Modelling the energy budget and prey choice of eider ducks

A part of this study has been carried out with financial support from the Commission of the European Communities, Agriculture and Fisheries (FAIR) specific RTD programme CT 98-4201 ESSENSE.
It does not necessarily reflect its views and in now way anticipates the Commission's future policy in this area.
The study was extended in the framework of the EVA-II study on the effects of shell fishery on nature values in the Dutch Wadden Sea, ordered by the Ministry of Agriculture, Nature Management and Fisheries on October 25th, 2001, ref. nr. TRCDWK/2001/3261 and on May 27th, 2002, ref. nr. LNV:OND/2002-1/5b/01

Modelling the energy budget and prey choice of eider ducks

A.G. Brinkman
B.J.Ens
R. Kats

Alterra-rapport 839

Alterra, Wageningen, 2003

ABSTRACT

Brinkman, A.G., B.J.Ens & R. Kats, 2003. *Modelling the energy budget and prey choice of eider ducks*. Wageningen, Alterra, Alterra-rapport 839. 134 pp. 35 figs.; 39 refs.

We developed an energy and heat budget model for eider ducks. All relevant processes have been quantified. Food processing, diving costs, prey heating, the costs of crushing mussel shells, heat losses during diving as well as during resting, and heat production as a result of muscle activity are distinguished. In the report, blue mussels are regarded as food source, and we computed the profitability of small to large sized mussels for a duck. Finally, the possibility for a duck to survive winter conditions when feeding on a mussel population with a known size frequency distribution is calculated during a simulation run. Conditions for four sites in The Netherlands, France, Denmark and Sweden are regarded and there implications for the foraging behaviour of an eider duck are computed. The implications of the presence of barnacle epibionts are briefly discussed.

Keywords: eider duck, energy budget, heat budget, simulation model, Wadden Sea, food uptake, blue mussel.

Main cover photo: A. v/d Berg, Utrecht
Small cover photo's: A.G. Brinkman

ISSN 1566-7197

This report can be ordered by paying € 30,- to bank account number 36 70 54 612 by name of Alterra Wageningen, IBAN number NL 83 RABO 036 70 54 612, Swift number RABO2u nl. Please refer to Alterra-rapport 839. This amount is including tax (where applicable) and handling costs.

P.O. Box 47; 6700 AA Wageningen; The Netherlands
Phone: + 31 317 474700; fax: +31 317 419000; e-mail: info@alterra.nl

[Alterra-rapport 839/november/2003]

Contents

Summary

The aim of this report is to describe the energy budget of eider ducks and to simulate the energy budget under various ecological conditions. By means of such an energy budget, we intend to study which foraging areas are preferred by eider ducks, and why they are preferred. As a third target, we want to find out why eider ducks winter in areas like the Dutch Wadden Sea and not or hardly in the at first sight suitabele areas Easter Scheldt (SW-Netherlands) or the Bay d' Oléron (West France).

The project was partly carried out in the framework of the EU-ESSENSE project (FAIR-RTD Programme CT 98-4201), and extended as Alterra project 11936-01 "food availability for birds", sub-project "food demand of eider ducks", plus as part of the EVA-II study on the effects of shell fishery on nature values in the Dutch Wadden Sea, both ordered by the Ministry of Agriculture, Nature Management and Fisheries.
The major objectives were:

1. analysis of the impact of predation on mussels and mussel requirements of birds,
2. to develop a simple distribution model for the eider duck,
3. to validate the distribution model with field data,
4. to perform scenario calculations on the effects of shellfishing and shellfish culture on shellfish eating birds.

In the report, we distinguish six main sections:

- a brief description of foraging habits of eider ducks (ch 2),
- an overview of an energy budget for a duck (ch 3, 4),
- characteristics of eider ducks and of shells (ch 5),
- a detailed description of the eider duck energy budget and the prey decision strategy as implemented in the model (ch 6, 7),
- a brief description of the model (ch 8),
- discussion of results, where we compare four sites in Sweden, Denmark, The Netherlands and France, and describe the local energetic profits for a duck (ch 11).

Based on the model computations we conclude that:

- in summer eiderducks cannot profit from shells smaller than 20 mm. In winter only shells larger than 30 mm are taken. In summer costs are lower (especially heat losses are much lower) while the profits (meat content) per shell is higher. Maybe, eider ducks prefer smaller shells , but have to take the larger ones in winter because of energetic needs
- daily energetic expenditures (DEE) ranges from 2.9 10^6 J in winter to about 1.5 10^6 J in summer
- eider ducks are probably capable to remove a significant part of the subtidal mussel stock the model can not yet be used to study the interaction between

fishery activities and foraging of eider ducks.

1 Introduction

An individual based energy budget model for the Common Eider (*Somateria mollissima*) is developed to study the influence of availability and quality of prey on the food demand and foraging behaviour of eider ducks in winter.

In North West Europe, the Common Eider breeds from the Dutch Wadden Sea to the Northern Baltic Sea areas. The total population coupled to the North western European flyway is about 3 million birds (LWVT/SOVON, 2002). At the end of summer and the beginning of autumn, a proportion of the ducks from the northern regions migrates in a south-west direction and spends the winter in the Dutch (150.000 exx, LWVT/SOVON, 2002), German and Danish Wadden Sea and the adjacent North Sea (250.000-300.000 eider ducks total) (Meltofte et al. 1994).

In the Wadden Sea, they predominantly feed upon blue mussels (*Mytilus edulis*), partly occurring on wild mussel beds, and partly on mussel culture lots. The shells are swallowed and crushed in the bird's stomach. Other potential prey like crabs (*Carcinus meanas*), starfish (*Asterias rubens*) and other shellfish like cockles (*Cerasteroderma edule*), baltic tallins (*Macoma balthica*), and razor shells *Ensis americanus* are also taken. Crabs are not considered as a valuable food source. Cockles have a lower and less profitable flesh/shell mass ratio compared to mussels when shells of the same size are considered, because the cockle shells are thicker, and probably harder to crush.

During the last decade of the 20th century the eider ducks in the Netherlands showed a shift in distribution to the North Sea. Here they are probably feeding on shellfish species, such as *Spisula subtruncata*, *Donax vittatus* and *Ensis americanus*.

In the framework of the EU-ESSENSE (FAIR-RTD Programme CT 98-4201) project, and the Alterra project 11936-01 “food availability for birds”, sub-project "food demand of eider ducks”, carried out for Ministry of Agriculture, Nature Management and Fisheries in The Netherlands, we intended to quantify the several relevant eider duck energy and heat budget terms as accurate as possible. With such budget models, one is able to check the consistency of the many separate descriptions that can be found in literature. Thus, we hoped to collect additional information on the prey selection mechanisms of eider ducks.
The results were applied in the EVA-II study on the effects of shell fishery on nature values in the Dutch Wadden Sea, ordered by the Ministry of Agriculture, Nature Management and Fisheries.

2 Feeding habits of the Common Eider

2.1 Prey collection

Eider ducks feed on prey that is submerged most of the time. Depending on the water level during the tidal cycle, eider ducks dive and catch their prey from and out of the sediment; and sometimes they dabble in shallow waters.

Prey species attached on the sediment, like mussels in mussel beds, are collected using a different feeding technique compared to shellfish species living in the sediment. Mussels in mussel beds are attached to each other with byssus threads and are collected in clumps depending on the size of the mussels in the clump. Large mussels are expected to be collected one by one and with decreasing size the number of mussels collected in the clumps during one dive will increase. Once a mussel or a clump of mussels is taken to the surface, the mussels are handled before ingestion. Large mussels are swallowed one by one and mussels in clumps depending on the size of the mussels in the clump are swallowed in clumps. The same occurs when feeding on crabs (*Carcinus maenas*) or starfish (*Asterias rubens*). After returning to the surface, the ducks swallow small prey completely. Large crabs and starfish are shaken so that legs fall off and then, the body is eaten. After the prey is swallowed the duck may quickly dive again to catch the rests of the prey, or other shells from the clump (Swennen 1976). Nehls (1995) assumed that eider ducks only take one mussel per dive. Crop analysis (Kats et al, 2002), laboratory observations on captured birds (Swennen 1976), and clump selection by other diving ducks (Leeuw 1997) show that mussels are also collected in clumps and handled at the surface before swallowing.

In the Dutch Wadden Sea, cockles are taken by eiders ducks and can be in some years almost as important as mussels (Swennen 1976), although prey availability, predictability, quality and profitability may influence the bird's choice. Cockles live in the sediment, are not tied to each other and are found in cockle beds. The cockles are mostly collected from high density cockle beds. In (very) shallow water, the dabbling duck tramples a part of the sediment containing cockles and thus creates a dish-like depression in the sediment with a diameter of about 30 to 40 cm. Thus, the cockles become visible and can easily be selected and swallowed by the duck.

From mussel culture lots, where mussels occur more loose, the duck needs less time to catch and handle a prey.

2.2 Size selection

Eider ducks are capable to feed upon a large size spectrum ranging from very small prey of a few millimetres in length to large prey of more than 100 mm shells or crabs (Swennen 1976; Beauchamp et al, 1992; Madsen 1954). See Kats et al (2002, in prep) for a review on prey size selection. Cockles were taken up to 49 mm shell length,

although under normal conditions (Nehls, 1995), the upper cockle limit was about 40 mm. Mussels, being more slender, were accepted up to 65 mm (Nehls 1995) although the preferred range was 30-52 mm.

Nehls (1995), Swennen (1976), Hamilton et al. (1999) mention that from a food supply with large variation in size eider ducks prefer medium sized mussel and cockle shells, although larger shells probably are more profitable regarding energetic yield. Large shells contain more flesh and shell mass. Energetic yield is defined as the difference between the energetic profit and all the costs needed to collect, handle and digest that prey. During winter, this preference shifted towards larger prey sizes (Hamilton et al. 1999; Nehls 1995). Also, diving ducks usually took somewhat larger shells than dabblers did (Nehls, 1995). Swennen (1976) describes a number of threats a duck may suffer if it swallows too large shells, like suffocation, or damaging the stomach muscle.

Barnacle overgrowth negatively effects the size spectrum taken, not only because of the increased size of the mussel compared to the mussel alone, but also because barnacles may be an additional risk for the duck during swallowing.

Swennen (1976) described that ducks inspected the shell surface before swallowing the shells, which might be related to the danger of damaging digestive organs. Another explanation of this phenomenon might be that damaged shells might indicate the presence of parasites.

2.3 Feeding rates

When a duck has an empty gullet, stomach and gut, it can eat a lot of shells in a short time. Swennen (1976) mentioned a duck that swallowed 28 25 mm cockle shells in 1.5 minutes, after being hungered for about 12 hours. After 58 minutes, the first faeces appeared, which stopped after 86 minutes. In this period, about 65 grams of grit was excreted, which corresponded to the total shell mass of the ingested cockles. Common eider ducks are possibly able to feed at different feeding rates and still meet their daily energy intake. Feeding can take place on different time scales between continuous feeding at low intake rates and/or interval feeding at high intake rates. When the crop is full feeding stops. Eider ducks feed most of the time on submerged prey, although it's not completely clear why.

2.4 Interference

When feeding, birds may interact with other birds. Mostly, this interaction is an attack by another bird that is interested in the prey. For foraging oystercatchers (*Haematopus ostralegus*) this process is well documented (Ens & Cayford, 1996; Goss-Custard et al, 1996; Hulscher, 1996; Stillman et al, 1997; Van der Meer & Ens, 1997; Zwarts & Drent,1981). The foraging efficiency (or intake rate) of oystercatchers decreases with increasing oystercatcher density, which has implications for the

number of oystercatchers that can sufficiently feed on a certain area. Eider ducks usually show much less interference. An observation in the Texel harbour learned that there was a major interaction between gulls (Herring gulls *Larus argentatus* and lesser black-backed gulls *Larus fuscus*, not the smaller black-headed gulls *Larus ridibundus*) and eider ducks. The ducks, feeding on small crabs (*Carcinus maenas*), of about 10 cm maximum size (leg to leg), were chased by the gulls and had to dive frequently to avoid contact with the gulls. There was not any attempt by a duck trying to catch another duck's prey.

For the present model development, we decided not to implement a description of interference. Thus, in the model the ducks can forage without being influenced by their own density.

3 Energetic considerations

3.1 Introduction

For a bird it is only relevant to forage if the energy that can be gained is larger than the energetic costs of catching and processing that same prey. Also, the bird needs a certain basic amount of energy per day; the prey energy surplus has to cover that basic amount as well. In the next parts, we consider four types of costs:

- Σ Basic costs, needed to maintain the organism
- Σ Environmental costs, related to heat losses
- Σ Feeding and prey processing costs, describing the costs of e.g. shell crushing and digestion
- Σ Activity costs, that describe the costs of flying and swimming.

3.2 Heat and kinetic energy

Most of the authors only discuss energy as an all-embracing quantity. In this report, we distinguish between heat and kinetic plus potential energy as two different types of energy. When swimming, as an example, the bird has to overcome friction forces. Movement of the feet comprises an energetic cost. But part of that energy is a loss in terms of heat, which increases body temperature. Even, water is moved then; after dissipation as a result of viscosity, this part of the kinetic energy is converted into heat as well. Thus, we are not only bookkeeping the energy budget, but also the heat budget.

3.3 Processes

The energy gain of a prey is directly related to the meat content of the prey, the assimilation efficiency *and* the energy content of the meat. The non-assimilated part will be excreted as faeces.

When a bird is not active it still has to use energy reserves for maintenance. This is called the basal metabolic rate (BMR). BMR is an energetic cost but it becomes manifest as heat and thus adds to the heat budget. This is the reason that a non-feeding bird can maintain its body temperature under not too cold conditions (Jensen et al, 1989a,b).

The energy losses concern all the other processes. Prey is heated (to body temperature), there is a heat flow (loss) through the feather layer and breathing results in heat losses. All these losses have to do with the temperature difference between the bird's body (40 °C) and the temperature of the air or water (somewhere between 20 and -10 °C). Digestion also costs energy because the animal has to

produce enzymes which are lost as chemical energy together with the faeces.
Movements of the birds cost energy. While swimming, the bird has to overcome a drag that is related to the shape of the animal and to the characteristics of the fluid. While diving, the bird first will accelerate, and then it has to overcome a form and viscosity drag. The bird also has to overcome buoyancy. The specific mass of the bird plus feathers plus lungs is less than that of the fluid. While diving, the bird gets an increasing potential energy that is lost as soon the bird emerges later on. When the sediment is reached, the bird has to stay there and has to overcome buoyant forces. It also has to remove mussels attached to the bottom (or to other mussels) one by one or by clump selection, or to gather cockles or other shellfish from the sediment. During the stay at the sediment or the diving the bird does not increase it's own kinetic energy and thus all the energy produced by the animal is lost. As long as the animal stays at the sediment, it has to paddle its feet, producing a force directed towards the sediment and moving the fluid in the opposite direction. By this, the animal adds energy to the environment which dissipates and actually results in heating the water. This production of energy does not happen without any losses and thus the duck itself is heated as well. The same happens during the actual dive. The efficiency of producing a force downward (overcoming the buoyancy and drag forces) is not 100%. First, not all the energy transferred to the water is transformed in a downward directed force, secondly the production of energy in the duck's body will not completely be used to move the feet. The first heat loss implies that the duck will loose more energy than strictly needed. It will however not heat the bird, but only the water. The second loss means that the energy production is not 100% efficient. The energy loss results in heat.

The last energetic term that is to be considered is flying. The duck flies from the summer breeding areas to the wintering grounds. It may fly from one foraging site to another, and it may fly as a result of disturbance. Knowledge of flight costs may help us to understand decisions a bird has to make on staying on a certain site or flying to a site with better food conditions.
Flying forms an only very limited part of the present model. We assume a certain fly activity per day and compute its costs.

In the next section the energy terms will be discussed.

4 Energy budget of an adult eider duck

The energy budget is based on an adult bird. Yearly growth is zero and therefore storage may be a better term for the difference between yield and losses. Generally, the energy budget reads:

$$storage = assimilation - costs \tag{1}$$

During bad feeding conditions, the animal can use body reserves down to a certain level. Such a minimum level may be a useful characteristic when the description of survival is at stake. Thus:

$$mass(target) = f(month)$$

$$mass \in [mass_{\min}, ..., mass_{\max}] \tag{2}$$

$$if\ (mass(real) < mass_{\min}) \rightarrow increased\ mortality$$

In order to find out how much food an animal needs an energy budget is needed. Therefore, the costs have to be divided into 'standard costs', i.e. all the costs an animal would make when it is not feeding at all, and 'foraging costs'. Foraging costs include all the *extra* costs that a bird makes in order to gather food. Foraging results in a gain: 'foraging yield'. The difference between yields and costs is the profit. For a balanced energy budget, the total profit has to equal the standard energy costs. The number of dives follows from:

$$number\ of\ dives = \frac{standard\ energy\ costs}{profit\ per\ dive} \tag{3}$$

Now, we have to specify both nominator and denominator.

First, the standard energy costs concern BMR, heat loss during resting and breathing, and the costs of some movements the bird makes. In the model it is assumed that eider ducks always fly a litle, and always swim a little. Therefore, the costs associated with these activities are considered to be standard costs:

$$\begin{aligned} standard\ costs = {} & heat\ loss\ feathers + heat\ loss\ breath \\ & + swim_{\min} + basic\_flying + BMR \end{aligned} \tag{4}$$

where BMR is the basal metabolic rate. A specification is given in section 6.

Next, the profit per dive follows from:

$$profit\ per\ dive = yield\ per\ dive - costs\ per\ dive \tag{5}$$

The yield per dive *and* the costs per dive both depend on the kind of prey. Thus, for a certain prey we have to sum up all costs and yields. Costs and yields will, among others, depend on characteristics of the prey such as size and availability. The latter may include density of the prey as well, although this aspect is not treated in his report..

First, the costs will be specified. Costs are related to all activities of the duck: heat demand, diving costs, costs for crushing the shells, and costs for production of e.g. enzymes that serve to transform the food into assimilation-ready substances. Also, when more kinetic energy is produced, the duck has to increase breathing and will lose more heat. All costs are expressed here in Joules. We distinguish between costs associated with the eider duck and costs that are associated with the prey:

$$costs\ per\ dive = animal\_costs + prey\_costs \tag{6}$$

The energy costs associated with the eider duck include:

$$\begin{aligned} animal\_costs = {} & extra\ heat\ loss\_feathers + diving\_costs \\ & + extra\ swim\ \min g \\ & + extra\ breathing + \\ & + extra\ flying + extra\ resting \end{aligned} \tag{7}$$

Extra swimming and flying are not specified further (in this report) in the prey_costs because we considered one site at a time. All terms except diving are additional costs next to the basic expenditures already mentioned in eq. (4). The prey_costs include:

$$prey\_costs = heating\_shells + crushing + digestion + salt\_excretion \tag{8}$$

These terms will each be explained in chapter 6 and on.

In the model, food uptake (mass per unit of time) is transformed to energy uptake (Joules per unit of time). The part of the food that is not assimilated is excreted as faeces. Thus:

$$assimilation = uptake * assimilation\_efficiency \tag{9}$$

and

$$faeces = uptake - assimilation \tag{10}$$

Assimilation efficiency depends on stomach and gut passage time and thus on food intake rate. It may be important whether an animal feeds slowly but continuously or fast with resting periodes to digest the food. The basics of the so-called *digestive bottle-neck* may be applied here.

The amount of food caught each dive depends on the size and the flesh content of the prey. Although it is believed that usually only one prey is caught per dive, a model may contain a ‘more-prey-per-dive’ option, to test the profitability of a more efficient foraging behaviour.

For a more precise specification of energy costs, first a description of some characteristics of an eider duck (shape, buoyancy, etc) is needed.

5 Size and mass characteristics of eider ducks, mussel and cockle shells

5.1 Eider ducks

Body area including and excluding feathers, body volume and buoyancy
For the computation of fat - and feather thickness we need to know the body surface area (A, m^2) of an eider duck in relation to duck mass (W, kg) and/or duck length (L, m). In order to estimate additional characteristics, some assumptions on the body shape of an eider duck are made. By trial and error it appeared that a cone shape can be used to describe body shape (App. A). The frontal area of a bird (to compute body drag while diving) and feather thickness (to compute isolating properties of the feathers and fat) can be estimated with equations that describe the cone shape.

In text box 1 and 2 we summarized some literature data on duck characteristics. The data given by several authors had to be unravelled, because they appeared to be inconsistent with respect to methods and units.

Body area
Wallsberg & King (1978) describe the body area as a function of animal mass, underestimating the body area as given by Lovvorn et al (1991b). The latter gave two equations. The authors preferred a linear relation. Our 'cone'-equation, including feathers, reproduced the allometric equation by Lovvorn et al (1991) quite nicely (Figure 1). The equations used by Jensen et al (1989a,b) produced a slightly larger body area (5% compared to Lovvorn et al (1991b) for a 1000 g bird and 10% for a 2000 g bird).

Body volume
Lovvorn & Jones (1991a, 1991b) and Woakes & Butler (1983) gave linear equations for body volume as a function of mass (in their equation "volume=a+b mass" the value of the constant a was #0, so the equation was not apllicable for small ducks). The proportionality constant (b) in this equation represents the specific mass of the duck plus the volume of the respiratory system.

The non-linear equation by Lovvorn & Jones (1991b) resulted in much higher values of body volume for eider ducks and scoters than the linear ones did (Figure 2).

Buoyancy and volume of the respiratory system
Buoyancy is the resultant of body volume plus feather air volume and the volume of the respiratory system. Lovvorn and Jones (1991a,b) mention that eider ducks and scoters have a relatively large buoyancy compared to most other diving ducks. It is not clear whether they distinguished between diving and swimming eider ducks. We found a difference regarding buoyancy during diving and during resting. Eider ducks, for example, have an average specific mass during diving of about 0.7 kg dm^{-3} (Lovvorn & Jones, 1991b). If the same specific mass is valid for a swimming duck this would imply that 70% of the duck would be submerged and 30% emerged. Our

impression is that this is not the case. At least 50% of the duck is above water (estimated by four specialist). In some cases the duck is almost under water except the head but normally it looks like at least half of the duck is above water. In the model a value of 50% is used. Partly, the difference between specific mass during diving and during resting is the result of exhalation diving ducks perform prior to diving. Tufted ducks can decrease the respiratory content by 42% from 232 tot 165 cm^3 kg^{-1} (Stephenson et al, 1989), which represents a buoyancy reduction from 1.6 to 1.1 N regarding the respiratory system alone. Since buoyancy of a tufted duck (mass about 700 g for a male adult) is about 3 to 4 N, Stephenson's data imply a 30% (26% during diving, 40 to 50% during swimming) contribution of the respiratory system to buoyancy. Others give that the respiratory content represents about 160 to 220 cm^3 kg^{-1} (Kooijman, 1975; Stephenson et al, 1989b) resulting in 1.1 to 1.5 N contribution to buoyancy for a 700 g tufted duck. Keijer & Butler (1982) found a volume of the respiratory system for a tufted duck of 180 cm^3 kg^{-1}(130 cm^3 for a 700 g bird; 1.3 N buoyancy contribution). The respiratory system of an eider duck (2000 g) amounts about 320 to 440 cm^3 (Stephenson et al; 1989b) or 320 cm^3 Lasiewski & Calder; 1971). This represents a contribution to buoyancy of 3.2 to 4.4 N on a total of about 10 N. With a specific mass of about 0.7, a 2000 g bird would have a buoyancy of about 8.6 N (and a total air volume of 860 cm^3).

If the specific mass for a swimming bird is lower than 0.7, e.g. 0.5 kg dm^{-3}, the buoyancy of a 2000 g eider duck would be 20 N and the total air volume 2 dm^3. In Figure 3 buoyancies are given as a function of duck mass.

Feather volume and thickness

Feather volume is mainly computed from the computed body volume minus the content of the respiratory system. Feather volume divided by feather area gives feather thickness. Wilson et al (1992) give a formula for feather volume but the result (thickness=volume/area; area as an average of a body surface of a bare bird plus feathered area) does not seem to be in line with most of the other results. For the eider duck example in the previous paragraph the total air volume amounts from 860 cm^3 to 2000 cm^3. Subtracting a respiratory volume of 320 cm^3 (the minimum value) to 440 cm^3 (the maximum value), the feather volume ranges from 540 cm^3 to about 1560 cm^3. Divided by the body area (the mean between bare area represented by the cone-equation, and the feathered area) the thickness of the feather layer is computed. This yields an area of 0.5*(1217 cm^2 + 1378 cm^2) = 1300 cm^2 for a 2000 g eider duck, and 0.42 cm to 1.1 cm for the feather thickness. In Figure 4 feather data are presented.

Frontal area

We did not find data on frontal areas but had to estimate these from the cone characteristics (app. A). In Figure 5, results are shown. These data are needed for the computation of hydrodynamic body drag during diving.

Salinity effects

A possible salinity effect on buoyancy depends on the difference between specific mass of fresh water and of salt water. Since the specific mass of water with normal salinity (about 35‰) is only about 3.5% higher than of fresh water, effects of salinity

may be neglected. Lovvorn and Jones (1991a) did some computations on the salinity effect but came –as expected- to only minor effects.

Mass ranges and variation

The mass of an adult duck varies during a year. This is the result of a strategy to be efficient with food and energy and associated costs of carrying a heavy body around (predation risk) or an effect of season, moult, breeding etc.

In an ideal situation, an eider duck would show an ideal development of mass. As an average a male adult weighs about 2200 grams (Bauer & Glutz von Blotzheim, 1969). Since flying costs are high, the bird has to be fat prior to migration (up to 2600 grams) and will be lean after the migration (down to about 1800 grams). During the breeding season the bird's mass will vary around it's preferred average Before the winter period starts it will need to have sufficient reserves. The variation in mass during a year is presented in Figure 6 (after Nehls, 1995). The real mass during a year depends on food availability, food quality, weather conditions, and more In our model the duck has its ideal mass from this we compute the required amount of food.

Text box 1. Characteristics of ducks.

All characteristics are expressed as value = $a + b * Mass + g * Mass^2$, exceptions are explained in the table (see also text box 2). Mass in gram.

Body volume (cm^3)	α	β	γ	
Lovvorn & Jones, 1991b	65.74	1.144	2.49e-4	All diving ducks
	13.7	1.43		Diving + surface feeding ducks, except eiders and scoters
	-29,94	1.467		All diving ducks, except eiders and scoters
Lovvorn & Jones, 1991a	42.97	1.387		All species, summer
	339.2	0.9711		Lesser Scaup, winter
	213.6	1.2506		Redhead + Canvasback, winter
Woakes & Butler, 1983	339	0.9771		Tufted Duck

Buoyancy (N)	α	β	γ	
Lovvorn & Jones, 1991b	1.21		3.17e-6	All diving ducks
	-0.303	4.58e-3		All diving ducks, Mass<1200 g
	0.131	4.18e-3		Diving + surface feeding ducks, except eiders and scoters
	$8.034e\text{-}4 * M^{1.1253}$			All diving ducks, including scoters and eiders. Overestimates scaups, underestimates scoters and eiders

Area (cm^2)	α	β	γ	M in gram
Lovvorn , Jones & Blake, 1991	345	0.611		Canvasback, Lesser Scaup & Redhead; bill + feathered body
	$15.2 * M^{0.595}$			Feathered body without bill
Wallsberg & King (1978)	$8.11 * M^{0.667}$			Feathered body without bill

Text box 2. Characteristics of ducks.

All characteristics are expressed as value = $a + b * Mass + g * Mass^2$, exceptions are explained in the table (see also text box 1). Mass in gram.

Feather air volume (cm^3)	α	β	γ	M in gram
Wilson et al, 1992 Method after Cramp & Simmons, 1983 and Harpe et al, 1985		0.29		Surface diving, or pursuit.
		0.67		Surface feeding or flying
		0.45		plunge or occasionally surface diving
Wilson et al, 1992	-0.138 - 0.44 * Volume (duck)			Volume in cm^3
Lovvorn & Jones, 1991	???	0.1232		

Lung + air sack volume (cm^3)	α	β	γ	M in gram
Keijer & Butler, 1982		0.180		Tufted Duck
		0.112		Mallard
Kooyman 1975, Stephenson et al, 1989b		0.16 à 0.22		
Lasiewski & Calder, 1971	$1.61\ M^{0.91}$			

Frontal area (cm^2)	α	β	γ	M in gram
No literature data	Computed from a cone-shape, including feathers			

Physical limitations concerning prey sizes

There is a minimum and a maximum to the prey size that can be handled bij eider ducks.In Figure 7 an overview of a number of studies is given showing minimum and maximum shell sizes taken up by an eider duck. Nehls (1995) found a normal uptake of about 55 mm mussels. Maximum size of a mussel in the uptake was about 68 mm. The smallest mussels taken by an eider duck were about 3-5 mm large (Nehls, 1995). In other words, probably there is no lower physical limit. When the sizes of shells in the uptake are compared with the availability, the preference can be computed. Figure 8 shows the preference as presented by Nehls (1995).

The upper size limit is determined by the size of the mussels and by the occurrence of barnacle epibionts. Barnacles increase the size of the mussel and have sharp edges which may be dangerous for a duck's gullet. Therefore a duck will decrease the preferred prey size in case of barnacle overgrowth (see section 5.2).This effect also depends on the extent of barnacle overgrowth.

Digestive bottleneck

The feeding rate of eider ducks is limited. Digestion takes time and therefore a duck cannot continuously swallow shells at a high rate. Sometimes there may be restrictions on feeding periods and a bird needs to find a complete meal within a relatively short period. A complete meal is considered here to be the maximum amount of food a duck can carry in its stomach and guts. For birds feeding on tidal flats, where emersion determines the prey availability, birds need to fill their stomach and guts prior to high tide. In this case the volume of guts and stomach, combined with the digestion rate, is limiting the maximum uptake rate.

If it is not necessary to take up complete meals in a short period and a bird can feed more or less continuously, ans consequently, the implementation of a digestive bottleneck is less crucial. Our estimate is that eider ducks do not need to feed within a short period and therefore, we decided to refrain from the incorporation of this phenomenon.

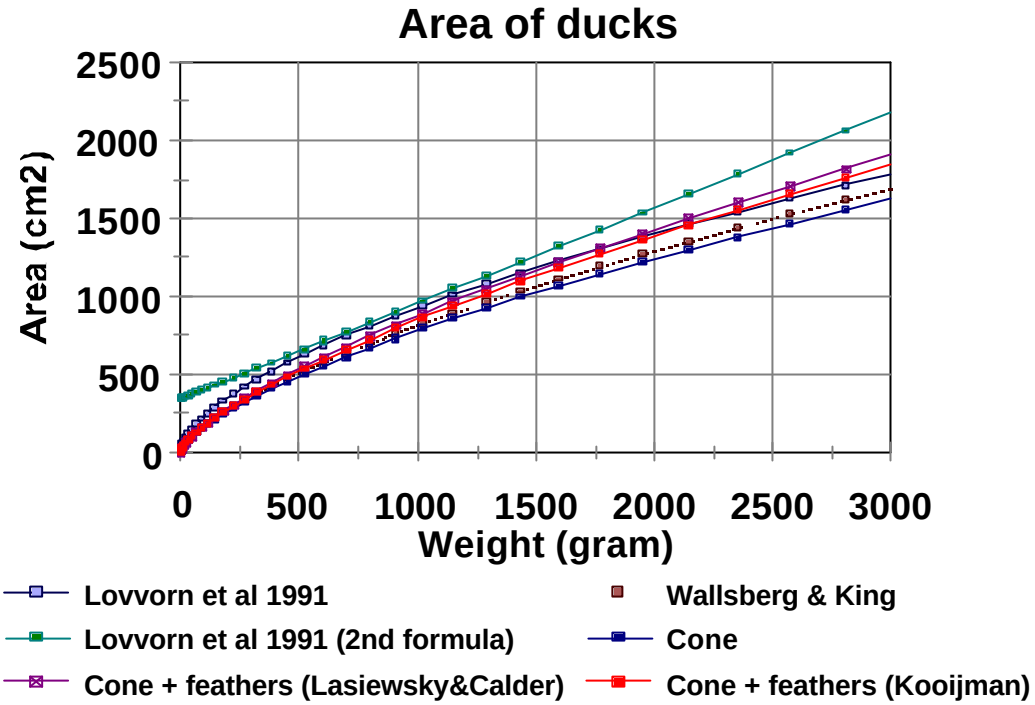


Figure 1. Skin surface area related to body mass of ducks See also table 1.

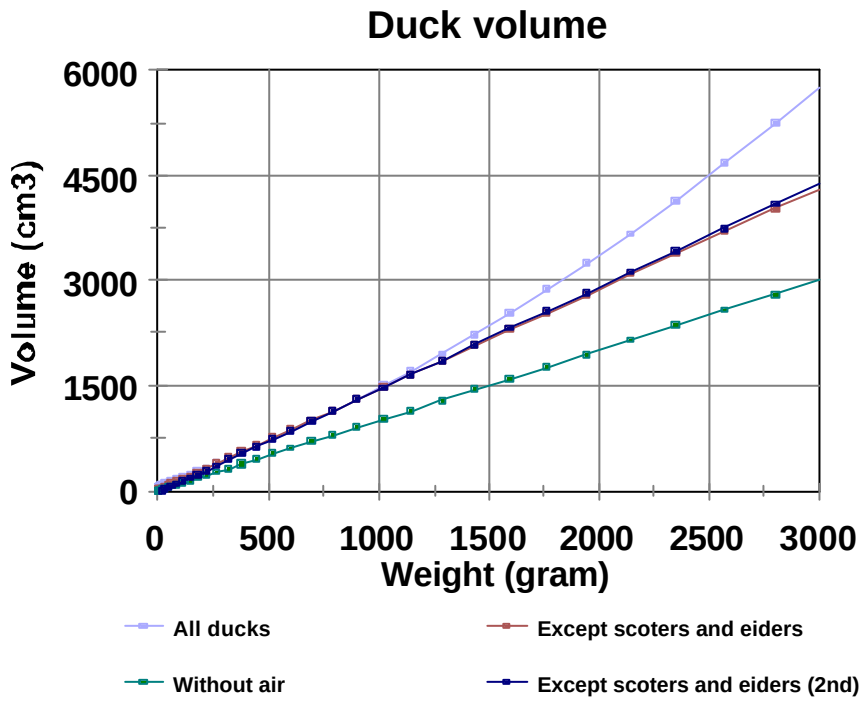


Figure 2. Body volume related to body mass of ducks. See also table 1. From Lovvorn & Jones, 1991b. Without air computed from line for All ducks minus air volume .

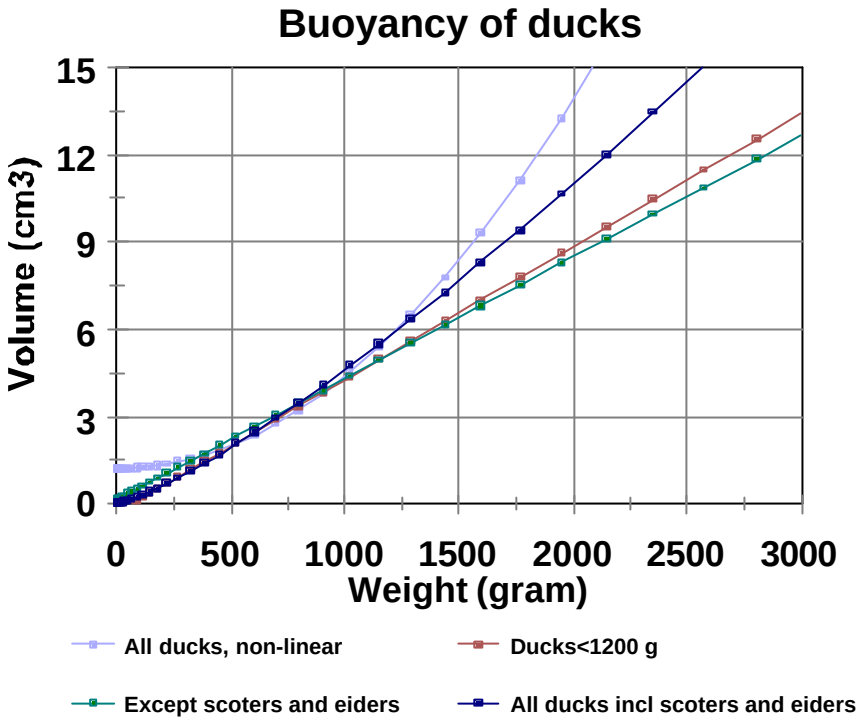


Figure 3. Buoyancy of ducks related to body mass. From Lovvorn & Jones, 1991b. See also table 1.

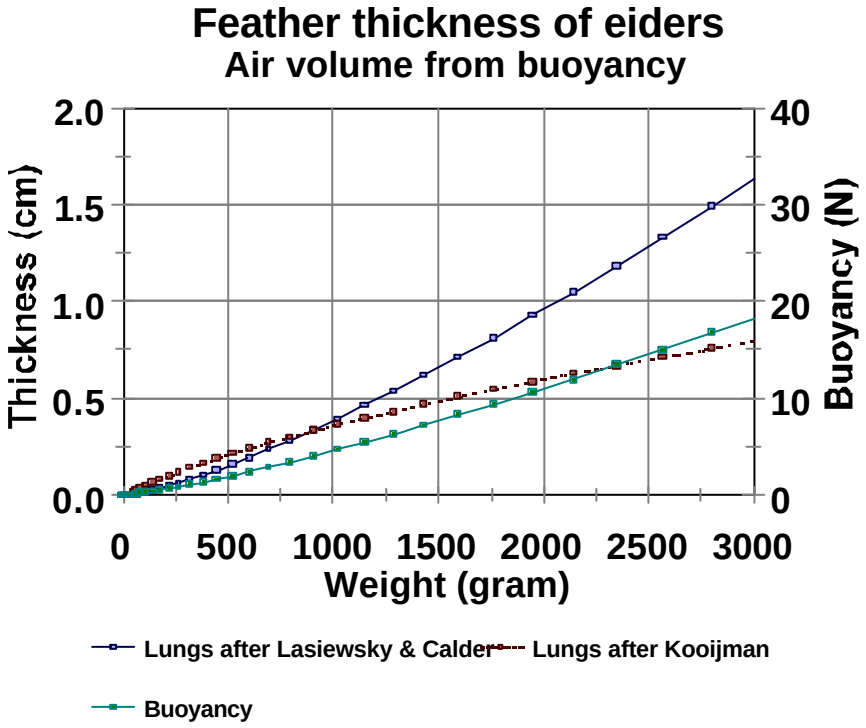


Figure 4. Relationship between thickness of the feather layer and body mass of ducks. See also table 1. Buoyancy equals line "all ducks incl scoters and eiders" from fig. 3.

