PB87-132791

EPA/600/3-86/057
December 1986

GETS, A SIMULATION MODEL FOR DYNAMIC BIOACCUMULATION OF NONPOLAR ORGANICS BY GILL EXCHANGE: A User's Guide

by

Luis A. Suárez, M. Craig Barber, and Ray R. Lassiter

Biology Branch
Environmental Research Laboratory
Athens, Georgia 36013

ENVIRONMENTAL RESEARCH LABORATORY
OFFICE OF RESEARCH AND DEVELOPMENT
U.S. ENVIRONMENTAL PROTECTION AGENCY
ATHENS, GEORGIA 30613

TECHNICAL REPORT DATA
(Please read Instructions on the reverse before completing)

1. REPORT NO.	2.	3. RECIPIENT'S ACCESSION NO.
EPA/600/3-86/057		PB87 132791/AS

4. TITLE AND SUBTITLE	5. REPORT DATE
GETS, A SIMULATION MODEL FOR DYNAMIC BIOACCUMULATION OF NONPOLAR ORGANICS BY GILL EXCHANGE: A User's Guide	December 1986
	6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT NO.
Luis A. Suarez, M. Craig Barber, and Ray R. Lassiter	
9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT NO.
Environmental Research Laboratory U.S. Environmental Protection Agency Athens GA 30613	ACUL1A
	11. CONTRACT/GRANT NO.
12. SPONSORING AGENCY NAME AND ADDRESS	13. TYPE OF REPORT AND PERIOD COVERED
Environmental Research Laboratory - Athens, GA Office of Research and Development U.S. Environmental Protection Agency Athens, GA 30613	14. SPONSORING AGENCY CODE EPA/600/01

15. SUPPLEMENTARY NOTES

16. ABSTRACT

A Fortran program that estimates the absorption and depuration of a chemical across fish gills is described. The program is based on a set of diffusion and forced convection differential equations. Gill morphometric parameters are computed by the program via its own internal database. This database spans approximately 20 species. The program requires that the user input 12 relatively easily obtainable parameters.

17. KEY WORDS AND DOCUMENT ANALYSIS

a. DESCRIPTORS	b. IDENTIFIERS/OPEN ENDED TERMS	c. COSATI Field/Group

18. DISTRIBUTION STATEMENT	19. SECURITY CLASS *(This Report)*	21. NO. OF PAGES
RELEASE TO PUBLIC	UNCLASSIFIED	54
	20. SECURITY CLASS *(This page)* UNCLASSIFIED	22. PRICE

EPA Form 2220-1 (9-73)

DISCLAIMER

The information in this document has been funded wholly or in part by the United States Environmental Protection Agency. It has been subject to the Agency's peer and administrative review, and it has been approved for publication as an EPA document. Mention of trade names or commercial products does not constitute endorsement or recommendation for use by the U.S. Environmental Protection Agency.

FOREWORD

As environmental controls become more expensive and penalties for judgment errors become more severe, environmental management requires more precise assessment tools based on greater knowledge of relevant phenomena. As part of this Laboratory's research on the occurrence, movement, transformation, impact, and control of environmental contaminants, the Biology Branch conducts research to predict the rate, extent, and products of biological processes that control pollutant fate in soil and water and develops methods for forecasting ecosystem level effects suitable for exposure and risk assessment.

The Athens Environmental Research Laboratory, along with Office of Research and Development laboratories in Corvallis, OR, Duluth, MN, and Gulf Breeze, FL, is developing a system to assess ecological risks from exposure to environmental toxicants. This system will provide the capability to assess risk associated with different uses of chemicals resulting from various options for regulating pesticides and toxic chemicals to protect organisms in their natural environment. This report describes a component of that system, the Gill Exchange of Toxic Substances model.

Rosemarie C. Russo
Director
Environmental Research Laboratory
Athens, GA

ABSTRACT

A FORTRAN program that estimates the absorption and depuration of a chemical across fish gills is described. The program is based on the set of diffusion and forced convection differential equations. Gill morphometric parameters are computed by the program via its own internal database. This database spans approximately 20 species. The program requires that the user input 12 relatively easily obtainable parameters.

This report covers a period from October 1985 to September 1986, and work was completed as of September 1986.

INTRODUCTION

When fish are exposed to dissolved organic chemicals, such substances are accumulated within the fish by diffusive transport across the fish's gills. During acute exposures, chemical exchange across the gill is the fish's prevailing route of exposure. During chronic exposure in the environment, exposure via contaminated food can become increasingly important or even greatly exceed direct gill uptake. Nevertheless gill exchange still reciprocally controls the fish's body concentrations by determining the fish's excretory rates of the chemical.

Ultimate levels of organic toxicants in aquatic organisms can be explained, in large measure, as thermodynamic partitioning between toxicant in the aqueous environment and hydrophobic components of the organisms (primarily lipid). Reports of exceptions to this simple rule are numerous, however, particularly for chemicals having high partition coefficients. The frequency with which these exceptions have been noted has often led to questions of the reliability and utility of the relationship between bioconcentration factors (BCF) and thermodynamic partitioning. Based on our review of reported results for both laboratory and field investigations of BCF, uptake, and depuration, we have concluded that such results are both expected and explainable on an essentially thermodynamic (but not equilibrium) basis. Furthermore, we have developed a thermodynamically-based kinetic model called GETS, which predicts whole-body burdens and concentrations of organic chemicals in fish.

GETS (Gill Exchange of Toxic Substances) is a FORTRAN simulation model that predicts a fish's whole body concentration (i.e., ppm = μg chemical (g live weight fish)$^{-1}$) of a nonmetabolized, organic chemical which is exchanged across a fish's gill by thermodynamic concentration gradients. These concentration dynamics are calculated algebraically after simulating a fish's total body burden of the chemical, Bf = μg chemical fish^{-1}, and its live weight, W = g live weight fish^{-1}. The temporal dynamics of these two quantities are generated by the system of coupled differential equations

$$\frac{dBf}{dt} = S*kw*(Cw - (PL*KL)^{-1}Cf) \qquad (1)$$

$$\frac{dW}{dt} = gamma * W \qquad (2)$$

where S is the fish's total gill area (i.e., cm^2), kw is the chemical mass conductance through the interlamellar gill water (i.e., cm/day), Cw is the chemical's environmental concentration (i.e., ppm = mg l^{-1}), PL is the fish's lipid content as a fraction of its total live weight, KL is a lipid to water partition coefficient (i.e., (mole chemical/g lipid)/(mole chemical/g water)), Cf = Bf/W is the fish's whole body concentration of the chemical, and gamma is the fish's specific growth rate (i.e., g/g/day). A full description of the theoretical basis and development of these equations is presented by Barber et al. (1986).

GETS was developed and intended for use on a VAX 11/785 computer and is coded in VAX FORTRAN V4 3-145. A complete documented listing of GETS is presented in Appendix A.

USER INPUT

To use GETS, a user must specify twelve relatively straight-forward input parameters. These are:

1. the scientific name of the fish to be analyzed (e.g., Salmo gairdneri)
2. the fish's family (e. g. Salmonidae)
3. the fish life form (i.e., freshwater or marine)
4. the chemical's molecular weight
5. the chemical's log K_{OW} (e.g. logP)
6. the fish's initial live weight (g)
7. the fish's lipid content as a proportion of its total body weight (g lipid/g live weight)
8. the fish's specific growth rate (i.e., g/g/day)
9. the fish's initial whole body concentration of the chemical (i.e., μg (g live weight fish)$^{-1}$)
10. the chemical's environmental concentration as ppm = mg l^{-1}
11. a kinetic adjustment factor that is discussed below (unitless)
12. the length of the desired simulation in days

All of these values must be specified by the user and supplied to GETS via a user-defined data file, e.g., GETS.DAT. The name of this data file is specified to GETS interactively. Within this user-defined data file, each of the above input values is defined on a separate card image or record with a specific leading character code which is used to check that the user input sequence was correct. The user's input must be specified in the order listed above and is read by GETS in FORTRAN-free format. See Table 1 for an example input file that could be used to analyze the bioaccumulation of a tetrachlorobiphenyl by rainbow trout.

Table 1. Example of user input to GETS.

(Note: Quotation marks (i.e., single quotes or apostrophes) are significant.)

Record Number	Required Code	Data Value
1	'spplab'	'Salmo gairdneri'
2	'famlab'	'Salmonidae'
3	'liflab'	'freshwater'
4	'molwt'	292
5	'logP'	5.8
6	'wt'	200
7	'PL'	0.08
8	'gamma'	0.0015
9	'Cfish'	0.0
10	'Cwater'	0.
11	'adjust'	0.1
12	'TEND'	1880

The fish's species name, family, and life form are used to assign the gill morphometric parameters that GETS's subroutine, GILRAT, uses to estimate the fish's net exchange rate (S*kw). To make these assignments, the data file MORPHO.DAT is read by the subroutine RDMORPH. This data file (Appendix B), which is supplied with GETS, contains the coefficients and exponents of the allometric functions arranged by species,

S = total gill area, $cm^2 = s1\ W^{s2}$ and

RHO = # lamellae (mm gill filament)$^{-1} = \rho 1\ W\rho^2$

where W is the fish's live weight (g). As MORPHO.DAT is read, five geometric means are calculated by GETS using subroutine LOGSUM for each of the values, s1, s2, $\rho 1$, and $\rho 2$. For example, GETS calculates a geometric mean for s1, first using all the data reported in MORPHO.DAT. Concurrently, GETS also calculates geometric means for s1, using only data records which have the same life form, family, genus, and species of the fish designated by the user. GETS then attempts to assign s1, s2, $\rho 1$, and $\rho 2$ using first the species geometric means. If, however, the species is not represented in MORPHO.DAT, GETS then tries to assign the geometric means that might have been calculated for the same genus as the desired species. In like fashion, if the genus is not found in MORPHO.DAT, geometric means for the fish's family are assigned. If these assignments are not possible, the geometric means for the same life form (i.e., freshwater vs. marine) as the desired fish are used.

