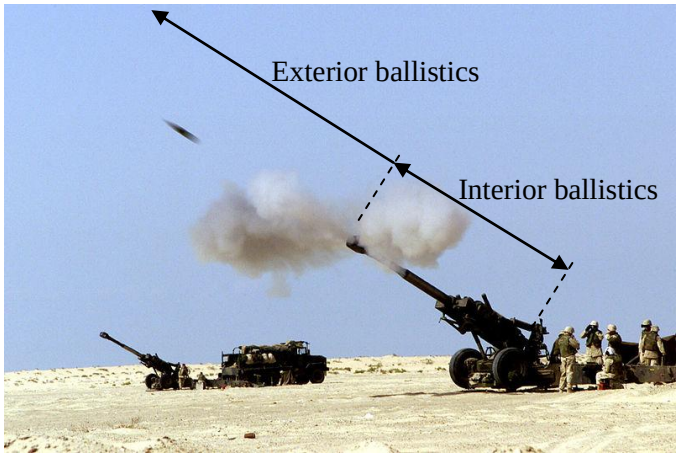
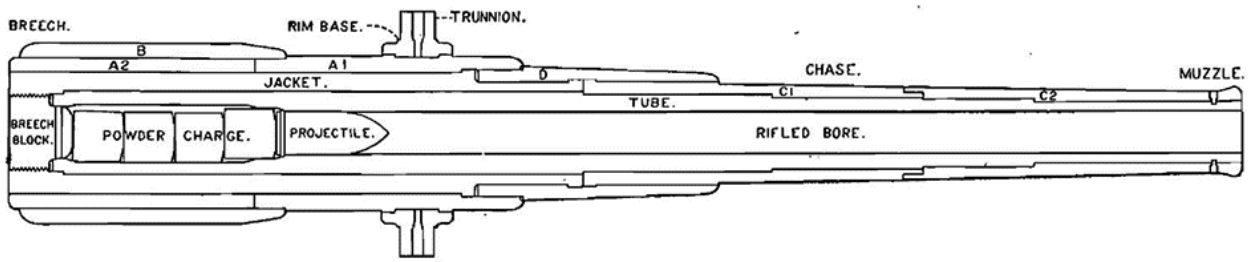
Interior ballistics of a large naval gun or artillery piece

Interior ballistics describes what happens inside the barrel. The behaviour of the gas released as the propellant burns lies at the heart of the problem.



The following cross-section of a 12-inch gun was taken from *Ordnance and Gunnery: A textbook prepared for the Cadets of the United States Military Academy*, by Ormon Lissak, 1907. The layout is typical of large naval guns and artillery pieces, even a century later.

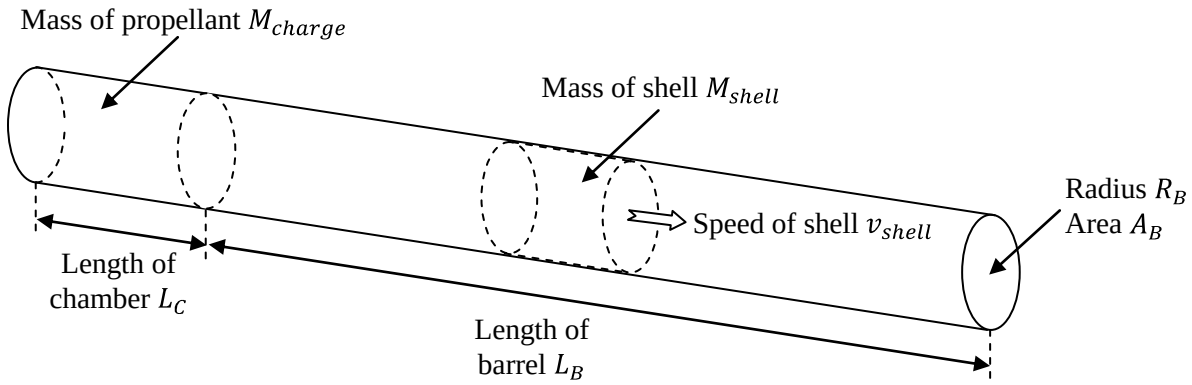


12-Inch Rifle, Model of 1900, 40 Calibers, 59.10 Tons.
(Diameters Exaggerated.)

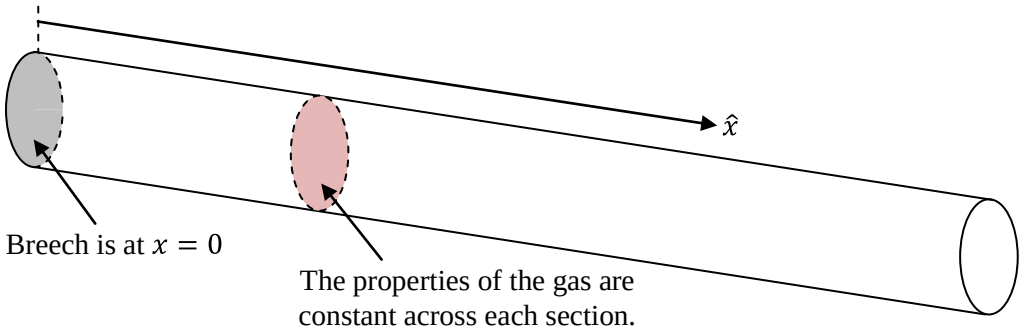
The projectile, which I will call the "shell", for this gun weighs about 1,130 pounds. The powder charge, which I will call the "propellant", weighs 360 pounds. For ease in manhandling, the charge is stowed in four silk bags each containing 90 pounds. After the shell is rammed home through the breech, the bags of propellant are thrown into the chamber. It can be seen that the propellant does not fill the chamber completely. The propellant for this gun is about 90% nitrocellulose. Nitrocellulose burns relatively slowly, so there are additives like nitroglycerin to help get the burning started quickly over the entire exposed surface area of the propellant. There are also chemical stabilizers and gelatinizing agents. The propellant is molded into small cylinders, called "grains". For this gun, the grains have a diameter of about 7/8-inch and a length of about 1-1/2 inch.

The physical shape and size of the grains is not a trivial matter. The objective of their geometry is to control, and particularly to slow down, the rate at which they burn. Contrary to popular belief, the propellant does not explode. Nor does it burn up quickly compared with the time it takes the shell to travel down the barrel. In the ideal case, the propellant would continue to burn during the entire time the shell remains inside the barrel. Indeed, it sometimes occurs that unburned propellant is ejected along with the shell.

The following figure shows the simplifications to the geometry I am going to make. Certain of the basic quantities are also defined.



I am going to analyze the gas in one-dimension, along the central axis of the gun. Distance along this axis will be measured by the x-co-ordinate, as shown in the following figure. I will assume that the properties of the gas are the same all across the circular cross-section at each longitudinal x-station. It is therefore convenient to assume that the chamber (in which the propellant is placed) has the same radius as the barrel, as is shown in the figure above.



It is apparent that the breech is at $x = 0$. At the time of ignition, the aft face of the shell is at the forward end of the chamber, that is, at $x = L_C$. The shell travels a distance L_B inside the barrel, and leaves the influence of the propellant when its aft face is at co-ordinate $x = L_C + L_B$. It may be that the rear end of the shell is tapered so there is a gap between the circumference at the rear end and the inside of the barrel while the shell is exiting. If so, that can be compensated for in the modeling by setting the length of the barrel L_B to some value a little less than the physical length of the barrel.

