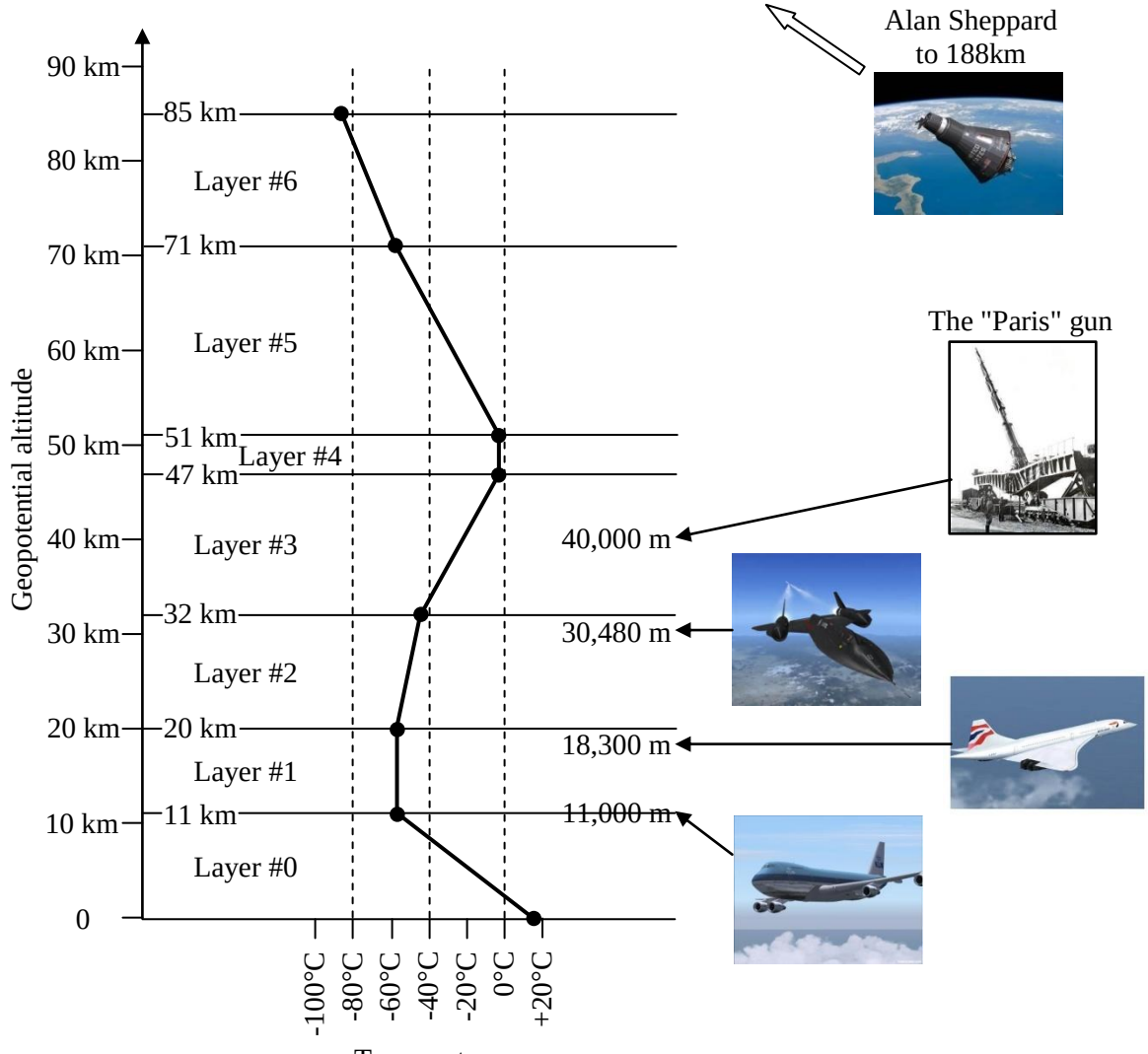
Formulae and code for the U.S. Standard Atmosphere (1976)

It is sometimes convenient to have at hand a mathematical model for the atmosphere at various altitudes, and a computer code to implement it. In this paper, I will look at the so-called U.S. Standard Atmosphere, as published in 1976. It models altitudes up to about 85 kilometers and is intended to represent average year-round conditions at mid-latitudes during periods of average solar activity.

