ARMY RESEARCH LABORATORY

Calculating Drag Coefficients for Spheres and Other Shapes Using C++

by Robert J. Yager

ARL-TN-612 **June 2014**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-TN-612 **June 2014**

Calculating Drag Coefficients for Spheres and Other Shapes Using C++

Robert J. Yager
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
June 2014	Final	June 2013–August 2013

4. TITLE AND SUBTITLE
Calculating Drag Coefficients for Spheres and Other Shapes Using C++

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Robert J. Yager

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-WML-A
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-612

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report presents a set of functions, written in C++, that can be used to estimate fluid drag coefficients. The code does not perform estimations based on first-principle calculations. Rather, it relies on user-supplied tables that relate drag coefficients to Mach and Reynolds numbers. Bilinear interpolations are used to estimate drag coefficients between tabulated values. Two sample user-input tables, both relating to spheres, are provided.

15. SUBJECT TERMS
drag coefficient, sphere, fluid, atmosphere, Mach, Reynolds

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Robert J. Yager
Unclassified	Unclassified	Unclassified	UU	24	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6689

Contents

List of Figures

Acknowledgments

I would like to thank Mr. Benjamin Flanders of the U.S. Army Research Laboratory's Weapons and Materials Research Directorate, who provided editorial recommendations that improved the quality of this report.

1. Introduction

This report presents a set of functions, written in C++, that can be used to estimate fluid drag coefficients. The code does not perform estimations based on first-principle calculations. Rather, it relies on user-supplied tables that relate drag coefficients to Mach and Reynolds numbers. Bilinear interpolations are used to estimate drag coefficients between tabulated values. Two sample user-input tables, both relating to spheres, are provided.

2. Background

Drag, the force that acts on an object in a manner that opposes the motion of the object through a fluid, can be calculated using the equation

$$F = \frac{1}{2}\rho A C_D v^2 \tag{1}$$

where ρ is the density of the fluid, A is the cross-sectional area of the object, v is the speed of the object relative to the fluid, and C_D is the object's drag coefficient. For relatively slow speeds, C_D is commonly assumed to be a function solely of Reynolds number.

Reynolds number is defined by the equation

$$R_E = \frac{vL}{\eta} \tag{2}$$

where L is a characteristic linear dimension (e.g., diameter for spheres) and η is the fluid's kinematic viscosity.

Kinematic viscosity is defined by the equation

$$\eta = \frac{\mu}{\rho} \tag{3}$$

where μ is the fluid's dynamic viscosity and ρ is the fluid's density.

Figure 1 presents a graph of C_D as a function of R_E for a sphere. The graph was created by using digitizing software (*1*) to generate tabulated data based on a figure given by Munson et al. (*2*).

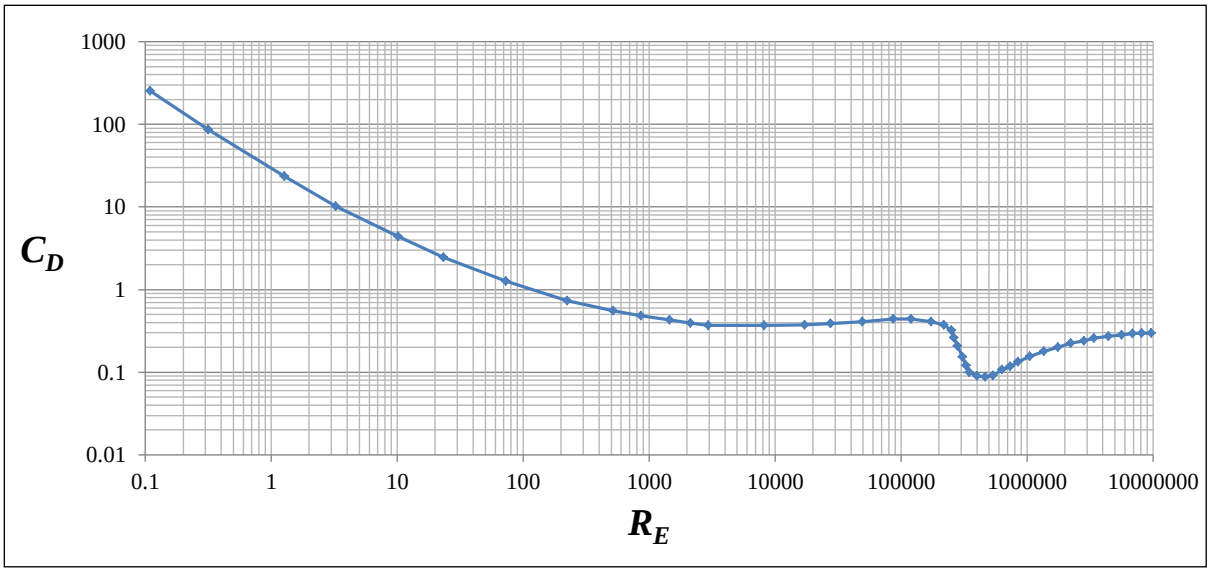


Figure 1. Drag coefficient as a function of Reynolds number for slow-moving spheres.

The relationship shown in figure 1 applies to incompressible fluid flow. If the speed of the object relative to the fluid becomes large enough, compressibility effects become important. According to Munson et al., "experimental data obtained from studies of regular solids indicate that the drag coefficient is a strong function of Mach number, especially in the low supersonic and subsonic Mach regimes."

Mach number is defined by the equation

$$M = \frac{v}{c} \tag{4}$$

where c is the speed of sound in the fluid.

Figure 2 presents a graph for C_D as a function of both R_E and M for spheres. The graph was created by using digitizing software to generate tabulated data based on a figure given by Bailey and Hiatt (*3*).

