - \mathbf{u_f})$, with parameters given by the matrix $\mathbf{P} = \mathbf{P^T} > 0$, and possible preset input levels, $\mathbf{u_f}$. Augmenting to eq. (11) the control objective derivative becomes:

$$\dot{v}^*(\ \tau\ , \dot{\mathbf{u}}, \mathbf{u}) = \mathbf{E}(\bar{\mathbf{x}})(\mathbf{\Phi}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \tau)\dot{\mathbf{x}}$$

$$+ \mathbf{\Gamma}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \tau)\dot{\mathbf{u}}) + \tau(\mathbf{u} - \mathbf{u_f})\mathbf{P}\dot{\mathbf{u}}^{\mathbf{T}} \quad (14)$$

In order to minimise eq. (14) we can only minimise terms depending on, $\dot{\mathbf{u}}$. Therefore, we neglect terms not depending on $\dot{\mathbf{u}}$, and end up with the following term that we will denote, $\mathbf{c}$:

$$\mathbf{c}(\tau, \mathbf{u})\dot{\mathbf{u}} = (\mathbf{E}(\bar{\mathbf{x}})\mathbf{\Gamma}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \tau) + \tau(\mathbf{u} - \mathbf{u_f})\mathbf{P})\, \dot{\mathbf{u}} \quad (15)$$

The minimisation will be performed over a control horizon that corresponds to the next scheduled integrator output event in the QSS2 simulation, since it is known that the matrices used in the above calculations are constant within this time horizon. Furthermore, since it is not possible to evaluate the infinite sum of eq. (13), it is approximated using the first n terms, where n is the number of states in the system. This choice ensures that minimisation includes full information about the state controllability. The minimisation problem can now be stated as:

$$\begin{aligned} &\text{minimize } \mathbf{c}(\tau, \mathbf{u})\dot{\mathbf{u}} \qquad (16)\\ &\text{subject to:}\\ &\quad \dot{\mathbf{u}} \preceq \dot{\mathbf{u}}_{\max}\\ &\quad \dot{\mathbf{u}} \succeq \dot{\mathbf{u}}_{\min} \end{aligned}$$

where the constraints $\dot{\mathbf{u}}_{\max}$ and $\dot{\mathbf{u}}_{\min}$ are computed in each control step to satisfy both rate constraints and saturation constraints over the control horizon. Denoting the minimiser $\dot{\mathbf{u}}^*$ it can be seen that due to the simple dot-product form of the problem then each component, index i, of $\dot{\mathbf{u}}^*$ can easily be found as:

$$\dot{u}_i^* = \begin{cases} \dot{u}_{\min,i} & \text{if } c_i > 0 \\ \dot{u}_{\max,i} & \text{if } c_i < 0 \end{cases} \quad (17)$$

The minimiser $\dot{\mathbf{u}}^*$ is also the minimiser of eq. (11) in terms of the parameter $\dot{\mathbf{u}}$ and hence represents the input that gives the most rapid decay of the control objective function. If the minimiser results in a positive number for $\dot{v}$ in eq. (11), then there is not enough control authority to stabilise the system at this operating point, but the controller will provide the control input that gives the least possible growth of the objective function.

3.2 Multiple Objective Control

An extension to multiple objective control is possible in cases where the actuators can be divided in complementary sets assigned to one objective function. For multiple objectives the control objective function will no longer be scalar, i.e.:

$$\mathbf{v}(\mathbf{x}) = \begin{cases} v_1(\mathbf{x}) \\ \vdots \\ v_l(\mathbf{x}) \end{cases} \quad (18)$$

where each scalar function $v_k(x)$ of $\mathbf{v}(\mathbf{x})$ is assigned an actuator set a_k such that:

$$\bigcap_{k=1}^{l} a_k = \emptyset \quad (19)$$

For multiple objective control eq. (16) and eq. (17) are solved independently for each control objective function.

3.3 Control Procedure

The following list of actions gives an overview of how the control will be applied to the QSS2 system.

(1) The QSS2 model executes the next event and distributes information. New matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{E}$ are calculated in the process
(2) The controller solves eq. (17)
(3) New input slopes $\dot{\mathbf{u}}$ are applied
(4) The QSS2 model re-evaluates trajectories event times given new inputs
(5) Repeat from step 1

Control set allocation, i.e. separating the controls into non-overlapping sets, a_k, can be changed dynamically with no overhead. This is interesting for plants where actuators availability may vary, e.g. in the presence of faults.

4. CONTROL OF AN AUTONOMOUS UNDERWATER VEHICLE

For the simulation study of the control method the AUV model described in Healey and Lienard (1993) will be used. The AUV is 5.3m long and weighs 5.4T. A sketch of it is shown in fig. 4.

