Statistics Norway
Research Department

*Dag Kolsrud*



# Documentation of Computer Programs that Extend the SEEM Model and Provide a Link to the RAINS Model

# Contents

# 1 Introduction

This note documents three programs that (1) extends the energy model SEEM[1], (2) supports a link from SEEM to the polution model RAINS[2] and (3) provides an example of how the interactive operation of RAINS can be automated. Such supporting programs may contribute substantially to the "value added" by the two disjoint models. This is particularly true when it comes to models that are used repeatedly and systematically for the same purposes.

In addition to providing detailed technical information on the system specific macros and batch files, this document hopefully fulfills and thus illustrates the requirement of proper documentation of computer programs. The SEEM project has run for several years in the Division for resource and environmental economics (SRM) at the Research Department, Statistics Norway. During this time focus has been on the economic content of the model and on quantitative aspects of the modelling. Consequently, and to obey partial time constraints, the computer implementation of the model has proceeded in an ad hoc manner, paying little or no attention to structured programming and to documentation. In the long run such an approach is suboptimal regarding quality (code error reduction), maintainance (including changes and extensions to the model/code), efficiency (run time and data storage) and productivity (using the model in analysis). Negative consequences imediately become manifest if crucial persons leave the project. Loss of human capital can be reduced if some capital is left behind in the form of «sufficient» documentation. If the SEEM(-RAINS) project is continued and developed further, it is my hope that this document may be found useful by the people taking over.

The SEEM model calculates estimated energy demand in 13 West-European countries, assuming that any level of demand will be supplied. Given activity levels in the form of exogenous GDP estimates, the SEEM model outputs disaggregated energy consumption figures. For all fuel-sector-country combinations the model outputs simulated time series of energy consumption figures. The model is implemented in the TROLL system on either a personal computer (pc) running the DOS operating system or on a workstation running the Unix operating system. The programs in this document are all implemented on a pc running the DOS operating system. Thus, their names and the names of directories and files read and written by the progams have to obey the 8-character limit of DOS.

The TROLL macro seemsum reads SEEM output files and calculates all aggregates over the three dimensions of fuels, sectors and countries, plus a few categorical aggregates in addition. Alternatively, each disaggregated consumption figure is premultiplied by a fuel specific $CO_2$-emission coefficient, thereby calculating disaggregated and aggregated emission figures rather than energy consumption figures. The TROLL macro runs under both the DOS and the Unix operating system.

The RAINS model is a commercial program running under the DOS operating system on a pc. It comes with a database of scenario specific energy consumption figures, from which it calculates emissions, atmospheric transport and downfall in Europe. As an alternative to the "built-in" data, a user may provide his own data to replace RAINS' figures. The TROLL macro sim2rain allows such an operation, by aggregating and disaggregating SEEM output figures to fit into RAINS' fuel and sector categories. The data are reformated to RAINS' input requirements, and stored on disk files as fixed format ASCII tables that can be imported by RAINS. The RAINS model is made to be run in an interactive mode by a user in front of the computer. To automatize the data transfer from SEEM to RAINS, a small batch file running a keyboard simulator has been written. The DOS batch file rimport makes the keyboard simulator navigate among RAINS menu options and chooses the right ones for the task of data import and scenario calculations. Once the SEEM scenarios have been imported by RAINS, the user may choose among the original RAINS scenarios *and* the SEEM scenarion when calculating the environmental consequences of energy consumption.

---

[1] SEEM is the abbreviation of Sectorial European Energy Model, developed by the Division for resource and environmental economics (SRM) at the Research Department, Statistics Norway, cf. Boug, P. (1995): User's Guide, The SEEM-model Version 2.0, Statistics Norway, and Brubakk, L. et. al. (1995): SEEM – An Energy Demand Model for Western Europe, Reports XX/95, Statistics Norway.

[2] RAINS is short for Regional Acidification INformation and Simulation model of Europe. It is developed by IIASA (International Institute for Applied Systems Analysis), Austria. The RAINS project is summerized in Alcamo, J, Shaw, R, and Hordijk, L, (eds): *The RAINS model of acidification, science and strategies in Europe*, Kluwer Academic Publishers, 1990.

# 2 The TROLL macro seemsum

The SEEM model outputs disaggregated energy consumption figures. For each single fuel $f$ in each single sector $s$ within each single country $c$, the SEEM model outputs a simulated time series of energy consumption figures $x_{c,s,f}(t)$. The seemsum macro (in the seem directory) does either

(i) aggregate the energy consumption figures over the three dimensions of fuel, sector and country, i.e. calculate the different aggregates

$$X_{s,c}(t)=\sum_f x_{c,s,f}(t), \qquad X_{f,c}(t)=\sum_s x_{c,s,f}(t), \qquad X_{f,s}(t)=\sum_c x_{c,s,f}(t),$$

$$X_c(t)=\sum_s\sum_f x_{c,s,f}(t), \qquad X_s(t)=\sum_c\sum_f x_{c,s,f}(t), \qquad X_f(t)=\sum_c\sum_s x_{c,s,f}(t),$$

$$X(t)=\sum_c\sum_s\sum_f x_{c,s,f}(t),$$

and, in addition, a few special category aggregates over subsets of the fuels and subsets of the sectors, or

(ii) calculate disaggregated $CO_2$ emissions: $e_{c,s,f}(t) = a_f x_{c,s,f}(t)$, where $a_f$ is a $CO_2$ emission coefficient specific to the fuel $f$, before calculating the corresponding aggregates

$$E_{s,c}(t)=\sum_f e_{c,s,f}(t), \qquad E_{f,c}(t)=\sum_s e_{c,s,f}(t), \qquad E_{f,s}(t)=\sum_c e_{c,s,f}(t),$$

$$E_c(t)=\sum_s\sum_f e_{c,s,f}(t), \qquad E_s(t)=\sum_c\sum_f e_{c,s,f}(t), \qquad E_f(t)=\sum_c\sum_s e_{c,s,f}(t),$$

$$E(t)=\sum_c\sum_s\sum_f e_{c,s,f}(t).$$

In addition come the few special category aggregates over subsets of the fuels and subsets of the sectors.

The SEEM model contains 13 countries, 7 sectors and 14 fuels, which imply that the seemsum macro either calculates more than 400 aggregated energy series or about 600 disaggregated and aggregated emission series. Consult the documentation of the SEEM model (Boug, 1995) for more details on the countries, sectors and fuels, or read the comments to the TROLL macro code on the following pages.

## 2.1 The code of the TROLL macro seemsum

The tasks (i) and (ii) are performed by the same macro by multiplying each disaggregated energy consumption figure output from SEEM by either 1 or a fuel specific $CO_2$-emission coefficient before aggregation. In the latter case (ii) the calculated disaggregated emission figures are also saved to disk along with the aggregates.

The macro performs the calculations in nested loops. It runs through all fuels within all sectors within all countries, initializing and updating aggregation variables, and storing them on disk after final update. To make the macro more readable and understandable we name each variable according to its content. But, the variables amounting to more than half a thousand in number motivates a scheme of automatic naming of the individual variables. The names are constructed by concatenating the *contents* of a few name-variables. Let us say that the variables named *fuel, sector* and *country* at a certain stage contain the names "oil", "in" (short for industry) and "no" (short for Norway). By concatenating (the content of) the three variables and an explicit text string, like *name* = *fuel* || "co2" || *sector* || *country,* we get *name* = "oilco2inno". In TROLL the construct *&name* returns the content of the variable *name*. Hence the macro command ">>*DOFILE &name* = ... " implies the TROLL command "DOFILE oilco2inno = ...", which files the right hand side of the equation as the disaggregated $CO_2$ emission from oil consumption in Norwegian industries in a disk variable named "oilco2inno". The alternative would be an indexed variable (e.g. *emission*[*c,s,f*]), which would not reveal its content, thus yielding an incomprehensible code. This would hamper or effectively block code maintenance (from a changing staff). Besides, it would create a problem of naming each disk variable.

The macro is extensively commented, to such an extent that we only sketch an outline of its structure:

1. User interface and initialization,
2. For c = first_country to last_country
   for s = first_sector to last_sector
     for f = first_fuel_within_sector to last_fuel_within_sector
       update relevant variables;
       if final update store variables;
3. Store rest of the variables.

```
/* Portable TROLL macro: SEEMSUM                                                                    */

/* Programmed by: Dag Kolsrud, SRM, Research Department, Statistics Norway                          */
/* Updates: Version 1 in February 1995 by Dag Kolsrud                                               */

/* Purpose: Either to (1) sum up energy consumption figuress or (2) calculate and sum up CO2-emission figures s for several */
/* European countries. A possible third task of calculating and summing up fuel expenditure may be implemented the same way*/
/* as the CO2-emission.                                                                             */

/* Short description: This macro does the following:                                                */
/* - decides whether to calculate (1) consumption or (2) emmision figures from user's choice input,  */
/* - reads disaggregated energy consumption figures for various fuel-sector-country from a FORMDATA text file, */
/* - initializes and updates aggregates with disaggregates in single and nested loops through fuels, sectors and countries, */
/* - stores the aggregates in an ouput TROLL FORMDATA text file or a FAME file after final updates. */

/* Details: The calculations are performed by traversing the 3 "dimensions" of country-, sector- and fuel specific energy */
/* consumption figures, calculating disaggregated and/or aggregated figures. The structure of the calculations are the same */
/* in both cases (1) and (2), hence the two tasks of aggregating energy consumption figures and calculating and aggregating */
/* emission figures can be gathered into one single macro. The macro prompts the user to choose between the two tasks */
/* (1) and (2), which are independent of each other. The user also has to provide a number for the scenario for which the */
/* consumption or emission aggregates are to be calculated.                                         */
/*                                                                                                  */
/* The SEEM model has 13 countries: AUstria, BElgium, BundesRepublik, Canton Helvetica, DenmarK, FRance, Great Britain,*/
/* ITaly, NetherLands, NOrway, Suomi Finland, SPain and SWeden: (AU, BE, BR, CH, DK, FR, GB, IT, NL, NO, SF, SP, SW).*/
/* There are 4 STationary sectors: ELectricity, HOseholds, INdustry and SErvices: (EL, HO, IN, SE), and 3 MObile sectors: */
/* Transport Passengers, Transport Freigth and Freight Air: (TP, TF, FA). The stationary sectors use 6 fuels: NUClear power, */
/* RENewables, COAl, OIL, Natural GaS and ELEctricity: (NEC, REN, COA, OIL, NGS, ELE), while the mobile sectors use */
/* 8 fuels: Bus Diesel, GAs, GasOline, Rail Electricity, Rail Diesel, DIesel, ELectricity and OIl: (BD, GA, GO, RE, RD, BD, DI, */
/* EL, OI). TROLL FORMDATA files output from the SEEM model contain disaggregated country-sector-fuel specific energy */
/* consumption data in the form of time series. Some series may be equal to zero. Not all fuels and sectors combine, e.g. */
/* there is no use of ELEctricity in the production of ELectricity, and some fuels are transport specific, e.g. Rail Diesel in the */
/* sector Transport Passengers.                                                                     */
/*                                                                                                  */
/* For the STationary sources this macro calculates (2) disaggregated CO2-emissions from each of   */
/*      1: #countries x 4 x 3 = #countries x 12 <= 156 country-sector-fuel combinations, contained in the TROLL variables */
/*         fuel_CO2_sector_country (see below for explanations of the composition of names),        */
/* and/or (2 or 1) it sums up disaggregated energy consumption figures or the calculated emission figures for */
/*      2: #countries x 4 <= 52 totals for the country-sector combinations (aggregating over all STationary fuels), named */
/*         ENR_{CON or CO2}_sector_country,                                                         */
/*      3: #countries x (6 or 3) <= 78 or 39 totals for the country-fuel combinations (aggregating over ST sectors), named */
/*         fuel_{CON or CO2}_ST_country,    (= fuel_{CON or CO2}_TOT_country (except for NGS, cf 31), which is omitted) */
/*      4: (5 or 3) + 3 x (4 or 3) = (17 or 12)  totals for the fuel in sector combinations (aggregating over all countries), named */
/*         fuel_{CON or CO2}_sector_EU    (or country-aggregate codes - C#countries, e.g. C3 - if a subset of countries), */
/*      5: #countries <= 13 totals for the countries (aggregating over all STationary fuels and sectors), named */
/*         ENR_{CON or CO2}_ST_country,                                                             */
/*      6: 4 totals for the sectors (aggregating over all STationary fuels and all countries), named */
/*         ENR_{CON or CO2}_sector_EU,                                                              */
/*      7: 6 or 3 totals for the fuels (aggregating over all STationary sectors and all countries), named */
/*         fuel_{CON or CO2}_ST_EU,              (= fuel_{CON or CO2}_TOT_EU (except for NGS, cf 32), which is omitted) */
/*      8: 1 total for the STationary sources (aggregating over all STationary country-sector-fuel combinations), named */
/*         ENR_{CON or CO2}_ST_EU,                                                                  */
/* where 4 or 3 refers to 4 CONsumption or 3 CO2 aggregates, since the fuel ELE emits no CO2. The numbering corresponds */
/* to the numbered TROLL code statements in the macro. This sums up to (1) #countries x 11 + 28 <= 171 consumption */
/* aggregates or (2) #countries x 12 <= 156 disaggregated and #countries x 8 + 20 <= 124 aggregated emissions in the */
/* STationary sectors.                                                                              */
/*                                                                                                  */
/* For the MObile sources this macro calculates (2) disaggregated CO2-emissions from each of       */
/*      9: #countries x (5 + 2 + 1) = #countries x 8 <= 104 country-fuel in sector combinations, named */
/*         fuel_{CON or CO2}_sector_country,                                                        */
```

```
/* and/or (2 or 1) it sums up disaggregated energy consumption figures or the calculated emission figures for */
/*      10: #countries x 2 <= 26 totals for the country-sector (TP and TF only, since FA sector uses only one single fuel) */
/*          combinations (aggregating over all MObile fuels), named ENR_{CON or CO2}_sector_country, */
/*      11: #countries x 2 <= 26 totals for the country-fuel (DI and OI only, since the other MObile sectors each uses only */
/*          one single fuel) combinations (aggregating over all MObile sectors), */
/*          named fuel_{CON or CO2}_MO_country,          (= fuel_{CON or CO2}_TOT_country, which is omitted) */
/*      12: (6 or 5) + (3 or 2) + 1= (10 or 8) totals for each fuel-sector combination (aggregating over all countries), named */
/*          fuel_{CON or CO2}_sector_EU, */
/*      13: #countries <= 13 totals for the countries (aggregating over all MObile fuels and sectors), named */
/*          ENR_{CON or CO2}_MO_country, */
/*      14: 2 totals for sectors TP and TF (FA uses only OI, aggregating over all MObile fuels and all countries), named */
/*          ENR_{CON or CO2}_sector_EU, */
/*      15: 2 totals for the fuels DI and OI, cf. 11 (aggregating over all MObile sectors and all countries), named */
/*          fuel_{CON or CO2}_MO_EU,                     (= fuel_{CON or CO2}_TOT_EU, which is omitted) */
/*      16: 1 total for the MObile sources (aggregating over all MObile country-sector-fuel combinations), named */
/*          ENR_{CON or CO2}_MO_EU. */
/* This sums up to (1) #countries x 5 + 15 <= 80 consumption aggregates or (2)  #countries x 8 <= 104 disaggregated */
/* and #countries x 5 + 13 <= 78 aggregated emissions in the MObile sectors. */
/* */
/* The STationary and the MObile sectors together are summed up by the following: */
/*      17: #countries totals for each country (the sum of STationary total and MObile total in each country), named */
/*          ENR_{CON or CO2}_TOT_country, */
/*      18: 1 total for all STationary and MObile sources (aggregating over everything), named */
/*          ENR_{CON or CO2}_TOT_EU, */
/* This sums up to #countries + 1 <= 14 consumption or emission aggregates. Since each fuel is specific to the STationary */
/* and the MObile sectors there is no fuel TOTal (which would equal the ST and MO total). */
/* */
/* In addition come some special aggregates. First, the sector aggregate called Final Consumption = TOtal - ELctricity */
/* producing STationary sector yields the extra aggregates: */
/*      19: #countries x (3 or 2) <= (39 or 26) FC totals for the country-fuel combinations (aggregating over the fuels */
/*          COA, NGS (and ELE if not CO2 calculations) in ST sectors HO, IN and SE. The FC total for OIL containing oil */
/*          fuels in MO sectors is calculated in 25), named fuel_{CON or CO2}_FC_country, */
/*      20: 3 or 2 FC totals for the fuels (aggregating over the fuels COA, NGS (and ELE if not CO2 calculations) in ST */
/*          sectors HO, IN and SE. The FC total for OIL containing oil fuels in MO sectors is calculated in 28), named */
/*          fuel_{CON or CO2}_FC_EU, */
/*      21: #countries <= 13 FC totals for the countries (aggregating over all ST fuels and FC sectors), named */
/*          ENR_{CON or CO2}_FC_country,     (= ENR_{CON or CO2}_TOT_country - ENR_{CON or CO2}_EL_country), */
/*      22: 1 total for the FC sectors (aggregating over all country-FC sector-ST fuel combinations), named */
/*          ENR_{CON or CO2}_FC_EU,     (= ENR_{CON or CO2}_TOT_EU - ENR_{CON or CO2}_EL_EU). */
/* This sums up to #countries x 4 + 4 <= 56 consumption or #countries x 3 + 3 <= 42 emission aggregates. */
/* */
/* Second, the fuel aggregates OIL = BD+GO+RD+DI+OI in MO sectors (not RE and EL in MO sectors) plus OIL in */
/* ST sectors. This yields the extra aggregates: */
/*      23: #countries x 2 <= 26 OIL totals for the country-TP and -TF sector specific combinations, named */
/*          OIL_{CON or CO2}_{TP and TF}_country, */
/*      24: #countries <= 13 OIL totals for the country-MO sectors specific combinations, named */
/*          OIL_{CON or CO2}_MO_country, */
/*      25: #countries x 2 <= 26 OIL totals for the country-FC and -TOT sector specific combinations, named */
/*          OIL_{CON or CO2}_{FC and TOT}_country, */
/*      26: 2 OIL totals for the TP and TF sectors (aggregating over all countries), named */
/*          OIL_{CON or CO2}_{TP and TF}_EU, */
/*      27: 1 OIL totals for the MObile sectors (aggregating over all countries), named */
/*          OIL_{CON or CO2}_MO_EU, */
/*      28: 2 totals for the FC and TOT sectors (aggregating over all countries), named */
/*          OIL_{CON or CO2}_{FC and TOT}_EU. */
/* This sums up to #countries x 5 + 5 <= 70 consumption or emission aggregates. Since FA sector uses only one OIL product */
/* OI, it is not included in the fuel-sector aggregates (i.e. in 23 and 26). */
```

```
/* Third, the fuel consumption aggregate ELE = RE+EL in MO sectors and ELE in ST sectors, yielding the extra aggregates: */
/*      29: #countries x 2 <= 26 OIL totals for the country-MO and -TOT sector specific combinations, named */
/*          ELE_CON_{MO and TOT}_country, */
/*      30: 2 ELE totals for the MO and TOT sectors, (aggregating over all countries), named */
/*          ELE_CON_{MO and TOT}_EU. */
/* This sums up to #countries x 2 + 2 <= 28 consumption aggregates. There is no calculation of ELEctricity aggregate if */
/* CO2 emissions are calculated, since ELEctricity emits no CO2. */
/* */
/* Fourth, the fuel aggregate NGS = GA in MO sectors and NGS in ST sectors, yielding the extra aggregates: */
/*      31: #countries  <= 13 gas totals for the country (aggregating over all ST sectors and TP sectors), named */
/*          NGS_{CON or CO2}_TOT_country, */
/*      32: 1 NGS total (aggregating over all countries) named */
/*          NGS_{CON or CO2}_TOT_EU. */
/* This sums up to #countries + 1 <= 14 consumption or CO2 aggregates. */
/* */
/* The total number of calculated new series are (1) #countries x 29 + 56 <= 433 consumption aggregates or */
/* (2) #countries x 43 + 43 <= 602 emission variables. */

/* Macro programming: The macro uses local variables to hold both scalar constants and strings that are letter codes for */
/* the fuels, sectors and countries. Looping through the fuels, sectors and countries the letter codes are concatenated */
/* into names of temporary TROLL variables. These memory (not disk) variables are initialized to zero and then updated with */
/* disaggregated figures in (nested) loops. The disaggregated figures are read from FORMDATA output files from simulations */
/* of the SEEM model. In the case of calculating emmision figures, the consumption disaggregates are first multiplied with */
/* emission coefficients. (Fuel expenditurs can be calculated the same way, by multiplying consumption figures with fuel */
/* prices (and exchange rates)). After final updates the temporary TROLL variables are written into an output FORMDATA */
/* file. The number of the input scenario is used for naming both the input files and the single output file. */
/* */
/* Names of the aggregate energy consumption or CO2-emission variables written to the output FORMDATA file */
/* are all concatenations of 4 components: */
/*      1: a disaggregated fuel code or the all fuels aggregated code ENR, */
/*      2: the energy consumption code CON or the CO2-emission code CO2, */
/*      3: a single sector code or one of the sector aggregation codes ST, MO, FC or TO, */
/*      4: a single country code or the code for the number of countries included in the summation C# or EU. */
/* A disaggregated output variable may e.g. be named OIL_CO2_HO_NO, while a most aggregated variable may be */
/* named ENR_CON_TOT_EU, where the _ separating the different components of the names is dropped when storing */
/* the files in the FORMDATA file on disk. */

ADDFUN MAIN;
PROCEDURE MAIN()
BEGIN;

/* ---------- USER INTERFACE AND INITIALIZING ---------- */

    /* First the user decides whether the macro should calculate emission figures or only sum up the energy consumption */
    /* figures already simulated. Later, the, boolean variables will decide whether energy consumption figures have to be */
    /* multipied with CO2-emission coefficients: */

    GET FROM TERMINAL NUMBER task "\n Answer 1 for summing energy consumption or 2 \n for calculating and
summing CO2-emission figures. \n Any other answer terminates the macro: ";
    co2_emission = task == 2;                    /* TRUE implies calculating CO2-emissions and all their aggregates */
    energy_summation = task == 1;                /* TRUE implies calculating all aggregates of energy consumptions */

    /* Electricity, nuclear power and renewables count when summing up energy consumption, but not when calculating */
    /* CO2-emission. Hence the difference in (the numbers of) STationary fuels in the two cases. For the STationary */
    /* sectors we have that */
    /*  EL sector uses          NUC, REN, COA, OIL, NGS,      i.e. fuel no. 1 to 5, while emitting from COA, OIL NGS,*/
    /*  HO, SE and IN sector uses          COA, OIL, NGS, ELE, i.e. fuel no. 3 to 6, while emitting from COA, OIL NGS,*/
```

```
/* Rail Electricity and ELectricity count when summing up energy consumption, but not when calculating CO2-emmision. */
/* For the MObile sectors we have that                                                                                */
/*   TP sector uses BD, GA, GO, RE, RD, DI,          i.e. fuel no. 1 to 6, while emitting from all but RE,            */
/*   TF sector uses                  DI, EL, OI, i.e. fuel no. 6 to 8, while emitting from DI and OI,                 */
/*   FA sector uses and emits from               OI, i.e. fuel no. 8.                                                 */

IF (energy_summation) THEN BEGIN;          /* The consumption of hydroelectric energy (ELE and RE and EL) counts */
    summ          = "CON";                 /* Name postfix reflecting summation of energy consumption figures */
    ST_fuel       = COMBINE("NUC", "REN", "COA", "OIL", "NGS", "ELE");
    final_ST_fuel = 6;
    MO_fuel       = COMBINE("BD", "GA", "GO", "RE", "RD", "DI", "EL", "OI");
    final_MO_fuel = 8;
    final_TP_fuel = 6;                                                    /* TP uses the first 6 fuels */
END ELSE IF (co2_emission) THEN BEGIN;     /* Hydroelectric energy does not contribute to any CO2-emission */
    summ          = "CO2";                 /* Name postfix reflecting calculation of emmision consumption figures */
    coef_NGS      = 0.0024;                /* Amount of CO2-emission pr energy unit of the various fuels */
    coef_GA       = 0.0024;
    coef_OIL      = 0.0031;
    coef_BD       = 0.0031;
    coef_GO       = 0.0031;
    coef_RD       = 0.0031;
    coef_DI       = 0.0031;
    coef_COA      = 0.0039;
    ST_coef       = COMBINE(coef_COA, coef_OIL, coef_NGS);   /* Array corresponding to the array ST_fuel */
    ST_fuel       = COMBINE("COA", "OIL", "NGS");   /* NUC, REN and ELE is ommited due to zero CO2 emission */
    final_ST_fuel = 3;
    MO_coef       = COMBINE(coef_BD, coef_GA, coef_GO, coef_RD, coef_DI, coef_OIL);
    MO_fuel       = COMBINE("BD", "GA", "GO", "RD", "DI", "OI"); /* RE and EL is ommited due to zero CO2 emission */
    final_MO_fuel = 6;
    final_TP_fuel = 5;                                    /* TP uses the first 5 fuels (which all emit CO2) */
END ELSE BEGIN; PRINT("\n Wrong answer, ending macro! "); EXIT(); END;          /* Error message and exit */

/* The user then decides whether to sum up for a single country, a few countries or for all 13 countries:    */

GET FROM TERMINAL choice "\n Decide what countries to sum up. Answer the letter S \n for a singel country, the letter
M for many (or a few) countries \n or the letter A for all 13 countries. \n Any other answer terminates the macro: ";
choice = UPPER(choice);                                                   /* User's choice = S, M or A */
single_country = choice == "S";
many_countries = choice == "M";
all_countries = choice == "A";

IF (single_country) THEN BEGIN;                                  /* Construct a single entry array */
    GET FROM TERMINAL answer "\n Give ONE SINGLE country code from the following: AU, BE, \n BR, CH, DK, FR,
GB, IT, NL, NO, SF, SP or SW: ";
    all_country = COMBINE(UPPER(answer));                                 /* Single country code */
    final_country = 1;
    geo = "C1";                                                           /* Code for country aggregation */
END; /* IF single_country */
ELSE IF (many_countries) THEN BEGIN;          /* Construct an array with user's country codes as entries */
    all_country = COMBINE();                                              /* Empty array of zero length */
    final_country = 0;                                                    /* Initializing */
    GET FROM TERMINAL answer "\n List countries by the following codes: \n AU, BE, BR, CH, DK, FR, GB, IT, NL, NO,
SF, SP or SW \n separated by blanks and terminated bya blank and \n a semicolon (e.g. like NO SF SW ;): ";
    WHILE (answer <> ";") BEGIN;                                          /* Not finnished country input */
        all_country = COMBINE(all_country, UPPER(answer));                /* Adding next country to the array */
        final_country = final_country + 1;
        GET FROM TERMINAL answer;
    END; /* WHILE answer */
```

```
    geo = "C" || CONVERT(final_country);                    /* CONVERT changes number to string */
    IF (final_country == 0) THEN BEGIN; PRINT("\n Wrong option, ending macro! "); EXIT(); END;/* Error message, exit */
END; /* IF many_countries */

ELSE IF (all_countries) THEN BEGIN;
    all_country = COMBINE("AU", "BE", "BR", "CH", "DK", "FR", "GB", "IT", "NL", "NO", "SF", "SP", "SW");
    final_country = 13;
    geo = "EU";                                             /* Code for country aggregation */
END; /* IF all_countries */
ELSE BEGIN; PRINT("\n Wrong option, ending macro! "); EXIT(); END;          /* Error message and exit */

/* The SEEM simulation scenario on which to base the calculations have to be input. Later the scenario code goes into the*/
/* name of the FORMDATA file containing the individual TROLL files (variables). The user chooses between reading input */
/* files and storing output file in the format of either FORMDATA or FAME databases.                                    */

GET FROM TERMINAL scenario "\n Give the code (e.g. IS, FS, NS) of the scenario which data to calculate/summarize: ";
GET FROM TERMINAL in_format "\n Write FORMDATA or FAME for the format of input files: ";
IF (UPPER(in_format) == "FAME" ) THEN in_extension  = ".db"; ELSE in_extension = ".txt";

/* The calculated TROLL DATA files will be written into one single FORMDATA or FAME file (for all countries), containing */
/* disaggregated and aggregated emissions or aggregated energy use. The filename is converted to lowercase letters,     */
/* being common in the UNIX environment (and of no consequences under MS DOS):                                          */

GET FROM TERMINAL out_format "\n Write FORMDATA or FAME for desired output format: ";
IF (UPPER(out_format) == "FAME" ) THEN out_extension = ".db"; ELSE out_extension = ".txt";
out_file = LOWER("sum" || scenario || summ || out_extension);/* Concatenating name of FAME or FORMDATA  output file */
IF (HOSTEXIST(out_file)) THEN BEGIN;          /* The (name of the) FORMDATA file to be written does already exist */
    PRINT("Formdata output file ", out_file, " already exists. Do you want overwrite it,");     /* Informing user of macro */
    GET FROM TERMINAL answer "creating a new file with the same name (answer Y or N)? ";
    IF (UPPER(answer) == "N" ) THEN BEGIN;
        PRINT("\n NOT overwriting existing file, thus ending macro! "); EXIT();
    END;
END;
>> ACCESS outfile TYPE &(out_format) ID &(out_file) MODE C;          /* Creating new FORMDATA or FAME file */
>> SEARCH DATA outfile W;      /* Set up to write results to the accessed outputfile */

/* ---------------------------------- CALCULATING, UPDATING AND STORING ---------------------------------- */

ST_sector       = COMBINE("EL", "HO", "IN", "SE");      /* Putting sector codes into vectors. The sectors ... */
final_ST_sector = 4;                                    /* ...are the same no matter what the user answers */
MO_sector       = COMBINE("TP", "TF", "FA");                          /* Vital order of MO sectors */
final_MO_sector = 3;

/* Main loop through the 13 countries. Both the STationary and the MObile sources are dealt with in this loop:         */

FOR (i = 1; i <= final_country; i = i + 1) BEGIN;                     /* Looping through the countries */
    first_country = i == 1;                                                          /* =TRUE or FALSE */
    last_country = i == final_country;                                               /* =TRUE or FALSE */
    country = all_country[i];                                        /* Getting country name from vector */
    PRINT("Summing up for country: ", country);          /* Informing user which country is being handled */

    /* The TROLL DATA files containing the simulation results for the specified scenario are read from a single        */
    /* FORMDATA file for each country. All disaggregated energy consumption figures are read from this file:           */

    in_file = LOWER(country || "/" || "out" || scenario || country || in_extension);     /* Name of FORMDATA file to read */
    IF (NOT HOSTEXIST(in_file)) THEN BEGIN; PRINT("Does not find the input file: ", in_file); EXIT(); END;
    >> ACCESS infile TYPE &(in_format) ID &(in_file) MODE R;
    >> SEARCH FIRST DATA infile;                          /* Ready to read energy use in current country */
```

```
/* ---------- STATIONARY SOURCES ---------- */

/* For each country looping through 4 or 3 fuels within 4 sectors calculating all CO2-emissions or energy-use */
/* aggregates for the STationary sources:                                                                      */

FOR (j = 1; j <= final_ST_sector; j = j + 1) BEGIN;          /* Looping through 4 sectors: EL, HO, IN and SE */
    first_ST_sector = j == 1;                                                          /* =TRUE or FALSE */
    second_ST_sector = j == 2;                                /* =TRUE if first Final Consumption sector else FALSE */
    last_ST_sector = j == final_ST_sector;                                             /* = FALSE or TRUE */
    sector = ST_sector[j];                                                  /* Getting sector name from vector */

    /* Deciding which fuels to scan depending on type of calculation (CO2 or not) and current sector. Then scan: */

    first = 1;                                                                          /* Default start index */
    last = final_ST_fuel;                                                               /* Default end index */
    IF (energy_summation) THEN          /* only first or last has to be altered (ELSE first and last are correct) */
        IF (sector == "EL") THEN        /* If calculating CO2-emission no emission from NUC, REN and ELE fuel */
            last = 5;                                                    /* last = 5 since EL sector uses no ELE fuel */
        ELSE first = 3;                                                  /* first = 3 since NUC and REN only in EL sector */

    FOR (k = first; k <= last; k = k + 1) BEGIN;     /* Looping through 3 (4) fuels: COA, OIL, NGS (and ELE) */
        first_ST_fuel = k == first;                                                     /* = TRUE or FALSE */
        last_ST_fuel = k == last;                                                       /* = FALSE or TRUE */
        fuel = ST_fuel[k];                                                  /* Getting country name from vector */
        variable = fuel || "CON" || sector || country;                                  /* Concatenating filename */

        /* 1: Calculating and storing #countries x 4 x 3 = #countries x 12 <= 156 country-sector-fuel specific emission */
        /* disaggregates or just reading country-sector-fuel specific energy consumptions:                              */

        IF (co2_emission) THEN BEGIN;                                   /* Calculating disaggregated CO2-emission */
            co2_coef = ST_coef[k];                                      /* Getting the right emission coeffisient */
            >> DOCORE fuel_use = GETDATA("&variable", -1);              /* Reading consumption from file */
            >> DOCORE disaggregate = &(co2_coef) * fuel_use;            /* CO2-emission in fuel-sector-country */
            name = fuel || "CO2" || sector || country;                  /* Concatenating name of FORMDATA file variable */
            >> DO PUTDATA(disaggregate, "&name", -1);                   /* Write emission to disk file variable */
        END; /* IF co2_emission */
        ELSE >> DOCORE disaggregate = GETDATA("&variable", -1);         /* Only energy consumptions */

        /* 3: Initializing, updating and/or storing #countries x (6 or 3) <= (78 or 39) totals for the country-fuel */
        /* combinations (aggregating over all ST sectors):                                                          */

        variable = fuel || "_" || summ || "_ST_" || country;                            /* Name of local variable */
        IF (first_ST_sector) THEN                                                       /* EL sector */
            >> DOCORE &variable = disaggregate;                                         /* Initializing */
        ELSE BEGIN;                                                                     /* Not EL sector */