GETS is parameterized for a particular chemical of concern by specifying the chemical's molecular weight and log K_{ow}. The chemical's molecular weight is used to estimate its aqueous diffusivity which is needed to estimate the conductance, kw. The chemical's log K_{ow} is used in the calculation of the fish's excretion rate, $k_2 = S*kw*(PL*KL)^{-1}$.

The physical characteristics of the fish that are required as input are the fish's live weight (W), its lipid fraction (PL) and its specific growth rate (gamma). The specific growth rate, gamma, specified as input should be nonnegative. Although negative growth is exhibited by organisms under stress in natural ecosystems, it is not known whether fish that are losing weight significantly alter their gill morphometry. Therefore, although negative growth rates, per se, can be input to GETS, the resulting simulations, which depend on gill morphometry, may not be meaningful.

Because the user can specify any initial whole body concentration, Cf, of the chemical in the fish as well as any environmental concentration, Cw, the user can analyze either absolute uptake (i.e., Cf = 0 and Cw $\neq$ 0), pure depuration (i.e., Cf $\neq$ 0 and Cw = 0) or any scenario between these two extremes.

The user-supplied adjustment factor, adjust, is used to calibrate the mass conductance of the chemical that is estimated by GILRAT. In general, the value specified for this parameter should be in the range of 0.1 > adjust > 0.05 since the estimated conductance, kw, is generally between ten and twenty times higher than laboratory studies would indicate (see Barber et al. 1986).

GETS OUTPUT

Three files or logical units receive output from GETS. These are userfile.OUT, userfile.PLX, and the logical unit for the terminal itself. The terminal output is identical to the output written to userfile.OUT. This output includes a summary of all the user's input variables as well as the fish's live weight (W), total body burden (Bf), and total body concentration (Cf). If the user specifies TEND $\leq$ 500.0, W, Cf, and Bf are output daily, otherwise these model results are output only every (TEND/500) day.

Appendix C presents a sample user session of GETS which includes both input file construction via the text editor EDT and model execution and output.

The file, userfile.PLX, is an output generated by subroutine PLX010 and is a CALCOMP™ plot file. Figure 1 shows this file for the simulation presented in Appendix C.

SUMMARY OF GETS SUBROUTINES

Program MAIN:

Besides directing model input, output, and subroutine calls, MAIN also performs an Euler integration of equations (1) and (2). During the development of GETS, fifth order Runge-Kutta integration of these equations were also performed. However, since there were no significant differences between Euler and Runge-Kutta integrations, the simple Euler method was coded herein.

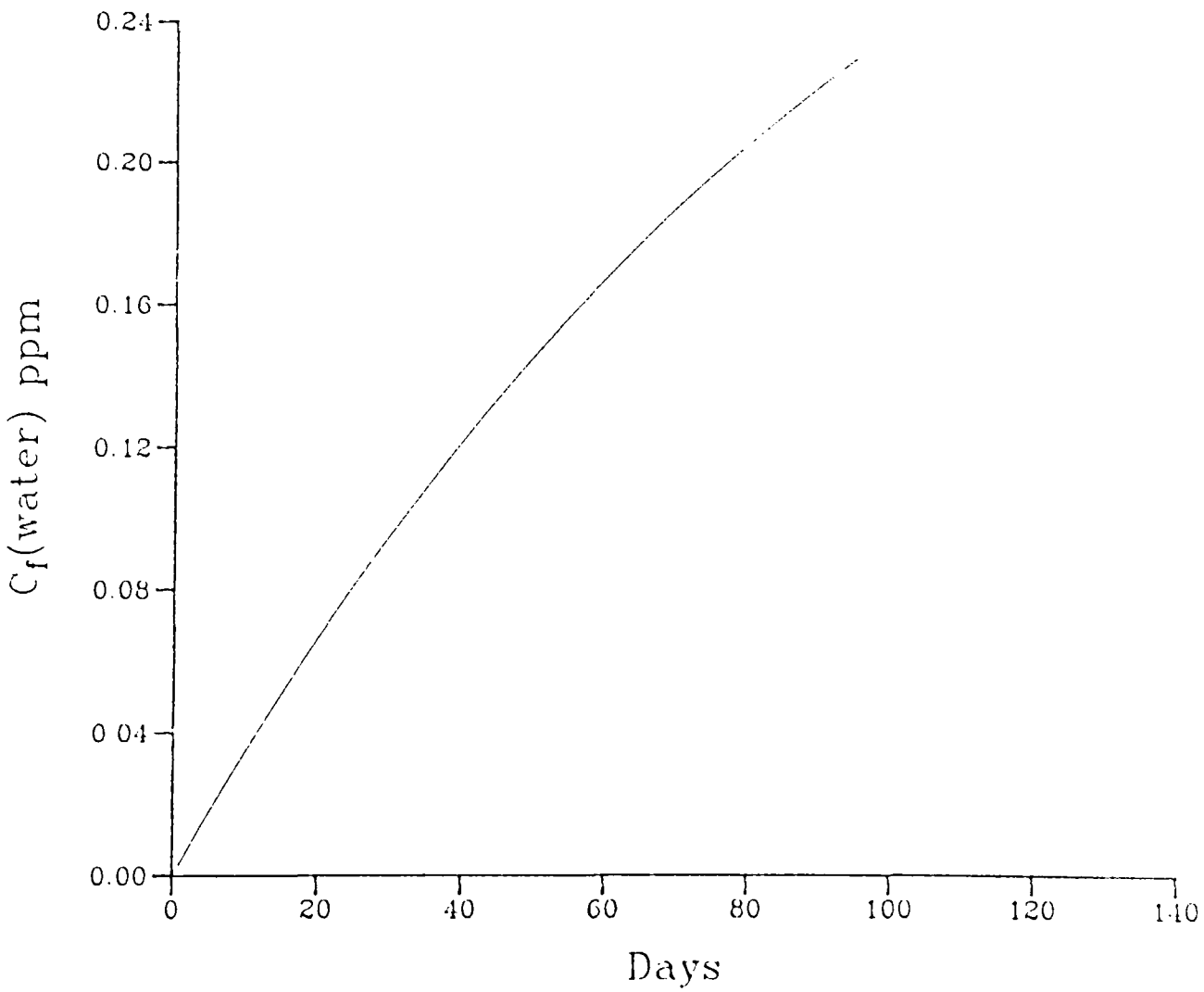
Subroutine DBDT:

This subroutine defines equation (1) for the Euler quadrature.

Subroutine DWDT:

This subroutine defines equation (2) for the Euler quadrature. The user should note that DWDT actually can define the fish's growth dynamics either by equation (2) or by a more mechanistic formulation that is outlined by Barber et al. (1986). The later formulation, however, can be accessed only when analyzing the bioaccumulation of salmonids (i.e., 'famlab' = 'Salmonidae') with the added input designation, 'gamma = -999.0. This simulation option will be expanded later to other fish families.

Figure 1. pcb2233.dat: Gill Exchange of Toxic Substances



Subroutine RDMORH:

This subroutine reads the gill morphometric coefficients and exponents from the data file MORPHO.DAT.

Subroutine LOGSUM:

This subroutine accumulates the logarithmic sums of the morphometric parameters that are read from MORPHO.DAT. These sums are then used to calculate geometric means of those parameters.

Subroutine GILRAT:

This subroutine calculates the mass conductance, kw, of the chemical through the interlamellar water as described in Barber et al. (1986).

Subroutine PHYSIO:

This subroutine calculates feeding and metabolic rates for simulating fish growth when gamma = -999.0. See the discussion of DWDT above.

Subroutine DOFILE:

This subroutine standardizes user supplied input. For example, all character data are transliterated to lower case by subroutine UP2LO.

Subroutine USRERR:

This subroutine identifies errors which GETS encountered regarding the user's input file.

Subroutine INIVAR:

This subroutine initializes the character codes which must accompany the user's input values.

Subroutine PLX010:

This subroutine generates CALCOMP™ plot files.

Subroutine UP2LO:

See DOFILE above.

Function LENSTR:

This function determines the length of the nonblank segment of character variables. This function is used by GETS prior to certain character comparisons and output sequences.

Subroutine C2byte:

This subroutine translates character data to bytes. This subroutine is system dependent and is used for CALCOMP™ plot generated on the Athens ERL VAX 11/785.

Subroutine NAMES:

This subroutine extracts the name of the user's input file to define a model output file and a CALCOMP™ plot file.

Subroutine SCALE:

This subroutine determines aesthetically pleasing axis scaling parameters, e.g., adjusted minimum and increment per tic mark.

Subroutine KDEC:

This subroutine determines the minimum number of decimal places required to uniquely identify the axis' tic marks.

Function LASTNZ:

This integer function determines the position of the first nonzero character of a record (record scanned from right to left). This function is used by KDEC.