The mathematical model for firing this gun can be captured in seven relationships, relating to:

1. Conservation of mass of the gas
2. Conservation of momentum of the gas
3. Conservation of internal energy of the gas
4. Description of the heat generated as the propellant is burned
5. Dynamic equation governing the acceleration of the shell
6. Thermal equation of state for the gas, relating its physical parameters to temperature
7. Calorific equation of state for the gas, relating its physical properties to energy

I will deal with each relationship in a separate section.

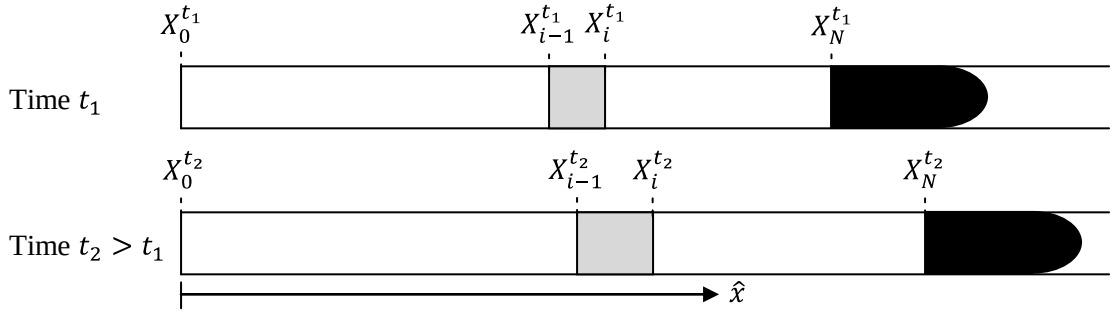
Part 1 – Conservation of mass of the gas

The bags containing the grains of propellant are cylindrical. Typically, a flat pancake made from another kind of explosive is inserted into a pocket at each end of the bag. The gun is fired by igniting these flat pancakes, which are used because they are easier to ignite than the propellant itself. Since the objective of the pancakes is to initiate burning on the entire surface area of all the grains in all the bags, there will be a great commotion during ignition. The pressure inside the chamber will rise sharply. Typically, the shell will be held back until the pressure reaches a certain value, determined by the engraving band, after which the shell will be allowed to begin accelerating down the barrel.

I have given this description of the ignition to justify the assumption I will make about the location of the propellant grains. I will assume that, at all times, the grains of propellant are uniformly distributed throughout the volume between the breech and the rear face of the shell. In other words, the commotion inside the barrel jostles the grains in such a way that they do not remain inside the chamber once the shell begins moving. They move apart from one another, taking advantage of the increased volume as it becomes available.

This assumption allows us to build conservation of mass into the behaviour of the gas right from the outset. Here is what I will do. I will divide the volume occupied by the gas at any instant of time into N (a large number) of "elements", each of which will be a thin disk. Initially, the N thin disks will have the same thickness and, since the length of the chamber is L_C, that thickness will be equal to L_C/N. Since the total original mass of the propellant is M_{charge}, the mass of (unburned) propellant initially contained inside each thin disk will be equal to M_{charge}/N.

I am going to arrange things so that the <u>total</u> mass contained inside each thin disk does not change with time. It remains constant at M_{charge}/N. The proportion between unburned propellant and gas molecules will change as time progresses, but the total mass will remain unchanged. What does change with time is the x-locations of the faces of the disks. The numbering scheme I propose to use is illustrated in the following figure. I will use the symbol X_i^t for the x-location of the shell-end face of the i^{th} element at a particular time t. The figure shows the locations of the faces of element #i at time t_1 and again at a later time t_2. This i^{th} element is shaded in grey. Unless both faces moved by the same distance from time t_1 to time t_2, the volume of element #i will have changed. The important thing is that we will cause the equations of motion to move the faces so that the same gas is always present inside element #i. Hence, mass will be conserved.



It follows from these definitions that the location of the breech end of the 1st element, that is X_0^t, will be zero at all times t. It will not move. The barrel end of the last element, that is element #N, is located at X_N^t. At all times, this will be the rear face of the shell.

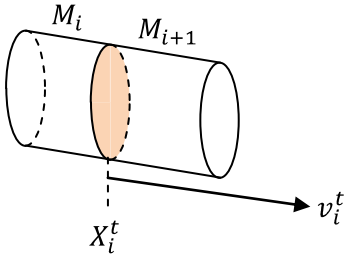
Part 2 – Conservation of momentum of the gas

In this section, I will look at conservation of momentum of the gas. Taken along with conservation of mass from the previous section, the two ideas will ensure the conservation of the gas's kinetic energy. If we were considering potential energy in our model of the gun, we would include the potential energy of the gas here, along with its kinetic energy. Even if the barrel of the gun is not horizontal, though, the potential energy of the gas is insignificant compared with its kinetic energy. I mention the potential energy just to put it into its proper place. This Part 2 deals with the energy of the gas's motion and location; Part 3 will deal with its internal energy.

As we get into this discussion, we are going to be faced with a bit of a problem about how to describe the locations of things. Some quantities of interest, like the mass of the gas, which relate to a whole element, are best treated as if they exist or apply at the physical "center" of the elements. Other quantities, like the location of each element, are best expressed by reference to the faces which are the boundaries of the element. In essence, this is a question of how we discretize phenomena which we would prefer to model mathematically in a continuous and differential form. I am going to skirt around the problem by developing the equation of motion directly in discrete form.

Consider the boundary face between elements #i and #$i+1$. (I will always assume that the breech end of the gun lies to our left, the shell exits to the right, and the location indices are numbered from left to right.) In accordance with our numbering scheme, this boundary face is located at x-co-ordinate X_i^t at time t. I will use the symbol v_i^t for the speed of this boundary face at time t. (I will reserve the capital letter V for volumes and use the small letter v for speeds.) We will have solved the equation of motion for the gas if we can calculate the speeds at all the boundary faces.

To the left of this boundary face is element #i, with total mass M_i. To the right lies element #$i+1$, with total mass M_{i+1}. The situation look like this.



Notionally, we are going to assign some mass to the boundary face shaded in pink. Suppose we assign to this face half of the mass of the element to the left and half of the mass of the element to the right. So long as we make the right adjustments to the left-most face (at the breech) and right-most face (at the shell), and assign only half masses to them, this approximation scheme will approach perfection as the size of the elements is reduced.

The mechanical momentum (mass multiplied by speed) of this "massy" boundary face can be written as:

$$momentum_i = \left(\frac{M_i + M_{i+1}}{2}\right) v_i^t$$

Since the masses are constant with respect to time, per Section 1, we can write the rate-of-change of this momentum as:

$$\frac{dmomentum_i}{dt} = \left(\frac{M_i + M_{i+1}}{2}\right) \frac{dv_i^t}{dt}$$

According to Newton's Laws, this rate-of-change of momentum will be equal to the net force which acts on this massy boundary face. Since I have set up the distance and speed convention as increasing towards the right, a net force will be algebraically positive if it also acts towards the right. Therefore:

$$Net\ force_i = \left(\frac{M_i + M_{i+1}}{2}\right)\frac{dv_i^t}{dt}$$

Since we are conducting a one-dimensional analysis, we need concern ourselves only with forces which act parallel to the long axis of the gun. There is gas to the left and gas to the right, so the only force will be exerted by static pressure. The force acting on the left side of this massy face will be the pressure inside element #i, say P_i, multiplied by the area of the face. The force acting on the right side of this massy face will be the pressure inside element #$i+1$, P_{i+1}, again multiplied by area A_B. This latter pressure acts towards the left, however, tending to decelerate the massy face. Under the influence of these pressures, the massy face will accelerate in accordance with:

$$\left(\frac{M_i + M_{i+1}}{2}\right)\frac{dv_i^t}{dt} = \left(P_i^t - P_{i+1}^t\right)A_B \quad (1)$$

Note that I have added the superscript t to the pressures to denote that these are their values at time t. No such superscript is needed on the masses, since the total mass inside each element remains constant.