            IF (second_ST_sector AND fuel == "ELE") THEN
                >> DOCORE &variable = disaggregate;                     /* Initializing ELE_CON_ST_country */
            ELSE >> DOCORE &variable = &variable + disaggregate;        /* Updating with sector's emission/usage */

            /* 19: In addition: #countries x (3 or 2) <= (39 or 26) Final Consumption totals = ST total - fuel in EL sector.*/
            /* The variable OIL_&summ_FC_&country is not stored after final update since OIL_&summ_MO_&country*/
            /* has to be added before storing it on disk.                                                                  */

            xtra_variable = fuel || "_" || summ || "_FC_" || country;           /* Name of an extra local variable */
            IF (second_ST_sector) THEN
                >> DOCORE &xtra_variable = disaggregate;                        /* Initializing FC total for fuel or... */
            ELSE >> DOCORE &xtra_variable = &xtra_variable + disaggregate;      /* ...updating */
```

```
      IF (last_ST_sector) THEN BEGIN;                                  /* Final sector hence store... */
         IF (fuel <> "OIL") THEN                                       /* ...non OIL fuel in a... */
            >> DO PUTDATA(&xtra_variable, "&(fuel)&(summ)FC&country", -1);   /* ...fuel-country FC-total */
         >> DO PUTDATA(&variable, "&(fuel)&(summ)ST&country", -1);    /* ...ST-total from fuel-country */

         /* 7: Last sector in country: the country total can be added to the ST total (aggregating over all ST   */
         /* sectors and all countries to get 6 or 3 totals). Note that 'variable' contains the country's ST total. */
         /* 20: In parallel with 7: initializing, updating and storing 3 or 2 fuel-FC totals. The variable        */
         /* OIL_&summ_FC_&geo needs updating with the MObile total before storing (hence it is not stored)  */

         IF (first_country) THEN BEGIN;
            >> DOCORE &(fuel)_&(summ)_FC_&geo = &xtra_variable;          /* Initializing FC total */
            >> DOCORE &(fuel)_&(summ)_ST_&geo = &variable;               /* Initializing ST total */
         END; /* IF first country */
         ELSE BEGIN;                                                     /* Updating and storing */
            >> DOCORE &(fuel)_&(summ)_FC_&geo = &(fuel)_&(summ)_FC_&geo + &xtra_variable;
            >> DOCORE &(fuel)_&(summ)_ST_&geo = &(fuel)_&(summ)_ST_&geo + &variable;
            IF (last_country) THEN BEGIN;                                /* Final country for fuel hence storing */
               IF (fuel <> "OIL") THEN
                  >> DO PUTDATA(&(fuel)_&(summ)_FC_&geo, "&(fuel)&(summ)FC&geo", -1); /* Non OIL*/
               >> DO PUTDATA(&(fuel)_&(summ)_ST_&geo, "&(fuel)&(summ)ST&geo", -1);
            END; /* IF last country */
         END; /* ELSE not first country */
      END; /* IF last ST sector */
   END; /* ELSE not first ST sector */

   /* 4: Initializing, updating and/or storing (5 or 3) + 3 x (4 or 3) =(17 or 12) totals for the fuel in sector   */
   /* combinations (aggregating over all countries):                                                              */

   variable = fuel || "_" || summ || "_" || sector || "_" || geo;       /* Name of local variable */
   IF (first_country) THEN
      >> DOCORE &variable = disaggregate;                               /* Initializing or... */
   ELSE BEGIN;
      >> DOCORE &variable = &variable + disaggregate;                   /* ...updating with country's emission/fuel */
      IF (last_country) THEN                                            /* Final country: store... */
         >> DO PUTDATA(&variable, "&(fuel)&(summ)&(sector)&geo", -1);   /* ...total from fuel in sector */
   END; /* ELSE not first country */

   /* 2: Initializing or updating (and storing immediately after the fuel-loop) #countries x 4 <= 52 totals for the */
   /* country-sector combinations (aggregating over all STationary fuels):                                          */

   variable = "ENR_" || summ || "_" || sector || "_" || country;        /* Name of local variable */
   IF (first_ST_fuel) THEN
      >> DOCORE &variable = disaggregate;                               /* Initializing or... */
   ELSE >> DOCORE &variable = &variable + disaggregate;                 /* ...updating with fuel emission/consumption */
END; /* fuel-loop */
>> DO PUTDATA(&variable, "ENR&(summ)&(sector)&country", -1);           /* Storing total from sector-country */

/* 6: Initializing, updating and/or storing 4 totals for the sectors (aggregating over all STationary fuels and all */
/* countries). Note that variable = ENR_&(summ)_&(sector)_&country from above:                                       */

xtra_variable = "ENR_" || summ || "_" || sector || "_" || geo;          /* Name of local variable */
IF (first_country) THEN
   >> DOCORE &xtra_variable = &variable;                                /* Initializing with first country's sector total or... */
ELSE BEGIN;
   >> DOCORE &xtra_variable = &xtra_variable + &variable;               /* ... updating with it */
   IF (last_country) THEN                                               /* Final country for sector... */
      >> DO PUTDATA(&xtra_variable, "ENR&(summ)&(sector)&geo", -1);     /* ...storing total from sector */
END; /* ELSE not first country */
```

```
    /* 5: Initializing or updating (and storing immediately after the sector-loop) #countries <= 13 totals for the countries */
    /* (aggregating over all STationary fuels and sectors). Note that from above we have                                    */
    /* variable = ENR_&(summ)_&(sector)_&country:                                                                           */

    xtra_variable =  "ENR_" || summ || "_ST_" || country;                                  /* Name of local variable */
    IF (first_ST_sector) THEN
        >> DOCORE &xtra_variable = &variable;                        /* Initializing with first country's sector total or... */
    ELSE >> DOCORE &xtra_variable = &xtra_variable + &variable;                                /* ...or updating with it */
END; /* sector-loop */
>> DO PUTDATA(&xtra_variable, "ENR&(summ)ST&country", -1);                                  /* Storing total ST-country */

/* 8: Initializing and updating (and storing after the country-loop) 1 total for from STationary sources (aggregating      */
/* over all STationary country-sector-fuel combinations). Update ST total with country-ST total. Note that we have       */
/* xtra_variable = ENR_&(summ)_ST_&country from above:                                                                    */

variable =  "ENR_" || summ || "_ST_" || geo;                                               /* Name of local variable */
IF (first_country) THEN
    >> DOCORE &variable = &xtra_variable;                                                    /*  Initializing total ST */
ELSE>> DOCORE &variable = &variable + &xtra_variable;                                 /* updating total ST with country */

/* ---------------------------------------- MOBILE SOURCES ---------------------------------------- */

/* For each country looping through different fuel subset within the 3 sectors TP, TF and FA calculating all             */
/* CO2-emissions or energy-use aggregates for the MObile sources:                                                         */

FOR (j = 1; j <= final_MO_sector; j = j + 1) BEGIN;          /* Looping through 3 MObile sectors: TP and TF and FA */
    first_MO_sector = j == 1;                                                                  /* = TRUE or FALSE */
    last_MO_sector = j == final_MO_sector;                                                     /* = FALSE or TRUE */
    sector = MO_sector[j];

    /* Deciding which fuels to loop through for the current sector, as different MObile sectors use different fuels:      */

    IF (sector == "TP") THEN BEGIN;                    /* TP uses the 5/6 first fuels BD, GA, GO, (RE), RD and DI*/
        first_fuel = 1;
        last_fuel = final_TP_fuel;                                                             /* = 6 or 5 */
    END; /* IF TP-sector */
    ELSE IF (sector == "TF") THEN BEGIN;          /* TF uses the 3/2 next fuels DI, (EL) and the oil product OI */
        first_fuel = final_TP_fuel;                                          /* = 6 if CONsumption or 5 if CO2 */
        last_fuel = final_MO_fuel;                                           /* = 8 if CONsumption or 6 if CO2 */
    END; /* IF TF-sector */
    ELSE BEGIN;                                                  /* FA sector uses only the oil product OI */
        first_fuel = final_MO_fuel;                                          /* = 8 if CONsumption or 6 if CO2 */
        last_fuel = first_fuel;
    END; /* IF FA-sector */

    /* For each sector in each country loop through a subset of BD, GA, GO, (RE,) RD, DI, (EL) and OI:                   */

    FOR (k = first_fuel; k <= last_fuel; k = k + 1) BEGIN;   /* Looping through 6 or 5 and 3 or 2 and finally just 1 fuel */
        first_sector_fuel = k == first_fuel;                                                   /* = TRUE or FALSE */
        last_sector_fuel = k == last_fuel;                                                     /* = FALSE or TRUE */
        first_MO_fuel = k == 1;                                                                /* = TRUE or FALSE */
        last_MO_fuel = k == final_MO_fuel;                                                     /* = FALSE or TRUE */
        fuel = MO_fuel[k];                                                    /* Getting fuel name from vector */
        IF (fuel == "OI") THEN
            SEEM_fuel = "OIL";          /* SEEM output fuel name = OIL different from macro output fuel name = OI */
        ELSE SEEM_fuel = fuel;                                  /* SEEM output fuel name = macro output fuel name */
        variable = SEEM_fuel || "CON" || sector || country;          /* Concatenating name of SEEM output variable */
```

```
/* 9: Calculating #countries x 8 <= 104 emissions or reading energy disaggregates in sector-country:      */

IF (co2_emission) THEN BEGIN;                                  /* Calculating disaggregated CO2-emission */
    co2_coef = MO_coef[k];                                     /* Getting the right emission coeffisient */
    >> DOCORE fuel_use = GETDATA("&variable", -1);             /* Reading consumption from file */
    >> DOCORE disaggregate = &(co2_coef) * fuel_use;           /* CO2-emission in fuel-sector-country */
    name = fuel || "CO2" || sector || country;                 /* Concatenating name of FORMDATA file variable */
    >> DO PUTDATA(disaggregate, "&name", -1);                  /* Writing emission to disk file variable */
END; /* IF co2_emission */
ELSE BEGIN;                                                    /* Energy consumption is... */
    >> DOCORE disaggregate = GETDATA("&variable", -1);         /*... fuel-sector-country specific */

    /* 29: If fuel is electricity, calculating #countries x 2 <= 26 ELE-CON totals for MO and TOT country. The 2*/
    /* ELE fuels in the MO sectors are visited once, hence we do the storing once in the fuel-sector loop:    */

    IF (fuel == "RE") THEN                                     /* El-consumption in TP sector */
        >> DOCORE ELE_CON_MO_&country = disaggregate;          /* initializing El aggregate for country */
    ELSE IF (fuel == "EL") THEN BEGIN;                         /* El-consumption in TF sector */
        variable = "ELE_CON_MO";                               /* Updating El aggregate for country with consumption in ... */
        >> DOCORE &(variable)_&country = &(variable)_&country + disaggregate;   /* ...TF sector */
        >> DO PUTDATA(&(variable)_&country, "ELECONMO&country", -1);            /* ...and storing total */
        xtra_variable = "ELE_CON_TOT";                         /* Total ELEctricity consumption */
        >> DOCORE &(xtra_variable)_&country = &(variable)_&country + ELE_CON_ST_&country;
        >> DO PUTDATA(&(xtra_variable)_&country, "ELECONTOT&country", -1);      /* ...and storing */

        /* 30: Initializing and updating 2 ELEctricity consumption MO and TOT total:                        */

        IF (first_country) THEN                                /*Initializing total MObile ELE-consumption */
            >> DOCORE &(variable)_&geo = &(variable)_&country; /* ...with consumption in country... */
        ELSE BEGIN;                                            /* ...or updating... */
        >> DOCORE &(variable)_&geo = &(variable)_&geo + &(variable)_&country;   /* ...with it */
            IF (last_country) THEN BEGIN;
                >> DO PUTDATA(&(variable)_&geo, "ELECONMO&geo", -1);    /* ...and storing total MO */
                >> DOCORE &(xtra_variable)_&geo = &(variable)_&geo + ELE_CON_ST_&geo;
                >> DO PUTDATA(&(xtra_variable)_&geo , "ELECONTOT&geo", -1);     /* ...and storing */
            END; /* IF last country */
        END; /* ELSE IF not first country */
    END; /* ELSE IF last electricity fuel in MObile sectors */
END; /* ELSE energy consumption */
IF (fuel == "GA") THEN                                         /* Saving disaggregate for operation 31 below */
    >> DOCORE GA_&(summ)_TP_&country = disaggregate;

/* 11: Initializing updating and/or storing #countries x 2 <= 26 totals for those-fuel combinations that    */
/* are different from the disaggregates. When a MObile sectoris the only one using the current fuel the    */
/* aggregate &(fuel)&(summ)MO&country = disaggregate is not stored.                                         */
/* 15: Aggregating the fuel-country combinations over all countries for the MObile fuels DI and OI, which are */
/* used in two MO sectors:                                                                                  */

IF (fuel == "DI") THEN                                         /* DI in TP and TF sector */
    IF (sector == "TP") THEN                                   /* Sector = TP */
        >> DOCORE DI_var = disaggregate;                       /* Remembering DI fuel to add to TF's DI series */
    ELSE BEGIN;                                                /* Sector = TF */
        >> DOCORE DI_var = DI_var + disaggregate;              /* Adding DI fuel  to TP's DI series */
        >> DO PUTDATA(DI_var, "DI&(summ)MO&country", -1);      /* ...and storing as DI-country total */
        variable = "DI_" || summ || "_MO_" || geo;             /* Name of local variable */
        IF (first_country) THEN                                /* Total DI fuel*/
            >> DOCORE &variable = DI_var;                      /*  Initializing or.... */
        ELSE BEGIN;
            >> DOCORE &variable = &variable + DI_var;          /* ...updating */
```

```
            IF (last_country) THEN                                  /* Final country for fuel... */
                >> DO PUTDATA(&variable, "DI&(summ)MO&geo", -1);    /* ...storing fuel total */
        END; /* ELSE not first country */
    END; /* ELSE TF sector */
ELSE IF (fuel == "OI") THEN                                         /* OI in TF and FA sector */
    IF (sector == "TF") THEN                                        /* Sector = TP */
        >> DOCORE OI_var = disaggregate;        /* Remembering OI fuel to add to FA's OI series */
    ELSE BEGIN;                                                     /* Sector = FA */
        >> DOCORE OI_var = OI_var + disaggregate;       /* Adding OI fuel  to TF's OI series */
        >> DO PUTDATA(OI_var, "OI&(summ)MO&country", -1);           /* Storing OI-country total */
        variable = "OI_" || summ || "_MO_" || geo;                  /* Name of local variable */
        IF (first_country) THEN                                     /* Total OI fuel*/
            >> DOCORE &variable = OI_var;                           /*  Initializing or.... */
        ELSE BEGIN;
            >> DOCORE &variable = &variable + OI_var;               /* ...updating */
            IF (last_country) THEN                                  /* Final country for fuel... */
                >> DO PUTDATA(&variable, "OI&(summ)MO&geo", -1);    /* ...storing fuel total */
        END; /* ELSE not first country */
    END; /* ELSE FA sector */

/* 12: Initialize, update and/or store (6 or 5) + (3 or 2) + 1 = (10 or 8) totals for each fuel-sector combination  */
/* (aggregating over all countries)                                                                                 */

variable = fuel || "_" || summ || "_" || sector || "_" || geo;      /* Name of local variable */
IF (first_country) THEN
    >> DOCORE &variable = disaggregate;             /* Initialize with value for the first country */
ELSE BEGIN;                                                         /* Updating fuel in sector... */
    >> DOCORE &variable = &variable + disaggregate;     /* ...with country's emission/consumption */
    IF (last_country) THEN                                          /* Final country... */
        >> DO PUTDATA(&variable, "&(fuel)&(summ)&(sector)&geo", -1);    /* ...storing total fuel in sector */
END; /* ELSE not first country */

/* 23: TP and TF sectors use several oil products, hence initializing, updating and storing the #countries x 2  */
/* <= 26 oil aggregates in TP and TF sector. FA sector is not included since it uses only OI:                   */

IF (fuel <> "GA" AND fuel <> "RE" AND fuel <> "EL") THEN BEGIN;     /* OIL fuel in sector */
    IF (sector <> "FA") THEN BEGIN;                                 /* TF or TP sector */
        variable = "OIL_" || summ || "_" || sector || "_" || country;   /* Name of local variable */
        IF (first_sector_fuel) THEN         /* Initializing oil aggregate for sector in country... */
            >> DOCORE &variable = disaggregate;             /* ...with fuel consumption/emission */
        ELSE BEGIN;                         /* Updating oil aggregate for sector in country... */
            >> DOCORE &variable = &variable + disaggregate;     /* ...with fuel consumption/emission */
            IF (last_sector_fuel) THEN BEGIN;                       /* Final update hence... */
                >> DO PUTDATA(&variable, "OIL&(summ)&(sector)&country", -1);    /* ...storing total */

                /* 26: Updating 2 OIL aggregates in TP and TF sector                                    */

                xtra_variable = "OIL_" || summ || "_" || sector || "_" || geo;  /* Name of local variable */
                IF (first_country) THEN
                    >> DOCORE &(xtra_variable) = &variable;     /* Initializing oil aggregate for sector */
                ELSE BEGIN;
                    >> DOCORE &(xtra_variable) = &(xtra_variable) + &variable;/* Updating sector aggregate */
                    IF (last_country) THEN
                        >> DO PUTDATA(&(xtra_variable), "OIL&(summ)&(sector)&geo", -1);/* ...storing total */
                END; /* ELSE not first country */
            END; /* IF last sector fuel */
        END; /* ELSE not first sector fuel */
    END; /* IF not FA sector */
```

```
        /* 24: Initializing, updating and/or storing #countries <= 13 total OIL aggregates in MO sectors in country:  */

        variable = "OIL_" || summ || "_MO_" || country;
        IF (first_MO_sector AND first_MO_fuel) THEN              /* Initializing oil aggregate for MO in country... */
            >> DOCORE &variable = disaggregate;                     /* ...with consumption/emission in sector */
        ELSE BEGIN;                                             /* Updating oil aggregate for MO in country... */
            >> DOCORE &variable = &variable + disaggregate;         /* ...with consumption/emission in sector */
            IF (last_MO_sector AND last_MO_fuel)  THEN BEGIN;                   /* Final sector update... */
                >> DO PUTDATA(&variable, "OIL&(summ)MO&country", -1);   /* storing total OIL in MO-country */
                >> DOFILE OIL&(summ)FC&country = OIL_&(summ)_FC_&country + &variable;/* FC i ST + MO */

                /* 27: Initializing, updating and/or storing 1 total OIL consumption/emission in all MObile sectors:  */

                xtra_variable =  "OIL_" || summ || "_MO_" || geo;               /* Name of local variable */
                IF (first_country) THEN
                    >> DOCORE &(xtra_variable) = &variable;             /* Initializing total oil in MO sectors */
                ELSE BEGIN;
                    >> DOCORE &(xtra_variable) = &(xtra_variable) + &variable;          /* Updating total oil... */
                    IF (last_country) THEN BEGIN;
                        >> DO PUTDATA(&(xtra_variable), "OIL&(summ)MO&geo", -1);  /* Storing total in sector */
                        >> DOFILE OIL&(summ)FC&geo = OIL_&(summ)_FC_&geo + &(xtra_variable);
                    END; /* IF last country */
                END; /* ELSE not first country */
            END; /* IF last.... */
        END; /* ELSE not first MO sector */
    END; /* IF OIL fuel */

    /* 10: Updating and storing #countries x 2 <= 26 totals for the country-sector combination (aggregating over   */
    /* all MObile fuels). Since FA uses the single OI fuel, only TP and TF are included:                            */

    variable = "ENR_" || summ || "_" || sector || "_" || country;               /* Name of local variable */
    IF (sector <> "FA") THEN                                                        /* TP or TF sector */
        IF (first_sector_fuel) THEN                                             /* Initializing fuel total with ... */
            >> DOCORE &variable = disaggregate;                 /* ...sector's first fuel's emission/consumption */
        ELSE BEGIN;                                                             /* Updating fuel total with ... */
            >> DOCORE &variable = &variable + disaggregate;         /* ...with fuel's emission/consumption */
            IF (last_sector_fuel) THEN
            >> DO PUTDATA(&variable, "ENR&(summ)&(sector)&country", -1);    /* Storingsector-country total */
        END; /* ELSE not first fuel in sector */
    ELSE >> DOCORE &variable = disaggregate;                        /* variable = OI_&(summ)_FA_&country */
END; /* fuel-loop */

/* 14: Initializing, updating and storing 2 totals for the sectors TP and TF (aggregating over all MObile fuels and all:*/
/* countries). Since sector FA uses only one fuel OI, there is no fuel aggregation (ENR = OI):                          */

IF (sector <> "FA") THEN BEGIN;
    xtra_variable = "ENR_" || summ || "_" || sector || "_" || geo;              /* Name of local variable */
    IF (first_country) THEN
        >> DOCORE &xtra_variable = &variable;                   /* Initializing with first country's sector total */
    ELSE BEGIN;
        >> DOCORE &xtra_variable = &xtra_variable + &variable;      /* Updating with country's sector total */
        IF (last_country) THEN
            >> DO PUTDATA(&xtra_variable, "ENR&(summ)&(sector)&geo", -1);   /* Storing total from sector */
    END; /* ELSE not first country */
END; /* IF sector not FA */
```

```
    /* 13: Initializing and updating (but storing immediately after the sector-loop) #countries <= 13 totals for all MObile */
    /* sectors in the country, aggregating over all MObile fuels and sectors:                                              */

    xtra_variable = "ENR_" || summ || "_MO_" || country;                    /* Name of local variable */
    IF (first_MO_sector) THEN
        >> DOCORE &xtra_variable = &variable;                    /* Initializing with first country's sector total */
    ELSE >> DOCORE &xtra_variable = &xtra_variable + &variable;                          /* Updating with it */
END; /* sector-loop */
>> DO PUTDATA(&xtra_variable, "ENR&(summ)MO&country", -1);        /* Storing total in MO sectors in country */

/* 16: Initializing and updating (but storing after the country-loop) 1 total for MObile sources (aggregating over all      */
/* MObile sector-fuel combinations in all countries) with country's MO total. Note that we have xtra_variable holding      */
/* the string ENR_&(summ)_MO_&country from above, while variable is given a new string:                                   */

variable = "ENR_" || summ || "_MO_" || geo;                             /* Name of local variable */
IF (first_country) THEN
    >> DOCORE &variable = &xtra_variable;                                          /* Initializing or */
ELSE >> DOCORE &variable = &variable + &xtra_variable;                          /* Updating with country */

/* 17: Adding MObile sector totals to Stationary sector totals and storing as #countries <= 13 country-TOtals:           */

>> DOCORE ENR_&(summ)_TOT_&country = ENR_&(summ)_ST_&country + ENR_&(summ)_MO_&country;
>> DO PUTDATA(ENR_&(summ)_TOT_&country, "ENR&(summ)TOT&country", -1);

/* 21: Storing #countries <= 13 totals for FC sectors, aggregated over all fuels and sectors but EL:                     */

>> DOFILE ENR&(summ)FC&country = ENR_&(summ)_TOT_&country - ENR_&(summ)_EL_&country;

/* 25: Calculating #countries x 2 <= 26 TOtal and Final Consumptions or CO2-emissions from oil products in each           */
/* country, aggregated over all fuels and all sectors and FC = all sectors but EL:                                       */

>> DOCORE OIL_&(summ)_TOT_&country = OIL_&(summ)_ST_&country + OIL_&(summ)_MO_&country;
>> DO PUTDATA(OIL_&(summ)_TOT_&country, "OIL&(summ)TOT&country", -1);
>> DOFILE OIL&(summ)FC&country = OIL_&(summ)_FC_&country + OIL_&(summ)_MO_&country;

/* 31: Calculating #countries <= 13 TOtals for aggregated gas consumption or CO2-emission:                               */

>> DOFILE GAS&(summ)TOT&country = NGS_&(summ)_ST_&country + GA_&(summ)_TP_&country;

/* Delete search path and access to current country input file that is now done with, so that a new input file can       */
/* be accessed and searched by the alias file name infile:                                                               */

>> DELSEARCH infile;
>> DELACCESS infile;

END; /* main loop through the single country, the given countries or all 13 countries */

/* 8: Storing 1 total for all STationary sectors, aggregated over all fuels, all ST sectors and all countries:          */

>> DO PUTDATA(ENR_&(summ)_ST_&geo, "ENR&(summ)ST&geo", -1);

/* 16: Storing 1 total for all MObile sectors, aggregated over all fuels, all MO sectors and all countries:             */

>> DO PUTDATA(ENR_&(summ)_MO_&geo, "ENR&(summ)MO&geo", -1);

/* 18: Storing 1 TOtal energy consumption or CO2 emission, aggregated over all fuels, all sectors and all countries:    */

>> DOFILE ENR&(summ)TOT&geo = ENR_&(summ)_ST_&geo + ENR_&(summ)_MO_&geo;
```

```
    /* 22: Storing 1 total for FC sectors (= all sectors but EL), aggregated over all fuels, all sectors but EL and all countries:   */

    >> DOFILE ENR&(summ)FC&geo = ENR_&(summ)_ST_&geo + ENR_&(summ)_MO_&geo - ENR_&(summ)_EL_&geo;

    /* 28: Calculating 2 TOtals and Final Consumptions or CO2-emissions from oil products, aggregated over all fuels, all      */
    /* sectors - with FC = all sectors but EL - and all countries:                                                             */

    >> DOCORE OIL_&(summ)_TOT_&geo = OIL_&(summ)_ST_&geo + OIL_&(summ)_MO_&geo;
    >> DO PUTDATA(OIL_&(summ)_TOT_&geo, "OIL&(summ)TOT&geo", -1);
    >> DOFILE OIL&(summ)FC&geo = OIL_&(summ)_TOT_&geo - OIL_&(summ)_EL_&geo;

    /* 32: Calculating TOtal aggregated gas consumptions or CO2-emissions:                                                     */

    >> DOFILE GAS&(summ)TOT&geo = NGS_&(summ)_ST_&geo + GA_&(summ)_TP_&geo;

    /* Some special fuel aggregates used by ECN, summed across countries:                                                      */

    IF (final_country > 1) THEN BEGIN;
        >> DOFILE AIR&(summ)TOT&geo = OI_&(summ)_FA_&geo;
        >> DOFILE ROAD&(summ)TOT&geo = OIL_&(summ)_MO_&geo - OI_&(summ)_FA_&geo - RD_&(summ)_TP_&geo;
        >> DOFILE RAIL&(summ)DIE&geo = RD_&(summ)_TP_&geo;
        >> DOFILE WAT&(summ)TOT&geo = OI_&(summ)_TF_&geo;
        IF (energy_summation) THEN
            >> DOFILE RAILCONTOT&geo = ELE_CON_MO_&geo;
    END;

    /* Deleting TROLL's local variables, search paths and access to output database:                                           */

    >> DELCORE ALL;
    >> DELSEARCH ALL;
    >> DELACCESS ALL;
END; /* macro */
```