REFERENCE

Barber, M. Craig, L.A. Suarez, and Ray R. Lassiter. 1986. Kinetic exchange of nonpolar organic pollutants by fish. Environ. Toxicol. Chem. (submitted)

Appendix A. Documented source listing of GETS

```
c     program GETS.f
c
c\begin
c    program: GETS.f
c    version: 24-SEP-1986 12:54:46
c    purpose: simulate toxicant accumulation via gill exchange
c    required files:
c         - parameter file (input) - default: GETS.dat
c         - morpho.dat (input)
c         - output file (name depends on the input file name)
c
c    references:
c    Barber, C., L. Suarez, and R. Lassiter. 1986.
c\end
c
c general declarations.
c
c
      integer    stdlen,    maxcol
      parameter (stdlen=80, maxcol=132)
      integer  jin, jout, jplx
      logical  done
c
c file io declarations
c
      character*(stdlen) scrfil
c
c data io declarations
c
      character*(stdlen) infil,  outfil, plxfil
c
      integer    kstr,    kvar,    ktotal
      parameter (kstr=03, kvar=09, ktotal=kstr+kvar)
      character*(stdlen) usrstr(kstr), idpara(ktotal)
      character*(stdlen) spplab, famlab, liflab
      real               usrvar(kvar)
      real               molwt, logp, uwt, uplfish, gamma, cfish
      real               cwater, adjust, tend
c
      common / g3i10s / usrstr, idpara
      common / g3i10v / usrvar
c
      equivalence (usrstr(1), spplab), (usrstr(2), famlab)
      equivalence (usrstr(3), liflab)
      equivalence (usrvar(01), molwt),     (usrvar(02), logp)
      equivalence (usrvar(03), uwt),       (usrvar(04), uplfish)
      equivalence (usrvar(05), gamma),     (usrvar(06), cfish)
      equivalence (usrvar(07), cwater),    (usrvar(08), adjust)
      equivalence (usrvar(09), tend)
c
      integer stdin, stdout, stderr
      common / g3i11v / stdin, stdout, stderr
c
c model declarations
c
      character*(stdlen)  family, species, genus, form
      character*(stdlen)  genlab
```

```
      real cvar(3), evar(3)
      real cs, es, crho, erho
      real s1(5), s2(5), rho1(5), rho2(5)
      data s1, s2, rho1, rho2     / 20 * 0.0/
      real ns1(5), ns2(5), nrho1(5), nrho2(5)
      data ns1, ns2, nrho1, nrho2 / 20 * 0.0/
c
      real              d, l, kow, cf
      real              phi, beta1, beta2, beta3, mu,
     *                  p1fish, k1, s, kw, cw
      common/ coeff/    phi, beta1, beta2, beta3, mu,
     *                  p1fish, k1, s, kw, cw
      real              wt, gmax, g, imax, ifood
      common/ masses/   wt, gmax, g, imax, ifood
      real              ingest, evac, assim, egest, respi
      common/ fluxes/   ingest, evac, assim, egest, respi
      integer  n1, n2
      parameter (n1=04, n2=01)
      real     y1(n1), y1prime(n1), y2(n2), y2prime(n2)
      real     k1, k2, gamma0
      integer  division
      real     del
      integer  iprint
c
c plotting declarations
c
      character*(stdlen)  ylabel
      character*(stdlen)  toxlab, wlabel, hlabel, clabel, tlabel
      character*(stdlen)  lab01, lab02, lab03, lab04
      integer    maxpoints
      parameter (maxpoints=503)
      real       ttline(maxpoints), wtline(maxpoints)
      real       cfline(maxpoints)
      integer    ny, nu
c
 9110 format ('$', 'parameter file name: ')
 9120 format (a)
 9130 format (/' ', '??? order of input parameters incorrect')
 9140 format (/' ', '??? unexpected end of file, some parameters ',
     *   'are missing')
 9150 format (' ', 'USER:  input file: ', a)
 9160 format (' ', 'USER: output file: ', a)
 9170 format (' ', 'USER:   plot file: ', a)
 9180 format (' ', 'USER:       molwt: ',1pg11.3/
     *         ' ', 'USER:        logp: ',1pg11.3/
     *         ' ', 'USER:      fishid: ',a/
     *         ' ', 'USER:          wt: ',1pg11.3, 1x, a/
     *         ' ', 'USER:      p1fish: ',1pg11.3/
     *         ' ', 'USER:       gamma: ',1pg11.3/
     *         ' ', 'USER:      adjust: ',1pg11.3/
     *         ' ', 'USER:       cfish: ',1pg11.3, 1x, a/
     *         ' ', 'USER:      cwater: ',1pg11.3, 1x, a/
     *         ' ', 'USER:        tend: ',1pg11.3, 1x, a)
 9190 format (' ', a, 1pg11.3)
 9200 format ('    day     wt          bf          cf')
 9210 format (' ', f6.0, 5(1x, 1pg11.3))
 9220 format (/' ', 'MODEL:          k1: ',1pg11.3/
     *         ' ', 'MODEL:          k2: ',1pg11.3/
     *         ' ', 'MODEL:       gamma: ',1pg11.3)
c
```

```
      call inivar (done)
c
c     set default input and output files and other options
c
      outfil = 'gets.out'
      plxfil = 'gets.plx'
      spplab = 'fish'
      toxlab = 'toxicant'
      tlabel = 'days'
      clabel = 'ppm'
      wlabel = 'live weight, g'
      hlabel = 'gill exchange model'
      infil  = 'gets.dat'
c
c     get input file name
c
      write (stderr, 9110)
      read (stdin, 9120) scrfil
      if (lenstr(scrfil,stdlen) .gt. 0) infil = scrfil
c
c     generate output file names
c
      call names (infil, outfil, plxfil, hlabel)
c
      jin = 01
      open (unit=jin,file=infil,status='old',readonly)
c
c     read and echo all info from file; close file when done
c
      call dofile (jin, ierror)
      wt = uwt
      close (unit=jin)
      if (ierror .ne. 0) then
         if (ierror .eq. 100) write (stderr,9130)
         if (ierror .eq. 101) write (stderr,9140)
         stop
      endif
c
c     open output file
c
      jout = 02
      open (unit=jout,file=outfil,status='new',carriagecontrol='list')
c
      nn = lenstr(infil,stdlen)
      write (stderr,9150) infil(1:nn)
      nn = lenstr(outfil,stdlen)
      write (stderr,9160) outfil(1:nn)
      nn = lenstr(plxfil,stdlen)
      write (stderr,9170) plxfil(1:nn)
      nn = lenstr (spplab, stdlen)
      nw = lenstr (wlabel, stdlen)
      nc = lenstr (clabel, stdlen)
      nt = lenstr (tlabel, stdlen)
      write (stderr,9180) molwt, logp, spplab(1:nn), wt, wlabel(1:nw),
     *    plfish, gamma, adjust, cfish, clabel(1:nc),
     *    cwater, clabel(1:nc),  tend, tlabel(1:nt)
c
c     put header info in output file "jout";
c
      nn = lenstr(infil,stdlen)
```

```
      write (jout,9150) infil(1:nn)
      nn = lenstr(outfil,stdlen)
      write (jout,9160) outfil(1:nn)
      nn = lenstr(plxfil,stdlen)
      write (jout,9170) plxfil(1:nn)
      nn = lenstr (spplab, stdlen)
      nw = lenstr (wlabel, stdlen)
      nc = lenstr (clabel, stdlen)
      nt = lenstr (tlabel, stdlen)
      write (jout,9180) molwt, logp, spplab(1:nn), wt, wlabel(1:nw),
     *    plfish, gamma, adjust, cfish, clabel(1:nc),
     *    cwater, clabel(1:nc), tend, tlabel(1:nt)
c
c read gill morphometric regressions and accumulate sums to
c calculate geometric means of the power function coefficients
c and exponents
c
      input = 10
      open (unit=input,file='morpho.dat',status='old',readonly)
      rewind input
  110 continue
      call rdmorp (input,species,genus,family,form,cvar,evar,done)
      if (.not. done) then
         cs = cvar(1)
         es = evar(1)
         crho = cvar(2)
         erho = evar(2)
c
c calculate sums for species, genus, family, life form, and
c all fish
c
         i = 1
         nn = 40
         if (species(1:nn) .eq. spplab(1:nn)) call logsum (
     *      ns1(i), s1(i), cs,
     *      ns2(i), s2(i), es,
     *      nrho1(i), rho1(i), crho,
     *      nrho2(i), rho2(i), erho)
c
         i = 2
         nn = index(species, ' ')
         genlab = species(1:nn)
         if (genus(1:nn) .eq. genlab(1:nn)) call logsum (
     *      ns1(i), s1(i), cs,
     *      ns2(i), s2(i), es,
     *      nrho1(i), rho1(i), crho,
     *      nrho2(i), rho2(i), erho)
c
         i = 3
         nn = 20
         if (family(1:nn) .eq. famlab(1:nn)) call logsum (
     *      ns1(i), s1(i), cs,
     *      ns2(i), s2(i), es,
     *      nrho1(i), rho1(i), crho,
     *      nrho2(i), rho2(i), erho)
c
         i = 4
         nn = 20
         if (form(1:nn) .eq. liflab(1:nn)) call logsum (
     *      ns1(i), s1(i), cs,
```

```
     *       ns2(i), s2(i), es,
     *       nrho1(i), rho1(i), crho,
     *       nrho2(i), rho2(i), erho)
c
          i = 5
          call logsum (
     *       ns1(i), s1(i), cs,
     *       ns2(i), s2(i), es,
     *       nrho1(i), rho1(i), crho,
     *       nrho2(i), rho2(i), erho)
          go to 110
        endif
        close (unit=input)
c
c calculate geometric means and set gill morphometric parameters
c
        do 130 ii = 5, 1, -1
           if (ns1(ii) .gt. 0.0) cs = 10.0 ** (s1(ii) / ns1(ii))
           if (ns2(ii) .gt. 0.0) es = 10.0 ** (s2(ii) / ns2(ii))
           if (nrho1(ii) .gt. 0.0) crho = 10.0 ** (rho1(ii) / nrho1(ii))
           if (nrho2(ii) .gt. 0.0) erho = -10.0 ** (rho2(ii) / nrho2(ii))
   130  continue
        nn = lenstr (spplab, stdlen)
        lab01 = 'MODEL: ' // spplab(1:nn) // '   s1: '
        lab02 = 'MODEL: ' // spplab(1:nn) // '   s2: '
        lab03 = 'MODEL: ' // spplab(1:nn) // ' rho1: '
        lab04 = 'MODEL: ' // spplab(1:nn) // ' rho2: '
        nn = lenstr(lab01,stdlen)
        write(stderr, 9190) lab01(1:nn), cs
        write(jout,   9190) lab01(1:nn), cs
        nn = lenstr(lab02,stdlen)
        write(stderr, 9190) lab02(1:nn), es
        write(jout,   9190) lab02(1:nn), es
        nn = lenstr(lab03,stdlen)
        write(stderr, 9190) lab03(1:nn), crho
        write(jout,   9190) lab03(1:nn), crho
        nn = lenstr(lab04,stdlen)
        write(stderr, 9190) lab04(1:nn), erho
        write(jout,   9190) lab04(1:nn), erho
c
c       set state variables
c
c x      = time (in days)
c y1(1) = body weight of fish;
c y1(2) = mass of food in stomach;
c y1(3) = mass of food in intestine;
c y1(4) = integrated specific growth rate;
c y2(1) = mass of toxicant in fish due to gill exchange;
c
        xmin = 0.0
        if(tend .le. float(maxpoints-3)) then
           iprint = 1
        else
           iprint = 1 + int(tend / float(maxpoints-3))
        endif
        x = xmin
        y1(1) = wt
        if (gamma .lt. -990.0) then
           call physio
     *    (famlab, wt, gmax, imax, phi, beta1, beta2, beta3, mu)
```

```
         y1(2) = 0.01 * gmax
         y1(3) = 0.01 * imax
         y1(4) = 0.00
      endif
      cw = cwater
      cf = cfish
      y2(1) = cf * wt
c
c set plotting variables
c
      nxy = 1
      ttline(nxy) = xmin
      wtline(nxy) = y1(1)
      cfline(nxy) = y2(1) / y1(1)
c
c set constant model parameters and simulation limits
c      common/ coeff/  phi, beta1, beta2, beta3, mu,
c     *                plfish, K1, s, kw, cw
c
      if (uplfish .gt. 0.0) then
         plfish = uplfish
      else
         plfish = 0.05
      endif
      kow = 10.0 ** logp
      k1 = 1.528 * kow ** 0.973
      k1 = 0.0
      k2 = 0.0
