MODELING, IDENTIFICATION AND CONTROL, 1992, VOL. 13, NO. 3, 163–174

Computing energy-optimal trajectories for an autonomous underwater vehicle using direct shooting

INGE SPANGELO† AND OLAV EGELAND†

Keywords: *ROV, optimal control, nonlinear programming, direct shooting*

Energy-optimal trajectories for an autonomous underwater vehicle can be computed using a numerical solution of the optimal control problem. The vehicle is modeled with the six dimensional nonlinear and coupled equations of motion, controlled with DC-motors in all degrees of freedom. The actuators are modeled and controlled with velocity loops. The dissipated energy is expressed in terms of the control variables as a nonquadratic function. Direct shooting methods, including control vector parameterization (CVP) are used in this study. Numerical calculations are performed and good results are achieved.

1. Introduction

The use of energy-optimal trajectories can extend the period of operation for an autonomous battery powered underwater vehicle significantly by reducing the energy consumption by about 50%. This type of vehicle can be used for underwater inspection and telerobotics. In certain applications like operation in a cluttered environment it may be advantageous that the vehicle is untethered, and batteries have to be used for small vehicles.

Optical control problems can be solved indirectly using the necessary conditions of optimality (Athans and Falb 1966, Bryson and Ho 1975) and solving the resulting two point boundary value problem (Keller 1968).

Direct shooting methods (Kraft 1989) belong to the class of vector parameterization methods (CVP). Using CVP methods the infinite dimensional optimal control problem is transformed into a finite dimensional nonlinear program. Other CVP methods are CVP with multiple shooting (Bock and Plitt 1985) and direct collocation (Hargraves and Paris 1987). Direct shooting is also called CVP with single shooting.

The use of optimal control theory to generate energy-optimal trajectories was proposed in Spangelo and Egeland (1992). Optimal trajectories were computed numerically using two different methods, a conjugate gradient method in function space (Lasdon, Mitter and Warren 1967), and a direct shooting method with finite difference gradient calculations (Kraft 1989).

In this work, direct shooting has been applied with both numerical (finite difference) and analytical gradient calculations. Computing analytical gradients calculus of variation and the resulting costate equations are used (Goh and Teo 1988, Kraft 1985). The methods were applied to a model of the NEROV vehicle (Fossen and Sagatun 1991) developed at the Norwegian Institute of Technology. Computational results are presented in the paper.

Received 6 April 1992.

† Division of Engineering Cybernetics, The Norwegian Institute of Technology, N-7034 Trondheim, Norway.

2. Energy-optimal trajectories for the NEROV vehicle

2.1. *Vehicle model*

The equations of motion for an underwater vehicle are written (Fossen and Sagatun 1991, Fossen 1991):

$$\boldsymbol{M}\dot{\boldsymbol{v}}+\boldsymbol{C}(\boldsymbol{v})\boldsymbol{v}+\boldsymbol{D}(\boldsymbol{v})\boldsymbol{v}+\boldsymbol{\gamma}(\boldsymbol{\xi})=\tilde{\boldsymbol{B}}(\boldsymbol{v})\boldsymbol{u} \tag{1}$$

Here $\boldsymbol{M}$ is the 6×6 inertia matrix containing vehicle inertia and hydrodynamic added inertia. The six dimensional vector

$$\boldsymbol{v}=(u\ v\ w\ p\ q\ r)^{\mathrm{T}} \tag{2}$$

is the velocity in vehicle coordinates where u, v, and w are the linear velocities in the x, y, and z directions of the vehicle, and p, q and r are the angular velocities about the vehicle x, y, and z axes of the vehicle. $\boldsymbol{C}$ is a 6×6 matrix so that $\boldsymbol{Cv}$ is the vector of Coriolis, centripetal and hydrodynamic terms according to the theory of ideal fluids. $\boldsymbol{D}$ is a 6×6 matrix containing coefficients describing dissipative hydrodynamic terms. $\boldsymbol{\gamma}(\boldsymbol{\xi})$ is the 6 dimensional vector of restoring forces and moments caused by gravity and buoyancy. $\tilde{\boldsymbol{B}}$ is the 6×6 input matrix.

