Integrated Michaelis-Menten equation in DynaFit. 2. Application to 5α-ketosteroid reductase

BIOKIN TECHNICAL NOTE TN-2016-02

Petr Kuzmič

BioKin Ltd., Watertown, Massachusetts

Abstract

The DynaFit software package (http://www.biokin.com/dynafit/) allows the user to specify the theoretical model for an arbitrary bio/chemical mechanism by using a symbolic rather than mathematical notation. However, for convenience, DynaFit also contains a small number of built-in, predefined fitting models. This Technical Note describes a family of predefined algebraic fitting models that describe the time-course of enzyme reactions conforming to the classic Michaelis-Menten kinetic mechanism. An illustrative example includes previously published data on the substrate kinetics of α-ketosteroid reductase. It is shown that the least squares fit of a single enzymatic progress curve, observed at an optimally chosen substrate concentration, results in a K_M value that is exactly identical to the value obtained by using the much laborious and less precise initial rate method.

Key words:
enzyme progress curves; global fit; integrated Michaelis-Menten equation; ketosteroid reductase; Lambert omega function; substrate kinetics;

Contents

1. Introduction

Arguably the main advantage of software package DynaFit [1, 2] lies in that user can specify the mathematical model for any chemical, biochemical, or biophysical process by using the symbolic notation, rather than by using the often tedious mathematical formalism. For example, to specify the time-course of an enzyme reaction that follows the familiar Michaelis-Menten mechanism [3, p. 19], the user might input the following text:

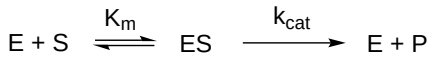
```
[task]
   data = progress
   ...
[mechanism]
   E + S <==> E.S    :    k1    k-1
   E.S ---> E + P    :    k2
[constants]
   k1 = ..., k-1 = ...
   k2 = ...
[concentrations]
   E = ...
   S = ...
[responses]
   P = ...
```

When presented with this symbolic input, the DynaFit software package automatically derives a system of first-order ordinary differential equation (ODEs) corresponding to the specified mechanism. The internally derived system of differential equations can then be used to either simulate artificial data or, more importantly, to fit any available experimental data with the aim of determining the microscopic rate constants k_1, k_{-1}, and k_2.

Aside form this ability to construct an arbitrary mathematical model "on the fly", the DynaFit software package also contains a certain number of built-in ("hard-wired") mathematical models encoded in a streamlined fashion, frequently using specialized the algebraic (as opposed

to differential-equation) formalism. One example of such built-in models is discussed in this Technical Note.

The catalytic mechanism we are concerned with is depicted in the scheme below, representing the *Michaelis-Menten mechanism*, where **E** is the enzyme, **S** is the substrate, **ES** is the Michaelis complex, and **P** is the reaction product.



The Michaelis-Menten mechanism occurs in enzyme kinetics investigations with such high frequency that it became convenient to embed an algebraic model for the reaction progress directly into the software code. Thus, in order specify the reaction time course of any enzyme reaction following the Michaelis-Menten mechanism, the DynaFit user had the opportunity to provide much simplified input code, similar to the fragment below:

```
[task]
   data = generic
   code = built-in
   ...
[equation]
   MichaelisMentenProgressKmKcat
[parameters]
   Km = ...
   kcat = ...
   Eo = ...
   So = ...
   rP = ...
```

In the above encoding, `Eo` represents the total or analytic enzyme concentration in appropriate units; `So` represents the substrate concentration in identical units; and `rP` represents the specific molar response coefficient of the reaction product.

In this Technical Note we illustrate the utility of this particular fitting model, "hard-coded" into DynaFit, but analyzing previously published [4] experimental data representing the substrate kinetics of 5α-ketosteroid reductase. It is shown that by applying the build-in algebraic model to a *single enzymatic progress curve*, we obtained the same results as those that were previously [5] obtained from *multiple progress curves*, using the standard initial rate method.

Thus, the built-in algebraic model for the time course of any enzyme reaction following the Michaelis-Menten mechanism can potentially lead to significant savings in time, effort, and material costs.

2. Methods

2.1. Mathematical models

This section presents a family of two closely related mathematical models predefined in the software package DynaFit [1, 2]. Both models can be used to fit the time course of enzyme reactions following the Michaelis-Menten mechanism, with the aim of determining the kinetic parameters K_M, k_{cat}, or $k_S \equiv k_{cat}/K_M$. Along with the mathematical definition, each model below is described also in terms of the encoding of parameter names expected by the software.