      jend = ifix(tend)
c
      write (stderr,9200)
      write (jout,  9200)
      write (stderr,9210) x, y1(1), y2(1), cf
      write (jout,  9210) x, y1(1), y2(1), cf
c
      do 150 jj = 1, jend
         xj = float(jj)
         wt0 = y1(1)
c
c calculate feeding and metabolic rates.
c
         if (gamma .lt. -990.0) then
            call physio
     *      (famlab, wt0, gmax, imax, phi, beta1, beta2, beta3, mu)
         endif
c
c calculate gill morphometrics
c
         s =  cs * wt0 ** es
         rho = crho * wt0 ** erho
         d = 0.1030 * rho ** (-1.1416)
         l = 0.0187 * wt0 ** 0.208
c
c calculate exchange rates. first convert mass conductance, kw = cm/ sec,
c to kw = cm/ day. calculate first order uptake and elimination rates, k1
c and k2, respectively, with any desired adjustment (see Barber et al. 1976)
c
         call gilrat(molwt, s, d, l, kw)
         kw  = adjust * 86400.0 * kw
         k1 = k1 + s * kw / wt0
```

```
         k2 = k2 + s * kw / (wt0 * plfish * kl)
c
c integrate toxicokinetic model
c
         division = 100
         del = 1.0 / float(division)
         do 140 idelta = 1, division
            call dwdt(n1, x, y1, y1prime)
            y1(1) = y1(1) + del * y1prime(1)     ! wt
            y1(2) = y1(2) + del * y1prime(2)     ! s
            y1(3) = y1(3) + del * y1prime(3)     ! i
            y1(4) = y1(4) + del * y1prime(4)     ! integral gamma0
            call dbdt(n2, x, y2, y2prime)
            y2(1) = y2(1) + del * y2prime(1)     ! bf
  140    continue
         if (mod(jj,iprint) .eq. 0) then
            cf = y2(1) / y1(1)
            write (stderr,9210) xj, y1(1), y2(1), cf
            write (jout,  9210) xj, y1(1), y2(1), cf
            nxy = nxy + 1
            nn = min (nxy, maxpoints-3)
            ttline(nn) = xj
            wtline(nn) = y1(1)
            cfline(nn) = cf
         endif
  150 continue
      k1 = k1/ float(jend)
      k2 = k2/ float(jend)
      gamma0  = y1(4)
      gamma0  = gamma0/ float(jend)
c
c     output summary parameters to file "jout";
c
      write (stderr, 9220) k1, k2, gamma0
      write (jout,   9220) k1, k2, gamma0
c
      close (unit=jout)
c
c     plot
c
      jplx = 03
      open (unit=jplx,file=plxfil,status='new')
c
      call plots (0.0,0.0,jplx)
      call style (012)
      call darkns (0.012,2)
      call script (0.80)
c
c plot growth dynamics
c
      call plx010 (tlabel, wlabel, hlabel,
     *             ttline, wtline, nxy)
c
c plot Cfish dynamics predicted by gill exchange
c
      lab01 = 'c(f(water)'
      nu = lenstr (clabel, stdlen)
      ny = lenstr (lab01, stdlen)
      ylabel = lab01(1:ny) // ' ' // clabel(1:nu)
      call plx010 (tlabel, ylabel, hlabel,
```

```
     *                    ttline, cfline, nxy)
c
  999 continue
      call plot (0.0,0.0,999)
      stop
c
      end
```

```
      subroutine dwdt(n, x, y, yprime)
c
c This subroutine formulates the differential equations required to
c model fish's growth, i.e.,
c
c    dWt = INGEST - EGEST - RESPI = gamma * wt
c    dt
c
c    dG = INGEST - EVAC
c    dt
c
c    dI = EVAC - ASSIM - EGEST
c    dt
c
c    dGamma0 = (dW/dt) / Wt
c    dt
c
c Input arguments:
c -n     = number of differential equations;
c -x     = time (days);
c -y(1) = body weight of fish(g live);
c -y(2) = mass of food in stomach(g);
c -y(3) = mass of food in intestine(g);
c -y(4) = integrated specific growth rate;
c
c Output arguments:
c -yprime(1) = dWT/dt (see above)
c -yprime(2) = dG /dt
c -yprime(3) = dI /dt
c -yprime(4) = dGamma0 / dt
c
      integer    stdlen,    maxcol
      parameter (stdlen=80, maxcol=132)
c
      integer n
      real  y(n), yprime(n), x
      real             phi, beta1, beta2, beta3, mu,
     *                 plfish, kl, s, kw, cw
      common/ coeff/   phi, beta1, beta2, beta3, mu,
     *                 plfish, kl, s, kw, cw
      real             wt, gmax, g, imax, i
      common/ masses/  wt, gmax, g, imax, i
      real             ingest, evac, assim, egest, respi
      common/ fluxes/  ingest, evac, assim, egest, respi
c
      integer    kstr,    kvar,    ktotal
      parameter (kstr=03, kvar=09, ktotal=kstr+kvar)
      character*(stdlen) usrstr(kstr), idpara(ktotal)
      character*(stdlen) spplab, famlab, liflab
      real               usrvar(kvar)
      real               molwt, logp, uwt, uplfish, gamma, cfish
      real               cwater, adjust, tend
c
      common / g3i10s / usrstr, idpara
      common / g3i10v / usrvar
c
      equivalence (usrstr(1), spplab), (usrstr(2), famlab)
      equivalence (usrstr(3), liflab)
      equivalence (usrvar(01), molwt),     (usrvar(02), logp)
      equivalence (usrvar(03), uwt),       (usrvar(04), uplfish)
```

```
      equivalence (usrvar(05), gamma),     (usrvar(06), cfish)
      equivalence (usrvar(07), cwater),    (usrvar(08), adjust)
      equivalence (usrvar(09), tend)
c
      integer stdin, stdout, stderr
      common / g3i11v / stdin, stdout, stderr
c
      wt = y(1)
      g = max(0.0001, y(2))
      i = max(0.0001, y(3))
      if (gamma .lt. -990.0) then
         ingest = phi * (gmax - g)
         evac = min (beta1 * sqrt(g), imax - i)
         assim = beta2 * sqrt(i)
         egest = beta3 * sqrt(i)
         respi = mu * wt
         yprime(1) = ingest - egest - respi
         yprime(2) = ingest - evac
         yprime(3) = evac - assim - egest
         yprime(4) = yprime(1) / wt              ! gamma0
      else
         yprime(1) = gamma * wt
         yprime(2) = 0.00
         yprime(3) = 0.00
         yprime(4) = 0.00
      endif
c
      return
      end
```

```
      subroutine dbdt(n, x, y, yprime)
c
c This subroutine formulates the differential equation that describes
c the whole body burden of a chemical in fish due to gill exchange which
c is modeled by diffusive transport, i.e.,
c
c
c     dBf = s*kw * (Cw - Ca),
c     dt
c
c     dBf = s*kw * (Cw - Cf/(Pl*Kl)),
c     dt
c
c Input arguments:
c -n     = number of differential equations;
c -x     = time (days);
c -y(1) = mass of toxicant in fish (micro g);
c
c Output arguments:
c -yprime(1) = dBf/dt
c
      integer n
      real  y(n), yprime(n), x
      real              phi, beta1, beta2, beta3, mu,
     *                  plfish, kl, s, kw, cw
      common/ coeff/  phi, beta1, beta2, beta3, mu,
     *                  plfish, kl, s, kw, cw
      real              wt, gmax, g, imax, i
      common/ masses/ wt, gmax, g, imax, i
      real              ingest, evac, assim, egest, respi
      common/ fluxes/ ingest, evac, assim, egest, respi
c
      real  ca, cf
c
      cf = y(1) / wt
      ca = cf / (plfish * kl)
      yprime(1) = s * kw * (cw - ca)
c
      return
      end
```

```
      subroutine rdmorp (jin, species, genus, family, lifeform,
     *                   cvar, evar, done)
c
c This subroutine reads the coefficients and exponents for the
c allometric functions,
c
c     s   =  gill area (cm**2) = cvar(1) * wt ** evar(1),
c     rho =  # lamellae / mm gill filament = cvar(2) * wt ** evar(2),
c     l   =  lamellar length (cm) = cvar(3) * wt ** evar(3),
c
c from the GETS data base.
c
c Input arguments:
c -jin = the logical unit number to which the GETS data base is attached.
c
c Output arguments:
c -species  = species  name of the record just read (character, lowercase)
c -genus    = genus    name of the record just read (character, lowercase)
c -family   = family   name of the record just read (character, lowercase)
c -lifeform = lifeform type of the record just read (character, lowercase)
c -cvar     = see above
c -evar     = see above
c -done     = truth of "end of file marker read"
c
      integer               jin, jj
      character*(*)         species, genus, family, lifeform
      real                  cvar(1), evar(1)
      logical               done
c
 9110 format (6a)
 9120 format (6e10.3)
c
      species  = ' '
      genus    = ' '
      family   = ' '
      lifeform = ' '
      done     = .false.
c
      read (jin, 9110, end=110) species(1:40), family(1:20),
     *                          lifeform(1:20)
      jj = index (species, ' ')
      genus = species (1:jj)
      call up2lo (species, 40)
      call up2lo (family,  20)
      call up2lo (lifeform, 20)
      call up2lo (genus,   jj)
      read (jin,*,end=110)      ! ignore next record (ref)
      read (jin, 9120, end=110) ( (cvar(jj),evar(jj)), jj = 1, 3)
      done     = .false.
      go to 999
c
  110 continue
      done = .true.
c
  999 continue
      return
c
      end
```

```
      subroutine logsum (
     *   ns1, s1, cs,
     *   ns2, s2, es,
     *   nrho1, rho1, crho,
     *   nrho2, rho2, erho)
c
c This subroutine accumulates the logarithmic sums needed to calculate the
c geometric means of the coefficients and exponents of the allometric
c functions,
c
c     s   =  gill area (cm**2) = cs * wt ** es,
c     rho =  # lamellae / mm gill filament = crho * wt ** erho,
c
c Input arguments (modified by logsum):
c -ns1      = cummulative number of records having data for cs
c -s1       = cummulative sum of log10(cs)
c -ns2      = cummulative number of records having data for es
c -s2       = cummulative sum of log10(es)
c -nrho1    = cummulative number of records having data for crho
c -rho1     = cummulative sum of log10(crho)
c -nrho2    = cummulative number of records having data for erho
c -rho2     = cummulative sum of log10(erho)
c
c Input arguments (NOT modified by logsum):
c -cs       = coefficient read from the GETS data base
c -es       = exponent    read from the GETS data base
c -crho     = coefficient read from the GETS data base
c -erho     = exponent    read from the GETS data base
c
c Note:
c  Missing values in GETS data base signified by -999.000.
c
      real  ns1, s1, cs
      real  ns2, s2, es
      real  nrho1, rho1, crho
      real  nrho2, rho2, erho
      if (cs .gt. -999.) then
        ns1 = ns1 + 1.0
        s1 = s1 + alog10(cs)
      endif
      if (es .gt. -999.) then
        ns2 = ns2 + 1.0
        s2 = s2 + alog10(es)
      endif
      if (crho .gt. -999.) then
        nrho1 = nrho1 + 1.0
        rho1 = rho1 + alog10(crho)
      endif
      if (erho .gt. -999.0 .and. erho .lt.0) then
        nrho2 = nrho2 + 1.0
        rho2 = rho2 + alog10(-erho)
      endif
      return
      end
```

```
      subroutine gilrat (molwt, s, d, l, kw)
c
c This subroutine calculates gill morphometric and mass transport
c parameters for the gill exchange model,
c
c      dBf = s*kw * (Cw - Ca),
c      dt
c
c      dBf = s*kw * (Cw - Cf/(Pl*Kl)),
c      dt
c
c where Bf is the body burden of toxicant.
c
c Input arguments:
c   -molwt = molecular weight of the chemical;