The vector

$$\boldsymbol{\xi}=(x\ y\ z\ \phi\ \theta\ \psi)^{\mathrm{T}} \tag{3}$$

denotes the global position where x, y and z are the position coordinates and ϕ, θ and ψ are the roll, pitch and yaw angles. Accordingly

$$\dot{\boldsymbol{\xi}}=\boldsymbol{J}(\boldsymbol{\xi})\boldsymbol{v} \tag{4}$$

where the Jacobian $\boldsymbol{J}(\boldsymbol{\xi})$ is given by

$$\boldsymbol{J}=\begin{pmatrix}\boldsymbol{R} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{E}\end{pmatrix} \tag{5}$$

$\boldsymbol{R}$ is the 3×3 orthogonal rotation matrix (Spong and Vidyasagar 1989)

$$\boldsymbol{R}=\boldsymbol{R}_{z,\psi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{x,\phi}=\begin{pmatrix} c\psi c\theta & -s\psi c\phi+c\psi s\theta s\phi & s\psi s\phi+c\psi s\theta c\phi\\ s\psi c\theta & c\psi c\phi+s\psi s\theta s\phi & -c\psi s\phi+s\theta s\psi c\phi\\ -s\theta & c\theta s\phi & c\theta c\phi\end{pmatrix} \tag{6}$$

and $\boldsymbol{E}$ is a 3×3 transformation matrix. $c(\cdot)=\cos(\cdot)$ and $s(\cdot)=\sin(\cdot)$.

$$\boldsymbol{E}=\begin{pmatrix}1 & s\phi s\theta/c\theta & c\phi s\theta/c\theta\\ 0 & c\phi & -s\phi\\ 0 & s\phi/c\theta & c\phi/c\theta\end{pmatrix} \tag{7}$$

which in general is non-orthogonal. It is assumed that the pitch angle θ is bounded by $|\theta|<\frac{\pi}{2}$ so that $\boldsymbol{E}$ is bounded.

The system can be described in state space form by defining the 12 dimensional state vector

$$\boldsymbol{x}=\begin{pmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{pmatrix} \tag{8}$$

where $\boldsymbol{x}_1=\boldsymbol{\xi}$ and $\boldsymbol{x}_2=\boldsymbol{v}$. This results in the model

$$\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{B}(\boldsymbol{x})\boldsymbol{u} \tag{9}$$

where

$$f(x)=\begin{pmatrix} J(x_1)x_2 \\ M^{-1}[-C(x_2)x_2-D(x_2)x_2-g(x_1)] \end{pmatrix} \tag{10}$$

and

$$B(x)=\begin{pmatrix} 0 \\ M^{-1}\tilde{B}(x_2) \end{pmatrix} \tag{11}$$

2.2. *Actuator models*

The NEROV vehicle has six thrusters driven by DC motors. The DC motors are equipped with velocity loops using feedback from tachometers, which simplifies the development of the vehicle control system.

The dynamic model of the DC motor is:

$$L_a\frac{d}{dt}i_a=-R_a i_a-2\pi K_M n+u_a \tag{12}$$

$$2\pi J_m\frac{d}{dt}n=K_M i_a-Q \tag{13}$$

where L_a is the armature inductance, R_a is the armature resistance, u_a is the armature voltage, K_M is the motor torque constant, J_m is the moment of inertia of the motor and thruster and n is the velocity of the motor in revolutions per second.

A PI velocity loop gives accurate control of the propeller velocity n.

From basic propeller theory it is known that the load torque from the propeller is given by (Newman 1977)

$$Q=\rho D^5 K_Q(n,v)n|n| \tag{14}$$

where ρ is the mass density of water, D is the diameter of the propeller, and K_Q is the torque coefficient of the propeller.

The thrust force is given by

$$T=\rho D^4 K_T(n,v)n|n| \tag{15}$$

where $K_T(n,v)$ is the thrust coefficient. For simplicity we assume that the thrust coefficients are constant, $\bar{K}_T$.

The control variables are defined by

$$u_i=\bar{T}_i=\rho D^4\bar{K}_T n_i|n_i| \tag{16}$$

where T_i is the thrust force of thruster number i.

2.3. *Control energy*

The dynamics of the armature circuit are assumed to be fast, and in the following eqn. (12) is approximated by

$$u_a=R_a i_a+2\pi K_M n \tag{17}$$

Dissipated electrical power is

$$P=u_a i_a=R_a i_a^2+2\pi K_M n i_a \tag{18}$$

The first term on the right-hand side is due to dissipation in the armature resistance, and the second term denotes the conversion from electrical to mechanical power. The first term is assumed to be much smaller than the second and is therefore ignored. Inserting

$$i_a = \frac{2\pi J_m}{K_M}\dot{n} + \frac{Q}{K_M} \tag{19}$$

into the second term of eqn. (17) gives

$$P = (2\pi)^2 J_m n\dot{n} + 2\pi Q n \tag{20}$$

The first term on the right-hand side is power needed to accelerate the propeller, and the last term is power dissipation due to the interaction between propeller and water. Accordingly the following approximation of the mechanical power dissipation is used for motor i:

$$P_i = 2\pi Q n_i = 2\pi\rho D^5 K_Q n_i^2 |n_i| \tag{21}$$

In terms of the control variables this is written

$$P_i = \frac{2\pi D K_Q}{K_T}|u_i|^{1.5} \tag{22}$$

2.4. *Formulation of the optimal control problem (OCP)*

The optimal control problem investigated in this paper can be formulated as follows: Find the trajectory, that is the state and control time histories, which minimize the performance index

$$\min_{\boldsymbol{u}} J_0 = \int_{t_0}^{t_f} L(\boldsymbol{u})\,dt = \int_{t_0}^{t_f} \sum_{i=1}^{6} |u_i(t)|^{1.5} dt \tag{23}$$

for the system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u} \tag{24}$$

where the initial and final states are given:

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{25}$$

$$\boldsymbol{x}(t_f) = \boldsymbol{x}_f \tag{26}$$

The time consumption is not critical so a fixed final time is used.

3. Numerical solution of the OCP by direct shooting

The solution of the OCP (eqns. 23–26) gives an open loop trajectory, which has to be found offline. Generally a numerical method has to be applied for such problems. This section describes how the problem can be solved by direct shooting, or CVP with single shooting.

By parameterizing the control vector $\boldsymbol{u}(t)$ the infinite dimensional optimization problem (23–26) can be approximated by a finite dimensional nonlinear program (NLP) (see e.g. Kraft 1985):

$$\min F(P_u) = \min_{Pu} \int_{t_0}^{t_f} \sum_{i=1}^{6} |u_i(t, P_u)|^{1.5} dt \tag{27}$$

with constraints

$$c(P_u)=x(t_f)-x_f=0 \tag{28}$$

where P_u is a finite set of parameters uniquely defining the controls $u_i(t)$. Observe that the constraint equations (28) are not linear in the parameters, and that the state space equations (24–25) have to be solved each time $c(P_u)$ is to be found.

The NLP (27–28) can be solved by standard optimization techniques (Gill, Murray and Wright 1981, Luenberger 1984), giving a suboptimal solution of the optimal control problem (eqn. 23–26).

Solving an NLP effectively, the gradients and the values of the performance index and the constraints have to be calculated. The gradients can be calculated numerically (Kraft 1989, Horn 1990), by finite differences, or analytically by using the adjoint equations (Kraft 1980, Goh and Teo 1988). The two approaches are both applied to the NEROV vehicle.

3.1. *Parameterization of the control variables*

First the time interval $[t_0, t_f]$ is partitioned into q subintervals

$$[t_j, t_{j+1}], \quad j=0 \ldots q-1.$$

The breakpoints t_j do not necessarily have to be equidistantly distributed, and can therefore be included in the parameter set P_u. In the sequel it is assumed that the breakpoints are equidistantly distributed, and equal for each control.

In each subinterval $[t_j, t_{j+1}]$ the controls $u_i(t)$ are approximated by interpolating polynomials $v_{ij}(t)$. The control approximation functions v_i are defined as

$$v_i(t, P_u)=v_{ij}(t), \quad t\in[t_j, t_{j+1}] \tag{29}$$

and

$$\boldsymbol{v}(t, P_u)=(v_1(t, P_u) \ldots v(t, P_u))^T \tag{30}$$

Eqns. (24–25) are then approximated by

$$\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{B}(\boldsymbol{x})\boldsymbol{v}(t, P_u) \tag{31}$$

$$\boldsymbol{x}(t_0)=\boldsymbol{x}_0 \tag{32}$$

In this work $\boldsymbol{u}(t)$ is approximated by piecewise linear polynomials.

$$v_{ij}(t)=u_{ij}+\frac{u_{ij+1}-u_{ij}}{t_{j+1}-t_j}(t-t_j) \tag{33}$$

where $u_{ij}=u_i(t_j)$. Alternative interpolation schemes for the control parameterization are described (Kraft 1989).

A set of $6(q+1)$ parameters are now given

$$P_u=(u_{10}, u_{11}, \ldots, u_{1q}, \ldots\ldots\ldots, u_{6q}) \tag{34}$$

The resulting finite-dimensional optimization problem can be arbitrarily close to the optimal control problem (eqn. 23–26) by choosing a large size for the parameter set. The choice of a sensible size is discussed in the computational study in the next section.