2.1.1. Variant 1: K_M and k_{cat}

The time course of an enzyme reaction following the Michaelis-Menten kinetic mechanism [3, p. 19] can be conveniently described by Eqn (1) [6–8], where F is some appropriate experimental variable, such as for example fluorescence, recorded at the reaction time t; F_0 is the experimental signal observed at $t = 0$ (i.e., baseline offset – essentially a property of the instrument); $[S]_0$ is the initial substrate concentration; K_M is the Michaelis constant; and r_P is the specific molar response coefficient of the reaction product; $[E]_0$ is the concentration of the enzyme active sites; t is the reaction time; and k_{cat} is the turnover number. The symbol ω represents the value of the *Lambert omega* function, also referred to as Lambert W function [6–8]. Eqn (2) represents the *instantaneous observed reaction rate*, i.e., the first derivative with respect to time t of the physical variable F being monitored.

$$F = F_0 + r_P \left\{ [S]_0 - K_M \, \omega \left[\frac{[S]_0}{K_M} \, \exp\left(\frac{[S]_0}{K_M} - \frac{k_{cat}}{K_M} \, [E]_0 \, t \right) \right] \right\} \tag{1}$$

$$\frac{dF}{dt} = r_P \, k_{cat} \, [E]_0 \frac{\alpha}{1+\alpha} \tag{2}$$

The equation name built into DynaFit is `MichaelisMentenProgressKmKcat`. The names of model parameters expected by DynaFit encoding are shown in the table below:

parameter	symbol	encoding
reaction time (independent variable)	t	`t`
enzyme concentration	$[E]_0$	`Eo`
substrate concentration	$[S]_0$	`So`
turnover number	k_{cat}	`kcat`
Michaelis constant	K_M	`Km`
molar response coefficient of product	r_P	`rP`
baseline offset	F_0	`Fo`

2.1.2. Variant 2: K_M and k_S

An alternate, and algebraically equivalent way of expressing the integrated rate law is given by the recently proposed [9] Eqn (3). In this case, the integrated rate equation does not contain k_{cat} as a model parameter, but rather k_S (and K_M, as before). Note that k_S is the *specificity number* defined as k_{cat}/K_M. Note that k_S has the dimension of a second-order (bimolecular association)

rate constant. The derivative of the model function with respect to time, i.e., the instantaneous observed reaction rate, is defined by Eqn (4).

$$F = F_0 + r_P \left\{ [S]_0 - K_M \, \omega \left[\frac{[S]_0}{K_M} \, \exp\left(\frac{[S]_0}{K_M} - k_S \, [E]_0 \, t \right) \right] \right\} \tag{3}$$

$$\frac{dF}{dt} = r_P K_M \, k_S \, [E]_0 \frac{\beta}{1+\beta} \tag{4}$$

The equation name built into DynaFit is `MichaelisMentenProgressKmKs`. The names of model parameters expected by DynaFit encoding are shown in the table below:

parameter	symbol	encoding
reaction time (independent variable)	t	
enzyme concentration	$[E]_0$	`Eo`
substrate concentration	$[S]_0$	`So`
specificity number	$k_S \equiv k_{cat}/K_M$	`kS`
Michaelis constant	K_M	`Km`
molar response coefficient of product	r_P	`rP`
baseline offset	F_0	`Fo`

The alternate use of Eqn (1) or Eqn (3) depends on which combination of steady-state parameters (K_M or k_{cat} along with k_S) is of greater interest to the investigator.

2.2. ANSI C implementation

A number of more or less robust computer codes are available on the Internet, all of which implement the Lambert Omega function, using a variety of alternate algorithms. DynaFit contains an implementation closely derived from the ANSI C code available from ref. [10] and listed in Appendix B.

The author of the original code is Dr. Keith Briggs, a senior mathematician at BT Research (British Telecom) and an Alan Tayler Visiting Lecturer in the Mathematical Institute of the University of Oxford, United Kingdom.