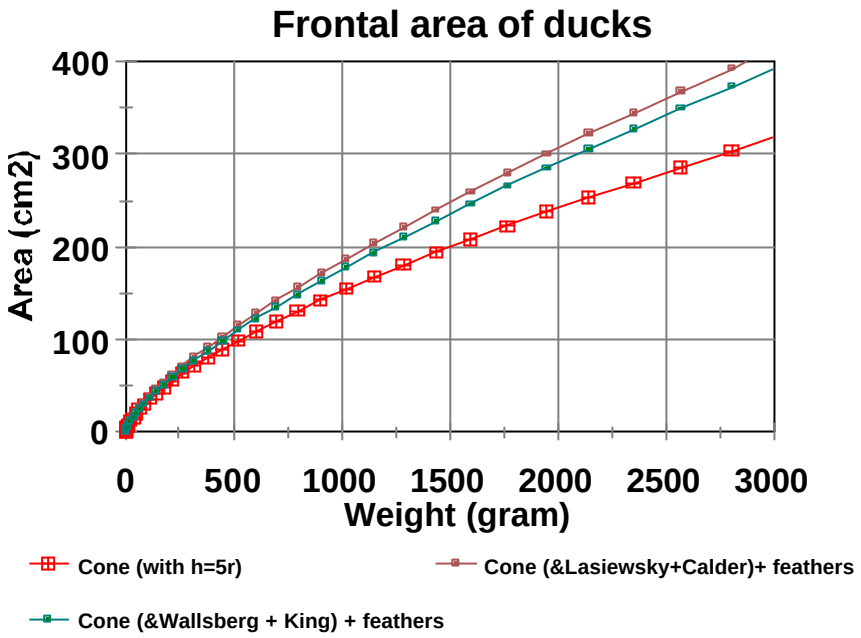


Figure 5. Relationship between frontal area and body mass of ducks. "Cone" refers to idealized shape of an eider duck as explained in Appendix A.

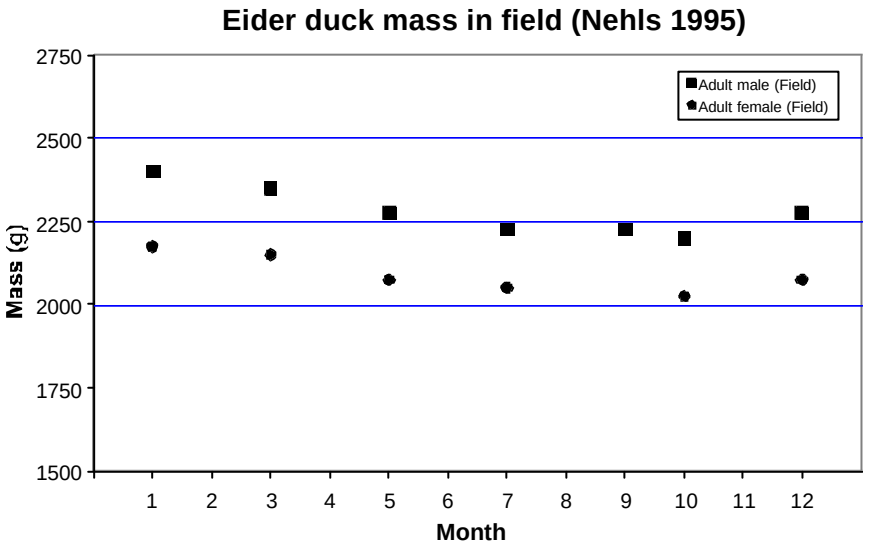
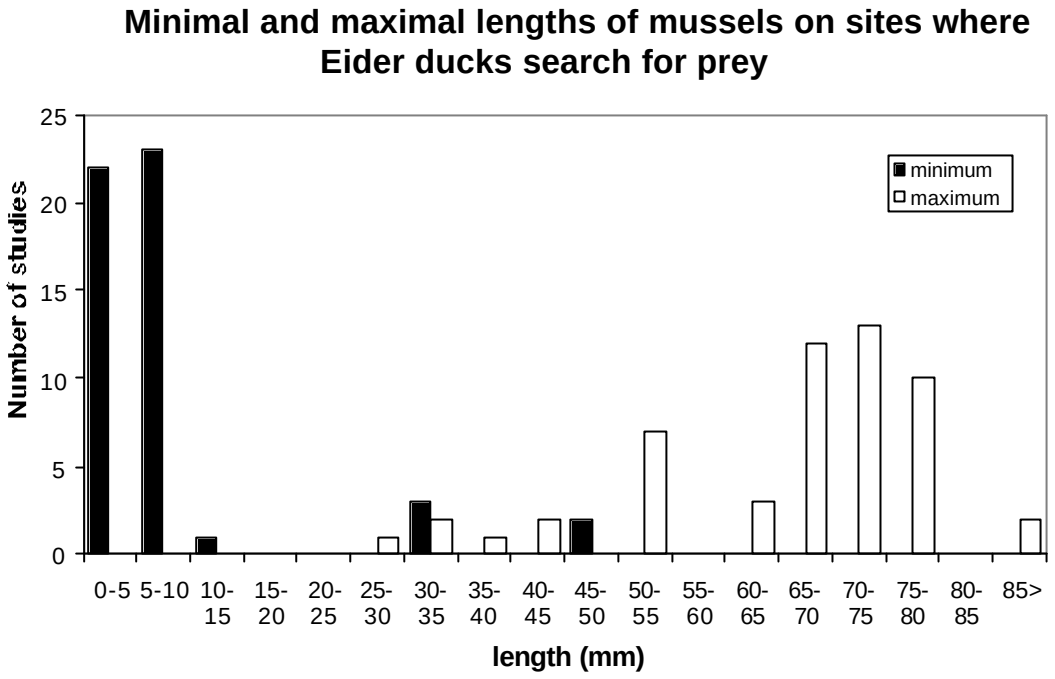


Figure 6. Mass of eider ducks during a year (taken from Nehls 1995).



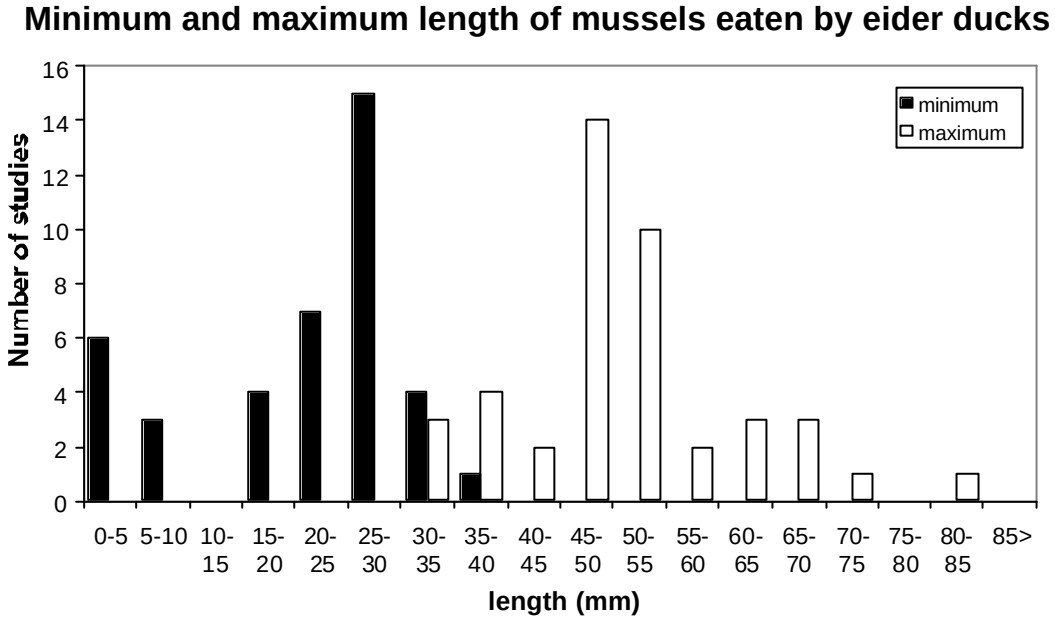


Figure 7. Prey size selection of eider ducks (1). What sizes are available and what sizes are eaten. An overview from several studies.

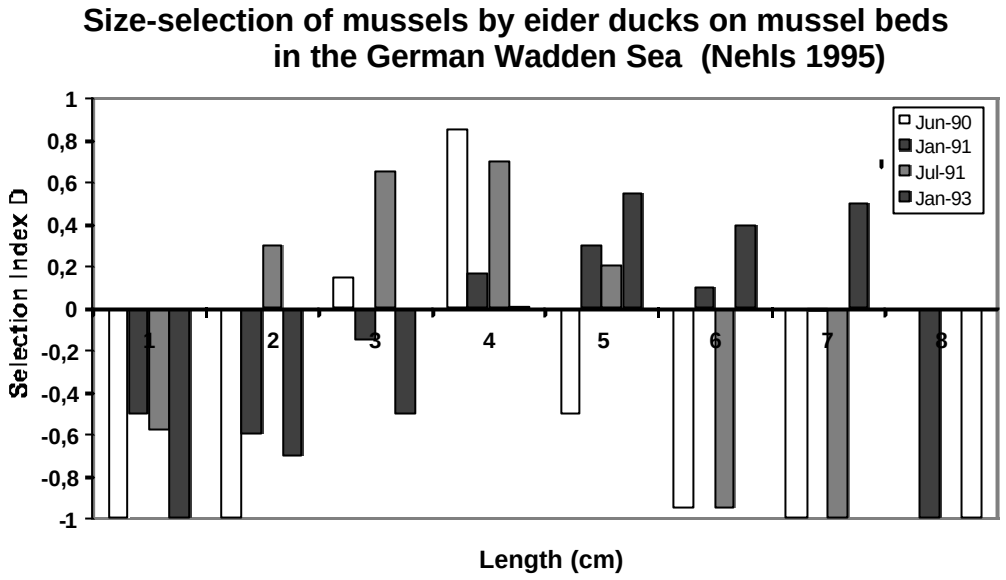


Figure 8. Prey size selection of eider ducks (2). Selection index D: size contribution to uptake / contribution to prey availability. From Nehls (1995).

5.2 Shells

Shellfish in the sub-tidal and the intertidal area are the main food source for eider ducks. Mussels (intertidal and sub-tidal), cockles (mainly intertidal), spisulas and donax (subtidal, in the North Sea coastal zone) may be distinguished in model computations. The shells are crushed in the bird's stomach and the energetic costs of this crushing are related to the shell mass and the shell thickness. The profits are directly related to the amount of flesh present in the shell. Therefore, we need an overview of shell characteristics such as shell mass, breaking forces, shell thickness, flesh mass, ash free dry masses (AFDW), and also the energetic value of flesh mass.

As an additional topic, we discuss the presence of epibionts on the shells that may influence the edibility of shells.

Mostly, an allometric equation is used to express wet, dry, shell and total mussel mass or ash free dry mass as a function of shell length:

$$W = aL^b \quad \text{(g)} \qquad (11)$$

Shell size

Mussels (*Mytilus edulis*) have a maximum size of about 7 cm. On intertidal mussel beds, shells grow slower then in sub-tidal regions. In the Netherlands, the larger part of the sub-tidal mussels occur on mussel culture lots (van Stralen, 1998; Bult et al, 2003) where they grow to about 5-6 cm in 2.5 years. Mussels on culture lots occur more loose than on mussel beds where bissus-threads may fix a mussel tightly. Intertidal mussels grow to a size of 4.5-5 cm in 2.5 years. On culture lots the average

size ratio length: width: height is about 6:2.3:2.9 (own measurements, based on about 2000 subtidal mussel shells (De Jager, 2002; see Figure 9). Cockles (*Cerasteroderma edulis*) have a maximum size of about 5 cm and are much thicker and wider than mussels. Data on thickness are available, but not analysed yet. The shells are individually buried in the sediment and can occur in extremely low to extremely high densities. Relatively sandy areas are preferred habitats, but cockles also occur in silty areas. Consequently, cockles are more difficult to find, but easier to handle.

Spisulas (*Spisula subtruncata*) occur in the North Sea coastal zone. The maximum size is about 4-5 cm. Spisulas are an important food source for eider ducks foraging in the North Sea.

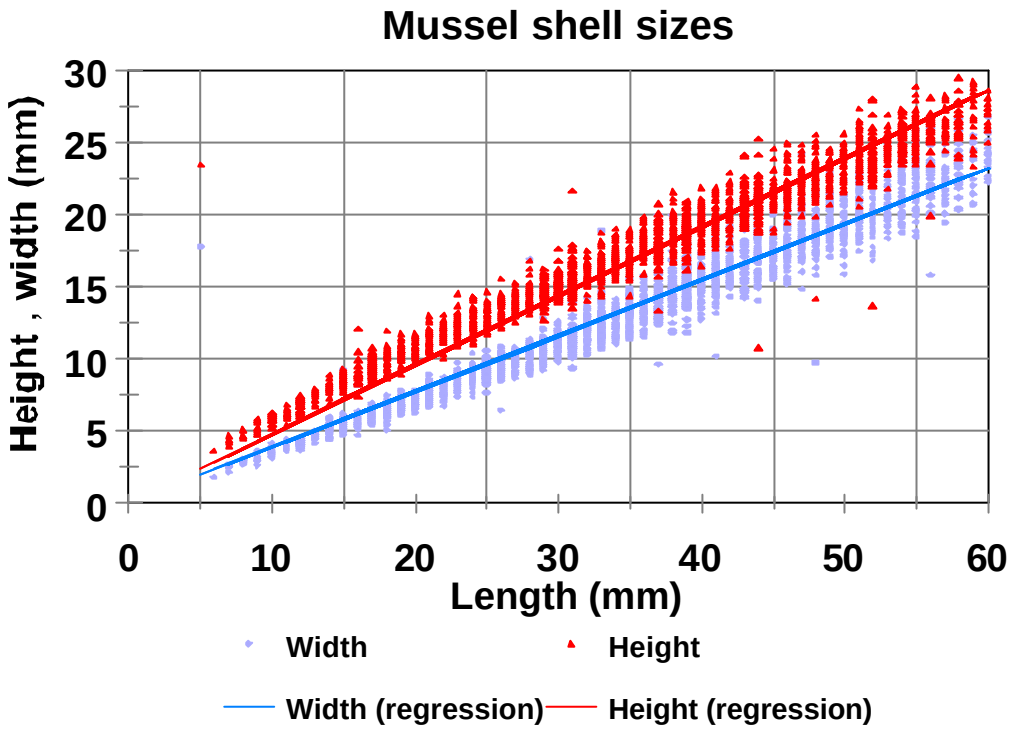


Figure 9. Shell width and height, data from about 2100 sub-tidal shells from the Dutch Wadden Sea. The lines represent linear regression results through (0,0). It is obvious that for small shells, shell height is larger than the regression predicts.

Shell ash free dry mass (flesh content

As an average, mussels contain about 0.05 g AFDW / g (fresh shell). In the allometric equation (11) b is about 2.8, and a=9e-6 to 1.8e-5 (W in gram, L in mm). (Brinkman, 1993). Highest values occur in the late summer, when a 50 mm shell may contain up to 1 g AFDW; lowest values shortly after winter, and just after spawning. A 50 mm shell may contain down to 0.5 g AFDW. Cockles contain somewhat less flesh then mussels do (2/3 as an average).

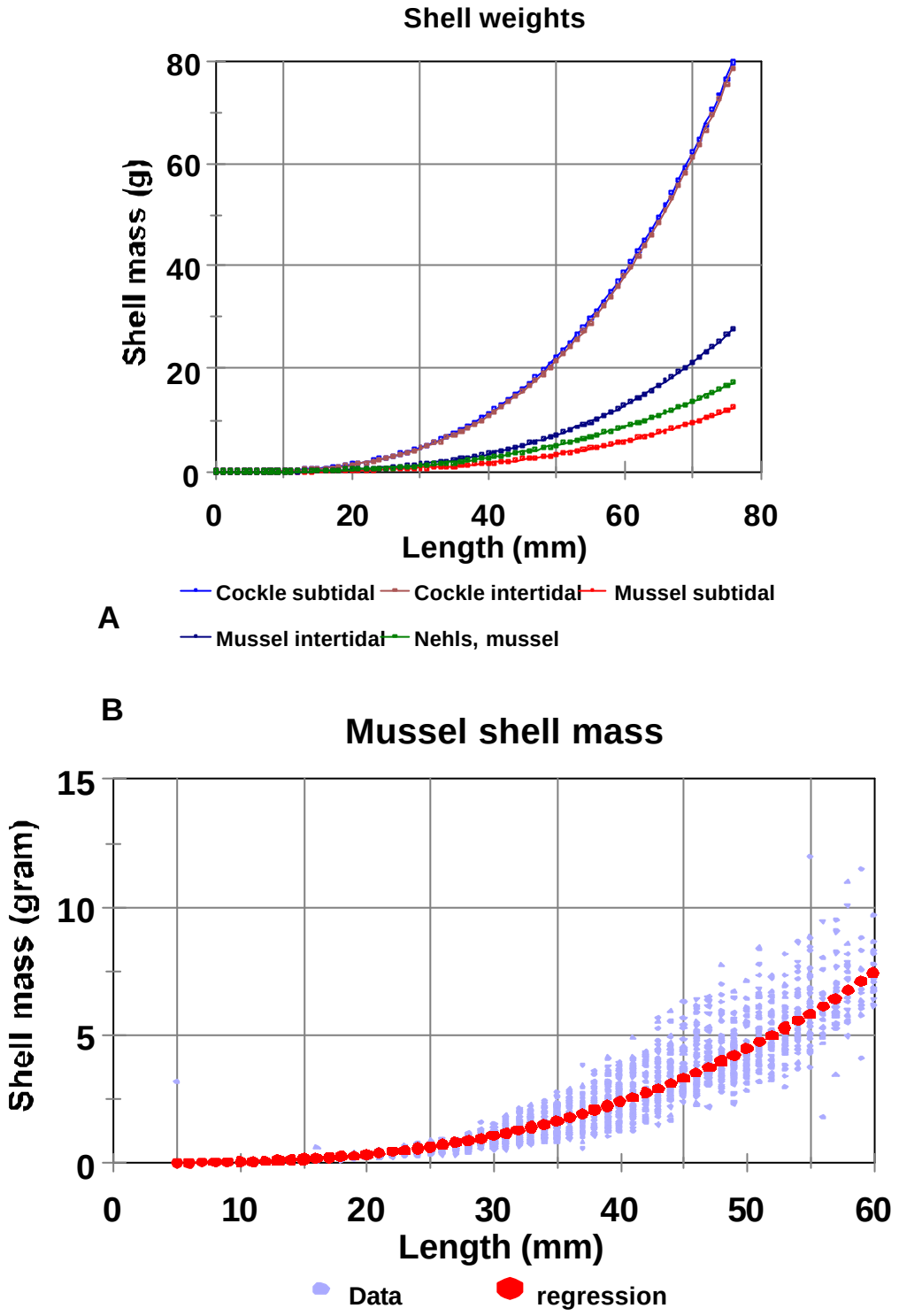


Figure 10. Shell masses for cockles and mussels: A) data from literature, B) data Alterra of 2000 subtidal mussels. Regression line in B (red dots) from mass = 7.82e-5 length^2.8.

Shell mass

Shell mass and shell thickness determine energy costs of breaking a shell. Hamilton et al (1999) related shell thickness (mm) to shell length (mm): thickness = 0.0625 * length$^{0.65}$. The lengths ranged between 10 and 55 mm. The exponent 0.65 (being smaller than 1.0) implies that with increasing length shell masses are somewhat decreasing relative to flesh ash free dry mass. But Hamilton's data show a large variability. Probably it is not easy to distinguish between 0.65 and (i.e.) 1.0 as exponent.

Nehls (1995) mentions shell mass (g) = 5.64 10^{-5} length$^{2.943}$ (length in mm). For a 50 mm shell, mass is about 5.5 g. Dekker (unpublished results) presented us data for cockles and mussels sampled in subtidal and intertidal areas. Cockles have a much larger shell mass than mussels. Dekker's data are well in line with Nehls' relation who did not distinguish between subtidal and intertidal animals (Figure 10). Nehls (1995) also states that per gram AFDW, an eider duck swallows about 7 g mussel shells.

Water content

Besides flesh and shell, mussels consist of water. After Nehls (1995), water content of a shell (g) is:

$$water = 2 \cdot 10^{-4} l^{2.7} \quad \text{(g)} \qquad (12)$$

with shell length l in mm. According to eq.(12), a 50 mm shell contains about 8 g water.

In the model, water content is computed as the difference between fresh mass (see below) and flesh mass plus shell mass.

Fresh masses

Shellfish surveys mostly measure shell fresh masses: water plus shell mass plus flesh (g fresh). Nehls (1995) gives for the total shell mass:

$$total\,shell\,weight = 1.561 0^{-4} len^{2.92} \quad \text{(g)} \qquad (13)$$

For our purpose we are interested in shell meat contents (as ash free dry mass) and in shell mass (g shell). Therefore fresh masses are converted into flesh and into shell masses. As an average for mussels, ash-free dry mass is about 5% of fresh mass (pers. comm. C.J. Smit, Alterra). Flesh masses are about 25% of total fresh masses. This implies that about 75% is shell plus water. A 50 mm shell weighs about 20 g, of which 15 g consist of shell and water. According to the equation for shell mass of Nehls (1995), shell mass of a 50 mm mussel is about 5.5 g and thus accompanying water is about 9.5 g. According to eq.(12), this should be 8 g. So there is a small discrepancy between both estimates.

A mussel with a total fresh mass of 20 g contains 5 g flesh with an ash free dry mass of 1 g. These data are meant as a rule of thumb and vary with season, site and individual.

Breaking force and breaking costs

Hamilton et al (1999) measured the force needed to break a single mussel shell: force (N) = 2.754* length$^{1.08}$, for a shell length ranging from 10 to 55 mm. This implies that with increasing shell mass (mass will increase with area (~length^2) times thickness (~length$^{0.65}$), and thus according to length$^{2.65}$) breaking costs will relatively decrease.

Nehls (1995) gives a linear relationship between the mechanical costs (in kJ) of shell crushing and shell mass: costs (kJ) = 0.643 * Shell mass (g) - 0.317. For a 5.5 g shell costs would be 3.3 kJ. According to this equation eating about 100 mussels of 50 mm per day costs about 300 kJ. In chapter 6 costs are specified more in detail.

Mussel attachment and resistance against removal

Byssus threads attach mussel shells to each other. Hamilton et al (1999) state that the smallest mussels are easiest to detach, the largest ones the most difficult. According to Nehls (1995), mussels on culture lots are the easiest to remove and mussels on tidal mussel beds the most difficult.

Barnacle epibionts

Additional characteristics, such as the presence of barnacle epibionts, may be important. Barnacles increase the size of the shell considerably and therefore decrease the edibility. They also form sharp edges on the shell which makes it more dangerous for an eider duck to swallow. Thus, the profitability of a prey decreases with increasing barnacle overgrowth. Especially on adult mussels, barnacle overgrowth can be substantial (Saier & Buschbaum, 2001; Buschbaum, 2001). In an overview Saier et al (2002) argued that subtidal mussels suffer less from barnacle epibionts than intertidal mussels, mainly due to predation by juvenile starfish and adult green crabs upon juvenile barnacles. Young mussels are capable to clean their own shell within the reach of their foot. From about a size of 20 mm this is no longer possible for the whole shell.

Usually two different species of barnacles may be present. On intertidal mussels, *Semibalanus balanoides* can be found with a size of up to 1.5 cm diameter and about 1 cm heigh. Subtidal mussels have a *Balanus cretatus* overgrowth with a size of up to 2 cm diameter and sometimes of the same height (Campbell, 1976).

An eider duck swallowing a mussel will profit from the slender shape of the shell which is an important reason for prey size selection differences between mussels and cockles. Barnacles add to the width and the height of the mussels. Since a mussel has an average length : width : height ratio of about 6:2:3 (see above), the contribution to width has most effect on the edibility. Even an overgrowth with small barnacles (about 5 mm height) will increase width of a 4 cm mussel from 1.4 to 2.4 cm. This is the same width as a clean mussel with a length of 7.2 cm. Combined with the sharp edges, the extent of barnacle overgrowth in a mussel bed will reduce the prey availability and thus prey selection for ducks drastically. Own data (De Jager, 2002; see Figure 11 and Figure 12) show that it is clear that width is more affected by

barnacle overgrowth than height..

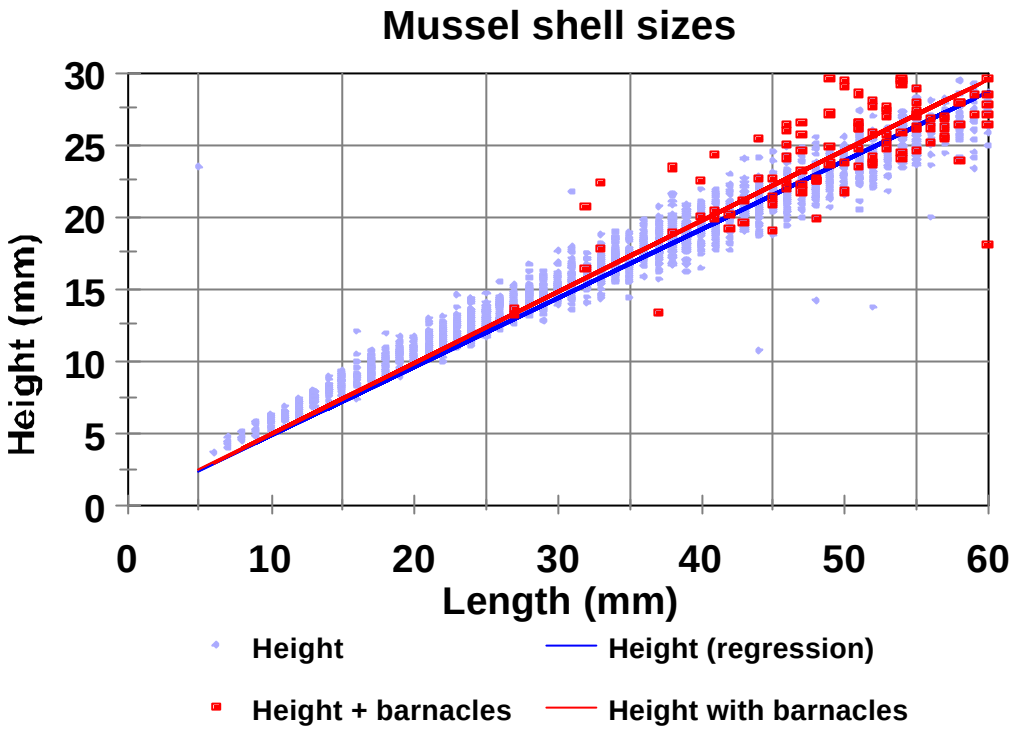


Figure 11. Shell heights when barnacles are present (data Alterra). Lines represent the result of a linear regression.

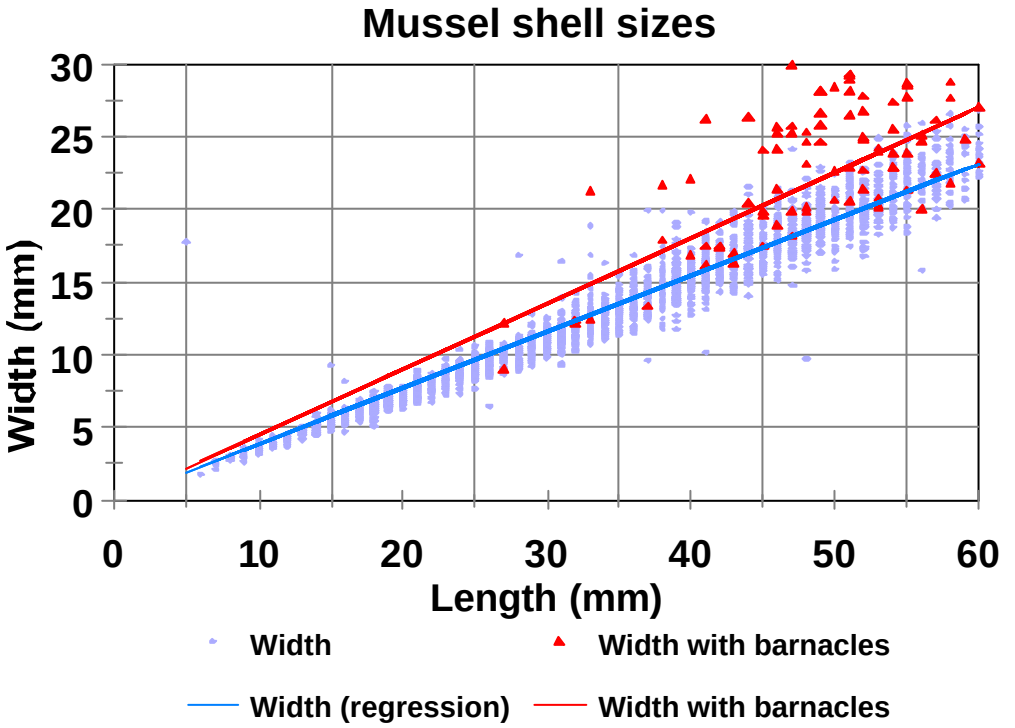


Figure 12. Shell widths when barnacles are present (data Alterra). Lines represent the result of a linear regression.

6 Specification of all energy terms

6.1 Heat loss of an eider duck through conductance

Flux of heat through fat layer and feathers

Because of the temperature difference between an eider duck's body and the environment, there's always loss of heat. Fat and feathers minimize this loss as much as possible. The rate of the heat flow through an isolating layer depends on the thickness of that layer, the temperature difference over that layer (together they make up the temperature gradient), and the heat conductance coefficient (W m^{-1} K^{-1}) of the medium. Figure 13 shows differences between a diving and a swimming bird because of a different feather thickness.

In all cases, the heat flux F reads:

$$\Phi = -a\frac{dT}{dx} \qquad (\text{W m}^{-2}) \qquad (14)$$

with

a = heat conductance coefficient of the medium (fat, air) (W m^{-1} K^{-1})

T = temperature (K)

x = distance (m)

For the heat flow from the duck's body to the air, the flux through the fat (F_{fat}) equals the flux through the feathers (F_{feat}). With dT/dX = ?T/?x (since there is no heat production in fat nor feathers, T decreases linearly with x), one can compute the skin temperature T_{skin}:

$$T_{skin} = \frac{bT_{body} + T_{envir}}{1+b} \qquad (^{o}\text{C}) \qquad (15)$$

with

$$b = \frac{a_{fat}}{a_{feathers}} \cdot \frac{d_{feathers}}{d_{fat}} \qquad (-) \qquad (16)$$

and $d_{feathers}$ and d_{fat} as mean the thickness (m) of the feather layer and the fat layer, respectively. Multiplication with the duck's body area (see below) yields the heat loss per animal:

$$Heat\,loss = \Phi \cdot Surface\,area \qquad (\text{W}) \qquad (17)$$

For the situation of swimming or resting at the shore, $d_{feathers}$ is larger than when diving. The result for the first situation gives the basic heat loss, the result for the

second situation the heat loss during diving for food. The difference between both is the extra heat loss as a result of foraging. In the model, it is accounted for that during foraging, the bird also spends some time above water (handling the prey and recovering from the dive). During that period, the heat loss depends on the heat loss in the water (for that part of the duck that has contact with the water), and the heat loss in the air (for the other part of the duck that is in contact with air).

With T_{body} = 40 ºC, T_{envir} = 5 ºC, $a_{feathers}$ = 2.5e-02 and a_{fat} = 2.5e-01 (W m^{-1} K^{-1}), d_{fat} = 4e-03 m, and $d_{feathers}$ =20e-3 m, it follows that b = 50. Thus, T_{skin} is almost equal to the body temperature (39.3 ºC). During diving, $d_{feathers}$ is about 6e-3 m, and T_{skin} = 37.8 ºC. This implies that in both cases T_{skin} is close to T_{body} and thus isolating properties of the fat layer are not very important as long as the feathers are functioning well. Only if the feather layer is damaged and the fat layer is thin heat loss can become large.. Heat loss while diving in 5 ºC water amounts about 200 W m^{-2}, while in air it amounts about 60 W m^{-2}. More data on $d_{feathers}$ are given in section 5.1.

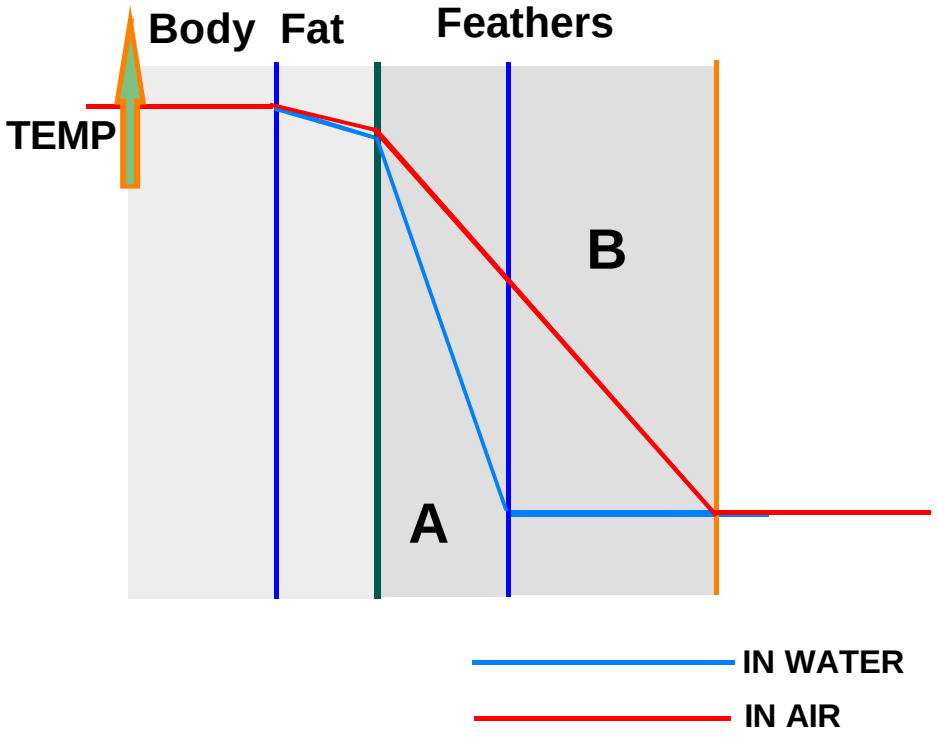


Figure 13. Heat loss through fat and feathers. In case of swimming (B), the feather layer is thicker than in case of diving (A). Feather thickness also determines buoyancy by altering the duck's specific mass, which is 0.7 during diving, and about 0.4-0.5 while swimming.

We validated the equations on some of De Leeuw's measured data (De Leeuw, 1997). His results indicate that a tufted duck (*Aythya fuligula*, female; mass 606 g; body surface 688 cm^2; feather thickness about 2.4 - 3.5 mm, depending on the assumption for lung volumes; all data according to section 5.1) shows a maximum cooling rate of 8e-3 ºC s^{-1} and while diving in water of about 4.7-6.2 ºC. This implies a heat flow of

about 20 W (based on a specific duck heat of 4.2 J g^{-1} K^{-1}, the same as for water is assumed here), and a flux of about 296 W m^{-2}. The calculated temperature difference (?T in eq. (14)) between body and water should then be between 41 - 29 ºC. The real difference ?T is 31-35 ºC (skin temperature about 37.8-39.3 ºC minus water temperature 4.7-6.2 ºC).
This result suggests that the here presented computation method and De Leeuw's measurements are consistent.

Text box 3. Heat transfer approach by De Leeuw (1997).

De Leeuw (1997) used an alternative method to estimate the heat transfer from body to water. He assumed that during a dive there are two major processes going on: heat production by basic metabolism and the heat transfer. During a dive, the heat transfer is the largest of both, resulting in a decrease in body temperature. After the dive, the time to recover is mainly needed to restore the normal body temperature. De Leeuw's method to compute the several contributions is not suitable for our model, but the approach gives us an opportunity to estimate the recovery time. Thus: diving heat is lost and thus, body temperature will decrease. After the dive, the duck will rest until the body temperature is back at the normal level. Since the last process is a first order process depending on (T_{final} - T_{body}), it contains an asymptote (the target temperature Tbody) and one can only use this approach to compare one recovery time with another.

Eqs. (14) - (16) assume optimal isolating properties of the feathers. In reality, heat loss will be larger. Especially during the stay above water we expect an extra heat loss during strong winds and high air humidity. These effects are mentioned under 'wind chill effects', see section 6.13. In that section also heat loss as a result of extra evaporation of feather water is discussed.