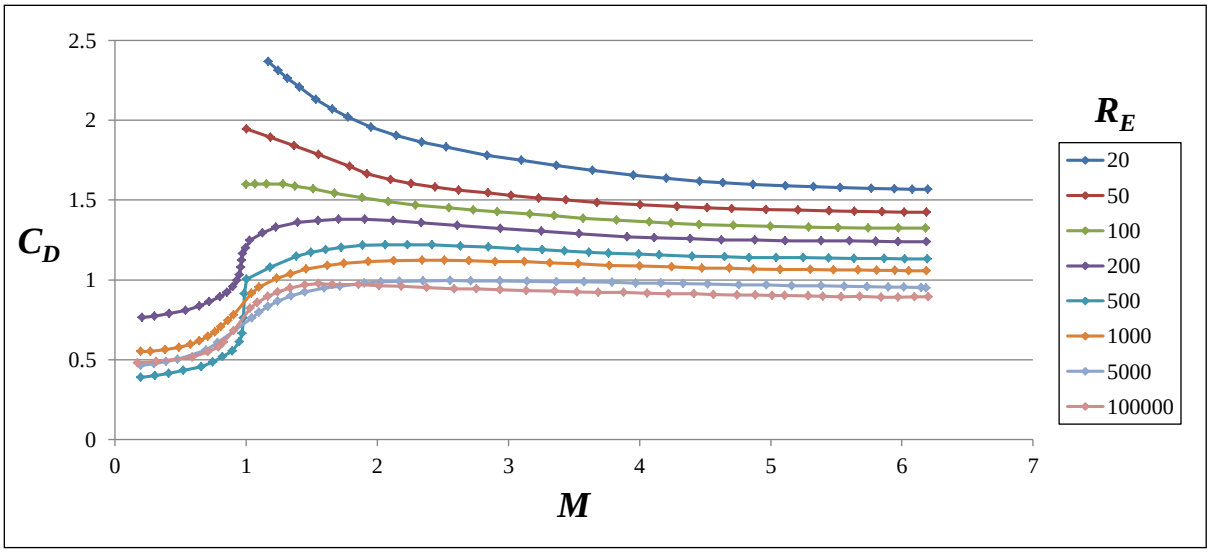


Figure 2. Drag coefficient as a function of both Reynolds number and Mach number.

For cases where the fluid is the Earth's atmosphere, tabulated values for η, μ, ρ, and c can be found in *U.S. Standard Atmosphere, 1976* (*4*). Sea-level values are presented in figure 3.

Pressure:	P_0=101325 N/m^2
Density:	ρ_0=1.2250 kg/m^3
Temperature:	T_0=288.15 K
Dynamic viscosity:	μ_0=1.7894 x 10^{-5} kg/(m·s)
Kinematic viscosity:	η_0=1.4607 x 10^{-5} m^2/s
Speed of sound:	c_0=340.29 m/s

Figure 3. Sea-level values for atmospheric parameters.

Values for atmospheric pressure, density, temperature, and speed of sound can also be calculated using the yAtmosphere namespace (*5*). Kinematic viscosity can be calculated using equation 3. According to *U.S. Standard Atmosphere, 1976*, atmospheric dynamic viscosity can be calculated using the equation

$$\mu = \frac{\beta T^{3/2}}{T + S} \tag{5}$$

where β is a constant equal to 1.458×10^{-6} kg/(s $\cdot$ m $\cdot$ K$^{1/2}$) and S is Sutherland's constant, which is equal to 110.4 K. Equation 5 is inaccurate for altitudes greater than 86,000 m. Equation 5 is also inaccurate for very high temperatures, so its application to artificially elevated atmospheric temperatures (such as result from explosions) may be limited.

3. Storing Drag Versus Mach and Reynolds Numbers – DMRTABLE Structs

DMRTABLE structs can be used to store drag coefficients as functions of Mach and Reynolds numbers. The tables are assumed to be in a form where a two-dimensional array of drag coefficients is associated with a single set of Mach numbers and a single set of Reynolds numbers, as shown in figure 4.

$$\begin{bmatrix} & \begin{bmatrix} R_{E,0} & R_{E,1} & R_{E,2} & \cdots & R_{E,n-2} & R_{E,n-1} \end{bmatrix} \\ \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{m-2} \\ M_{m-1} \end{bmatrix} & \begin{bmatrix} C_{D,(0,0)} & C_{D,(0,1)} & C_{D,(0,2)} & \cdots & C_{D,(0,n-2)} & C_{D,(0,n-1)} \\ C_{D,(1,0)} & C_{D,(1,1)} & & & & \\ C_{D,(2,0)} & & & & & \\ \vdots & & & \ddots & & \\ C_{D,(m-2,0)} & & & & & \\ C_{D,(m-1,0)} & & & & & C_{D,(m-1,n-1)} \end{bmatrix} \end{bmatrix}$$

Figure 4. Tabulated data for C_D as a function of M and R_E.

3.1 DMRTABLE Code

```
struct DMRTABLE{//<=======TABULATED DRAG AS A FUNCTION OF MACH AND REYNOLDS #s
  std::vector<double>M;//<---------MACH NUMBERS (UNIQUE & IN INCREASING ORDER)
  std::vector<double>Re;//<----REYNOLDS NUMBERS (UNIQUE & IN INCREASING ORDER)
  std::vector<std::vector<double> >Cd;//<-----DRAG COEF. (M[i],Re[j],Cd[i][j])
};//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
```

3.2 DMRTABLE Parameters

M — **M** stores a set of Mach numbers. **M** values should be unique and increasing (i.e., **M**[i]>**M**[i-1]).

Re	**Re** stores a set of Reynolds numbers. **Re** values should be unique and increasing (i.e., **Re**[j]>**Re**[j-1]).
Cd	**Cd** stores a set of drag coefficients such that **Cd**[i][j] corresponds to **M**[i] and **Re**[j].

4. Interpolating DRMTABLE Structs – the DragCoefficient() Function

The DragCoefficient() function can be used to calculate C_D given values for M, L, c, η, and a DMRTABLE. A bilinear interpolation is used to calculate drag coefficients between tabulated values.

4.1 DragCoefficient() Code

```
inline double DragCoefficient(//<CALCULATES A DRAG COEFFICIENT FROM A DMRTABLE
    double M,//<-----------------------------------------------------MACH NUMBER
    double L,//<-------------CHARACTERISTIC LENGTH (METERS FOR DEFAULT VALUES)
    DMRTABLE&T,//<-----------------------------------------------A DRMTABLE
    double c=340.29,//<----LOCAL SPEED OF SOUND (DEFAULT - SEA-LEVEL ATM. m/s)
    double eta=1.4607E-5){//<-------KINIMATIC VISCOSITY (SEA-LEVEL ATM. m^2/s)
  double x=M,y=M*c*L/eta;
  const std::vector<double>&X=T.M,&Y=T.Re;
  const std::vector<std::vector<double> >&Z=T.Cd;
  int i;/*<-*/for(i=1;i<(int)X.size()-1;++i)if(x<X[i])break;// X[i]<=x<=X[i+1]
  int j;/*<-*/for(j=1;j<(int)Y.size()-1;++j)if(y<Y[j])break;// Y[j]<=y<=Y[j+1]
  double Za=(y-Y[j-1])/(Y[j]-Y[j-1])*(Z[i-1][j]-Z[i-1][j-1])+Z[i-1][j-1];
  double Zb=(y-Y[j-1])/(Y[j]-Y[j-1])*(Z[i][j]-Z[i][j-1])+Z[i][j-1];
  return (x-X[i-1])/(X[i]-X[i-1])*(Zb-Za)+Za;
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
```

4.2 DragCoefficient() Parameters

M	**M** specifies the Mach number, *M*, of an object moving through a fluid.
L	**L** specifies an object's characteristic length, *L*. For the default values of **c** and **eta**, the units for **L** should be meters.
T	**T** specifies a DMRTABLE struct.
c	**c** specifies the speed of sound, *c*, in the fluid. The default value is 340.29 m/s, the sea-level value given in *U.S. Standard Atmosphere, 1976*.
eta	**eta** specifies the kinematic viscosity, η, of the fluid. The default value is 1.4607×10^{-5} m^2/s, the sea-level value given in *U.S. Standard Atmosphere, 1976*.