2.3. DynaFit scripting

In order to either fit or simulate the progress of enzyme reactions according to Eqn (1), the DynaFit input script file must contain at least the following encoding:

```
[task]
   data = generic
   code = built-in
   ...
[equation]
   MichaelisMentenProgressKmKcat
[parameters]
```

```
   Eo = ...
   So = ...
   kcat = ...
   Km = ...
   rP = ...
   Fo = ...
[data]
   variable t
   ...
[end]
```

In order to either fit or simulate the progress of enzyme reactions according to Eqn (3), the DynaFit input script file must contain at least the following encoding:

```
[task]
   data = generic
   code = built-in
   ...
[equation]
   MichaelisMentenProgressKmKs
[parameters]
   Eo = ...
   So = ...
   kS = ...
   Km = ...
   rP = ...
   Fo = ...
[data]
   variable t
   ...
[end]
```

3. Results and discussion

3.1. Raw experimental data

The enzyme 5α-ketosteroid reductase (active site concentration 50 pM) was used to catalyzes the conversion of testosterone to dihydrotestosterone (initial concentration 31 nM) in the presence of NADPH. The reaction was quenched at various times (t = 2, 4, 6, ... 60 minutes) and analyzed by HPLC with radiometric detection. The raw experimental data taken from ref. [4], Figure 2, are listed in *Table 1*. The originally published data (percent product formation) were converted to absolute product concentrations (nM).

3.2. Least squares fit to Eqn (3)

The experimental data listed in section 3.1 were fitted to Eqn (3) by using the DynaFit script listed in Appendix A.1. The results are summarized graphically in *Figure 1* and numerically in *Table 2*.

t, sec	[P], nM
120	1.99
240	3.86
360	6.05
480	7.90
600	9.53
720	11.77
840	13.22
960	14.70
1080	16.87
1200	17.80
1320	19.65
1440	20.35
1560	21.88
1680	22.84
1800	23.53
2040	25.09
2280	26.67
2520	27.66
2760	28.63
3000	29.12
3240	29.42
3600	29.91

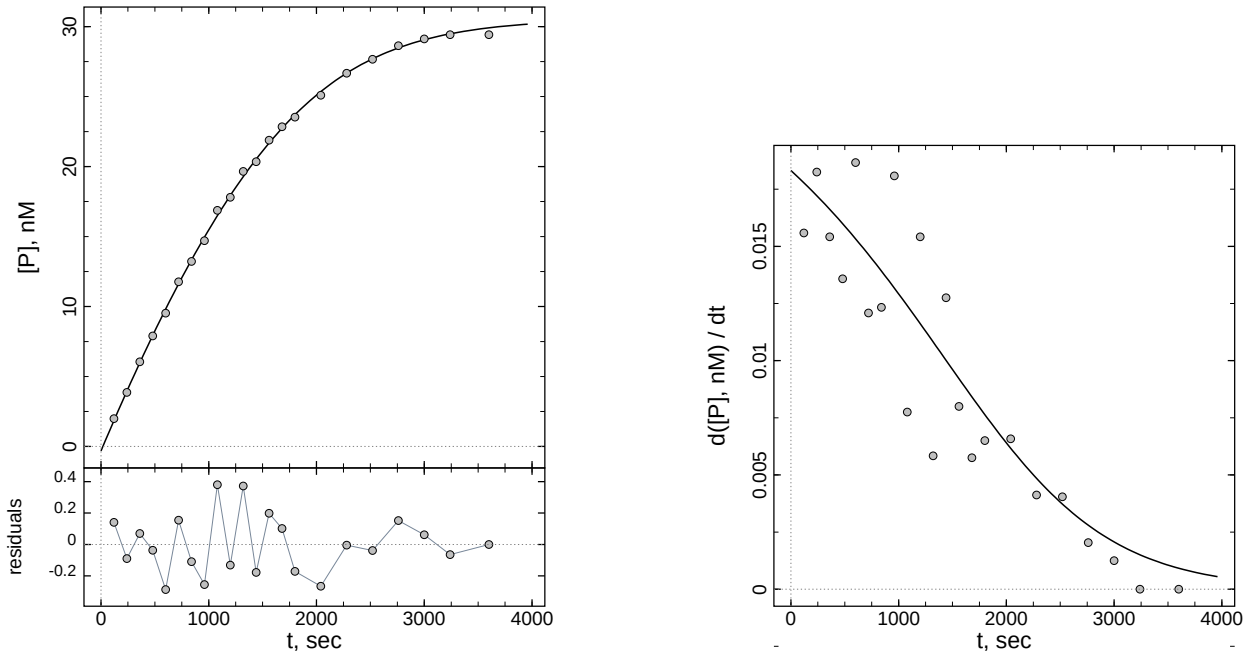
Table 1: Time course of testosterone as substrate ($[S]_0$ = 30 nM) conversion to dihydrotestosterone as product ([P]) catalyzed by 5α-ketosteroid reductase. Experimental data form ref. [4], Figure 2.