# 3 The TROLL macro sim2rain

The sim2rain macro (in the seem directory) provides a link between the energy model SEEM and the pollution model RAINS. The macro transforms energy consumption figures simulated by the SEEM model in such a way that they can replace a subset of the Official Energy Pathway (OEP) data in the RAINS model.

## 3.1 From SEEM output to RAINS input

The RAINS program and data system comes with a database that contains fuel-sector-country-year specific data for certain energy consumption scenarios. The data in RAINS' energy database are in a special non-readable format. But data can be exported and imported interactively when running the RAINS program. This chapter describes how to export a subset of the RAINS data and how to replace some figures in the subset with figures simulated by the SEEM model. Chapter 4 explains how to read the modified data back into the RAINS program.

### 3.1.1 RAINS' Official Energy Pathway fuels-over-sectors output file: the OEP table

Running the RAINS program interactively, the data can be exposed through different displays. One particular display is the *fuels-over-sectors* table for a single country in a reference year and a certain scenario. The RAINS model comes equiped with energy data for most European countries and a few scenarios. The relevant reference years are 1990, 1995 and 2000. Our starting point is energy consumption data for a scenario called the Official Energy Pathway (OEP)[3] . The fuels-over-sectors table displayed on the PC's screen can be saved to an ASCII text file. For France in the year 2000 the saved table with additional headlines looks like this:

```
 1
...............R.A.I.N.S...Version.6.0.-.1992.............(c).IIASA...PAGE...1
.ROUTE:13000000.............................................printed.on..08/11/95
.Official.Energy.Pathway.ECE.03/1992.....................................2000.....
.@FRA.....................................................................
.....................................................................
.......................ENERGY_USE_PER_SECTOR_AND_FUEL..(PJ).................
.....................................................................
..............CON.......PP......DOM......TRA......IND......OTH.....SUM.......
.....................................................................
.....BC............0........0........0........0........0........0........0.....
.....HC...........25......213.......44........0......122........0......404.....
.....DC............0........0.......41........0......206........0......247.....
.....MD............0........0......502......502.......65........0.....1069.....
.....HF...........226.......34.......45........0......108........0......413.....
.....LF............0........0........0......985.......85......417.....1487.....
.....GAS...........4.......13......691........0......497........0.....1205.....
.....OS............0........9........3........0.......26........0.......38.....
.....NUC...........0.....3737........0........0........0........0.....3737.....
.....HYD...........0......689........0........0........0........0......689.....
.....ELE.........337.....-1808......813.......34......463........0.....-161.....
.....DH............0........0........0........0........0........0........0.....
.....................................................................
.....SUM.........592.....2887.....2139.....1521.....1572......417.....9128.....
```

*TABLE 1: The 2000* fuels-over-sectors *OEP table for France as it looks when saved to the file* enedb.prt.

Dots are replaced for blanks (space) to show the fixed-positions format. Boldface type denotes the part of the file that can be read back into the RAINS model from an external file. The first column *TABLE 1* of contains the fuel names and the first row contains the sector names. The figures are the fuel-sector specific energy consumption figures. The fuels-over-sectors tables of data — one table for each country-year combination — can be displayed on screen and interactively exported out of the RAINS program and into an ASCII text file named enedb.prt by default. In a single RAINS session several tables may be exported. Each new table is then added to the bottom of the file enedb.prt. After the RAINS session the enedb.prt file should be renamed, since it will be cleared by the next table exporting RAINS session. The file contains a table of a fixed format for each country-

[3] The fuels-over-sectors table is displayed in the menu driven RAINS program by interactively choosing the options 1, 1, 1 and 3, then the OEP scenario and finally the country and the year, cf. page 2-7 in the RAINS manual and chapter 4. The table displayed on screen is written to a file named enedb.prt by issuing the command *print*. We do not consider other RAINS scenarios than the Official Energy Pathway.

year. Splitting the file of tables into several files of single tables yields one file for each country-year specific table. We have chosen to name all the exported OEP tables oep*ccyy*.prt, where *cc* denote a two-digit country number and *yy* denote the last two digits of the year. When splitting the export file (i.e. enedb.prt) into separate files (i.e. oep*ccyy*.prt), we only have to save the central part of each table (displayed in boldface above). The headlines and summation row and column are omitted. For France in the year 2000 the saved OEP table (without headlines and summation row and column) looks like this:

| | CON | PP | DOM | TRA | IND | OTH |
|---|---|---|---|---|---|---|
| BC | 0 | 0 | 0 | 0 | 0 | 0 |
| HC | 25 | 213 | 44 | 0 | 122 | 0 |
| DC | 0 | 0 | 41 | 0 | 206 | 0 |
| MD | 0 | 0 | 502 | 502 | 65 | 0 |
| HF | 226 | 34 | 45 | 0 | 108 | 0 |
| LF | 0 | 0 | 0 | 985 | 85 | 417 |
| GAS | 4 | 13 | 691 | 0 | 497 | 0 |
| OS | 0 | 9 | 3 | 0 | 26 | 0 |
| NUC | 0 | 3737 | 0 | 0 | 0 | 0 |
| HYD | 0 | 689 | 0 | 0 | 0 | 0 |
| ELE | 337 | -1808 | 813 | 34 | 463 | 0 |
| DH | 0 | 0 | 0 | 0 | 0 | 0 |

*TABLE 2: A 2000* fuels-over-sectors *OEP table for France stored in the file* oep0800.prn. *The boldface submatrix is to be replaced by simulation output from the SEEM model.*

The table is named oep0800.prt, being RAINS' OEP data for France in the year 2000. A comment on the file names is now called for. The sim2rain macro runs on a UNIX workstation. The UNIX operating system discriminates between lower and upper case letters in file names. Hence, note the convention of lower case UNIX file names. We will return to this issue when addressing the practical procedure of running macros and simulating models in section 3.2.

RAINS' internal database can be overruled by the user, who may supply other Energy Pathway files similar to the OEP exports, cf. *TABLE 2*. The SEEM model outputs country-year specific energy consumption data that may well replace parts or all of the data in the OEP tables. The sim2rain macro reads the tabular data in the files oep*ccyy*.prt, and replaces certain figures in the tables with data from a simulation of the SEEM model. The OEP figures to be replaced are denoted by boldface type in *TABLE 2*. Consult the RAINS manual for explanations of the fuels and sectors in the table. The modified OEP scenario tables are written to files in a certain format. The files can be input to simulations of the RAINS model.

The replacement of certain energy figures in an existing OEP file with corresponding figures originating from a simulation of the SEEM model can obviously be done element by element. But, the fuels and sectors of the SEEM model does not correspond one-to-one with those in the RAINS model. Two fuel aggregates in SEEM have to be disaggregated and two sectors in SEEM have to be aggregated. As an alternative to an element by element based calculation procedure, a more elegant way to do the required transformations and replacements is by matrix algebra. A procedure based on matrix algebra is implemented in the sim2rain macro. The following subsections describe the matrices and then the matrix operations involved in the operations. Then follows a section on what is required before the macro can be run. The last section outlines the structure and operations of the macro, before finally the TROLL code of the macro is listed.

### 3.1.2 RAINS' fuels-over-sectors input file: the SEP table

A subset of RAINS' OEP data can be replaced by data read from input files, which are ASCII text files in the shape of fixed format tables or matrices of fuel-sector specific energy consumption figures. We shall use the term SEEM Energy Pathway (SEP) files for such input files where the certain replaced figures come from a simulation of the SEEM model. *TABLE 2* above shows the fuels-over-sectors table of RAINS' OEP data for a specific country-year combination. The central part of the table, marked with boldface type, contains the data that are to be replaced by simulation output figures from SEEM. Not all OEP figures are to be replaced by SEEM figures. Since there is no one-to-one correspondence between sectors and fuels in the SEEM and RAINS models, the OEP table entries to be replaced are the ones where (i) corresponding SEEM figures exist or can be constructed and (ii) the SEEM fuels are organic. The latter criteria (ii) omits the clean fuels, i.e. the renewable energy sources hydro-electric and nuclear power, since their use do not lead to any emissions in the RAINS model. After replacment the table is saved in a (slightly different) fixed format ASCII text file named *sccyy*.prn (cf. *TABLE 3* below) where *s* denotes any single letter that identidies the SEEM simulation (e.g. i0800.prn). The

country and the actual year is part of the name of the SEP files, hence it is not recorded in the data. Following the naming convention required by the RAINS program, the properly formated SEP tables can be input to the RAINS program. Note the extension prn of an input file, compared to the extansion prt of an exported file).

A fuels-over-sectors input file to RAINS has a fixed format which looks like this:

```
............CON......PP.......DOM......TRA......IND......OTH......
.....................................................................
BC..........0........0........0........0........0........0........
HC..........25.......452......49.......0........138......0........
DC..........0........0........46.......0........233......0........
MD..........0........0........803......662......88.......0........
HF..........226......127......72.......0........146......0........
LF..........0........0........0........1300.....115......417......
GAS.........4........2........829......26.......568......0........
OS..........0........9........3........0........26.......0........
NUC.........0........3737.....0........0........0........0........
HYD.........0........689......0........0........0........0........
ELE.........337......-1808....813......34.......463......0........
DH..........0........0........0........0........0........0........
```

*TABLE 3: A 2000* fuels-over-sectors *SEP table for France, stored in the ASCII file* i0800.prn. *The original OEP figures have been replaced by (boldface) data originating from a simulation of the SEEM model. Note that the boldface SEEM figures are different from the OEP figures in TABLE 2.*

Again dots are replaced for blanks (space) to show the fixed-positions format. For the SEP table above to be valid format RAINS input the requirements are: 12 positions for the fuel name and 9 positions for the fuel consumption in each sector, all name codes and figures being left justified. The format of the SEP table above is slightly different from the format of the OEP table. The tables *sccyy*.prn differ from the tables oep*ccyy*.prn in that the columns are right justified and occupy more positions. The column and row names are unchanged.

The boldface submatrix of the OEP table is the part of the OEP data that is to be replaced by data originating from a SEEM simulation. Since the sectors and the fuels do not correspond one-to-one with the sector and fuel categories of the SEEM model, some of the SEEM output data have to be both disaggregated and aggregated to fit into the format of the (boldface) SEP submatrix, i.e. some SEEM fuels have to be disaggregated into RAINS fuels while two SEEM sectors have to be aggregated into a RAINS sector. Being able to do the required operations without refering explicitly to the individual variables/figures motivate the use of matrix algebra.

### 3.1.3 RAINS' Official Energy Pathway submatrix: the O matrix

The boldface submatrix in *TABLE 2* is to be replaced by data from a SEEM simulation. We let a 7 by 4 matrix **O** represent the submatrix of the OEP table:

$$\mathbf{O} = \begin{pmatrix} o_{11} & o_{12} & o_{13} & o_{14} \\ o_{21} & o_{22} & o_{23} & o_{24} \\ o_{31} & o_{32} & o_{33} & o_{34} \\ o_{41} & o_{42} & o_{43} & o_{44} \\ o_{51} & o_{52} & o_{53} & o_{54} \\ o_{61} & o_{62} & o_{63} & o_{64} \\ o_{71} & o_{72} & o_{73} & o_{74} \end{pmatrix}. \tag{3.1}$$

For France in the year 2000 the content or the original OEP numbers of the matrix (3.1) is

$$\mathbf{O} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 213 & 44 & 0 & 122 \\ 0 & 41 & 0 & 206 \\ 0 & 502 & 502 & 65 \\ 34 & 45 & 0 & 108 \\ 0 & 0 & 985 & 85 \\ 13 & 691 & 0 & 497 \end{pmatrix}.$$

### 3.1.4 The SEEM output matrix: the S matrix

RAINS' OEP submatrix **O** is to be replaced by a similar SEP matrix **R**, where the element values originate from a SEEM simulation. The relevant output from a simulation of the SEEM model and from running a summation macro named seemsum are read from two separate FORMDATA files. The files containing output from a SEEM simulation are named

<country code>/out<scenario code><country code>, (e.g. fr/outisfr),

and the single output file from the summation macro seemsum is

sum<scenario code>con, (e.g. sumiscon).