c   -s     = total gill surface area (cm**2);
c   -d     = average interlamellar distance (cm);
c   -l     = average lamellar length (cm);
c
c Output argument:
c   -kw    = mass conductance (cm/ sec);
c
c Internal variables:
c   -v = mean velocity of interlamellar flow (cm/ sec);
c   -NRe = (v * d) / nu = Reynolds number;
c   -difusi = diffusion coefficient (cm**2/ sec);
c   -nu = kinematic viscosity of water;
c   -NSc = nu / difusi = Schmidt number;
c   -x = axial distance down the lamellar channel and in particular
c    the channels entry length (cm);
c   -NGz = x / (d * NRe * NSc) = x * D / (d**2 * v) = Graetz number;
c   -NShf = (kw * d) / difusi = fluid side Sherwood number;
c
c
c   references:
c   Barber, C., L. Suarez, and R. Lassiter. 1986.
c   Colton, C., K. Smith, P.Stroeve, and E. Merrill. 1971.
c      AIChE Journ. 17:773-790.
c   Ingham, D. 1984. Int.J.heat Mass Transfer 27:2421-2422.
c
      real   molwt, s, d, l
      real   v, difusi, nu, nre, nsc, ngz
      real   x, inshl, insh2, nshw, nshf
      real   kw
c
c calculate lamellar Reynolds and Schmidt numbers.
c
      v = 8.333e3 * d**2 / l
      nu = 1.004e-2
      nre = d * v / nu
      difusi = 2.41e-5 * sqrt(32.0/molwt)
      nsc = nu/ difusi
c
c calculate length, x, of a lamellar channel's entrance region,
c i.e., NGz=0.1 where NGz=Graetz number of the lamellar channel.
c
      ngz = 0.1
      x = ngz * d * nre * nsc
c
c calculate integrated Sherwood numbers, INSh1 and INSh2, for the
```

```
c entrance region and post entrance region, respectively.
c For enterance region fluid side Sherwood number, NShf, see
c Ingham (1984). Since in the enterance region the NShw apparently
c converge to the constant flux curve (see Colton et al. 1971.),
c assume NShw=0.
c
      nshw = 0.0
      if ( x .lt. 1) then
          insh1 = 1.1829 * ngz **(-.333333) * 1.5 * x
     *          - (0.1167 * nshw + 0.1767 ) * x
          insh2 = (1 - x) * 3.77035
      else
          ngz = 1 / (d * nre * nsc)
          insh1 = 1.1829 * ngz **(-.333333) * 1.5 * 1
     *          - (0.1167 * nshw + 0.1767 ) * 1
          insh2 = 0.0
      endif
      nshf = ( insh1 + insh2 ) / 1
      kw = (nshf * difusi) / d
      return
      end
```

```
      subroutine physio
     *   (famlab, wt, gmax, imax, phi, beta1, beta2, beta3, mu)
c
c This subroutine calculates daily feeding and metabolic rates for
c the models,
c
c    dWt =  INGEST - EGEST - RESPI
c    dt
c
c    dG =  INGEST - EVAC
c    dt
c
c    dI =  EVAC - ASSIM - EGEST
c    dt
c
c    INGEST = phi * (Gmax - G)
c
c    EVAC   = beta1  * G ** 1/2
c
c    ASSIM = beta2 * I ** 1/2
c
c    EGEST = beta3 * I ** 1/2
c
c Input arguments:
c -famlab    = family name of fish (character, lowercase);
c -wt        = gram live weight of fish;
c
c Output variables:
c -Gmax      = maximum capacity of stomach (gram);
c -Imax      = maximum intestinal capacity (gram);
c -phi       = maximum feeding rate (1/day);
c -beta1     = stomach evacuation rate (gram**0.5/day);
c -beta2     = assimilation rate (gram**0.5/day);
c -beta3     = egestion rate (gram**0.5/day);
c -mu        = specific metabolic rate (gram/gram/day);
c
c Internal variables:
c -T         = environmental temperature (C);
c -Tsat      = time to satiation when feeding with an initially
c              empty stomach (min);
c -Fsat      = size of satiation meal consumed during (0, Tsat), gram;
c
c references:
c   Grove, D., L. Loizides, and J. Nott. 1978. J.Fish Biol. 12:507-516.
c   Jobling, M. 1981. J.Fish Biol. 19:245-257.
c   Staples, D. and M. Momura. 1976. J.Fish Biol. 9:29-43.
c   Windell, J., D. Norris, J. Kitchell, and J. Norris. 1969. J.Fish.Bd.
c      Canada 26:1801-1812.
c
      character*(*) famlab
      real wt, gmax, imax, phi, beta1, beta2, beta3, mu
      real t, tsat, fsat
c
      if (famlab(1:10) .eq. 'salmonidae') then
         t = 10.0
c
c feeding coefficients calculated for Salmo gairdneri
c (Grove et al. 1978).
c
         tsat = (0.031 * wt + 0.868 * t + 29.145) /
```

```
     *       (60.0 * 24.0)
          fsat = 0.95 * 0.029 * wt ** 1.02
          gmax = 0.075 * wt
          phi = - alog (1.0 - fsat / gmax) / tsat
          beta1 = 1.1413 * exp (0.05246 * t)
          imax = 0.020 * wt ** 0.303
c
c assume assimilation efficiency equals 0.80 and calculate
c beta2 and beta3 accordingly. also assume intestinal evacuation
c rates approximately equal stomach evacuation rates (see for
c example Grove et al. 1978, Table IV).
c
          beta2 = 0.80 * beta1
          beta3 = 0.20 * beta1
c
c RESPI calculated for feeding Salmo gairdneri(Staples and Nomura 1976)
c assuming R.Q.=0.8 and Q10=2.
c
          mu = 0.00376 * wt ** (-0.187) * exp(0.0693 * (t - 11.4))
       endif
       return
       end
```

```
      subroutine dofile (jin, ierror)
c
c This subroutine reads the user parameter file.
c
c Input arguments:
c -jin = the logical unit number to which the user parameter file is attached.
c
c Output arguments:
c -ierror = error flag
c         = 000 ==> no error
c         = 100 ==> incorrect data item found
c         = 101 ==> unexpected End Of File (some data may be missing)
c
      integer stdlen, maxcol
      parameter (stdlen=80, maxcol=132)
c
      integer    kstr,    kvar,    ktotal
      parameter (kstr=03, kvar=09, ktotal=kstr+kvar)
      character*(stdlen) usrstr(kstr), idpara(ktotal)
      character*(stdlen) spplab, famlab, liflab
      real               usrvar(kvar)
      real               molwt, logp, uwt, uplfish, gamma, cfish
      real               cwater, adjust, tend
c
      common / g3i10s / usrstr, idpara
      common / g3i10v / usrvar
c
      equivalence (usrstr(1), spplab), (usrstr(2), famlab)
      equivalence (usrstr(3), liflab)
      equivalence (usrvar(01), molwt),     (usrvar(02), logp)
      equivalence (usrvar(03), uwt),       (usrvar(04), uplfish)
      equivalence (usrvar(05), gamma),     (usrvar(06), cfish)
      equivalence (usrvar(07), cwater),    (usrvar(08), adjust)
      equivalence (usrvar(09), tend)
c
      integer stdin, stdout, stderr
      common / g3i11v / stdin, stdout, stderr
c
      integer jin, ierror
      character*(maxcol) idline
      integer  lenstr
      logical  ok, okdoky
c
      ok = .true.
      ierror = 00
c
      read (jin, *, end=999) idline, spplab
      call up2lo (spplab, stdlen)
      call usrerr (01, idline, okdoky)
      ok = (ok .and. okdoky)
c
      read (jin, *, end=999) idline, famlab
      call up2lo (famlab, stdlen)
      call usrerr (02, idline, okdoky)
      ok = (ok .and. okdoky)
c
      read (jin, *, end=999) idline, liflab
      call up2lo (liflab, stdlen)
      call usrerr (03, idline, okdoky)
      ok = (ok .and. okdoky)
```

```
c
      do 110 jj = 1, kvar
         read (jin, *, end=999) idline, vtemp
         nvar = kstr + jj
         call usrerr (nvar, idline, okdoky)
         if (okdoky) usrvar(jj) = vtemp
         ok = (ok .and. okdoky)
  110 continue
      if (.not. ok) ierror = 100
      return
c
  999 continue
      ierror = 101
      return
      end
```

```
      subroutine usrerr (idnum, idline, okdoky)
c
      integer stdlen, maxcol
      parameter (stdlen=80, maxcol=132)
c
      integer    kstr,    kvar,    ktotal
      parameter (kstr=03, kvar=09, ktotal=kstr+kvar)
      character*(stdlen) usrstr(kstr), idpara(ktotal)
      character*(stdlen) spplab, famlab, liflab
      real               usrvar(kvar)
      real               molwt, logp, uwt, uplfish, gamma, cfish
      real               cwater, adjust, tend
c
      common / g3i10s / usrstr, idpara
      common / g3i10v / usrvar
c
      equivalence (usrstr(1), spplab), (usrstr(2), famlab)
      equivalence (usrstr(3), liflab)
      equivalence (usrvar(01), molwt),     (usrvar(02), logp)
      equivalence (usrvar(03), uwt),       (usrvar(04), uplfish)
      equivalence (usrvar(05), gamma),     (usrvar(06), cfish)
      equivalence (usrvar(07), cwater),    (usrvar(08), adjust)
      equivalence (usrvar(09), tend)
c
      integer stdin, stdout, stderr
      common / g3i11v / stdin, stdout, stderr
c
      character*(*) idline
      integer       idnum, len, lenstr, n1, n2
      logical       okdoky
      external      lenstr, up2lo
c
      n1 = lenstr(idline, len(idline))
      call up2lo (idline, n1)
      if (idline(1:n1) .eq. idpara(idnum)(1:n1)) then
         okdoky = .true.
      else
         n2 = lenstr(idpara(idnum), len(idpara(idnum)))
         write (stderr, 9110) idline(1:n1), idpara(idnum)(1:n2)
         okdoky = .false.
      endif
c
      return
c
 9110 format (' ','??? error: user input "', a,
     *   '"; expecting "', a, '"')
c
      end
```

```
      subroutine inivar (done)

: This subroutine initializes the variables used by GETS.f

: Input arguments:
:  NONE
:
: Output arguments:
: -done = truth of "initialization completed successfully"
:
      integer stdlen, maxcol
      parameter (stdlen=80, maxcol=132)
c
      integer    kstr,    kvar,    ktotal
      parameter (kstr=03, kvar=09, ktotal=kstr+kvar)
      character*(stdlen) usrstr(kstr), idpara(ktotal)
      character*(stdlen) spplab, famlab, liflab
      real               usrvar(kvar)
      real               molwt, logp, uwt, uplfish, gamma, cfish
      real               cwater, adjust, tend
c
      common / g3i10s / usrstr, idpara
      common / g3i10v / usrvar
c
      equivalence (usrstr(1), spplab), (usrstr(2), famlab)
      equivalence (usrstr(3), liflab)
      equivalence (usrvar(01), molwt),     (usrvar(02), logp)
      equivalence (usrvar(03), uwt),       (usrvar(04), uplfish)
      equivalence (usrvar(05), gamma),     (usrvar(06), cfish)
      equivalence (usrvar(07), cwater),    (usrvar(08), adjust)
      equivalence (usrvar(09), tend)
c
      integer stdin, stdout, stderr
      common / g3i11v / stdin, stdout, stderr
c
      logical done
c
      stdin  = -3
      stdout = -2
      stderr = stdout
c
      idpara(01) = 'spplab'
      idpara(02) = 'famlab'
      idpara(03) = 'liflab'
      idpara(04) = 'molwt'
      idpara(05) = 'logp'
      idpara(06) = 'wt'
      idpara(07) = 'plfish'
      idpara(08) = 'gamma'
      idpara(09) = 'cfish'
      idpara(10) = 'cwater'
      idpara(11) = 'adjust'
      idpara(12) = 'tend'
      do 110, jj = 1, ktotal
         call up2lo (idpara(jj), stdlen)
  110 continue
      done = .true.
c
      return
      end
```

```
      subroutine plx010 (xlabel, ylabel, zlabel,
     *                   xxline, yyline, nxylin )
c
c This subroutine generates CALCOMP(tm) plot files.
c Warning: this subroutine is executable at the A-ERL; it may or may not