3.3. Comparison with published results

Faller *et al.* [5] determined the K_M value for human prostatic 5-alpha reductase, with respect to testosterone as substrate, by using the conventional method, as follows.[1]

Multiple enzyme reactions were initiated by the addition of enzyme (25 pg/mL final concentration) to a buffered solution containing 0.4 mM NADPH and [^{3}H]testosterone *varied* between 5 nM and 120 nM (eight separate experiments). Aliquots were removed after 5, 15, 25, and 35 min of incubation. Reactions were stopped by the addition of 2 mL of diethyl ether. The organic phase was then transferred into a new tube and evaporated to dryness in a water bath at 40 degrees for 30 min. Samples were resuspended by addition of 50 pL of a solution of ethanol containing cold testosterone and dihydrotestosterone (2 mg/mL each) as markers and applied to a silica-impregnated glass fiber sheet. Metabolites were separated by chromatography with dichloromethane/diethyl ether (9:l) as the running solvent. Steroids were located by spraying the dried sheets with a solution of pentahydroxyflavone and visualized under illumination at 366 nm. Spots were then cut out and transferred into polyethylene vials containing 6 mL of Irgascint A300 for scintillation counting, to determine the amount of dihydrotestosterone formed at each

[1] Because the method is quite laborious, it deserves to be described in full, in order to give an accurate impression of the tediousness involved and therefore also an impression of the savings in labor and material that can be achieved by the present method.



(a) **(b)**

Figure 1: Time course of testosterone as substrate ($[S]_0$ = 30 nM) conversion to dihydrotestosterone as product ([P]) catalyzed by 5α-ketosteroid reductase. **(a)**. Symbols: raw experimental data. Smooth curve: best-fit model according to Eqn (1). **(b)**. Symbols: rate-transformed experimental data. Smooth curve: best-fit instantaneous rate curve according to Eqn (2).

time point. The raw data (incubation time vs. dihydrotestosterone formed) were fit to the straight line model. The slopes derived from each linear fit were treated as the initial rates and were fit to the Michaelis-Menten equation. The best-fit value of the Michaelis constant was K_M = (20 ± 3) nM, see the *Figure 2*.

Thus, by using the classic initial rate method and *eight* separate experiments (i.e., substrate concentrations ranging from $[S]_0$ = 5 nM to 120 nM), Faller *et al.* [5] obtained K_M = (20 ± 3) nM. In contrast, by using the present method based on Eqn (1), we used a *single* experiment at $[S]_0$ = 30 nM to arrive at K_M = (21 ± 3) nM, essentially an identical value.

In summary, the results obtained by using standard method, based on multiple initial rates, agrees nearly perfectly with the single progress curve method (K_M = 20 vs. 21 nM). Importantly, the single progress curve method required a significantly lower expenditure in time, labor, and materials.

4. Summary and conclusions

1. Using only a *single* progress curve we can obtain a well-defined estimate of K_M and k_{cat}.

#	parameter	initial	final ± std.err.	cv,%	note
1	k_S, $nM^{-1}s^{-1}$	1	0.030 ± 0.003	8.7	
2	K_m, nM	10	20.6 ± 3.4	16.5	
3	r_P, –	1	0.996 ± 0.013	1.3	
4	F_0, nM	-0.1	-0.32 ± 0.19	59.4	

Table 2: Results of least-squares fit of experimental data in *Figure 1* to Eqn (3).