In some situations, a numerical integration of this expression would work out just fine. In our situation, I expect that the process will fail, with the failure manifesting itself in infinite speeds, negative pressures and other such impossible results. Since the gas inside the barrel is compressed under tremendous and rapidly changing pressures, shock fronts will almost certainly form. There will be sharp changes in the pressure, density and temperature of the gas over short distances. Such discontinuities are shock waves.

A remedy was proposed in 1950 by Messrs. Richtmyer and von Neumann. They proposed to add a kind of additional pressure. They formulated the added pressure in such a way that it would dissipate some of the gas's energy, just like a shock wave does. From the point-of-view of the numerical integration, the effects of a discontinuity would then be spread over several adjacent elements in the discretized model of the gas. The arbitrary nature of this remedy was reflected in the name they gave the added pressure, "artificial viscosity". Like all viscosities, the added pressure term removes energy from the gas at a microscopic level.

The added pressure is artificial, but not wholly arbitrary. There is a difference. Equation (1) is not an adequate model of the gas's behaviour when it becomes supersonic, so some correction must be made. The need for a correction exists and making a correction is not arbitrary. But there are different ways to implement the correction. The way I am going to implement the correction is this. Letting W_i^t be the artificial viscosity inside the i^{th} element at time t, I will revise the equation of motion to be:

$$\left(\frac{M_i + M_{i+1}}{2}\right)\frac{dv_i^t}{dt} = \left(P_i^t - P_{i+1}^t\right)A_B + \left(W_i^t - W_{i+1}^t\right)A_B \quad (2)$$

where the discretized expression for W_i^t is:

$$W_i^t = \begin{cases} +C_1\rho_i^t\left(v_i^t - v_{i-1}^t\right)^2 + C_2\rho_i^t S_i^t\left|v_i^t - v_{i-1}^t\right| & if\ v_{i-1}^t > v_i^t \\ 0 & if\ v_{i-1}^t \le v_i^t \end{cases} \quad (3)$$

In this expression, ρ_i^t is the density of the gas inside element #i and S_i^t is the local speed of sound there, both at time t. C_1 and C_2 are constants chosen to assist the realism of the new terms.

Aside: Heuristic derivation of an appropriate form for *W*

As I say, different forms can be used for the artificial viscosity. The form I have chosen makes intuitive sense to me, as the following heuristic derivation shows. Consider two finite masses M_1 and M_2 which have initial speeds v_1 and v_2, respectively. M_1 is on the left. If $v_2 > v_1$, the two masses will never hit each other. A collision will occur only if the mass on the left, M_1, is travelling faster than M_2 and eventually overtakes it. This is analogous to our gaseous situation if the gas is being compressed faster than it is being accelerated, causing some elements to overtake their counterparts further down the barrel.

If the collision between the two masses is elastic, then they will bounce off each other without any loss in their combined kinetic energy. This is analogous to our gaseous situation as described in Equation (1) which is, in effect, the "elastic" version of encounters between gas elements in the barrel. On the other hand, an inelastic collision between masses M_1 and M_2, in which the total kinetic energy is reduced, becomes analogous to our gaseous situation when our goal is to incorporate some form of energy dissipation. So, let's look more closely at inelastic collisions between M_1 and M_2.

First, recognize that we do not know how much inelasticity to build into the collision. Is a lot of the initial total kinetic energy lost, or just a little bit? The way to handle this uncertainty is to build in some flexibility. Let's assume that a little bit of mass M_1, in an amount δM_1, encounters a little bit of mass M_2, in an amount δM_2. Let's assume that the collision between δM_1 and δM_2 is perfectly inelastic. They stick together, at some final speed v. This final speed can be calculated using simple mechanics. Their combined initial and final momenta are:

$$\begin{aligned} Initial\ momentum &= v_1\,\delta M_1 + v_2\,\delta M_2 \\ Final\ momentum &= v(\delta M_1 + \delta M_2) \end{aligned}$$

The initial and final momenta must be the same whether the collision is elastic or not. Equating them, we get:

$$v = \frac{v_1\,\delta M_1 + v_2\,\delta M_2}{\delta M_1 + \delta M_2}$$

Now, let's notionally separate the two component submasses δM_1 and δM_2, both now moving at speed v, and recombine them with their parent masses. Physically, this is quite close to what happens with our gaseous elements in the gun. The masses are adjacent elements on either side of a boundary face, and they are in constant contact with one another. After recombining the prodigal submasses, the final momenta of the parent masses are as follows:

$$\begin{aligned} Final\ momentum\ of\ M_1 &= v_1(M_1 - \delta M_1) + v\,\delta M_1 \\ Final\ momentum\ of\ M_2 &= v_2(M_2 - \delta M_2) + v\,\delta M_2 \end{aligned}$$

Subtracting the parents' initial momenta gives the following changes in momentum:

$$\begin{aligned} Change\ in\ momentum\ of\ M_1 &= \delta M_1(v - v_1) \\ Change\ in\ momentum\ of\ M_2 &= \delta M_2(v - v_2) \end{aligned}$$

Substituting the expression for speed v into the change in momentum of the first mass gives:

$$\begin{aligned} Change\ in\ momentum\ of\ M_1 &= \delta M_1 \left(\frac{v_1\,\delta M_1 + v_2\,\delta M_2}{\delta M_1 + \delta M_2} - v_1 \right) \\ &= \delta M_1 \frac{(v_1\,\delta M_1 + v_2\,\delta M_2 - v_1\,\delta M_1 - v_1\,\delta M_2)}{\delta M_1 + \delta M_2} \\ &= \frac{\delta M_1 \delta M_2}{\delta M_1 + \delta M_2}(v_2 - v_1) \end{aligned}$$

Similar algebra for the second mass shows that its change in momentum is:

$$Change\ in\ momentum\ of\ M_2 = -\frac{\delta M_1 \delta M_2}{\delta M_1 + \delta M_2}(v_2 - v_1)$$

which is equal in magnitude but opposite in sign, as one would expect.

Let's start to apply these changes in momentum to the gas elements in our gun. First, let's figure out what we want to represent the so-called parent masses. Since we have already chosen to use boundary faces between elements as the loci for measuring speeds, let's continue to do so. Then, v_2 would represent the speed of the right-hand boundary face of some particular element and v_1 would represent the speed of its left-hand boundary face. In talking about "massy" faces above, I have already assigned effective masses to both boundary faces. In other words, the masses which correspond to M_1 and M_2 are the masses of the massy boundary faces on the left side and right side, respectively, of this particular element. The interaction between the two submasses δM_1 and δM_2 therefore takes place at what is the center of this particular element. That is handy, because we will have to talk about the density of the gas (below), and the density we calculate will be the average of conditions throughout the element's volume. We will localize the density to the center of the element, right where the interaction between δM_1 and δM_2 takes place. For now, let's say that the density at the center of this particular element is ρ.