The model consists of seven layers, in each of which temperature is assumed to change linearly with altitude. The model assumes that the mean molecular weight of air stays unchanged through all the layers. The model also assumes that the air is dry, with zero humidity. Since water (in gaseous form) has a lower density than the average molecules in dry air, the presence of humidity reduces the values of air density from those predicted by the model. The reduction is usually on the order of 1% but can be as high as 3% during extreme conditions.

The following diagram shows the layers used in the model. In the model, there are two main inputs which determine the characteristics of the air. The first, of course, in gravity. The second is temperature. To make their model easier to use, the designers of this standard atmosphere chose things so that the temperature would change at a linear rate inside each layer. The vertical curve through this graph shows how the temperature decreases, then increases, and then decreases once more as one ascends in altitude from the surface.



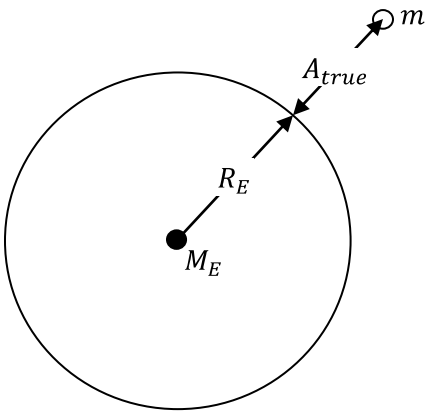
Geometric altitude versus geopotential altitude

The designers of the 1976 version of the U.S. Standard Atmosphere based their approach on an earlier version, published in 1962. Back in 1962, computers were rare beasts. Airplanes did not fly as high as they do nowadays, either. It was the end of the propeller era, in which all calculations were done by slide rule. The difficulties of doing calculations by hand had led engineers and scientists to make a simplifying assumption about the nature of gravity, namely, that it is constant. The force of gravity F_g exerted on some mass m is equal to that mass multiplied by the local gravitational constant g_0, whose value is approximately 32 feet per second squared, or 9.8 meters per second squared. That is a pretty good assumption throughout the envelope of flight achieved in the early days.

That assumption starts to fail as one gets to higher altitude. It becomes more important that is Earth is a sphere, and not just flat. The force of gravity decreases inversely with the square of the distance to the Earth's center. Since the Earth is pretty big, altitudes have to get pretty high before they start to constitute a meaningful fraction of the Earth's radius, and a more accurate description of gravity is needed.

A few extra lines of code are all that is needed today to deal with either one of these two formulations of gravity. But the difficulties of the earlier era became a part of the way certain data is presented, and those features were carried over into the 1976 version of the U.S. Standard Atmosphere.

The geometric altitude is the true altitude, which is to say, the distance above local mean sea level. This is the distance a radar altimeter would report if one was flying over the "mean" ocean. A second altitude – the geopotential altitude – is used to adjust between the two formulations of gravity. It is not hard to work out how the two altitudes are related.



The figure to the left shows a mass m at a true altitude, or geometric altitude, A_{true}. The more accurate description of gravity uses this altitude, the radius R_E and mass M_E of the Earth, and the general gravitational constant G. In this formulation, the force of gravity acting on mass m is given by:

$$F_g = G\frac{mM_E}{(R_E + A_{true})^2} \quad (1)$$

I propose to calculate the potential energy possessed by this mass when it is raised from the surface of the Earth to its true altitude A_{true}. If the mass is dropped from this altitude, its potential energy will be converted into real kinetic energy as the mass accelerates towards the surface.

