8th IFAC Conference on Control Applications in Marine Systems
Rostock-Warnemünde, Germany
September 15-17, 2010

Path Planning for Lock Entering Maneuvers Using Nonlinear Programming

A. Lachmeyer [*] **B. Herzer** [**] **E. D. Gilles** [***]

[*] *Institute for System Dynamics, Stuttgart, 70569 Germany (e-mail: lachmeyer@isys.uni-stuttgart.de).*
[**] *Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, 39106 Germany (e-mail: herzer@mpi-magdeburg.mpg.de)*
[***] *Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, 39106 Germany (e-mail: gilles@mpi-magdeburg.mpg.de)*

Abstract: This paper presents a path planning method for lock entering maneuvers that is based on nonlinear programming. Fairway boundaries, lock walls and the input saturation of the thrust devices of the vessel are accounted for as inequality constraints in the optimization. The environmental constraints are modeled as polygons or constructive solid geometry objects. Each of the methods is used to compute a path for a typical inland vessel with a bow thruster and a rudder and propeller configuration.

Keywords: path planning, constraint satisfaction, marine systems, model-based control, obstacle avoidance, optimization problems

1. INTRODUCTION

A lock entering maneuver requires the use of multiple actuators such as bow thrusters, the rudder and the main propellers. These actuators are subject to saturation. Furthermore, there is only a confined maneuvering space, the width of large inland vessels being only 0.4 m less than the width of a narrow lock. These circumstances must be taken into account in an automatic lock entering maneuver, at best already in the path planning phase.
There is a big variety of path planning algorithms aimed at robot motion, Karatas and Bullo (2001) propose nonlinear programming for trajectory planning of unmanned air vehicles, whereas Wang and Lane (1997) deal with subsea vehicles. To our knowledge the problem of automatic lock entering maneuvers has not been addressed in the literature so far. In this paper we propose to solve the path planning problem with a nonlinear programming approach.
Section 2 describes the dynamic model of the vessel motion, section 3 introduces the nonlinear programming problem formulation. The main difficulty is to convert the shape of the maneuvering space into inequality constraints for the nonlinear optimization software. In section 4 a polygon-based method for object representation is derived, section 5 contains a method proposed by Wang and Lane (1997) based on constructive solid geometry.

2. VESSEL MODEL EQUATIONS

The model equations describe the surge (u), sway (v) and yaw (r) motion of the vessel in a body-fixed (x_b, y_b) coordinate frame, where the forward velocity is assumed to be constant.

$$u = const. \tag{1}$$

$$\dot{v} = \frac{C_v}{m+m_a}\cdot v - \frac{m}{m+m_a}\cdot u\cdot r + \frac{F_1+F_2}{m+m_a} \tag{2}$$

$$\dot{r} = \frac{C_r}{J_r}\cdot r + \frac{L}{2\cdot J_r}\cdot(F_2 - F_1) \tag{3}$$

$$\dot{\psi} = r \tag{4}$$

$$\dot{x}_{cg} = u\cdot\cos\psi - v\cdot\sin\psi \tag{5}$$

$$\dot{y}_{cg} = u\cdot\sin\psi + v\cdot\cos\psi \tag{6}$$

$$\mathbf{x} = (v\ r\ \psi\ x_{cg}\ y_{cg})^T \tag{7}$$

m	...	mass of vessel
m_a	...	added mass
C_v	...	coefficient of lateral force due to v
C_r	...	coefficient of torque related to r
J_r	...	moment of inertia about yaw axis
L	...	length of vessel
$\mathbf{x}$	...	state vector with n states