c          be compatible elsewhere.
c
c Input arguments:
c -xlabel = x-axis label (character);
c -ylabel = y-axis label (character);
c -zlabel = plot title (character);
c -xxline = array with abscisae to be plotted;
c -yyline = array with ordinates to be plotted;
c -nxylin = number of coordinates;
c
      integer stdlen, maxcol
      parameter (stdlen=80, maxcol=80)
c
      character*(stdlen) xlabel, ylabel, zlabel
      integer            nxylin, ii
      real               xxline(1), yyline(1)
c
      byte qtemp(stdlen)
      real xxpar(5),yypar(5)
      real zmax, zmin, xorign, yorign, size1, size2
      real xlen, xmin, xdx, xdl, dxpow, xs, xe, xt, xn
      real ylen, ymin, ydx, ydl, dypow, ys, ye, yt, yn
      integer jj, ip, kobs
      integer nxdec, nydec, nx, ny, nz, lenstr, jxpow, jypow, nxypar
      external lenstr
c
      data nxypar  / 2 /, xorign / 1.0 /, yorign / 1.0 /
      data xlen / 7.0 /, ylen / 6.0 /, size1 / 0.12 /, size2 / 0.18 /
c
      if (nxylin .le. 1) return
      nxdec = 5
      nydec = 5
c
      call scale (xxline, xlen, nxylin, 1)
      xmin = xxline(nxylin+1)
      xdx  = xxline(nxylin+2)
c
      call scale (yyline, ylen, nxylin, 1)
      ymin = yyline(nxylin+1)
      ydx  = yyline(nxylin+2)
c
      call plot (xorign, yorign, -3)
c
c     put axes
c
      nx = lenstr (xlabel, stdlen)
      xs = xmin
      xe = (xmin   +   xlen * xdx)
      xn = xdx
      xt = xn
      call kdec (xs,xe,xt,xs,xs,xe,xn,nxdec)
      call linaxs (0.0, 0.0, xlen, 00.0, size1, xs, xe, xt, xs
     *           , xs, xe, xn, nxdec, 1)
      call c2byte (xlabel, nx, qtemp)
      call positn (xlen*0.5, -(size1*2.25+size2*2.50), size2
```

```
     *            , qtemp, 00.0, nx, 1)
c
      ny = lenstr (ylabel, stdlen)
      ys = ymin
      ye = (ymin    +    ylen * ydx)
      yn = ydx
      yt = yn
      call kdec (ys,ye,yt,ys,ys,ye,yn,nydec)
      call linaxs (0.0, 0.0, ylen, 90.0, size1, ys, ye, yt, ys
     *            , ys, ye, yn, nydec, 4)
      call c2byte (ylabel, ny, qtemp)
      call positn (-(size1*5.0+size2*2.0), ylen*0.5, size2
     *            , qtemp, 90.0, ny, 1)
c
c     put id stuff
c
      nz = lenstr (zlabel, stdlen)
      call c2byte (zlabel, nz, qtemp)
      xpc = xlen * 0.5
      ypc = 3.00  +  ylen  +  size2*2.50
      call positn (xpc, ypc, size2, qtemp, 00.0, nz, 1)
      ypc = ypc - 2.0 * size2
c
      call line (0.0,0.0,xxline,yyline,nxylin,1,00,23,size1,0.0)
c
      call plot (-xorign, -yorign, -3)
      call plot (xlen+xorign+2.00, 00.0, -3)
c
  999 continue
      return
      end
```

```
      subroutine up2lo (letter,nleter)
c
c Warning: this is a system-dependent routine
c This subroutine transliterates upper case to lower case.
c
c Input arguments:
c -letter = character array with characters to be transliterated:
c           will be modified on output;
c -nleter = number of characters to transliterate;
c
c     upper-case-a = 065; upper-case-z = 090
c     lower-case-a = 097; lower-case-z = 122
c
      integer nleter, offset, jj, hold
      character*1 letter(1)
c
      offset = 097 - 065
      do 110 jj = 1, nleter
         hold = ichar(letter(jj))
         if ((065.le.hold).and.(hold.le.090)) then
            letter(jj) = char( hold + offset)
         endif
  110 continue
c
      return
      end
```

```
      integer function lenstr (string, nchar)

This function determines the length of a character string;
trailing blanks will not be counted.

Input arguments:
-string = character array
-nchar  = maximum number of characters in array string

Output value:
-lenstr = number of characters in string

      integer nchar, nn
      character*(1) string(1)

      if (nchar .gt. 0) then
         nn = nchar + 1
 110     continue
         nn = nn - 1
         if (nn .gt. 0) then
            if (string(nn) .eq. ' ') go to 110
         endif
         lenstr = nn
      else
         lenstr = 00
      endif

      return
      end
```

```
      subroutine c2byte (zzchar, nchar, zzbyte)
c
c This subroutine translates characters to bytes. This
c subroutine is used only for plotting purposes.
c
c Input arguments:
c -zzchar = character array
c -nchar  = number of characters to be translated
c
c Output argument:
c -zzbyte = byte equivalent of the array zzchar
c
      integer        nchar, jj
      character*(1) zzchar(1)
      byte           zzbyte(1)
c
      do 110, jj = 1, nchar
         zzbyte(jj) = ichar(zzchar(jj))
  110 continue
c
      return
      end
```

```
      subroutine names (infil, outfil, plxfil, zlabel)
c
c This subroutine generates the names for the output file, the
c plot file and the title for the plot.
c
c Input arguments:
c -infil  = character array containing the name of the user input file;
c
c Output arguments:
c -outfil = character array containing the name of the user output file;
c -plxfil = character array containing the name of the user plot file;
c -zlabel = character array containing the plot title;
c
c Note:
c     name of the form "filename.ext"; NO (sub)directory info.
c
      character*(*) infil, outfil, plxfil, zlabel
      integer       nn, lenstr, len, ni
      external      lenstr
c
      ni = lenstr(infil, len(infil))
      nn = index (infil, '.')
      if (nn .gt. 0) then
         nn = nn - 1
      else
         nn = ni
      endif
c
      outfil = infil(1:nn) // '.out'
      plxfil = infil(1:nn) // '.plx'
      zlabel = infil(1:ni) //': '// 'gill exchange of toxic substances'
c
      return
      end
```

```
      subroutine scale (array,axlen,npts,inc)
c
calcomp hcbs pn 725012/925012  for ibm system/360  july 6,1970
copyright      1970   california   computer   products   inc.
c     update: 19-sep-1986 11:05:38.94; A-ERL (EPA)
c
c This subroutine determines "aesthetically pleasing" axis scaling
c parameters.
c
c Input parameters:
c -array   = name of array containing values to be scaled;
c -axlen   = length in inches over which array is to be scaled;
c -npts    = number of points to be scaled;
c -inc     = increment of location of successive points;
c
c Output parameters:
c -array(npts+inc)   = adjusted axis minimum;
c -array(npts+2*inc) = axis increment per inch (tic mark);
c
      integer nsave
      parameter (nsave=11)
      dimension array(1),save(nsave)
      data save / 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0,
     *            10.0, 20.0 /
c
      fad = 0.010
      k = iabs(inc)
      n = npts*k
      y0 = array(1)
      yn = y0
c
      do 140 i = 1, n, k
         ys = array(i)
         if (y0-ys) 120, 120, 110
  110    y0 = ys
         go to 140
  120    if (ys-yn) 140, 140, 130
  130    yn = ys
  140 continue
c
      firstv = y0
      if (y0) 150, 160, 160
c
  150 fad = fad-1.0
  160 deltav = (yn-firstv)/axlen
      if (deltav) 270, 270, 170
c
  170 i = alog10(deltav)+1000.0
      p = 10.0**(i-1000)
      deltav = deltav/p - 0.01
      do 180 i = 1, nsave-1
         is = i
         if (save(i)-deltav) 180, 190, 190
  180 continue
c
  190 deltav = save(is)*p
      firstv = deltav*aint(y0/deltav+fad)
      t = firstv+(axlen+0.01)*deltav
      if (t-yn) 200, 220, 220
c
```

```
  200 firstv = p*aint(y0/p+fad)
      t = firstv+(axlen+0.01)*deltav
      if (t-yn) 210, 220, 220
c
  210 is = is+1
      go to 190
  220 firstv = firstv-aint((axlen+(firstv-yn)/deltav)/2.0)*deltav
      if (y0*firstv) 230, 230, 240
  230 firstv = 0.0
  240 if (inc) 250, 250, 260
  250 firstv = firstv+aint(axlen+.5)*deltav
      deltav = -deltav
  260 n = n+k
      array(n) = firstv
      n = n+k
      array(n) = deltav
      return
c
  270 deltav = 2.0*firstv
      deltav = abs(deltav/axlen)+1.0
      go to 170
c
      end
```

```
      subroutine kdec (stic,etic,ticinc,sval,bval,eval,valinc,numdec)
c
c This subroutine determines the minimum number of decimal places
c required to make each tic mark unique.
c
c Input arguments:
c -stic      = value of starting tick mark on axis ( in units );
c -etic      = value of ending tick mark on axis ( in units );
c -ticinc    = positive distance between tick marks ( in units )
c -sval      = starting value at (x,y) on axis ( in units )
c -bval      = value of first number at a tic mark on axis ( in units )
c -eval      = ending value on axis ( in units )
c -valinc    = positive distance between numbers on axis ( in units )
c -numdec    = on input:
c              number of digits to right of decimal for numbering
c              axis (note: if numdec = -1, integer portion of
c              number is plotted).
c
c Output arguments:
c -numdec    = on output:
c              minimum number of digits to the right of decimal
c              required for uniqueness.
c
      real*4          stic,etic,ticinc,sval,bval,eval,valinc
      integer         numdec
c
      real            trange, value
      integer         ntics, nnum, ntbn, ncount, maxlen
      integer         nch, lenstr, ndec, ndot, last, lastnz, ndig
      parameter       (maxlen=030)
      character*(maxlen) kform, cnum
      external        lenstr
c
c        Calculate the range of tics in units
c        Calculate the number of tics to be drawn on the axis
c        Calculate range of tics with numbers
c        Calculate number of tics with numbers
c        NTBN is number of tics between numbers
c
      trange = abs(etic - stic)
      ntics = (trange / ticinc) + 1.0001
      nnum = (abs(etic - bval) / valinc) + 1.0001
      ntbn = (valinc / ticinc) + .0001
c
      write (unit=kform, fmt=9110) numdec  ! accept user guess
 9110 format ('(f30.', i5, ')')
      ndec = -1
c
      do 120 ncount = 1, ntics, ntbn
         value = bval  +  float(ncount-1) * valinc
         write (unit=cnum, fmt=kform) value
         nch = lenstr (cnum, maxlen)
         ndot = index (cnum, '.')
         if (ndot .le. 0) then
            ndec = numdec
            go to 999
         endif
         last = lastnz (cnum, nch)
         ndig = last - ndot
         ndec = max (ndec, ndig)
```

```
 120  continue
      if (ndec .eq. 0) then
         ndec = -1
      endif
c
 999  continue
      numdec = ndec
      return
c
      end
```

```
      integer function lastnz (cnum, nch)
c
c This subroutine determines the first non-zero character in array cnum
c (from right to left).
c
c Input arguments:
c -cnum    = character array
c -nch     = number of characters in cnum
c
c Output value:
c -lastz = position of the first non-zero character (from right to left)
c
      integer        nch, jj, nn
      character*(1) cnum(1)
c
      do 110 jj = nch, 1, -1
         nn = jj
         if (cnum(nn) .ne. '0') go to 120
  110 continue
c
  120 continue
      lastnz = nn
c
      return
      end
```