It takes a force to lift the mass off the surface and up to altitude A_{true}. Mechanical work is done as that lifting force pulls the mass up. The potential energy possessed by the mass at any altitude is the total mechanical work done to lift it up to that height. If a force F pushes or pulls a mass through some distance Δz, the mechanical work done is the product $F\Delta z$. If the force varies with distance, as it does in this case with gravity, it is necessary to break up the journey into little steps, each of distance dz, for example, and to add up using the Calculus the little Fdz contributions which are made during the little steps. We can calculate the potential energy as follows:

$$
\begin{aligned}
Potential\ energy\ at\ A_{true} &= \int_{z=0}^{z=A_{true}} \left(F_g\ at\ altitude\ z\right)dz \\
&= \int_{z=0}^{z=A_{true}} G\frac{mM_E}{(R_E+z)^2}\,dz \\
&= -GmM_E \int_{z=0}^{z=A_{true}} \frac{d(R_E+z)}{(R_E+z)^2} \\
&= -GmM_E\left(\frac{1}{R_E+z}\Big|_{z=0}^{z=A_{true}}\right) \\
&= GmM_E\left(\frac{1}{R_E}-\frac{1}{R_E+A_{true}}\right) \qquad (2)
\end{aligned}
$$

I want to compare this result to the potential energy one gets using the other formulation, in which the force of gravity is constant and equal to $F_g = g_0 m$. This time, we are going to raise the mass up to a slightly different altitude, the geopotential altitude $A_{potential}$. The calculation of the mechanical work is:

$$
\begin{aligned}
Potential\ energy\ at\ A_{potential} &= \int_{z=0}^{z=A_{potential}} \left(F_g\ at\ altitude\ z\right)dz \\
&= \int_{z=0}^{z=A_{potential}} mg_0\,dz \\
&= mg_0 A_{potential} \qquad (3)
\end{aligned}
$$

I now want to choose the geopotential altitude $A_{potential}$ so that the mass will have the same gravitational potential energy (calculated using the simple formulation) as it has at its true altitude A_{true} (calcuated using the inverse-square formulation). To find the equivalence, we set the potential energies of the two methods equal, as follows. (Along the way, I am going to use the identity $g_0 = GM_E/R_E^2$.)

$$
\begin{aligned}
& GmM_E\left(\frac{1}{R_E}-\frac{1}{R_E+A_{true}}\right) = mg_0 A_{potential} \\
\rightarrow \quad & \left(1-\frac{R_E}{R_E+A_{true}}\right) = \frac{1}{R_E}A_{potential} \\
\rightarrow \quad & \frac{R_E+A_{true}}{R_E} = \frac{R_E}{R_E-A_{potential}} \\
\rightarrow \quad & A_{true} = A_{potential}\frac{R_E}{R_E-A_{potential}} \qquad (4)
\end{aligned}
$$

Since the denominator $R_E - A_{potential}$ will always be a little bit less than the numerator R_E, the true altitude A_{true} will always be a little bit higher than the geopotential altitude $A_{potential}$. This makes sense, since the mass has to be lifted a little bit farther through the weaker gravity to get as much potential energy as having been lifted through surface-level gravity at constant strength g_0.

Here is the important point. The altitudes used to define the atmospheric layers, as shown in the figure on the first page, are geopotential altitudes, not true altitudes.

The three physical laws used in the model

The U.S. Standard Atmosphere is based on three physical "laws". Well, they are not really laws, *per se*, but relatively simple models of physical phenomena. To be sure, there are much more complicated versions of each of these laws, which use additional parameters to extend their validity over a greater range of conditions.

Law #1: The ideal gas law, in the form:

$$P = \rho RT \quad (5)$$

where:

- P is the static pressure, measured in Newtons per square meter,
- ρ is the density, in kilograms per cubic meter,
- R is not the universal gas constant, $8.21432\ Joules/mole \cdot °K$, as it would be if Equation (5) had been written using the number of moles as the measure of quantity. Here, Equation (5) has been written with the quantity of air measured by mass or, more precisely, by density. In these circumstances, the appropriate value for R is the "gas constant for air". It is derived by dividing the universal gas constant by the mean molecular weight of air, $0.0289644\ kg/mole$. The result of the division is $R = 287.053\ Joules/kg \cdot °K$, and
- T is the temperature, in degrees Kelvin $(°K)$.

Law #2: Equation of hydrostatic equilibrium, in the form:

$$\frac{dP}{dz} = -g_0\rho \quad (6)$$

where:

- z is the independent variable in the direction of the local vertical, in meters, and
- g_0 is the local surface-level gravitational constant, $9.80665\ m/sec^2$.