3.2. *Numerical gradient calculation*

The gradients can be calculated numerically by forward differences

$$\frac{\partial c_k(P_u)}{\partial u_{ij}} = \frac{c_k(u_{10},\ldots,u_{ij}+hp_{ij},\ldots,u_{6q}) - c_k(P_u)}{hp_{ij}} \tag{35}$$

where

$$(c_0 \ldots\ldots c_m)^T = (F\, c^T)^T \tag{36}$$

m is the number of constraints. Consequently the state space equations have to be solved $6(q+1)+1$ times for each iteration to find the gradients in the NLP (27–28). The hp_{ij}'s (the perturbation sizes) were calculated as proposed by Kraft (1989), based on the analysis in Horn (1990)

$$hp_{ij} = \max\left\{\frac{\partial}{10}, |u_{ij}|\frac{\partial}{10}, \varepsilon^{1/2}\right\} \tag{37}$$

where δ is the global accuracy of the initial value problem solver (NODE), and approximately equals the optimization convergence criteria. ε is the machine precision. It was shown in Horn (1990) that almost the same accuracy as in the NODE-solver can be achieved in the gradient calculation, provided that approximately the same timesteps are used for the unperturbed and the perturbed simulations. The 4th order Runge–Kutta method RK4 with constant timesteps was chosen as NODE-solver, in which case this timestep requirement is automatically satisfied. A general discussion on using finite-difference approximation to gradients in optimization is given in Gill, Murray and Wright (1981). The 4th order Simpson's rule was used in integrating the performance index.

3.3. *Analytical gradient calculations*

By using the adjoint equations the gradients can be calculated without using the forward-difference scheme.

Observe first that the performance index (eqn. 27) does not depend on the states, and consequently not on the state space equations. It is therefore easily seen that

$$\delta F(P_u) = \int_{t_0}^{t_f} \frac{\partial L(\boldsymbol{u})}{\partial \boldsymbol{u}} \delta \boldsymbol{u}\, dt \tag{38}$$

where $\delta F(P_u)$ and $\delta \boldsymbol{u}$ is the first variation of $F(P_u)$ and $\boldsymbol{u}$ respectively w.r.t. the parameters. $L(\boldsymbol{u})$ is defined in eqn. (23). This gives for the partial derivatives

$$\frac{\partial F(P_u)}{\partial u_{ij}} = \int_{t_0}^{t_f} \frac{\partial L(\boldsymbol{u})}{\partial u_i} \frac{\partial v_i}{\partial u_{ij}} dt \tag{39}$$

where v_i is defined in eqns. 29 and 33, and u_i denotes control number i. From eqn. (23)

$$\frac{\partial L(\boldsymbol{u})}{\partial u_i} = 1{\cdot}5\, \text{sign}\,(u_i)(|u_i|)^{1/2} \tag{40}$$

The equality constraints (eqn. 28) are given by the end time constraints in eqn. (26). Consequently the state space equations have to be included in the derivation of the derivatives of $\boldsymbol{c}(P_u)$ w.r.t. the parameters. This was done in the following way (Further discussions and proofs can be found in (Kraft 1980) or (Goh and Teo 1988)).

Consider the individual constraints $c_k(P_u)$ as Mayer types of object functionals:

$$J_{c_k} = \phi_k(x(t_f)) \tag{41}$$

Then proceed in the usual way to find the first variation of J_{c_k} w.r.t. the control vector (Athans and Falb 1966, Bryson and Ho 1975). Define a Hamilton function

$$H_{c_k}(u) = \lambda_{c_k}^T f(x, u) \tag{42}$$

for each of the constraints c_k. The first variation of J_{c_k} is given by

$$\delta J_{c_k} = \int_{t_0}^{t_f} \lambda_{c_k}^T \frac{\partial f(x, u)}{\partial u} \delta u \, dt = \int_{t_0}^{t_f} \lambda_{c_k}^T B(x) \delta u \, dt \tag{43}$$

where

$$\dot{\lambda}_{c_k} = -\frac{\partial f(x, u)^T}{\partial x} \lambda_{c_k} \tag{44}$$

and

$$\lambda_{c_k}(t_f) = \frac{\partial c_k}{\partial x}(t_f) \tag{45}$$

The gradients of the constraints are now given by eqn. (43).

$$\frac{\partial c_k}{\partial u_{ij}} = \int_{t_0}^{t_f} (\lambda_{c_k}^T B(x))_i \frac{\partial v_i}{\partial u_{ij}} \, dt \tag{46}$$

Consequently, each time the gradients of the constraints are to be calculated one has to solve m sets of adjoint equations, where m is the number of constraints.

For this optimal control problem the adjoint equations had to be solved 12 times, each with different boundary conditions. Alternatively the end time constraints (eqn. 26) can be reformulated as a quadrature (Goh and Teo 1988):

$$c_c = (x(t_f) - x_{t_f})^T (x(t_f) - x_{t_f}) \tag{47}$$

The number of constraints are then reduced to one, at the cost of increasing the nonlinearity. These two different ways of handling the constraints were compared, and the results are given in the next section.