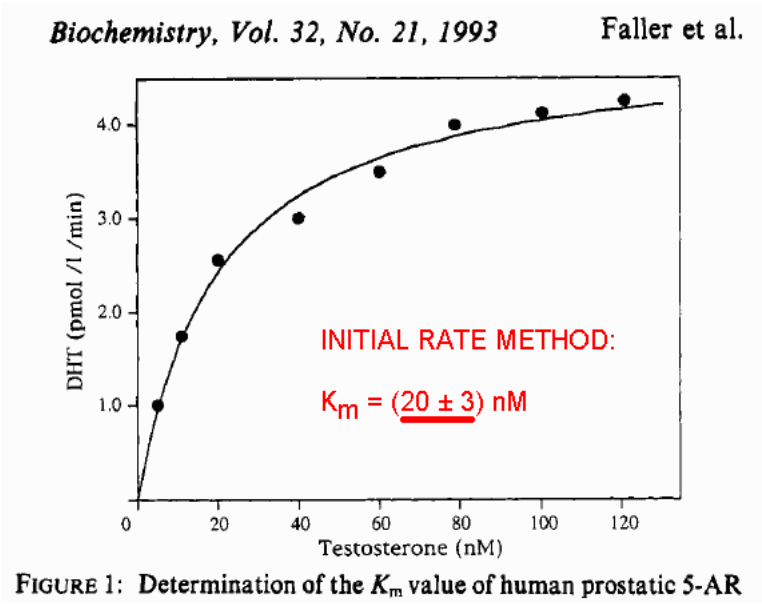


Figure 2: Results of K_M determination by the standard initial rate method [5]. The best-fit value of the Michaelis constant was $K_M = (20 \pm 3)$ nM.

2. The best-fit values are indistinguishable from the results obtained by the standard initial rate method.
3. The initial rate method requires multiple experiments, as opposed to just one well-designed experiment.
4. Thus the data-analytic method presented here potentially leads to significant savings in time and materials.
5. The requisite fitting model is built into the software package DynaFit [2] (equation code `MichaelisMentenProgressKmKcat`).

All experimental data utilized in this report, plus all DynaFit input (script) files that were used to produce the report, are available for download from `http://www.biokin.com/TN/2016/02`.

Acknowledgements

The author is indebted to Dr. Mey Ling Reytor González for stimulating discussions and for helpful suggestions to improve this manuscript.

Bibliographic Information

How to Cite this Publication:

Kuzmič, P. (2016) *Integrated Michaelis-Menten equation in DynaFit. 2. Application to 5α-ketosteroid reductase*, BioKin Technical Note TN-2016-02, BioKin Ltd., Watertown MA, [Online] www.biokin.com/TN/2016/02

References

[1] P. Kuzmič, Program DYNAFIT for the analysis of enzyme kinetic data: Application to HIV proteinase, Anal. Biochem. 237 (1996) 260–273.

[2] P. Kuzmič, DynaFit - A software package for enzymology, Meth. Enzymol. 467 (2009) 247–280.

[3] I. H. Segel, Enzyme Kinetics, Wiley, New York, 1975.

[4] M. L. Moss, P. Kuzmič, J. D. Stuart, G. Tian, A. G. Peranteau, S. V. Frye, S. H. Kadwell, T. A. Kost, L. K. Overton, I. R. Patel, Inhibition of human steroid 5-alpha reductases type I and II by 6-aza-steroids: Structural determinants of one-step vs. two-step mechanism, Biochemistry 35 (1996) 3457–3464.

[5] B. Faller, D. Farley, H. Nick, Finasteride: a slow-binding 5alpha-reductase inhibitor, Biochemistry 32 (1993) 5705–5710.

[6] S. Schnell, C. Mendoza, Closed form solution for time-dependent enzyme kinetics, J. theor. Biol. 187 (1997) 207–212.

[7] C. T. Goudar, J. R. Sonnad, R. G. Duggleby, Parameter estimation using a direct solution of the integrated Michaelis-Menten equation, Biochim. Biophys. Acta 1429 (1999) 377–383.

[8] M. Goličnik, On the Lambert W function and its utility in biochemical kinetics, Biochem. Eng. J. 63 (2012) 116–123.

[9] M. L. Reytor-González, S. Cornell-Kennon, E. Schaffer, P. Kuzmič, An algebraic model for the determination of Michaelis-Menten kinetic parameters by global nonlinear fit of enzymatic progress curves, Anal. Biochem. (2016) submitted.

[10] K. Briggs, Lambert W function, http://bit.ly/2eUfRO9, accessed 23 Sep 2016, [Online] (2001).

Appendix

A. DynaFit scripts

A.1. Least squares fit to Eqn (3)

This DynaFit [2] script performs the fit or experimental data described in section 3.1 to the built-in Eqn (3). The data file data.csv mentioned in the script below is a comma-separated value (CSV) file exported from Microsoft Excel. The first column holds the reaction time in seconds; the second column holds the concentration of the reaction product (dihydrotestosterone) in nanomolar units.

```
[task]
   task = fit
   data = generic
   code = built-in
[equation]
   MichaelisMentenProgressKmKs
[parameters]
   Eo = 0.05
   So = 31
   kS = 1 ?
   Km = 10 ?
   rP = 1 ?
   Fo = -0.1 ?
[data]
   variable t
   directory ./proj/IMM/5alpha-reductase/data
   sheet data.csv
   column 2
[output]
   directory ./proj/IMM/5alpha-reductase/output/fit
[settings]
{Output}
   XAxisLabel = t, sec
   YAxisLabel = [P], nM
   PlotRatesData = y
   WriteTeX = y
[end]
```