A duck has an extra mechanism to reduce its heat loss. It is capable to reduce blood flow in the fat layer region. This complicates the computation from eqs. (14) - (16), but the effect is omitted for the present study. The above outline suggests that the insulating properties of the fat layer are much less important than that of the feathers, as long as the feather layer is not disturbed.

Effect of wind (or: absence of wind)
Without doubt, wind is of importance for the overall heat transfer between duck and air. In the explanation above, it is assumed that the resistance against heat transfer is only determined by the insulating capacity of feathers and fat. Or, in other words, that **at** the feather surface, the temperature exactly equals the environmental temperature. The resistance against heat transfer in the air then is supposed to be zero. This, however, is not reality. Wind speed may be low, and then heat exchange decreases.
Usually this effect is described by a general equation:

$$Exhange = aUWI^{b} \qquad (18)$$

with b=2 as a common value for the exponent (see e.g. other heat budget computations where wind is involved; Thomann & Mueller, 1987). Since the overall heat transfer has to be computed, the effect of wind is a reduced resistance against

transfer, and this should be added to eqs.(15) and (16)(see appendix B). In Figure 14 a computed heat flux as a function of wind speed is shown.

When the surrounding medium is water, this resistance against heat transfer presumable is very low and any effect of water flow may be neglected.

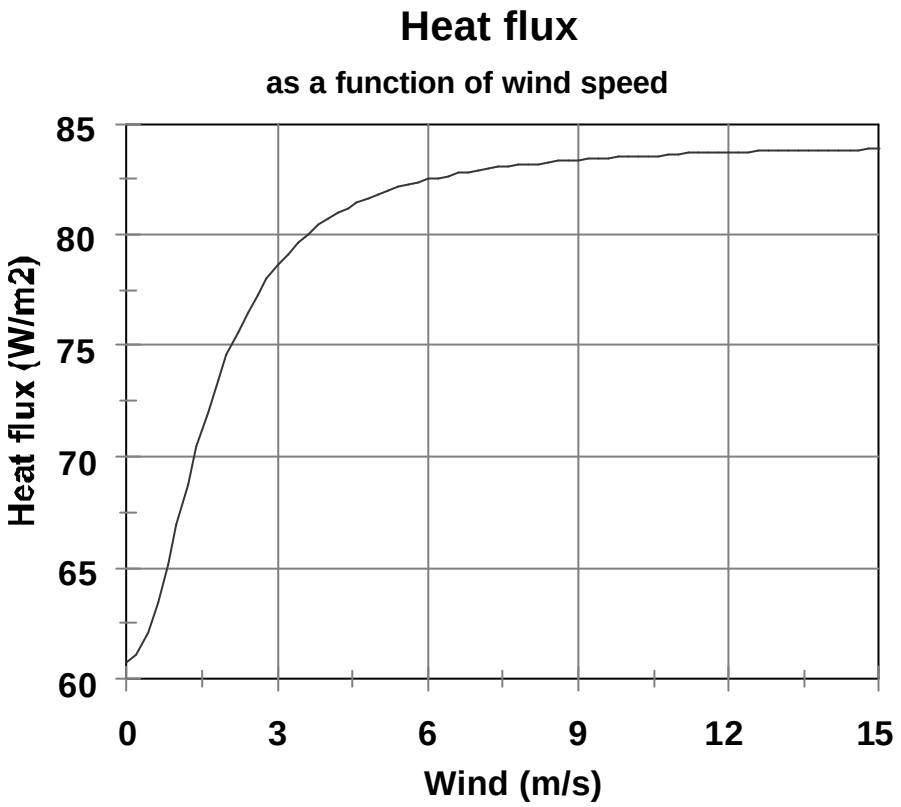


Figure 14. Best guess of wind speed effects on heat transfer, Tair = 5°C. The guess is implemented in the model, but will be subject to improvements. See appendix B for an extended explanation.

Effect of solar radiation and long wave back radiation

In day light a duck receives heat energy from the sun. This solar radiation can be measured, and directly gives the solar energy a duck receives. A fraction of the received energy is lost due tot reflection. During summer, daily average solar radiation is about 200-300 W m^{-2}. For a duck with a body area perpendicular to solar radiation of about 450 cm^2 (about 0.5*total feather area / v2) heat capture is about 6-10 W (over the whole 24 h period). In a normal winter period, this is about 10-40 W m^{-2}, or 0.3 to 1.2 W.

Long wave back radiation follows Stefan-Boltzmann's law (loss ~(?T)^4, with ?T as the temperature difference between feather surface and air). Cloudiness and air humidity are variables in the description.

In the present model, solar radiation and long wave back radiation are not accounted for.

6.2 Heat loss of an animal through breathing

Heating the inhaled air

A breathing bird, inhaling cold air, and exhaling air at body temperature, loses heat. The heat loss reads:

$$Loss_{breath} = Q_{breath} \cdot s \cdot r_{air} \cdot [T_{body} - T_{air}] \quad \text{(W)} \quad (19)$$

Q_{breath} = breath flow ($m^3\ s^{-1}$)
s = specific heat of air ($J\ kg^{-1}\ K^{-1}$)
? = specific mass of air ($kg\ m^{-3}$)
T_{body} = body temperature (K)
T_{air} = air temperature (K)

The breath flow Q_{breath} is related to the metabolic rate. For a certain amount of body mass respired a proportional amount of oxygen is needed. When CH_2O is taken as average body mass composition, respiration of 1 gram body mass (to CO_2 and H_2O) requires 32/30 gram of oxygen. The oxygen content of air is (a) 20% ($a=0.2\ g\ g^{-1}$), and each inhalation only a part (ß) of the oxygen is used. It is assumed that this part is 20% of the available oxygen (ß=0.2). With an energetic content of body mass of T ($J\ g^{-1}$ dry mass), breath flow (Q_{breath}) can be related to respiration Y (W):

$$Q_{breath} = \frac{Y}{\Theta} \cdot \frac{32}{30} \cdot \frac{10^5}{(r_{air} \cdot a \,.\, b)} \quad (m^3\ s^{-1}) \quad (20)$$

In case of resting, only the basic metabolism is relevant. In case of foraging the extra energetic action is relevant. In both cases, the energy expenditure is translated using eq.(20) into a breath flow (Qbreath) .

Effects of evaporation of water (in the lungs

A correction may be necessary to account for the energetic costs concerning the evaporation of water in the lungs. A maximum loss can be computed when one assumes that completely dry air is inhaled and completely saturated air is exhaled. The heat loss Evap (W) amounts

$$Evap = Q_{breath} \cdot \frac{P}{10^5} \cdot \frac{Mol}{V_{mol}} \cdot H_{evap} \quad \text{(W)} \quad (21)$$

where P is the vapour pressure (Pa), and 10^5 denotes the standard air pressure at which 1 Mol water (18 g) has a volume of V_{mol} ($22.4\ 10^{-3}$ m^3), and H_{evap} is the evaporation heat of water ($= 2.26\ 10^3$ J g^{-1}). These data are relatively accurate physical data and eq. (21) gives the maximum energy loss under the conditions as mentioned. Usually, it will be less since the exhaled air will not be completely saturated and the inhaled air will not be completely dry. Figure 15 outlines this relationship.

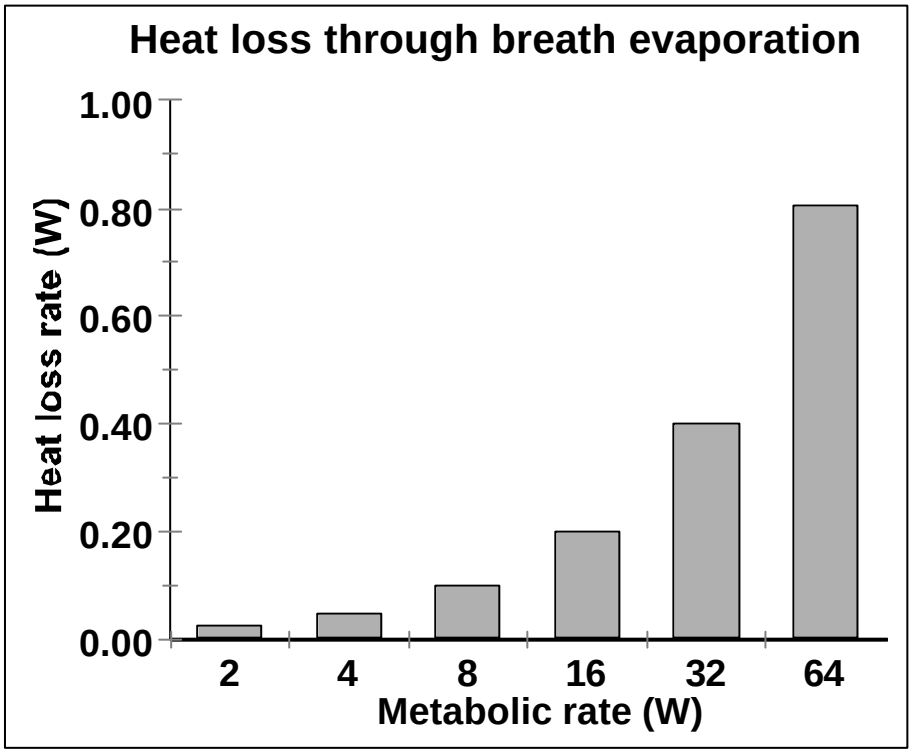


Figure 15. Evaporation losses through breathing, as a function of the metabolic rate. For an eider duck, 8-32 W is a relevant range.

6.3 Heating the prey and losses through defecation

When a prey is swallowed, it will be heated:

$$Heating\_shell = s_w * M_{shell} * (T_{duck} - T_{shell}) \quad \text{(J)} \quad (22)$$

with s_w= specific heat of the shell (J kg^{-1} K^{-1}), M_{shell} the mass of the shell (kg) and (T_{body}-T_{shell}) denotes the temperature difference between eider duck and shell (K). s_w usually will be close to $4.2\ 10^3$, because a large part of a shell consists of water.

For a more precise estimate of the specific heat of a mussel or cockle one can distinguish between the shell, the water and the meat. The specific heat s_w (J kg^{-1} K^{-1}) reads:

$$s_w = \frac{M_{shell} \cdot s_{shell} + M_{water} \cdot s_{water} + M_{flesh} \cdot s_{flesh}}{M_{shell} + M_{water} + M_{flesh}} \quad (\text{J kg}^{-1}\ \text{K}^{-1}) \qquad (23)$$

where M (shell, water, flesh) is the contribution (g) of shell, water and flesh to the total mass, and s (shell, water, flesh) the specific heat of shell, water and flesh. These are 0.9 (an estimate for calcite), 4.2 (water) and 3.5 (an estimate for flesh (De Leeuw, 1997)) (J kg^{-1} K^{-1}), respectively. Since shell mass is about 30%, water mass about 65% and flesh mass about 5% of total mass (see section 5.2), it follows that s_w = 3.2 (J kg^{-1} K^{-1}).

That part of the food that is excreted afterwards transports heat but is not a loss term for the heat budget. The loss is accounted for by eq. (22).

6.4 Crushing shells

The energy costs of crushing shells depend on the thickness of the shells and on the strength of the shells. Generally, one could use a description like:

$$Crushing = a\_crush \cdot ShellLength^{b-crush} \quad (\text{J}) \qquad (24)$$

The b_crush -parameter determines whether there is or there is not a shell size where yield based upon flesh content and crushing costs is maximal. A maximum only exists if b_crush is larger than the corresponding parameter for flesh content. A large value for b_crush also implies that shells become relatively thicker with age or with length. In case b_crush is smaller than the allometric parameter for flesh content, the profit for a duck will always increase with increasing shell size.

Mussels on tidal flats usually have a relatively thick shell, which is often ascribed to the poorer growing conditions on tidal flats and to the observation that shells of the same age have more or less the same shell thickness. Sometimes, data on shell crushing costs is not presented in terms of energy (J), but in terms of force (N).

Nehls (1995) computed crushing costs from the difference between nett food yield and the flesh yield (nett yield = flesh yield - crushing costs {heating the prey is taken into account}).
The nett yield is calculated as ln(E nett) = 3.737 ln (Length) - 12.815, and the flesh yield (22.5 kJ g^{-1}) follows from the flesh content of the prey: ln(5*flesh) = 2.919 ln (Length) - 8.764. The term 5* flesh is because Nehls uses fresh flesh mass, and not AFDW. With the relation between shell mass and length ln(Shell_Mass) = 2.943 ln (Length) - 9.783, Nehls was able to relate the energy needed for crushing shells to shell mass (g):

$$Crushing = 0.643 Shell\_Mass - 0.317 \quad (\text{J}) \qquad (25)$$

Nehls' results show that the nett energy yield increases with length$^{3.737}$, and the gross yield with length$^{2.919}$. Consequently, Nehls concludes that crushing costs increase slower with shell size than yields.

Nehls' relation is valid for subtidal mussels. This relation can be converted so that crushing costs are related to shell length: crushing costs (kJ/shell) = $-0.317 + 3.63\ 10^{-5}$ length$^{2.919}$.

Others present data on breaking force in stead of energetic costs. Piersma et al (1993) found: Force (N)=0.0101 Length$^{2.994}$ for *Cerasteroderma edule*, =0.0018 Length$^{3.527}$ for *Macoma balthica*, and =0.0446 Length$^{2.351}$ for *Mytilus edulis*. If correct, these data would give information on relative crushing energies. However, the formulas Piersma et al present suffer from an error that is often made when estimating parameters for allometric equations (Brinkman, 1993). Although the result of the equation gives a reliable estimate for the dependent variable (in this case the breaking force), the separate allometric parameters (the constant and the exponent) are not reliable at all, and show a large banana-shaped combined uncertainty region. This can already be observed when comparing the three constants, and the three exponents. The exponents vary between 2.5 for *Mytilus* and 3.5 for *Macoma* and the constants show a reverse picture: 0.04 for *Mytilus* and 0.002 for *Macoma*. A low value for the exponent is compensated by a large constant, and vice versa. One of the causes may be that most of the data that have been used by Piersma et al (1993) for the parameter estimation cover only the smaller sizes. Larger shells were not included. Parameter estimation on allometric relationships only produces reliable parameters if the data cover a broad size spectrum.

If only the result (the breaking force in this case) is needed, then the very limited reliability of the parameters is *not* a major draw-back. However, if, and this is what Nehls did, the costs of shell crushing are computed by combining two allometric equations, then the reliability of the allometric exponents is crucial.

So, it's not unlikely that Nehls (1995) equation has the same kind of error as the equations of Piersma et al (1993).

The translation of force (N) into crushing costs (J) is impossible to make without experimental data telling us how much work a bird has to do in order to crush shells of a certain size. The observations of work=f(shell size) can be applied to scale the amount of energy needed when crushing shells of different sizes. Thus, we need data like Nehls (1995) presented plus comparisons like the data Piersma et al (1993) presented.

Although the interpretation of the data mentioned is questionable, we tried to deduce a relationship for mussels and cockles based on both data sets. Assuming that Nehls' equation is correct, and the breaking costs are proportional to breaking forces, then a factor of about $7\ 10^{-3}$ kJ N^{-1} would be a first estimate to convert crushing costs in crushing force:

$$crush\_costs = 0.007 \cdot force \quad \text{(J)} \quad (26)$$

with force according to Piersma et al (1993) (see above). In Figure 17, crushing costs related to shell mass, and in Figure 16 crushing costs related to shell lengths are given. For the allometric line in figure 16, a= 3.12 10^{-4} and b= 2.35. The value for a is found with the equation of Piersma et al for breaking force, and eq. (26) is used for the conversion into J.

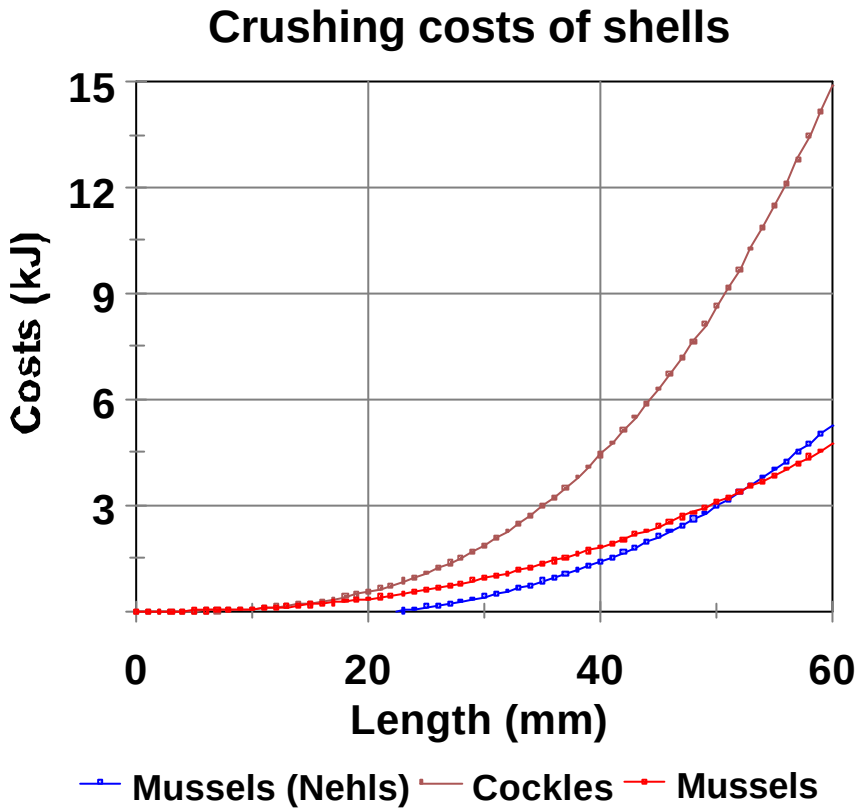


Figure 16. Costs of crushing shells as a function of shell length, according to Nehls (1995) and Piersma et al (1993) (estimated after conversion).

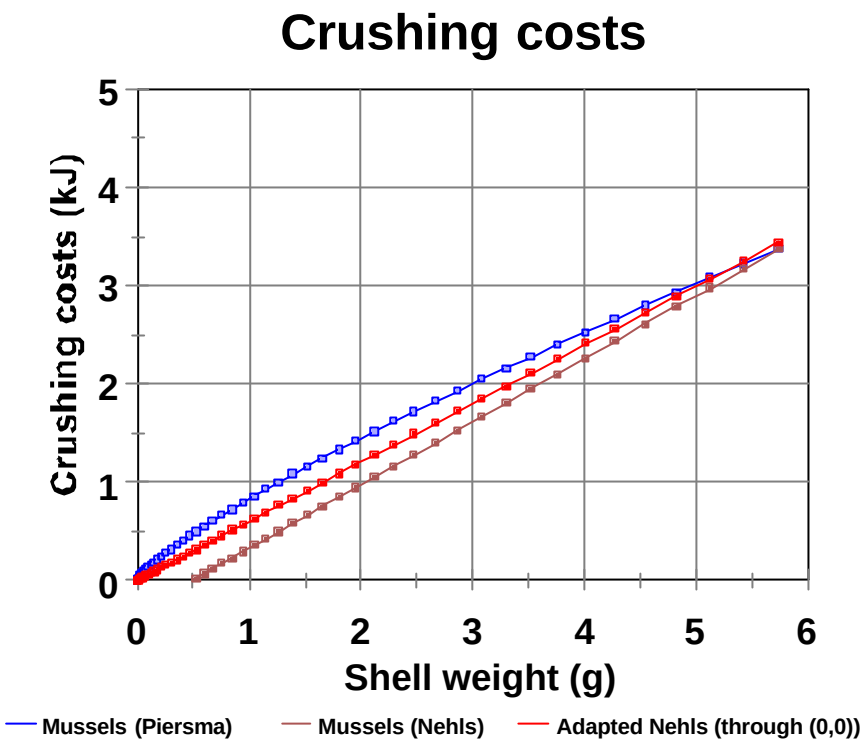


Figure 17. Costs of crushing shells as a function of shell mass, according to Nehls (1995) and Piersma et al (1993) (estimated after conversion).

6.5 Costs of digestion

The costs of digestion only concern the costs the eider duck has to make in order to make the food suitable for assimilation. It is a function of the amount of meat, of the quality of the food, and the passage time of stomach and guts. Since we are regarding average passage times, *and* average meat quality, the last two arguments are not considered in the rest of the paper.

Nehls (1995) gives as an estimate that about 14% of the energy equivalent of food is needed for the digestive costs. Thus, in

$$Digestive\_costs = g \cdot food\_energy\_uptake \quad \text{(J)} \quad (27)$$

the parameter ? = 0.14.

On average food_energy_uptake is about 22.5 kJ g^{-1} AFDW (see section 6.15).

More in detail, Nehls gives an equation of digestive costs related to food_uptake in g (AFDW):

$$Digestive\_costs = 3.874 \cdot food\_uptake - 7.113 \quad \text{(J)} \quad (28)$$

Digestive costs are chemical costs, i.e., the bird has to produce enzymes and other metabolically active substances. There is no (or hardly any) heat production involved. Thus, the costs appear on the energy budget, not on the heat budget. As an example, for a food_uptake of 10 g AFDW (equivalent of 220 kJ), digestive costs are 31.6 kJ. This equals 0.143 times the food_energy_uptake, as described by eq. (27). Nehls' data also allow an equation like eq. (28) without a constant. In that case, the proportionality factor is 3.15 (kJ g^{-1} AFDW-uptake). Food is used for storage. This appears as biomass on the budget, and is converted into energy (kinetic energy, heat) later on. The result of that process is loss of biomass.

6.6 Costs of diving

Acceleration

When a bird dives, it first has to accelerate, swim to the bottom of the water system, and stay at the bottom until a suitable prey is found and torn loose. The eider duck does not dive with a constant speed, but there are some variations that make up part of the energy losses.

Acceleration costs are very straightforward: the bird (with mass m) gets a kinetic energy

$$E_{kin} = \frac{1}{2} m v^2 \quad \text{(J)} \quad (29)$$

If, during a dive, a bird varies it's diving speed between $v_{min} < v < v_{max}$ (m s^{-1}), with a frequency of fr_{div} (s^{-1}), each time the bird accelerates from v_{min} to v_{max} the energy input is 0.5 * m * (v_{max}^2 - v_{min}^2). The energy loss that occurs when the speed decreases from v_{max} to v_{min} is dissipated into the environment as heat. Thus, the extra costs as a result of the not-constant speed is:

$$E_{extraaccel} = fr_{div} \cdot 0.5 m (v_{max}^2 - v_{min}^2) * duration_{dive} \quad \text{(J)} \quad (30)$$

with duration of the dive in (s). At the end of section 6.6 it is explained that the frequency fr_{div} depends on diving force (generated by the duck), upward force (as a result of buoyancy), drag forces during downward and upward movement and the value of v_{max} and v_{min}.

Drag

When diving, the bird encounters a drag force (or friction), which consists of a form drag related to the size and the shape of the duck's body, and a viscosity drag, related to the duck's surface area. Generally, the friction force reads:

$$F_f = AK\,f \qquad \text{(N)} \qquad (31)$$

with A as the object area (m^2) (perpendicular to the flow direction), K as the characteristic kinetic energy of the object per unit volume (J m^{-3}), and f as the friction factor (-). Eq. (31) is a definition for f. Usually, not F_f is measured, but the final falling velocity of an object.
In fluid mechanic science, friction forces for many objects and many flow velocities have been measured.
The formula to be used depends on the Reynolds number:

$$\mathrm{Re} = \frac{\rho_{fluid} v D}{\eta} \qquad \text{(-)} \qquad (32)$$

with

ρ_{fluid}	=	specific mass of the fluid (kg m^{-3})
v	=	velocity of the duck relative to the fluid (m s^{-1})
D	=	specific size of the object, usually the diameter of the object perpendicular to the flow direction (m)
η	=	viscosity of the fluid (kg m^{-1} s^{-1}) (often as centipoises, 1 kg m^{-1} s^{-1} = 1000 cPoise)

Since $\rho \sim 10^3$ (kg m^{-3}), $\eta \sim 10^{-3}$ (kg m^{-1} s^{-1}), v~ 1 m s^{-1} and D~ 0.15 m, Re~ $1.5\ 10^5$. For $\mathrm{Re} > 2\ 10^3$, and thus also in our case, the friction factor f from eq. (31) reads:

$$f = 0.44 \qquad \text{(-)} \qquad (33)$$

In case of a sphere, and thus, when the speed relative to the fluid = v (the diving speed), the friction force is:

$$F_f = 0.44\left(\frac{1}{2}\rho_{water} v^2\right)(0.5\pi D^2) \qquad \text{(N)} \qquad (34)$$

The energy needed to reach the bottom is (force times distance):

$$E_{diving} = F_f \cdot depth \qquad \text{(J)} \qquad (35)$$

The power needed reads:

$$W_{drag} = F_j \cdot v_{diving} \qquad \text{(W)} \qquad (36)$$

Because the shape of a duck is not spherical, the proportionality (or friction) factor f will differ somewhat from 0.44. With eq (35), the mechanical energy needed to reach the bottom is known. The frontal area ($0.5BD^2$) of the duck is described in section 5 (Figure 5).

Lovvorn et al (1991) measured F_f for ducks by gently sealing frozen ducks, pulling these through a fluid and measuring the force acting on the duck. They found a relationship $F_{fr} = -0.946 + 0.826M + 0.614\ v + 0.825\ v^2$ (N), where M is the bird's

mass (kg), and v the diving speed (m s^{-1}). Unfortunately, they did not document in what range the formula is valid. Their equation yields a positive drag at v=0. This should however be 0. In Figure 20, F_{fr} according to eq.(34) and F_{fr} according to Lovvorn's equation are given. We also inserted a slightly altered version of Lovvorn's equation where $F_{fr\ |v=0}$=0. From Figure 20 it is clear that (34) and Lovvorn's values are of the same order. We consider that as support for both approaches. For our purpose, a most appropriate value for the friction factor f turned out to be 0.10 to 0.15.

Buoyancy

The duck has to overcome the buoyant forces. Because of the difference between the specific mass Δ (kg m^{-3}) of the fluid and that of the duck, the buoyant force reads:

$$F_{bouy}=g(r_{water}-r_{duck})V_{duck} \quad \text{(N)} \qquad (37)$$

with
g = gravity acceleration (m s^{-2})
V_{duck} = volume of the duck (including feathers) (m^3)

So, the potential energy added to the duck as a result of diving is:

$$E_{bouyancy}=F_{bouy}\cdot depth \quad \text{(J)} \qquad (38)$$

Adding eqs. (29), (30), (35) and (38) gives the total energy needed for a dive.
The power needed to overcome buoyancy reads:

$$W_{buoy}=F_{buoy}\cdot v_{diving} \quad \text{(W)} \qquad (39)$$

Pressure has a significant effect on buoyancy; see the section on pressure effects below.

Efficiency

An eider duck is not a 100% efficient diver. It dives by 1) the acceleration at the beginning of the dive, and 2) by using its feet producing a force directed downwards. The duck produces heat because not all the energy is transformed into flipper movement (efficiency η_1), and secondly, not all the flipper movement is transformed into a force directed downward (efficiency η_2). Probably, the losses are larger or even much larger than the energy strictly needed for the dive.

So, the total amount of energy needed, is:

$$E_{diving\_real}=\frac{(E_{diving}+E_{buoyancy})}{\eta_1\cdot\eta_2} \quad \text{(J)} \qquad (40)$$

The heat production inside the duck's body amounts:

$$E_{heat\_body} = (1 - e_1) \cdot E_{diving\_real} \quad \text{(J)} \quad (41)$$

The heat loss to the water reads:

$$E_{heat\_water} = (1 - e_2) \cdot e_1 \cdot E_{diving\_real} \quad \text{(J)} \quad (42)$$

The last variable does not appear in the model. What does appear is the energy needed (eq (40)) and the heat production in the body, which adds to the heat budget.

Text box 4. Estimation of diving efficiency of tufted ducks. Measured data after De Leeuw (1997). Efficiencies that are used, are computed. See text for further explanation..

Mass of tufted duck	606	g	
Diving metabolic rate (DMR)	11	W	measured
Basic metabolic rate BMR	2.97	W	measured
DMR - BMR	8.03	W	computed
Needed for drag	0.7	W	estimated by De Leeuw
Needed for buoyancy	2.5	W	estimated by De Leeuw
Drag+Buoyancy	3.2	W	estimations added
DMR-BMR-(Drag+Buoyancy)	4.83	W	losses
Needed for drag	0.23	W	equation
Needed for buoyancy	1.79	W	equation
Needed for acceleration	0.79	W	equation
Drag+Buoyancy+Acceleration	2.81	W	equations added
Efficiency (measurement)	0.40	(-)	using estimations by De Leeuw
Efficiency (computed)	0.35	(-)	using estimations according to equations

To give an estimate of efficiency, measured costs have to be compared with theoretical costs. Up to this moment, we only used data of De Leeuw (1997) for tufted ducks (text box 4). With his data we estimated an efficiency of about 35-40%. Thus, about 60-65% of the energy needed, is converted into heat instead of movement. We assume that this part adds to the heat content of the bird, which is not entirely true because also a part of the heat is lost in the water (eq.(42)). No estimates could be found on that part yet.

Pressure effects

When diving, the buoyancy is affected by the compression of the air between the feathers and in the lungs of an eider duck. Using Boyle-Gay-Lussac's law pV=RT (the necessary corrections are omitted since the deviations from the standard

situation (1 atm) are not that large) it follows that the air volume between feathers is related to depth:

$$V_{air} = V_{air}(0).\frac{p(0)}{p(depth)} = V_{air}(0).\frac{10^5}{(10^5 + g \cdot r_{water} \cdot depth)} \qquad (m^3) \qquad (43)$$

The specific mass of the duck including air thus becomes:

$$r_{duck} = \frac{r_{air} \cdot V_{air} + r_{body} \cdot V_{body}}{V_{air} + V_{body}} \qquad (kg\ m^{-3}) \qquad (44)$$

Substituting equations 43 and 44 in eq 37 gives the relationship between buoyancy and depth. For a very shallow water body this effect is not relevant, but at a depth of about 3 m, a common situation in the Dutch Wadden Sea, buoyancy is reduced with about 30%. The energy needed to overcome buoyancy (Figure 21) therefore decreases.

In appendix C it is explained how the effect of changing buoyancy during descent is handled in the model.

Staying at the bottom

After the dive, the bird has to stay at the bottom for a while, searching for a prey or handling the prey. Drag forces will be smaller than during the dive, but the buoyant forces still have to be compensated. There is no potential energy added to the duck. All the energy dissipates partly inside the duck, and partly in the water. This implies that from eq. (40) some terms disappear, and some remain. Lovvorn et al (1991) estimate staying costs of 1.7 - 2.6 W kg^{-1} from the computed upward movement between two strokes of the duck's feet and the energy needed to get back to the bottom position. Since the efficiency of this process determines the energy needed, we applied our estimate of efficiency from De Leeuw's data (see text box 4). The power needed to overcome upward forces (buoyancy) and the power needed for the ongoing acceleration is quite low. A tufted duck requires about 8 (W) for the 'normal' diving process, including an efficiency of about 35%. For the stay at the bottom normal diving power (8 W) minus about 2.8 W (used to overcome drag, buoyancy and acceleration) = 6.2 W is needed. This is a maximum estimate. Another estimate is to use relative amounts: (8-2.8)/8 * 6.2 = 4 W.

Lowworn et al (1991) computed 1.7 W kg^{-1} for lesser scaups (817 g body mass) and 2.5 W kg^{-1} for redheads and canvasbacks (respectively 1013 and 1275 g body mass). Including 35% efficiency, we would arrive at 3.9 W for lesser scaups, and 9 W for canvasbacks as power requirement to stay at the bottom. For an eider duck of 2000 g body mass, we calculate a power requirement of $5/0.35 \sim 14$ W.

In order to come to a more precise estimate, we simulated the movements of an eider duck at the bottom of a water system, at two depths: 3 m and 5 m. The buoyancy differences are accounted for and the diving speed -with average 0 m s^{-1}- ranges between +0.1 and -0.1 m s^{-1}. The stroke frequency (measured by De Leeuw (1997) for tufted ducks as 3.06 s^{-1}, given by Lovvorn et al, 1991 as 2.8 s^{-1} for

canvasbacks, 3.3 s^{-1} for redheads and 3.6 s^{-1} for lesser scaups) is chosen to be 3 s^{-1}. An eider ducks stroke frequency is about 2.6 s^{-1}. Power requirement is not very sensitive to different values of stroke frequency.

For eider ducks, we calculate a force to stay at the bottom of about 7.2 N during the period of active strokes, and 5.6 N as an average for the whole period including the periods between two strokes. At 3 m depth, these data are 8.05N and 6.47 N for active force and average, respectively. These forces include the forces needed to overcome buoyancy, acceleration and drag forces. These last two are small since the speed is close to zero, and the speed amplitude is 0.1 m s^{-1} . With increasing amplitude, acceleration and, to a certain extent, also drag increases.

In Figure 18, velocities, acceleration and buoyancy is sketched for an eider duck staying at the bottom, 3 m deep.

When diving, drag forces are added (Figure 19). The force during strokes is about 10.3 N, and the average is 8.56 N. Acceleration (variations in diving speed) costs are about 0.4 W, which equals 0.4 N when speed is 1.0 m s^{-1}. Especially these acceleration costs depend on the assumption of minimum and maximum velocities. Compared to buoyancy costs, acceleration is less important, although it cannot be ignored.

The computed costs are real costs. Regarding a 35-40% efficiency, including the conversion from power (watts) needed to produce force (N), the diving will cost an eider duck about 21-24 W, and staying at the bottom about 14-16 W. Acceleration costs are 1.2 W for a diving duck (mean velocity =1.0 m s^{-1}, amplitude = 0.1 m s^{-1}) or 2.4 W if the amplitude is 0.2 m s^{-1}. At the bottom, acceleration costs are 0.2 W if the amplitude = 0.2 m s^{-1} and 0.06 W if this is 0.1 m s^{-1}.

Work against buoyancy depends on depth. The duck also has to perform work for the local variations in velocity (the duck does not stay constantly at the same vertical position, but there is a small variation).

Still, it is difficult to transform forces -needed to counteract upward buoyant forces- to power needed to produce that force. One may argue, and that's what we are doing, that when diving a duck has to produce a power of 11-12 W (force times speed or {buoyancy+drag+acceleration} times 1 m s^{-1} with about 2.8 W needed for drag and 0.4 for acceleration). Because of efficiency loss it costs an eider duck about 100/40 times as much energy, thus about 27 W. At the sediment, speed is zero, and only 16 W is needed. This is in case of zero m water depth and a buoyancy of 8.7 N. Thus, an estimate for the energy needed to stay at the bottom is 2 W $N^{-1.}$ This figure is used in the model.

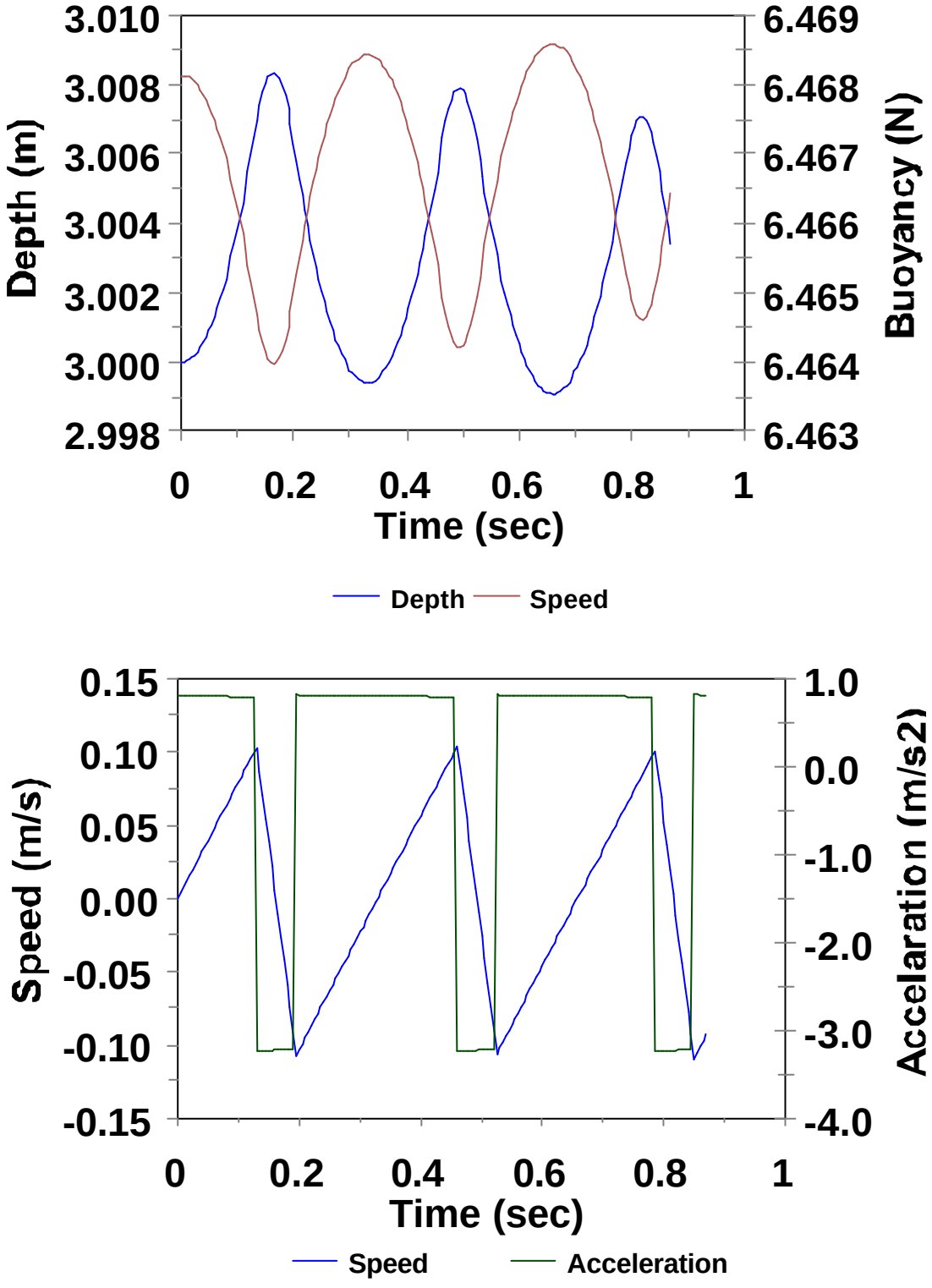


Figure 18. Velocity, buoyancy, and acceleration of an eider duck when foraging at the sediment surface (depth= 3m).