4.3 DragCoefficient() Return Value

The DragCoefficient() function returns a drag coefficient, C_D, that has been calculated based on the given input conditions.

---

5. Creating Drag Coefficient Versus Mach Tables – the DragTable() Function

---

The DragTable() function can be used to create a drag coefficient versus Mach number table for a given characteristic length and DMRTABLE struct. Output tabulated Mach numbers are drawn from the input DMRTABLE struct, with possible exceptions for the minimum and maximum values.

Let $M_{\min}$ and $M_{\max}$ represent the minimum and maximum M values associated with a particular DMRTABLE, respectively. Furthermore, let $R_{E,\min}$ and $R_{E,\max}$ represent minimum and maximum R_E values. Equations 2 and 4 can be used to write R_E as a function of M:

$$R_E(M) = \frac{McL}{\eta} \tag{6}$$

The minimum tabulated Mach number is chosen to be the maximum between $M_{\min}$ and M such that $R_E(M) = R_{E,\min}$. Similarly, the maximum output tabulated Mach number is chosen to be the minimum between $M_{\max}$ and M such that $R_E(M) = R_{E,\max}$. The result is that the range of the tabulated Mach values is limited to values for which C_D values can be estimated without extrapolation. This is shown graphically in figure 5 as the intersection between the blue diagonal line and the shaded region.

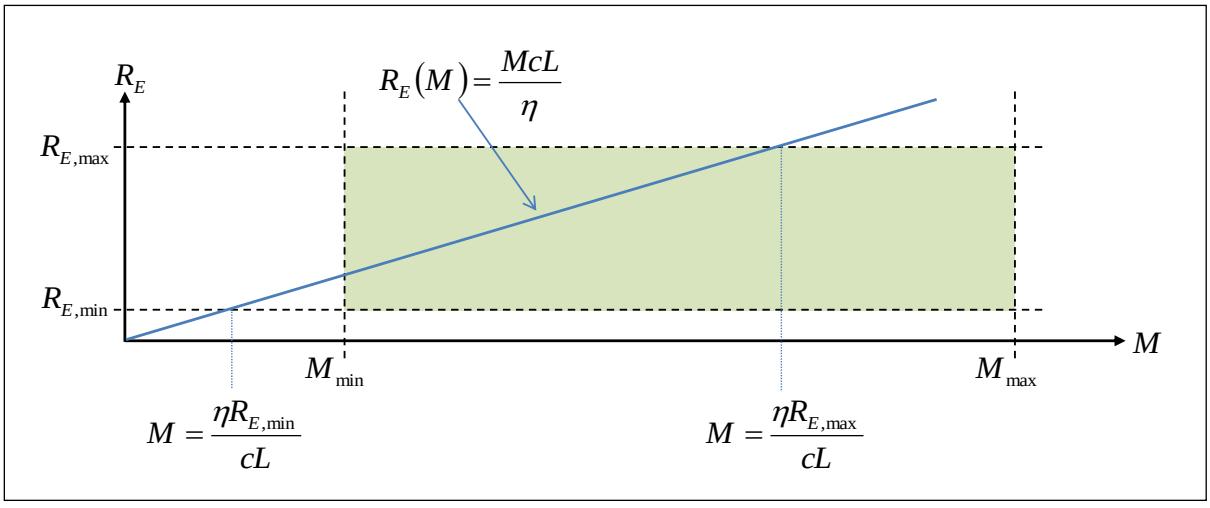


Figure 5. Reynolds number as a function of Mach number.

5.1 DragTable() Code

```
inline std::vector<std::vector<double> >DragTable(//<=CREATES A Cd VS. M TABLE
    double L,//<-------------CHARACTERISTIC LENGTH (METERS FOR DEFAULT VALUES)
    DMRTABLE&T,//<-----------------------------------------------A DRMTABLE
    double c=340.29,//<-SPEED OF SOUND IN FLUID (DEFAULT - SEA-LEVEL ATM. m/s)
    double eta=1.4607E-5){//<-------KINIMATIC VISCOSITY (SEA-LEVEL ATM. m^2/s)
  std::vector<std::vector<double> >X(2,std::vector<double>(0));
  double M_min=eta*T.Re[0]/c/L,M_max=eta*T.Re.back()/c/L;
  if(M_min>T.M.back()||M_max<T.M[0])return X;
  X[0].push_back(M_min>T.M[0]?M_min:T.M[0]);
  for(int j=1,n=T.M.size()-1;j<n;++j)
    if(T.M[j]>M_min&&T.M[j]<M_max)X[0].push_back(T.M[j]);
  X[0].push_back(M_max<T.M.back()?M_max:T.M.back());
  for(int j=0,n=X[0].size();j<n;++j)
    X[1].push_back(DragCoefficient(T.M[j],L,T,c,eta));
  return X;
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
```

5.2 DragTable() Parameters

L — **L** specifies an object's characteristic length, *L*. For the default values of **c** and **eta**, the units for **L** should be meters.

T — **T** specifies a DMRTABLE struct.

c — **c** specifies the speed of sound, *c*, in the fluid. The default value is 340.29 m/s, the sea-level value given in *U.S. Standard Atmosphere, 1976*.

eta — **eta** specifies the kinematic viscosity, η, of the fluid. The default value is 1.4607×10^{-5} m^2/s, the sea-level value given in *U.S. Standard Atmosphere, 1976*.

5.3 DragTable() Return Value

The DragTable() function returns a drag versus Mach table. The first index of the table relates to either a Mach number (first index = 0) or a drag coefficient (first index = 1). Mach values are unique and are stored in increasing order. For example, suppose that the output table is stored in the variable X. X[0][1] refers to the second smallest Mach number in the table, while X[1][1] refers to the corresponding drag coefficient.

It is possible for the output table to be empty. This will be the case when the blue diagonal line shown in figure 5 does not intersect the shaded region.

6. Creating a DMRTABLE Struct for Spheres – the Krumins1972() Function

The Krumins1972() function can be used to create a DMRTABLE that contains tabulated data given by Krumins's table I (*6*). Figure 6 presents a graph of the tabulated data.