4. Computational study

A numerical study was done for a coupled surge and sway motion on the NEROV vehicle. The tests were made on a SUN 4/75 station. The computational effort in terms of CPU seconds and the value of the converged performance index were compared for the two approaches of gradient calculations, and for different numbers of parameters. In the case of analytical gradient calculations the constraints were handled both as plain end-time constraints and as an error quadrature (eqn. 47).

A sequential quadratic programming subroutine (Kraft 1989) was used to solve the nonlinear program (27–28). A convergence criteria *acc* of 0·01 was chosen, which meant that the L1-norm of the constraint violations had to be less than 0·01. The RK4 NODE-solver and Simpson's rule of integration are both of 4th order, which means that they have a global accuracy proportional to h^4 where h is the step length of integration. The exact value of the global accuracy is difficult to find, but after some tuning effort using numerical gradient calculations a step length $h = 0{\cdot}25$ was chosen, which appeared to

keep the global accuracy safely below *acc*. Based on the discussion in the last section, and after some tuning effort the forward-difference perturbation in the numerical gradient calculation was chosen as (see eqn. 37)

$$hp_{ij} = \max\left\{h^5, |u_{ij}|\frac{h^5}{10}\right\} \tag{48}$$

4.1. *Vehicle model parameters*

The hydrodynamical parameters for the NEROV vehicle were found in Fossen (1991).

The inertia matrix was assumed to be diagonal:

$$\boldsymbol{M} = \text{diag}\{m - X_{\dot{u}}, m - Y_{\dot{v}}, m - Z_{\dot{w}}, I_x - K_{\dot{p}}, I_y - M_{\dot{q}}, I_z - N_{\dot{r}}\} \tag{49}$$

The mass is $m = 185\,\text{kg}$. The moments of inertia around the x, y and z axes are $I_x = 25\,\text{kgm}^2$, $I_y = 29\,\text{kgm}^2$ and $I_z = 28\,\text{kgm}^2$. The hydrodynamic added inertias are $X_{\dot{u}} = -30\,\text{kg}$, $Y_{\dot{v}} = -110\,\text{kg}$, $K_{\dot{p}} = -15\,\text{kgm}^2$, $M_{\dot{q}} = -20\,\text{kgm}^2$ and $N_{\dot{r}} = -1\,\text{kgm}^2$.

The matrix $\boldsymbol{C}$ was set to zero.

The drag matrix $\boldsymbol{D}$ was also assumed to be diagonal with diagonal

$$\begin{aligned}\boldsymbol{D}(\boldsymbol{v}) = \text{diag}\{&80 + 120|u|, 110 + 200|v|, 100 + 150|w|,\\ &30 + 50|p|, 40 + 40|q|, 10 + 15|r|\}\end{aligned} \tag{50}$$

The restoring moments were found to be

$$\boldsymbol{g} = (0\ 0\ 0\ z_B B c\theta s\phi\ \ z_B B s\theta\ \ 0)^{\mathrm{T}} \tag{51}$$

where $z_B = 0{\cdot}04\,\text{m}$ is the z-coordinate of centre of buoyancy and $B = 1800\,\text{N}$ is the buoyancy, which is equal to the weight mg of the vehicle.

The thruster configuration of NEROV results in the control matrix

$$\boldsymbol{B} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & l_3 & -l_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & -l_5 & l_6 \\ l_1 & l_2 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{52}$$

where it is assumed that all thrusters have the same K_T value, and that it is constant. l_i $(i = 1, 2 \ldots 6)$ are the thruster offsets giving a moment arm. Numerical values are: $l_1 = l_2 = 0{\cdot}4\,m$, $l_3 = 0{\cdot}21\,m$, $l_4 = 0{\cdot}02\,m$ and $l_5 = l_6 = 0{\cdot}43\,m$.

4.2. *Coupled surge and sway motion*

A motion increment of 5 m in 10 s in the initial surge and sway direction was studied. The initial state was

$$\boldsymbol{x}(0) = \boldsymbol{0} \tag{53}$$

and the final state was

$$\boldsymbol{x}(10) = (5\ 5\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)^{\mathrm{T}} \tag{54}$$