The SEEM data are in a different denomination than the RAINS data. The SEEM energy consumption figures are denoted in 1000 Mtoe, i.e. 1000 million tons oil equivalents, while the RAINS energy figures are denoted PJ, i.e. Peta Joule. Since 1 Mtoe = 42.3 PJ (cf. Rapporter 93/1, pp 63), to scale the SEEM figures to the RAINS scale we first multiply the relevant SEEM data with the scaling coefficient $c = 0.0423$. The figures read from the simulation files and the summation file can then be tabulated similar to the previous tables:

| | EL | HO | SE | MO | IN |
|---|---|---|---|---|---|
| COA | 452.3 | 94.7 | 0 | 0 | 371.2 |
| OIL | 126.8 | 102.6 | 771.9 | *1961.8* | 348.6 |
| NGS | 2.0 | 444.2 | 384.8 | 25.8 | 567.8 |

*TABLE 4: A tabular display of the relevant output from the SEEM model (normal type)and the* seemsum *macro (italic type) for France in 2000.*

Normal type denotes disaggregated figures read directly from the SEEM output file (i.e. fr/outfsfr), while the slanted numbers for OIL and NGS in MO sector are the only aggregated figures read from the summation macro output (i.e. sumfsc2). We let a 3 by 5 matrix **S** represent the content or numbers of the table

$$\mathbf{S} = \begin{pmatrix} s_{11} & s_{12} & s_{13} & s_{14} & s_{15} \\ s_{21} & s_{22} & s_{23} & s_{24} & s_{25} \\ s_{31} & s_{32} & s_{33} & s_{34} & s_{35} \end{pmatrix}.$$

For France in the year 2000 the scaled SEEM figures give the following content of the matrix **S**:

$$\mathbf{S} = \begin{pmatrix} 452.3 & 94.7 & 0 & 0 & 371.72 \\ 126.8 & 102.6 & 771.9 & 1961.8 & 348.6 \\ 2.0 & 44.2 & 384.8 & 25.8 & 567.8 \end{pmatrix}.$$

### 3.1.5 The replacement submatrix: the R matrix

A 7 by 4 matrix **R** which is to replace the OEP matrix **O**, has to be constructed from the SEEM matrix **S**. Hence, the 15 figures of the 3 by 5 matrix **S** is to replace the 28 figures of the 7 by 4 matrix **O**. This has to be done in a certain way, by splitting rows and joining columns of **S**. The fuel categories of the SEEM model have to be split into the fuel categories of the RAINS model, and two SEEM sectors have to be joined into one RAINS sector. Refering to *TABLE 2* and *TABLE 4*, we have the following correspondence between the rows of the SEEM matrix **S** and the rows of the OEP submatrix **O**:

COA = BC + HC + DC, OIL = MD + HF + LF and NGS = GAS,

while for the columns we have the following correspondence:

EL = PP, HO + SE = DOM, MO = TRA and IN = IND.

We use the ratio between RAINS' disaggregated fuel types to split SEEM's aggregegated fuels. The ratios are implicitly given in each country-year specific OEP table. The coal aggregate COA is split into the disaggregates BC, HC and DC according to their relative share of the total oil consumption:

$$BC_{SEEM} = COA * BC /(BC + HC + DC) \quad or \quad COA / 3,$$

$$HC_{SEEM} = COA * HC /(BC + HC + DC) \quad or \quad COA / 3,$$

$$DC_{SEEM} = COA * DC /(BC + HC + DC) \quad or \quad COA / 3,$$

where $BC_{SEEM} + HC_{SEEM} + DC_{SEEM} = COA$. The oil aggregate OIL is split the same way into

$$MD_{SEEM} = OIL * MD /(BC + HC + DC) \quad or \quad OIL / 3,$$

$$HF_{SEEM} = OIL * HF /(BC + HC + DC) \quad or \quad OIL / 3,$$

$$LF_{SEEM} = OIL * LF /(BC + HC + DC) \quad or \quad OIL / 3,$$

where $MD_{SEEM} + HF_{SEEM} + LF_{SEEM} = OIL$, accordingly. The *or* alternativ applies only if BC + HC + DC = 0. That is, *if* there are no OEP figures for the consumption of any kind of coal or oil in a sector, while nevertheless the SEEM model calculates some nonzero figures, the SEEM figures are distributed uniformly on the RAINS disaggregates. Being 3 disaggregated oils and coals, a third of each SEEM figure is put on the respective three disaggregates.

To do the row splitting and column joining as matrix operations without refering to any matrix elements, we identify three matrix sub-operations. First we expand the 3 by 5 matrix **S** to a 7 by 4 matrix **T** by tripling its first two rows and summing its second and third column. This is accomplished by premultiplying the **S** matrix by an expansion matrix **E** and postmultiplying it by a compression matrix **C**, so that

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{E}} \underbrace{\begin{pmatrix} s_{11} & s_{12} & s_{13} & s_{14} & s_{15} \\ s_{21} & s_{22} & s_{23} & s_{24} & s_{25} \\ s_{31} & s_{32} & s_{33} & s_{34} & s_{35} \end{pmatrix}}_{\mathbf{S}} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{C}}$$

gives

$$\mathbf{T} = \begin{pmatrix} s_{11} & s_{12}+s_{13} & s_{14} & s_{15} \\ s_{11} & s_{12}+s_{13} & s_{14} & s_{15} \\ s_{11} & s_{12}+s_{13} & s_{14} & s_{15} \\ s_{21} & s_{22}+s_{23} & s_{24} & s_{25} \\ s_{21} & s_{22}+s_{23} & s_{24} & s_{25} \\ s_{21} & s_{22}+s_{23} & s_{24} & s_{25} \\ s_{31} & s_{32}+s_{33} & s_{34} & s_{35} \end{pmatrix},$$

i.e. **T** = **ESC**. The splitting of the rows (so far) represents the disaggregation of fuels, while the summation of columns (so far) represents the aggregation of sectors. With the SEEM data for France in the year 2000 the result of these operations is

$$\mathbf{T} = \begin{pmatrix} 452.3 & 94.7 & 0 & 371.2 \\ 452.3 & 94.7 & 0 & 371.2 \\ 452.3 & 94.7 & 0 & 371.2 \\ 126.8 & 874.8 & 1961.8 & 348.6 \\ 126.8 & 874.8 & 1961.8 & 348.6 \\ 126.8 & 874.8 & 1961.8 & 348.6 \\ 2.0 & 829.0 & 25.8 & 567.8 \end{pmatrix}.$$

Next we construct a weighting matrix **W** with correct ratios by first premultiplying the **O** matrix with a block diagonal addition matrix **A** and then performing an *elementwise* division of **O** by the matrix product **AO**, so that

$$\begin{pmatrix} o_{11} & o_{12} & o_{13} & o_{14} \\ o_{21} & o_{22} & o_{23} & o_{24} \\ o_{31} & o_{32} & o_{33} & o_{34} \\ o_{41} & o_{42} & o_{43} & o_{44} \\ o_{51} & o_{52} & o_{53} & o_{54} \\ o_{61} & o_{62} & o_{63} & o_{64} \\ o_{71} & o_{72} & o_{73} & o_{74} \end{pmatrix} / \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} o_{11} & o_{12} & o_{13} & o_{14} \\ o_{21} & o_{22} & o_{23} & o_{24} \\ o_{31} & o_{32} & o_{33} & o_{34} \\ o_{41} & o_{42} & o_{43} & o_{44} \\ o_{51} & o_{52} & o_{53} & o_{54} \\ o_{61} & o_{62} & o_{63} & o_{64} \\ o_{71} & o_{72} & o_{73} & o_{74} \end{pmatrix}$$

$$\mathbf{O} \quad / \quad (\mathbf{A} \quad \mathbf{O})$$

gives

$$\mathbf{W} = \begin{pmatrix} \frac{o_{11}}{o_{11}+o_{21}+o_{31}} & \frac{o_{12}}{o_{12}+o_{22}+o_{32}} & \frac{o_{13}}{o_{13}+o_{23}+o_{33}} & \frac{o_{14}}{o_{14}+o_{24}+o_{34}} \\ \frac{o_{21}}{o_{11}+o_{21}+o_{31}} & \frac{o_{22}}{o_{12}+o_{22}+o_{32}} & \frac{o_{23}}{o_{13}+o_{23}+o_{33}} & \frac{o_{24}}{o_{14}+o_{24}+o_{34}} \\ \frac{o_{31}}{o_{11}+o_{21}+o_{31}} & \frac{o_{32}}{o_{12}+o_{22}+o_{32}} & \frac{o_{33}}{o_{13}+o_{23}+o_{33}} & \frac{o_{34}}{o_{14}+o_{24}+o_{34}} \\ \frac{o_{41}}{o_{41}+o_{51}+o_{61}} & \frac{o_{42}}{o_{42}+o_{52}+o_{62}} & \frac{o_{43}}{o_{43}+o_{53}+o_{63}} & \frac{o_{44}}{o_{44}+o_{54}+o_{64}} \\ \frac{o_{51}}{o_{41}+o_{51}+o_{61}} & \frac{o_{52}}{o_{42}+o_{52}+o_{62}} & \frac{o_{53}}{o_{43}+o_{53}+o_{63}} & \frac{o_{45}}{o_{44}+o_{54}+o_{64}} \\ \frac{o_{61}}{o_{41}+o_{51}+o_{61}} & \frac{o_{62}}{o_{42}+o_{52}+o_{62}} & \frac{o_{63}}{o_{43}+o_{53}+o_{63}} & \frac{o_{46}}{o_{44}+o_{54}+o_{64}} \\ 1 & 1 & 1 & 1 \end{pmatrix},$$

i.e. **W** = **O**/(**AO**), where / denotes *elementwise* division of **O** by the matrix product **OA**.

In (infrequent) cases of a zero valued denominator the three indeterminate fractions (with the same zero valued denominator) in row 1, 2 and 3 or row 4, 5 and 6 are replaced by 1/3. This yields a uniform distribution of the SEEM figure onto the RAINS categories. We have the same kind of problem with the last row also. Algebraically the OEP figures $o_{7x}$, $x = 1, 2, 3, 4$, in the fractions cancel out, but numerically a zero value renders the fraction indeterminate. Since the last row should be all 1s irrespective of the OEP values, every element in the last row is simply put equal to 1.

With OEP data for France in the year 2000 the weighting matrix above becomes

$$\mathbf{W} = \begin{pmatrix} 0 & 0 & 0.33 & 0 \\ 1 & 0.52 & 0.33 & 0.37 \\ 0 & 0.48 & 0.33 & 0.63 \\ 0 & 0.92 & 0.34 & 0.25 \\ 1 & 0.08 & 0 & 0.42 \\ 0 & 0 & 0.66 & 0.33 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Note that the elements in each column sum up to 3: One for the first three rows, one for the next three rows and one for the last row.

Now we have the matrix **T** containing the scaled SEEM data and the matrix **W** containing weights based on the relative shares of the OEP data. The wanted 7 by 4 matrix **R** to replace the submatrix **O** of OEP data can now be calculated by *elementwise* matrix multiplication **T**·**W**. Performing this last operation we get the matrix **R** displayed on the next page.

$$\mathbf{R} = \begin{pmatrix} \frac{s_{11}o_{11}}{o_{11}+o_{21}+o_{31}} & \frac{(s_{12}+s_{13})o_{12}}{o_{12}+o_{22}+o_{32}} & \frac{s_{14}o_{13}}{o_{13}+o_{23}+o_{33}} & \frac{s_{15}o_{14}}{o_{14}+o_{24}+o_{34}} \\ \frac{s_{11}o_{21}}{o_{11}+o_{21}+o_{31}} & \frac{(s_{12}+s_{13})o_{22}}{o_{12}+o_{22}+o_{32}} & \frac{s_{14}o_{23}}{o_{13}+o_{23}+o_{33}} & \frac{s_{15}o_{24}}{o_{14}+o_{24}+o_{34}} \\ \frac{s_{11}o_{31}}{o_{11}+o_{21}+o_{31}} & \frac{(s_{12}+s_{13})o_{32}}{o_{12}+o_{22}+o_{32}} & \frac{s_{14}o_{33}}{o_{13}+o_{23}+o_{33}} & \frac{s_{15}o_{34}}{o_{14}+o_{24}+o_{34}} \\ \frac{s_{21}o_{41}}{o_{41}+o_{51}+o_{61}} & \frac{(s_{22}+s_{23})o_{42}}{o_{42}+o_{52}+o_{62}} & \frac{s_{24}o_{43}}{o_{43}+o_{53}+o_{63}} & \frac{s_{25}o_{44}}{o_{44}+o_{54}+o_{64}} \\ \frac{s_{21}o_{51}}{o_{41}+o_{51}+o_{61}} & \frac{(s_{22}+s_{23})o_{52}}{o_{42}+o_{52}+o_{62}} & \frac{s_{24}o_{53}}{o_{43}+o_{53}+o_{63}} & \frac{s_{25}o_{45}}{o_{44}+o_{54}+o_{64}} \\ \frac{s_{21}o_{61}}{o_{41}+o_{51}+o_{61}} & \frac{(s_{22}+s_{23})o_{62}}{o_{42}+o_{52}+o_{62}} & \frac{s_{24}o_{63}}{o_{43}+o_{53}+o_{63}} & \frac{s_{25}o_{46}}{o_{44}+o_{54}+o_{64}} \\ s_{31} & (s_{32}+s_{33}) & s_{34} & s_{35} \end{pmatrix}.$$

By reading data from files into matrices **S** and **O**, and creating the operation matrices **E**, **C** and **A**, the wanted RAINS' EP (sub-)matrix **R** results from the concatenated *matrix* and *elementwise* operations

$$\mathbf{R} = (\mathbf{ESC})\cdot\mathbf{O}/(\mathbf{AO}). \tag{3.2}$$

Equation (3.2) shows the formula for construction of a submatrix with SEEM data to replace the (boldface) submatrix of the RAINS OEP file. The formula is implemented in the TROLL macro seem2rains. With the SEEM data for France in 2000 the final result of these operations is

$$\mathbf{R} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 452 & 49 & 0 & 138 \\ 0 & 46 & 0 & 233 \\ 0 & 803 & 662 & 88 \\ 127 & 72 & 0 & 146 \\ 0 & 0 & 1300 & 115 \\ 0 & 829 & 26 & 568 \end{pmatrix}.$$

When this matrix (**R**) replaces the boldface submatrix (**O**) of the OEP table shown in *TABLE 2*, we get the SEP table shown in *TABLE 3*. For France it have to be named i0800.prn, where i is any single letter denoting a user defined SEEM scenario. This SEP table is the format required for import into the RAINS program.

## 3.2 Beyond the year 2000

The database of RAINS 6.0 does not go beyond the year 2000. Nevertheless, we want to be able to use RAINS to calculate emissions from energy consumption figures simulated 10 or 20 years past that. The additional problem with this relative to what is discribed above is that there are no OEP-tables beyond year 2000 into which SEEM figures can be substituted. A way around this problem may be to use the latest RAINS tables, i.e. for the year 2000, and insert the SEEM figures into those. This is ok for the figures we replace, but what about the others? *TABLE 2* and *TABLE 3* show that the first (CON) and the last column (OTH), and the five last rows of the OEP-table are left unaltered. As far as calculating $SO_2$ and $NO_x$ emissions goes, the last five rows of the OEP tables are of no consequences. But the seven first entries of the two columns do cause emissions. Hence, they should not stay fixed at the OEP values of year 2000 in any future year and scenario.

A simple way to bypass this problem is to multiply each figure in the two columns with the averagegrowth fraction of the SEEM figures relative to the original OEP figures. For each entry in either colomn we let

$$col_{\text{future year}} = ROUND\left(\frac{(\text{PP + DOM + TRA + IND})_{\text{SEEM future year}}}{(\text{PP + DOM + TRA + IND})_{\text{OEP year 2000}}}\, col_{\text{OEP year 2000}}\right), \quad col = \text{CON, OTH,} \tag{3.3}$$

The footscripts denote simulated SEEM figures for the future year and OEP figures for year 2000. The rounding results in integer figures, which is a requirement by RAINS. This formula applies to each of the seven entries in both the CON column and the OTH column. In *TABLE 2* and *TABLE 3* the OEP figure of LF fuel in the CON sector, 18, is multiplied by the mean of SEEM's LF figures in the next four columns, i.e. the mean of the sectors PP, DOM, TRA and IND in *TABLE 3*, 116, and divided by the mean of the corresponding OEP figures in *TABLE 2*, 114, yielding 18 (due to rounding).

Equation (3.1) shows that the numerator of the fraction in equation (3.3) is a row sum of matrix **O**. Correspondingly, equation (3.2) shows that the denominator of the fraction in equation (3.3) is a row sum of matrix **R**. Using the matrix notation of those sections we may express the content of equation (3.3) in vector and matrix notation as

$$\mathbf{c}_f = \mathbf{c}_{2000} \cdot \mathbf{R}\,\mathbf{i}/\mathbf{O}\,\mathbf{i} \quad \text{and} \quad \mathbf{o}_f = \mathbf{o}_{2000} \cdot \mathbf{R}\,\mathbf{i}/\mathbf{O}\,\mathbf{i}, \tag{3.4}$$

where $\mathbf{i} = (1,1,1,1)'$ is a (column) summation vector so that $\mathbf{R}\,\mathbf{i}$ and $\mathbf{O}\,\mathbf{i}$ are (column) vectors of mean SEEM and OEP values of the PP, DOM, TRA and IND sectors, respectively. The (column) vectors $\mathbf{c}$ and $\mathbf{o}$ contain the seven first entries in the CON and OTH column, respectively, and the footscripts denote calculated future values and OEP values of year 2000. Note that in (3.3) we use *elementwise* multiplication and division. With $\mathbf{R}$ as a 7 by 4 marix $(r_{ij})$, $\mathbf{O}$ as a 7 by 4 marix $(o_{ij})$, $\mathbf{c}_{2000}$ as a 7 by 1 column vector $(c_j)$ and $\mathbf{o}_{2000}$ as 7 by 1 column vector $(o_j)$, equation (3.3) becomes

$$\mathbf{c}_f = \begin{pmatrix} c_1 \frac{r_{11}+r_{12}+r_{13}+r_{14}}{o_{11}+o_{12}+o_{13}+o_{14}} \\ c_2 \frac{r_{21}+r_{22}+r_{23}+r_{24}}{o_{21}+o_{22}+o_{23}+o_{24}} \\ c_3 \frac{r_{31}+r_{32}+r_{33}+r_{34}}{o_{31}+o_{32}+o_{33}+o_{34}} \\ c_4 \frac{r_{41}+r_{42}+r_{43}+r_{44}}{o_{41}+o_{42}+o_{43}+o_{44}} \\ c_5 \frac{r_{51}+r_{52}+r_{53}+r_{54}}{o_{51}+o_{52}+o_{53}+o_{54}} \\ c_6 \frac{r_{61}+r_{62}+r_{63}+r_{64}}{o_{61}+o_{62}+o_{63}+o_{64}} \\ c_7 \frac{r_{71}+r_{72}+r_{73}+r_{74}}{o_{71}+o_{72}+o_{73}+o_{74}} \end{pmatrix} \quad \text{and} \quad \mathbf{o}_f = \begin{pmatrix} o_1 \frac{r_{11}+r_{12}+r_{13}+r_{14}}{o_{11}+o_{12}+o_{13}+o_{14}} \\ o_2 \frac{r_{21}+r_{22}+r_{23}+r_{24}}{o_{21}+o_{22}+o_{23}+o_{24}} \\ o_3 \frac{r_{31}+r_{32}+r_{33}+r_{34}}{o_{31}+o_{32}+o_{33}+o_{34}} \\ o_4 \frac{r_{41}+r_{42}+r_{43}+r_{44}}{o_{41}+o_{42}+o_{43}+o_{44}} \\ o_5 \frac{r_{51}+r_{52}+r_{53}+r_{54}}{o_{51}+o_{52}+o_{53}+o_{54}} \\ o_6 \frac{r_{61}+r_{62}+r_{63}+r_{64}}{o_{61}+o_{62}+o_{63}+o_{64}} \\ o_7 \frac{r_{71}+r_{72}+r_{73}+r_{74}}{o_{71}+o_{72}+o_{73}+o_{74}} \end{pmatrix}. \tag{3.5}$$

which are just column vectors of expression (3.3).