Appendix B. Listing of the GETS's data file MORPHO.DAT which contains coefficients and exponents for the allometric functions

S = gill area cm**2 = s1 * W ** s2,

rho = # lamellae/ mm filament = rho1 * W ** rho2,

l = lamellar length cm = l1 * W ** l2,

where W is the fish's g live weight. The i-th record reports the species name, family, and life form. The (i+1) reports the reference for the data. The (i+2) record reports in order s1, s2, rho1, rho2, l1, and l2. Missing data are denoted by -999. The listing below is formatted as read by subroutine RDMORP (see Appendix A).

```
anabas testudineus                          \family              freshwater
hughes, g.m. 1972. respir.physiol. 14:1-25.
 5.560E+00 6.150E-01 3.650E+01-1.520E-01-9.990E+02-9.990E+02
blennius pholis                             \family              marine
de jager, s. and w.j. dekkers 1976. nether.j.zool. 25:276-308.
 1.156E+01 8.500E-01-9.990E+02-1.600E-01-9.990E+02-9.990E+02
channa punctata                             chanidae             freshwater
hakim et al. (1978). j.zool. 184:519-543.
 4.700E+00 5.920E-01-9.990E+02-9.990E+02-9.990E+02-9.990E+02
catostomus commersonii                      catostomidae         freshwater
de jager, s. and w.j. dekkers 1976. nether.j.zool. 25:276-308.
 7.980E+00 6.390E-01-9.990E+02-9.990E+02-9.990E+02-9.990E+02
catostomus commersonii                      catostomidae         freshwater
saunders, r.l. 1962. can.j.zool. 40:817-862.
 1.117E+01 5.870E-01 2.500E+01-1.070E-01-9.990E+02-9.990E+02
coryphaena hippurus                         coryphaenidae        marine
hughes, g.m. 1972. respir.physiol. 14:1-25.
 5.208E+01 7.130E-01 3.380E+01-3.600E-02-9.990E+02-9.990E+02
cyprinus carpio                             cyprinidae           freshwater
oikawa, s. and y. itazawa 1985. j.exp.biol 117:1-14.
 8.460E+00 7.940E-01 3.220E+01-7.900E-02-9.990E+02-9.990E+02
gambusia affinis                            poeciliidae          freshwater
murphy, p.g. and j.v. murphy 1971. bull.environ.contam. toxicol. 6:581-58
 2.330E+00 8.730E-01-9.990E+02-9.990E+02-9.990E+02-9.990E+02
ictalurus nebulosa                          ictaluriade          freshwater
de jager, s. and w.j. dekkers 1976. nether.j.zool. 25:276-308.
 2.650E+00 8.450E-01-9.990E+02-9.990E+02-9.990E+02-9.990E+02
ictalurus nebulosa                          ictaluriade          freshwater
saunders, r.l. 1962. can.j.zool. 40:817-862.
 4.980E+00 7.280E-01 1.580E+01-9.100E-02-9.990E+02-9.990E+02
katsuwonus pelamis                          scombridae           marine
muir, b.s. and g.m. hughes 1969. j.exp.biol. 51:271-285.
 5.675E+01 8.410E-01 5.900E+01-7.600E-02-9.990E+02-9.990E+02
lepidocephalichthys sp.                     \family              freshwater
singh, b.r., a.n. yadav, j.ojha, and j.s.d. munshi 1981. coepia 1981:224-
 4.940E+00 7.450E-01 4.500E+01-2.210E-01-9.990E+02-9.990E+02
lepisosteus sp.                             lepisosteidae        freshwater
landolt, j.c. and l.g. hill 1975. coepia 1975:470-475.
 3.940E+00 7.380E-01 3.880E+01-6.000E-02-9.990E+02-9.990E+02
micropterus dolomieu                        centrarchidae        freshwater
de jager, s. and w.j. dekkers 1976. nether.j.zool. 25:276-308.
 7.310E+00 8.200E-01 3.000E+01-6.200E-02-9.990E+02-9.990E+02
opsanus tau                                 batrachoididae       marine
hughes, g.m. and i.e. gray 1972. biol.bull. 143:150-161.
 5.600E+00 7.900E-01 1.600E+01-7.500E-02-9.990E+02-9.990E+02
raja clavata                                rajidae              marine
hughes, g.m., perry, s.f., piiper, j. 1986. j.exp.biol. 121:27-42
-9.990E+02 9.700E-01-9.990E+02-1.540E-01-9.990E+02-9.990E+02
```