	# par	J^*	# iter	# fc	tot. CPU	sqp CPU	gc CPU
NG	11	$12{\cdot}58 \times 10^3$	18	27	65	1·6	2·1
NG	9	$12{\cdot}69 \times 10^3$	13	20	35	0·9	1·8
NG	6	$12{\cdot}87 \times 10^3$	8	9	11·7	0·3	1·2
NG	5	$13{\cdot}00 \times 10^3$	8	8	9·5	0·2	1·1
NG	3	$14{\cdot}40 \times 10^3$	4	4	4·3	0·07	0·7
NG	2	$26{\cdot}12 \times 10^3$	4	4	1·9	0·03	0·5
AG	11	$12{\cdot}58 \times 10^3$	18	27	42	1·6	0·7
AG	9	$12{\cdot}60 \times 10^3$	17	23	29	0·9	0·7
AG	6	$12{\cdot}87 \times 10^3$	8	9	8·1	0·3	0·7
AG	5	$12{\cdot}96 \times 10^3$	10	15	8·7	0·2	0·7
AG	3	$14{\cdot}39 \times 10^3$	6	6	4·5	0·07	0·7
AG	2	$26{\cdot}12 \times 10^3$	4	4	2·6	0·03	0·7
QAG	11	$12{\cdot}23 \times 10^3$	21	39	29	1·2	0·07
QAG	9	$12{\cdot}14 \times 10^3$	21	39	18	0·7	0·07
QAG	6	$12{\cdot}18 \times 10^3$	25	49	10·4	0·25	0·07
QAG	5	$12{\cdot}31 \times 10^3$	24	48	7·8	0·17	0·07
QAG	3	$13{\cdot}34 \times 10^3$	27	55	5·8	0·04	0·07
QAG	2	$16{\cdot}59 \times 10^3$	29	57	5·3	0·02	0·07

Table 1.

Table 1 shows computational results using numerical gradient calculation (NG), analytical gradient calculations and plain constraints (AG), and analytical gradient calculations and error quadrature (QAG). The computations were done with different numbers of parameters. The converged performance indexes are shown in the third column. The differences between plain constraints and error quadrature for the same number of parameters can partly be explained by the fact that the boundary conditions are treated differently, and should therefore not be given too much importance. Columns four and five show the numbers of iterations and function calculations, respectively. A function calculation consists of computing the performance index and the constraint violations. The computing efforts in terms of CPU seconds are shown in the three last columns of Table 1. The total CPU-time is shown in the fifth column. The last two columns show the CPU-time spent each time a new search direction was calculated in the SQP, and a gradient calculation was done, respectively. A gradient calculation consists of computing the performance index gradient and the constraint Jacobian matrix. The CPU time spent when a function calculation was done was ca. 0·05 seconds in all cases, and is not shown in Table 1.

Theoretically the results (converged performance index and the numbers of iterations and function calculations) should have been the same for AG and NG, but there were some differences, probably due to numerical reasons. In the case of 5 parameters the calculations with AG were performed with half step length ($h = 0{\cdot}125$) in the integrators, which gave results that were much closer to NG (the same numbers of iterations and functions calculations). The NG method therefore seems to be more accurate than AG.

In all tests all the parameters were initially chosen to be zero.

The choice of an optimal number of parameters is a compromise between total CPU-time effort and performance index. It is seen from Table 1 that in the case of numerical gradient calculations the converged performance index with five parameters

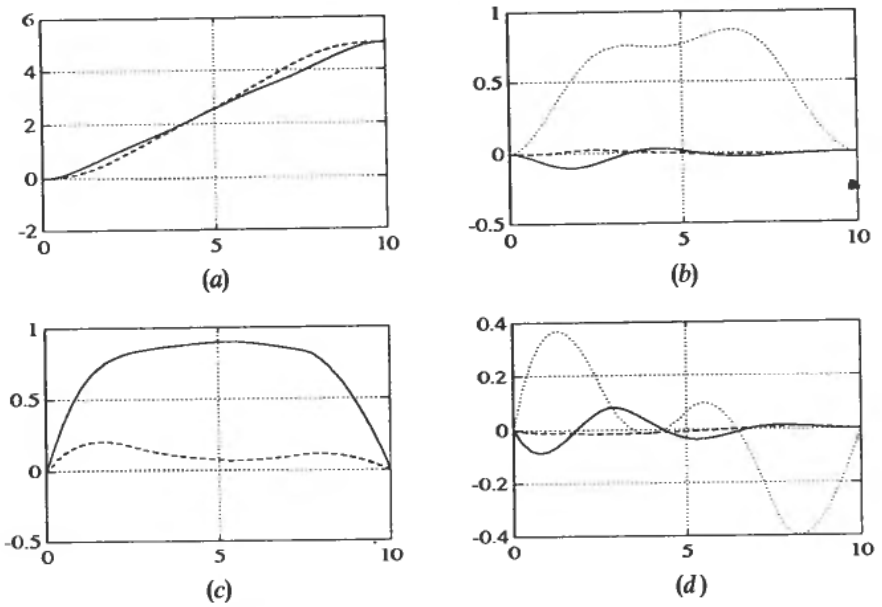

Figure 1. Computed optimal trajectories.