With data for France: OEP consumption figures for the year 2000, seen in *TABLE 2*, and consumption figures simulated by SEEM for the year 2010, seen in *TABLE 4*, the formulas (3.3)–(3.5) give the following two column

| | CON | PP | DOM | TRA | IND | OTH |
|---|---|---|---|---|---|---|
| BC | **0** | *0* | *0* | *0* | *0* | **0** |
| HC | **25** | *547* | *66* | **0** | *158* | **0** |
| DC | **0** | *0* | *62* | **0** | *267* | **0** |
| MD | **0** | *0* | *900* | *818* | *114* | **0** |
| HF | **226** | *189* | *81* | **0** | *189* | **0** |
| LF | **0** | *0* | *0* | *1606* | *149* | **417** |
| GAS | **4** | *2* | *1080* | *31* | *682* | **0** |
| OS | **0** | 9 | 3 | **0** | 26 | **0** |
| NUC | **0** | 3737 | **0** | **0** | **0** | **0** |
| HYD | **0** | 689 | **0** | **0** | **0** | **0** |
| ELE | 337 | -1808 | 813 | 34 | 463 | **0** |
| DH | **0** | **0** | **0** | **0** | **0** | **0** |

*TABLE 4: A* not *fully updated 2010* fuels-over-sectors *SEP table for France. The italics submatrix has been replaced by simulation output from the SEEM model (year 2010), while the boldface denotes the figures to be multiplied by relative growth fractions. The rest of the entries are OEP figures for the year 2000 not used by RAINS and may therefor be ignored.*

vectors to replace the CON and the OTH column in the SEP table (displayed in *TABLE 4*):

$$\mathbf{c}_{2010} = \begin{pmatrix} 0\ \frac{0}{0} \\ 25\ \frac{771}{379} \\ 0\ \frac{329}{247} \\ 0\ \frac{1832}{1069} \\ 226\ \frac{459}{187} \\ 0\ \frac{1755}{1070} \\ 4\ \frac{1795}{1201} \end{pmatrix} = \begin{pmatrix} ? \\ 51 \\ 0 \\ 0 \\ 555 \\ 0 \\ 6 \end{pmatrix} \quad \text{and} \quad \mathbf{o}_{2010} = \begin{pmatrix} 0\ \frac{0}{0} \\ 0\ \frac{771}{379} \\ 0\ \frac{329}{247} \\ 0\ \frac{1832}{1069} \\ 0\ \frac{459}{187} \\ 417\ \frac{1755}{1070} \\ 0\ \frac{1795}{1201} \end{pmatrix} = \begin{pmatrix} ? \\ 0 \\ 0 \\ 0 \\ 0 \\ 684 \\ 0 \end{pmatrix}. \tag{3.6}$$

As seen from (3.6) and the first entries in the two vectors, a problem with the formulas (3.3)-(3.5) arises when a row of **R** and/or **O** contains only zeros. Then it is not possible to calculate a growth fraction, and the result will be either a zero or an undefined entry in the column vector $\mathbf{c}_f$ or $\mathbf{o}_f$. This represents the case when we cannot use the growth in the fuel consumption in other sectors (as the fuel is not used at all in those sectors) to estimate the growth in the two non-SEEM sectors. Looking at *TABLE 2* and *TABLE 4*, if one of the upper three coal rows, one of the next three destilate/fluids rows or the gas row has zero in each of the PP, DOM, TRA and IND columns, then we may base the growth fraction on the mean of the twelve entries in the upper three coal rows, the mean of the next twelve entries in the three destilate/fluids rows or the mean of all twentyeight entries in the seven rows, respectively. Hence, if a fuel is not used in the PP, DOM, TRA and IND sectors in either **R** or **O**, we take the average consumption of several (related) fuels in the same sectors of both matrices. If we let $c$ and $o$ each denote a single element in the column vectors **c** and **o**, **r** and **o** each denote a row in **R** and **O**, then we have the element-version of (3.4) as

$$c_f = c_{2000}\ \mathbf{r'i/o'i} \quad \text{and} \quad o_f = o_{2000}\ \mathbf{r'i/o'i}. \tag{3.7}$$

Now, if either **r'i** or **o'i** is zero, we want to replace the zero or undefined fraction with

$$c_f = c_{2000}\ \mathbf{j'R\,i/\,j'O\,i} \quad \text{and} \quad o_f = o_{2000}\ \mathbf{j'R\,i/\,j'O\,i}, \tag{3.8}$$

where $\mathbf{j}' = (1,1,1,0,0,0,0)$ if $c$ is one of the first three entries, $\mathbf{j}' = (0,0,0,1,1,1,0)$ if $c$ is one of the next three entries and $\mathbf{j}' = (1,1,1,1,1,1,1)$ if $c$ is the seventh entry, gas, in which cases the averages are taken over all elements in the first three, next three or all rows of the matrices, respectively.

To calculate $\mathbf{c}_f$ and $\mathbf{o}_f$ we use the formula

$$\mathbf{c}_f = \mathbf{c}_{2000}\cdot\mathbf{w} \quad \text{and} \quad \mathbf{o}_f = \mathbf{o}_{2000}\cdot\mathbf{w},$$

where **w** is a vector of growth fractions. An individual growth fraction in **w** is the mean in (3.7) or the extended mean fraction in (3.8). The two kinds are combined in an including/excluding sum **w**, which is

$$\mathbf{w} = (\mathbf{j} - \mathbf{s})\cdot\mathbf{R\,i/O\,i} + \mathbf{s}\cdot\mathbf{J'R\,i/J'O\,i}, \tag{3.9}$$

where $\mathbf{j} = (1,1,1,1,1,1,1)'$ and $\mathbf{s} = (s_j)$, $s_j \in \{0,1\}$, is a selection vector so that $(\mathbf{j} - \mathbf{s})$ selects the entries from the vector **R i/O i** that represent non-zero rows in either **R** or **O**, while **s** selects the entries from the vector **J'R i/J'O i** that represent the zero rows in either **R** or **O**. The 7 by 7 matrix **J'** contains the row vectors **j'** in (3.8), i.e. three times (1,1,1,0,0,0,0), then three times (0,0,0,1,1,1,0) and finally one time (1,1,1,1,1,1,1). Hence, an element in the column vector **w** is

$$w_j = (1-s_j)\frac{\sum_{i=1}^{4} r_{ji}}{\sum_{i=1}^{4} o_{ji}} + s_j \frac{\sum_{i=1}^{4}\sum_{h=a}^{b} r_{hi}}{\sum_{i=1}^{4}\sum_{h=a}^{b} o_{hi}}, \quad \text{where } (a,b) = \begin{cases} (1,3) & \text{if } j \in \{1,2,3\}, \\ (4,6) & \text{if } j \in \{4,5,6\}, \\ 7 & \text{if } j = 7. \end{cases}$$

Either $1-s_j = 1$ or $s_j = 1$, in which case the first or the second term is selected, respectively. For the example with France in the year 2010, the first entries in the $\mathbf{c}_{2010}$ and $\mathbf{o}_{2010}$ vectors in (3.6) are both replaced by

$$0 \quad \frac{0+771+329}{0+379+247}$$

(which would have been non-zero if $c_{2000}$ and $o_{2000}$ were non-zero). The growth vector (3.9) then looks like

$$\mathbf{w} = \begin{pmatrix}0\\1\\1\\1\\1\\1\\1\end{pmatrix} \cdot \begin{pmatrix}0\\ \frac{771}{379}\\ \frac{329}{247}\\ \frac{1832}{1069}\\ \frac{459}{187}\\ \frac{1755}{1070}\\ \frac{1795}{1201}\end{pmatrix} + \begin{pmatrix}1\\0\\0\\0\\0\\0\\0\end{pmatrix} \cdot \begin{pmatrix}\frac{0+771+329}{0+379+247}\\ \frac{0+771+329}{0+379+247}\\ \frac{0+771+329}{0+379+247}\\ \frac{1832+459+1755}{1069+187+1070}\\ \frac{1832+459+1755}{1069+187+1070}\\ \frac{1832+459+1755}{1069+187+1070}\\ \frac{0+771+329+1832+459+1755+1795}{0+379+247+1069+187+1070+1201}\end{pmatrix} = \begin{pmatrix}\frac{1100}{626}\\ \frac{771}{379}\\ \frac{329}{247}\\ \frac{1832}{1069}\\ \frac{459}{187}\\ \frac{1755}{1070}\\ \frac{1795}{1201}\end{pmatrix}.$$

The resulting new SEP-table (to be stored as an ASCII text file on disk for further input to RAINS) is displayed in *TABLE 5*.

| | CON | PP | DOM | TRA | IND | OTH |
|---|---|---|---|---|---|---|
| BC | 0 | 0 | 0 | 0 | 0 | 0 |
| HC | 51 | 547 | 66 | 0 | 158 | 0 |
| DC | 0 | 0 | 62 | 0 | 267 | 0 |
| MD | 0 | 0 | 900 | 818 | 114 | 0 |
| HF | 555 | 189 | 81 | 0 | 189 | 0 |
| LF | 0 | 0 | 0 | 1606 | 149 | 684 |
| GAS | 6 | 2 | 1080 | 31 | 682 | 0 |
| OS | 0 | 9 | 3 | 0 | 26 | 0 |
| NUC | 0 | 3737 | 0 | 0 | 0 | 0 |
| HYD | 0 | 689 | 0 | 0 | 0 | 0 |
| ELE | 337 | -1808 | 813 | 34 | 463 | 0 |
| DH | 0 | 0 | 0 | 0 | 0 | 0 |

*TABLE 5: A 2010* fuels-over-sectors (final) *SEP table for France stored in the file. The boldface submatrix has been replaced by simulation output from the SEEM model (year 2010) and values extrapolated from year 2000 OEP figures by using growth rates calculated from the SEEM and OEP figures.*

By the method in this section we estimate what the fuel consumption in two non-SEEM sectors would be *if* those sectors experienced the same growth as certain other SEEM sectors. This gives a possible future path, based solely on increased demand for energy. It does not include any effects from possible or likly technological changes or developements. Hence, if there is no consumption of a certain fuel in a sector in the year 2000, then there will be no consumption of that fuel in the sector at any later stage, as seen in (3.6). If a non-zero figure were to replace a zero, it would imply that substitution and/or technological changes introduce a previously unused fuel. Exclusion of the such effects is only a minor drawback as SEEM is basically a demand based model with a small substitution elasticities and no endogenized technology.

## 3.3 The code of the TROLL macro sim2rain

The actual matrix operations constitute a minor part of the total macro. Most of the code deals with user interface and reading and writing files. The macro has the following outline:

1. User interface: Get calculation (SEEM) year and match with RAINS year, and get calculation countries;
2. Initialization and creation of the fixed calculation vctors and matrices;

For c = first_country to last_country

2. Reading data from textfiles into arrays: Read SEEM and SEEMSUM output from FORMDATA files and replace (empty) **S** matrix elements with selected SEEM output figures from these files;
3. Read RAIN's output OEP table in the form of an ASCII text file into a TROLL matrix (previously exported from RAINS), converting from text string format to number format;
4. Extract submatrix **O** from TROLL's OEP matrix
5. Matrix operations:
   5.1. Calculate by matrix operations **R** = c (**ESC**)·**O**/(**AO**);
   5.2. If SEEM year > 2000 calculate replacement columns for CON and OTH, i.e. the first and last number column of the SEP table;
6. Convert **R** and OEP matrix numbers to text strings, and write original OEP figures and calculated replacement figures to the output file in the form of an ASCII text SEP table;