Law #3: Dynamic viscosity as a function of temperature, in the form:

$$\mu = \beta \frac{T^{1.5}}{T + S} \quad (7)$$

where

- μ is the dynamic viscosity, in $kg/m \cdot sec$,
- β is the experimentally-determined coefficient for air, $1.458 \times 10^{-6}\ kg/m \cdot sec \cdot \sqrt{°K}$, and
- S is Sutherland's constant for air, $110.4°K$.

The parameters at sea level

It is useful to have at hand the values of the state variables (pressure, temperature, density and dynamic viscosity) at sea level, which is the zero altitude in the model. They are listed in the following table.

$$\left.\begin{array}{c} P_0 = 101{,}325\ N/m^2 \\ T_0 = 15.0°C + 273.15°C = 288.15°K \\ \rho_0 = 1.22500\ kg/m^3 \\ \mu_0 = 1.78938 \times 10^{-5}\ kg/m \cdot sec \end{array}\right\} \quad (8)$$

Derivation of the equations

A very important assumption in the U.S. Standard Atmosphere is that the temperature changes with height at a constant rate within each layer. Let's use the symbol λ_n for the "lapse rate" in layer #n. This is the number of degrees by which the temperature rises in that layer for each meter of increase in the altitude. (A negative lapse rate means the temperature falls as the height increases.)

Let's let T_n and T_{n+1} be the absolute temperatures at the bottom and top of the n^{th} layer, respectively. Using similar subscripts, z_n and z_{n+1} will be the geopotential altitudes at the bottom and top of the n^{th} layer, respectively. The temperature at the top of the n^{th} layer can be expressed in terms of the temperature at the bottom of the n^{th} layer as follows:

$$T_{n+1} = T_n + (z_{n+1} - z_n)\lambda_n \quad (9)$$

and the temperature $T(z)$ at altitude z anywhere inside the n^{th} layer can be written as:

$$T(z) = T_n + (z - z_n)\lambda_n \quad (10)$$

Re-arranging the equation for hydrostatic equilibrium, dividing by the pressure P and then substituting from the ideal gas law gives:

$$\begin{aligned} \frac{dP}{P} &= -\frac{g_0 \rho dz}{\rho R T} \\ &= -\frac{g_0}{R}\left(\frac{dz}{T}\right) \end{aligned} \quad (11)$$

The expression for the temperature given in Equation (10) can be differentiated wih respect to altitude z to give:

$$\frac{dT}{dz} = \lambda_n \quad (12)$$

The value of this derivative is different inside each layer, of course, but it is a constant within that layer. If the lapse rate λ_n is not zero, then we can re-arrange Equation (12) to give $dz = dT/\lambda_n$ and substitute that dz in Equation (11). On the other hand, if the lapse rate λ_n is zero (called an "isothermal" layer), then the temperature in Equation (11) is constant and Equation (11) can be integrated as it is. We must keep separate track of these two cases. We get:

$$For\ \lambda_n \neq 0 \quad \frac{dP}{P} = -\frac{g_0}{R\lambda_n}\left(\frac{dT}{T}\right) \quad (13A)$$

$$For\ \lambda_n = 0 \quad \frac{dP}{P} = -\frac{g_0}{RT_n}dz \quad (13B)$$

Both of these expressions can be integrated "upwards" with respect to altitude. We will not start with sea level as the bottom, but from the bottom of the n^{th} layer in which this particular version of Equation (10) obtains. We have to integrate the two cases separately. Doing so gives:

$$\begin{array}{cc} For\ \lambda_n \neq 0 & For\ \lambda_n = 0 \\ \int_{P=P_n}^{P(z)} \frac{dP}{P} = -\frac{g_0}{R\lambda_n}\int_{T=T_n}^{T(z)} \frac{dT}{T} & \int_{P=P_n}^{P(z)} \frac{dP}{P} = -\frac{g_0}{RT_n}\int_{z=z_n}^{z} dz \\ \left(\ln P|_{P=P_n}^{P=P(z)}\right) = -\frac{g_0}{R\lambda_n}\left(\ln T|_{T=T_n}^{T=T(z)}\right) & \left(\ln P|_{P=P_n}^{P=P(z)}\right) = -\frac{g_0}{RT_n}\left(z|_{z=z_n}^{z}\right) \\ \ln P - \ln P_n = -\frac{g_0}{R\lambda_n}(\ln T - \ln T_n) & \ln P - \ln P_n = -\frac{g_0}{RT_n}(z - z_n) \\ \frac{P}{P_n} = \left(\frac{T}{T_n}\right)^{-\frac{g_0}{R\lambda_n}} & \frac{P}{P_n} = e^{-\frac{g_0}{RT_n}(z-z_n)} \end{array} \quad (14)$$