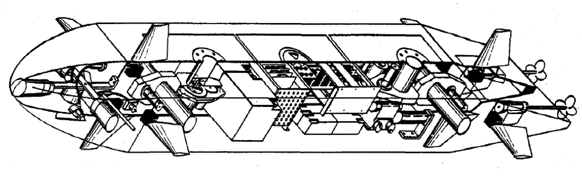


Fig. 4. A sketch of the Naval Postgraduate School Autonomous Underwater Vehicle (from (Healy and Lienard) (1993))

The model contains 12 states, six of which are described in a body-fixed coordinate system and six described in an assumed inertial North-East-Down frame (NED). The body fixed states are:

$$\mathcal{V}(t) = [u(t),\ v(t),\ w(t),\ p(t),\ q(t),\ r(t)] \quad (20)$$

which respectively represent: surge speed, sway speed, heave speed, roll rate, pitch rate, and yaw rate. The state variables in the NED frame are:

$$\eta(t) = [x(t),\ y(t),\ z(t),\ \phi(t),\ \theta(t),\ \psi(t)] \quad (21)$$

which respectively represent Earth fixed: x-, y-, and z-position, roll-, pitch-, and yaw-angle. The AUV has six controllable actuators:

$$\mathbf{u}(t) = [\delta_r(t),\ \delta_s(t),\ \delta_b(t),\ \delta_{bs}(t),\ \delta_{bp}(t),\ n(t)] \quad (22)$$

which respectively are: rudder, stern plane, top and bottom bow plane, starboard bow plane, port bow plane, and propeller speed. All control planes saturates at $\pm 20^o$ and the propeller can run at between 0 and 1500 RPM.

The model takes into account the following effects:

- Rigid body mass and added mass due to hydrodynamics
- Coriolis and centripetal forces and torques including added mass effects
- Hydrodynamic dampening forces and torques
- Propulsion forces and control torques
- Gravitational and buoyancy forces and torques
- The kinematical relation between the body and NED frame

The model is included in the "Marine GNC Toolbox" (Fossen 2002). The model contains around 120 constant parameters and consists of 30 non-linear equations and four integrals to be solved numerically for each simulation step (cross-flow drag coefficients). The model has singular points in $\theta = \pm\pi/2$ due to the Euler formulation of kinematics, and the thrust model is singular in $u = 0$. The AUV is neutrally buoyant and is passively roll and pitch stable.

4.1 Control Objectives and Parameters

As a simulation case we will present the results from running a multiple objective controller for the AUV. The control objectives and associated actuators are described in the following. Variables with subscript r are references.

Speed Control objective function for surge speed: $v_1 = 10(u - u_r)^2$ and is assigned the actuator-set: $a_1 = \{n\}$

Heading Control objective function for heading: $v_2 = 2(\psi_r - \psi)^2$ and is assigned the actuator-set: $a_2 = \{\delta_r,\ \delta_b\}$

Depth and pitch A control objective function for depth and pitch stabilisation : $v_3 = (z - z_r)^2 + 3(q)^2$ and is assigned the actuator-set: $a_3 = \{\delta_s,\ \delta_{bs},\ \delta_{bp}\}$

The third control objective is designed to control the depth and at the same time avoid oscillations of the lightly dampened pitch axis. With the above choice of control objectives, way point tracking can be accomplished by a guidance controller that supplies new references: u_r, ψ_r, and z_r each time a way-point has been reached.

Actuator saturation limits are as specified previously and the rate constraints have been set to $\dot{\mathbf{u}}_{max} = -\dot{\mathbf{u}}_{min} = [0.2\ 0.4\ 0.2\ 0.4\ 0.4\ 1.2]$ with units of rad/s for the rudders and RPM/s for the propeller shaft. A control cost matrix has been found as: $\mathbf{P} = \text{diag}([0.2\ 0.1\ 0.2\ 0.1\ 0.1\ 0.001])$

Since the model does not have a pure integrator between the propeller shaft input and the surge speed output, it is necessary to either set the control cost for the propeller shaft actuator to zero in order to provide robust set-point tracking or include a feed-forward term as introduced in eq. (14). The latter option has been chosen here as it provides faster disturbance rejection than the free integrator approach.

4.2 Simulation Results

The simulation case shown in the following graphs is for a combined maneuver where the AUV should dive 20m, while turning 0.9 rad, and increase the surge speed from 1.1m/s to 1.5m/s.

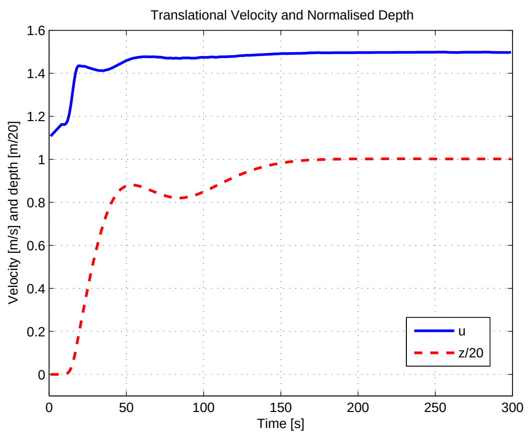


Fig. 5. Surge speed response and normalised depth response

Fig. 5 shows the results for the surge speed and normalised depth. It can be seen that both converges to their reference value. The initial dynamics of the surge-speed should be compared to the stern plane position of fig. 7; it is clear that the stern plane has a large effect on the surge speed.