The rest of this chapter contains a listing of the TROLL macro code with comments:

```
/* Portable TROLL macro: SIM2RAIN                                                    */

/* Programmed by: Dag Kolsrud, SRM, Research Department, Statistics Norway           */
/* All updates: Version 1 in November 1995 by Dag Kolsrud                            */

/* Purpose: To transform energy consumption figures simulated by the SEEM model so that they can replace a subset of the */
/* Official Energy Pathway (OEP) data in the RAINS model.                            */

/* Short description: This macro does the following:                                 */
/* - reads tables of RAINS OEP data from ASCII text files,                           */
/* - reads SEEM simulation data from TROLL files,                                    */
/* - replaces a submatrix of figures in the OEP tables with transformed SEEM data    */
/* - writes the modified tables to ASCII text files in a certain format required by RAINS. */

/* Details: The macro is a direct implementation of the procedure described in the documentation note. Only a minor part of the*/
/* code handles the numerical operations. The majority of commands deals with the user interface and the input-output of data*/
/* in text format. The TROLL matrices have the same names as they do in the note to enable direct correspondence.  */
/*   After a user dialog the macro loops through the required countries, reading OEP tables exported from RAINS, exchanging */
/* certain figures with SEEM data and storing the modified tables as fixed format ASCII text files.  */

/* Macro programming: The transformation of SEEM data to fit the RAINS categories involves aggregating and disaggragating*/
/* SEEM data according to ratios among the RAINS OEP data. Such operations are preferably performed by matrix algebra (on*/
/* a "macro level") which eliminates the need to adress individual variables/figures (on a "micro level"). TROLL has powerfull */
/* matrix functions which makes it possible to perform the required matrix algebra. Reading and writing text files and  */
/* transforming between strings and numbers is made somewhat cumbersome by the fact that only a subset of the functions in*/
/* mainframe TROLL has been ported to Unix and MS DOSTROLL (by the end of 1994). A separate macro STRING2ARRAY */
/* helps in converting a string of words into an array of those words (letters or numbers).  */

ADDFUN MAIN, STRING2ARRAY;
PROCEDURE MAIN()
BEGIN; /* macro */

  /* ---------------------------------------- USER INTERFACE ---------------------------------------- */

  /* The SEEM simulation scenario on which to base the calculations have to be input. The scenario code goes into the name*/
  /* of the FORMDATA files containing the individual countries energy consumption figures. Note lower case UNIX notation. */
  /* Note that the first letter in the scenario name code is used in naming output files. Hence, each scenario name code  */
  /* have to start with a letter that is different from the first letter of all other scenarios  */
```

```
GET FROM TERMINAL scenario "\n Give the name code of the scenario which data to use (e.g. IS, FS or NS): ";
letter_code = LOWER(SUBSTR(scenario, 1, 1));                    /* First letter to use in output filename */

/* Matrices can be constructed for the years 1990, 1995, 2000-2020. RAINS year 1990 is given SEEM output for 1991,  */
/* RAINS year 1995 is given SEEM output for 1995 and RAINS year 2000 is given SEEM output for 2000. But, since      */
/* version 6.0 of RAINS does not include any years exceeding 2000,  RAINS year 2000 is given any SEEM output from  */
/* year 2000 to 2020. Since SEEM figures replace most of the RAINS figures that cause emissions of CO2 and NOx,    */
/* this "extrapolation" of RAINS' data basis may for certain limited purposes be an acceptable extension (which does not */
/* take into account future technological progress, though).                                                       */

GET FROM TERMINAL rains_year "\n Give the last two digits of the year (1990, 1995 or 2000-2020) of the scenario: ";
year = CONVERT(rains_year);                                     /* Convert from text to number */
IF (rains_year == "90") THEN seem_year = "1991A";               /* The "A" denotes annual data */
ELSE IF (rains_year == "95") THEN seem_year = "1995A";
ELSE IF (year >= 0 AND year <= 20) THEN BEGIN;
    seem_year = "20" || rains_year || "A";
    rains_year = "00";                                          /* Trick RAINS to believe it handles the year 2000 */
    IF (year > 0) THEN
        GET FROM TERMINAL letter_code "\n Give single letter (not F, I or N) to code SEEM Energy Pathway table: ";
END;
ELSE BEGIN; PRINT("\n Wrong option, ending macro! "); EXIT(); END;          /* Error message and exit */
/* User decides whether to sum up for a single country, a few countries or for all 13 countries:                */

GET FROM TERMINAL choice "\n Decide what countries to sum up. Answer the letter S \n for a singel country, the letter
M for many (or a few) countries \n or the letter A for all 13 countries. \n Any other answer terminates the macro: ";
choice = UPPER(choice);                                         /* User's choice = S, M or A */
single_country = choice == "S";
many_countries = choice == "M";
all_countries = choice == "A";
country_codes = COMBINE("AU", "BE", "BR", "CH", "DK", "FR", "GB", "IT", "NL", "NO", "SF", "SP", "SW");
country_numbers = COMBINE("02", "03", "09", "23", "06", "08", "25", "14", "16", "17", "07", "21", "22");

IF (single_country) THEN BEGIN;                                 /* Construct a single entry array */
    GET FROM TERMINAL answer "\n Give ONE SINGLE country code from the following: AU, BE, \n BR, CH, DK, FR,
GB, IT, NL, NO, SF, SP or SW: ";
    all_country = COMBINE(UPPER(answer));                       /* Single country code */
    pos = (INDEX(JOINSTR(country_codes), all_country)+1)/2;     /* Calculating array position of codes and numbers */
    nr_country = country_numbers[pos];                          /* Country number from code */
    final_country = 1;
END; /* IF single_country */

ELSE IF (many_countries) THEN BEGIN;                            /* Construct an array with user's country codes as entries */
    all_country = COMBINE();                                    /* Empty array of zero length */
    final_country = 0;                                          /* Initializing */
    GET FROM TERMINAL answer "\n List countries by the following codes: \n AU, BE, BR, CH, DK, FR, GB, IT, NL, NO,
SF, SP or SW \n separated by blanks and terminated bya blank and \n a semicolon (e.g. like NO SF SW ;): ";
    WHILE (answer <> ";") BEGIN;                                /* Not finnished country input */
        all_country = COMBINE(all_country, UPPER(answer));      /* Adding next country to the array */
        final_country = final_country + 1;
        GET FROM TERMINAL answer;
    END; /* WHILE answer */

    IF (final_country == 0) THEN BEGIN; PRINT("\n Wrong option, ending macro! "); EXIT(); END;/* Error message, exit */
    pos = (INDEX(JOINSTR(country_codes), all_country)+1)/2;
    nr_country = country_numbers[pos];                          /*Country numbers from codes */
END; /* IF many_countries */
```

```
ELSE IF (all_countries) THEN BEGIN;
    all_country = country_codes;
    nr_country = country_numbers;
    final_country = 13;
END; /* IF all_countries */

ELSE BEGIN; PRINT("\n Wrong option, ending macro! "); EXIT(); END;                    /* Error message and exit */

/* ---------------------------------------- INITIALIZING ---------------------------------------- */

/* Putting sector codes into vectors :                                                              */

ST_sector     = COMBINE("EL", "HO", "SE", "IN");
final_sector  = 4;
ST_fuel       = COMBINE("COA", "OIL", "NGS");
final_fuel    = 3;

/*Creating the row duplicating expansion matrix E, the column compression matrix C and  the element adding matrix A.  */
/* The temporary matrix T is created inside the country loop. First the calculation coefficient is set:              */

coeff = 0.0423;
>> DOCORE e1 = COMBINE(1,0,0);
>> DOCORE e2 = COMBINE(0,1,0);
>> DOCORE e3 = COMBINE(0,0,1);
>> DOCORE e_rows = COMBINE(e1, e1, e1, e2, e2, e2, e3);
>> DOCORE E_matrix = TRANSP(RESHAPE(e_rows, 3, 7));                  /* 7 rows and 3 columns of 0s and 1s */
>> DOCORE c1 = COMBINE(1,0,0,0,0);
>> DOCORE c2 = COMBINE(0,1,1,0,0);
>> DOCORE c3 = COMBINE(0,0,0,1,0);
>> DOCORE c4 = COMBINE(0,0,0,0,1);
>> DOCORE c_rows = COMBINE(c1, c2, c3, c4);
>> DOCORE C_matrix = RESHAPE(c_rows, 5, 4);                          /* 5 rows and 4 columns of 0s and 1s */
>> DOCORE a1 = COMBINE(1,1,1,0,0,0,0);
>> DOCORE a2 = COMBINE(0,0,0,1,1,1,0);
>> DOCORE a3 = COMBINE(0,0,0,0,0,0,1);
>> DOCORE a_rows = COMBINE(a1, a1, a1, a2, a2, a2, a3);
>> DOCORE A_matrix = RESHAPE(a_rows, 7, 7);          /* 7 rows and 7 columns of 0s and 1s (block-diagonal) */
>> DOCORE i_vector = COMBINE(1,1,1,1);                          /* Vectors and matrix used  for calculation... */
>> DOCORE identity_vector = COMBINE(1,1,1,1,1,1,1);                        /* ...of CON and OTH sectors... */
>> DOCORE j_matrix = SETREP(A_matrix, 1, COMBINE(7,7,7,7,7,7,7), COL(A_matrix));        /* ...IF year > 2000 */

/* The TROLL DATA file containing the aggregated simulation results for the specified scenario are read from        */
/* a single FORMDATA file for all countries:                                                                        */

inSUMfile = LOWER("sum" || scenario || "con.txt");                  /* Concatenating name of FORMDATA file to read */
IF (NOT HOSTEXIST(inSUMfile)) THEN BEGIN; PRINT("Does not find the formdata file: ", inSUMfile); EXIT(); END;
>> ACCESS inSUM TYPE FORMDATA ID &inSUMfile MODE R;
>> SEARCH FIRST DATA inSUM;                          /* Ready to read the energy aggregates OILCONMO&country */

/* ---------------------------------- READING DATA FROM TEXT FILES INTO ARRAYS ---------------------------------- */

FOR (i = 1; i <= final_country; i = i + 1) BEGIN;                                  /* Looping through the countries */
    country = all_country[i];                                                      /* Getting country name from vector */
    number = nr_country[i];                                                        /* Getting country number from vector */
    PRINT(" Treating country: ", country);                                 /* Informing user which country is being handled */

    /* The TROLL DATA files containing the simulation results for the specified scenario are read from a single    */
    /* FORMDATA file for each country. All disaggregated energy consumption figures are read from this file:       */
```

```
inCONfile = LOWER(country || "/" || "out" || scenario || country || ".txt");          /* Name of FORMDATA file to read */
IF (NOT HOSTEXIST(inCONfile)) THEN BEGIN; PRINT("Does not find the formdata file: ", inCONfile); EXIT(); END;
>> ACCESS inCON TYPE FORMDATA ID &inCONfile MODE R;
>> SEARCH FIRST DATA inCON;                                                          /* Ready to read energy use in current country */

/* Recreates the SEEM output matrix for each country (to contain a subset of the SEEM output): For each country      */
/* and its 4 STationary sectors, each with 3 STationary fuels are read from the file containing disaggregated output  */
/* from the SEEM modelaggregates for the STationary sources:                                                           */

>> DOCORE S_matrix = CRMAT(3, 5, 0);                                                 /* 3 rows and 5 columns of zeros */
FOR (j = 1; j <= final_sector; j = j + 1) BEGIN;                                     /* Looping through 4 sectors EL, HO, SE and IN */
    sector = ST_sector[j];                                                           /* Getting country name from vector */
    IF (sector == "IN") THEN S_column = 5; ELSE S_column = j;                        /* S_column skips the 4th MO column */
    FOR (k = 1; k <= final_fuel; k = k + 1) BEGIN;                                   /* Looping through 3 fuels  COA, OIL, NGS */
        fuel = ST_fuel[k];                                                           /* Getting country name from vector */
        variable = fuel || "CON" || sector || country;                               /* Concatenating filename */

        /* Reading time series from disaggregated SEEM-output file for country, picking the value for the right year:  */

        >> DOCORE disaggregate = VALUES(GETDATA("&variable", -1), &(seem_year));
        >> DOCORE S_matrix = SETREP(S_matrix, disaggregate, &k, &(S_column));
    END; /* fuel loop */
END; /* sector loop */
/* Column 1, 2, 3 and 5 are now filled with correct values from STationary sectors. The 4th column with values from   */
/* the MObile sectors remains, but only row 2 and 3 is non-zero:                                                      */

variable = "GACONTP" || country;                                                     /* Name of local variable */
>> DOCORE disaggregate = VALUES(GETDATA("&variable", -1), &(seem_year));
>> DOCORE S_matrix = SETREP(S_matrix, disaggregate, 3, 4);                           /* Replacing S_matrix[3, 4] */
variable = "OILCONMO" || country;                                                    /* Name of local variable */
>> DOCORE aggregate = VALUES(GETDATA("&variable", -1), &(seem_year));
>> DOCORE S_matrix = SETREP(S_matrix, aggregate, 2, 4);                              /* Replacing S_matrix[2, 4] */
>> DOCORE S_matrix = &coeff * S_matrix;                  /* Scaling the S_matrix from SEEM to RAINS denomination */

/* Reads the RAINS Official Energy Pathway matrix for the country-year combination into an array of strings so that  */
/* each string contains one row from the OEP matrix                                                                  */

oep_file = "oep /oep" || number || rains_year || ".prt";          /* Concatenating name of official pathway file to read */
oep_string_array = XREAD(oep_file);                               /* Each row of the matrix is read into a string in an array */
>> DOCORE oep_matrix = CRMAT(1,6);                                /* Create an empty array, to which matrix rows will be added */
>> DOCORE first_column = COMBINE();                               /* Create an empty array, to which row names will be added */

FOR (j = 3; j <= 14; j = j + 1) BEGIN;          /* For 3rd to 14th row in the OEP matrix pass the row of the OEP matrix... */
    row_string = oep_string_array[j];           /* ...as an input string to a macro that converts each string element to... */
    &STRING2ARRAY; >> &row_string ;             /* ...a number element in an array which is a global TROLL variable */
    >> DOCORE oep_matrix = ADDROW(oep_matrix, &j-2, matrix_row);  /* Array of 6 numbers added as matrix row */
    >> DOCORE first_column = COMBINE(first_column, first_element);  /* Row name added as array element */
END; /* FOR each row in the OEP matrix */

>> DOCORE oep_matrix = DELROW(oep_matrix, 1);
>> DOCORE O_matrix = SUBMAT(oep_matrix, SEQ(7), COMBINE(2, 3, 4, 5));

/* ---------------------------------------- MATRIX OPERATIONS ----------------------------------------*/

/* Performs the matrix multiplication T = ESC and the matrix and elementwise operations W = O/(AO). This results in */
/* division by 0, producing NA values where we actually want 0.333, hence the replacement. To avoid inevitable      */
/* warnings, the mechanism is turned off and then back on.                                                           */

ON WARNING NOMSG;
```

```
>> DOCORE T_matrix = MATMULT(MATMULT(E_matrix, S_matrix), C_matrix);
>> DOCORE W_matrix = O_matrix / MATMULT(A_matrix, O_matrix);
>> DOCORE W_matrix = SETREP(W_matrix, 1, COMBINE(7,7,7,7), COL(W_matrix));      /* Change row 7 to all 1s */
>> DOCORE W_matrix = NAFILL(W_matrix, 0.33333);                  /* NAs from division by 0 is replaced with 1/3 */
ON WARNING ;

/* Performs the matrix calculations R = T*W = ESC * O / AO:                                                    */

>> DOCORE R_matrix = ROUND(T_matrix * W_matrix, 0);    /* Rounding off to nearest integer (required by RAINS) */

/* Calculating the CON and OTH columns if SEEM year greater than year 2000                                    */

IF (year > 0 AND year <= 21) THEN BEGIN;
    ON WARNING NOMSG;
    >> DOCORE co_column = SUBMAT(oep_matrix, SEQ(7), 1);                    /* CON column of 7 entries */
    >> DOCORE oo_column = SUBMAT(oep_matrix, SEQ(7), 6);                    /* OTH column of 7 entries */
    >> DOCORE ri_vector = MATMULT(R_matrix, i_vector);                  /* Mean of four sectors' SEEM figures */
    >> DOCORE oi_vector = MATMULT(O_matrix, i_vector);                   /* Mean of four sectors' OEP figures */
    >> DOCORE growth_vector = ri_vector / oi_vector;                  /* Single fuel SEEM growth relative to OEP */
    >> DOCORE oiri_vector = oi_vector / ri_vector;                              /* "Inverse" of growth_vector */
    >> DOCORE select_vector = NAFILL(growth_vector - oiri_vector, 1);           /* 1 for a zero row, otherwise 0 */
    >> DOCORE growth_vector = NAFILL(growth_vector, 0);                             /* Replace Nas with 0 */

    /* Testing for and performing a complex replacement (using a 0/1 weighted sum selecting either one or another */
    /* growth rate) or just doing a simple replacement                                                          */

    >> DOCORE troll_boolean = CUMSUM(select_vector * (co_column + oo_column))[7] > 0;              /* Test if... */
    replacement = GETDATA("troll_boolean", 2, FALSE);                          /* ...OEP column ≠ 0. If so then... */
    IF (replacement) THEN BEGIN;                                        /* ...there is at least one zero-row in R or O*/
        >> DOCORE jRiOi_vector = MATMULT(j_matrix, ri_vector) / MATMULT(j_matrix, oi_vector);
        >> DOCORE growth_vector = growth_vector * (identity_vector - select_vector) + jRiOi_vector * select_vector;
    END;
    >> DOCORE cs_column = ROUND(co_column * growth_vector, 0);          /* Rounding  required by RAINS */
    >> DOCORE os_column = ROUND(oo_column * growth_vector, 0);     /* cs_ and os_column are 7 by 1 vectors */
    ON WARNING ;
END;

/* Each calculated matrix is stored in a single ASCII TeXT file for each country. It written row by row with the XWRITE */
/* and XAPPEND commands. Hence there are no access or search. If the file (name) exists, the file is deleted:          */

outtextfile = "sep/" || letter_code || number || rains_year || ".prn";          /* Concatenating name of file to write */
IF (HOSTEXIST(outtextfile)) THEN BEGIN;     /* The (name of the) FORMDATA file to be written does already exist */
    PRINT("Deleting existing formdata file: ", outtextfile);                          /* Informing user of macro */
    >> HOST "rm -f &outtextfile";                  /* Unix command to delete any existing file by the same name */
END; /* file exists */

/* Each line in the output file has the following format:                                                      */
/* - the name begis in position 1 and ends in position 12. It is padded with blanks at the end to fill  12 positions, */
/* - the numbers occupy positions 13-21, 22-30, 31-39, 40-48, 49-47 and 48-56. They are left justified and padded */
/* with blanks at the end (like the names). The two first lines are common to all output files:                */

>> DO XWRITE("       CON     PP      DOM     TRA     IND     OTH     ", "&outtextfile");    /* 1. line of ouput file */
>> DO XAPPEND("                                                      ", "&outtextfile");    /* 2. line of ouput file */

/* The output matrix is composed of original (unaltered) numbers from the input Official Energy Pathway matrix and */
/* the new calculated sub matrix R. If the SEEM year is one of 2001,..., 2020, then figures are also taken from the */
/* cs and co vectors. Each row (the 12 following the two headlines above) in the final matrix is converted to a string */
/* which is formatted according to the requirements for input matrices to the RAINS program:                   */
```

```
    FOR (j = 1; j <= 12; j = j + 1) BEGIN;                          /* For each row 3 to 14 of the output file */
        >> DOCORE element = first_column[&j];                       /* Unaltered OEP matrix element: row name */
        >> DOCORE blanks = 12 - LENGTH(element);                    /* Number of blanks to append after letters */
        >> DOCORE pad = REPSTR(" ", blanks);                        /* A string with the right number of blanks */
        >> DOCORE string_row = element || pad;                      /* String contains padded row name */
        FOR (k = 2; k <= 7; k = k + 1) BEGIN;                       /* For each column 1 to 7 of the output file */
            IF (k > 2  AND k < 7 AND j < 8) THEN
                >> DOCORE element = CONVERT(R_matrix[&j, &k-2]);    /* Calculated sub-matrix element as string */
            ELSE IF (year > 0 AND year < 21 AND j < 8) THEN BEGIN;
                IF (k == 2) THEN >> DOCORE element = CONVERT(cs_column[&j]);        /* CON element as string */
                ELSE IF (k == 7) THEN >> DOCORE element = CONVERT(os_column[&j]);   /* OTH element as string */
            END; ELSE >> DOCORE element = CONVERT(oep_matrix[&j, &k-1]); /* Unchanged OEP element as string */

            /* Each (matrix element) number is now CONVERTed to a string.To satisfy the required RAINS format each  */
            /* string is padded with blanks (space) to the right to fill up the 9 positions RAINS require:          */

            >> DOCORE blanks = 9 - LENGTH(element);                 /* Number of blanks to append after digits */
            >> DOCORE pad = REPSTR(" ", blanks);                    /* A string with the right number of blanks */
            >> DOCORE string_row = string_row || element || pad;    /* Add element and padding blanks to row string */
        END; /* FOR each column in a row */
        >> DO XAPPEND(string_row, "&outtextfile");                  /* New line to ouput file */
    END; /* FOR each row */

    /* Delete search path and access to current country input file that is now done with, so that a new input file can be  */
    /* accessed and searched by the alias file name inCON. Then delete TROLL's local variables, search paths and          */
    /* access to output database:   */

    >> DELSEARCH inCON;
    >> DELACCESS inCON;
END; /* country loop */

>> DELCORE ALL;
>> DELSEARCH ALL;
>> DELACCESS ALL;
END; /* macro */


/* Portable TROLL macro: STRING2ARRAY                                                                                  */

/* Programmed by: Dag Kolsrud, SRM, Research Department, Statistics Norway                                             */
/* All updates: Version 1 in April 1995 by Dag Kolsrud                                                                 */
/* Purpose: Convert each number in a (text) string to an element in a (formated) array.                                */
/* Short description: A string of 7 elements has the format "letter-code1  number2  ...  number7. The letter-code1 element  */
/* is discarded, and the 6 numbers are read as textual  macro input from the calling macro into number elements in an array.  */
/* Details: By expanding the local macro variable the textual elements are passed to TROLL as number constants and    */
/* combined into an array.*/
/* Macro programming: By December 1994 Portable TROLL has no function to read a textstring into an array of words     */
/* and numbers. The function SPLITSTR slits a string into an array of letters and digits.                              */

PROCEDURE STRING2ARRAY()
BEGIN; /* macro */
    GET UP 1 element;                                               /* Discard first input element from calling macro */
    >> DOCORE first_element = "&element";                           /* First input element is a string */
    >> DOCORE matrix_row = COMBINE();                               /* Create empty array */
    FOR (k = 2; k <= 7; k = k + 1) BEGIN;                           /* For 2nd to 7th input element */
        GET UP 1 element;                                           /* Read it (a number) as a textstring */
        >> DOCORE matrix_row = COMBINE(matrix_row, &element);       /* Convert it to an element in a TROLL array */
    END;
END;
```

# 4 The DOS batch file rimport

Version 6.0 of the RAINS pollution model includes a database of energy consumption figures for most European countries. As a consequence of the energy consumption figures in the database scenarios, RAINS estimates emissions and atmospheric transports of $SO_2$, $NO_x$ and $NH_3$, and local environmental impacts from depositions of the three pollutants. This chapter explains how external data may be used in place of RAINS' internal data on energy consumption. More specifically, this chapter gives a detailed instruction on how to import back the previously exported OEP fuels-over-sectors tables that in the meantime have been modified with SEEM data (as explained in chapter 3). It also tells how to proceed with the imported data to calculate emmisions and environmental impacts. In doing so, this chapter documents how an interactive session can be simulated by a batch file issuing commands under the DOS operating system. This makes it possible to automatize running repetitive and large "interactive" sessions that would be prohibitively time consuming in a manually interactive modus operandi. This chapter does not tell how to extract information from RAINS. RAINS is an interactive, menu-driven program that needs no further explanations here.

## 4.1 Data import

We start where chapter 3 ended, with RAINS' OEP fuels-over-sectors tables modified by energy consumption figures simulated by the SEEM model, cf. *TABLE 3* in chapter 3. The modified tables are saved in separate files named *sccyy*.prn, which are all stored in the directory d:\RAINS6\data. To import the data tables into the RAINS program for further calculations, we have to do the following:

1. Start the RAINS program by issuing the command: rains (when placed in the directory RAINS6, or with a defined search path to this directory),
2. Choose the menu options 1, 1, 2 and 5 to arrive at the RAINS task: "Import of the country data to create a new scenario",
3. Choose the Official Energy Pathway (OEP) as the base scenario. For the countries not included in the SEEM model, the energy consumption data will be fetched from this scenario,
4. Provide the single scenario letter that is the first letter in the name of each import file (then wait...),
5. Name the new scenario to be based on the imported energy consumption figures (use the scenario code from SEEM, e.g. Integrated Europe Scenario).