saccobranchus fossilis \family marine
hughes, et al (1974); missing ref ********** !!!
1.860E+00 7.460E-01 3.160E+01-9.500E-02-9.990E+02-9.990E+02
salmo gairdneri salmonidae freshwater
de jager, s. and w.j. dekkers 1976. nether.j.zool. 25:276-308.
3.900E+00 9.000E-01-9.990E+02-9.990E+02-9.990E+02-9.990E+02
salmo gairdneri salmonidae freshwater
niimi, a.j. and morgan, s.l. 1980. j.fish.biol. 16:685-692.
1.840E+00 1.125E+00-9.990E+02-9.990E+02-9.990E+02-9.990E+02
salmo gairdneri salmonidae freshwater
hughes, g.m. 1984. fish physiology, vol xa. academic press.
3.140E+00 9.320E-01 2.750E+01-6.400E-02-9.990E+02-9.990E+02
scomber scomber scombridae marine
hughes, g.m. 1972. respir.physiol. 14:1-25.
4.420E+00 9.970E-01 2.710E+01 2.300E-02-9.990E+02-9.990E+02
scyliorhinus canicula scyliorhinidae marine
hughes, g.m. 1972. respir.physiol. 14:1-25.
2.620E+00 9.610E-01 1.720E+01-7.100E-02-9.990E+02-9.990E+02
scyliorhinus stellaris scyliorhinidae marine
hughes, g.m., perry, s.f., piiper, j. 1986. j.exp.biol. 121:27-42
-9.990E+02 7.790E-01-9.990E+02-1.670E-01-9.990E+02-9.990E+02
stizostedion vitreum percidae freshwater
niimi, a.j. and morgan, s.l. 1980. j.fish.biol. 16:685-692.
8.000E-01 1.129E+00-9.990E+02-9.990E+02-9.990E+02-9.990E+02
thunnus thynnus scombridae marine
muir, b.s. and g.m. hughes 1969. j.exp.biol. 51:271-285.
2.443E+01 9.010E-01 6.090E+01-8.900E-02-9.990E+02-9.990E+02
tinca tinca cyprinidae freshwater
de jager, s. and w.j. dekkers 1976. nether.j.zool. 25:276-308.
1.220E+01 6.570E-01-9.990E+02-9.990E+02-9.990E+02-9.990E+02
tinca tinca cyprinidae freshwater
hughes, g.m. 1972. respir.physiol. 14:1-25.
8.670E+00 6.980E-01 2.550E+01-3.000E-02-9.990E+02-9.990E+02
torpedo marmorata torpedinidae marine
hughes, g.m. 1978. j.exp.biol. 73:35-105.
1.180E+00 9.370E-01 3.420E+01-1.670E-01-9.990E+02-9.990E+02

Appendix C. Sample session of GETS. Dollar signs($) denote VAX system prompts and asterisks(*) denote EDT prompts. The following example analyzes laboratory bioaccumulation of 2,2',3,3'-tetrachlorobiphenyl by rainbow trout. See Oliver, B.G. and Niimi, A.J. 1985. Environ.Sci.Technol. 19:842-849.

```
$ edit/edt pcb2233.dat <return>
Input file does not exist
[EOB]
*insert <return>
        'spplab'          'Salmo gairdneri' <return>
        'famlab'          'salmonidae' <return>
        'liflab'          'freshwater' <return>
        'molwt'           292.0 <return>
        'logP'            5.8 <return>
        'wt'              200.0 <return>
        'plfish'          0.07 <return>
        'gamma'           -999.000 <return>
        'cfish'           0.0 <return>
        'cwater'          13e-6 <return>
        'adjust'          0.09 <return>
        'tend'            96 <return>
        <ctrl z>
[EOB]
*exit <return>
USER$DISK:[USER_ACCOUNT]PCB2233.DAT;1 12 lines

$ run gets <return>
Parameter file name: pcb2233.dat <return>
 USER:  input file: pcb2233.dat
 USER: output file: pcb2233.out
 USER:   plot file: pcb2233.plx
 USER:       molwt:    292.
 USER:        logp:    5.80
 USER:      fishid: salmo gairdneri
 USER:          wt:    200.      live weight, g
 USER:      plfish:   0.000E+00
 USER:       gamma:   -999.
 USER:      adjust:   9.000E-02
 USER:       cfish:   0.000E+00 ppm
 USER:      cwater:   1.300E-05 ppm
 USER:        tend:    96.0      Days
 MODEL: salmo gairdneri   s1:   2.82
 MODEL: salmo gairdneri   s2:  0.981
 MODEL: salmo gairdneri rho1:   27.5
 MODEL: salmo gairdneri rho2: -6.400E-02
    day      Wt          Bf          Cf
     0.     200.       0.000E+00   0.000E+00
     1.     216.       0.739       3.430E-03
     2.     217.        1.52       7.008E-03
     3.     218.        2.31       1.058E-02
     4.     219.        3.09       1.413E-02
     5.     220.        3.87       1.764E-02
     6.     220.        4.65       2.111E-02
     7.     221.        5.43       2.456E-02
     8.     222.        6.20       2.797E-02
     9.     222.        6.97       3.134E-02
    10.     223.        7.74       3.469E-02
    11.     224.        8.51       3.800E-02
    12.     225.        9.27       4.129E-02
```

13.	225.	10.0	4.454E-02
14.	226.	10.8	4.776E-02
15.	227.	11.5	5.095E-02
16.	227.	12.3	5.411E-02
17.	228.	13.1	5.724E-02
18.	229.	13.8	6.034E-02
19.	230.	14.6	6.341E-02
20.	230.	15.3	6.645E-02
21.	231.	16.0	6.947E-02
22.	232.	16.8	7.245E-02
23.	232.	17.5	7.541E-02
24.	233.	18.3	7.834E-02
25.	234.	19.0	8.124E-02
26.	235.	19.7	8.412E-02
27.	235.	20.5	8.696E-02
28.	236.	21.2	8.979E-02
29.	237.	21.9	9.258E-02
30.	238.	22.7	9.535E-02
31.	238.	23.4	9.810E-02
32.	239.	24.1	0.101
33.	240.	24.8	0.104
34.	241.	25.5	0.106
35.	241.	26.3	0.109
36.	242.	27.0	0.111
37.	243.	27.7	0.114
38.	244.	28.4	0.117
39.	244.	29.1	0.119
40.	245.	29.8	0.122
41.	246.	30.5	0.124
42.	247.	31.2	0.127
43.	247.	31.9	0.129
44.	248.	32.6	0.132
45.	249.	33.3	0.134
46.	250.	34.0	0.136
47.	250.	34.7	0.139
48.	251.	35.4	0.141
49.	252.	36.1	0.143
50.	253.	36.8	0.146
51.	253.	37.5	0.148
52.	254.	38.2	0.150
53.	255.	38.9	0.152
54.	256.	39.5	0.155
55.	256.	40.2	0.157
56.	257.	40.9	0.159
57.	258.	41.6	0.161
58.	259.	42.3	0.163
59.	260.	43.0	0.165
60.	260.	43.6	0.168
61.	261.	44.3	0.170
62.	262.	45.0	0.172
63.	263.	45.6	0.174
64.	263.	46.3	0.176
65.	264.	47.0	0.178
66.	265.	47.7	0.180
67.	266.	48.3	0.182
68.	267.	49.0	0.184
69.	267.	49.6	0.186
70.	268.	50.3	0.188
71.	269.	51.0	0.190
72.	270.	51.6	0.191

```
   73.     271.          52.3          0.193
   74.     271.          52.9          0.195
   75.     272.          53.6          0.197
   76.     273.          54.3          0.199
   77.     274.          54.9          0.201
   78.     275.          55.6          0.202
   79.     275.          56.2          0.204
   80.     276.          56.9          0.206
   81.     277.          57.5          0.208
   82.     278.          58.2          0.209
   83.     279.          58.8          0.211
   84.     279.          59.5          0.213
   85.     280.          60.1          0.215
   86.     281.          60.7          0.216
   87.     282.          61.4          0.218
   88.     283.          62.0          0.219
   89.     283.          62.7          0.221
   90.     284.          63.3          0.223
   91.     285.          63.9          0.224
   92.     286.          64.6          0.226
   93.     287.          65.2          0.227
   94.     288.          65.8          0.229
   95.     288.          66.5          0.231
   96.     289.          67.1          0.232

 MODEL:              k1:     277.
 MODEL:              k2:    5.879E-03
 MODEL:           gamma:    3.842E-03
FORTRAN STOP
$
```