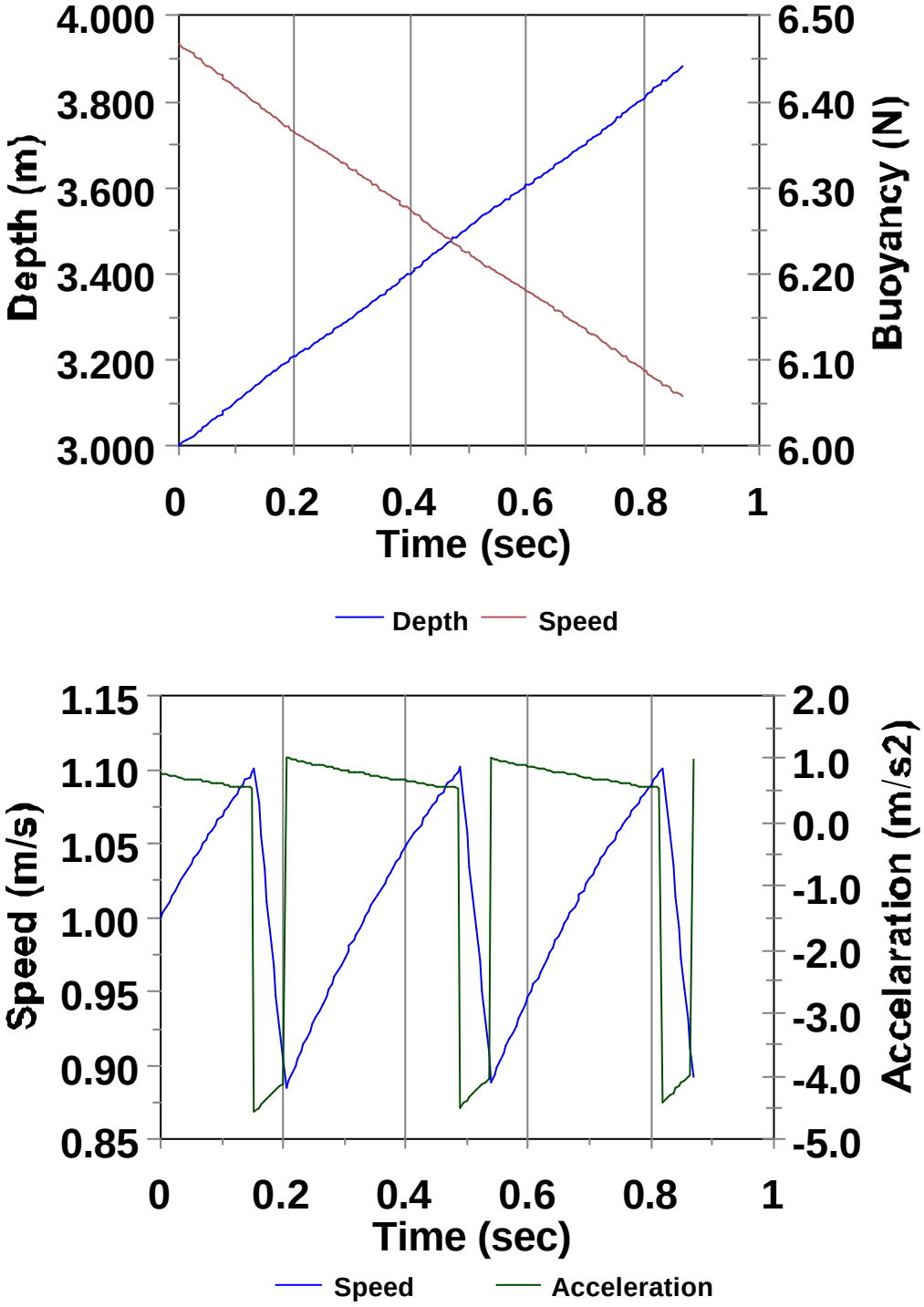


Figure 19. Velocity, buoyancy, and acceleration of an eider duck when diving, starting at 3 m depth.

Getting back to the surface

After the dive, the duck will float; the upward speed depends on the buoyant forces and the opposing drag forces. The approximate vertical speed of the duck can be computed since the buoyant force F_{buoy} (eq. (37)) equals the friction force F_f (eq. (34)), and thus:

$$v_{float} = \sqrt{\frac{g(r_{water} - r_{duck})V_{duck}}{f(\frac{1}{2}r_{water})(0.5p\,D^2)}} \qquad (\text{m s}^{-1}) \qquad (46)$$

The friction factor f equals 0.44 for a sphere, but is about 0.15 for an eider duck (Figure 20 and §6.6).

In Figure 21, also v_{float} is shown, including pressure effects on buoyancy, and assuming 30% air volume at the beginning of a dive. These results show that the time needed to get back to the surface is 30-50% shorter than the time needed to dive . Floating is without energetic demands, the potential energy of the bird dissipates during floating, and adds to the water temperature. Thus, it does not appear in the heat budget of the duck.

The velocities found in this way are higher than data from literature. De Leeuw's data on ascent time for a tufted duck show an upward velocity of about 0.9-0.95 m s^{-1}, data for an eider duck are not found in literature (so far). The difference between computation and observation can be explained by the way tufted ducks perform their ascent.

De Leeuw (1997) monitored diving tufted ducks during their descent and ascent. He clearly showed that a diving tufted duck has a much more hydrodynamic shape than a duck that is ascending as a result of buoyancy alone. The frontal area of a descending duck is smaller than of an ascending duck. From De Leeuw's photographs we estimated a 1.5 times larger frontal area for an ascending duck. The resulting upward velocity for a tufted duck (mass: 600 g, frontal area: 128 cm^2) becomes about 1.1 m s^{-1}. But, De Leeuw's photographs also show that the ducks exhale some air prior to rising. Assuming a buoyancy loss of about 15% as a result of such a loss of air (from 0.7 specific mass to 0.75 kg dm^{-3}), the speed of ascent decreases from 1.15 to 0.98 m s^{-1}. By these corrections, the computed upward velocity equals the measured velocity of tufted ducks. A typical computed eider duck upward velocity is about 1.3 m s^{1} assuming similar exhalation characteristics and changing frontal areas.

Effects of environmental temperature

Temperature (T) plays a minor role in the computation of diving costs. Viscosity of water changes from 1.78 cp at 0 °C to 1.3 cp at 10 °C and 1.0 cp at 20 °C (1 centipoise = 10^{-3} N s m^{-2}). But diving drag is mainly determined by form drag instead of viscosity drag, and therefore, effects are smaller.

Specific mass of air is about linearly related with T^{-1}, with T in K. From 0 °C to 20 °C (the normal range of water temperatures), the relative change is about 7%. But, because mainly the difference between specific mass of air (changing from small to even somewhat smaller with increasing temperature) and of water (a much larger density, which is constant within about 0.2% from 4 °C to 20 °C) is used, the effect will be much smaller, and thus ignorable.

6.7 Swimming

When a bird is swimming, it encounters a drag as a result of its speed. Since only a part of the body is in the water, the plane perpendicular to the swimming direction (the effective frontal area) is (much) smaller than while diving. On the other hand, at the water surface, surface tension effects add to the forces.

Baudinette and Gill (1985, cited by Wilson et al, 1992) presented an empirical formula:

$$Power = a \cdot F_{swim} - C \quad \text{(W)} \qquad (46)$$

with a as a proportionality factor (0.74 W N^{1} for ducks), and C as an empirical constant (0.08 W). Since power equals speed * force, this proportionality constant should be related to speed divided by efficiency. If we assume an efficiency factor of 0.35 to 0.40 (the same as for diving), then average swimming speed is about 0.3 m s^{-1} , which is not an unlikely average speed (pers. observation).

How the friction force F_{swim} is found, is not mentioned by Wilson et al, but if we assume that this is closely related to the friction force under water (while diving), then about 0.8 N would be our first estimate. Thus, swimming power would be about 0.6 W.

6.8 Flying

Flying is considered as very costly in terms of energy. Lovvorn and Jones (1994) mention an energy expenditure of 300 W. With 22.5 kJ per gram AFDW as an estimate for energetic content of body tissue, 300 W is equivalent to about 300/22.5e3= 1.3e-2 g AFDW of body mass per second. Per day, this amounts about 1100 g AFDW, or (with 20% AFDW to total mass) 5 kg fresh mass per day. Flight speed of an eider duck is about 24 m s^{-1} (= 85 km h^{-1}; Lovvorn & Jones, 1984), which implies that during one day, the flight span would be about 2000 km. It is impossible for an eider duck to fly such a distance without reducing the costs (by V-shape flying, for example) or without substantial extra foraging during the trip. Assuming an existing maximum reserve of about 1000 g fresh mass (which is a high value), thus about 200 g AFDW, a flight distance of about 360 km for an eider duck would be possible.

In the present study, flying costs are not crucial, since we only investigated differences between regions and sites, and not a decision strategy of a bird whether to fly to another area or to stay in the present one. We just assumed a small contribution of flying costs to the overall energy budget. During flying, a part of the produced energy will be a loss term that contributes to the heat budget. This loss is estimated to be 60% (same as for energy needed for diving). So, when flying 60 % of 300 W is heat loss.

6.9 Resting time or recovery

After a dive, a duck needs some time to recover. Probably recovery time is needed to restore the body temperature (De Leeuw, 1997). Or, in other words, during the dive, the heat loss is larger than the production, and thus, it needs some time to restore the internal heat content. Therefore, the resting (recovery) time can be deduced from the shortage of heat during the dive, and the excess of heat production afterwards. As soon as the loss has been compensated, the resting time is over.
For the present study, we did not work out this approach, but we kept ourselves to the observations by Nehls (1995) that in the summer period, the resting time after a dive was about 12 s, in winter about 15 s. Related to the duration of the dive, resting time can be described as:

$$resting\_time = 1.3 \cdot diving\_time \quad \text{(s)} \quad 47)$$

This relationship is implemented in the model, and is used to compute the total time needed for prey searching and handling.

De Leeuw (1997) measured time budgets for diving tufted ducks (*Aythya fuligula*), and found that resting_time was about 0.9 times diving_time at intermediate water temperatures. Surface time increased only slightly with diving depth. So, may be we overestimated resting_time.

6.10 Handling time at bottom

During the stay at the sediment surface, the duck has to find a shell and pull it loose. Probably, finding a mussel is not very time consuming. On mussel beds or culture lots mussels are present in high densities. If the mussels are loose, it will cost the duck little time to remove a shell. This will be the case on culture lots. On wild mussel beds, the duck needs more time. De Leeuw constructed a time budget for tufted ducks, and found 7 s foraging time in 1 m deep water and about 12-13 s foraging time in 3 and 5 m deep water, both excluding diving time. Nehls (1995) found average foraging times for Wadden Sea eider ducks of about 16.8 s in summer and 19.5 s in winter periods, both including diving time. Because the time needed for descent and ascent together will be in the order of 5-8 s for 3-5 m deep water, true foraging time for eider ducks is of the same magnitude as for tufted ducks.
A distinction between culture lot sites and natural mussel beds is not given by Nehls.

Nehls (1995) also found that when an eider duck could not find a suitable mussel, the diving was interrupted. Interrupted diving times were on average 22 s instead of 17 s.

6.11 Handling time above water

When a prey is found, it may have to be torn loose. For mussels from culture lots, this is hardly a problem. They are rather loose, and it does not take much time for a duck to handle them. In that case, the handling time of loose mussels is (Nehls 1995):

$$hand\_tim = 0.42 + 0.006len + 0.001len^2 \quad \text{(s)} \quad (48)$$

with len = length of the mussels in mm. The formula is completely empirical, and derived from field observations on eider ducks eating mussels that were offered in a tray.

In case the mussels originate from culture lots, a clutch of mussels is released, transported to the surface, and one mussel (or sometimes more than one) is loosened by a duck by shaking the clutch vigorously. Nehls (1995) gives an average handling time in summer of 12.8 s, and 16.1 s in winter.

6.12 Dabbling

Dabbling is relevant for ducks feeding on cockles in very shallow water. Since this has not been subject of our investigation, we left this topic out of the study.

6.13 Salt excretion

As far as we know, Nehls (1995) is the only one who took into account that mussels have to excrete the extra salt that they take up during feeding. The amount of salt is directly related to the amount of water in the shells. The salt (about 35 g of NaCl dm^{-3} sea water) needs about 1.5 kJ g^{1} NaCl upon excretion. This adds to the energy budget, not to the heat budget.

6.14 Evaporation of body water

After a dive, a duck is wet. Excess water will be shaken off, but the remains evaporate, and cause an extra cooling of the feather surface. The heat needed for evaporation will partly be withdrawn from the air, and partly from the duck. As a result of evaporation the feather surface temperature as used in eq. (15) and section 6.1 decreases and thus heat loss will increase according to eq. (17). However, this process in not implemented in the present version of the heat budget.

6.15 Energetic uptake

The flesh content of a prey is digested, and assimilated. Not all the flesh is used completely, nor is the assimilation complete. On average, assimilation efficiency (including digestion efficiency) is about 70-80%. The non-digested part is just leaving the guts without any contribution to the heat-, energy- or mass budget of the duck.

The food that passes the gut wall is converted into storage matter later on, and this conversion is not 100% efficient as well. This loss will appear as heat production on the heat budget. We do not have data for this part of the process.

Therefore, we omitted this in our model and used 75% assimilation efficiency as basic value for the uptake process.

6.16 Maximum uptake rates

The maximum uptake rate describes the maximum amount of meat or shells a duck can handle (per unit of time). The maximum amount of shells depends on the volume of guts and stomach, and their digestive capacities.

As written in section 2, Swennen (1974) described an eider duck that swallowed 28 mussels with a size of 25 mm in only a few minutes. It took the duck about one and a half hour to process this completely. Since a 25 mm shell contains about 0.12 g AFDW, and weighs about 2.5 g, it implies a processing rate of about 60 g total shell mass and 3 g AFDW per 1.5 h, or 400 shells with total mass of 960 g per day and 50 g AFDW per day (= 1.1 10^3 kJ day^{-1}).

According to the time budgets as described above (sections 6.9-6.11), an eider duck may process about 1 mussel per minute, or 1400 mussels per day. With an average of 0.6 g AFDW per shell this implies about 840 g AFDW per day, or 16 kg total fresh shell mass per day, and 16.6 10^3 kJ day^{-1} in terms of energy. De Leeuw (1997) mentions a maximum total daily fresh shell mass uptake for tufted ducks of about 3 times its body mass. For an eider duck this would mean that about 6 kg mussels per day is the maximum amount of food. This would equal about 300 g AFDW day^{-1}, and 6.6 10^3 kJ day^{-1}.

Nehls (1995) presents an uptake rate of 0.85-0.9 g AFDW min^{-1}, or 18.8 kJ min^{-1} which equals 1224 g AFDW day^{-1} or 39 10^3 kJ day^{-1}. These data were based on dabbling ducks and ducks were not active 24 hours continuously. Nehls (1995) also mentions a daily energy expenditure of about 3 10^3 kJ day^{-1}. Thus, the high uptake values mentioned above represent short time maxima. In the model we constructed we implemented an upper limit: the duck cannot gather more than this amount of shells. This is a model parameter, and has to be chosen. Thus, if a duck needs (from energetic considerations) to gather more food than this upper limit, the duck will not be able to meet its own energy requirements.

6.17 Basal metabolic rate

In order to maintain its own existence, an organism needs energy to prevent loss of essential organic and inorganic matter. This process is called the basal metabolism; the energy is provided by the oxidation of organic storage matter. As a result, the animal produces heat. Thus, the basal metabolic rate (BMR) is a continuous process irrespective of any other activity. The BMR is a negative term on the energy budget, and a positive one (with same magnitude) on the heat budget. In case of no feeding, a duck loses mass as a result of the process.

For eider ducks, an BMR average of about 4.05 W kg^{-1} is mentioned by Nehls (1995); for tufted ducks (*Aythya fuligula*) 4.9 W kg^{-1} is mentioned by De Leeuw (1997).
Rahn & Whittow (1984, cited by Wilson et al, 1992) give:

$$BMR = cM^{d} \qquad (49)$$

with
d = 0.744, and c = 2.29 10^{-7} (m^3 (oxygen) s^{-1} kg^{-1})[1], and BMR in m^3 oxygen s^{-1}.

Since oxygen is equivalent to 22.5 *30/32 = 21 kJ g^{1} (oxygen), and the density of oxygen is about 32/22.4 (gram per mol divided by dm^3 per mol) = 1.4 g dm^{-3}, 1 m^3 of oxygen (1 atm) is equivalent to 1000 * 21 * 1.4 kJ = 29.4 10^3 kJ. Thus, in eq. (49), c equals 2.29 10^{-7} * 29.4 10^3 = 6.7 (W kg^{-d}) in case BMR is in W, in stead of oxygen volume. For M= 2.0 kg, the BMR is about 11.2 W, which is considerably higher than the 8 W mentioned by Nehls. For a tufted duck (body mass of 600 g), eq. (49) gives 4.6 W, which is clearly above the 2.97 W De Leeuw (1997) mentions. We applied the description by Nehls, although we realize that a better description of the relationship with body mass is desirable.

6.18 Faeces production

Faeces production is not an item of interest when modelling heat or energy budgets. It has been argued above that faeces do not contribute to the heat budget. The amount of faeces (if wanted) is computed directly from the food and shell uptake, and the digestion efficiency, which is estimated to be about 75%.

[1] Note that the mentioned units by Rahn &Whittow are not correct: the unit of c is m^3 (oxygen) s^{-1} kg^{-d}. The value of d is part of the unit of c

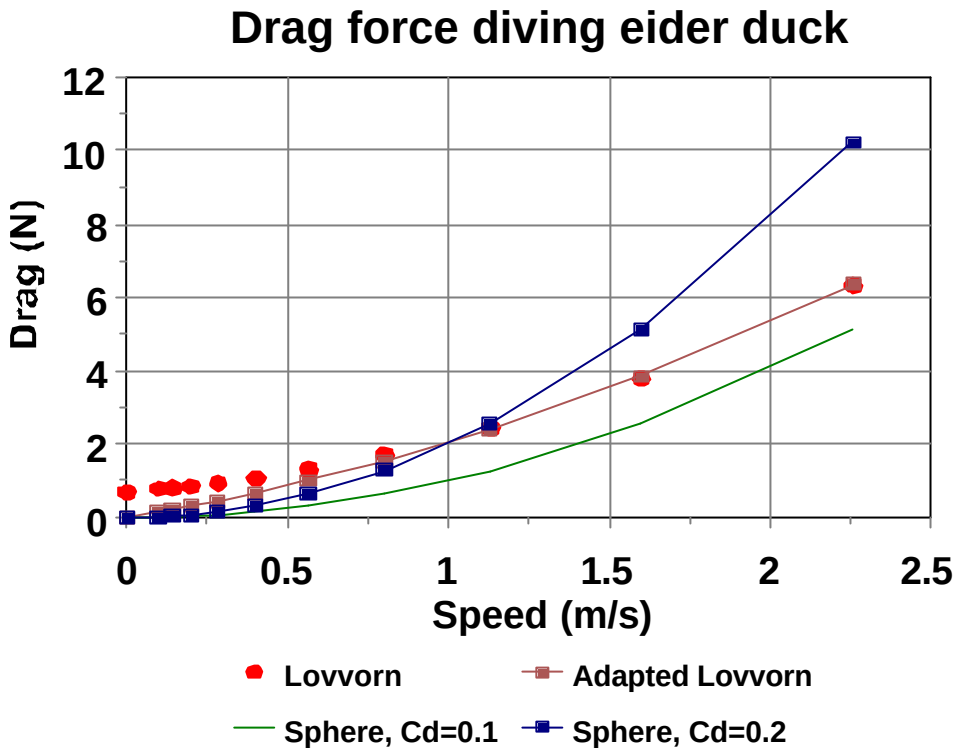


Figure 20. Friction forces (drag) for a diving duck. Red: original equation from Lovvorn (1991), Adapted Lovvorn (brown): gives drag=0 at speed=0. Drag around sphere: computed according to eq. (34); green line C_d=0.1, bleu squares C_d=0.2. The diameter of a duck is estimated.

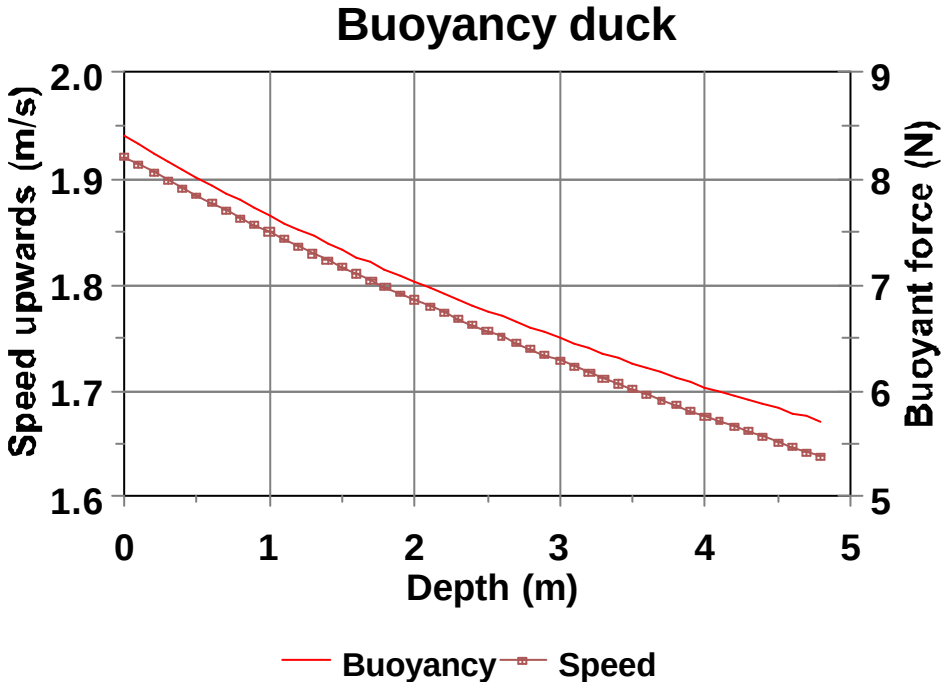


Figure 21. Buoyancy of a duck and upward velocity as a function of depth. Drag forces according to eq.(34). When Lovvorn et al (1991b) is followed, upward velocities would be somewhat higher.

7 Prey decision strategy

7.1 General

An assumption that is crucial for the model is the way we expect the bird to decide whether to take a prey or to find another (and better) one. When the duck reaches a shellfish in or on the sediment, it can take the first prey it encounters, or it can decide to search until the best possible prey is found.

We assume that the choice is directly related to the energetic profit of a prey of a certain length and the occurrence of the shell (coupled to the size frequency distribution of the shells on the foraging site). Thus, the profit distribution together with the real length frequency distribution is the weight factor that determines the food uptake of the duck. There are some limitations to the uptake (section 6.1). Certain sizes are impossible to catch for an eider duck. Very large shells cannot be swallowed and very small shells cannot be handled.

Thus, when the net energy gain for shell size A is twice the energy gain for shell size B, shell A is taken twice as much as shell B, assuming equal appearances. Such a weighting can be handled in many ways. Not only a linear relationship with the net gain can be assumed, but a stronger one is possible too. There is no sound basis for the linear weighting we used for our model computations.

7.2 Model

In the simulation model, we consider prey with a certain size distribution. For each size class (we used 5 mm sub-ranges) the energetic profit is computed as the difference between yield and costs. This is exactly according to eq. (5), when each dive yields one prey of a particular size. The model also computes the time needed to catch and handle a prey (based on diving speed, upward velocity and searching time at the bottom, see chapter 6), thus an energetic profit per unit of time is known. Thus, for each prey size, we know the net profit in terms of J s^{-1}. Those size ranges that yield a negative profit are excluded. Then, the preference of each size class follows from the values of these net profits. Thus, let profit(len) be the profit of a shell of length len (J s^{-1}), then the preference of a bird for size class len equals (for N size classes) is:

$$\Pr eference(len) = \frac{\Pr ofit(len)}{\sum_{l=1}^{N} \Pr ofit(l)} \qquad (-) \qquad (50)$$

The prey choice of the bird also depends on the availability of the size classes and physical limitations. When Freq (len) is the relative availability (based on numbers) of

size class len, and Phys(len) the capability to take a prey of size len, then the real prey choice is:

$$Choice(len) = \frac{Profit(len) \cdot Freq(len) \cdot Phys(len)}{\sum_{l}^{N} Profit(l) \cdot Freq(l) \cdot Phys(l)} \quad (-) \quad (51)$$

Note that eq. (51) actually says that in case of a large part of physically unmanageable or less manageable preys (Phys (l)<1), the duck has a total prey choice considerably smaller than 1:

$$Choice\_tot = \sum_{l=1}^{N} Choice(l) \quad (-) \quad (52)$$

For the model computations, this implies that per dive there are (1-Choice_tot) failures.

7.3 Food uptake per dive

Eq. (52) gives the food preference of an eider duck. It also is the size frequency distribution of preys gathered by an eider duck, when many dives are considered. Thus, the energy budget of each average dive can now be computed. The average flesh content is computed according to

$$\overline{flesh} = \frac{\sum_{l=1}^{N} choicel_l \cdot flesh_l}{\sum_{l=1}^{N} choice_l} \quad (g\ dive^{-1}) \quad (53)$$

Similar equations give the (average) costs per dive for crushing the shells, for heating the prey, etc. From section 6, diving costs and diving time are known, and per dive an energy and heat budget is known.

7.4 Kinetic plus potential energy and heat

Two specific situations can be considered: a submerged duck diving for a prey and an emerged duck, which is handling prey or resting. During resting the duck loses heat, thus heat production is needed to overcome that loss. A duck produces heat because of the basal metabolic rate. The energy budget above water is negative (there is no energy provision during resting), the heat budget may be positive or negative. If negative, the bird needs to convert energy to heat. While diving the duck needs energy and produces heat. The energy need is always negative. The heat budget may

be negative or positive. The collected food provides the duck with energy , which is always a positive term on the overall energy budget.

We now have four different situations to handle.

a- Heat budget is negative during resting and during diving

In both situations there is a shortage of heat and this adds to the energy demand in eq. (3). The heat demand during resting appears in the nominator, the heat demand per dive appears as a negative profit in the denominator.

b- Heat budget is positive during resting and during diving

In both situations there is a surplus of heat. The bird has to loose that heat by decreasing the insulating properties of the feathers, by extra dives, or by taking smaller preys than possible. The last behaviour also implies more dives per day than strictly necessary.

c- Heat budget is positive during resting and negative during diving

The surplus heat during resting may compensate the heat demand as a result of the dives. Without such compensation, the energy budget per dive (costs plus food profit) would be the amount of the heat demand lower than the sum of diving energy costs and food profit. It is tested whether the resting heat surplus is enough for compensating the complete heat loss during diving or not.

d- Heat budget is negative during resting and positive during diving

The heat surplus per dive is used to (partly) compensate the negative budget during resting. It is also tested whether the heat surplus from the dive is enough to compensate the heat demand during resting completely or just for a part.

7.5 Special case: more than one prey per dive

Nehls (1995) mentions that a duck does not catch more than one prey at a time. Later he stated (pers. comm.) that especially when mussels are small, (many) more than one mussel per clutch could be swallowed. To test the effects of more than one prey per dive, a multiple-prey factor (e.g. 'MultPrey') may be implemented in the model. This factor may depend on size (len) of the prey and may thus be valid for all prey sizes or only part of these sizes (e.g. the smaller ones only). In the present version of the model a multi-prey factor is not included.

8 Simulation model

The whole set of equations as mentioned above is implemented in the simulation model EIDER. A flow diagram is presented in Figure 22. The model is written in Borland Pascal for Windows, vs. 7.0. and runs on any machine equipped with Windows9x, WinNT, Win2000 or XP. The complete listing is not added to this report, but is available upon request. In appendix D, some relevant parts are listed, including examples of the necessary input-files, the definition file (eisysdef.pas) and the crucial budget computations (eifunt1.pas and eifunct2.pas).

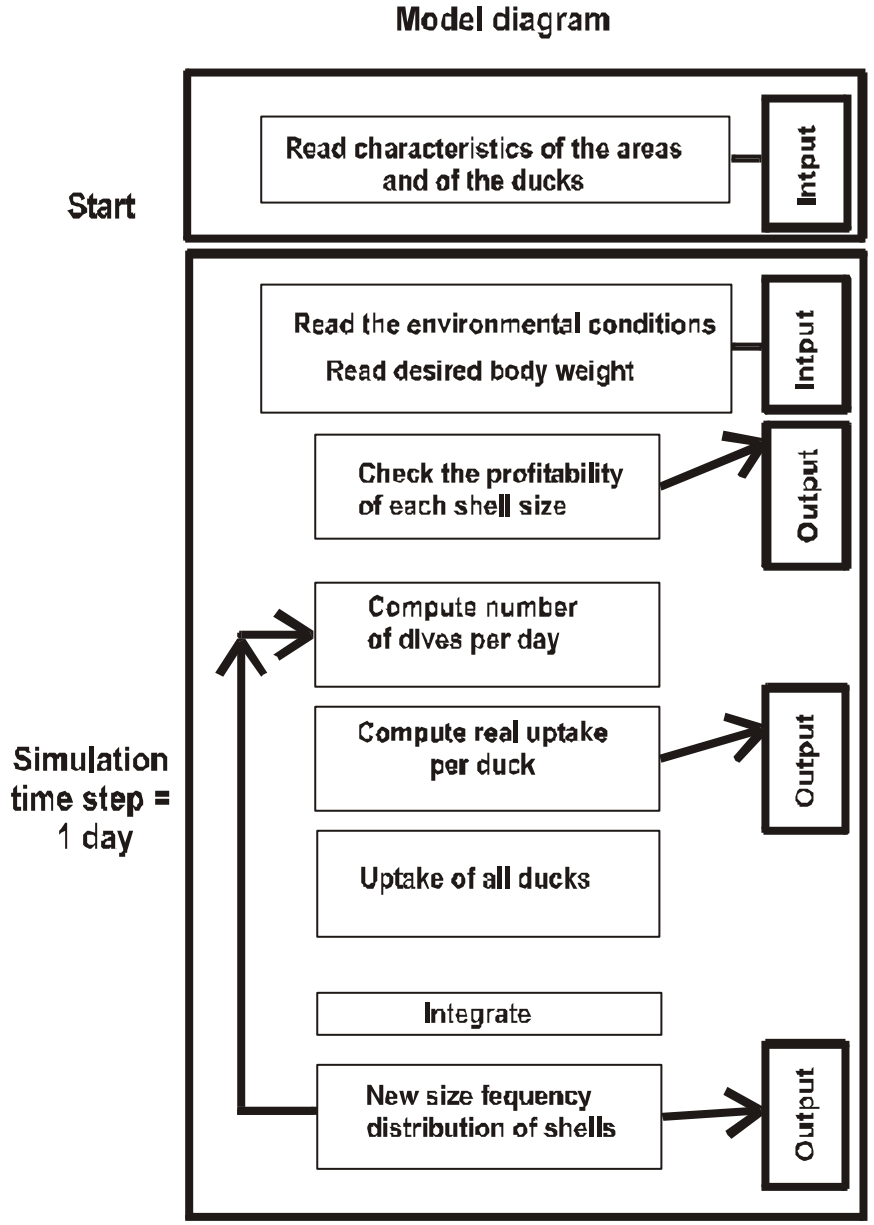


Figure 22. Flow diagram of model EIDER.

9 Air and water data

For realistic simulations, meteorological data (wind speed, air temperature) and water temperature data are needed. Humidity is not used as a variable in the present version of the model.

We had temperature data for four sites at our disposal, and wind data for one site:

- The western part of the Dutch Wadden Sea. Water temperature as monthly averages (computed from data for the period 1975-2001) for a central site, and air temperatures and wind speed as monthly averages (computed from data for the period 1975-2001) for the weather station De Kooy (The Netherlands, near Den Helder).
- The Limfjorden area (Logstoer area) in northern Denmark. F. Mohlenberg & P. Dolmer have supplied water and air temperatures as monthly averages over the period 1989-1999. Wind values were not available at this moment, and are taken from the Dutch data set.
- The Ljunskile area at the Swedish west coast. Water and air temperatures (averages for the period 1999-2000) have been supplied by L.O.Loo, and partly taken from Anderson (2000). Wind data were taken from the Dutch data set.
- The area in front of La Rochelle, west France. Monthly means for 1999 were supplied by CREMA/IFREMER (La Rochelle). Wind data are copied from the Dutch data set.

In Figure 23 an overview of one-year monthly weather data for the Netherlands is given. In Figure 24 data for Denmark, Sweden and France are presented.

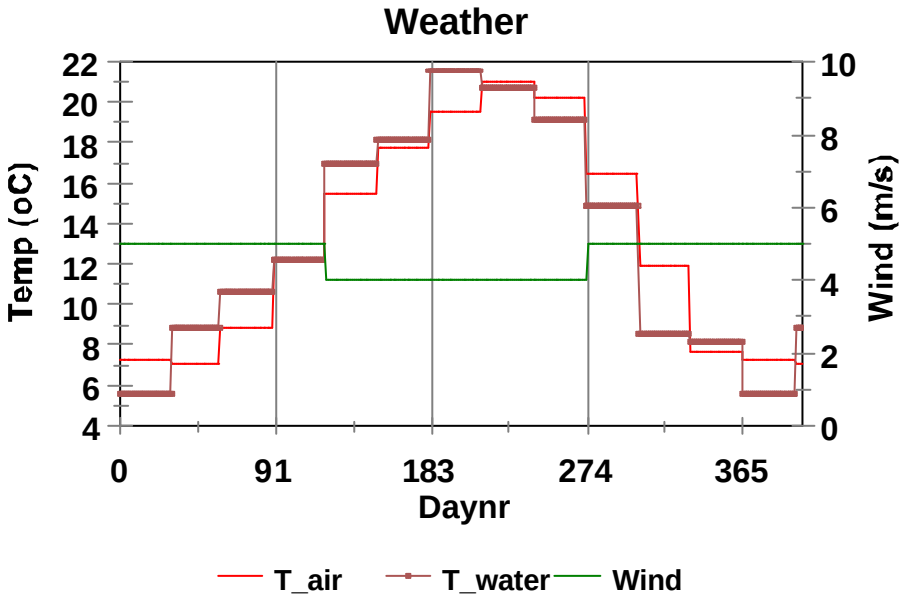


Figure 23. Monthly weather data. Station De Kooij (near Den Helder), The Netherlands.

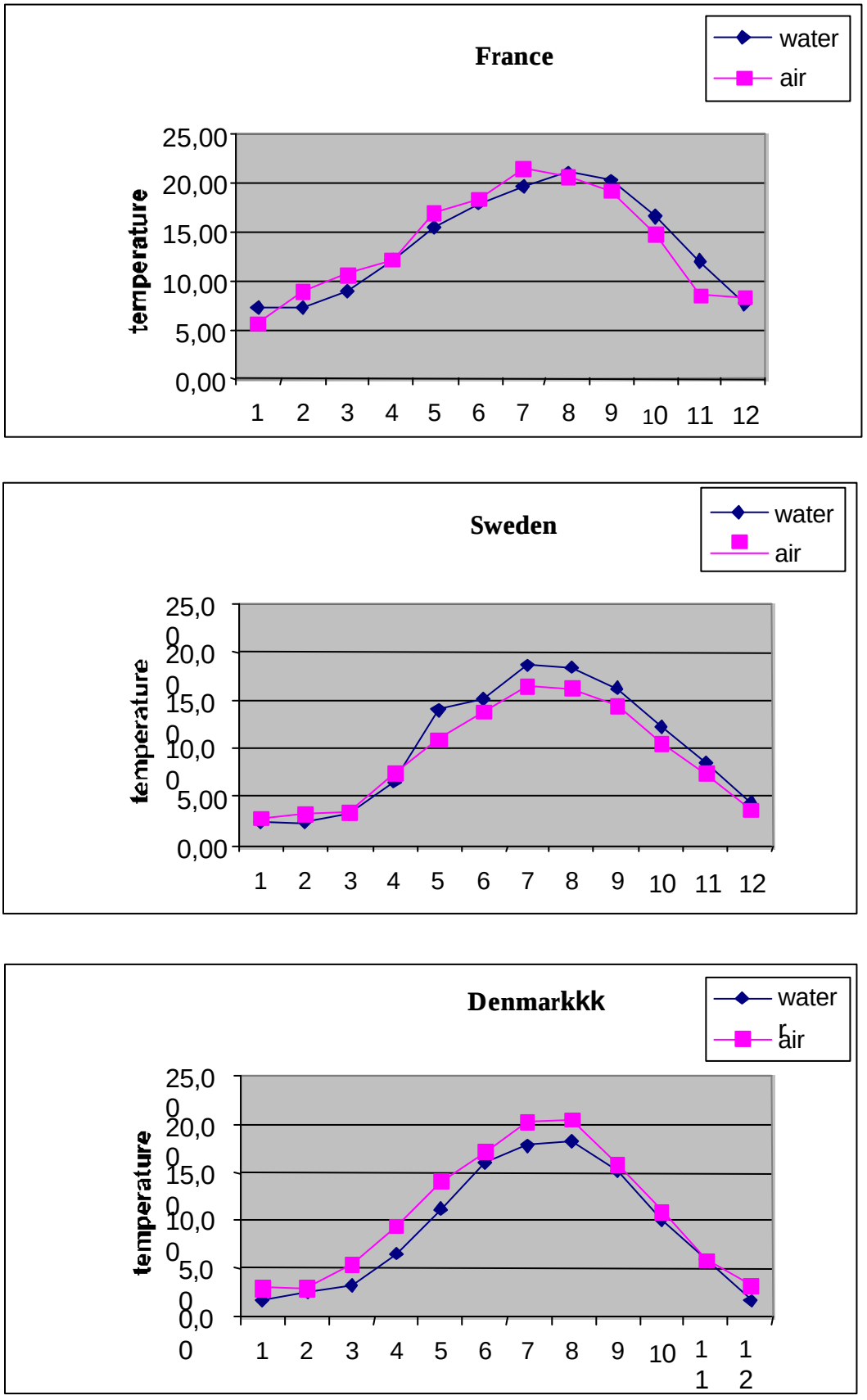


Figure 24. Air and water data for France, Sweden and Denmark. Explanation see text.