Secondly, recall that we chose the magnitude of the interacting masses δM_1 and δM_2 arbitrarily. There is nothing to prevent them from being the same, so that $\delta M_1 = \delta M_2$. If ρ is the density of the gas at the spot where these masses interact, then we can express the masses δM as the product of the density and a volume. An appropriate volume for this purpose could be a very thin circular cylinder, with the circle being the open barrel. The area of the cylinder is the same area of the barrel A_B we used above. The thickness of the cylinder can be chosen to suit. Let's say the thickness is δX. With these definitions, we can express the masses δM_1 and δM_2 in terms of the density ρ as follows:

$$\delta M_1 = \delta M_2 = \rho A_B \delta X$$

Having expressed the submasses in this way, the changes in the momenta of the parent masses are:

$$\left.\begin{aligned} Change\ in\ momentum\ of\ M_1 &= +\tfrac{1}{2}\rho A_B \delta X (v_2 - v_1) \\ Change\ in\ momentum\ of\ M_2 &= -\tfrac{1}{2}\rho A_B \delta X (v_2 - v_1) \end{aligned}\right\} \quad (4)$$

Remember that there will not be any collision at all unless $v_1 > v_2$. In that case, the change in momentum of M_1 will be algebraically negative (it loses momentum) and the change in momentum of M_2 will be algebraically positive (it gains momentum).

The thickness δX is still arbitrary. We can, without limiting anything we have done so far, express this distance as the product of a speed and a time. We already have a speed at hand, $v_1 - v_2$, which is the relative speed at which the submasses approach each other. There will be a corresponding time period, an "interaction time" τ, if you will, which can be used along with the relative speed to calculate a "depth" δX of the interaction. With these definitions, we can express the depth as:

$$\delta X = \tau(v_1 - v_2)$$

Since the interaction distance only makes physical sense if it is a positive number, this expression only makes sense if $v_1 > v_2$. Making the substitution into Equation (4), the changes in momenta can be written as:

$$\left.\begin{array}{ll} Change\ in\ momentum\ of\ M_1 & = +½\rho A_B(v_2 - v_1)(v_1 - v_2)\tau \\ Change\ in\ momentum\ of\ M_2 & = -½\rho A_B(v_2 - v_1)(v_1 - v_2)\tau \end{array}\right\} \quad (5)$$

Yes, I realize that the speed-difference terms can be combined as squares. And, in due course I will combine them, but not just yet. I am not ready to pick which one of v_1^2 or v_2^2 should come first in the subtraction.

What do we get if we divide both sides of one of these equations by the interaction time τ? The result is the rate-of-change in the momentum during the interaction period. That sounds like exactly the kind of artificial adjustment we hoped to make to the equation of motion (1), namely, a dissipation of energy during a short interaction time. Executing the division on both equations, we get:

$$\left.\begin{array}{ll} Rate\text{–}of\text{–}change\ \ in\ momentum\ of\ M_1 & = +½\rho A_B(v_2 - v_1)(v_1 - v_2) \\ Rate\text{–}of\text{–}change\ in\ momentum\ of\ M_2 & = -½\rho A_B(v_2 - v_1)(v_1 - v_2) \end{array}\right\} \quad (6)$$

If the particular element we have been looking at is the i^{th} element, the relevant density in the discretized versions of these equations would be denoted by ρ_i or, since the density changes with time, then as ρ_i^t at time t. Speed v_2 corresponds to the speed of the right-hand boundary face of this element, v_i^t. Speed v_1 corresponds to the speed of the left-hand boundary face of this element, v_{i-1}^t. Combining the factor ½ into a more general constant C_1 means that the artificial viscosity to be added to the right-hand side of Newton's equation (1) has the form:

$$W_i^t A_B = +C_1\rho_i^t\left(v_i^t - v_{i-1}^t\right)\left(v_{i-1}^t - v_i^t\right)A_B \quad (7)$$

This will always be algebraically negative. But the whole concept only applies if $v_1 > v_2$ or, in terms of the speeds at the element's boundary faces, $v_{i-1}^t > v_i^t$.

Observe that Equation (7) corresponds to the first of the two terms in my expression for the artificial viscosity in Equation (3). But, there is a second term in Equation (3) as well. Here's why. The correction (that is, dissipation of energy) in Equation (7) is said to "capure strong shocks and prevent zone inversions", but can still leave "unphysical oscillations" behind the shock front. A second term, similar in form but only linear in the speed-difference term $v_{i-1}^t > v_i^t$, is sometimes added to deal with this. Relative speeds get higher as one approaches a shock front, so a term which is linear in the relative speed rather than squared will decrease less rapidly as one gets further away from the shock front. In other words, a linear correction term would extend its ameliorating influence further away from the shock front than does Equation (7).

A good place to begin is back at Equation (4). In Equation (4), the change in momentum due to inelastic, or viscous, effects was expressed in terms of an interaction distance δX. In the first run-through, I expressed the interaction distance in terms of the local relative speed $v_1 - v_2$ muliplied by an interaction time τ. There are other ways to drum up estimates for interaction distances. For example, one could use the product of the local speed of sound S and the interaction time τ. Just like the density ρ, the local speed of sound is a quantity that we can deal with on an element-centered basis. I will set aside for the moment the question of how we go about calculating the speed of sound, and just assume that we know its value. The interaction distance can then be written as:

$$\delta X = \tau S$$

Working through the same steps as before, we arrive at the following expression for the rate-of-change in the momentum through the interaction zone:

$$W_i^t A_B = +C_2 \rho_i^t S_i^t \left(v_i^t - v_{i-1}^t\right) A_B \quad (8)$$

where S_i^t is the speed of sound at the center of element #i at time t and C_2 is a different constant for this different formulation of the artificial viscosity. This expression will also be algebraically negative in the circumstance of interest to us, when $v_{i-1}^t > v_i^t$. Note that Equation (8) corresponds to the second of the two terms in my expression for the artificial viscosity in Equation (3).

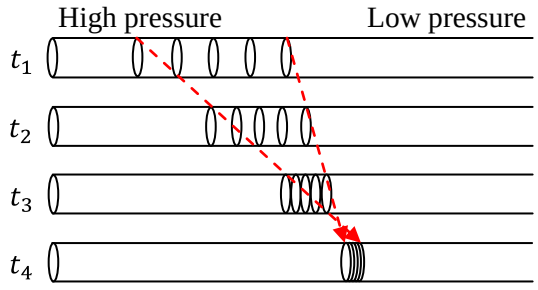
I believe the best way to add up the effects in Equations (7) and (8) to avoid continuing confusion over algebraic signs is as follows:

$$W_i^t = -C_1 \rho_i^t \left(v_i^t - v_{i-1}^t\right)^2 - C_2 \rho_i^t S_i^t \left|v_i^t - v_{i-1}^t\right| \quad (9)$$

All the factors are guaranteed to be positive and the dissipative nature of the correction shows up in the explicit minus signs. As always, the dissipation occurs only when the relative speeds dictate, namely, when $v_{i-1}^t > v_i^t$.

But, this is not quite the end of the story. The story so far has tracked through what the gas would do if the elements interacted in an inelastic manner. Our problem is the reverse of this. What we really want to do is to artificially introduce the effect of inelasticity, so that our otherwise perfectly colliding elements respond as if they experienced inelastic collisions. We need to add the reverse of the inelastic effects described in Equation (9). It is for that reason that the addition I made to adjust Equation (2) is the negative of Equation (9) and not Equation (9) itself.

Let me explain this using a different approach. The following figure shows four successive elements in the barrel which are subject to a gradient of static pressure which is high on the left-side and low on the right-side. I have shown the elements at four successive times.



Under the influence of the static pressure only, the boundary end-faces of the elements at the high pressure end will be accelerated more quickly than the end-faces at the low pressure end. The high pressure faces will overtake their counterparts at the low pressure end. If the time steps of the numerical integration are "too long" in duration, or the elements "too big" in size, it can happen that the high pressure faces pass right through the low pressure ones. Obviously, we cannot permit that.