The x-axis of the earth-fixed local coordinate frame is parallel to the lock chamber walls (see Fig. 1). The position of the vessel in the earth-fixed frame is denoted by x_{cg} and y_{cg}, its heading ψ is measured relative to the earth-fixed x-axis. The body-fixed coordinate frame originates in the center of gravity (CG).
We define two control forces F_1 and F_2, where F_1 is a bow thruster force with fixed angle (y_b direction) and a fully variable but constrained thrust force

$$0 \le F_1 \le F_{1max}. \tag{8}$$

F_2 is the y_b component of the rudder force caused by the main propeller thrust. It is also supposed to be fully variable with a maximum value

$$0 \le F_2 \le F_{2max}. \tag{9}$$

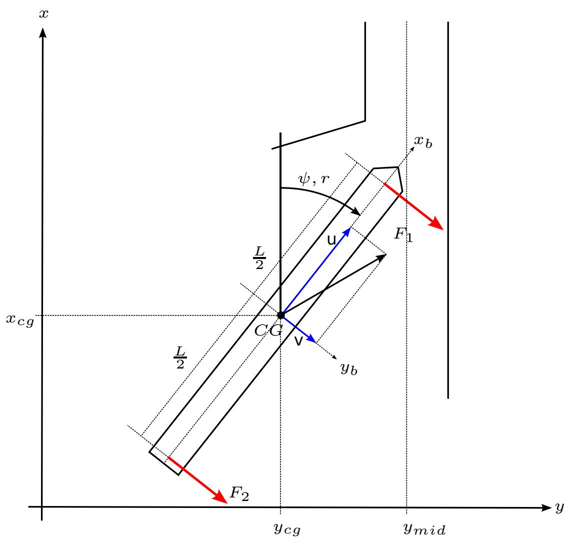

Fig. 1. Vessel dimensions and coordinate frames

The model does not take into account any effects of the water flow around the hull when entering into the lock such as squat or forward velocity changes.

3. NONLINEAR PROGRAMMING PROBLEM

In this paper we consider the optimization problem given by the objective function

$$J(\mathbf{x}, \mathbf{u}) = \int_0^{t_f} L(\mathbf{x}, \mathbf{u})\, dt \tag{10}$$

that is to be minimized, the equality constraints representing the differential equations of the vessel motion

$$\mathbf{h}(\mathbf{x}, \mathbf{u}) = 0 \tag{11}$$

and the inequality constraints

$$\mathbf{g_l} \leq \mathbf{g}(\mathbf{x}, \mathbf{u}) \leq \mathbf{g_u} \tag{12}$$

$$\mathbf{u_l} \leq \mathbf{u(t)} \leq \mathbf{u_u} \tag{13}$$

$$\mathbf{x_l} \leq \mathbf{x(t)} \leq \mathbf{x_u} \tag{14}$$

representing walls, constrained inputs and general state bounds. In order to be able to solve this problem numerically by using nonlinear programming methods, the objective function and the equality constraints are approximated using collocation as explained by Betts (2001). This leads to an Euler forward integration of the model equations. The input variables $\mathbf{u}(t)$ are assumed to remain constant during one time step Δt. This results in the following equality constraints:

$$h^{(k \cdot n)+i} = -\mathbf{x}^k + \mathbf{x}^{k-1} + \dot{\mathbf{x}}^{k-1} \Delta t \tag{15}$$

with

$$k = 1 \ldots \frac{t_f}{\Delta t} + 1. \tag{16}$$

k is the number of the sample time steps, $i = 1 \ldots n$ the number of states and t_f the fixed stop time of the maneuver.

4. POLYGON-BASED CONSTRAINTS

In this section polygons are used to represent lock walls, fairway boundaries and other obstacles. They are converted to inequality constraints that are needed in the