Substituting Equation (10) for the temperature into the first of Equations (14) gives the final form for the two pressure equations:

$$For\ \lambda_n \neq 0 \quad P(z) = P_n\left[1 + \frac{(z - z_n)\lambda_n}{T_n}\right]^{-\frac{g_0}{R\lambda_n}} \quad (15A)$$

$$For\ \lambda_n = 0 \quad P(z) = P_n e^{-\frac{g_0}{RT_n}(z-z_n)} \quad (15B)$$

Once the temperature at altitude z has been determined using Equation (10), the pressure can be calculated using Equation (15). Calculating the density is a straight-forward application of the ideal gas law:

$$\rho(z) = \frac{P(z)}{RT(z)} \quad (16)$$

The dynamic viscosity is another straight-forward calculation, using:

$$\mu(z) = \beta\frac{T(z)^{1.5}}{T(z) + S} \quad (17)$$

In many applications, the kinematic viscosity is a more useful parameter than the dynamic viscosity. If so, it can easily be calculated from its dependence on the dynamic viscosity and the density:

$$\nu(z) = \frac{\mu(z)}{\rho(z)} \quad (18)$$

The lapse rates

The following table sets out the lapse rates assumed in the U.S. Standard Atmosphere (1976).

Bottom altitude (meters)	Layer #	Top altitude (meters)	Lapse rate (°C/meter)
0	0	11,000	−0.0065
11,000	1	20,000	0
20,000	2	32,000	+0.0010
32,000	3	47,000	+0.0028
47,000	4	51,000	0
51,000	5	71,000	−0.0028
71,000	6	85,000	−0.0020

Implementing the equations

If the state variables are known at the bottom of layer #n, in which layer the lapse rate is λ_n, then Equations (10) and (15) through (18) are all that are needed to calculate the state variables anywhere inside that layer. However, one will not know the state variables at the bottom of layer #n until the underlying layers have been processed first.

For this reason, I have chosen to implement the equations in two parts, which correspond to the two subroutines in the code. The first part/subroutine calculates the state variables at the altitudes which are the boundaries between adjacent layers. These boundary values are saved in vectors where they are readily available for calculations within any particular layer. The calculations for any particular altitude in any particular layer are carried out by the second part/subroutine.

I have listed in Appendix "A" a Visual Basic module which contains the two subroutines. The code was developed using Visual Basic 2010 Express. The first subroutine, named `InitializeAtmsphereLayers`, must be called before any other calculations can be carried out. It initializes the boundary layer values. This subroutine does not have any arguments.

The second subroutine is called `LookUpAtmosphericStateVariables`. It has only one input argument, the geometric, or true, altitude, measured in meters above mean sea level. The subroutine returns seven values. In addition to the five state variables we have talked about (temperature, pressure, density, dynamic viscosity and kinematic viscosity) it also returns the geopotential altitude (just for reference if it is needed by the calling routine) and the speed of sound. All of the arguments are stated in S.I. units. The variables are returned to the calling procedure through their declaration as `ByRef` arguments.

The speed of sound

For my own purposes, I have included the calculation of the local speed of sound in the subroutine `LookUpAtmosphericStateVariables`. To do this, I used the following relationship:

$$SS(z) = \sqrt{\gamma \frac{P(z)}{\rho(z)}} \quad (19)$$

where:

- SS is the speed of sound at altitude z, in meters per second,
- P and ρ are the local pressure and density, repsectively, and
- $\gamma = 1.401$ is the ratio of specific heats for air, also called the Adiabatic Index for air.

State variables at the boundary altitudes

For reference purposes, I have set out in the following table the values of several state variables at the boundary altitudes.