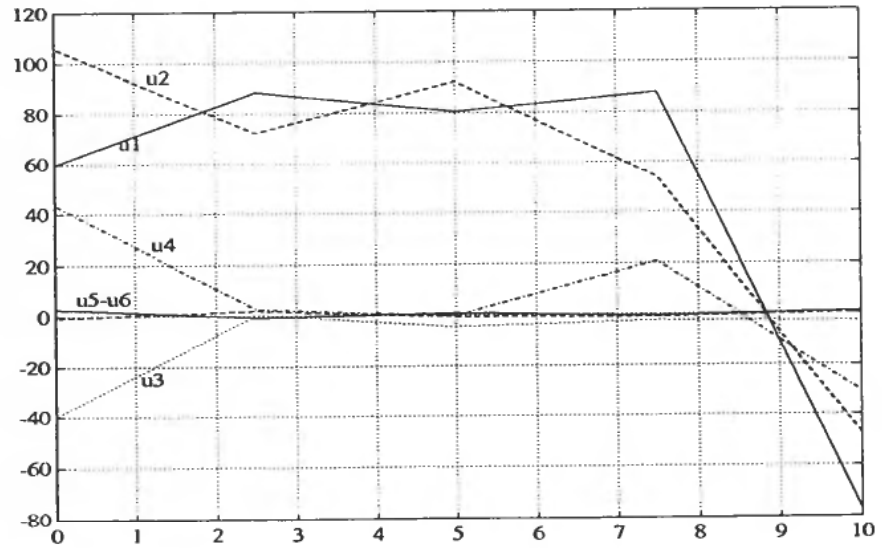

Figure 2. Computed optimal thrusts.

is only ca. 3% higher than with eleven parameters, whereas the total CPU-time effort is only ca. 15%. More than 5 or 6 parameters seem to be unnecessary.

It is seen from Table 1 that when the number of parameters becomes large, QAG is considerably faster than AG which on the other hand is considerably faster than NG. When the number of parameters is moderate (5 or 6) the different approaches do not differ much in total CPU-time consumption. The NG approach uses a bit more CPU-time, but on the other hand it is more accurate with the chosen integration step time.

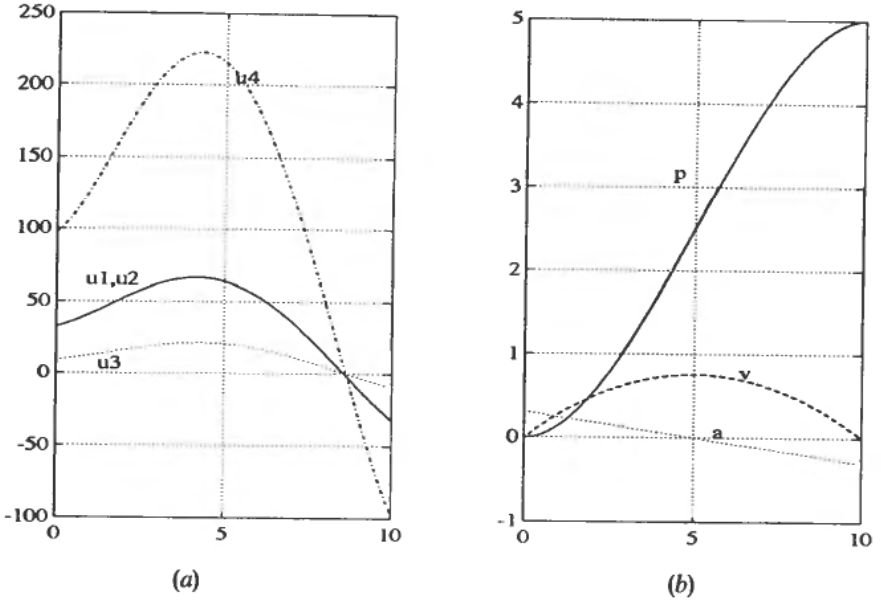

Figure 3. Computed heuristic thrusts and trajectories.

Another very important argument in favour of the numerical approach is the higher flexibility because one does not need the adjoint equations.

With simple boundary conditions as constraints only a couple of iterations are needed before the performance index is higher than the converged, while with the quadrature it increases slowly until convergence. This is because in the former case the SQP quickly finds the manifold of feasible solutions, while in the latter case this is difficult. This results in a higher number of iterations when the error quadrature representation of the end-time constraints is used.