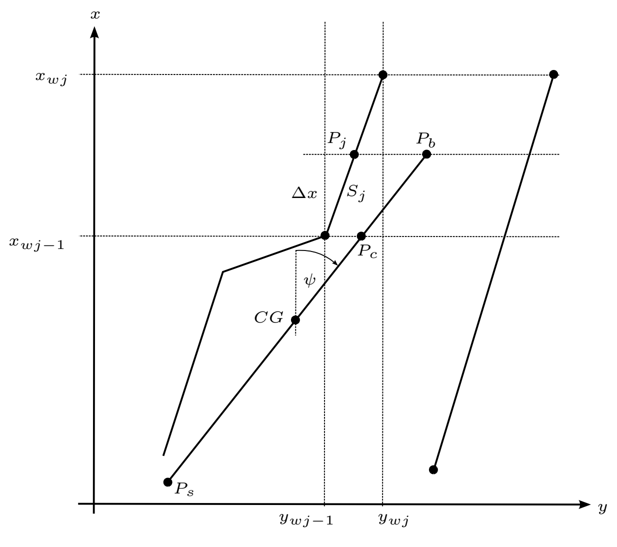

Fig. 2. Location of vessel and wall polygon

nonlinear programming problem formulation. The wall polygon is a set $(\mathbf{x}_w, \mathbf{y}_w)$ of points, containing p points on the left and q points on the right side, given in the local earth-fixed coordinate frame:

$$\mathbf{x}_w = (x_{w1} \;\ldots\; x_{wp} \; x_{wp+1} \;\ldots\; x_{wp+q})^T \tag{17}$$

$$\mathbf{y}_w = (y_{w1} \;\ldots\; y_{wp} \; y_{wp+1} \;\ldots\; y_{wp+q})^T. \tag{18}$$

The vessel is described by a line segment from P_s (stern) to P_b (bow). Since the width of the hull is neglected this way, the polygons defining the environment must be adjusted accordingly. The distance Δy between a point P_j on a wall segment $S_j = \{(x_{wj-1}, y_{wj-1}), (x_{wj}, y_{wj})\}$ and the bow as depicted in Fig. 2 is computed as follows:

$$P_{jx} = P_{bx} \tag{19}$$

$$P_{jy} = y_{wj-1} + \Delta y \tag{20}$$

$$\Delta y = \Delta x \frac{dy}{dx} = (P_{bx} - x_{wj-1}) \frac{y_{wj} - y_{wj-1}}{x_{wj} - x_{wj-1}}. \tag{21}$$

For the stern, P_j is computed as in (20).
There is no collision between bow, stern and the left wall if:

$$P_{by} - P_{jy} > 0 \wedge P_{sy} - P_{jy} > 0 \rightarrow g_{l1},\; g_{l2}. \tag{22}$$

For the right wall, the following must be true:

$$P_{by} - P_{jy} < 0 \wedge P_{sy} - P_{jy} < 0 \rightarrow g_{u1},\; g_{u2}. \tag{23}$$

Since the wall polygon may be convex, two more constraints are necessary to prevent the vessel from moving over convex corner points. The projection of such a corner point to the vessel line segment is called P_c and computed as follows:

$$P_{cx} = x_{wj-1} \tag{24}$$

$$P_{cy} = y_{cg} + \tan\psi \cdot (x_{wj} - x_{cg}). \tag{25}$$

There is no collision between a left wall corner point and the vessel if:

$$P_{cy} - y_{wj} > 0 \rightarrow g_{l5}. \tag{26}$$

For a right wall corner point the following must hold:

$$P_{cy} - y_{wj} < 0 \rightarrow g_{u6}. \tag{27}$$

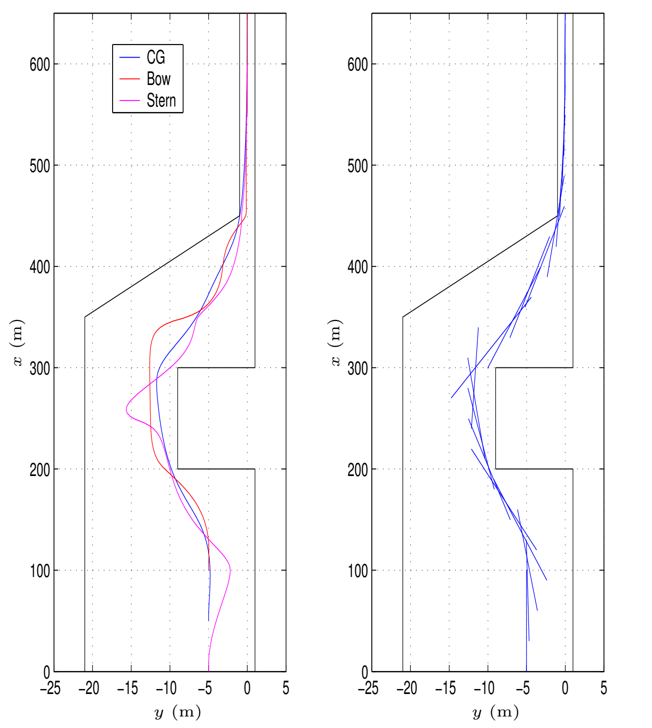

Fig. 3. Polygon-based optimization: Vessel position and orientation

This computation is done for all corner points between bow and stern. Alltogether, there are $N = 6$ constraints that represent the environment, two checking collisions between bow and the walls, two checking checking collisions between the stern and the walls and two preventing corner point contact.

The objective function for the lock entering maneuver consists of three parts that are minimized. The first part is a measure for the distance to the lock chamber middle position y_{mid}:

$$d_{mid} = y_{cg} - y_{mid}. \tag{28}$$

Additionallly, we introduce a function d_l depending on the distance between bow and stern and the left wall:

$$d_l = \frac{1}{0.1 + (P_{by} - P_{jy})^2} + \frac{1}{0.1 + (P_{sy} - P_{jy})^2}. \tag{29}$$

The corresponding function d_r for the right wall is computed in the same way. These functions are used to introduce a safety margin between the computed path and the boundaries.

The objective function J is computed as a sum over all sample time steps:

$$J = \sum_{k=1}^{h} a_1(d_{mid}^k)^2 + a_2(d_l^k + d_r^k) + a_3((F_1^k)^2 + (F_2^k)^2). \tag{30}$$

Furthermore, it minimizes the use of thrust force by including F_1 and F_2. To change the characteristics of the planned path the weight factors a_1, a_2 and a_3 can be tuned manually between optimization runs.

The optimization problem given above is solved using the *Ipopt* large scale nonlinear optimization software package. The computation of jacobian and hessian matrices is done by the *ADOL-C* automatic differentiation package. Waechter and Biegler (2006) give further information about *Ipopt*, Griewank and Walther (2008) describe the methods for automatic differentiation. *ADOL-C* offers a

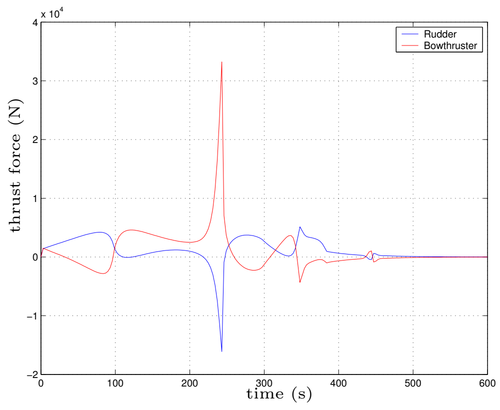

Fig. 4. Polygon-based optimization: Thrust force

C++ programming interface to *Ipopt* with the possibility to exploit the sparse structure of jacobian and hessian matrices.

Using the model parameters given in Appendix A with step size $\Delta t = 3$ s, the problem consisting of 1400 variables was solved in 9.4 s on a Core$^{\mathrm{TM}}$2 Duo CPU at 2.53 GHz. The resulting trajectory is shown in Fig. 3. The left plot contains the positions of bow, center of gravity and stern. The position of the midship line is plotted on the right side in 30 s intervals. Fig. 4 shows the thrust force of the bow thruster and the rudder with the propellers. The vessel does not touch the lock walls and the obstacle, the saturation of the thrust force is not reached. There is no contact between the planned path and the corner points. If viewed in detail, Fig. 3 shows that the corner point at $x_c = 450$ m, $y_c = -1$ m is left of the trajectory.