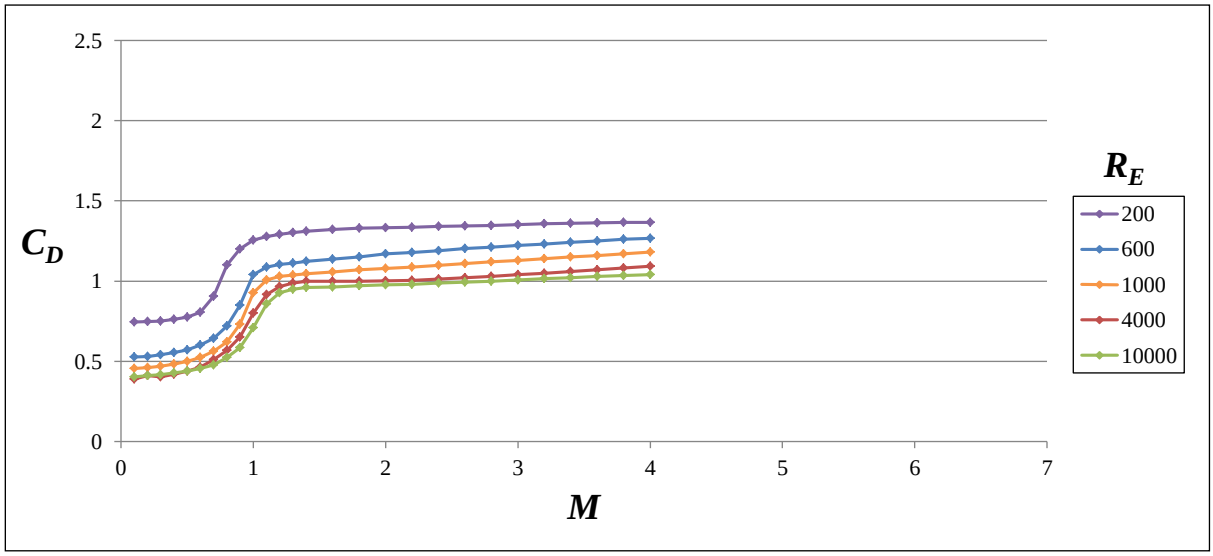


Figure 6. Drag as a function of both Reynolds number and Mach number, Krumins1972() function.

There appears to be an error in the tabulated values given by Krumins. For $R_E = 600$ and $M = 1.4$, C_D is stated to be 1.222. However, 1.222 is inconsistent with neighboring values. A value of 1.122 seems to be more likely, and that is what is used in the Krumins1972() function.

For sea-level atmospheric conditions, tabulated values for the upper Mach numbers are only applicable to very small spheres. That is, at sea-level atmospheric conditions, for $R_E = R_{E,\max}$ and $M = M_{\max}$,

$$L = \frac{\left(1.46107 \times 10^{-5}\ \text{m}^2/\text{s}\right)\left(10^4\right)}{(4)(343.29\ \text{m/s})} = 1.06 \times 10^{-4}\ \text{m} \tag{7}$$

6.1 Krumins1972() Code

```
inline DMRTABLE Krumins1972(){//<===SPHERE (CHARACTERISTIC LENGTH IS DIAMETER)
  double M[27]={.1,.2,.3,.4,.5,.6,.7,.8,.9,1,1.1,1.2,1.3,1.4,1.6,1.8,2,2.2,
                2.4,2.6,2.8,3,3.2,3.4,3.6,3.8,4};
  double Re[5]={200,600,1000,4000,10000};
  double Cd[27][5]={ 0.745 , 0.527 , 0.455 , 0.388 , 0.402 ,//..from "A Review
                     0.747 , 0.531 , 0.462 , 0.410 , 0.410 ,//  of Sphere Drag
                     0.750 , 0.540 , 0.470 , 0.403 , 0.418 ,//    Coefficients
                     0.761 , 0.554 , 0.483 , 0.420 , 0.429 ,//   Applicable to
                     0.775 , 0.572 , 0.500 , 0.440 , 0.440 ,//     Atmospheric
                     0.805 , 0.601 , 0.525 , 0.465 , 0.455 ,//         Density
                     0.905 , 0.643 , 0.562 , 0.510 , 0.478 ,//        Sensing"
                     1.100 , 0.721 , 0.622 , 0.568 , 0.523 ,//              by
                     1.200 , 0.850 , 0.730 , 0.650 , 0.585 ,//     Maigonis V.
                     1.255 , 1.040 , 0.928 , 0.800 , 0.710 ,//        Krumins,
                     1.277 , 1.086 , 1.008 , 0.915 , 0.857 ,//           Naval
                     1.292 , 1.105 , 1.028 , 0.967 , 0.927 ,//       Ordinance
                     1.302 , 1.113 , 1.037 , 0.988 , 0.950 ,//     Laboratory,
                     1.312 , 1.122 , 1.047 , 0.998 , 0.959 ,// 18 January 1972
                     1.321 , 1.138 , 1.058 , 1.000 , 0.962 ,//
                     1.330 , 1.152 , 1.070 , 1.000 , 0.971 ,//     (note error
                     1.334 , 1.169 , 1.080 , 1.002 , 0.978 ,//     in original
                     1.335 , 1.179 , 1.088 , 1.005 , 0.981 ,//        document
                     1.340 , 1.190 , 1.098 , 1.012 , 0.988 ,//     for M=1.4 &
                     1.344 , 1.202 , 1.110 , 1.021 , 0.994 ,//         Re=600,
                     1.348 , 1.212 , 1.119 , 1.029 , 1.000 ,//        Cd=1.122
                     1.352 , 1.222 , 1.130 , 1.040 , 1.008 ,//      NOT 1.222)
                     1.357 , 1.232 , 1.140 , 1.050 , 1.015 ,
                     1.360 , 1.241 , 1.150 , 1.060 , 1.021 ,
                     1.362 , 1.250 , 1.160 , 1.070 , 1.028 ,
                     1.365 , 1.260 , 1.171 , 1.081 , 1.034 ,
                     1.365 , 1.267 , 1.180 , 1.092 , 1.040 };
  DMRTABLE T={std::vector<double>(M,M+27),std::vector<double>(Re,Re+5),
    std::vector<std::vector<double> >(27,std::vector<double>(5))};
  for(int i=0;i<27;++i)for(int j=0;j<5;++j)T.Cd[i][j]=Cd[i][j];
  return T;
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
```

6.2 Krumins1972() Return Value

The Krumins1972() function returns a DMRTABLE that is based on the information presented in figure 6.