Geopotential altitude (meters)	**Temp (°C)**	**Pressure (N/m^2)**	**Density (kg/m^3)**	**Dynamic viscosity (kg/m.sec)**	**Kinematic viscosity (m^2/sec)**	**Speed of sound (m/s)**
0	+15.0	101,325	1.22500	1.78938E-5	1.46072E-5	340.294
11,000	−56.5	22,632.1	0.363918	1.42161E-5	3.90641E-5	295.070
20,000	−56.5	5,474.88	8.80348E-2	1.42161E-5	1.61483E-4	295.070
32,000	−44.5	868.018	1.32250E-2	1.48679E-5	1.12423E-3	303.131
47,000	−2.5	110.906	1.42753E-3	1.70368E-5	1.19344E-2	329.799
51,000	−2.5	66.9388	8.61604E-4	1.70368E-5	1.97733E-2	329.799
71,000	−58.5	3.95641	6.42108E-5	1.41060E-5	0.219682	293.704

Jim Hawley

If you found this description helpful, please let me know. If you spot any errors or omissions, please send an e-mail. Thank you.

Appendix "A"

Listing of the VB2010 code for the U.S. Standard Atmosphere (1976)

I have listed the module which contains the two subroutines. I have also listed a short Windows Forms application which handles interactive queries.