The disadvantage of this polygon-based implementation is that the constraints representing the environment are evaluated only at the segments closest to bow and stern. Therefore, the constraints cannot be differentiated correctly by the automatic differentiation software.

5. CONSTRUCTIVE SOLID GEOMETRY CONSTRAINTS

To overcome this disadvantage we will present a second method to define the boundaries. Wang and Lane (1997) proposed to model the environment by combining several simple geometric objects (primitives) using the boolean operations *AND* and *OR*. The so called constructive solid geometry (CSG) is very common in computer graphics and can be used to build complex environments.

The primitive used to represent the lock walls is a deformed circle defined by its radius r_c, its center point $(x_c,\ y_c)$ and its deformation coefficient b:

$$(x_c^{2\cdot b} + y_c^{2\cdot b})^{1/b} = r_c^2. \tag{31}$$

The following equation can be used to test if P_b or P_s are outside the wall objects:

$$((x_c - P_{bx})^{2\cdot b} + (y_c - P_{by})^{2\cdot b})^{1/b} > r_c^2. \tag{32}$$

In the given example, this is done for $n_o = 6$ wall objects for bow and stern at every sample time step t^k. As in section 4, relying only on constraints of the type given in (32) results in a path moving over the corners of the walls. To prevent that, a second type of constraints is introduced that guarantees that no corner point (x_{wj}, y_{wj}) is inside an ellipse defining the area covered by the ship. This is

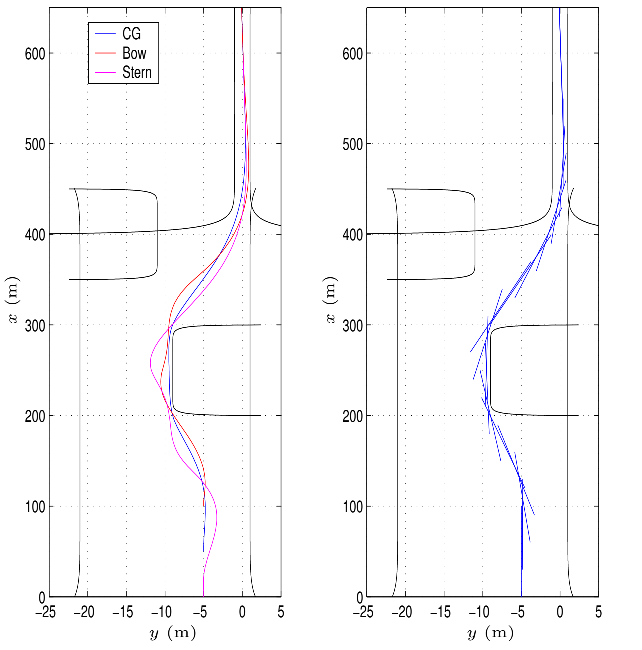

Fig. 5. CSG-based optimization: Vessel position and orientation

the case if the corner point transformed to the body-fixed coordinate frame

$$x^b_{wj} = (x_{wj} - x_{cg}) \cdot \cos\psi + (y_{wj} - y_{cg}) \cdot \sin\psi) \quad (33)$$
$$y^b_{wj} = (-x_{wj} + x_{cg}) \cdot \sin\psi + (y_{wj} - y_{cg}) \cdot \cos\psi) \quad (34)$$

meets the following condition:

$$\frac{(x^b_{wj})^2}{L} + \frac{(y^b_{wj})^2}{L} > \left(\frac{B}{2}\right)^2. \quad (35)$$