After the above procedure the user defined scenario will be listed next to RAINS' own (internal) scenarios. Any RAINS analysis may be based on this scenario rather than one of its own built-in scenarios. By issuing the menu options 1,1,1 and 3, choosing the user created scenario, e.g. named Integrated Europe Scenario (following the above procedure 1-5), the country France and the year 2000, the fuels-over-sectors table for this scenario is displayed as *TABLE 3*. We notice that certain figures (the correct ones) have changed compared to the original OEP data in *TABLE 1*.

ENERGY_USE_PER_SECTOR_AND_FUEL (PJ)

| | CON | PP | DOM | TRA | IND | OTH | SUM |
|---|---|---|---|---|---|---|---|
| BC | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| HC | 25 | 452 | 49 | 0 | 138 | 0 | 664 |
| DC | 0 | 0 | 46 | 0 | 233 | 0 | 279 |
| MD | 0 | 0 | 803 | 662 | 88 | 0 | 1553 |
| HF | 226 | 127 | 72 | 0 | 146 | 0 | 571 |
| LF | 0 | 0 | 0 | 1300 | 115 | 417 | 1832 |
| GAS | 4 | 2 | 829 | 26 | 568 | 0 | 1429 |
| OS | 0 | 9 | 3 | 0 | 26 | 0 | 38 |
| NUC | 0 | 3737 | 0 | 0 | 0 | 0 | 3737 |
| HYD | 0 | 689 | 0 | 0 | 0 | 0 | 689 |
| ELE | 337 | -1808 | 813 | 34 | 463 | 0 | -161 |
| DH | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SUM | 592 | 3208 | 2615 | 2022 | 1777 | 417 | 10631 |

*TABLE 4: The 2000* fuels-over-sectors Integrated Europe Scenario *table for France, as it looks when printed from the RAINS screen to the file* enedb.prt. *Compare the SEEM figures in this table to the OEP figures in the table shown in TABLE 1 in chapter 3.*

## 4.2 Calculating emissions

We are now in a position were we can let the RAINS model calculate emissions of $SO_2$, $NO_x$ and $NH_3$, and local environmental impacts from depositions of the three pollutants, as a consequence of the energy consumption figures from SEEM simulations and the OEP scenario. We continue where step 5 in section 4.1 left off:

6. Return to the previous menu, choose option 1 to let RAINS calculate emissions from the new and user defined energy scenario.
7. Exit the RAINS program.

After (re-)starting RAINS, issuing the complete sequence of menu options: 1, 2, 1, 1 and 3, choosing the user created scenario, e.g. named Integrated Europe Scenario (following the above procedure 1-7), the country France and the year 2000; the fuels-over-sectors table of $SO_2$ emissions for the SEEM simulated scenario is displayed as *TABLE 5*:

1

R A I N S  Version 6.0 - 1992 (c) IIASA PAGE 1
ROUTE:11300000 printed on 8/11/95

Fractionated Europe Scenario France 2000

SO2_EMISSIONS_BY_SECTORS_AND_FUELS

(in kt SO2)

| | CON | PP | DOM | TRA | IND | TOTAL |
|---|---|---|---|---|---|---|
| BC | .0 | .0 | .0 | .0 | .0 | .0 |
| HC | 15.5 | 279.7 | 30.3 | .0 | 63.6 | 389.1 |
| DC | .0 | .0 | 29.8 | .0 | 86.2 | 116.0 |
| MD | .0 | .0 | 188.9 | 155.8 | 20.7 | 365.4 |
| HF | 381.2 | 214.2 | 48.6 | .0 | 203.3 | 847.3 |
| PROC | | | | | | 104.0 |
| SUM | 396.7 | 494.0 | 297.7 | 155.8 | 373.8 | 1821.9 |

*TABLE 5: The 2000 $SO_2$ emissions from the* fuels-over-sectors Integrated Europe Scenario *table for France as it looks when the screen is saved to the file* enm.prt.

In addition, by issuing the complete sequence of menu options: 1, 2, 2, 1 and 3, choosing the user created scenario, e.g. named Integrated Europe Scenario (following the above procedure 1-7), the country France and the year 2000; we get the fuels-over-sectors table of $NO_x$ emissions for the SEEM simulated scenario displayed as *TABLE 6*:

NOx_EMISSIONS_BY_SECTORS_AND_FUELS

[kt NO2]

| | CON | PP | DOM | TRA | IND | TOTAL |
|---|---|---|---|---|---|---|
| BC | .0 | .0 | .0 | .0 | .0 | .0 |
| HC | 5.8 | 104.5 | 3.9 | .0 | 31.7 | 145.9 |
| DC | .0 | .0 | 3.2 | .0 | 32.6 | 35.8 |
| MD | .0 | .0 | 56.2 | 805.3 | 6.2 | 867.6 |
| HF | 38.4 | 22.1 | 23.0 | .0 | 24.8 | 108.4 |
| LF | .0 | .0 | .0 | 975.0 | 8.1 | 983.0 |
| GAS | .0 | .2 | 49.7 | .0 | 39.8 | 89.7 |
| OS | .0 | .0 | .2 | .0 | 3.4 | 3.6 |
| PROC | | | | | | 91.0 |
| SUM | 44.2 | 126.7 | 136.3 | 1780.3 | 146.5 | 2325.0 |

*TABLE 6: The 2000 $NO_x$ emissions from the* fuels-over-sectors Integrated Europe Scenario *table for France as it looks when the screen is saved to the file* enm.prt.

## 4.3 The code of the DOS batch file rimport

The RAINS commands 1-7 can be issued interactively by the user when he or she runs the RAINS program. The interactive user interface is fine for the odd calculation or for non-repetetive tasks. On the other hand, interactive operation soon becomes repetitive, tireing and slow when

- repeating certain operations over and over (e.g. displaying and fileing tables and plots),
- doing the same post calculations following each simulation of the SEEM model (à la an assembly line),
- iterating between the SEEM model and the RAINS model in search for an optimum (a future possibility?)

and doing other routine calculations with the RAINS model. Unfortunately, the RAINS program cannot be run in batch mode, as a batch job. One way to avoid the required interaction is to have a program simulate the user hitting the keyboard. Luckily, there is a freeware utility program called Stuffit or Stuffkey (different version of the same program) that does exactly this, it automates the keyboard. Given code for the key-strokes required by the task we want the RAINS program to perform, Stuffit writes the keystrokes into the keyboard buffer. The keystrokes are fetched from the keyboard buffer by the RAINS program when needed. From RAINS' point of view, the keystrokes are in the buffer as if they had been typed in by the user.

The rimport program automates a rather non-repetitive task, and is written mainly as an illustrative example of how an interactive session with RAINS can be made automatic. If one is going to work a lot with a SEEM-RAINS link automating certain parts of the RAINS interactive sessions may be worthwhile.

We collect all the commands required to (i) start RAINS, (ii) run through the wanted operations in RAINS and (iii) end the RAINS session, into a DOS batch file (with extension .bat). We let the Stuffit utility do all the typing, also of invoking the RAINS program. After an interactive session with the RAINS program to become familiar with the required keystrokes, the following code should be understandable. For more info on the Stuffit or Stuffkey program consult the files Stuffit.doc or Stuffkey.doc, which are part of the compressed package Stuff32.lhz (unpack with the lha program) and Stuffkey41.zip (unpack with pkunzip), respectively. These freeware programs are available from the pc-net.

The rimport.bat file, which is located in the \rains6 directory, looks like this:

```
@echo off

REM  MS DOS batch file: RIMPORT.BAT, located in the directory D:\RAINS6

REM  Programmed by: Dag Kolsrud, SRM, Statistics Norway (based on a prototype by Hugo Birkelund and Martin Moe)
REM  Updates: Version 1 in October 1995 by D.K.

REM  Purpose: This file runs the Stuffit utility program with specific commands. Stuffit is a utility that writes keystrokes
REM  into the keyboard buffer. Thus, it can simulate a user running a program in interactive mode. Here Stuffit runs the
REM  RAINS program and feeds it a sequence of keystrokes that "simulates" the user's choices in interactive sessions.
REM  These choices directs RAINS to import data for a new scenario and calculate the resulting emissions.

REM  Description: The user has to provide a single letter denoting the scenario which files to import, and the full name of
REM  the scenario (not to exceed 8 words) when invoking this batch file. An example of correct syntax is the command:
REM  "RAIMPORT i Integrated Europe Scenario" (without the double quote marks). The RAINS program imports user
REM  provided data from files named <single letter><country code><year>.PRN, like for instance the file i0800.prn, into a
REM  new scenario. The code below consists almost exclusively of keystroke "commands" to Stuffit, and they are explained.

REM  ---------- THE RAINS SESSION ----------

REM  Stuffit starts the RAINS program. The first carriage return ends RAINS' start-up command, while the second gets past
REM  RAINS' start-up screen. Stuffit chooses options 1, 1, 2 and 5, then the Official Energy Pathway data as a basis by
REM  typing a blank (i.e. space-bar) and a carriage return:

Stuffit +0 "rains" {CR} {CR} "1125" { } {CR}

REM  The parameter %1 contains the scenario letter provided by the user, while the parameters %2, %3, ... each contains
REM  a single word of the full scenario name (or are left empty). The "1" {CR} make RAINS calculate emissions from the
REM  new scenario. The final keystrokes exits the RAINS program:

Stuffit +0 "%1" +12
Stuffit +0 "%1 %2 %3 %4" {CR} + {ESC} "1" {CR} +15 {ESC} "qyqy"
```

# 5. Organization and execution of the programs

RAINS is a model running under the DOS operating system on a pc. SEEM is a model in the portable TROLL system, running under both the DOS operating system on a pc and the Unix operating system on a workstation. The sim2rain macro is written in the TROLL programming language. Since the macro reads data tables in the ASCII text format output by RAINS and modifies the tables to serve as new input to RAINS, it is natural to run the sim2rain macro on the pc. As long as we are interested in the SEEM-RAINS link, it is more practical to stick to the pc and avoid splitting the tasks between a Unix and a DOS computer. This implies that the SEEM model is simulated on the pc, and that the results are summed up by the seemsum macro, also on the pc. Accordingly, nothing needs to be executed on a workstation.

If, on the other hand, the SEEM model is simulated on a workstation, it is possible to do everything except running RAINS on the workstation. First, the seemsum macro is run. Next, the sim2rain macro is run, reading RAINS output, the OEP tables, which have been tranfered from the PC to the UNIX workstation in advance. The resulting files, the RAINS input, have to be transfered from the workstation to the PC, where RAINS can be run with the SEP tables as input.

There are two possible problems with the Unix and DOS environment. First, there may be minor differences between the implementation of Portable TROLL in the two environments. The code of the Portable TROLL macro issues host commands, that may work slightly differently between the two. On a stage of the development of the seemsum macro, it ran well on the workstation, but not on the pc. The problem was due to keeping a time series in TROLL's memory and storing it in as disk file, both having the same name. The problem was easily solved, but it illustrates the the kind of problems that may appear. Another possible problem is the naming of disk files. Under DOS there is an eight character limit (hence sim2rain instead of seem2rains), and all names are converted to upper case. In Unix there is no such limit, and lower and upper case letters are different. In the seemsum and sim2rain macros all file names contain eight or less characters, and are in lower case letters.

*FIGURE 1* shows how the models, programs and files relate to each other.

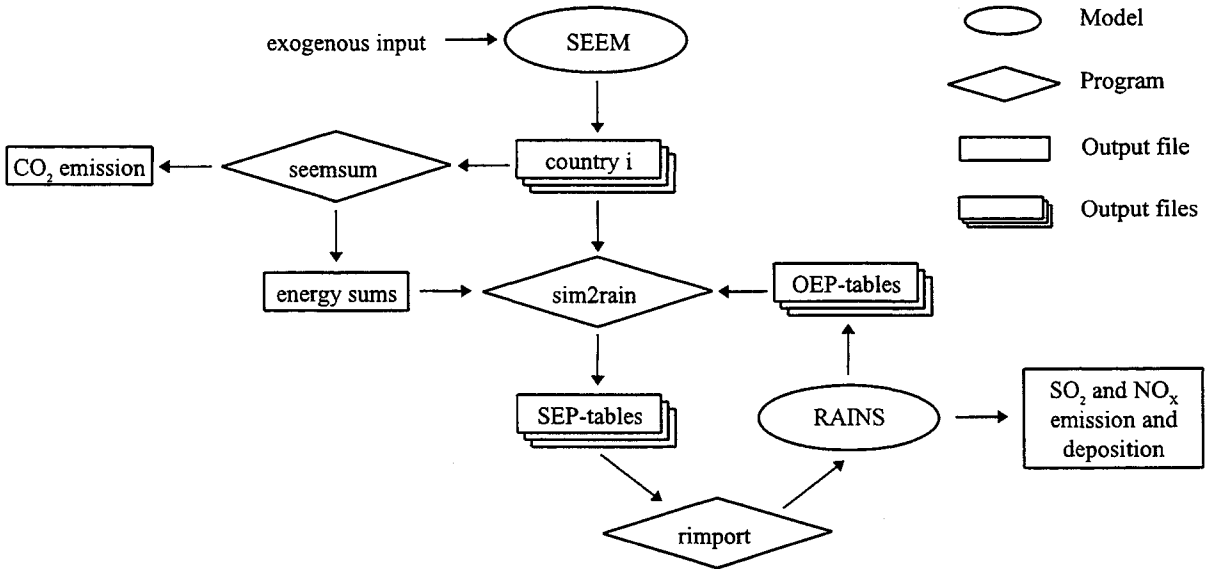


*FIGURE 1: Schematic view of the programs, macros and files constituting the SEEM-RAINS link.*

The following describes how to get from the exogenous input to the SEEM model, to the resulting emission output from the RAINS model:

1. Go to the pc directory: d:\seem,
2. Start TROLL by issuing the command troll (assuming that there is a path to the TROLL directory)
3. Inside TROLL issue the command &seemsim, which starts running a TROLL macro that simulates the SEEM model to get the SEEM output files <country code>/out<scenario code><country code>.txt, (e.g. fr/outfsfr.txt). Note that each output file holding the disaggregated results for a single country is automatically placed in its country's sub-directory,

4. In TROLL issue the command &seemsum, which starts running a TROLL macro that sum up the disaggregated results from the above SEEM simulation. The aggregated consumption figures are output to the file sum<scenario code>con.txt, (e.g. sumiscon.txt). Exit TROLL,

Steps 5-9 need only be executed once:

5. Change directory to d:\rains6,
6. Start RAINS by issuing the command rains,
7. Follow the correct path trough the menu options by issuing 1, 1, 1 and 3. Choose the OEP scenario, and supply the wanted country and year,
8. Export the necessary OEP data from RAINS (several country-year combinations may be given). The exported screen-tables are automatically written to a file named enedb.prt in the rains6 directory, (cf. page 2-7 in the RAINS manual). This file overwrites any old version. After final export exit from RAINS, rename the file to enedb*yy*.prt and store the (relevant) exported PRT files from RAINS together in a sub-directory under d:\seem, e.g. d:\seem\rainsout,
9. Split the enedb*yy*.prt file into individual files oep*ccyy*.prt (like oep0800.prt), one for each country-year combination. Store the files in a sub-directory , e.g. d:\seem\oep,

10. Start TROLL and issue the command &sim2rain, which starts running the TROLL macro that transforms SEEM output to RAINS input. The resulting files named *sccyy*.prn are written to the sub-directory d:\seem\sep (sep = SEEM Energy Pathway). Exit TROLL,
11. Goto the directory d:\rains6,
12. Copy all the individual (and releveant) SEP files in the directory \seem\oep to the directory \rains6\data.
13. Run the command file rimport.bat (in the d:\rains6 directory) by issuing the command rimport,
14. Start RAINS, and run an interactive session where emissions from the imported SEEM-scenarios are calculated and saved to disk files.

The programs seemsum, sim2rain and rimport all promt the user for names on input and output files, unless these options are provided in a correct sequence after the call to the programs. The layout of the programs and files on disk may be organized differently from that which is described above. In such cases it will be necessary to change some directories and file names in the TROLL macros to reflect the changes.

# Issued in the series Documents

94/1 *H. Vennemo (1994):* Welfare and the Environment. Implications of a Recent Tax Reform in Norway.

94/2 *K.H. Alfsen (1994):* Natural Resource Accounting and Analysis in Norway.

94/3 *O. Bjerkholt (1994):* Ragnar Frisch 1895-1995.

95/1 *A.R. Swensen (1995):* Simple Examples on Smoothing Macroeconomic Time Series.

95/2 *E.Gjelsvik, T. Johnsen, H.T. Mysen and A. Valdimarsson (1995):* Energy Demand in Iceland

95/3 *C. Zhao, O. Bjerkholt, T. Halvorsen and Y. Zhu (1995):* The Flow of Funds Accounts in China

95/4 *Nordic Indicator Group (1995):* Nordic Environmental Indicators. Draft document. English version with main points from comments received.

95/5 *H.A. Gravningsmyhr (1995):* Analysing Effects of Removing Survivors' Pensions, Using the Microsimulation Model LOTTE.

95/6 *P. Boug (1995):* User's Guide. The SEEM-model Version 2.0.

95/7 *E. Bowitz, N.Ø. Mæhle, V.S. Sasmitawidjaja and S.B. Widoyono (1995):* MEMLI – An Environmental model for Indonesia. Technical Documentation of data programs and procedures

95/8 *K. H. Alfsen, T. Bye, S. Glomsrød and H. Wiig (1995):* Integrated Assessment of Soil Degradation and Economic Growth in Ghana

95/9 *O. Bjerkholt (1995):* Ragnar Frisch and the Foundation of the Econometric Society and Econometrica

95/10 *P.J. Bjerve (1995):* The Influence of Ragnar Frisch on Macroeconomic Planning and Policy in Norway.

96/1 *D. Kolsrud (1996):* Documentation of Computer Programs that Extend the SEEM Model and Provide a Link to the RAINS Model

Statistics Norway
Research Department
P.O.B. 8131 Dep.
N-0033 Oslo

Tel.: + 47 - 22 86 45 00
Fax: + 47 - 22 11 12 38

ISSN 0805-9411

**Statistics Norway**
**Research Department**