7. Creating a DMRTABLE Struct for Spheres – the Bailey1972() Function

The Bailey1972() function can be used to create a DMRTABLE that contains tabulated data based on the data that was used to create the graph in figure 2. However, the tabulated data presented in figure 2 could not be used directly. As a result of the digitizing process, each set of drag coefficients associated with a particular Reynolds number had its own set of Mach numbers. Recall that a requirement for DMRTABLE structs is that there be a single set of Mach numbers, where each Mach number/Reynolds number pair corresponds to a drag coefficient. Once a single set of Mach numbers was chosen, linear interpolations were used to find corresponding drag coefficients. For values outside the existing tabulated data (where interpolations were not possible), the drag coefficient was chosen to be equal to the known drag coefficient for the same Reynolds number that had the closest Mach number to that of the unknown value. This was done to allow the data obtained from Bailey and Hiatt (*3*) to be compatible with the DMRTABLE format. The extrapolated values are not meant to be reliable estimations. Figure 7 shows the reworked tabulated data.

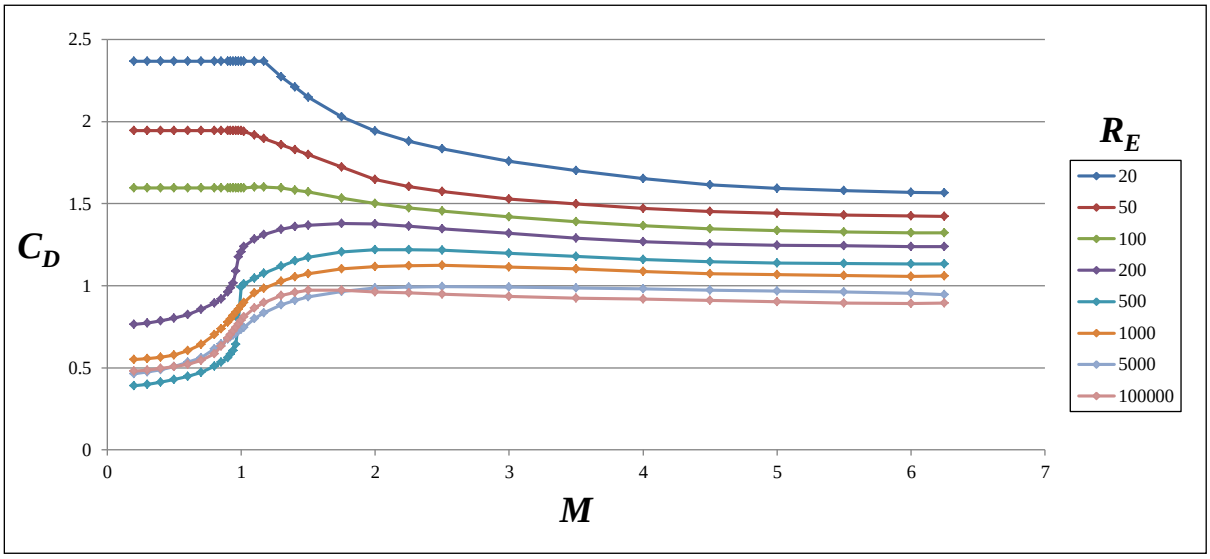


Figure 7. Drag as a function of both Reynolds number and Mach number, Bailey1972() function.

As was the case with the Krumins data, for sea-level atmospheric conditions, tabulated values for the upper Mach numbers are only applicable to very small spheres. That is, at sea-level atmospheric conditions, for $R_E = R_{E,\max}$ and $M = M_{\max}$,

$$L = \frac{\left(1.46107\times10^{-5}\ \mathrm{m^2/s}\right)\left(10^5\right)}{\left(6.25\right)\left(343.29\ \mathrm{m/s}\right)} = 6.81\times10^{-4}\ \mathrm{m} \qquad (8)$$

A second complication exists for the tabulated data shown in figure 2. The original graph was monochromatic and of low resolution. As a result of the line crossings that are shown in figure 2, the vertical ordering of the lines at the left of the graph could not be determined with certainty.

Due to problems inherent in Bailey and Hiatt's data, output from the Bailey1972() function may not be reliable.

7.1 Bailey1972() Code

```
inline DMRTABLE Bailey1972(){//<====SPHERE (CHARACTERISTIC LENGTH IS DIAMETER)
  double M[32]={.2,.3,.4,.5,.6,.7,.8,.85,.9,.92,.94,.96,.98,1,1.02,1.1,1.17,
    1.3,1.4,1.5,1.75,2,2.25,2.5,3,3.5,4,4.5,5,5.5,6,6.25};
  double Re[8]={20,50,100,200,500,1000,5000,100000};
  double Cd[32][8]={//.....from "Sphere Drag Coefficients for a Broad Range of
    2.36870,1.94649,1.59636,0.76371,0.38939,0.55163,0.46373,0.48074,//Mach and
    2.36870,1.94649,1.59636,0.77221,0.39864,0.55543,0.47538,0.48674,//Reynolds
    2.36870,1.94649,1.59636,0.78615,0.41172,0.56415,0.48921,0.49675,//Numbers"
    2.36870,1.94649,1.59636,0.80233,0.42849,0.57864,0.50728,0.50741,//      by
    2.36870,1.94649,1.59636,0.82408,0.44648,0.60392,0.53403,0.51976,//    A.B.
    2.36870,1.94649,1.59636,0.85629,0.47137,0.64207,0.56240,0.54619,//  Bailey
    2.36870,1.94649,1.59636,0.89466,0.50966,0.70254,0.61646,0.58626,//     and
    2.36870,1.94649,1.59636,0.91986,0.53299,0.73813,0.64609,0.63122,//      J.
    2.36870,1.94649,1.59636,0.96119,0.56086,0.77832,0.67571,0.67948,//   Hiatt
    2.36870,1.94649,1.59636,0.98587,0.58237,0.79756,0.68756,0.70119,//    AIAA
    2.36870,1.94649,1.59636,1.01559,0.60388,0.81743,0.69941,0.72333,// JOURNAL
    2.36870,1.94649,1.59636,1.09007,0.64162,0.83731,0.71126,0.74547,//VOL. 10,
    2.36870,1.94649,1.59636,1.17506,0.79767,0.85718,0.72312,0.76761,//  NO. 11
    2.36870,1.94649,1.59641,1.20641,0.99182,0.87706,0.73497,0.78974,
    2.36870,1.94120,1.59741,1.23807,1.01151,0.89693,0.74682,0.81188,
    2.36870,1.91813,1.60031,1.28403,1.04528,0.95660,0.80002,0.86507,//    note
    2.36756,1.89794,1.60103,1.31113,1.07482,0.98499,0.83612,0.89758,//   that,
    2.27367,1.85910,1.59661,1.34361,1.11994,1.02851,0.88487,0.94018,//  due to
    2.21047,1.82915,1.58323,1.36083,1.15182,1.05418,0.91203,0.95996,//problems
    2.14975,1.79936,1.57171,1.36849,1.17336,1.07425,0.93241,0.97173,//inherent
    2.02965,1.72291,1.53307,1.38049,1.20554,1.10308,0.96463,0.97201,//      in
    1.94368,1.64832,1.50093,1.37597,1.21885,1.11707,0.98666,0.96330,//Bailey &
    1.88095,1.60474,1.47375,1.36296,1.22008,1.12308,0.99178,0.95714,// Hiatt's
    1.83547,1.57565,1.45427,1.34791,1.21673,1.12396,0.99567,0.94755,//   data,
    1.76001,1.52960,1.42090,1.31939,1.19860,1.11483,0.99306,0.93589,//  output
    1.70090,1.49750,1.39121,1.29015,1.17815,1.10250,0.98768,0.92563,//    from
    1.65234,1.47045,1.36566,1.26726,1.16070,1.08658,0.98047,0.91797,//    this
    1.61551,1.45186,1.34613,1.25435,1.14757,1.07450,0.97410,0.91001,//function
    1.59385,1.44071,1.33514,1.24665,1.13922,1.06699,0.96672,0.90206,// may not
    1.58020,1.43193,1.32701,1.24370,1.13658,1.06315,0.96138,0.89582,//      be
    1.56886,1.42477,1.32378,1.23890,1.13280,1.05767,0.95420,0.89272,//reliable
    1.56718,1.42394,1.32315,1.23827,1.13217,1.05872,0.94701,0.89344};
  DMRTABLE T={std::vector<double>(M,M+32),std::vector<double>(Re,Re+8),
    std::vector<std::vector<double> >(32,std::vector<double>(8))};
  for(int i=0;i<32;++i)for(int j=0;j<8;++j)T.Cd[i][j]=Cd[i][j];
  return T;
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
```