The purpose of artificial viscosity is to add some extra, artificial, pressure inside those elements which are being compressed most vigorously. The extra pressure inside the elements which are being squeezed retards the advance of the elements, and their boundary end-faces, which are crushing in from the left. The "end" of this act is to increase the pressure inside the elements which are being squeezed; the "means" by which this is done is to use the viscosity concept to slow down the on-coming end-faces. They are two sides of the same coin.

An element which is being squeezed is one whose left face, which is moving at speed v_{i-1}^t, is moving faster than its right face, which is moving at speed v_i^t. The relative speed at which the faces are closing on each other is $v_{i-1}^t - v_i^t$. When this is positive, the element is being compressed. The extra pressure needs to be added to the inside of this element. Hence the algebraic sign of the addition of artificial viscosity in Equation (3).

End of aside

We are not quite done with our equation of motion. Equation (2) still contains the derivative dv_i^t/dt, which needs to be discretized before we can use it in a numerical procedure. Suppose the time step used in the numerical integration is ΔT. The usual way to discretize the derivative would be to make the approximation:

$$\frac{dv_i^t}{dt} = \frac{v_i^{t+\Delta T} - v_i^t}{\Delta T}$$

I am going to make a slightly different approximation. I am going to center the derivative on time t by offsetting the speeds used in the approximation one-half time step forwards and backwards, like this:

$$\frac{dv_i^t}{dt} = \frac{v_i^{t+½\Delta T} - v_i^{t-½\Delta T}}{\Delta T} \quad (10)$$

In the limit as ΔT shrinks to zero, the two approximations are the same. But, before ΔT gets all the way to zero, they are not quite the same. In practice, particularly when the time step is not as short as one would like, the second approximation avoids the lopsided bias built into the first approximation. Using the second approximation, our equation of motion can be written as:

$$v_i^{t+½\Delta T} = v_i^{t-½\Delta T} - \left(\frac{2\Delta T}{M_i + M_{i+1}}\right)\left[\left(P_{i+1}^t - P_i^t\right) + \left(W_{i+1}^t - W_i^t\right)\right]A_B \quad (11)$$

where W_{i+1}^t and W_i^t are still computed using Equation (3). In effect, this means that the speeds will be calculated at half-integral time steps, while all other quantities will be calculated at integral time steps. I believe that some practitioners call this method "leap-frogging".

Once the speeds have been calculated using Equation (11), the locations of the boundary faces can then be advanced through the next time step. For example:

$$X_i^{t+\Delta T} = X_i^t + \left(v_i^{t+½\Delta T}\Delta T\right) \quad (12)$$

The benefits of having calculated the speeds at half-integral time steps is a little clearer in Equation (12), where $v_i^{t+½\Delta T}$ enters as the speed at the middle of the time step, and not the speed at the start of the time step.

Once the locations of the boundary faces at time $t + \Delta T$ have been calculated using Equation (12), the total volume of each element can be computed, as follows:

$$V_i^{t+\Delta T} = \left(X_i^{t+\Delta T} - X_i^t\right) A_B \quad (13)$$

Equations (11) and (12) are the equations which ensure that mass and momentum are conserved. In our numerical procedure, we will use them to advance the speed and location variables from one time step to the next. Because of the special geometry of our gun, their use automatically advances the total volumes of the elements, too, via Equation (13).

Note that the variables which are the ingredients in these equations, on the right-hand side, are referenced to time t. Once we have solved for all quantities at time t, these three equations will be our first foray into time $t + \Delta T$.

Constants C_1 and C_2 must be specified before the artificial viscosities can be calculated. My heuristic derivation suggests setting C_1 to a value of ½. If the speed of sound is expected to be greater than the relative closing speeds of adjacent boundary faces, then the two viscosity approaches will make equal contributions if C_2 is set to a fraction of C_1. I observe that some investigators suggest starting with $C_1 = 1$ and $C_2 = 1/10$.

A word or two about the air inside the chamber

In the previous section, we dealt with the total mass of "stuff" inside the i^{th} element, and used the symbol M_i for it. It will remain constant during the entire firing process. Whatever was inside the i^{th} element when firing commenced stayed inside that element for the duration of the entire process.

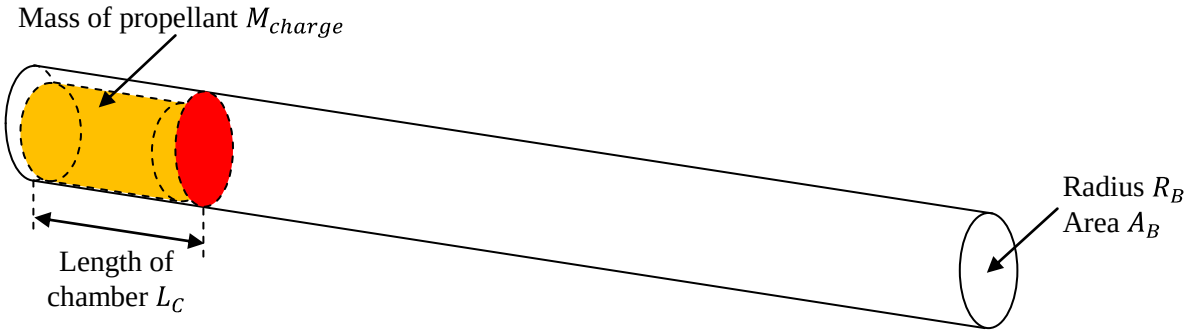
The "stuff" inside each element consists of three things. It consists of unburned propellant, the gas produced by the propellant which has been burned and, not to be forgotten, is the ambient air which was inside the chamber when the bags of propellant were placed inside. The amount of ambient air is quite small compared to the amount of gas which will be generated when the propellant burns. But, its effect on the peak pressures which will be experienced during the process are not necessarily insignificant. When an element is severely compressed by its neighbouring elements, the pressure and density will increase dramatically. The increase will be particularly acute near shock fronts. It is best if we try to account for the things we do know, so we do not needlessly present ourselves with surprises.

The total mass inside element #i can be expanded as:

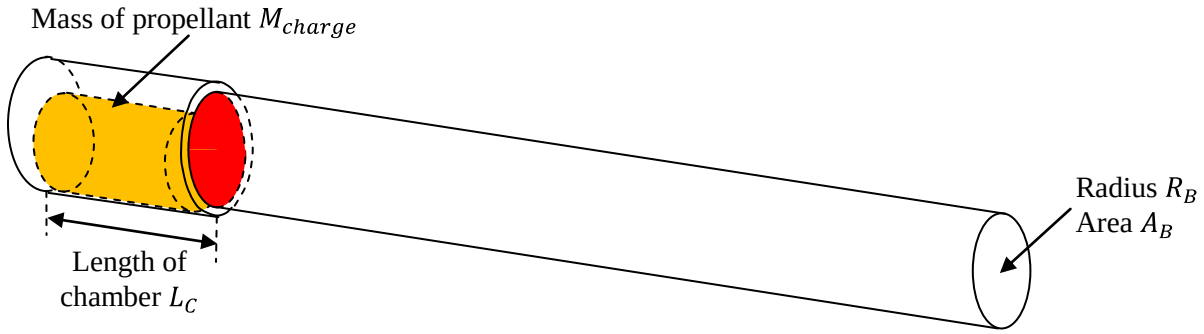
$$M_i = M_i^{Pt} + M_i^{Gt} + M_i^{Air} \quad (14)$$

where the superscripts record that these three terms represent the unburned Propellant, the propellant Gas and the original Air, respectively. The total mass M_i and its component of air M_i^{Air} will be constant with time and so do not need a "t" superscript. The total mass of unburned and burned propellant $M_i^{Pt} + M_i^{Gt}$ will also be constant, but the proportions will change. In fact, the numerical simulation is based on dividing the chamber into elements with the same size, so M_i, M_i^{Air} and the sum $M_i^{Pt} + M_i^{Gt}$ will actually be the same for all elements, as well as being constant with time.