10 Characteristics of foraging areas

Sites

The Dutch Wadden Sea contains a number of mussel sites suitable for foraging eider ducks. Mussel beds occur as wild intertidal mussel beds, wild subtidal mussel beds and subtidal mussel culture lots. For most of the areas some information is available on sizes, stocks and size distribution.

In this study we restricted ourselves to wild subtidal mussel beds. Data were available from our own sampling, and from surveys by RIVO (Van Stralen, 1998). We used average bird numbers and shell data for the whole Wadden Sea because interference is not implemented as a relevant process. At this moment, our only goal was to compute quality of each site. For the Wadden Sea area, necessary data are:

- average water depth (while submerged),
- emersion time (as fraction of the tidal period),
- total size of the area (m^2),
- density and average size class distribution of shells, size class intervals were 5 mm, from 0 up to 75 mm (number m^{-2}),
- allometric constants (a, b) for ash free dry mass, shell mass and fresh mass,
- the variation of the condition factor (the a-value) for ash free dry mass during the season.

In appendix D, the input file for area characteristics is given.

Number of eider ducks

We do not intend to estimate here the exact number of eider ducks that can forage in the Dutch Wadden Sea. Although there are approximately 80.000 eider ducks in the Dutch Wadden Sea in wintertime, we used 20.000 as an overall year-value for the present simulations. For the other three sites, we did not perform a simulation with the number of birds because an accurate combination of number of overwintering ducks, available foraging area, and prey densities was not available.

For the sites in Denmark, Sweden and France, we just restricted ourselves to a comparison of the foraging possibilities for a single duck.

Bird parameters are given in the input file on bird characteristics (appendix D). The parameters apply to all the four different countries.

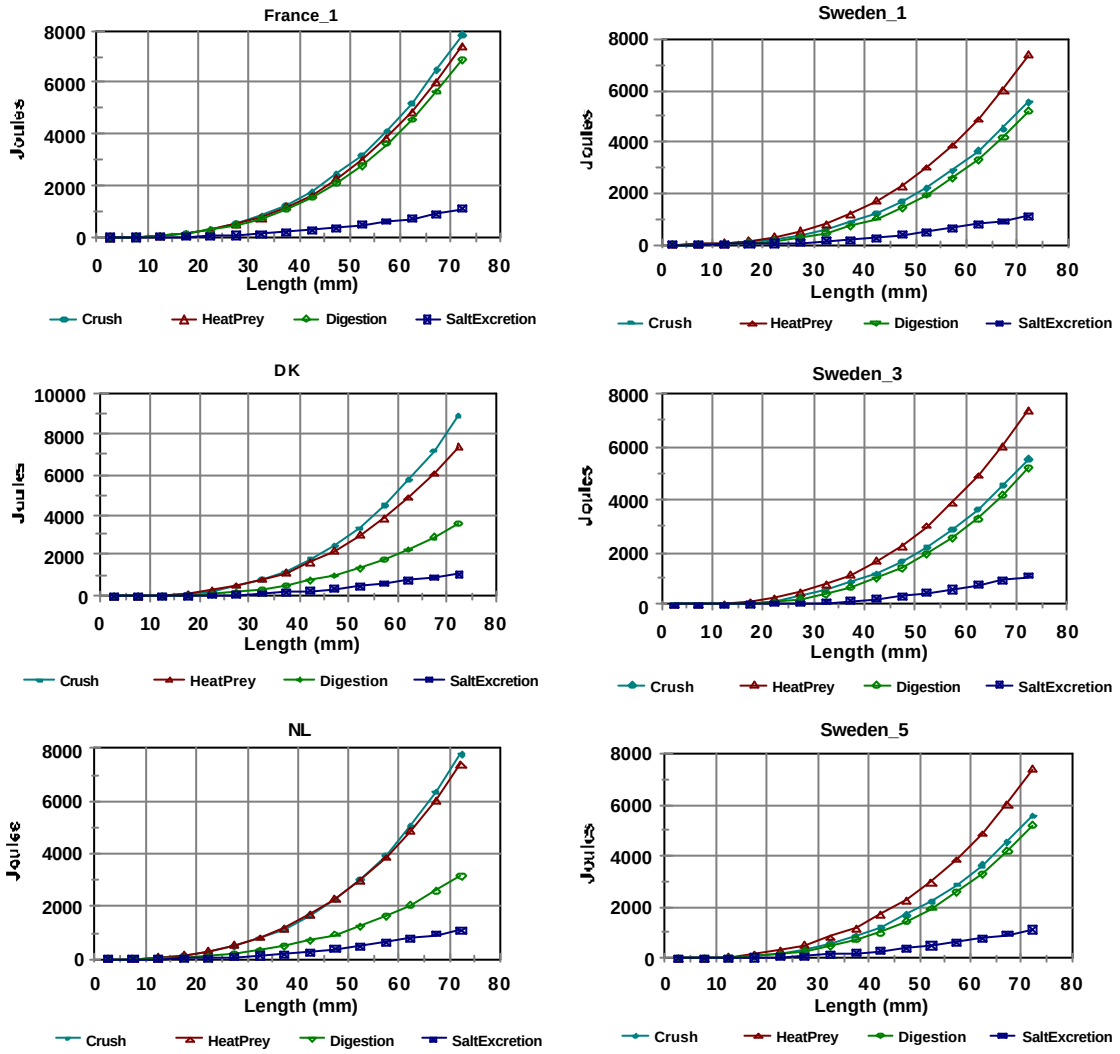


Figure 25. Costs of prey handling in four countries. For Sweden all three sites are shown. SW are long lines: SW_1 is one meter deep, SW_3 three, and SW_5 five meters deep. FR_1 is the long line situation in France. DK and NL are benthic mussel communities in Denmark and The Netherlands respectively. Especially crushing costs are high in Denmark.

11 Results

11.1 Performed computations

We performed two types of computations. For the 4 countries we computed the number of necessary dives per day for a single duck. We used this figure as a quantity that shows us how profitable a site is for a duck. Costs for handling the prey and the amount of flesh per dive are given below. We also computed profitability of the three different depths in Sweden, and the profitability of the mussels on the long lines and at the bottom of a pole in France.

Results for the latter indicated that the food at the bottom part of the poles in France is insufficient for an eider duck; therefore, details of the energy budget are not mentioned here.

11.2 Costs, flesh profits and number of dives

Prey handling costs for each shell length are presented in Figure 25. All data are valid for the winter period. Salt excretion is the same in all cases because there is not a different salt content assumed. In France digestion costs are high because of the high meat content of shells. In Denmark and The Netherlands digestion costs are low. In Sweden digestion costs are intermediate. Crushing costs are especially high in Denmark; the shells are much thicker than in the other situations.

The amount of flesh, and thus energy, a bird collects per dive is shown in Figure 26. The situation in Sweden and Denmark is that the birds find a relatively low quantity of food per dive. In France, at the long lines, they find a large quantity, and in The Netherlands the benthic mussel beds provide an intermediate amount of flesh per dive.

The number of dives a bird needs per day is more or less a resultant of total costs (which of course also involve heat losses and basal metabolic costs) and the amount of flesh that a bird colects per dive. In Figure 27 the number of dives (in wintertime) is sketched. In The Netherlands, the number of necessary dives in the first winter is about the same as in Denmark. In the next winter, the number of necessary dives becomes higher in The Netherlands because there was a depletion of food (in our simulations). In summer, the Dutch situation is better, which has to be assigned to the better condition of the shellfish. In France, the necessary number of dives per day is much lower; a result of the much higher flesh content of the shells. In Sweden the number of dives is of the same order as in The Netherlands and Denmark, but the distribution over a year differs from the Danish and Dutch situation. Careful examination of the graphs learns that diving to 1 metre is (only) slightly more efficient than to 3 or 5 metres deep.

11.3 Prey choices

For the Dutch situation we were able to perform a simulation for a more or less realistic number of birds and a realistic food resource situation. About 20.000 eider ducks feed on 22000 ha of mussel area. Shell densities are according to the first graph of Figure 28. With the used numbers of eider ducks and mussel area the simulations show that during a year food sources are partially depleted. The densities of large shells decrease during the year. We did not implement the natural increase in mussel sizes. In reality mussels grow and fill up the gap caused by the ducks. Also, we only considered wild mussel beds and left the mussel culture plots out of the computations. But also, normal wintering numbers of ducks are considerably higher than 20.000. Thus our simulation should be considered as an example.

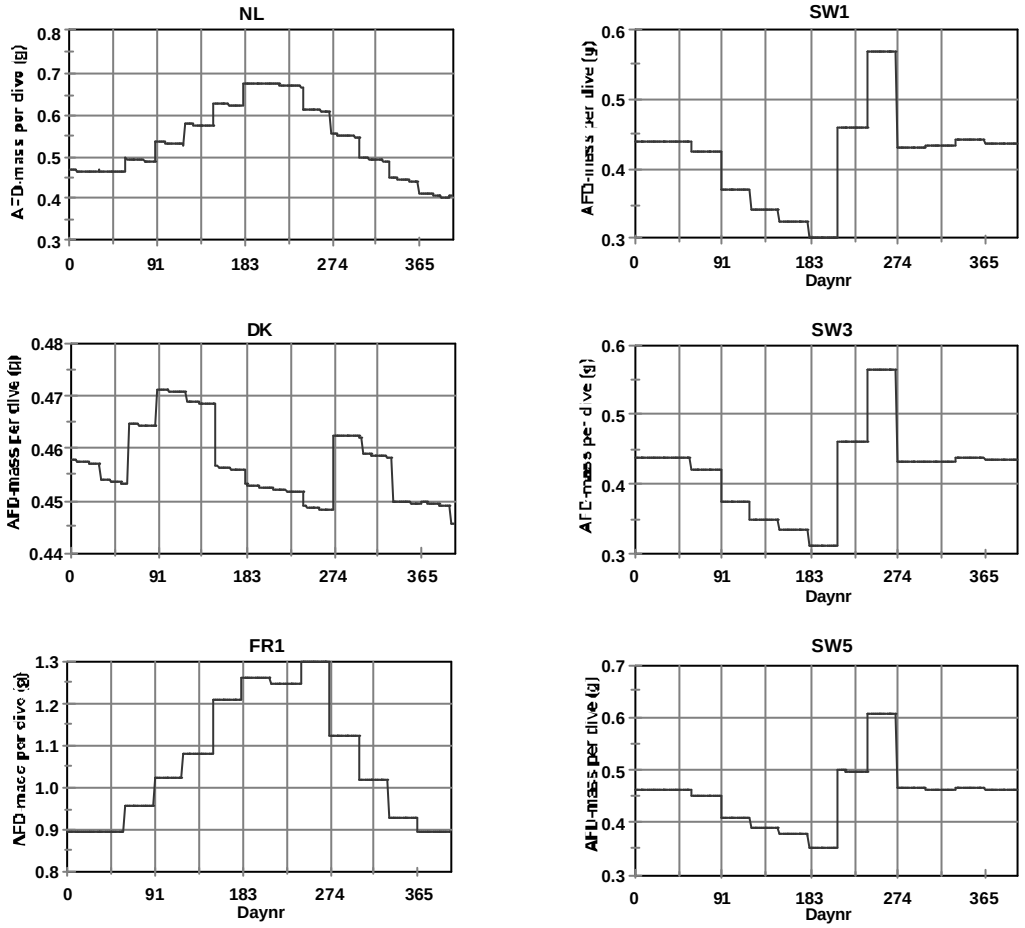


Figure 26. Amount of flesh (in grams AFDW) a duck collects per dive. It is assumed that a duck can take one prey per dive .

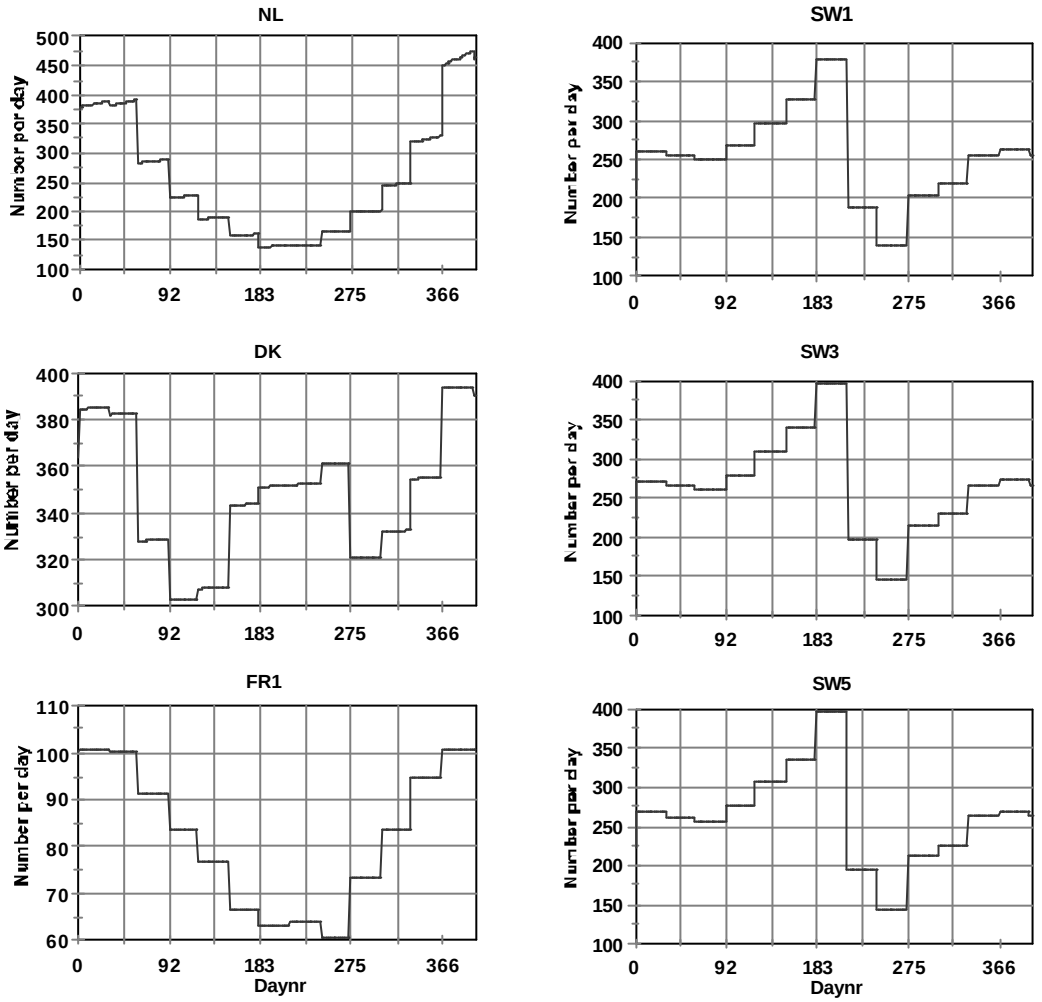


Figure 27. Number of dives a duck needs to make per day to get enough food. The mussels at the bottom of a pole in France are not of sufficient quality to feed a bird .

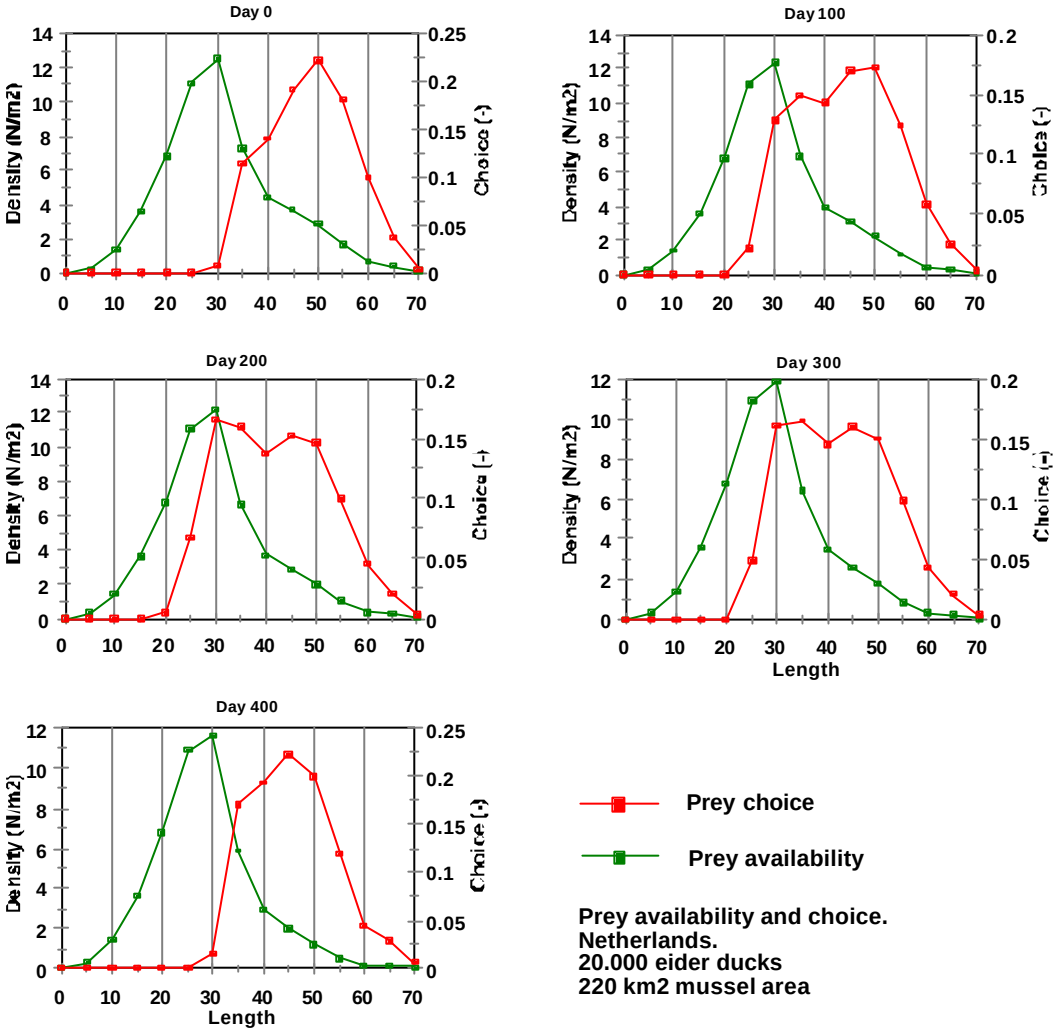


Figure 28. Prey choice and prey availability simulation for the Dutch Wadden Sea: 20.000 eider ducks, and 22000 ha mussel area; densities of shells according to the upper left graph.

The simulation showed that eider ducks can benefit from mussels down to 20 mm during summer, while they have to restrict themselves to shells larger than 30 mm in winter. The reason is that in summer costs are lower (especially heat losses are much lower) while the profits (meat content) per shell is higher. This result fits very well the observations (Nehls, pers. comm.; Nehls, 1995) that ducks prefer smaller shells in summer than in winter despite of the better profit of large shells in summer. De Leeuw found similar feeding characteristics for tufted ducks (*Aythya fuligula*). May be, eider ducks prefer small shells but have to take the larger ones in winter because of energetic reasons.

11.4 An energy budget

Finally, we can compute an energy budget for an eider duck during a dive. In Figure 29a, characteristics of a large mussel are shown. In fig. 29b the heat budget for a winter day in The Netherlands is shown, and in fig. 29c the energy budget. In fact all the terms are costs, except the input term in the energy budget: the food. Part of the energy costs appear as input (gain) on the heat budget: BMR and crushing costs, mainly. The net gain of one dive is about 1.6e3 Joules, which is used to compensate the 'standard costs'.

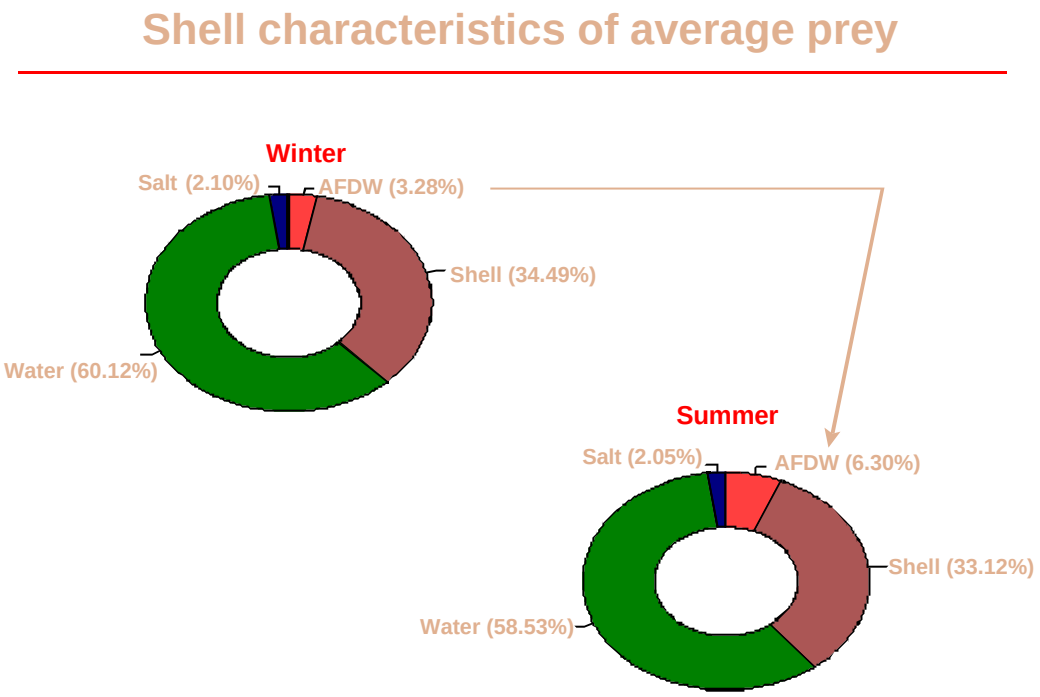


Figure 29a. Composition of mussel shells Netherlands, winter and summer situation. Especially the relative shell mass differs from site to site, and differs between species.

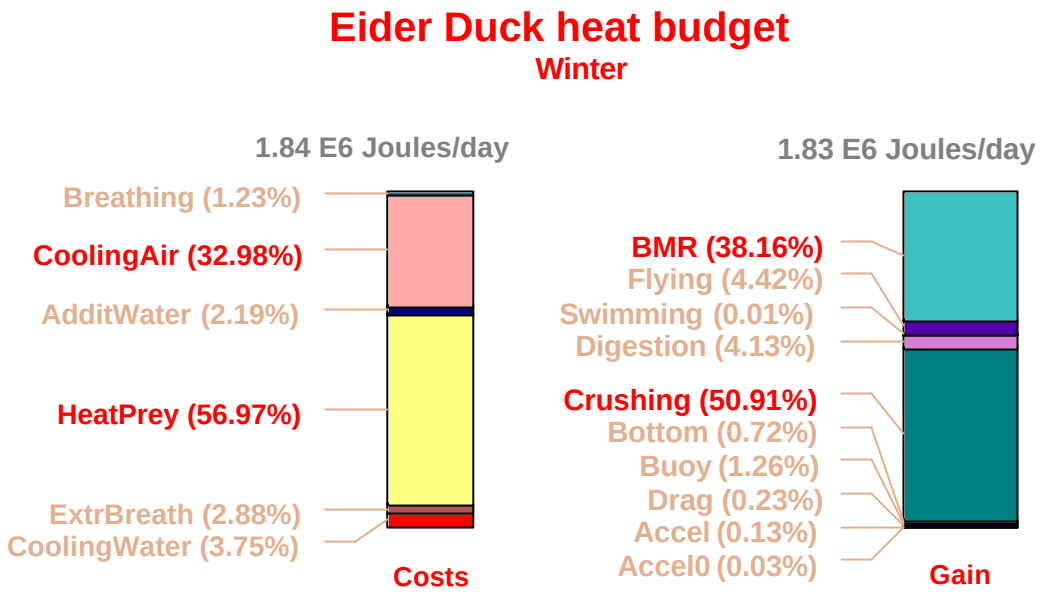


Figure 29b. Heat budget for an eider duck (winter situation, The Netherlands; start of the simulation). The duration of one dive is 20.5 seconds. The whole cyclus (including resting and handling) is about 52 sec. It is obvious that the most important costs concern the heat loss to the air and the heat needed to increase the temperature of the prey. Heat losses during diving are of minor importance. Crushing the shells costs a lot of energy (fig. 29c), but since this is converted into heat, it is also an important gain for the heat budget. The heat corresponding to the basal metabolic rate is the second important entry.

The budget will change with changing weather conditions and shell characteristics.

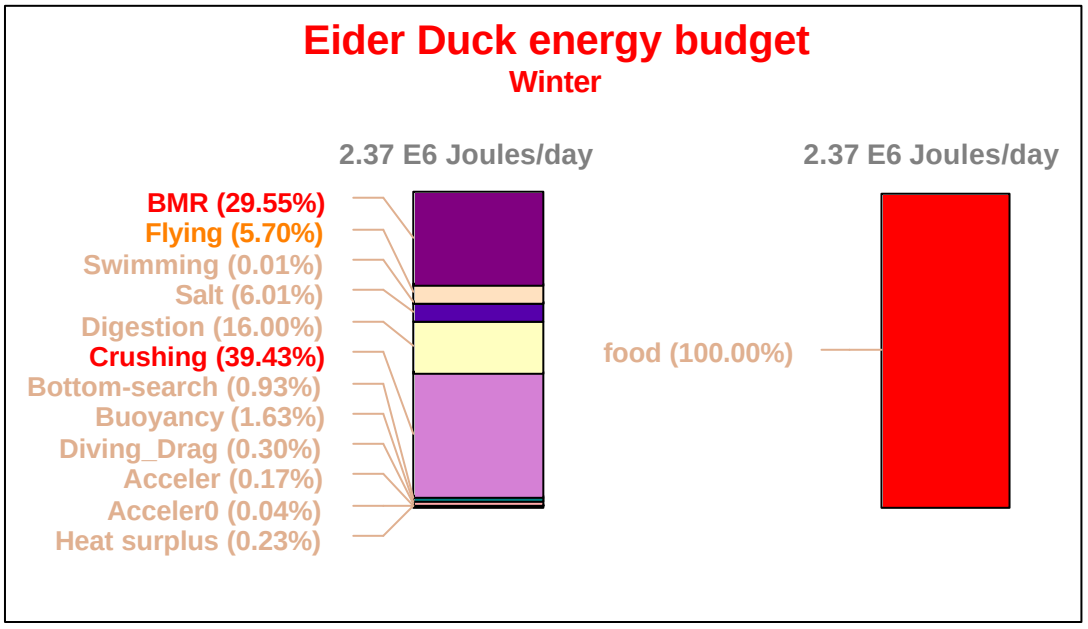


Figure 29c. Energy budget for an eider duck (winter situation, The Netherlands; start of the simulation). The duration of one dive is 20.5 seconds. The whole cyclus (including resting and handling) is about 52 sec. Only positive entry concerns the food; most important costs is the crushing energy and the basal metabolism. Digestion costs is the third most important item. Note that flying is of minor importance here, but this will change drastically when the duck moves to other areas in Europe.

This budget will change with changing weather conditions and shell characteristics

Daily energetic expenditures (DEE) (again the Dutch situation only) are given in Figure 30. DEE as a function of environmental temperature is presented in Figure 31. Figure 30 is in good agreement with Nehls (1995) who mentioned a DEE of about 3.0e6 J in winter, and about 2.1e6 J in summer. We calculate a lower demand in summer, but our winter values are almost the same. The reasons for the differences in summer are not completely clear yet.

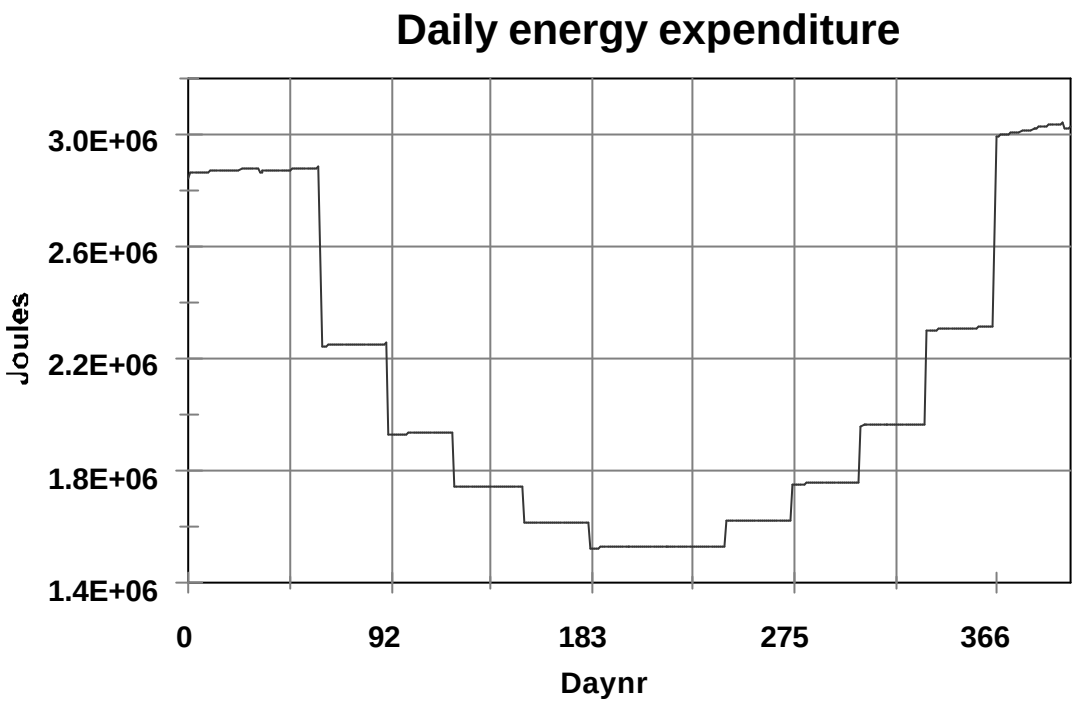


Figure 30. Daily energy expenditure of an eider duck in The Netherlands, as computed by the model. Note that the budgets are slightly different from those in figs 29b and 29c.

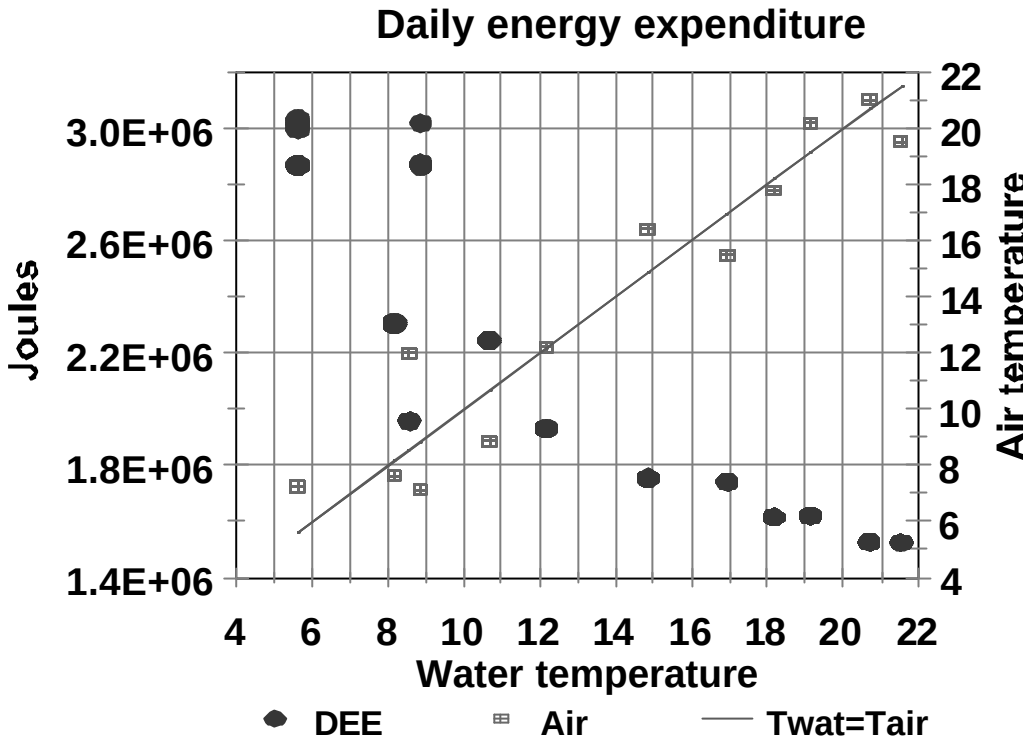


Figure 31. Daily energy expenditure as a function of environmental temperature as computed by the model. Situation for the Netherlands, all simulation results are pooled.

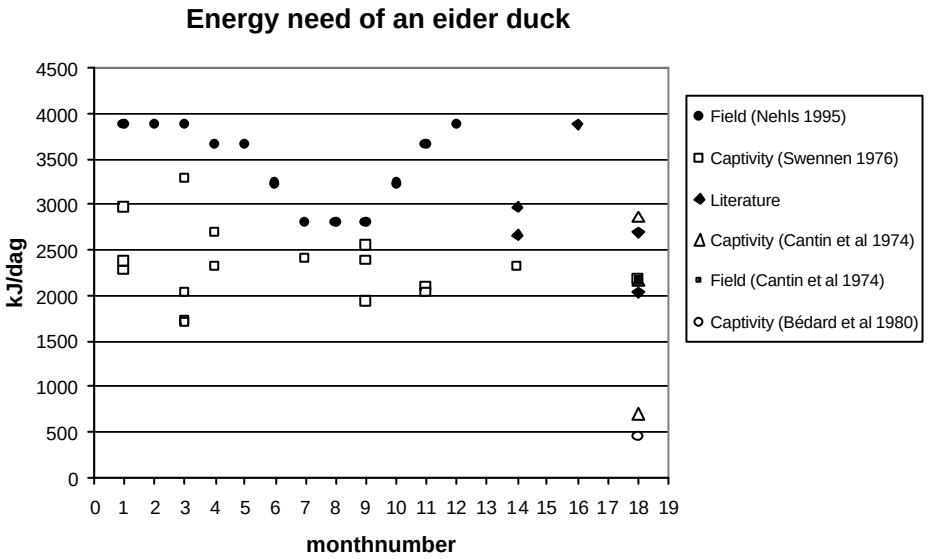


Figure 32. Energy demand of an eider duck based on literature data. Note that these data are based on the energy content of the prey; the data in figure 30 and 31 are based on 75% assimilation efficiency of that same prey. Thus, the data in this graph have to be multiplied by 0.75 before comparing them with the model results.

12 Final

We performed a first analysis with mussels as a characteristic food source for eider ducks. In future, cockles and spisulas will be subject of our investigations. The developed model proved to be useful; it made a comparison between the several sites in Europe possible. In a next version we intend to implement a site decision strategy in the model, which should allow us to understand better why an eider duck does or does not shift from one site to another.

The preliminary simulations done for the Dutch situation showed that the eider ducks are capable to eat a significant part of the mussel stock. Based on the present work, however, it is too early to draw any conclusions on the relationship between fishery activities and a possible food shortage for the ducks. Therefore, also cockles have to be taken into account as a food resource, as well as spisulas and may be other shellfish species in the North Sea coastal zone.

References

Bauer, K.M., & U.N. Glutz von Blotzheim. 1969. Handbuch der Vögel Mitteleuropas, Band 3, 2. Teil). Akademische Verlagsgesellchaft. Frankfurt amM.

Beauchamp, G., M. Guillemette & R. Ydenberg, 1992. Prey selection while diving by common eiders, *Somateria mollissima* Anim. Behav. 44: 417-426

Bédard J., Therriault J.C. & Bérubé J. 1980. Assessment of the importance of nutrient cycling by seabirds in the St. Lawrence estuary. Canadian Journal of Fisheries and Aquatic Sciences 37: 583-588.

Brinkman, A.G.1993. Estimation of length and weight growth parameters in populations with a discrete reproduction characteristic. IBN Research Report 93/5. 27 pp +app.

Brinkman, A.G. & B.J. Ens, 1998. Effecten van bodemdaling in de Waddenzee op wadvogels. IBN-DLO rapport 371, 249 pp.