7.2 Bailey1972() Return Value

The Bailey1972() function returns a DMRTABLE that is based on the information presented in figure 7.

---

8. Example – Creating Drag Versus Mach Tables

The example code presented in section 8.1 creates a drag coefficient versus Mach number table that is applicable to spheres having a diameter of 0.0001 m and sea-level atmospheric conditions. The code uses the Krumins1972() function to create a table based on Krumins's data. Substituting the Bailey1972() function for the Krumins1972() function allows the tables to be based on the data from Bailey and Hiatt. Figure 8 presents a graphical interpretation of the tabulated data.

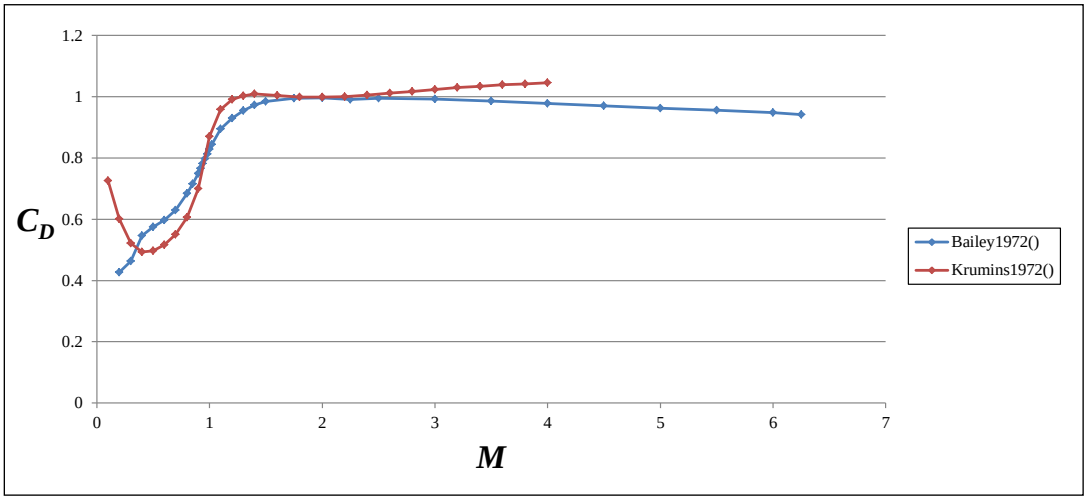


Figure 8. Drag vs. Mach curves created for a 0.0001-m sphere at sea-level atmospheric conditions.

8.1 Example Code

```
#include <cstdio>//.....................................................printf()
#include "y_drag.h"//.....................................yDrag,<vector>
int main(){
  std::vector<std::vector<double> >X=
    yDrag::DragTable(.0001,yDrag::Krumins1972());
//  yDrag::DragTable(.0001,yDrag::Bailey1972());
  for(int i=0,n=X[0].size();i<n;++i)printf("%f,%f\n",X[0][i],X[1][i]);
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
```

9. Code Summary

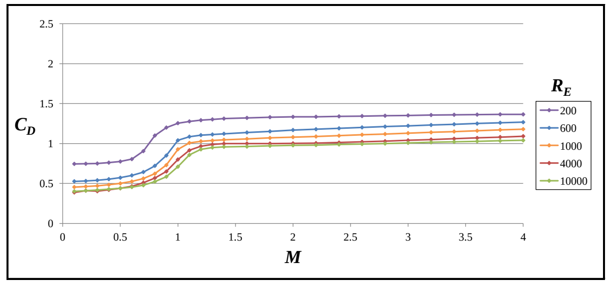
A summary sheet is provided at the end of this report. It presents the yDrag namespace, which contains the single struct and four functions that are described in this report.