The following figure shows the chamber just before firing, as I have modeled it. The equations are simpler when the entire barrel, including the chamber, has a constant cross-section. The red ellipse marks the rear face of the shell.



A more exact physical model would include chambrage – the fact that the chamber sometimes has a larger diameter than the barrel. This is shown in the following figure.



The difference between these two physical models is the proportion of the chamber's real volume which is occupied by solid propellant. This proportion is usually expressed in terms of the "loading density" ρ_0 of the propellant. This should not be confused with one's normal definition of density, which in this case is usually called the "crystalline density" ρ_c of the propellant. In the numerical simulation, I will use the following values:

$$\rho_c = 1660\ kg/m^3 \quad (15A)$$
$$\rho_0 = 680\ kg/m^3 \quad (15B)$$

Let me talk about the crystalline density first. Water has a density of $1000\ kg/m^3$, so propellant does not float. A given volume of this propellant weighs 66% more than an equivalent volume of water. In the numerical simulation, I will model a 5-inch naval gun which has the following parameters:

$$\left.\begin{array}{ll} M_{charge} & = 8.85\ kg \\ L_C & = 1.03\ m \\ A_B & = 0.0127\ m^3 \end{array}\right\} \quad (16)$$

The volume of the chamber in our model is the length of the chamber multiplied by the cross-sectional area of the barrel: $1.03 \times 0.0127 = 0.01308\ m^3$. The physical volume of the propellant is the total mass of propellant divided by its crystalline density: $8.85 \div 1660 = 0.00533\ m^3$. Solid propellant occupies $0.00533 \div 0.01308 = 40.75\%$ of the volume of the chamber in our model.

Now, let me talk about the loading density. $8.85\ kg$ of propellant is placed into a space in which it comprises $680\ kg$ per cubic meter of the available space. It follows that the available space must be

equal to $8.85 \div 680 = 0.01308\, m^3$. It is coincidence that the effective volume in this case happens to be equal to the volume of the chamber in our model. (Well, it is not a mere coincidence. The 5-inch gun I will be simulating has no chambrage – the chamber has the same diameter as the barrel.)

In our case, the 59.25% of the chamber's volume which is not filled with solid propellant will be filled with air. The actual volume filled with air will be $0.01308 - 0.00533 = 0.00775\, m^3$. More generally, including cases where the chambrage is non-zero, the physcial volume occupied by air would be calculated as follows:

$$\begin{aligned} Volume\ of\ air &= True\ volume\ of\ chamber - Volume\ of\ propellant \\ &= \frac{M_{charge}}{\rho_0} - \frac{M_{charge}}{\rho_c} \\ &= M_{charge}\left(\frac{1}{\rho_0} - \frac{1}{\rho_c}\right) \end{aligned} \qquad (17)$$

I will treat the air in the chamber as an ideal gas. We can use the Ideal Gas Law to determine how many moles of air will accompany the propellant. In order to use the Ideal Gas Law in this circumstance, we need to know the pressure and temperature. In the numerical simulation, I assume that the ambient pressure is $101{,}300\ N/m^2$ (standard atmosphere at sea level) and that the temperature is $100°C$ (a little higher than standard, perhaps due to heat left over from a previous firing). The general formula and the particular result for the numerical simulation are:

$$\begin{aligned} &Ideal\ Gas\ Law: && PV = nRT \\ &General\ case: && Moles\ of\ air = \frac{P_{ambient} \times Volume\ of\ air}{R \times T_{ambient}} \\ &This\ gun: && Moles\ of\ air = \frac{101{,}300 \times 0.00775}{8.314 \times (100 + 273.15)} = 0.253 \end{aligned} \qquad (18)$$

The molar weight of (dry) air is $0.02897\ kg/mole$ which means that the mass of the air inside the chamber is $0.253 \times 0.02897 = 0.00733\, kg$. This is quite small compared to the $8.85\, kg$ of propellant, which be be gaseous if and when it has been completely burned. In any event, this $0.00733\, kg$ mass of air, when divided by the number of elements, gives the value M_i^{Air} for all elements. A similar division can be done for the number of moles of air to give N_i^{Air}, which I will refer to below as the number of moles of air inside each element.

Part 3 – Conservation of internal energy of the gas

I will use the symbol u_i^t for the internal energy density of the gas inside element #i at time t. u_i^t is the total material energy (U_i^t) of the gas in the element divided by the mass of that gas. There are different phenomena which can change the material energy. Energy can be conveyed from one element to another by radiation, the exchange of electrons, and so on, but I am going to ignore all causes of changes in internal energy except one. The one I am going to focus on is the mechanical work performed by the pressure as the volume of the element changes. The pressure which does this work is not just the static pressure P_i^t but includes the artificial viscosity W_i^t as well. Although it is called a viscosity, W_i^t affects the gas in exactly the same way as the static pressure. Their combined outward effect (from the point-of-view of an element pressing on its neighbours) leads to the equation of motion. Their combined inward effect (from the point-of-view of an element experiencing pressure exerted by its two neighbours) changes the internal energy of an element.

Consider the one-dimensional travel of a rigid body. If a force F causes the body to move through a distance ΔD, the mechanical work done on the object is the product $F \times \Delta D$. Similarly, if a pressure P causes the volume of a fixed amount of gas to expand by ΔV, the work done is the product $P \times \Delta V$. Some agent must do the work, of course. In our case, the agent driving the expansion is the internal energy of the gas. The work done as the volume increases reduces the internal energy of the gas. Over a given interval of time, the equality can be written as:

$$Increase\ in\ internal\ energy\ U = -Pressure \times Increase\ in\ volume \quad (19)$$

Since these are changes take place with respect to time, let's set a starting time t and an ending time $t + \Delta T$ for the interval. If we have solved the equations in Part 2 already, then we will know the volumes of each element at the start and end of this time interval. They will be V_i^t and $V_i^{t+\Delta T}$, respectively. Starting to make substitutions into Equation (19), we get:

$$U_i^{t+\Delta T} - U_i^t = -Pressure\left(V_i^{t+\Delta T} - V_i^t\right)$$
$$\rightarrow \quad u_i^{t+\Delta T} - u_i^t = -\frac{Pressure}{Mass\ of\ gas}\left(V_i^{t+\Delta T} - V_i^t\right) \quad (20)$$

What shall we do about the $Mass\ of\ gas$ inside element #i? It is not M_i. M_i is the <u>total</u> mass of material inside the element. Only some of it is a gas, capable of doing work by expansion. In the previous section, we separated the total mass into three components:

$$M_i = M_i^{Pt} + M_i^{Gt} + M_i^{Air} \quad (14)$$

where the two gaseous components are $M_i^{Gt} + M_i^{Air}$. It is this mass whose internal energy is changed by the change in volume.

Now, what shall we do about the $Pressure$? We know it is going to be a sum like $P_i + W_i$, but that sum changes with time. A suitable quantity for our purpose would be the simple average of the pressures at the start and the end of the interval. That will converge to perfection as the time intervals are made smaller and smaller. Therefore:

$$u_i^{t+\Delta T} - u_i^t = -\frac{1}{M_i^{Gt} + M_i^{Air}} \frac{\left[\left(P_i^{t+\Delta T} + W_i^{t+\Delta T}\right) + \left(P_i^t + W_i^t\right)\right]}{2}\left(V_i^{t+\Delta T} - V_i^t\right) \quad (21)$$

Oops, I have forgotten something very important. As the propellant burns, it releases heat. The addition of heat to the gas increases its internal energy. I am going to ignore the heat that is absorbed by the metal in the barrel, and assume that the gas absorbs all of the heat generated.