Bult,T.P., van Stralen,M.R., Brummelhuis,E. & Baars,D. Mosselvisserij en - kweek in het sublitoraal van de Waddenzee. RIVO Rapport - Concept voor stuurgroep EVA II, 1-74. 2003b. Yerseke, RIVO.

Buschbaum, C., 2001. Direct and indirect effects of Littorinas littorea (L.) on barnacles growing on mussel beds in the Wadden Sea. Hydrobiologia 440:119-128

Campbell, A.C., 1976. The Hamlyn guide to the seashore ans shallow seas of Britain and Europe. Hamlyn Publ. Group, London

Cantin M., Bédard J. & Milne H. 1974. The food and feeding of Common Eiders in St. Lawrence estuary in summer. Canadian Journal of Zoology 52: 319-334.

De Leeuw J. 1997. Demanding divers - Ecological energetics of food exploitation by diving ducks. Proefschrift, RU-Groningen.

Ens, B.J. & Cayford, J.T. 1996. Feeding with other Oystercatchers. Chapter 4 in J.D. Goss-Custard (ed.), The Oystercatcher: from individuals to populations, Oxford Univ. Press, Oxford, 77-104.

Goss-Custard, J.D., West, A.D., Caldow, R.W.G., Durell, S.E.A. le V. dit, McGrorty, S. & Urfi, J. 1996. An empirical optimality model to predict the intake rates of feeding oystercatchers *Haematopus ostralegus* feeding on mussels *Mytilus edulis.* Ardea 84A: 199-214.

Hamilton D.J., Nudds, T. D. & Neate, J. 1999. Size-selective predation of blue mussels (Mytilus edulis) by Common Eiders (Somateria mollissima) under controlled field conditions. Auk 116 (2): 403-416.

Hawkins, P.A.J., P.J. Butler, A.J. Woakes & J.R. Speakman, 2000. Estimation of the rate of oxygen consumption of the common Eider Duck (Somateria mollissima), with some measurements of heart rate duting voluntary dives. Journ. Exp. Biology 203:2819-2832

Hulscher, J.B. 1996. Food and feeding behaviour. Chapter 1 in J.D. Goss-Custard (ed.), The Oystercatcher: from individuals to populations, Oxford Univ. Press, Oxford, 7-29.

Jenssen B.M., Ekker, M. & Bech, C. 1989a. Thermoregulation in winter-acclimatized common eiders (Somateria mollissima) in air and water. Canadian Journal of Zoology 67 (3): 669-673.

Jenssen B.M., Ekker, M. & Bech, C. 1989b. Erratum: Thermoregulation in winter-acclimatized common eiders in air and water. Canadian Journal of Zoology 67:669-673.

Kooijman, G.L. 1975. Behaviour and phyiology of diving.In: B. Stonehouse (ed). The biology of penguins, pp. 115-137. MacMillan, London.

Lasiewski, R.C. & W.A. Calder. 1971. A preliminary allometric analysis of respiratory variables in resting birds. Respiratory Physiology 11:152-166

LWVT/SOVON, 2002. Bird migration over The Netherlands (in Dutch). Schuyt & co, Haarlem. 2002

Lovvorn, J.R., D.R. Jones & R.W. Blake, 1991. Mechanics of underwater locomotion in diving ducks: drag, buoyancy and acceleration in a size gradient of species. J. exp. Biol. 159: 89-108

Lovvorn, J.R, & D.R. Jones, 1991a. Effects of body size, body fat, and change in pressure with depth on buoyancy and costs of diving ducks (Aythya spp.). Can. J. Zool. 69:2879-2887

Lovvorn, J.R. & D.R. Jones, 1991b. Body mass, volume and buoyancy of some aquatic birds, and their relation to locomotor strategies. Can. J. Zool. 69: 2888-2892

Lovvorn, J.R. & D.R. Jones, 1994. Biomechanical conflicts between adaptations for diving and aerial flight in estuarine birds. Estuaries, 17 (1A): 62-75

Madsen, F.J. 1954. On the food habits of the diving ducks in Demark. Dan. Rev Game Biol. 2: 157-226

Meltofte, H., J. Blew, J. Frikke, H-U. Roessner & C.J. Smit, 1984. Numbers and distribution of waterbirds in the Wadden Sea. IWRB-Publication 34. Wader Study Group Bulletin 74, special issue. Common Secretariat for the Coperation of the Protection of the Wadden Sea. Wilhelmshaven.

Nehls G. 1995. Strategien der Ernahrung und ihre Bedeutung fur Energiehaushalt und Ökologie der Eiderente. Dissertatie, Universitieit Kiel (173 blz).

Saier, B. & C. Buschbaum. 2001 Growth of the mussel Mytilus edulis L. in the Wadden Sea affected by tidal emergence and barnacle epibionts J. Sea Res. 45: 27-36

Saier, B., C. Buschbaum & K. Reise, 2002. Subtidal mussel beds in the Wadden Sea: threatened oases of biodiversity Wadden Sea Newsletter, 25 (1) (in press)

Stephenson, R., J.R. Lovvorn, M.R.A. Heieis, D.R. Jones & R.W Blake (1989). A hydromechanical estimate of the power requirements of diving and surface swimming in lesser scaup (Aythya affinis). J. exp. Biol. 147-507-519

Stillman, R.A., Goss-Custard, J.D. & Caldow, R.W.G. 1997. Modelling interference from basic foraging behaviour. J. Anim. Ecol. 66: 692-703.

Swennen C. 1976. Population structure and food of the Eider Somateria m. mollissima in the Dutch Waddensea. Ardea 64 (3/4): 311-371.

Swennen, C. 1991. Ecology and population dynamics of the Common Eider in the Dutch Wadden Sea. PhD-Thesis Univ.Groningen, 144 pp.

Van der Meer, J. & Ens, B.J. 1997. Models of interference and their consequences for the spatial distribution of ideal and free predators. J. Anim. Ecol. 66: 846-858.

Van Stralen, M.R., 1998. The development of the mussel stock in the Wadden Sea and the Eastern Scheldt after 1992 (in Dutch). RIVO-DLO-report C.006.98

Wallsberg, G.E. & J.R. King. 1978. The relationship of the external surface of birds to skin surface area and body mass. J. Exp. Biol 76: 185-189

Wilson, R.P., K.Hustler, P.G. Ryan, A.E. Burger & A.C. Nöldeke, 1992. Diving birds in cold water: do Archimedes and Boyle determine energetic costs? Am. Nat. 140: 179-200

Woakes A.J. & P.J. Butler. 1983. Swimming and diving in tufted ducks, *Aythya fuligula*, with particular reference to heart rate and gas exchange. J. Exp. Biol. 107: 311-329

Zwarts, L. & Drent, R.H. 1981. Prey depletion and the regulation of predator density: Oystercatchers (*Haematopus ostralegus*) feeding on mussels (*Mytilus edulis*). In: V.H. Jones & W.J. Wolff (eds) Feeding and survival strategies of estuarine organisms: 193-216. Plenum Press, New York.

Appendix A Description of bird sizes following a cone geometry

When pure spherical, the area A would read:

$$A = 4p\left(\frac{W}{r_{body}}\frac{3}{4p}\right)^{\frac{2}{3}} = 4.836\left(\frac{W}{r_{body}}\right)^{\frac{2}{3}} = 0.04824\,W^{\frac{2}{3}} \quad (m^2) \quad (54)$$

with $?_{body}$ is 1000 kg m^{-3} and W=body mass (kg)

Based on a cone shape the following equations can be used:

$$r = \sqrt{\frac{V}{h}\cdot\frac{3}{2p}} \quad (m) \quad (55)$$

$$A_b = p\,r^2 \quad (m^2) \quad (56)$$

$$A = p\,r\,(r+s) \quad (m^2) \quad (57)$$

$$s = \sqrt{r^2+h^2} \quad (m) \quad (58)$$

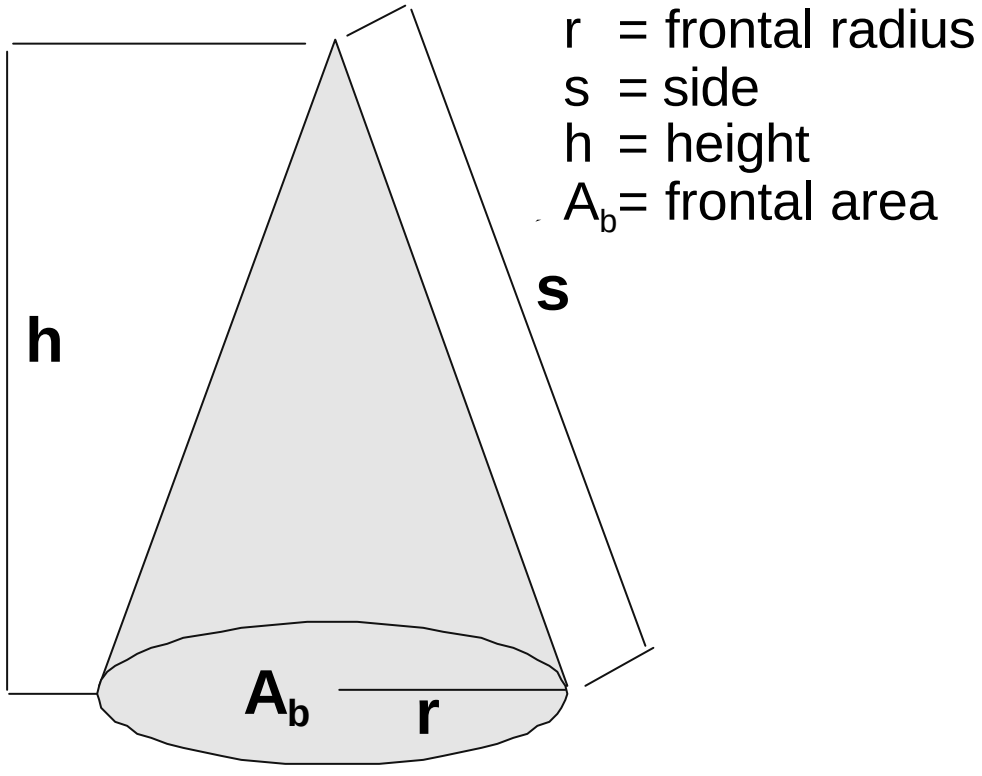


Figure 33. A cone shape to estimate duck characteristics.

For eider ducks, h = c times r, and it seems that for a body mass of about 2000 g, the value for the constant c is about 4 - 5. For h=c r, eq. (55) changes into

$$r = \sqrt[3]{\frac{3V}{2 \cdot c \cdot p}} \quad \text{(m)} \qquad (59)$$

(c is about 4 - 5). The remaining equations stay the same. For other birds, or for smaller sized eider ducks, we used this same approximation. For adult birds, that become meagre in winter, it does not hold, since they do not decrease in length. Thus, for these birds, h stays the same, and r decreases upon starvation, or increases as a result of fattening.

Appendix B Description of the effect of wind on conductive heat transfer from an eider duck to the environment

Heat transfer from a duck's body through the fat and feather layers is a conductive process. Heat transport from the feather's surface to the air is a matter of convection only in those cases where wind is effective enough to keep the feather's temperature exactly equal to the air temperature. This will be the case at higher wind speeds. Under conditions with a low wind speed, heat transfer from the feathers to the air is a combination of conduction and convection. In the extreme case of no wind, convection is purely driven by the density differences between the air and the (slightly) heated air close to the feathers.

As a first guess, we described this effect by the assumption of an extra insulating layer in the air, which a thickness d (m) only depending on the square of the wind speed. Such a dependency of the square of the wind speed is not uncommon. It is often found in descriptions where wind speed effects on heat transfer are at stake (see eg Thomann & Mueller, 1987). Thus:

$$d_{air} = \frac{1}{(a + b.WIND^{c})} \qquad \text{(m)} \qquad (60)$$

with c =2 in many cases.

Thus, a is the value for zero wind speed. Both a and b are estimated very roughly and only serve to supply some relationship between wind conditions and heat loss for a duck.

The solution of this problem is similar to the one of eqs. Figure 13, (16) and (15). The flux through the plane between feathers and fat and the flux between feathers and air is the same and there' s no heat production in the fat or the feather layer. Actually, we have to speak of heat flow equality here (instead of fluxes), because the area of the plane between feathers and air is somewhat larger than the area of the plane between feathers and fat. But for reason of simplicity we assume these to be equal, and thus not only the flows but also the fluxes are equal.
We use again:

$$b = \frac{a_{fat}}{a_{feathers}} \cdot \frac{d_{feathers}}{d_{fat}} \qquad \text{(-)} \qquad (61)$$

and (now T_{feath} in stead of T_{air}, as in eq. (15))

$$T_{skin} = \frac{bT_{body} + T_{envir}}{1 + b} \qquad (^{o}C) \qquad (62)$$

We introduce for convenience:

$$d23 = \frac{d_{feathers}}{d_{air}} \qquad \text{(-)} \qquad (63)$$

and after some conversions, one can find for the temperature at the feather surface:

$$T_{feat} = \frac{\frac{d23 * T_{air}}{(1+d23)} + \frac{Tb}{(1+b)(1+d23)}}{1 - \frac{b}{(1+b)(1+d23)}} \quad (^{o}C) \quad (64)$$

with (62) T_{skin} is known. The heat flux follows from:

$$flux_{heat} = a_{feathers} \cdot \frac{T_{skin} - T_{feath}}{d_{feathers}} \quad (W\ m^{-2}) \quad (65)$$

The total heat loss of the duck follows again from eq.(17). Note that we assumed that the heat loss is uniformly distributed over the body and that there are no spots with very large losses.

We performed some calculations and found ourselves content with the parameter values as mentioned in text box 5. But this is not more than a first try and the values have to be validated.

Text box 5 Parameter values for relationship between wind and heat transfer, for d_{air} in m

A=10
B= 5
C=2

Appendix C Effect of changing buoyancy during descent on diving costs

The buoyancy of a bird with a specific mass of $?_d$, and volume V_d is given by eq. (37). During descent, the specific mass $?_d$ increases because of the compression of the air in the feathers and the respiratory system. The result is given by eqs. (44) and (43). If the specific mass of the birds body is set equal to 1.0 kg dm^{-3} buoyancy equals: :

$$buoy = 9.81 \cdot Vol_{air} \cdot r_{water} \cdot \frac{10}{10 + depth} \quad \text{(N)} \quad (66)$$

with depth in m, the water specific mass ($?_{water}$) in kg m^{-3} and the total volume of air (Vol_{air}) in m^3.
Now, the total work that has to be done to counteract buoyancy during a descent with velocity v_d (m s^{-1}) is:

$$work = \int_{t=0}^{t=te} power \cdot dt \quad \text{(J)} \quad (67)$$

with

$$power = v_d \cdot buoy \quad \text{(W)} \quad (68)$$

Combining eqs. (66) - (68) gives, after substituting $a = 9.81 * Vol_{air} * 10 * ?_{water}$, $b = 10\ v_d^{-1}$ and depth = $v_d\ t$:

$$work = \int_{t0}^{te} \frac{a}{b + t} dt \quad \text{(J)} \quad (69)$$

with a= b/a and ß=1/a work can be described with:

$$work = \left[\frac{1}{b} \ln(a + b\,t) \right]_{t0}^{te} = \frac{1}{b} \ln(a + b\,t_e) - \frac{1}{b} \ln(a) \quad \text{(J)} \quad (70)$$

This formula (70) is implemented in the model.

Appendix D Input data for the eider duck simulation model

D1 Weather data for The Netherlands

```
{========================================================================#
#                           ALTERRA                                      #
#------------------------------------------------------------------------#
#                    Model DEPLETE_Eider                                 #
# DEPLETEEider describes the way eider ducks find and utilize food       #
# and how the food fits their energetic needs                            #
#------------------------------------------------------------------------#
# Alterra          PO Box 167   1790 AD Den Burg-NL                      #
#                  Phone () 31 222 369728                                #
#                  Fax   () 31 222 319235                                #
#------------------------------------------------------------------------#
# Developers       Bert Brinkman       a.g.brinkman@alterra.wag-ur.nl    #
#                                                                        #
#                  Bruno Ens           b.j.ens@alterra.wag-ur.nl         #
#========================================================================#
# File             weather.dat                                           #
#                                                                        #
# Description      Contains weather data                                 #
#                  Called by input routine                               #
#                  Weather and water temp data for the                   #
#                  Dutch Wadden Sea                                      #
# Data                                                                   #
# Created          2002-April-17                                         #
# Last Modified    2002-April-17                                         #
#------------------------------------------------------------------------#
# Called by        EIplIO.PAS routine ReadSystemData                     #
#------------------------------------------------------------------------#
# First lines are reserved for comment.                                  #
# And, each data line has thirty positions free                          #
# for additional comment.                                                #
#                LA ROCHELLE - data                                      #
#                                                                        #
# _comment_________________30|                                          #
#========================================================================}
                            mon    Twat    Tair   wind
data:
month. water. air. wind       1    7.24    5.62    5
month. water. air. wind       2    7.14    8.87    5
month. water. air. wind       3    8.90    10.68   5
month. water. air. wind       4    12.22   12.19   5
month. water. air. wind       5    15.52   16.95   4
month. water. air. wind       6    17.79   18.18   4
month. water. air. wind       7    19.53   21.55   4
month. water. air. wind       8    21.03   20.69   4
month. water. air. wind       9    20.22   19.13   4
month. water. air. wind       10   16.44   14.87   5
month. water. air. wind       11   11.98   8.57    5
month. water. air. wind       12   7.66    8.19    5
```

D2 Bird characteristics

```
{=========================================================================#
#                            ALTERRA                                      #
#-------------------------------------------------------------------------#
#                     Model DEPLETE_Eider                                 #
# DEPLETEEider describes the way eider ducks find and utilize food        #
# and how the food fits their energetic needs                             #
#-------------------------------------------------------------------------#
# Alterra          PO Box 167   1790 AD Den Burg-NL                      #
#                  Phone () 31 222 369728                                 #
#                  Fax   () 31 222 319235                                 #
#-------------------------------------------------------------------------#
# Developers       Bert Brinkman       a.g.brinkman@alterra.wag-ur.nl     #
#                                                                         #
#                  Bruno Ens           b.j.ens@alterra.wag-ur.nl          #
#=========================================================================#
# File             BirdPar.dat                                            #
#                                                                         #
# Description      Bird characteristics data file                         #
#                  Called by input routine                                #
# Data                                                                    #
# Created          2002-January-17                                        #
# Last Modified    2002-April-23                                          #
#-------------------------------------------------------------------------#
# Called by        EIplIO.PAS routine ReadSystemData                      #
#-------------------------------------------------------------------------#
# First lines are reserved for comment.                                   #
# And, each data line has thirty positions free                           #
# for additional comment.                                                 #
#                                                                         #
# _comment_____________________30|                                       #
#=========================================================================}
data:

NameDutch                         Eidereend
NameEngl,                         Common Eider
NameGerm : String;                Eiderente
BMR basic metab rate W/kg         4.05
FlyCosts costs of flying W/kg     150
frictionfact                      0.20
Swima,   Watts                    0.6
Swimb,   x                        0.0
swimTimperDay (s)                 500
flyTimperDay (s)                  900
Botta,   Watts                    0.8
WeightMax upper bound of mass     2400
WeightMin lower bound of mass     1600
WeightAvg mass average            2000
YearWeight_Jan                    2000
YearWeight_Feb                    2000
YearWeight_Mar                    2000
YearWeight_Apr                    2000
YearWeight_Mei                    2000
YearWeight_Jun                    2000
YearWeight_Jul                    2000
YearWeight_Aug                    2000
YearWeight_Sep                    2000
YearWeight_Okt                    2000
YearWeight_Nov                    2000
YearWeight_Dec                    2000
SpecMass eider swimming           0.45
SpecMass eider diving kg/dm3      0.67
```

```
PreySpecMass spec mass prey      1
Temp body temp of eider (oC)     41
SwimmingSpeed m/s                0.3
FySpeed (m/s)                    25
DivingSpeed m/s when diving      1.0
DivingSpeedAmpl                  0.2
DivingAmpFreq                    2.63
BottomSpeed m/s at bottom        0.1
BottomSpeedAmpl,                 0.2
BottomAmpFreq,                   2.62
searchTime0 loose mussel         4
searchTime1 bunched mussel       12
handlingtime loose mussel        4
handlingtime bunched mussel      16
MaxIntakeRate     xxx            0.1   (not active yet)
MaxGrowthRateConstxxx            0.1   (not active yet)
assimilation efficiency          0.8
EnergContentFat  J/gDW           22.5e3
EnergContentAnimal J/gDW         22.5e3
airexhaled before diving         0.20
cone proportionality const       4.0
LungVol_a (lit=dm3, -> cm3)      1.61e-1
LungVol_b (mass (g)^b)           0.91
diving_energetic efficiency      0.35

Now the length-preferences
preynr 1=mussels                 1
LenPref[0-5]    1                0
LenPref[5-10]   2                0
LenPref[10-15]  3                0.1
LenPref[15-20]  4                0.5
LenPref[20-25]  5                1
LenPref[25-30]  6                1
LenPref[30-35]  7                1
LenPref[35-40]  8                1
LenPref[40-45]  9                1
LenPref[45-50]  10               1
LenPref[50-55]  11               1
LenPref[55-60]  12               1
LenPref[60-65]  13               1
LenPref[65-70]  14               0.5
LenPref[70-75]  15               0.2
```

D3 Area characteristics, The Netherlands

```
{====================================================================#
#                          ALTERRA                                   #
#--------------------------------------------------------------------#
#                    Model DEPLETE_Eider                             #
# DEPLETEEider describes the way eider ducks find and utilize food   #
# and how the food fits their energetic needs                        #
#--------------------------------------------------------------------#
# Alterra          PO Box 167   1790 AD Den Burg-NL                  #
#                  Phone () 31 222 369728                            #
#                  Fax   () 31 222 319235                            #
#--------------------------------------------------------------------#
# Developers       Bert Brinkman       a.g.brinkman@alterra.wag-ur.nl #
# Database setup   Bruno Ens           b.j.ens@@alterra.wag-ur.nl     #
# Data provision   Rieneke de Jager                                   #
#                                                                    #
#====================================================================#
# File             AreaCharNL.dat                                    #
#                                                                    #
# Description      Area characteristics data file                    #
#                  Called by input routine                           #
#                  Data for Dutch Wadden Sea                         #
#====================================================================#
# Data                                                               #
# Created          2002-January-17                                   #
# Last Modified    2002-April-17                                     #
#--------------------------------------------------------------------#
# Called by        EIplIO.PAS routine ReadSystemData                 #
#--------------------------------------------------------------------#
# First lines are reserved for comment.                              #
# And, each data line has thirty positions free                      #
# for additional comment.                                            #
# Standard situation                                                 #
#                                                                    #
# _comment_____________30|                                          #
#====================================================================}
___comment_____________30|

data:
number of compartments          1
Number of prey species          1
Number of size classes          15
First simulation day            1
Last simulation day             400
Total population at day0        2.0E+4
Number of population sim's      1
Step size populations           0.0E+4

salt_content (g /gr)            0.035
salt_costs J/g exret            1.5e3

Preynr ([x]=month) mussels      1
rel a-value[1]                  0.5
rel a-value[2]                  0.5
rel a-value[3]                  0.6
rel a-value[4]                  0.7
rel a-value[5]                  0.8
rel a-value[6]                  0.9
rel a-value[7]                  1
rel a-value[8]                  1
rel a-value[9]                  0.9
rel a-value[11]                 0.8
```

```
rel a-value[11]                0.7
rel a-value[12]                0.6

Crush -a (nehls, thru 0,0)     600
Crush-b                        0

Site:                          1
Name                           Site1
x-km position                  120         (No meaning yet)
y-km position                  430         (No meaning yet)
dry-time   (%)                 0
depth (m)                      5
size of area (m2)              2.2e8
Cockles (1) Mussels(2)         1
Loose (0) or bunch (1)         1
flesh alfa-value               9.5e-6
flesh beta-value               2.91
shell=alfa                     4.8e-5
shell-beta                     2.92
fresh weight -alfa             3.5e-4
fresh weight -beta             2.8
water weight -alfa             2.0e-4
water weight -beta             2.7
1 Size class [0-5] nr/m2       0.0
2 Size cl [5-10] nr/m2           0.3
3 Size cl [10-15] nr/m2        1.4
4 Size cl [15-20] nr/m2        3.6
5 Size cl [20-25] nr/m2        6.8
6 Size cl [25-30] nr/m2        11.1
7 Size cl [30-35] nr/m2        12.5
8 Size cl [35-40] nr/m2        7.3
9 Size cl [40-45] nr/m2        4.4
10 Size cl [45-50]nr/m2        3.7
11 Size cl [50-55]nr/m2        2.9
12 Size cl [55-60]nr/m2        1.7
13 Size cl [60-65]nr/m2        0.7
14 Size cl [65-70]nr/m2        0.4
15 Size cl [70-75]nr/m2        0.1
```

D4 System definition file EISysdef.pas

```
{=============================================================================#
#                              ALTERRA                                        #
#-----------------------------------------------------------------------------#
#                    Model DEPLETE_Eider                                      #
# DEPLETE describes the way eider ducks find and utilize food                 #
# and how he food fits their energetic needs                                  #
#-----------------------------------------------------------------------------#
# IBN              PO Box 167   1790 AD Den Burg-NL                           #
#                  Phone () 31 222 369728                                     #
#                  Fax   () 31 222 319235                                     #
#-----------------------------------------------------------------------------#
# Developers       Bert Brinkman       a.g.brinkman@alterra.wag-ur.nl         #
#                                                                             #
#=============================================================================#
# File             EISysDef.PAS                                               #
#                                                                             #
# Description      EISysDef contains all the data types, record               #
#                  descriptions and system definitions.                       #
#                                                                             #
# Created          2002-Januari-17                                            #
# Last Modified    2002-April-23                                              #
#-----------------------------------------------------------------------------#
# Calls            none                                                       #
#-----------------------------------------------------------------------------#
# Used by          All the modules                                            #
#=============================================================================}

UNIT EIsysdef;

INTERFACE

Uses WinDos;

Const   MaxAreas    = 12;
        MaxSim      = 100;
        MaxPreySpec = 3;
        MaxLengths  = 15;
        MeatToWet_conversion = 20.0;
        SecPerDay   = 86400.0;
        Pi          = 3.1416;

Type    DateTime= RECORD
                    year,
                    month,
                    day,
                    dayofweek,
                    hour,
                    minute,
                    second,
                    sec100 : WORD;
                  END;

        Physic_Char
              = RECORD
                    Fat_heatcond,               {* W /m/K fat         *}
                    Wat_heatcond,               {* W /m/K water       *}
                    Air_heatcond,               {* W /m/K air         *}
                    Fat_specheat,               {* J/g /K fat         *}
                    Wat_specheat,               {* J/g /K fat         *}
                    Air_specheat,               {* J/g /K fat         *}
                    Waterdens,                  {* specmass_water     *}
```

```
                   AirDens,                      {* specmass_air       *}
                   windChill_a,                  {* chill-fact a       *}
                   windChill_b,                  {* chill-fact a       *}
                   windChill_c,                  {* chill-fact a       *}
                   gravity  : REAL;              {* versnelling zwkrcht*}
                END;

        Weather_Rec
              = RECORD
                   airTemp,                      {* oC                 *}
                   watTemp,                      {* oC                 *}
                   windSpeed: ARRAY[1..12] OF REAL; {* m/s             *}
                END;


Type    Food  = RECORD                           {* dens used at start *}
                   FoodDens  : ARRAY [1..MaxLengths] OF REAL;
                   DW_a,                         {* a,b for AFDW       *}
                                                 {* read per month     *}
                   DW_b,                         {* in shell           *}
                   Shell_A,                      {* a,b for shell      *}
                   Shell_B,                      {* thickness          *}
                   Fresh_A,                      {* a,b for shell      *}
                   Fresh_b,                      {* fresh masses       *}
                   Water_a,                      {* a,b for            *}
                   Water_b  : REAL;              {* water content shell*}
                END;

Type    AreaChar=
                RECORD
                   Xpos,                         {* km coordinate      *}
                   Ypos,                         {* km coordinate      *}
                   Size,                         {* size in m2         *}
                   depth,                        {* m                  *}
                   DryPeriod   : REAL;           {* fract of time      *}
                   Name        : String;         {* name of site       *}
                   preyKind    : INTEGER;        {* type benthic animal*}
                   prey        : ARRAY[0..MaxPreySpec] OF Food;
                                                 {* only at start      *}
                   musselKind  : INTEGER;        {* 0=loose, 1=bunch   *}
                END;

Type     ShellChar=                              {* some char of shell *}
                RECORD
                   SpecMass,                     {* spec mass of prey  *}
                   SpecHeat,                     {* spec heat of prey  *}
                   EnergyContent,                {* energ content J/gDW*}
                   Salt_Water,                   {* salt content water *}
                                                 {* g NaCl/dm3         *}
                   Salt_energy : REAL;           {* J/g salt excreted  *}
                   a_month  : ARRAY[1.. MaxPreySpec] OF
                              ARRAY[1..12 ] OF REAL;
                                                 {* gives the relative *}
                                                 {* condition factor   *}
                   Crush_a  : ARRAY [1.. MaxPreySpec] OF REAL;
                   Crush_b  : ARRAY [1.. MaxPreySpec] OF REAL;
                   DisgestionEnergy  : REAL;     {* J/kg DW needed for *}
                                                 {* digestion          *}
                   length      : ARRAY[0..MaxLengths] OF REAL;
                END;                             {* intervals lengths  *}
```

```
Type     Duck = RECORD
                  NameDutch,
                  NameEngl,
                  NameGerm : String;
                  BMR,                          {* basic metab rate   *}

                  FlyCosts,                     {* costs of flying W  *}
                  frictionFact,                 {* drag-parameters    *}
                  Swima,
                  Swimb,
                  swimtimPerDay,                {* sec per day swimmi *}
                  flytimPerDay,                 {* secs per day flying*}
                  Botta,                        {* watts per N buoyan *}
                  WeightMax,                    {* upper bound of mass*}
                  WeightMin,                    {* lower bound of mass*}
                  WeightAvg  : REAL;            {* mass average       *}
                  YearWeight : ARRAY[0..12] OF REAL;
                                                {* factor telling us  *}
                                                {* what the normal    *}
                                                {* weight has to be   *}
                  SpecMass_Swim,                {* spec mass kg/dm3swim}
                  SpecMass_Div,                 {* spec mass kg/dm3 div}
                  PreySpecMass,                 {* spec mass of prey  *}
                  BodyTemp,                     {* body temp of eider *}
                  SwimmingSpeed,                {* m/s when swimming  *}
                  FlySpeed,                     {* speed in flight    *}
                  DivingSpeed,                  {* m/s when diving    *}
                  DivingSpeedAmpl,
                  DivingAmpFreq,
                  BottomSpeed,                  {* m/s when at bottom *}
                  BottomSpeedAmpl,
                  BottomAmpFreq,
                  searchTime0,                  {* needed loose mussel*}
                  searchTime1,                  {* bunched mussels    *}
                  handlingTime0,                {* needed loose mussel*}
                  handlingTime1,                {* bunched mussels    *}
                  MaxIntakeRate,                {* max gram/sec intake*}
                  MaxGrowthRateConstant,        {* max rate const  1/d*}
                  assimilationEffic,            {* efficieny of ass.(-)}
                  EnergContentFat,              {* energ content J/gDW*}
                  EnergContentAnimal,           {* energ content J/gDW*}
                  exhPartbefDiv,                {* part of air exhaled*}
                                                {* prior to diving    *}
                  hr_c,                         {* cone proportion.   *}

                  LungVol_a,                    {* a and b for        *}
                  LungVol_b,                    {* respiratory volume *}
                  energ_eff                     {* energ_eff diving   *}
                         : REAL;
                  LenPref : ARRAY[0..MaxPreySpec] OF
                            ARRAY[0..MaxLengths] OF REAL;
                  Interfer: InterFerPar;        {* Interference par's *}
                  a,                            {* max search rate    *}
                  h,                            {* handling time      *}
                  irMin,                        {* Min intake rate    *}
                  normalInRate: REAL;           {* Normal intake rate *}

                END;
```

```
        {************************************************************}
        {* Sysdef contains the area characteristics                 *}
        {* Only those one that do not change as a result of         *}
        {* predation.                                               *}
        {************************************************************}
Type    SysDef  = Record
                    nareas,
                    FirstDay,
                    LastDay,
                    npreyspec,
                    npreylengths: INTEGER;
                    Interfer    : Interference;
                    shell       : ShellChar;
                    typeOfArea  : INTEGER; {* may overrule dryperiods *}
                    Method      : INTEGER;
                    SerialComp  : BOOLEAN; {* serie of many populations*}
                    PreyFactor  : REAL;    {* Prey factor , used to
                                              easily perform scenario
                                              simulatons             *}
                    PopStart,              {* first number of more    *}
                                           {* population computations *}
                    PopStep     : REAL;    {* stepsize population size*}
                    NofSteps    : INTEGER; {* number of simulations   *}
                    Area        : Array[1..MaxAreas] of ^AreaChar;
                    Twat,                  {* read from a table       *}
                    Tair,                  {* read from a table       *}
                    Humair,                {* read from a table       *}
                    WindSpeed   : REAL;    {* read from a table       *}
                  End;

        {************************************************************
        * FoodRes is used to store simulation results for food. Will   *
        * be part of simres-record                                     *
        ************************************************************}
        FoodRes = Record
                    FoodDens,
                    FoodProfit,
                    FoodChoice,
                    FoodPrefChoice,
                    FoodLoss  : ARRAY [1..MaxLengths] OF REAL;
                  END;

        PreyResult =
                  RECORD
                    Ndives    : REAL;
                    preyres   : ARRAY [1..MaxPreySpec] OF FoodRes;
                  END;

        {************************************************************
        * All the results on the energy budget are stored, for each    *
        * time step. The same is done for the heat budget, and the     *
        * time budget                                                  *
        ************************************************************}
        MassBud    =
                  RECORD
                    uptakeShells,
                    faeces,
                    weightIncr  : REAL;
                  END;
```

```
{*************************************************************
* In energy the costs (in Joules) of a process are stored      *
*************************************************************}
EnergyBud  =
          RECORD
            BMR,
            uptakeReal,
            uptakeAss,
            condAir,
            condSwim,
            condDiv,
            heatPrey,
            heatbreath,
            evapbreath,
            crushing,
            digestion,
            divingAccel0,
            divingAccel,
            divingDrag,
            divingBuoy,
            divingAccEff,
            divingDragEff,
            divingBuoyEff,
            divingBott,
            swimming,
            flying,
            saltEx,
            basic      : REAL;
          END;
HeatBud    =
          RECORD
            BMR,
            condAir,
            condSwim,
            condDiv,
            heatPrey,
            heatbreath,
            evapbreath,
            crushing,
            divingAccel0,
            divingAccel,
            divingDrag,
            divingBuoy,
            divingAccEff,
            divingDragEff,
            divingBuoyEff,
            divingBott,
            swimming,
            flying,
            basic      : REAL;
          END;

TimeBud    =
          RECORD
            swimming,
            flying,
            descent,
            bottom,
            ascent,
            handling,
            resting,
            NDives      : REAL;
          END;
```

```
{************************************************************}
{* SimRes contains the computation results                  *}
{************************************************************}
SimRes  = Record
            SimDate     : DateTime; {* Date and time of moment*}
                                    {* of simulation         *}
            RegrNum     : INTEGER;  {* registration number    *}
            Day         : INTEGER;  {* Day number during simul*}
            Date        : DateTime; {* date of day           *}
            ElapsTime   : REAL;     {* elapsed time (seconds) *}
            TotPopulat,             {* Total birds in system  *}
            AvgIntake,              {* Avg(IntakeRate*BirdDens*}
            IntakeRate,             {* Prey number per bird/s *}
            FeedingTime : REAL;     {* Part of the day used   *}
            CompFeedTime,           {* for feeding            *}
            BirdDensity,
            AvgBirdDens,
            BirdNumber,
            AvgBirdNumb : Array[1..MaxAreas] of REAL;
            Area        : Array[1..MaxAreas] of PreyResult;
            TotFood,
            TotPrefFood : ARRAY[1..MaxAreas] OF REAL;
            Energy      : EnergyBud;
            Heat        : HeatBud;
            Time        : TimeBud;
            Mass        : MassBud;

          End;

BirdRes = Record
            Weight,                  {* mass of bird (g)      *}
            StorageWeight,           {* mass of storage       *}
            BufferWeight,            {* buffer size           *}
            FatThickness,            {* thickness fat layer   *}
            bodArea,                 {* area of body alone cm2*}
            lungVol,                 {* repiratory volume     *}
                  {* swimming birds ------------------------*}
            SVol,                    {* total volume cm3      *}
            SairVol,                 {* total air volume      *}
            SfeatVol,                {* airvolume in feathers *}
            Sdfeat,                  {* thickness of feathers *}
            SfeatArea,               {* feathered body area   *}
            SfrontArea,              {* fronal area = feathers*}
                  {* diving birds   ------------------------*}
            DVol,                    {* total volume cm3      *}
            DairVol,                 {* total air volume      *}
            DfeatVol,                {* airvolume in feathers *}
            Ddfeat,                  {* thickness of feathers *}
            DfeatArea,               {* feathered body area   *}
            DfrontArea,              {* fronal area = feathers*}
            weightIncreaseRate,      {* increase rate (input) *}
            IntakeRate  : Real;      {* intake rate of food   *}
          End;

PopRes  = Record
            DDay        : ARRAY[1..MaxSim] OF INTEGER;
                                     {* Depletion Day         *}
            PreyStart,               {* Total Prey at begin and*}
            PreyEnd,                 {* end of simulation     *}
            PopulatStart,            {* Population start and  *}
            PopulatEnd  : ARRAY[1..MaxSim] OF REAL;   {* end  *}
            ComPreyEnd,
            ComPreyStart: ARRAY[1..MaxSim,1..MaxAreas] OF REAL;
          End;
```

```
        FileName = STRING[30];

        SortArr  = ARRAY[1..MaxAreas] OF INTEGER;{* sorting  area     *}
                                                 {* profitability     *}


IMPLEMENTATION
BEGIN
END.
```