Optimal trajectories and optimal controls computed with the NG-method and with five parameters are shown in Figures 1–2.

Finally a heuristic trajectory was computed. A solution with quadratic velocity profiles in the x and y directions was used. The resulting controls are shown in Fig. 3 (a). The acceleration (a), velocity (v) and the trajectory (p) are shown in Fig. 3 (b). The resulting performance index was $J_h = 24{\cdot}88 \times 10^3$, which is almost twice the optimal solutions that were computed.

5. Conclusions

A significant reduction in the energy consumption (ca. 50%) can be achieved by using optimal control for an autonomous underwater vehicle. Energy minimization for such vehicles implies the solution of an optimal control problem with nonlinear state space equations and nonquadratic performance index.

A control vector parameterization method with single shooting has been tested. Two different numerical approaches for computing gradients were tried, numerical differentiation with forward-differences and an analytic one including the adjoint equations of the system. In the latter case the end-time constraints were treated in two

different ways, as plain constraints and as an error quadrature. A piecewise linear parameterization of the control was used and different numbers of parameters were tried. With a moderate number of parameters none of the approaches seemed to be superior. The numerical gradient calculation scheme was a bit slower, but more accurate than the two analytical gradient calculation schemes.

In the future we plan to test other numerical methods, primarily trying to reduce the CPU-time effort, and apply it in physical experiments. We will also solve more complicated problems like collision avoidance.

Acknowledgments

This work was supported by the Royal Norwegian Council for Scientific and Industrial Research through the MOBATEL Programme at the Norwegian Institute of Technology.

References

Athans, M. and Falb, P. L. (1966). *Optical Control. An Introduction to the Theory and its Application*, (McGraw-Hill Book Company).

Bock, H. G. and Plitt, K. J. (1985). A multiple shooting algorithm for direct solution of optimal control problems. *Proceedings of the 9th Triennal World Congress of IFAC*, Budapest 1984, (Pergamon-Press, Oxford), **3**, pp. 1603–1608.

Bryson, A. E. and Ho, Y. C. (1975). *Applied Optimal Control. Optimization, Estimation, and Control*, (Hemisphere Publishing Corporation, Washington, D.C.)

Fossen, T. I. and Sagatun, S. I. (1991). Adaptive control of nonlinear systems: A case study of underwater vehicles, *J. Robotic Systems*, **8**, 393–412.

Fossen, T. I. (1991). Nonlinear modelling and control of underwater vehicles, doctoral dissertation, Division of Engineering Cybernetics. The Norwegian Institute of Technology, 1991.

Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical Optimization*, (Academic Press Limited, London).

Goh, C. J. and Teo, K. L. (1988). Control parameterization: a unified approach to optimal control problems with general constraints. *Automatica*, **24**, 3–18.

Hargraves, C. R. and Paris, S. W. (1987). Direct trajectory optimization using nonlinear programming and collocation. *J. Guidance and Control*, **10**, 338–342.

Horn, M. K. (1990) Solution of the optimal control problem using the software package STOMP. In H. B. Siguerdidjane and P. Bernhard (ed.) *Control Applications of Nonlinear Programming and Optimization 1989*. IFAC Workshop Series, Number 2, 51–56.

Keller, H. B. (1968). *Numerical methods for two-point boundary-value problems*, (Blaisdell Publishing Company).

Kraft, D. (1985). Comparing mathematical programming algorithm based on Lagrangian functions for solving optimal control problems. In H. E. Rauch (ed.) *Control Applications of Nonlinear Programming*, (Pergamon Press, New York).

Kraft, D. (1985). On converting optimal control problems into nonlinear programming problems, In K. Schittkowski (ed.) *Computational Mathematical Programming*, (Berlin, Springer).

Kraft, D. (1989). TOMP–Fortran modules for optimal control calculations, *Lehrgang R1.06: Optimierungsverfahren—Software und Praktische Anwendungen*, (Carl-Cranz-Gesellschaft, Oberpfaffenhofen, Germany).

Lasdon, L. S., Mitter, S. K. and Warren, A. D. (1967). The conjugate gradient method for optimal control problem, *IEEE Trans. Aut. Control*, **12**, 132–138.

Luenberger, D. G. (1984). *Linear and nonlinear programming*, 2nd edition (Reading, Mass. Addison-Wesley).

Newman, J. N. (1977). *Marine hydrodynamics* (Massachusetts, MIT Press).

Spangelo, I. and Egeland, O. (1992). Generation of energy-optimal trajectories for an autonomous underwater vehicle, *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France, 2107–2112.

Spong, M. W. and Vidyasagar, M. (1989). *Robot dynamics and control*, (New York, John Wiley).