To be precise about things, we should treat the gas trapped inside the barrel as a closed system. We should use the First Law of Thermodynamics to revise Equation (19) to:

$$Increase\ in\ internal\ energy\ U = Heat\ added\ Q - (Pressure \times Increase\ in\ volume) \quad (19')$$

My first version of Equation (19) only included the $P\Delta V$ term, which is the mechanical work done. The correct version Equation (19′) includes the heat added. We can remedy Equation (21) by adding a term to represent the heat added during the t to $t + \Delta T$ time step. I will use the following symbol:

$$Heat\ added\ to\ element\ \#i\ from\ time\ t\ to\ time\ t + \Delta T = \Delta Q_i^{t,t+\Delta T} \quad (22)$$

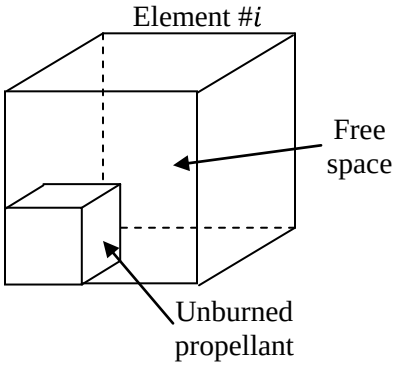
In due course (below), we will have to figure out how to calculate how much heat is added. For now, I will simply amend Equation (21) by including the heat term. The result is:

$$u_i^{t+\Delta T} = u_i^t + \frac{\Delta Q_i^{t,t+\Delta T} - \tfrac{1}{2}\left[\left(P_i^{t+\Delta T} + W_i^{t+\Delta T}\right) + \left(P_i^t + W_i^t\right)\right]\left(V_i^{t+\Delta T} - V_i^t\right)}{M_i^{Gt} + M_i^{Air}} \quad (23)$$

Equation (23) is our equation of conservation of internal energy. In the numerical procedure, we will use it to advance the internal energy from one time step to the next. Note that the variables which are the ingredients in this equation, on the right-hand side, include some which are referenced to time $t + \Delta T$. We will have to figure out (below) whether we can calculate the internal energy $u_i^{t+\Delta T}$ first or the pressure $P_i^{t+\Delta T}$ first or, alternatively, whether a simultaneous solution will be needed.

A word or two about the volume of a particular element

It is time to get more precise about what we mean by the "volume" of an element. We will be refering to volume in two senses. The absolute volume of element #i is the distance between its two end-faces multiplied by the cross-sectional area of the barrel. The other volume of interest is the volume occupied by gas. As long as there is unburned propellant, the volume occupied by gas will be less than the absolute volume of the element. The following figure highlights the difference from a conceptual point-of-view.



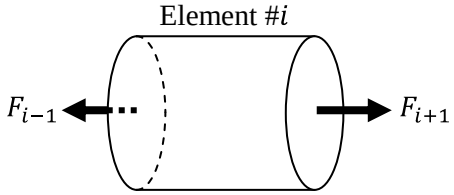
The absolute volume of element #i at time t is V_i^t. The mass of unburned propellant at time t is M_i^{Pt}. Dividing this mass by the crystalline density of the propellant ρ_c gives the absolute volume of the unburned propellant, M_i^{Pt}/ρ_c. The difference between these two volumes is the volume of free space which is available to the molecules of gas.

$$Volume\ of\ free\ space\ for\ gas = V_i^t - \frac{M_i^{Pt}}{\rho_c} \quad (24)$$

The molecules of gas consist of a mass M_i^{Gt} of propellant combustion products and the mass M_i^{Air} of original air. We will have occasion to talk about the number of moles of these two constituents. I have already mentioned N_i^{Air}, the number of moles of ambient air inside the element. The stoichiometric ratio for converting the propellant from solid to gas is this: burning one kilogram of propellant produces 40 moles of gas. Therefore:

$$N_i^{Gt} = 40 M_i^{Gt} \quad (25)$$

It is obvious that the process of burning changes the volume of free space which is available for gas. A question arises: does this change in volume contribute to the $P\Delta V$ mechanical work discussed in the previous section? My answer is no, it does not. I refer to the following figure in support of my answer.

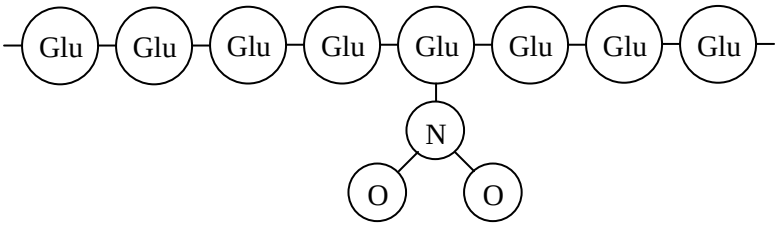


The mechanical work discussed above derived from the pressure acting uniformly across the two end-faces of the element. The pressures gave rise to forces on the neighbouring elements. If the end-faces moved, mechanical work was done. In the case of the combustion which occurs inside the element, the conversion of a certain volume of solid into an equivalent volume of gas did not arise from a displacement-due-to-pressure process. It did not constitute mechanical

work. Accordingly, the volumes refered to in Equation (23), which are the absolute volumes of the elements, are satisfactory as they stand.

Part 4 – Heat generated as the propellant is burned

Nitrocellulose is often called "gun cotton". Indeed, it usually made from cotton. Cotton is a polymer, made up of hundreds of glucose units bonded to one another in a long chain. The following diagram shows the glucose backbone of the chain.



The raw cotton is treated with sulphuric acid and nitric acid. The treatment is called "nitration" and it adds an NO_2 nitro group to each glucose unit. My diagram shows the nitro group on only one of the glucose units, but every glucose unit will have its own nitro group.

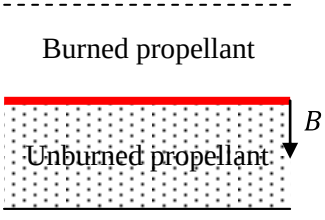
When the molecule is oxidized (that is, burns), the beauty of this arrangement becomes clear. The oxygen atoms needed during the oxidation process are physically nearby. The oxygen does not have to be extracted from the ambient air, and then mixed with molecules of fuel, as must happen in internal combustion engines. There are two important consequences: the combustion of nitrocellulose is fast and leaves little residue.

When used as a propellant, nitrocellulose is usually mixed with a smaller quantity of a similar compound, as well as gelatins and stabilizing agents. Nitroglycerin is often used as the additive. The resulting compound is easily molded or extruded and, when dried, makes ideal grains. Because there are two explosive compounds, both with their own combination of fuel and oxygen, these propellants are called "double-based" propellants. Furthermore, by adjusting the ratio of nitrocellulose and the additive, the "heat of explosion" or "calorimetric value" of a given mass of propellant can be varied. This has led to such propellants being called "cool" or "hot" or "extremely hot".

Propellants used in large naval guns are at the "cool" end of the range. The heat released when one gram of naval propellant is converted into a gas is about 820 calories. Hotter variants are used as solid rocket fuel. I am going to use the symbol Q_0 as the heat of explosion of the propellant for our gun. Converting from calories to Joules at the rate of $1\ calorie = 4.1868\ J$, we have:

$$Q_0 = 820\ \frac{cal}{g} = 3{,}430{,}000\ \frac{Joules}{kg} \qquad (26)$$

I am going to describe the rate of burning using a so-called "burn rate" B. I envision that burning is something that occurs uniformly over a surface. As the thin layer which is burning is consumed, and converted into gas, the burning surface eats its way down into the unburned propellant.