B. C-Language implementation of Lambert Omega function

This Appendix captures the complete ANSI C language source code published at the URL listed below, last accessed October 28, 2016. The DynaFit [2] implementation of the Lambert W function is a minor modification of the original source code listed below.

http://keithbriggs.info/software/LambertW.c

```
/* Lambert W function.
   Was ~/C/LambertW.c written K M Briggs Keith dot Briggs at bt dot com 97 May 21.
   Revised KMB 97 Nov 20; 98 Feb 11, Nov 24, Dec 28; 99 Jan 13; 00 Feb 23; 01 Apr 09

   Computes Lambert W function, principal branch.
   See LambertW1.c for -1 branch.

   Returned value W(z) satisfies W(z)*exp(W(z))=z
   test data...
      W(1)= 0.5671432904097838730
      W(2)= 0.8526055020137254914
      W(20)=2.2050032780240599705
   To solve (a+b*R)*exp(-c*R)-d=0 for R, use
   R=-(b*W(-exp(-a*c/b)/b*d*c)+a*c)/b/c

   Test:
     gcc -DTESTW LambertW.c -o LambertW -lm && LambertW
   Library:
     gcc -O3 -c LambertW.c
*/

#include <math.h>
#include <stdio.h>

double LambertW(const double z);
const int dbgW=0;

double LambertW(const double z) {
  int i;
  const double eps=4.0e-16, em1=0.3678794411714423215955237701614608;
  double p,e,t,w;
  if (dbgW) fprintf(stderr,"LambertW: z=%g\n",z);
  if (z<-em1 || isinf(z) || isnan(z)) {
    fprintf(stderr,"LambertW: bad argument %g, exiting.\n",z); exit(1);
  }
  if (0.0==z) return 0.0;
  if (z<-em1+1e-4) { // series near -em1 in sqrt(q)
    double q=z+em1,r=sqrt(q),q2=q*q,q3=q2*q;
    return
     -1.0
     +2.331643981597124203363536062168*r
     -1.812187885639363490240191647568*q
     +1.936631114492359755363277457668*r*q
     -2.353551201881614516821543561516*q2
     +3.066858901050631912893148922704*r*q2
     -4.175335600258177138854984177460*q3
     +5.858023729874774148815053846119*r*q3
     -8.401032217523977370984161688514*q3*q;  // error approx 1e-16
  }
  /* initial approx for iteration... */
  if (z<1.0) { /* series near 0 */
```

```
    p=sqrt(2.0*(2.718281828459045235360287471352662 5*z+1.0));
    w=-1.0+p*(1.0+p*(-0.333333333333333333 3+p*0.152777777777777777 7777));
  } else
    w=log(z); /* asymptotic */
  if (z>3.0) w-=log(w); /* useful? */
  for (i=0; i<10; i++) { /* Halley iteration */
    e=exp(w);
    t=w*e-z;
    p=w+1.0;
    t/=e*p-0.5*(p+1.0)*t/p;
    w-=t;
    if (fabs(t)<eps*(1.0+fabs(w))) return w; /* rel-abs error */
  }
  /* should never get here */
  fprintf(stderr,"LambertW: No convergence at z=%g, exiting.\n",z);
  exit(1);
}

#ifdef TESTW
/* test program...  */
int main() {
  int i;
  double z,w,err;
  for (i=0; i<100; i++) {
    z=i/100.0-0.3678794411714423215955; w=LambertW(z);
    err=exp(w)-z/w;
    printf("W(%8.4f)=%22.16f, check: exp(W(z))-z/W(z)=%e\n",z,w,err);
  }
  for (i=0; i<100; i++) {
    z=i/1.0e-1-0.3; w=LambertW(z);
    err=exp(w)-z/w;
    printf("W(%8.4f)=%22.16f, check: exp(W(z))-z/W(z)=%e\n",z,w,err);
  }
  return 0;
}
#endif

#ifdef INTW
int main() {
  int i,n=1000;
  double w,z,s=0,err;
  for (i=1; i<=n; i++) {
    z=i/(double)n;
    w=LambertW(1/z)/(1+z);
    s+=w;
    printf("%8.4f %8.4f\n",z,w);
  }
  fprintf(stderr,"%8.4f\n",exp(s/n/log(2)));
  return 0;
}
#endif
```