D5 Lowest level computation file EIFunct1.pas

```
{=====================================================================#
#                               ALTERRA                               #
#---------------------------------------------------------------------#
#                   Model DEPLETE_Eider                               #
# DEPLETEEider describes the way eider ducks find and utilize food    #
# and how the food fits their energetic needs                         #
#---------------------------------------------------------------------#
# Alterra          PO Box 167   1790 AD Den Burg-NL                  #
#                  Phone () 31 222 369728                             #
#                  Fax   () 31 222 319235                             #
#---------------------------------------------------------------------#
# Developers       Bert Brinkman       a.g.brinkman@alterra.wag-ur.nl  #
#                                                                     #
#=====================================================================#
# File             EIFunct1.PAS                                       #
#                  Contains the detailed process funtions             #
#                                                                     #
# Created          2002-Januari-17                                    #
# Last Modified    2002-April-24                                      #
#---------------------------------------------------------------------#
# Calls            no other routines                                  #
#---------------------------------------------------------------------#
# Called by        Routines from EIfunct2.pas, that combine           #
#                  most of these functions and routines to            #
#                  energybudgets                                      #
#=====================================================================}
UNIT EIFunct1;

INTERFACE

USES bbmath,
     EIsysdef;

{*********************************************************************
# Computation of frontal area and feathered area of a duck ,          #
# plus the volumes of lungs, feathers, etc.                           #
*********************************************************************}
PROCEDURE AreasDuck    (var dck : Duck;
                        var brd : BirdRes);

FUNCTION Shell_Weight (var system: Sysdef;
                           Areanr,
                           preynr:INTEGER;
                           len   : REAL) : REAL;

{*********************************************************************
# Computation of fresh weightess                                      #
*********************************************************************}
FUNCTION Fresh_Weight (var system: Sysdef;
                           Areanr,
                           preynr: INTEGER;
                           len   : REAL    ) : REAL;

FUNCTION Meat_content (var system: Sysdef;
                           Areanr,
                           preynr: INTEGER;
                           len,
                           monfact   : REAL) : REAL;
{*********************************************************************
# Computation of of water content in shells                           #
*********************************************************************}
FUNCTION Water_content(var system: Sysdef;
```

```
                          Areanr,
                          preynr: INTEGER;
                          len   : REAL) : REAL;

{*********************************************************************
# Computation of salt content in shells                              #
*********************************************************************}
FUNCTION Salt_content (var system: Sysdef;
                          water : REAL) : REAL;

FUNCTION Crushing_Energy (var system: Sysdef;
                          Areanr,
                          preynr   : INTEGER;
                          length   : REAL   ) : REAL;

{*********************************************************************
# Computation of energic costs of heating prey (1 piece of prey)     #
*********************************************************************}
FUNCTION HeatingPreyInd_Energy (var freshweight : REAL;
                                    dck         : Duck;
                                var system      : sysdef;
                                    shllchr     : ShellChar) : REAL;


{*********************************************************************
# Computation of energic costs of heating prey                       #
*********************************************************************}
{FUNCTION HeatingPrey_Energy (var Meat_Content: REAL;
                                    dck         : Duck;
                              var   system      : sysdef;
                                    shllchr     : ShellChar;
                              var   birdresult  : BirdRes) : REAL;

{*********************************************************************
# Computation of salt-intake related energy losses . Energy in J/sec #
# It's assumed that it's about 20% of BMR (see Nehls)                #
*********************************************************************}
FUNCTION Salt_Energy ( var salt : REAL;
                       var sd   : sysdef  ) : REAL;

{*********************************************************************
# Computation of energic needs for digestion                         #
*********************************************************************}
FUNCTION Digestion_Energy (var Meat_Content: REAL;
                           var sd          : Sysdef;
                           var dck         : Duck) : REAL;

{*********************************************************************
# Computation of energic benefit from prey                           #
*********************************************************************}
FUNCTION PreyInd_Energy (var Meat_Content: REAL;
                         var sd          : Sysdef;
                         var dck         : Duck) : REAL;


{*********************************************************************
# Computation of energic benefit from prey                           #
*********************************************************************}
{FUNCTION Prey_Energy (var br           : BirdRes;
                           Meat_Content : REAL;
                       var sd           : Sysdef;
                       var dck          : Duck) : REAL;

{*********************************************************************
# Computation of energic benefit from prey                           #
```

```
*************************************************************************}
FUNCTION PreyMass_Energy (var br           : BirdRes;
                          var sd           : Sysdef) : REAL;

{************************************************************************
# Computation of diving energy                                         #
*************************************************************************}
PROCEDURE Diving_Energy (var dck     : Duck;
                         var br      : BirdRes;
                         var ener    : EnergyBud;
                         var heat    : heatBud;
                         var time    : TimeBud;
                         var system  : sysdef;
                             physchr : Physic_Char;
                             areanr  : INTEGER);


{************************************************************************
# Computation of energy costs at bottom                                #
*************************************************************************}
PROCEDURE Bottom_Energy (var sd        : Sysdef;
                             dck       : Duck;
                         var brd       : Birdres;
                         var heat      : HeatBud;
                         var ener      : EnergyBud;
                         var time      : TimeBud;
                         var physCh    : Physic_Char;
                             areanr    : INTEGER);

{************************************************************************
# Computation of swimming energy costs                                 #
*************************************************************************}
FUNCTION Swimming_Energy (var dck : Duck) : REAL;

{************************************************************************
# Computation of breathing energy losses                               #
*************************************************************************}
FUNCTION Breathing_Energy(     energy: REAL;
                           var phys  : Physic_Char;
                           var dck   : Duck;
                           var sd    : sysdef  )  : REAL;

{************************************************************************
# Computation of time needed to rest after a dive                      #
*************************************************************************}
PROCEDURE Resting_Time (var dck    : Duck;
                        var system : sysdef;
                        var time   : TimeBud;
                            nrarea : INTEGER);

{************************************************************************
# Computation of time needed to  dive                                  #
*************************************************************************}
{FUNCTION Diving_Time (var dck    : Duck;
                       var system : sysdef;
                           nrarea : INTEGER) : REAL;

{************************************************************************
# Computation of time needed to search at bottom                       #
*************************************************************************}
{FUNCTION Bottom_Time (musselknd : INTEGER;
                       dck       : Duck) : REAL;

{************************************************************************
# Computation of time needed to search at bottom                       #
```

```
*************************************************************************}
FUNCTION Handling_Time (    musselknd : INTEGER;
                        var dck       : Duck;
                        var system    : Sysdef;
                            nrarea    : INTEGER) : REAL;

{*************************************************************************
# Computation of time needed to digest the prey                         #
*************************************************************************}
FUNCTION Digestion_Time (flesh    : REAL;
                         dck      : Duck) : REAL;

{*************************************************************************
# Computation of flying energy costs                                    #
*************************************************************************}
FUNCTION Flying_Energy (flytime : REAL;
                        dck     : Duck) : REAL;

{*************************************************************************
# Computation of fat volume eider feathers                              #
# weight of bird is in grams                                            #
*************************************************************************}
FUNCTION Eider_FatVolume (var dck :Duck;
                          var br  : BirdRes)  : REAL;

{*************************************************************************
# Computation of fat thickness eider feathers                           #
# weight of bird is in grams                                            #
*************************************************************************}
PROCEDURE Eider_FatThick (var dck :Duck;
                          var br  : BirdRes);

{*************************************************************************
# Computation of energic costs of heating                               #
*************************************************************************}
FUNCTION HeatingEider_Energy (var physch : Physic_Char;
                                  dck    : Duck;
                              var br     : BirdRes;
                              var sd     : Sysdef;
                                  pos    : INTEGER) : REAL;


{*************************************************************************
# Computation of maximum uptake volume rate                             #
# It is assumed that an eider cannot take up more than a certain        #
# volume of prey per unit of time. This sets the limits to the prey     #
# to be caught.                                                         #
*************************************************************************}
FUNCTION MaxVol_Uptake  : REAL;

{*************************************************************************
# Computation of maximum uptake rate                                    #
# An eider can be meager, and be at the lower level of its weight       #
# In that case, the possible uptake rate is maximal. At the other       #
# hand, the animal can be fat. In that case; only the daily energy      #
# costs have to be covered.                                             #
*************************************************************************}
FUNCTION MaxMass_Uptake  : REAL;

{*************************************************************************
# An eider has a certain strategie for its weight development           #
# during the year. This cannot be computed here; it's the result of     #
# of the average weight, and the yearweightfactor. This determines      #
# what the preferred mass ingestion and assimilation rate is            #
*************************************************************************}
```

```
FUNCTION DesiredWeightdevelopment (var  br   : BirdRes;
                                        dck  : Duck;
                                        date : datetime) : REAL;


IMPLEMENTATION

{*********************************************************************
# Computation of frontal area and feathered area of a duck            #
*********************************************************************}
PROCEDURE AreasDuck    (var dck : Duck;
                        var brd : BirdRes);
VAR a,b,c,h,r,s, f, nill,
    buoy,wei,hr_c,y,pp       : REAL;
BEGIN
  wei          := brd.weight;                      {* grams           *}
  hr_c         := dck.hr_c;                        {* (-) proport cons*}
  y            := (wei*3.0/2.0/hr_c/Pi);
  pp           := 0.33;
  r            := power(y,pp);                     { radius, cm      *}
  h            := r * hr_c;                        {* cone length     *}
  s            := sqrt(h*h+r*r);                   {* cone side length*}

  brd.lungVol  := dck.LungVol_a
                  * power(wei,dck.LungVol_b);      {* cm3             *}
  brd.bodArea  := 2*pi*r*(r+s);                    {* body area no fea*}
  {*****************************************************************}
  {* swimming birds first                                          *}
  {*****************************************************************}
  brd.SVol     := brd.weight / dck.specMass_Swim;  {* gram / kg/dm3   *}
                                                   {* bird vol in cm3 *}
  brd.SairVol  := brd.SVol - brd.weight;           {* cm3             *}
  buoy         := brd.SairVol;
  brd.SfeatVol := brd.SairVol
                  - (1.0+dck.exhPartbefDiv)
                  * brd.lungVol;                   {* cm3             *}

  a            := pi*r+pi*s/2.0;
  b            := pi*r*r+pi*r*s;                   {* the cone is only*}
  c            := - brd.SfeatVol/2.0;              {* taken half!!    *}
  brd.Sdfeat   := (-b+sqrt(b*b-4*a*c))/2/a;        {* better est dfeat*}
  f            := brd.Sdfeat;
  brd.SfeatArea:= 2*pi*(r+f)*(r+f+s);              {* cm2             *}
  brd.SfrontArea:= pi*(r+f)*(r+f);                 {* cm2             *}
  {*****************************************************************}
  {* diving birds next                                             *}
  {*****************************************************************}
  brd.DVol     := brd.weight / dck.specMass_Div;   {* gram / kg/dm3   *}
                                                   {* bird vol in cm3 *}
  brd.DairVol  := brd.DVol- brd.weight;            {* cm3             *}
  buoy         := brd.DairVol;

  brd.DfeatVol := brd.DairVol - brd.lungVol ;      {* cm3             *}

  a            := pi*r+pi*s/2.0;
  b            := pi*r*r+pi*r*s;                   {* the cone is only*}
  c            := - brd.DfeatVol/2.0;              {* taken half!!    *}
  brd.Ddfeat   := (-b+sqrt(b*b-4*a*c))/2/a;        {* better est dfeat*}
  f            := brd.Ddfeat;
  brd.DfeatArea:= 2*pi*(r+f)*(r+f+s);              {* cm2             *}
  brd.DfrontArea:= pi*(r+f)*(r+f);                 {* cm2             *}
  nill:=0.0;
  a:=1/nill;  }
END;    {* Now areas and thichness have been computed               }
```

```
{************************************************************************
# Computation of shell thickness                                       #
************************************************************************}
FUNCTION Shell_Weight (var system: Sysdef;
                           Areanr,
                           preynr: INTEGER;
                           len   : REAL    ) : REAL;
VAR a,b,result: REAL;
BEGIN
  a     :=  system.Area[Areanr]^.prey[preynr].Shell_A;
  b     :=  system.Area[Areanr]^.prey[preynr].Shell_B;
  result:= a* power (len,b);
  Shell_Weight := result;
END;

{************************************************************************
# Computation of fresh weightess                                       #
************************************************************************}
FUNCTION Fresh_Weight (var system: Sysdef;
                           Areanr,
                           preynr: INTEGER;
                           len   : REAL    ) : REAL;
VAR a,b,result: REAL;
BEGIN
  a     :=  system.Area[Areanr]^.prey[preynr].Fresh_A;
  b     :=  system.Area[Areanr]^.prey[preynr].Fresh_B;
  result:= a* power (len,b);
  Fresh_Weight := result;
END;

{************************************************************************
# Computation of shell flesh content  AFDW                             #
************************************************************************}
FUNCTION Meat_content (var system: Sysdef;
                           Areanr,
                           preynr: INTEGER;
                           len,
                           monfact: REAL) : REAL;
VAR a,b,result: REAL;
BEGIN

  a     :=  system.Area[Areanr]^.prey[preynr].DW_A *monfact  ;
  b     :=  system.Area[Areanr]^.prey[preynr].DW_B  ;

  result:= a* power (len,b);

  Meat_content := result;
END;

{************************************************************************
# Computation of of water content in shells                            #
************************************************************************}
FUNCTION Water_content (var system: Sysdef;
                           Areanr,
                           preynr: INTEGER;
                           len   : REAL) : REAL;
VAR a,b,result: REAL;
BEGIN
  a     :=  system.Area[Areanr]^.prey[preynr].Water_a  ;
  b     :=  system.Area[Areanr]^.prey[preynr].Water_b  ;

  result:= a* power (len,b);

  Water_content := result;
```

```
END;

{*********************************************************************
# Computation of salt content in shells                              #
*********************************************************************}
FUNCTION Salt_content (var system: Sysdef;
                           water : REAL) : REAL;
VAR a,b,result: REAL;
BEGIN
  a     :=  system.shell.Salt_Water;
  result:=  a * water;

  Salt_content := result;
END;

{*********************************************************************
# Computation of shell thickness                                     #
*********************************************************************}
FUNCTION Crushing_Energy (var system: Sysdef;
                           Areanr,
                           preynr   : INTEGER;
                           length   : REAL    ) : REAL;
VAR a,b,result: REAL;
BEGIN
  a     :=  system.shell.Crush_A[preynr];
  b     :=  system.shell.Crush_B[preynr];

  result:= a* Shell_Weight (system,Areanr,preynr,
                           length) +b;
  Crushing_Energy := result;
END;

{*********************************************************************
# Computation of energic costs of heating prey (1 piece of prey)     #
*********************************************************************}
FUNCTION HeatingPreyInd_Energy (var freshweight : REAL;
                                    dck         : Duck;
                                var system      : sysdef;
                                    shllchr     : ShellChar) : REAL;
VAR result,shell,
    tempdiff    :REAL;
BEGIN
  shell         := freshweight;                {  gram fresh-weight   *}
  tempdiff      := dck.bodytemp- system.Twat;  {* T-diff water-duck  *}

  result        := ShllChr.SpecHeat * shell
                   *tempdiff    ;
  HeatingPreyInd_Energy := result;
END;

{*********************************************************************
# Computation of energic costs of heating prey                       #
*********************************************************************}
FUNCTION HeatingPrey_Energy (var Meat_Content: REAL;
                                    dck         : Duck;
                               var  system      : sysdef;
                                    shllchr     : ShellChar;
                               var  birdresult  : BirdRes) : REAL;
VAR result,indheating :REAL;
BEGIN
  indheating    :=
HeatingPreyInd_Energy(Meat_Content,dck,system,shllchr);
  {* product of intakerate * heating per individual                 *}
  result        := BirdResult.IntakeRate
                   * indheating;
```

```
  HeatingPrey_Energy := result;
END;

{*********************************************************************
# Computation of energic needs for digestion                         #
*********************************************************************}
FUNCTION Digestion_Energy (var Meat_Content: REAL;
                           var sd          : Sysdef;
                           var dck         : Duck) : REAL;
VAR result: REAL;
BEGIN
  result        := sd.shell.DisgestionEnergy * Meat_Content
                   *dck.assimilationEffic  ;
  Digestion_Energy   := result;
END;

{*********************************************************************
# Computation of energic benefit from prey                           #
*********************************************************************}
FUNCTION PreyInd_Energy (var Meat_Content: REAL;
                         var sd          : Sysdef;
                         var dck         : Duck) : REAL;
VAR result: REAL;
BEGIN
  result        := sd.shell.EnergyContent * Meat_Content
                   *dck.assimilationEffic  ;
  PreyInd_Energy:= result;
END;

{*********************************************************************
# Computation of energic benefit from prey                           #
*********************************************************************}
FUNCTION Prey_Energy (var br           : BirdRes;
                          Meat_Content : REAL;
                      var sd           : sysdef;
                      var dck          : Duck) : REAL;
VAR result: REAL;
BEGIN
  {*****************************************************************}
  {* BirdResInd.IntakeRate is in prey-numbers per second          *}
  {*****************************************************************}
  result        := br.IntakeRate
                   * PreyInd_Energy(Meat_Content,sd,dck) ;
  Prey_Energy   := result;
END;

{*********************************************************************
# Computation of energic benefit from prey                           #
*********************************************************************}
FUNCTION PreyMass_Energy (var br           : BirdRes;
                          var sd           : Sysdef) : REAL;
VAR result: REAL;
BEGIN
  {*****************************************************************}
  {* BirdRes.IntakeRate is in mass meat per second               *}
  {*****************************************************************}
  result            := br.IntakeRate * sd.shell.energyContent  ;
  PreyMass_Energy   := result;                  {* Watts          *}
END;

{*********************************************************************
# Computation of diving energy in Joules per dive                    #
*********************************************************************}
PROCEDURE Diving_Energy (var dck     : Duck;
                         var br      : BirdRes;
```

```
                        var ener    : EnergyBud;
                        var heat    : heatBud;
                        var time    : TimeBud;
                        var system  : sysdef;
                            physchr : Physic_Char;
                            areanr  : INTEGER);
VAR ds,dsmin,dsmax,
    accelperiod,
    acceler0,
    accelerdiv,
    diep, rho,
    duckairvol, dragen,
    drag,frontarea,f,
    divspeed,kinener,
    te,a,b,alfa,beta,
    W0,We, work,power : REAL;
BEGIN
  {*********************************************************************
  * For all the computations: mind the units!! brd.Dvol eg is in cm3   *
  * brd.weight in grams                                                *
  *********************************************************************}
  ds            := Dck.divingspeed;
  dsmin         := Dck.divingspeed-Dck.DivingSpeedAmpl;
  dsmax         := Dck.divingspeed+Dck.DivingSpeedAmpl;
  accelperiod   := 1.0 / Dck.DivingAmpFreq;
  acceler0      := 0.5* br.Weight/1e3 * ds * ds;{* initial accela     *}
  ener.divingAccel0:= acceler0 / dck.energ_eff; {* Joules            *}
  heat.divingAccel0  := (1.0-dck.energ_eff)     {* not effectively    *}
                  * ener.divingAccel0;          {* used --> heat (J)  *}
  {*********************************************************************}
  {* now the acceleration energy at the start is known                 *}
  {*********************************************************************}
  accelerdiv    := 0.5* br.Weight/1e3
                       *(dsmax*dsmax
                         -dsmin*dsmin);         {* acc. during diving *}
  ener.divingAccel:= accelerdiv / dck.energ_eff;{* Watts              *}
  heat.divingAccel  := (1.0-dck.energ_eff)      {* not effectively    *}
                  * ener.divingAccel;           {* used --> heat (W)  *}

  {*********************************************************************}
  {* now the acceleration energy is known                              *}
  {*********************************************************************}
  diep          := system.area[areanr]^.depth;  {* system depth on    *}
                                                {* site areanr m      *}
  frontarea     := br.DfrontArea/1.0e4;         {* m2 frontal area    *}
  f             := Dck.frictionFact;            {* ==Cd               *}

  rho           := physchr.WaterDens;           {* spec mass wat kg/m3*}
  divspeed      := Dck.divingSpeed;             {* m/s diving         *}
  kinEner       := 0.5*rho*divspeed*divspeed;   {* kinetic cont water *}

  drag          := f*frontarea*kinEner;         {* force needed for   *}
                                                {* diving             *}
  dragEn        := drag * divspeed;             {* Watts              *}

  ener.divingDrag:=dragEn / Dck.energ_eff;      {* produced Watts     *}
  heat.divingDrag:=(1.0-Dck.energ_eff)          {* converted into     *}
                  * ener.divingDrag;            {* heat  Watts        *}

  {*********************************************************************
  * Next, the effect of buoyancy is computed                           *
  * Here, the effect of a changing buoyancy with depth is computed     *
  * See appendix C of the eider duck report                            *
  * We assume a constant diving speed                                  *
  *********************************************************************}
```

```
  duckairvol    := br.DAirVol/1.0e6;            {* cm3/ 1e6  =m3      *}
  te            := diep / divspeed;             {* duration s         *}
  a             := 10.0 * physchr.gravity
                   *duckairvol
                   *physchr.WaterDens ;         {* m/s2 / m3          *}
  b             := 10.0 / divSpeed;
  alfa          := b/a;
  beta          := 1.0/a;
  W0            := 1.0/beta * ln(alfa);
  We            := 1.0/beta * ln(alfa+beta*te);
  work          := We-W0;                       {* Joules             *}
  power         := work / te;                   {* Watts              *}
  ener.divingBuoy:= work / Dck.energ_eff;       {* energy asked       *}
  heat.divingBuoy:= (1.0 - Dck.energ_eff )
                    * ener.divingBuoy;          {* heat produced      *}
  time.descent  := te;                          {* store diving time  *}
  time.ascent   := te*0.7;                      {* store ascent time  }
  {*********************************************************************}
  { The amounts computed above partly are Watts, and partly Joules.   *}
  {* since we are interested in Joules per dive, and later on Joules  *}
  {* per day, all power are converted into Joules                      *}
  {*********************************************************************}
  ener.divingAccel:= ener.divingAccel * te;     {* Joules             *}
  heat.divingAccel:= heat.divingAccel * te;     {* Joules             *}
  ener.divingDrag := ener.divingDrag  * te;     {* produced Joules    *}
  heat.divingDrag := heat.divingDrag  * te;     {* produced Joules    *}
  ener.divingBuoy := ener.divingBuoy  * te;     {* energy asked       *}
  heat.divingBuoy := heat.divingBuoy  * te;     {* produced heat J    *}
END;

{*********************************************************************
# Computation of energy costs at bottom.                             #
*********************************************************************}
PROCEDURE Bottom_Energy (var sd        : Sysdef;
                             dck       : Duck;
                         var brd       : Birdres;
                         var heat      : HeatBud;
                         var ener      : EnergyBud;
                         var time      : TimeBud;
                         var physCh    : Physic_Char;
                             areanr    : INTEGER );
VAR result,depth,
    searchtime,yy,
    buoy,power     : REAL;
BEGIN
  IF sd.area[areanr]^.musselKind=0
  THEN
    searchtime  := Dck.searchTime0
  ELSE
    searchtime  := Dck.searchTime1;

  time.bottom   := searchtime;
  depth         := sd.Area[areanr]^.depth;{* meters depth          *}
  buoy          := brd.DairVol          {* cm3                     *}
                   *physCh.gravity      {* zwaartekrachtversnelling *}
                   /1.0e3               {* naar dm3 (rho-wat=1)     *}
                   *10.0/(10.0+depth);  {* diepte-correctie         *}
  {*********************************************************************
  * Buoyancy is corrected for depth                                   **
  *********************************************************************}
  power         := buoy * dck.Botta;    {* N * W/N-> W              *}
  heat.divingBott := (1-dck.energ_eff)
                   * power
                   / dck.energ_eff;     {* only losses              *}
  ener.divingBott := heat.divingBott;   {* W                        *}
```

```
  heat.divingBott := heat.divingBott * searchTime; {* Joules          *}
  ener.divingBott := ener.divingBott * searchTime; {* Joules          *}

  result        := 6.0 * searchTime  ;  {* demand = 6 Watts          *}
END;

{*********************************************************************
# Computation of salt-intake related energy losses . Energy in J/sec   #
# It's assumed that it's about 20% of BMR (see Nehls)                  #
# Note that bmr is in watts/kg, and bird-weight in grams !!            #
*********************************************************************}
FUNCTION Salt_Energy ( var salt : REAL;
                       var sd   : sysdef  ) : REAL;
VAR i,j    : INTEGER;
    result : REAL;
BEGIN
  result := salt * sd.shell.Salt_energy;            {* Joules         *}
  Salt_Energy := result;
END;

{*********************************************************************
# Computation of swimming energy costs   (Watts)                       #
*********************************************************************}
FUNCTION Swimming_Energy (var dck : Duck) : REAL;
VAR result: REAL;
BEGIN

  result          := dck.Swima   ;                  {* Watts          *}
  Swimming_Energy := result;
END;

{*********************************************************************
# Computation of breathing energy losses . Energy in J/sec             #
# It's 20 kJ / gram DW - > 20 kJ / gram O2                             #
*********************************************************************}
FUNCTION Breathing_Energy ( energy: REAL;
                            var phys: Physic_Char;
                            var dck : Duck;
                            var sd  : sysdef  ) : REAL;
VAR gramO2ps,
    volO2ps,
    volair,
    volbreath,
    Tdiff,
    result: REAL;
BEGIN
  gramO2ps   := energy  /20.0e3;                 {* W                 *}
  volO2ps    := gramO2ps/phys.AirDens;           {* airdens = 1.3g/l  *}
  volair     := volO2ps * 5.0;                   {* 20% of air = O2   *}
  volbreath  := volair  * 5.0;                   {* only 20% of oxygen*}
                                                 {* is used each time *}
  Tdiff      := dck.BodyTemp - sd.Tair;          {* temp difference   *}
  result     := TDiff * volbreath * phys.Air_specheat;
                                                 {* J/s               *}
  Breathing_Energy := result;
END;

{*********************************************************************
# Computation of time needed to rest after a dive                      #
*********************************************************************}
PROCEDURE Resting_Time (var dck    : Duck;
                        var system : sysdef;
                        var time   : TimeBud;
                            nrarea : INTEGER);
VAR result: REAL;
```

```
BEGIN
  time.resting    := (time.descent + time.ascent+time.bottom)
                     *0.75;               {* guessed after Nehls p36 *}
END;

{*********************************************************************
# Computation of time needed to  dive                                 #
*********************************************************************}
FUNCTION Diving_Time (var dck    : Duck;
                      var system : sysdef;
                          nrarea : INTEGER) : REAL;
VAR a,b,result: REAL;
BEGIN
  a               := system.area[nrarea]^.depth; {* local depth        *}
  b               := dck.divingspeed;           {* diving speed duck  *}
  result          := a/b   ;                    {* diving time        *}
  Diving_Time := result;
END;

{*********************************************************************
# Computation of time needed to search at bottom                      #
*********************************************************************}
{FUNCTION Bottom_Time (musselknd : INTEGER;
                       dck       : Duck) : REAL;
VAR result: REAL;
BEGIN
  IF musselknd =0
  THEN
    result := dck.searchtime0
  ELSE
    result := dck.searchtime1;

  Bottom_Time := result;
END;   }

{*********************************************************************
# Computation of time needed to search at bottom                      #
*********************************************************************}
FUNCTION Handling_Time (    musselknd : INTEGER;
                        var dck       : Duck;
                        var system    : Sysdef;
                            nrarea    : INTEGER) : REAL;
VAR result: REAL;
BEGIN
  IF musselknd =0
  THEN
    result := dck.handlingtime0
  ELSE
    result := dck.handlingtime1;

  Handling_Time := result;
END;

{*********************************************************************
# Computation of time needed to digest the prey                       #
# Note: NOT used; handling time covers this part                      #
*********************************************************************}
FUNCTION Digestion_Time (flesh    : REAL;
                         dck      : Duck) : REAL;
VAR result: REAL;
BEGIN
  result := 0.0;         {<<vooralsnog: handling time covert het  <<<<<}

  Digestion_Time := result;
END;
```

```
{************************************************************************
# Computation of flying energy costs                                    #
************************************************************************}
FUNCTION Flying_Energy (flytime : REAL;
                        dck     : Duck) : REAL;
VAR result,y1,y2: REAL;
BEGIN
  result        := Dck.flycosts * flytime   ;
  Flying_Energy := result;
END;



{************************************************************************
# Computation of fat volume eider feathers                              #
# weight of bird is in grams                                            #
************************************************************************}
FUNCTION Eider_FatVolume (var dck :Duck;
                          var br  : BirdRes)  : REAL;
VAR result,
    fatdensity,
    y1          : REAL;
BEGIN
  fatdensity    := 1.0;
  y1            := Br.StorageWeight;     {* fatweight                *}
  result        := y1 / fatdensity;      {* fat volume cm3           *}

  Eider_FatVolume := result;
END;

{************************************************************************
# Computation of fat thickness eider feathers                           #
# weight of bird is in grams                                            #
************************************************************************}
PROCEDURE Eider_FatThick (var dck :Duck;
                          var br  : BirdRes);
VAR result,y1,y2: REAL;
BEGIN
  y1 := br.bodArea;                     {* body area (no feat) cm2   *}
  result         := Eider_FatVolume(dck,br)
                    / y1;               {* fat volume cm3           *}
  br.FatThickness:= result;             {* cm                       *}
END;

{************************************************************************
# Computation of energic costs of heating                               #
************************************************************************}
FUNCTION HeatingEider_Energy (var physch : Physic_Char;
                                  dck    : Duck;
                              var br     : BirdRes;
                              var sd     : Sysdef;
                                  pos    : INTEGER) : REAL;
VAR result,b,
    temp,
    Tskin,
    heatflux,yy,
    wind,zz,xx,
    d2d3,a1,a2,Tb,Tfeat,
    dfeat, Tenv,
    Askin,Afeat,Area,
    heatflow    : REAL;
BEGIN
  IF pos =1                                 {* thick air layer   *}
```

```
  THEN                                          {* in water          *}
    yy   := 1.0e4
  ELSE                                          {* in air            *}
    yy   := physch.WindChill_a
          * power (sd.windspeed,physch.WindChill_b);
  zz   := 1.0/yy;                               {* thickness air layer}

  Eider_FatThick    (dck,br);                   {* thickness fat layer}
  xx   := br.FatThickness*1e-2;                 {* m                 *}
  IF pos =0                                     {* thickness feathers*}
  THEN                                          {* in air            *}
    yy   := br.Sdfeat*0.01
  ELSE                                          {* in water          *}
    yy   := br.Ddfeat*0.01;

  a1   := physch.Fat_heatcond;
  a2   := physch.Air_heatcond;
  b    := a1 / a2 * yy / xx;                    {* b-value           *}

  d2d3 := yy / zz;                              {* d2 / d3           *}
  Tb   := dck.bodyTemp;

  IF pos =0                                     {* temp environment  *}
  THEN                                          {* in air            *}
    Tenv:= sd.Tair
  ELSE                                          {* in water          *}
    Tenv:= sd.Twat;

  {*********************************************************************
  * Note that fat and feather thicknesses were in cm, but since        *
  * they appear both, it's OK                                          *
  *********************************************************************}
  Tfeat := (d2d3*Tenv/(1+d2d3)+Tb/(1+b)/(1+d2d3)){* feat surface temp *}
          /(1-b/(1+b)/(1+d2d3));

  Tskin         := (b* Tb + Tfeat)/(1.0+b);     {* skin temp oC      *}

  heatflux      := a2                           {* W /m /K           *}
                   *(Tskin - Tfeat)             {* oC                *}
                   /yy;                         {* feat thick in m   *}

  ASkin         := br.bodArea*1e-4;             {* m2                *}
  IF pos=0
  THEN
    Afeat       := br.SFeatArea*1e-4
  ELSE                                          {* in water          *}
    Afeat       := br.DFeatArea*1e-4;
  Area          := (Askin + Afeat)/2.0;         {* aver. surface cm2 *}

  heatflow      := heatflux *Area ;             {* flux = W/m2       *}
                                                {* area in m2        *}
  HeatingEider_Energy := heatflow;              {* W per animal      *}
END;



{*********************************************************************
# Computation of maximum uptake volume rate                          #
# It is assumed that an eider cannot take up more than a certain     #
# volume of prey per unit of time. This sets the limits to the prey  #
# to be caught.                                                      #
*********************************************************************}
FUNCTION MaxVol_Uptake  : REAL;
VAR result,y1,y2: REAL;
BEGIN
```

```
  result        := 0.0   ;
{<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<invullen <<<<<}
  MaxVol_Uptake := result;
END;


{***********************************************************************
# Computation of maximum uptake rate                                   #
# An eider can be meager, and be at the lower level of its weight      #
# In that case, the possible uptake rate is maximal. At the other      #
# hand, the animal can be fat. In that case; only the daily energy     #
# costs have to be covered.                                            #
***********************************************************************}
FUNCTION MaxMass_Uptake  : REAL;
VAR result,y1,y2: REAL;
BEGIN

  result        :=  0.0  ;
{<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<invullen <<<<<}
  MaxMass_Uptake := result;
END;


{***********************************************************************
# An eider has a certain strategie for its weight development          #
# during the year. This cannot be computed here; it's the result of    #
# of the average weight, and the yearweightfactor. This determines     #
# what the preferred mass ingestion and assimilation rate is           #
***********************************************************************}
FUNCTION DesiredWeightdevelopment (var  br   : BirdRes;
                                        dck  : Duck;
                                        date : datetime) : REAL;
VAR result,
    presentweight,
    desiredweight,
    diff,
    specrate    : REAL;
BEGIN
  presentWeight := br.weight;
  desiredweight := dck.weightavg *dck.yearweight[date.month];
  diff          := desiredWeight- presentWeight;
  specrate      := diff/ presentWeight;
  {*********************************************************************
  * We assume first that the bird tries to reach this difference       *
  * But, the rate cannot be larger or smaller than the maximum growth  *
  * rate. Only if the bird cannot find food, the growth rate can be    *
  * lower.                                                             *
  *********************************************************************}
  IF specrate > dck.maxgrowthrateConstant
  THEN
    diff := dck.maxgrowthrateConstant ;

  IF diff < - dck.maxgrowthrateConstant
  THEN
    diff := - dck.maxgrowthrateConstant;
  {*********************************************************************
  * Now we know what the birds wants to do. Next is to estimate        *
  * whether it can reach this target; ie: is there enough food?        *
  *********************************************************************}
  DesiredWeightdevelopment := diff;
END;

END. {* of Unit EIFunct1.pas *}
```