Listing of the module

```
Option Strict On
Option Explicit On

' List of subroutines
'   InitializeAtmosphereLayers()
'   LookupAtmosphericStateVariables(returns ByRef)

Public Module USStandardAtmosphere

    ' ***** Definition of variables at the boundary altitudes *************************
    Private AltAtBotOfLayer(6) As Double
    Private AltAtTopOfLayer(6) As Double
    Private TempAtBotOfLayer(6) As Double
    Private PresAtBotOfLayer(6) As Double

    ' ***** Definition of lapse rates ************************************************
    Private LapseRateInLayer(6) As Double
    Private LapseExponentInLayer(6) As Double

    ' ***** Physical constants *******************************************************
    ' g0 = Local gravitational constant, in m/s^2
    ' Rair = Ideal Gas Constant for air, in J/kg-degK
    ' EarthRadius = Radius of the Earth, to mean sea level, in meters
    ' DViscCoef1 = Dynamic viscosity coefficient #1, units to suit
    ' DViscCoef2 = Dynamic viscosity coefficient #2, in degK
    ' GAMMAair = Ratio of specific heats for air, Cp/Cv
    Private g0 As Double = 9.80665
    Private Rair As Double = 287.053
    Private EarthRadius As Double = Val("6356766")
    Private DViscCoef1 As Double = Val("1.458E-6")
    Private DViscCoef2 As Double = 110.4
    Private GAMMAair As Double = 1.4

    Public Sub InitializeAtmosphereLayers()
        ' Set sea level conditions
        TempAtBotOfLayer(0) = 15 + 273.15
        PresAtBotOfLayer(0) = 101325
        ' Set altitude of boundaries between layers, in meters
        AltAtBotOfLayer(0) = 0 : AltAtTopOfLayer(0) = 11000
        AltAtBotOfLayer(1) = 11000 : AltAtTopOfLayer(1) = 20000
        AltAtBotOfLayer(2) = 20000 : AltAtTopOfLayer(2) = 32000
        AltAtBotOfLayer(3) = 32000 : AltAtTopOfLayer(3) = 47000
        AltAtBotOfLayer(4) = 47000 : AltAtTopOfLayer(4) = 51000
        AltAtBotOfLayer(5) = 51000 : AltAtTopOfLayer(5) = 71000
        AltAtBotOfLayer(6) = 71000 : AltAtTopOfLayer(6) = 85000
        ' Set lapse rates, in degrees Kelvin per meter
        LapseRateInLayer(0) = -6.5 / 1000
        LapseRateInLayer(1) = 0
```

```
    LapseRateInLayer(2) = 1 / 1000
    LapseRateInLayer(3) = 2.8 / 1000
    LapseRateInLayer(4) = 0
    LapseRateInLayer(5) = -2.8 / 1000
    LapseRateInLayer(6) = -2 / 1000
    ' Set lapse "exponents" and temperatures at the boundary altitudes
    For I As Int32 = 0 To 6 Step 1
        ' Calculate the lapse "exponent" in this layer
        If (LapseRateInLayer(I) = 0) Then
            LapseExponentInLayer(I) = -g0 / (Rair * TempAtBotOfLayer(I))
        Else
            LapseExponentInLayer(I) = -g0 / (Rair * LapseRateInLayer(I))
        End If
        ' Calculate the temperature at the top of layers #0 through #5
        If (I <= 5) Then
            Dim lThicknessOfLayer As Double = _
                AltAtTopOfLayer(I) - AltAtBotOfLayer(I)
            TempAtBotOfLayer(I + 1) = TempAtBotOfLayer(I) + _
                (lThicknessOfLayer * LapseRateInLayer(I))
        End If
    Next I
    ' Set pressures at the boundary altitudes
    For I As Int32 = 0 To 5 Step 1
        Dim lThicknessOfLayer As Double = _
            AltAtTopOfLayer(I) - AltAtBotOfLayer(I)
        Dim lTemp As Double
        If (LapseRateInLayer(I) = 0) Then
            lTemp = LapseExponentInLayer(I) * lThicknessOfLayer
            PresAtBotOfLayer(I + 1) = _
                PresAtBotOfLayer(I) * Math.Exp(lTemp)
        Else
            lTemp = TempAtBotOfLayer(I + 1) / TempAtBotOfLayer(I)
            PresAtBotOfLayer(I + 1) = _
                PresAtBotOfLayer(I) * (lTemp ^ LapseExponentInLayer(I))
        End If
    Next I
End Sub

Public Sub LookupAtmosphericStateVariables( _
    ByVal lGeometricAlt As Double, _
    ByRef lLocalTemp As Double, _
    ByRef lLocalPres As Double, _
    ByRef lLocalDens As Double, _
    ByRef lLocalDVisc As Double, _
    ByRef lLocalKVisc As Double, _
    ByRef lLocalSpeedOfSound As Double, _
    ByRef lGeopotentialAlt As Double)
    Dim lLayer As Int32             ' Layer which contains lGeometricAlt
    Dim lDifferenceInAlt As Double  ' Distance to the bottom of its layer
    Dim lTemp As Double             ' A temporary variable
    ' Convert the geometric altitude to a geopotential altitude
    lGeopotentialAlt = EarthRadius * _
        (1 - (EarthRadius / (EarthRadius + lGeometricAlt)))
    ' Check that the given lGeometricAlt is within the range of data
    If (lGeopotentialAlt < 0) Then
        MsgBox( _
            "The given altitude of " & Trim(Str(lGeometricAlt)) & _
            " meters is below sea level." & vbCrLf & _
```

```
                "No calculations will be done.")
            Exit Sub
        End If
        If (lGeopotentialAlt > AltAtTopOfLayer(6)) Then
            MsgBox( _
                "The given altitude of " & Trim(Str(lGeometricAlt)) & _
                " meters lies above Layer #6 in the model atmosphere." & vbCrLf & _
                "No calculations will be done.")
            Exit Sub
        End If
        ' Determine which layer the lGeopotentialAlt lies within
        For I As Int32 = 0 To 6 Step 1
            If (lGeopotentialAlt <= AltAtTopOfLayer(I)) Then
                lLayer = I
                Exit For
            End If
        Next I
        ' Calculate the temperature at this lGeopotentialAlt
        lDifferenceInAlt = lGeopotentialAlt - AltAtBotOfLayer(lLayer)
        lLocalTemp = _
            TempAtBotOfLayer(lLayer) + (lDifferenceInAlt * LapseRateInLayer(lLayer))
        ' Calculate the pressure at this lGeopotentialAlt
        If (LapseRateInLayer(lLayer) = 0) Then
            lTemp = LapseExponentInLayer(lLayer) * lDifferenceInAlt
            lLocalPres = _
                PresAtBotOfLayer(lLayer) * Math.Exp(lTemp)
        Else
            lTemp = lLocalTemp / TempAtBotOfLayer(lLayer)
            lLocalPres = _
                PresAtBotOfLayer(lLayer) * (lTemp ^ LapseExponentInLayer(lLayer))
        End If
        ' Calculate the density at this lGeopotentialAl
        lLocalDens = lLocalPres / (Rair * lLocalTemp)
        ' Calculate the dynamic viscosity at this lGeopotentialAlt
        lLocalDVisc = DViscCoef1 * (lLocalTemp ^ 1.5) / (lLocalTemp + DViscCoef2)
        ' Calculate the kinematic viscosity at this lGeopotentialAlt
        lLocalKVisc = lLocalDVisc / lLocalDens
        ' Calculate the local speed of sound
        lLocalSpeedOfSound = Math.Sqrt(GAMMAair * lLocalPres / lLocalDens)
    End Sub

End Module
```