B is the width of the hull of the vessel.
For the wall setting shown in Fig. 3 we need to introduce $n_c = 3$ corner points as additional constraints, one on the left wall and two on the right wall. The total number of wall constraints per sample time step is

$$N = n_c + 2 \cdot n_o = 15. \quad (36)$$

This number increases with the number of environment objects, in contrast to the method presented in section 4. With

$$J = \sum_{k=1}^{h} a_1 (d^k_{mid})^2 + a_3((F^k_1)^2 + (F^k_2)^2) \quad (37)$$

as objective function, the nonlinear program is solved in 13.2 s given the same set of parameters that were used in section 4. Fig. 5 and 6 show the computed path and thruster actuation. Due to the rounded corners of the objects, the path of bow and stern runs through areas that were forbidden in section 4. To prevent this, the object size must be increased.

6. CONCLUSION

We presented two methods for the planning of lock entering maneuvers based on nonlinear programming. The numerical solution of the optimization problem is computed by the software packages *Ipopt* and *ADOL-C*. The two methods differ in the way how inequality constraints

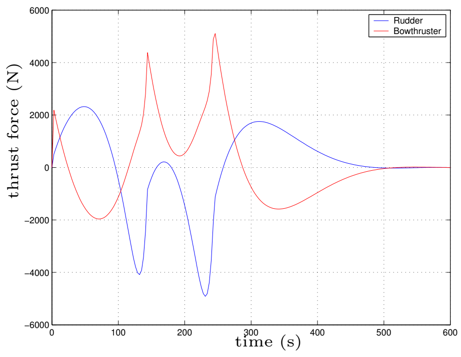

Fig. 6. CSG-based optimization: Thrust force

are derived from lock walls and fairway boundaries. The advantage of the polygon-based approach is the constant number of inequality constraints regardless of the complexity of the polygons. However, the position-dependent selection of the active parts of the polygon leads to difficulties in the differentiation of the constraints. Nonetheless, *Ipopt* was able to find an optimal solution. A method that relies on constraints that can be differentiated correctly was presented in section 5. In this case, the number of inequality constraints increases with two times the number of environment objects. Closed-loop simulations with real world lock environments and more detailed actuator models will be carried out to test both methods for their usability on a research vessel.

REFERENCES

Betts, J.T. (2001). *Practical methods for optimal control using nonlinear programming.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Griewank, A. and Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* Number 105 in Other Titles in Applied Mathematics. SIAM, Philadelphia, PA, 2nd edition.

Karatas, T. and Bullo, F. (2001). Randomized searches and nonlinear programming in trajectory planning. In *Proceedings of the 40th IEEE Conference on Decision and Control, 2001*, volume 5, 5032 –5037.

Waechter, A. and Biegler, L.T. (2006). On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106, 25–57.

Wang, Y. and Lane, D. (1997). Subsea vehicle path planning using nonlinear programming and constructive solid geometry. *IEE Proceedings Control Theory and Applications*, 144(2), 143 –152.

Appendix A. MODEL PARAMETERS AND INITIAL VALUES

$L = 100.0\,\text{m}$	$B = 10.0\,\text{m}$
$C_q = 3500$	$Cv = -350000$
$m = 2 \times 10^6\,\text{kg}$	$m_a = 5 \times 10^5\,\text{kg}$
$C_r = -2.0417 \times 10^{+08}$	$J_r = 1.6667 \times 10^{+09}$
$F_{1max} = F_{2max} = 8.1667 \times 10^{+04}$	
$a_1 = 0.0022$	$a_2 = 1.0$
$a_3 = 1.4994 \times 10^{-09}$	
$u_0 = 1\,\text{m/s}$	$v_0 = 0\,\text{m/s}$
$r_0 = 0\,1/\text{s}$	$\psi = 0$
$x_{cg0} = 50\,\text{m}$	$y_{cg0} = -5\,\text{m}$