D6 Level 2 computation file EIFunct2.pas

```
{=======================================================================#
#                               ALTERRA                                 #
#-----------------------------------------------------------------------#
#                    Model DEPLETE_Eider                                #
# DEPLETEEider describes the way eider ducks find and utilize food      #
# and how the food fits their energetic needs                           #
#-----------------------------------------------------------------------#
# Alterra          PO Box 167   1790 AD Den Burg-NL                    #
#                  Phone () 31 222 369728                               #
#                  Fax   () 31 222 319235                               #
#-----------------------------------------------------------------------#
# Developers       Bert Brinkman       a.g.brinkman@alterra.wag-ur.nl   #
#                                                                       #
#=======================================================================#
# File             EIFunct2.PAS                                         #
#                  Contains the computation of the enrgy budget         #
#                                                                       #
# Created          2002-Januari-17                                      #
# Last Modified    2002-April-17                                        #
#-----------------------------------------------------------------------#
# Calls            The detail-functions from EIfunct1.pas               #
#-----------------------------------------------------------------------#
# Called by        Routine TotPopulation from RatesAndDensities         #
#                  in file EIRatDen.pas                                 #
#=======================================================================}
UNIT EIFunct2;

INTERFACE

USES bbmath,
     eisysdef,
     eifunct1,
     eiplio;
VAR flesh,
    ener_inp,
    heat_inp,
    shellw,freshw,
    waterw,saltw,
    heatprey,
    accel0en,
    accel0het,
    acceleng,
    accelheat,
    drageng,
    dragheat,
    dbuoyeng,
    dbuoyheat,
    botteng,
    bottheat,
    crush,
    heatprod,
    digestion,
    AirCool,H2OCool,
    ExtraCool,
    BreathBMR,BreathFod,
    ExtraBreat,
    saltcost,
    bottomtime,
    restingtime,
    divingtime,
    handlingtime,
    digestiontime,
    tot_time,underw_time,
    energy,
```

```
    heat,
    energain,
    extraenergy,
    en_rate        : ARRAY[1..MaxLengths] OF REAL;

PROCEDURE Compute_BasicEnergy
                         (var sd        : Sysdef;
                          var dck       : Duck;
                          var res       : SimRes;
                          var bird      : BirdRes;
                          var physics   : Physic_Char;
                          var detailfile: text;
                          {var breath,air: REAL; }
                              detail    : INTEGER   );

PROCEDURE Check_Profitability
                        (var sd    : sysdef;
                         var detfil: text;
                         var dck   : Duck;
                         var br    : BirdRes;
                             physch: Physic_Char;
                         var sim   : SimRes ;
                             detail: INTEGER     ) ;

PROCEDURE Area_FoodValue (var sd        : Sysdef;
                          var dck       : Duck;
                          var res       : SimRes;
                          var bird      : BirdRes;
                          var physics   : Physic_Char;
                          var detailfile: text;
                              detail    : INTEGER   );

{*********************************************************************
# Computation food value of an area                                  #
*********************************************************************}
PROCEDURE FindBestArea (var nrf,nrfpref: INTEGER;
                             sd         : sysdef;
                         var res        : Simres      );


{*********************************************************************
* In order to find the number of dives necessary, the basis energy    **
* demand and the demand related to dives has to be known              **
*********************************************************************}
PROCEDURE Compute_NumberOfDives
                        (var sd    : sysdef;
                         var detfil: text;
                         var dck   : Duck;
                         var br    : BirdRes;
                             physch: Physic_Char;
                         var sim   : SimRes ;
                             detail,
                             sitenr: INTEGER;
                         VAR final : INTEGER     );

PROCEDURE Compute_PreyLosses
                         (var sd        : Sysdef;
                          var dck       : Duck;
                          var res       : SimRes;
                          var bird      : BirdRes;
                          var physics   : Physic_Char;
                          var detailfile: text;
                              sitenr    : INTEGER   );
```

```
IMPLEMENTATION

{************************************************************************}
{* Computes the basic-energy-demand for a bird                          *}
{************************************************************************}
PROCEDURE Compute_BasicEnergy
                         (var sd        : Sysdef;
                          var dck       : Duck;
                          var res       : SimRes;
                          var bird      : BirdRes;
                          var physics   : Physic_Char;
                          var detailfile: text;
                        {   var breath,air: REAL;   }
                              detail    : INTEGER   );
VAR i,j       : INTEGER;
   AirCool,
   BreathBMR,SaltEnerg,
   FlyingBas, SwimBas,
   dailyheatbas, yy,
   DailyBasicExpenditure   : REAL;

BEGIN
  {************************************************************************
  * First the basal metabolic rate is handled . Note that BMR (as       *
  (* characteristics) is given per kg body mass (that's in g)            *
  ************************************************************************}
  res.heat.bmr := dck.BMR * bird.weight/1.0e3;          {* W          *}
  res.energy.bmr := res.heat.bmr;                       {* W          *}
  {************************************************************************}
  {* First the amount of work needed normally                           *}
  {* Note that some terms are comuted in W, and some in J per action    *}
  {* just a loss of heat.                                               *}
  {************************************************************************}
  {* first term: cooling in the air ; position == 0                     *}
  AirCool   := HeatingEider_Energy
               (physics,dck,bird,sd,0) ;                {* W          *}
  res.heat.condAir := AirCool;                          {* W          *}
  {************************************************************************
  # Computation of breathing energy losses . Energy in J/sec            #
  # It's 20 kJ / gram DW - > 20 kJ / gram O2 . Always needed is BMR-    #
  # related breathing                                                   #
  # Note: it is only used to account for the additional losses after    #
  # foraging                                                            #
  ************************************************************************}
  BreathBMR := Breathing_Energy
               ( dck.BMR*bird.weight/1.0e3,
                 physics,dck,sd );                      {* W          *}
  res.heat.heatBreath := BreathBMR;                     {* W          *}
  {************************************************************************
  * Note that birdweight was in grams                                    *
  ************************************************************************}
  {************************************************************************}
  {* Basic flytime is assumed to be 0.25 hours = .25*3600 sec           *}
  {* Flying costs energy, of which part is vonverted into body heat     *}
  {* inefficient production of fly-movement                             *}
  {* Similar for swimming . Joules per day                              *}
  {************************************************************************}
  FlyingBas := Flying_Energy (dck.flytimperday, dck);   {* Joules     *}
  res.energy.flying := FlyingBas;                       {* Joules     *}
  res.heat.flying   := (1.0-dck.energ_eff)*FlyingBas;   {* Joules     *}

  SwimBas   := Swimming_Energy (dck) *dck.swimtimPerDay;{* Joules     *}
  res.energy.swimming:= SwimBas;                        {* Joules     *}
  res.heat.swimming  := (1.0-dck.energ_eff)*SwimBas;    {* Joules     *}
```

```
  {*********************************************************************}
  {* Note that it is assumed that a duck swims about 500 m/day        *}
  {*********************************************************************}
  res.energy.basic      := {* these are basic energy terms             *}
                           - res.energy.bmr * SecPerDay   {* Joules     *}
                           - FlyingBas+SwimBas;           {* Joules     *}

  res.heat.basic        :=( res.energy.bmr               {* W          *}
                            - res.heat.condAir           {* W          *}
                            - res.heat.heatBreath
                            ) * SecPerDay                {* Joules     *}
                            + res.heat.flying            {* Joules     *}
                            + res.heat.swimming;         {* Joules     *}
  {*********************************************************************
  * energy.basic is the amount of energy needed anyway per day (<0)   **
  * Part of that energy is converted into heat, and the net heat gain **
  * is heat.basic. If this is negative, more heat is needed. May be   **
  * there is sme compensation from heat produced (possibly) when diving*
  * If it is positive, it may be used to compensate losses resulting  **
  * (possibly) from diving. If both are losss, feeding has to         **
  * compensate both. if both are gains, the animal has to loose the   **
  * surplus heat.                                                     **
  *********************************************************************}

END;

{*********************************************************************
# Computation profitability of one prey                              #
# The profitability is defined as the amount of energy a duck        #
# can get out of one prey, divided by the time needed to catch that  #
# prey. Thus, it's the nett energy rate in J/sec/per prey.           #
*********************************************************************}
FUNCTION Profitability (var sd    : sysdef;
                        var detFil: text;
                        var dck   : Duck;
                        var br    : BirdRes;
                            physch: Physic_Char;
                        var sim   : SimRes;
                            preynr,
                            sitenr,
                            detail: INTEGER;
                            length: REAL;
                            len   : INTEGER
                                        ) : REAL;

VAR Ndives_need,
    weightIncRate,
    heat_inc,
    basic_heat,
    basic_energy_tot,
    totheat,
    monthfact,
    heatbasic,
    enerbasic,
    yy,xx : REAL;
BEGIN
  {*********************************************************************}
  {* The profitability of one prey depends on the energ content of    *}
  {* the animal and the energy costs for a bird to catch an andle the *}
  {* prey                                                             *}
  {*********************************************************************}
  monthfact    := sd.shell.a_month[preynr,sim.date.month];
  flesh[len]   := Meat_Content   (sd,sitenr,preynr,length,monthfact);
                                                    {* gram flesh    *}
  ener_inp[len]:= PreyInd_Energy (flesh[len],sd,dck);{* Joules/prey   *}
```

```
{*********************************************************************}
{* Here we find a coupling between heat and energy. If the bird     *}
{* has a constant mass, the ener_inp is also directly converted into*}
{* heat. If there is a certain increase in weight expected          *}
{* (based on 'desired masses'), then not all ener_inp is used       *}
{* for heat production, but also for storage. If, on the other hand,*}
{* the masses can decrease a bit, it adds to the heat production    *}
{* And, if there not enough food the provide the necessary energy   *}
{* the body mass MUST decrease. This is accounted for at the end of *}
{* this routine.                                                    *}
{*********************************************************************}
heat_inp[len]:= ener_inp[len];                    {* if all ->heat *}

shellw[len]  := Shell_Weight   (sd,sitenr,preynr,length);
                                                  {* gram shell    *}

waterw[len]  := Water_Content  (sd,sitenr,preynr,length);
                                                  {* gram water    *}
saltw[len]   := Salt_Content   (sd,waterw[len]);  {* gram salt     *}
saltcost[len]:= Salt_Energy    (saltw[len],sd);   {* J for the prey*}


yy           := Crushing_Energy(sd,sitenr,preynr,length);
sim.energy.crushing:= yy;                         {* energy Joules *}
sim.heat.crushing  := yy;                         {* energy Joules *}
crush[len]   := yy;
freshw[len]  := Fresh_Weight   (sd,sitenr,preynr,length);

heatprey[len]:= HeatingPreyInd_Energy
                         (freshw[len],dck,sd,
                          sd.shell);              {* heating Joules*}

Diving_Energy (dck,br,sim.energy,sim.heat,        {* energy for    *}
               sim.time,sd,physch,sitenr);        {* diving Joules *}

accel0en [len]:= sim.energy.divingAccel0;         {* joules        *}
accel0het[len]:= sim.heat.divingAccel0;           {* joules        *}
acceleng [len]:= sim.energy.divingaccel;          {* joules        *}
accelheat[len]:= sim.heat.divingaccel;            {* joules        *}
drageng  [len]:= sim.energy.divingDrag;           {* Joules        *}
dragheat [len]:= sim.heat.divingDrag;             {* Joules        *}
dbuoyeng [len]:= sim.energy.divingBuoy;           {* Joules        *}
dbuoyheat[len]:= sim.heat.divingBuoy;             {* Joules        *}

Bottom_Energy (sd,dck,br,sim.heat,sim.energy,     {* energy for    *}
               sim.time,physch,sitenr);           {* searching J   *}
botteng  [len]:= sim.energy.divingBott;           {* Joules        *}
bottheat [len]:= sim.heat.divingBott;             {* Joules        *}

digestion[len]:= Digestion_energy (flesh[len],
                 sd,dck);                         {* digest costs J*}

divingtime[len]   := sim.time.descent
                   + sim.time.ascent;             {* dive time     *}
bottomtime[len]   := sim.time.bottom;             {* time searching*}
handlingtime[len] := Handling_Time(sd.Area[sitenr]^.musselKind,
                           dck,sd,sitenr);        {* time handling *}
Resting_Time (dck , sd ,sim.time,sitenr);         {* time needed to*}
                                                  {* recover       *}
restingtime[len]  := sim.time.resting;            {* secs          *}
digestiontime[len]:= Digestion_Time(flesh[len],dck);{* 1/digest rate*}


tot_time[len]     := bottomtime[len] + handlingtime[len]
                  + divingtime[len]
```

```
                 + digestiontime[len]
                 + restingtime[len];              {* sec           *}
underw_time[len]  := bottomtime[len] + divingtime[len] ;  {* sec    *}
{*********************************************************************}
{* Extra cooling of animal as a result of diving                     *}
{*********************************************************************}
{*********************************************************************}
{* First the amount of work needed normally                          *}
{* Note that some terms are comuted in W, and some in J per action   *}
{*********************************************************************}
{* first term: cooling in the air ; position == 0                    *}
{*********************************************************************}
AirCool[len]   := HeatingEider_Energy
             (physch,dck,br,sd,0);                {* Watts      *}
{* second: cooling in the water ; position == 1                      *}
H2OCool[len]   := HeatingEider_Energy
             (physch,dck,br,sd,1);                {* Watts      *}
ExtraCool[len] := (H2OCool[len] - AirCool[len])
                * underw_time[len];               {* Joules     *}
{*********************************************************************
# Computation of breathing energy losses . Energy in Joules here,    #
# because it's about the amount of flesh that's assimilated          #
#=====================================================================
# Note :: first it was assumed that Breathfod - BreathBMR was the    #
# extra breathing, but that's not true: BreathFod alone is the extra #
# cooling.                                                            #
**********************************************************************}
BreathBMR[len] := Breathing_Energy
             ( dck.BMR*br.weight/1.0e3,           {* bmr =W/kg  *}
               physch,dck,sd ) * tot_time[len];   {* Joules     *}
BreathFod[len] := Breathing_Energy
             ( ener_inp[len], physch,dck,sd );    {* Joules     *}
ExtraBreat[len]:= BreathFod[len] ;                {* Joules     *}
{*********************************************************************
* Note that birdweight was in grams                                   *
**********************************************************************}

{*********************************************************************}
{* Now the profit of a prey of length len is known.                  *}
{*********************************************************************}
heat[len]   :=  crush        [len]          {* crushing prey        *}
              - heatprey     [len]          {* heating prey         *}
              + accel0het    [len]          {* starting acceleration*}
              + accelheat    [len]          {* acceleration diving  *}
              + dragheat     [len]          {* drag                 *}
              + dbuoyheat    [len]          {* buoy during diving   *}
              + bottheat     [len]          {* buoy during searching*}
              - extracool    [len]          {* extra cooling diving *}
              - extrabreat   [len];         {* extra breathing      *}

energy[len] :=- saltcost     [len]          {* salt excretion       *}
              - crush        [len]          {* needed for crushing  *}
              - accel0en     [len]          {* initial acceleration *}
              - acceleng     [len]          {* acceleration diving  *}
              - drageng      [len]          {* drag-energy          *}
              - dbuoyeng     [len]          {* buoyancy energy      *}
              - botteng      [len]          {* energy at bottom     *}
              - digestion    [len];         {* needed for digestion *}

yy          :=  heat_inp     [len];         {* if all prey-> heat   *}

{*********************************************************************
* Energy[len] is <0, meaning that it costs energy. Heat[len] has to **
* be added to the basic heat costs. One of both may (partly)        **
* compensate the other.                                             **
```

```
* heat_inp[len] is always positive.                                    **
*======================================================================**
* Before calling this routine , and routine CheckProfitability,       **
* the basal energy demand BMR is already computed. This part          **
* consists of an energy part and a heat part as well.                 **
* So, it is possible now to compare these terms.                      **
************************************************************************}
heatbasic    := sim.heat.basic;             {* Joules/day <0 = need *}
enerbasic    := sim.energy.basic;           {* Joules/day <0 = need *}
IF heat[len]<0 THEN                         {* <0: costs, >0 gain   *}
  xx         := heat[len]
ELSE
  xx         := 0.0;

energain[len]:= (energy[len]+ xx + yy);     {*energy gain per dive  *}
en_rate[len] := energain[len]/tot_time[len];{*energy gain per sec   *}
{*********************************************************************}
{* Note that en_rate is only used to computed profitabilities, and  *}
{* prey choices. The final energy buf=dget is computed based on     *}
{* heatbasic, enerbasic and heat_inp                                *}
{*********************************************************************}

{*********************************************************************}
{* Store all this; it gives detailed info on the energy budget terms*}
{* Results for each length are printed.                             *}
{*********************************************************************}
IF detail =1
THEN
  StoreEnergyDetails(DetFil,sim,
               length,                  {* length of shell mm  *}
               flesh    [len],          {* AFDW content shell g*}
               ener_inp [len],          {* Energy content J    *}
               shellw   [len],          {* shell weight gram   *}
               waterw   [len],          {* water content g     *}
               saltw    [len],          {* salt content g      *}
               saltcost [len],          {* salt costs J        *}
               freshw   [len],          {* fresh weight gram   *}
               crush    [len],          {* cruhing shells J    *}
               heatprey [len],          {* heating prey J      *}
               accel0en [len],          {* Joules              *}
               accel0het[len],          {* joules              *}
               acceleng [len],          {* joules              *}
               accelheat[len],          {* joules              *}
               drageng  [len],          {* Joules              *}
               dragheat [len],          {* Joules              *}
               dbuoyeng [len],          {* Joules              *}
               dbuoyheat[len],          {* Joules              *}
               botteng  [len],          {* Joules              *}
               bottheat [len],          {* Joules              *}
               digestion[len],          {* Joules              *}
               extracool[len],          {* Joules              *}
               extrabreat[len],         {* Joules              *}
               bottomtime [len],        {* time searching (sec)*}
               restingtime[len],        {* time resting (sec)  *}
               divingtime[len],         {* time diving (sec)   *}
               handlingtime[len],       {* time handling (sec) *}
               digestiontime[len],      {* time digesting (sec)*}
               tot_time[len],           {* total time for prey *}
               underw_time[len],        {* time under water    *}
               energy[len],             {* gain JOULES         *}
               heat[len],               {* heat Joules         *}
               en_rate[len],            {* Watts during search *}
               heat_inc,                {* result of storage   *}
               basic_heat,              {* basal heat          *}
               basic_energy_tot,        {* basal energy        *}
```

```
                ndives_need);

  Profitability := en_rate[len];

END;

{************************************************************************
# Computation profitability of one prey                                 #
# It's nothing else then the energy gain per sec                        #
************************************************************************}
PROCEDURE Check_Profitability
                       (var sd    : sysdef;
                        var detfil: text;
                        var dck   : Duck;
                        var br    : BirdRes;
                            physch: Physic_Char;
                        var sim   : SimRes ;
                            detail: INTEGER     );
VAR sitenr,preynr,lengthnr : INTEGER;
    dl,length,yy           : REAL;
BEGIN
  dl :=  (sd.shell.length[2] - sd.shell.length[1])/2.0;
  FOR sitenr:=1 TO sd.nareas DO
  BEGIN
    preynr:= sd.area[sitenr]^.preyKind;

    FOR lengthnr := 1 TO sd.npreylengths DO
    BEGIN
      length := sd.shell.length[lengthnr]+dl;
      yy     := Profitability(sd,  detfil,dck, br ,
                              physch, sim, preynr,
                              sitenr,detail,length,lengthnr);
       IF yy<1e-10 THEN
        yy   := 0.0;
      sim.Area[sitenr].PreyRes[preynr].FoodProfit[lengthnr]:= yy;

    END;
    WriteLn(DetFil);
  END;
END;    {* of the check what the profitability is of each prey        *}


{************************************************************************
# Computation food value of an area                                     #
# plus computation of the choices of the duck                           #
************************************************************************}
PROCEDURE Area_FoodValue (var sd        : Sysdef;
                          var dck       : Duck;
                          var res       : SimRes;
                          var bird      : BirdRes;
                          var physics   : Physic_Char;
                          var detailfile: text;
                              detail    : INTEGER   );
VAR sitenr,
    preynr,
    len     : INTEGER;
    yy      : REAL;
BEGIN
  {********************************************************************}
  {* First the amount of work needed normally                         *}
  {* Note that some terms are comuted in W, and some in J per action  *}
  {********************************************************************}

  Compute_BasicEnergy( sd,dck,res,bird, physics,
                               detailfile,{breath0,air0,}
```

```
                                detail );             {* Joules       *}

{*********************************************************************}
{* NOw the profitability is needed, telling us what the profit per  *}
{* dive is                                                           *}
{*********************************************************************}
Check_Profitability(sd, detailfile, dck, bird,
                    physics,res,detail);
FOR sitenr:=1 TO sd.nareas DO
BEGIN
  res.TotFood[sitenr]          := 0.0;
  res.TotPrefFood[sitenr]      := 0.0;
  preynr:= sd.area[sitenr]^.preyKind;

  FOR len := 1 TO sd.npreylengths DO
  BEGIN
     {*****************************************************************
     * Food value of each food site. Preference of an eider duck     *
     * is not important.                                             *
     *****************************************************************}
     res.TotFood[sitenr]:=               {* Joules / m2 /sec       *}
          res.TotFood[sitenr]
         +res.area[sitenr].preyres[preynr].FoodProfit[len]
         *res.area[sitenr].preyres[preynr].FoodDens[len];
                                         {* Joules/s * aantal /m2  *}
     {*****************************************************************
     * Food value of each food site, including preference of an      *
     * eider duck                                                    *
     *****************************************************************}
     res.TotPrefFood[sitenr]:=           {* Joules / m2            *}
          res.TotPrefFood[sitenr]
         +res.area[sitenr].preyres[preynr].FoodProfit[len]
         *res.area[sitenr].preyres[preynr].FoodDens[len]
         *dck.LenPref[preynr,len];       {* Joules * aantal /m2    *}
  END;
END;
{*********************************************************************
* Now the food value of each food site is known. It is the product   *
* of the prey densituy and the food value of each prey.              *
*--------------------------------------------------------------------*
* Now, the preference of an eider duck for a prey has to be computed,*
* giving the prey choice of the duck when foraging. This is needed   *
* for our bookkeeping: we ned to know which prey is caught.          *
*********************************************************************}
FOR sitenr:=1 TO sd.nareas DO
BEGIN
  preynr:= sd.area[sitenr]^.preyKind;
  FOR len := 1 TO sd.npreylengths DO
  BEGIN
      {*****************************************************************
      * Choice without particular bird preference                     *
      *****************************************************************}
      IF res.TotFood[sitenr]>1.0e-10
      THEN
        res.area[sitenr].preyres[preynr].FoodChoice[len]:=
         res.area[sitenr].preyres[preynr].FoodProfit[len]
         *res.area[sitenr].preyres[preynr].FoodDens[len]
         /res.TotFood[sitenr]
      ELSE
         res.area[sitenr].preyres[preynr].FoodChoice[len]:= 0.0;
      {*****************************************************************
      * Choice with particular bird preference                        *
      *****************************************************************}
      IF res.TotPrefFood[sitenr]>1.0e-10
      THEN
```

```
          res.area[sitenr].preyres[preynr].FoodPrefchoice[len]:=
            res.area[sitenr].preyres[preynr].FoodProfit[len]
            *res.area[sitenr].preyres[preynr].FoodDens[len]
            *Dck.LenPref[preynr,len]
            /res.TotPrefFood[sitenr]
        ELSE
          res.area[sitenr].preyres[preynr].FoodPrefChoice[len]:= 0.0;
    END;
  END;

END; {* of Area_FoodValue                                              *}


{*********************************************************************
# Computation food value of an area (food in J/m2/s)                  #
*********************************************************************}
PROCEDURE FindBestArea (var nrf,nrfpref: INTEGER;
                            sd         : sysdef;
                        var res        : Simres      );
VAR ff, ffpref : REAL;
    i          : INTEGER;
BEGIN
  ff     := 0.0;
  ffpref := 0.0;
  FOR i:=1 TO sd.nareas DO
  BEGIN
    IF res.totfood[i]>ff
    THEN
    BEGIN
      ff  :=res.totfood[i];
      nrf := i;
    END;

    IF res.TotPrefFood[i]>ffpref
    THEN
    BEGIN
      ffpref  :=res.TotPrefFood[i];
      nrfpref  := i;
    END;
  END; {* all sites are checked. Best site is known now.              *}

END; { FindBestArea }


{*********************************************************************
* In order to find the number of dives necessary, the basis energy    **
* demand and the demand related to dives has to be known              **
*********************************************************************}
PROCEDURE Compute_NumberOfDives
                       (var sd    : sysdef;
                        var detfil: text;
                        var dck   : Duck;
                        var br    : BirdRes;
                            physch: Physic_Char;
                        var sim   : SimRes ;
                            detail,
                            sitenr: INTEGER;
                        VAR final : INTEGER    );
VAR i,j,len,preynr  : INTEGER;
    yy,
    DailyBasicExpenditure,
    breath0,air0   : REAL;
    totflesh,
    totener_inp,
    totshellw,
```

```
    totwaterw,
    totsaltw,
    totsaltc,
    totfreshw,
    totcrush,
    totheat_inp,
    totheatprey,
    totaccel0en,
    totaccel0het,
    totacceleng,
    totaccelheat,
    totdragen,
    totdragheat,
    totdbuoyen,
    totdbuoyheat,
    totbottomeng,
    totbottomhet,

    totdiving,
    totbottom,
    totdigestion,
    totAirCool,
    totH2OCool,
    totExtraCool,
    totBreathBMR,
    totBreathFod,
    totExtraBreat,
    totheat,
    totdivingeng,
    totdivingheat,
    totbottomtime,
    totrestingtime,
    totdivingtime,
    tothandlingtime,
    totdigestiontime,
    tottot_time,
    totunderw_time,
    totenergy,
    totenergain,
    toten_rate,
    weightIncRate,
    totheatcosts,
    totengcosts,
    basiceng,
    basicheat,
    daycosts,
    heatsurpl,
    heat_inc,
    totgain,
    netgain,
    DEE,
    choic ,
    ndives,totcosts,
    timeneed,
    in_tot,
    out_tot       : REAL;

BEGIN
  {*********************************************************************}
  {* The amount of work needed normally has already been computed     *}
  { in AreaFoodvalue                                                   *}
  {*********************************************************************}

  {*********************************************************************}
  {* Then: compute the work coupled with one dive (positive and       *}
```

```
  {* negative). Choice tells us which preys actually are taken        *}
  {*****************************************************************}
  totflesh             := 0.0;
  totshellw            := 0.0;
  totfreshw            := 0.0;
  totsaltw             := 0.0;
  totwaterw            := 0.0;
  totcrush             := 0.0;
  totener_inp          := 0.0;
  totheat_inp          := 0.0;
  totsaltc             := 0.0;
  totheatprey          := 0.0;
  totaccel0en          := 0.0;
  totaccel0het         := 0.0;
  totacceleng          := 0.0;
  totaccelheat         := 0.0;
  totdragen            := 0.0;
  totdragheat          := 0.0;
  totdbuoyen           := 0.0;
  totdbuoyheat         := 0.0;
  totbottomeng         := 0.0;
  totbottomhet         := 0.0;

  totdiving            := 0.0;
  totbottom            := 0.0;
  totdigestion         := 0.0;
  totAirCool           := 0.0;
  totH2OCool           := 0.0;
  totExtraCool         := 0.0;
  totBreathBMR         := 0.0;
  totBreathFod         := 0.0;
  totExtraBreat        := 0.0;
  totheat              := 0.0;
  totdivingeng         := 0.0;
  totdivingheat        := 0.0;
  totenergain          := 0.0;
  totenergy            := 0.0;
  toten_rate           := 0.0;
  totheatcosts         := 0.0;
  totengcosts          := 0.0;

  totbottomtime        := 0.0;
  totrestingtime       := 0.0;
  totdivingtime        := 0.0;
  tothandlingtime      := 0.0;
  totdigestiontime     := 0.0;
  tottot_time          := 0.0;
  totunderw_time       := 0.0;

  preynr          := sd.area[sitenr]^.preyKind;

  FOR len:= 1 TO sd.npreylengths DO
  BEGIN
   choic          :=
sim.area[sitenr].preyres[preynr].FoodPrefchoice[len];
   totflesh       := totflesh        + flesh             [len]*choic;
   totshellw      := totshellw       + shellw            [len]*choic;
   totwaterw      := totwaterw       + waterw            [len]*choic;
   totsaltw       := totsaltw        + saltw             [len]*choic;
   totfreshw      := totfreshw       + freshw            [len]*choic;
   totener_inp    := totener_inp     + ener_inp          [len]*choic;
   totheat_inp    := totheat_inp     + heat_inp          [len]*choic;
   totsaltc       := totsaltc        + saltcost          [len]*choic;
   totcrush       := totcrush        + crush             [len]*choic;
   totheatprey    := totheatprey     + heatprey          [len]*choic;
```

```
 totaccel0en    := totaccel0en     + accel0en          [len]*choic;
 totaccel0het   := totaccel0het    + accel0het         [len]*choic;
 totacceleng    := totacceleng     + acceleng          [len]*choic;
 totaccelheat   := totaccelheat    + accelheat         [len]*choic;
 totdragen      := totdragen       + drageng           [len]*choic;
 totdragheat    := totdragheat     + dragheat          [len]*choic;
 totdbuoyen     := totdbuoyen      + dbuoyeng          [len]*choic;
 totdbuoyheat   := totdbuoyheat    + dbuoyheat         [len]*choic;
 totbottomeng   := totbottomeng    + botteng           [len]*choic;
 totbottomhet   := totbottomhet    + bottheat          [len]*choic;
 totdigestion   := totdigestion    + digestion         [len]*choic;

 totAirCool     := totAirCool      + AirCool           [len]*choic;
 totH2OCool     := totH2OCool      + H2OCool           [len]*choic;
 totExtraCool   := totExtraCool    + ExtraCool         [len]*choic;
 totBreathBMR   := totBreathBMR    + BreathBMR         [len]*choic;
 totBreathFod   := totBreathFod    + BreathFod         [len]*choic;
 totExtraBreat  := totExtraBreat   + ExtraBreat        [len]*choic;

 totenergy      := totenergy       + energy            [len]*choic;
 totenergain    := totenergain     + energain          [len]*choic;
 totheat        := totheat         + heat              [len]*choic;

 toten_rate     := toten_rate      + en_rate           [len]*choic;

 totdivingtime  := totdivingtime   + divingtime        [len]*choic;
 totbottomtime  := totbottomtime   + bottomtime        [len]*choic;
 totrestingtime := totrestingtime  + restingtime       [len]*choic;
 tothandlingtime:= tothandlingtime + handlingtime      [len]*choic;
 totdigestiontime  := totdigestiontime + digestiontime [len]*choic;
 tottot_time    := tottot_time     + tot_time          [len]*choic;
 totunderw_time := totunderw_time  + underw_time       [len]*choic;

END;

totdivingheat   := totdivingheat   + totaccel0het
                                   + totaccelheat
                                   + totdragheat
                                   + totdbuoyheat
                                   + totbottomhet
                                   - totextracool
                                   - totextrabreat;

totdivingeng    := totdivingeng    - totaccel0en
                                   - totacceleng
                                   - totdragen
                                   - totdbuoyen
                                   - totbottomeng;

totheatcosts    := totheatcosts    + totcrush
                                   - totheatprey
                                   + totdivingheat;
totengcosts     := totengcosts     - totsaltc
                                   - totcrush
                                   - totdigestion
                                   + totdivingeng;
totener_inp     := totener_inp;    {* no addition                  *}
{*****************************************************************}
{* There is also an extra energy need or an extra heat production  *}
{* because the mass of the duck has to increase or can decrease    *}
{*****************************************************************}
weightIncRate := br.weightIncreaseRate;            {* gram AFDW/day*}
{** Note that heat)inc is per day!!, and inp per dive CHANGE!!     *}
heat_inc      := - weightIncRate                   {* gram * 22.5e3*}
                 * dck.EnergContentAnimal;         {* = heat extra *}
```

```
                                                    {* <0 costs     *}
totgain       := totener_inp + heat_inc;            {* food+reserves*}
{*********************************************************************
* energy.basic : basal energy demand <0 : need                       *
* heat.basic   : basal heat demand   <0 : need, >0 gain              *
* tot.engcosts : total energy costs per dive <0 : need               *
* totheatcosts : total heat costs per dive   <0 : need, >0 gain      *
* totgain      : tot results from food       <0 need, >0 gain        *
* heat_inc     : energy result from weightchanges                    *
* If totgain is negative: the bird is in trouble; it cannot feed     *
* anyhow. If totgain >0, then the number of dives can be computed.   *
* If the number of dives er day needs more then the time per day,    *
* then there is not enough time to forage. In fact, we have chosen   *
* 80% of the day time as foraging lmit.                              *
* We have to check for a positve heat from diving as compensation    *
* for a heat demand while wimming, or vice versa. If both are        *
* needs, they simply add to the energy need. If both are positive    *
* they simply are Joules, the bird has to get rid off.               *
* In no way can heat profits compensate kinetic energy demands!!     *
*********************************************************************}
basiceng    := sim.energy.basic;                  {* <0 need       *}
basicheat   := sim.heat.basic;                    {* <0 need >0 gain}

IF (basicheat>=0) AND (totheatcosts>=0)           {* no heat needs *}
THEN BEGIN
  netgain   := totengcosts +totener_inp;          {* ener from food*}
  daycosts  := basiceng + heat_inc;               {* daily costs   *}
  NDives := - daycosts /netgain;
  heatsurpl := basicheat + NDives*totheatcosts;   {* daily heat    *}
                                                  {* surplus       *}
  {*****************************************************************}
  {* The bird has to loose this surplus heat                        *}
  {*****************************************************************}
  DEE       := totengcosts*NDives + basiceng;     {* daily expend  *}
END;
IF (basicheat<0) AND (totheatcosts<0)             {* heat need in  *}
THEN BEGIN                                        {* both cases    *}
  netgain   := totengcosts +totener_inp
               + totheatcosts;
  daycosts  := basiceng + heat_inc
               + basicheat;
  NDives := - daycosts /netgain;
  heatsurpl := basicheat + NDives*totheatcosts;   {* daily heat    *}
                                                  {* surplus <0,now*}
  {*****************************************************************}
  {* But, this heat need is accounted for                           *}
  {*****************************************************************}
  DEE       := (totengcosts+totheatcosts)*NDives
               + basiceng + basicheat;            {* daily expend  *}
END;

IF (basicheat>0) AND (totheatcosts<0)             {* heat need in  *}
THEN BEGIN                                        {* diving alone  *}
  netgain   := totengcosts +totener_inp
               + totheatcosts;
  daycosts  := basiceng + heat_inc;
  {*****************************************************************}
  {* If the diving costs heat, and the swimming has a positive heat*}
  {* gain, then actually basicheat/Ndive can be added to the        *}
  {* divingheatcosts: netgain1 := netgain + basicheat/Ndives        *}
  {* Then, NDives follows from                                      *}
  {* Ndives:= -daycosts/(netgain1) =                                *}
  {*          -daycosts/(netgain + basicheat/Ndives)                *}
  {* Left and right times NDives and rearranging gives              *}
  {* ND*ND*netgain + ND * basicheat = -daycosts * ND   ==>          *}
```

```
  {* ND =0 or ND = -(basicheat + daycosts) / netgain              *}
  {*-------------------------------------------------------------*}
  {* There is a restriction: if basicheat/ND is > -totheatcosts   *}
  {* then we should only put -totheatcosts/ND into the above      *}
  {* computation                                                  *}
  {*************************************************************}
  NDives    := - (basicheat + daycosts) /netgain;
  heatsurpl := basicheat + NDives*totheatcosts;   {* daily heat    *}
                                                  {* surplus       *}
  DEE       := (totengcosts+totheatcosts
                + basicheat/NDives) *Ndives
                + basiceng;
  IF heatsurpl>0 THEN                             {* restriction   *}
  BEGIN                                           {* gest active   *}
    NDives  := -(daycosts) /(netgain-totheatcosts);
    heatsurpl:= basicheat + NDives*totheatcosts;  {* daily heat    *}
                                                  {* surplus       *}
    DEE     := (totengcosts+totheatcosts) *Ndives {* is the same eq*}
               + basiceng +basicheat;             {* as one above  *}
  END;
END;

IF (basicheat<0) AND (totheatcosts>0)             {* heat need in  *}
THEN BEGIN                                        {* diving alone  *}
  netgain   := totengcosts +totener_inp;
  daycosts  := basiceng + heat_inc;
  {*************************************************************}
  {* If the diving gains heat, and the swimming has a negative heat*}
  {* gain, then actually totheatcosts*Ndive can be added to the   *}
  {* basicheat : daycosts1 := basiceng + basicheat+heat_inc+ND*thc *}
  {* Then, NDives follows from                                    *}
  {* Ndives:= -daycosts1/(netgain) =                              *}
  {*          -(daycosts+ND*thc)/netgain                          *}
  {* Rearranging gives                                            *}
  {* ND*(netgain+thc)= -daycosts    ==>                           *}
  {* ND = -daycosts) / (netgain +thc)                             *}
  {*-------------------------------------------------------------*}
  {* There is a restriction: if thc*ND is > -basicheat            *}
  {* then we should only put -basicheat/ND into the above         *}
  {* computation                                                  *}
  {*************************************************************}
  NDives := - (daycosts) /(netgain+totheatcosts);
  heatsurpl := basicheat + NDives*totheatcosts;   {* daily heat    *}
                                                  {* surplus       *}
  DEE    :=   totengcosts  * NDives
            + basiceng + basicheat
            + totheatcosts * NDives;
  IF heatsurpl>0 THEN                             {* restriction   *}
  BEGIN                                           {* gest active   *}
    NDives:= -(-basicheat+daycosts) /netgain;
    heatsurpl:= basicheat + NDives*totheatcosts;  {* daily heat    *}
                                                  {* surplus       *}
    DEE    := totengcosts  * NDives
            + basiceng ;
  END;
END;

DailyBasicExpenditure := DEE;

in_tot      := Ndives * totener_inp ;
out_tot     := DailyBasicExpenditure;
timeneed    := Ndives * tottot_time;
IF (timeneed >0.8*secperDay)
OR (timeneed<0)
THEN
```

```
  BEGIN
    Write ('the duck cannot feed itself sufficiently');
    final :=1;
  END;

  sim.area[sitenr].NDives := NDives;
  IF detail =1
  THEN

    StoreDetails( DetFil,sim, sd,
                  totflesh,  totshellw,    totwaterw,   totsaltw,
                  totfreshw, totaccel0het, totaccelheat,totdragheat,
                  totdbuoyheat, totbottomhet,
                  totextracool, totextrabreat,
                  totdivingheat,
                  totaccel0en,  totacceleng, totdbuoyen,  totdragen,
                  totbottomeng, totdivingeng,
                  totcrush,  totheatprey,  totsaltc,  totdigestion,
                  totener_Inp,   heat_inc,  totgain,
                  netgain,   DailyBasicExpenditure,  heatsurpl,
                    NDives,    in_tot,
                  out_tot,   timeneed,
                  totdivingtime,totbottomtime,
                    totrestingtime,   tothandlingtime,
                  totdigestiontime,  tottot_time,  totunderw_time);
END;

{*********************************************************************}
{* Computes the prey losses. The length-choices are known now. The     *
*  numbers of birds are known as well.                                *}
{*********************************************************************}
PROCEDURE Compute_PreyLosses
                         (var sd        : Sysdef;
                          var dck       : Duck;
                          var res       : SimRes;
                          var bird      : BirdRes;
                          var physics   : Physic_Char;
                          var detailfile: text;
                              sitenr    : INTEGER   );
VAR i,j,preynr,
    len        : INTEGER;
    nrbirds,
    ndives,
    areasize   : REAL;
BEGIN
  preynr  := sd.area[sitenr]^.preyKind;     {* mussels or cockles?    *}
  nrbirds := res.totpopulat;
  ndives  := res.area[sitenr].NDives;
  areasize:= sd.area[sitenr]^.size;
  FOR len:= 1 TO sd.npreylengths DO
  BEGIN
    res.area[sitenr].preyres[preynr].foodloss[len]:=
      res.area[sitenr].preyres[preynr].FoodPrefchoice[len]
      * Ndives                              {* number dives/day/bird  *}
      * nrbirds                             {* number of birds        *}
      / areasize ;                          {* areasize (loss is on   *}
                                            {* densities              *}
      {*****************************************************************}
      {* Losses are in numbers per m2 per day, for each length-class  *}
      {*****************************************************************}
  END;
END
```