yDrag Summary

y_drag.h

```cpp
#ifndef Y_DRAG_GUARD//       See Yager, R. J. "Calculating Drag Coefficients for
#define Y_DRAG_GUARD//                      Spheres and Other Shapes Using C++"
#include <cmath>//................................................sqrt(),pow()
#include <vector>//.....................................................vector
namespace yDrag{//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  struct DMRTABLE{//<=======TABULATED DRAG AS A FUNCTION OF MACH AND REYNOLDS #s
    std::vector<double>M;//<---------MACH NUMBERS (UNIQUE & IN INCREASING ORDER)
    std::vector<double>Re;//<----REYNOLDS NUMBERS (UNIQUE & IN INCREASING ORDER)
    std::vector<std::vector<double> >Cd;//<-----DRAG COEF. (M[i],Re[j],Cd[i][j])
  };//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
  inline double DragCoefficient(//<CALCULATES A DRAG COEFFICIENT FROM A DMRTABLE
      double M,//<---------------------------------------------------MACH NUMBER
      double L,//<-------------CHARACTERISTIC LENGTH (METERS FOR DEFAULT VALUES)
      DMRTABLE&T,//<---------------------------------------------------A DRMTABLE
      double c=340.29,//<----LOCAL SPEED OF SOUND (DEFAULT - SEA-LEVEL ATM. m/s)
      double eta=1.4607E-5){//<-------KINIMATIC VISCOSITY (SEA-LEVEL ATM. m^2/s)
    double x=M,y=M*c*L/eta;
    const std::vector<double>&X=T.M,&Y=T.Re;
    const std::vector<std::vector<double> >&Z=T.Cd;
    int i;/*<-*/for(i=1;i<(int)X.size()-1;++i)if(x<X[i])break;// X[i]<=x<=X[i+1]
    int j;/*<-*/for(j=1;j<(int)Y.size()-1;++j)if(y<Y[j])break;// Y[j]<=y<=Y[j+1]
    double Za=(y-Y[j-1])/(Y[j]-Y[j-1])*(Z[i-1][j]-Z[i-1][j-1])+Z[i-1][j-1];
    double Zb=(y-Y[j-1])/(Y[j]-Y[j-1])*(Z[i][j]-Z[i][j-1])+Z[i][j-1];
    return (x-X[i-1])/(X[i]-X[i-1])*(Zb-Za)+Za;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
  inline std::vector<std::vector<double> >DragTable(//<=CREATES A Cd VS. M TABLE
      double L,//<-------------CHARACTERISTIC LENGTH (METERS FOR DEFAULT VALUES)
      DMRTABLE&T,//<---------------------------------------------------A DRMTABLE
      double c=340.29,//<-SPEED OF SOUND IN FLUID (DEFAULT - SEA-LEVEL ATM. m/s)
      double eta=1.4607E-5){//<-------KINIMATIC VISCOSITY (SEA-LEVEL ATM. m^2/s)
    std::vector<std::vector<double> >X(2,std::vector<double>(0));
    double M_min=eta*T.Re[0]/c/L,M_max=eta*T.Re.back()/c/L;
    if(M_min>T.M.back()||M_max<T.M[0])return X;
    X[0].push_back(M_min>T.M[0]?M_min:T.M[0]);
    for(int j=1,n=T.M.size()-1;j<n;++j)
      if(T.M[j]>M_min&&T.M[j]<M_max)X[0].push_back(T.M[j]);
    X[0].push_back(M_max<T.M.back()?M_max:T.M.back());
    for(int j=0,n=X[0].size();j<n;++j)
      X[1].push_back(DragCoefficient(T.M[j],L,T,c,eta));
    return X;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
  inline DMRTABLE Krumins1972(){//<===SPHERE (CHARACTERISTIC LENGTH IS DIAMETER)
    double M[27]={.1,.2,.3,.4,.5,.6,.7,.8,.9,1,1.1,1.2,1.3,1.4,1.6,1.8,2,2.2,
                  2.4,2.6,2.8,3,3.2,3.4,3.6,3.8,4};
    double Re[5]={200,600,1000,4000,10000};
    double Cd[27][5]={ 0.745 , 0.527 , 0.455 , 0.388 , 0.402 ,//..from "A Review
                       0.747 , 0.531 , 0.462 , 0.410 , 0.410 ,//  of Sphere Drag
                       0.750 , 0.540 , 0.470 , 0.403 , 0.418 ,//    Coefficients
                       0.761 , 0.554 , 0.483 , 0.420 , 0.429 ,//   Applicable to
                       0.775 , 0.572 , 0.500 , 0.440 , 0.440 ,//     Atmospheric
                       0.805 , 0.601 , 0.525 , 0.465 , 0.455 ,//         Density
                       0.905 , 0.643 , 0.562 , 0.510 , 0.478 ,//        Sensing"
                       1.100 , 0.721 , 0.622 , 0.568 , 0.523 ,//              by
                       1.200 , 0.850 , 0.730 , 0.650 , 0.585 ,//     Maigonis V.
                       1.255 , 1.040 , 0.928 , 0.800 , 0.710 ,//        Krumins,
                       1.277 , 1.086 , 1.008 , 0.915 , 0.857 ,//           Naval
                       1.292 , 1.105 , 1.028 , 0.967 , 0.927 ,//       Ordinance
                       1.302 , 1.113 , 1.037 , 0.988 , 0.950 ,//     Laboratory,
                       1.312 , 1.122 , 1.047 , 0.998 , 0.959 ,// 18 January 1972
                       1.321 , 1.138 , 1.058 , 1.000 , 0.962 ,//
                       1.330 , 1.152 , 1.070 , 1.000 , 0.971 ,//     (note error
                       1.334 , 1.169 , 1.080 , 1.002 , 0.978 ,//     in original
                       1.335 , 1.179 , 1.088 , 1.005 , 0.981 ,//        document
                       1.340 , 1.190 , 1.098 , 1.012 , 0.988 ,//     for M=1.4 &
                       1.344 , 1.202 , 1.110 , 1.021 , 0.994 ,//         Re=600,
                       1.348 , 1.212 , 1.119 , 1.029 , 1.000 ,//        Cd=1.122
                       1.352 , 1.222 , 1.130 , 1.040 , 1.008 ,//      NOT 1.222)
                       1.357 , 1.232 , 1.140 , 1.050 , 1.015 ,
                       1.360 , 1.241 , 1.150 , 1.060 , 1.021 ,
                       1.362 , 1.250 , 1.160 , 1.070 , 1.028 ,
                       1.365 , 1.260 , 1.171 , 1.081 , 1.034 ,
                       1.365 , 1.267 , 1.180 , 1.092 , 1.040 };
    DMRTABLE T={std::vector<double>(M,M+27),std::vector<double>(Re,Re+5),
      std::vector<std::vector<double> >(27,std::vector<double>(5))};
    for(int i=0;i<27;++i)for(int j=0;j<5;++j)T.Cd[i][j]=Cd[i][j];
    return T;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
  inline DMRTABLE Bailey1972(){//<====SPHERE (CHARACTERISTIC LENGTH IS DIAMETER)
    double M[32]={.2,.3,.4,.5,.6,.7,.8,.85,.9,.92,.94,.96,.98,1,1.02,1.1,1.17,
      1.3,1.4,1.5,1.75,2,2.25,2.5,3,3.5,4,4.5,5,5.5,6,6.25};
    double Re[8]={20,50,100,200,500,1000,5000,100000};
    double Cd[32][8]={//.....from "Sphere Drag Coefficients for a Broad Range of
      2.36870,1.94649,1.59636,0.76371,0.38939,0.55163,0.46373,0.48074,//Mach and
      2.36870,1.94649,1.59636,0.77221,0.39864,0.55543,0.47538,0.48674,//Reynolds
      2.36870,1.94649,1.59636,0.78615,0.41172,0.56415,0.48921,0.49675,//Numbers"
      2.36870,1.94649,1.59636,0.80233,0.42849,0.57864,0.50728,0.50741,//      by
      2.36870,1.94649,1.59636,0.82408,0.44648,0.60392,0.53403,0.51976,//    A.B.
      2.36870,1.94649,1.59636,0.85629,0.47137,0.64207,0.56240,0.54619,//  Bailey
      2.36870,1.94649,1.59636,0.89466,0.50966,0.70254,0.61646,0.58626,//     and
      2.36870,1.94649,1.59636,0.91986,0.53299,0.73813,0.64609,0.63122,//      J.
      2.36870,1.94649,1.59636,0.96119,0.56086,0.77832,0.67571,0.67948,//   Hiatt
      2.36870,1.94649,1.59636,0.98587,0.58237,0.79756,0.68756,0.70119,//    AIAA
      2.36870,1.94649,1.59636,1.01559,0.60388,0.81743,0.69941,0.72333,// JOURNAL
      2.36870,1.94649,1.59636,1.09007,0.64162,0.83731,0.71126,0.74547,//VOL. 10,
      2.36870,1.94649,1.59636,1.17506,0.79767,0.85718,0.72312,0.76761,//  NO. 11
      2.36870,1.94649,1.59641,1.20641,0.99182,0.87706,0.73497,0.78974,
      2.36870,1.94120,1.59741,1.23807,1.01151,0.89693,0.74682,0.81188,
      2.36870,1.91813,1.60031,1.28403,1.04528,0.95660,0.80002,0.86507,//    note
      2.36756,1.89794,1.60103,1.31113,1.07482,0.98499,0.83612,0.89758,//   that,
      2.27367,1.85910,1.59661,1.34361,1.11994,1.02851,0.88487,0.94018,//  due to
      2.21047,1.82915,1.58323,1.36083,1.15182,1.05418,0.91203,0.95996,//problems
      2.14975,1.79936,1.57171,1.36849,1.17336,1.07425,0.93241,0.97173,//inherent
      2.02965,1.72291,1.53307,1.38049,1.20554,1.10308,0.96463,0.97201,//      in
      1.94368,1.64832,1.50093,1.37597,1.21885,1.11707,0.98666,0.96330,//Bailey &
      1.88095,1.60474,1.47375,1.36296,1.22008,1.12308,0.99178,0.95714,// Hiatt's
      1.83547,1.57565,1.45427,1.34791,1.21673,1.12396,0.99567,0.94755,//   data,
      1.76001,1.52960,1.42090,1.31939,1.19860,1.11483,0.99306,0.93589,//  output
      1.70090,1.49750,1.39121,1.29015,1.17815,1.10250,0.98768,0.92563,//    from
      1.65234,1.47045,1.36566,1.26726,1.16070,1.08658,0.98047,0.91797,//    this
      1.61551,1.45186,1.34613,1.25435,1.14757,1.07450,0.97410,0.91001,//function
      1.59385,1.44071,1.33514,1.24665,1.13922,1.06699,0.96672,0.90206,// may not
      1.58020,1.43193,1.32701,1.24370,1.13658,1.06315,0.96138,0.89582,//      be
      1.56886,1.42477,1.32378,1.23890,1.13280,1.05767,0.95420,0.89272,//reliable
      1.56718,1.42394,1.32315,1.23827,1.13217,1.05872,0.94701,0.89344};
    DMRTABLE T={std::vector<double>(M,M+32),std::vector<double>(Re,Re+8),
      std::vector<std::vector<double> >(32,std::vector<double>(8))};
    for(int i=0;i<32;++i)for(int j=0;j<8;++j)T.Cd[i][j]=Cd[i][j];
    return T;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
}//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#endif
```