The angular position during the maneuver is shown in fig. 6. It can be seen that the heading control objective is handled very well even as the AUV is speeding up and pitching at the same time. The pitch response is negative at first in order to carry out the ordered dive, where after it converges to zero. Almost no roll response is evident, confirming that passive roll stabilisation is adequate.

Finally, the control surface deflections from zero is plotted in fig. 7. The stern rudder initially makes a

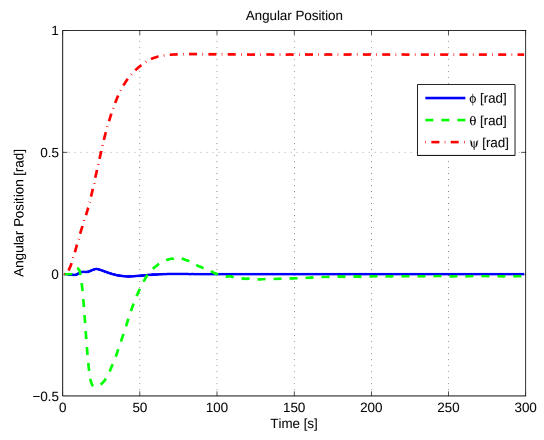


Fig. 6. Angular position throughout the maneuver

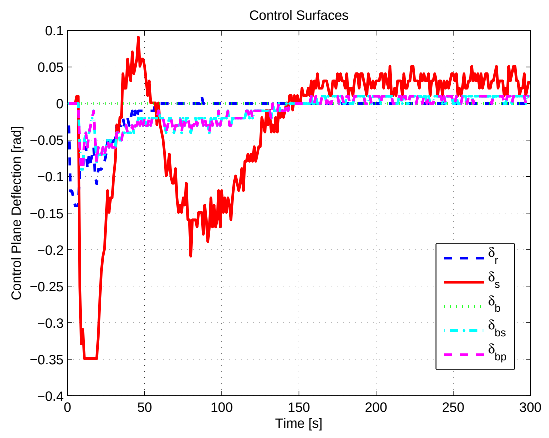


Fig. 7. Position of control surfaces throughout the maneuver

large deflection and then converges to zero, at the same time the top-bottom plane is virtually not used. This is due to the fact that by its physical position relative to the centre of mass it has very little control authority over the yaw-axis.

The dive is controlled foremost by the stern plane, but is also helped by the bow port and starboard planes. This clearly demonstrates how the control method can make use of redundant actuators in order to maximise performance.

On a contemporary lap-top (1.6GHz) the controller executes 35 times faster than real-time and profiling shows that 36% is used for QSS2 simulation and 52% for control calculations, the remaining 12% is spent in the discrete event execution framework of Alminde et al. (2006) in which the controller is implemented. The controller performs a total of 15334 control calculations corresponding to a mean update rate of 51 Hz.

5. CONCLUSIONS AND FUTURE WORK

This paper has described and demonstrated a new control approach that in many ways resemble model-predictive control, but is making use of a quantised state approach to provide local linear models. This makes minimisation of one or more control objective functions efficient and automatically selects appropriate sampling times dependent on the system dynamics.

The method has been applied on a model of an autonomous underwater vehicle and it was found that it was possible to design a well-performing controller which could make use of all available actuators, including redundant actuators.

The presented results are for open-loop control. Future work will focus on methods for injecting state corrections into the integrators as supplied by e.g. an extended Kalman filter algorithm in order to achieve closed loop control.

A rigorous comparison with classical gain scheduling methods, for particular classes of systems, will be highly relevant. Since the QSS2 method provides bounds on the trajectory errors and thus automatically regulates how often system parameters and controller gains should be re-computed, it is likely to operate well under circumstances where normal gain scheduling methods also work, possibly providing better performance and less computational cost than the classical methods.

REFERENCES

Alminde, Lars, Jan D. Bendtsen and Jakob Stoustrup (2006). A Quantized State Approach to On-line Simulation for Spacecraft Autonomy. AIAA. In 2006 Modeling and Simulation Technologies Conference Proceedings. American Institute of Aeronautics and Astronautics, Keystone, Colorado, August 2006.

Fossen, Thor I. (2002). Marine GNC Toolbox for matlab. Marine Cybernetics. available at: `www.marinecybernetics.no`.

Healey, Anthony J. and David Lienard (1993). Multivariable Sliding Mode Control for Autonomous Diving and Steering of Unmanned Underwater Vehicles. IEEE. Journal of Oceanic Engineering, Vol. 18, No. 3, July 1993.

Khalil, Hassan K. (2000). Non-linear Systems. 3 ed.. Prentice Hall.

Kofman, Ernesto (2002). A Second Order Approximation for DEVS Simulation of Continous Systems. SIAM. Journal of Simulation, issue 78, p. 76-89.

Oort, E., Q. Chu and J. Mulder (2006). Robust Model Predictive Control of a Feedback Linearized F-16/MATV Aircraft Model. AIAA. In 2006 Guidance, Navigation, and Control Proceedings. American Institute of Aeronautics and Astronautics, Keystone, Colorado, August 2006.