Listing of the calling form

```
Option Strict On
Option Explicit On

Public Class Form1
    Inherits System.Windows.Forms.Form

    Public Sub New()
        InitializeComponent()
        With Me
            Text = "U.S. Standard Atmosphere 1976"
            FormBorderStyle = Windows.Forms.FormBorderStyle.None
            Size = New Drawing.Size(600, 600)
```

```
        MinimizeBox = True
        MaximizeBox = True
        FormBorderStyle = Windows.Forms.FormBorderStyle.Fixed3D
        With Me
            Controls.Add(labelAltitude) : labelAltitude.BringToFront()
            Controls.Add(textboxAltitude) : textboxAltitude.BringToFront()
            Controls.Add(buttonLookUp) : buttonLookUp.BringToFront()
            Controls.Add(buttonExit) : buttonExit.BringToFront()
        End With
        Visible = True
        PerformLayout()
    End With
    ' Call the first subroutine as soon as the form loads
    InitializeAtmosphereLayers()
End Sub

'////////////////////////////////////////////////////////////////////////////////
'// Controls and handlers
'////////////////////////////////////////////////////////////////////////////////

Private labelAltitude As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(100, 20), _
     .Location = New Drawing.Point(5, 5), _
     .Text = "Enter true altitude (meters)"}

Private textboxAltitude As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(50, 20), _
     .Location = New Drawing.Point(110, 5), _
     .Text = ""}

Private WithEvents buttonLookUp As New Windows.Forms.Button With _
   {.Size = New Drawing.Size(155, 30), _
    .Location = New Drawing.Point(5, 30), _
    .Text = "Look up state variables"}

Private Sub buttonLookUp_Click() Handles buttonLookUp.Click
    Dim lGeometricAlt, lGeopotentialAlt As Double
    Dim lT, lP, lD, lDV, lKV, lSS As Double
    ' Read the altitude from the textbox
    lGeometricAlt = Val(textboxAltitude.Text)
    ' Call the second subroutine for the user-supplied altitude
    LookupAtmosphericStateVariables( _
        lGeometricAlt, lT, lP, lD, lDV, lKV, lSS, lGeopotentialAlt)
    ' Display the results
    MsgBox( _
        "Geometric altitude (m) = " & Trim(Str(lGeometricAlt)) & vbCrLf & _
        "Temperature (degK) = " & Trim(Str(lT)) & vbCrLf & _
        "Temperature (degC) = " & Trim(Str(lT - 273.15)) & vbCrLf & _
        "Pressure (N/m^2) = " & Trim(Str(lP)) & vbCrLf & _
        "Density (kg/m^3) = " & Trim(Str(lD)) & vbCrLf & _
        "Dynamic viscosity (kg/m.sec) = " & Trim(Str(lDV)) & vbCrLf & _
        "Kinematic viscosity (m^2/sec) = " & Trim(Str(lKV)) & vbCrLf & _
        "Speed of sound (m/sec) = " & Trim(Str(lSS)) & vbCrLf & _
        "Geopotential altitude (m) = " & Trim(Str(lGeopotentialAlt)))
End Sub

Private WithEvents buttonExit As New Windows.Forms.Button With _
   {.Size = New Drawing.Size(155, 30), _
```

```
        .Location = New Drawing.Point(5, 65), _
        .Text = "Exit"}

    Private Sub buttonExit_Click() Handles buttonExit.Click
        Application.Exit()
    End Sub

End Class
```