Krumins Data



Symbols and Equations

Object-Related Symbols

R_E	Reynolds number
v	speed relative to fluid
L	characteristic length (diameter for spheres)
M	Mach number

Fluid-Related Symbols

η	kinematic viscosity (in atmosphere, at sea level, η_0=1.4607×10^{-5} m^2/s)
μ	dynamic viscosity (in atmosphere, at sea level, μ_0=17894×10^{-5} kg/(m·s))
ρ	density (in atmosphere, at sea level, ρ_0=101325 N/m^2)
c	speed of sound (in atmosphere, at sea level, c_0=340.29 m/s)
T	temperature (in atmosphere, at sea level, T_0=288.15 K)
β	constant equal to 1.458×10^{-6} kg/(s·m·K$^{1/2}$)
S	Sutherland's constant (110.4 K)

Definitions

$$R_E \equiv \frac{vL}{\eta} \quad , \quad \eta \equiv \frac{\mu}{\rho} \quad , \quad M \equiv \frac{v}{c}$$

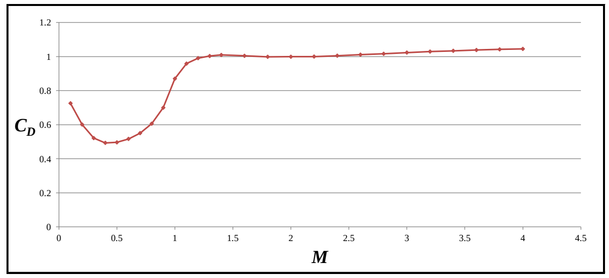
Atmospheric Dynamic Viscosity

$$\mu = \frac{\beta T^{3/2}}{T+S}$$

Example

```cpp
#include <cstdio>//....................................................printf()
#include "y_drag.h"//..........................................yDrag,<vector>
int main(){
  std::vector<std::vector<double> >X=
    yDrag::DragTable(.0001,yDrag::Krumins1972());
  for(int i=0,n=X[0].size();i<n;++i)printf("%f,%f\n",X[0][i],X[1][i]);
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~04SEP2013~~~~~~
```

GRAPHED OUTPUT:



10. References

1. Yager, R. J. *Digitizing Printed Graphs Using C++;* ARL-TN-577; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, 2013.

2. Munson, B. R.; Young, D. F.; Okiishi, T. H. *Fundamentals of Fluid Mechanics;* 3rd ed.; John Wiley & Sons, Inc.: New York, 1998.

3. Bailey, A. B.; Hiatt, J. Sphere Drag Coefficients for a Broad Range of Mach and Reynolds Numbers. *AIAA Journal* **1972,** *10* (11), 1436–1440.

4. *U.S. Standard Atmosphere, 1976;* NASA-TM-X-74335; National Aeronautics and Space Administration: Washington, DC, October 1976.

5. Yager, R. J. *Calculating Atmospheric Conditions (Temperature, Pressure, Air Density, and Speed of Sound) Using C++;* ARL-TN-543; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, 2013.

6. Krumins, M. V. *A Review of Sphere Drag Coefficients Applicable to Atmospheric Density Sensing;* NOLTR 72-34; Naval Ordinance Laboratory: Silver Spring, MD, 1972.

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

2 (PDF) DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT

1 (PDF) GOVT PRINTG OFC A MALHOTRA

1 (PDF) DIR USARL RDRL WML A R YAGER