The burn rate B can be thought of as the speed, in meters per second, say, at which the burning surface penetrates the unburned propellant. In the diagram shown to the left, burning began on the top surface. The burning surface (rendered in red) is moving downwards at speed B.

The following graph is useful for our purpose. It shows the burn rate for double-base propellants of various warmth. The speed B is measured along the vertical axis; the static pressure being exerted on the burning surface is measured along the horizontal axis. Both axes are logarithmic.

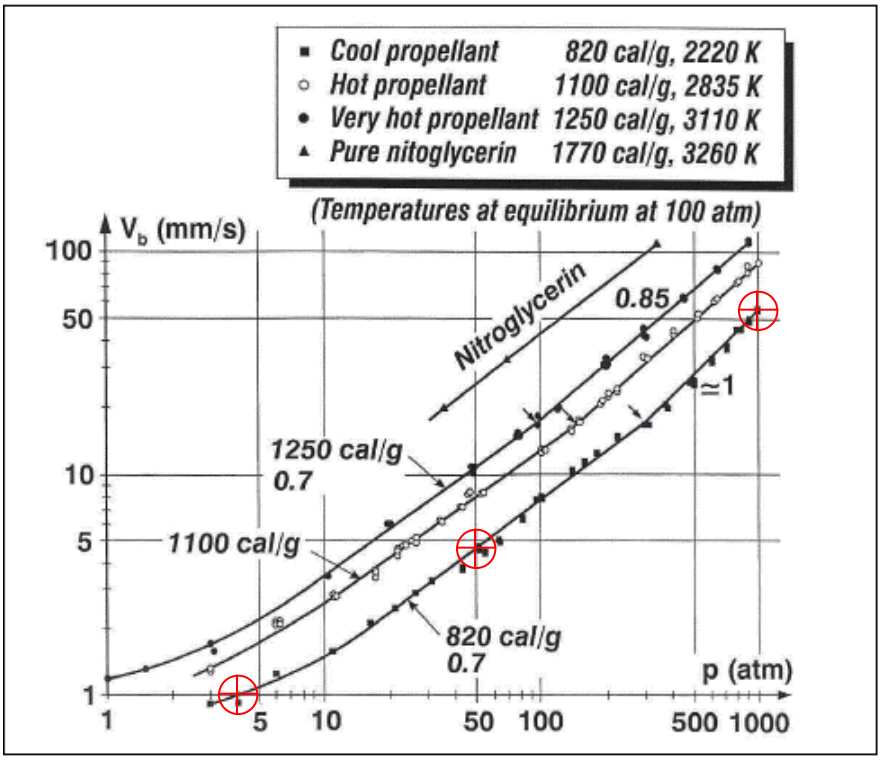


Figure 2: Burning Rate vs Pressure.
(Double Base Propellants)

I am going to use the "cool propellant" data points in the graph as representative of our propellant. Its heat of explosion is $820\ cal/g$. I am going to fit a quadratic curve to these data points. The proposed equation has the form:

$$\ln V_b = a(\ln P)^2 + b(\ln P) + c$$

To calculate the coefficients a, b and c, I have selected three particular data points, which are idenified on the graph with red crosshairs. The co-ordinates of the three points and the solutions for the coefficients are as follows:

$\ln P$	$\ln V_b$
4	1
50	4.5
1000	58

$$\begin{aligned} a &= 0.046696597 \\ b &= 0.34808898 \\ c &= -0.572295873 \end{aligned}$$

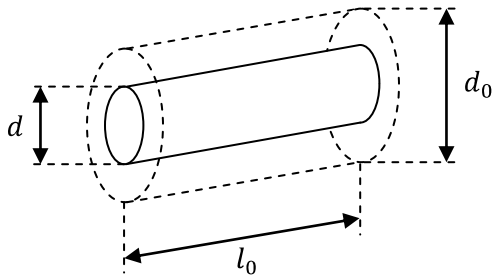
I intend to do all calculations in S.I. units. Before using this expression, it will be necessary to convert the pressures inside the elements from N/m^2 into atmospheres. Then, after computing V_b in mm/s, it will be necessary to convert it into the m/s of our burn rate B. The complete burn rate equation is:

$$\ln(1000B) = 0.046696597\left(\ln\frac{P}{101{,}300}\right)^2 + 0.34808898\left(\ln\frac{P}{101{,}300}\right) - 0.572295873 \quad (27)$$

It will be convenient if we introduce variable f^t as the fraction (by mass or, equivalently, by unburned volume) of the propellant which has been burned by time t. f^t starts at zero when the gun is fired and will rise to one when all of the propellant has been consumed. Since the burn rate depends on pressure, the rate at which the propellant burns will vary from element to element as well as with time. To be precise, we need to define the fraction f_i^t so that every element has its own. Among other uses, it will help us keep track of the masses of burned and unburned propellant in each element at time t, as follows:

$$\left.\begin{aligned} M_i^{Pt} &= \left(1 - f_i^t\right) M_i^{P,t=0} \\ M_i^{Gt} &= f_i^t M_i^{P,t=0} \end{aligned}\right\} \quad (28)$$

We can also use the fractions f_i^t to determine the area of propellant burning at any particular time. Suppose the cylindrical grains of unburned propellant have diameter d_0 and length l_0. The grains will burn from the outside inwards. I will ignore any holes in the web, which extend axially through the grain. (This is a much more important assumption than it seems.) I will also ignore burning of the end-faces. (This is not such an important assumption, since the area of the end-faces is small compared with the area of the curved surface. And, this assumption will be even less significant if the propellant is extruded into long strands or cords, and called cordite.) Some time after ignition, a typical grain will look like this:



The original grain is outlined with a dashed line. The unburned portion at this particular time is outlined with the solid line. The burning surface at this time is the curved surface of the cylinder with diameter d and length l_0. The fraction of propellant burned is equal to the ratio of the burned and original volumes:

$$\begin{aligned} f &= \frac{Burned\ volume}{Original\ volume} \\ &= \frac{Original\ volume - unburned\ volume}{Original\ volume} \\ &= \frac{¼\pi d_0^2 l_0 - ¼\pi d^2 l_0}{¼\pi d_0^2 l_0} = 1 - \frac{d^2}{d_0^2} \\ \rightarrow \quad d &= \sqrt{1-f}\, d_0 \end{aligned} \quad (29)$$

Given the diameter d, the area of the burning surface of this grain A_{grain} at this time is:

$$\begin{aligned} A_{grain} &= Circumference \times length \\ &= \pi d l_0 \\ &= \pi\sqrt{1-f}\, d_0 l_0 \end{aligned} \quad (30)$$

This is a good time to generalize things to include all the propellant in the gun. Here's how we can do that. We know that the unburned propellant originally has a total mass M_{charge}. Suppose that the

crystalline density of the unburned propellant is ρ_c. The original mass of each grain m_{grain} is therefore equal to:

$$m_{grain} = Density \times original\ volume = \tfrac{1}{4}\pi\rho_c d_0^2 l_0 \quad (31)$$

and the total number of grains in the gun n is equal to:

$$n = \frac{M_{charge}}{m_{grain}} = \frac{M_{charge}}{\tfrac{1}{4}\pi\rho_c d_0^2 l_0} \quad (32)$$

Since we have divided the available volume inside the barrel into N elements, and since each element always contains the same propellant solids and gases, the number of grains contained inside each element is n/N. The total burning surface area of all the grains in element #i when the burn fraction is f_i^t can be found by simple multiplication:

$$\begin{aligned} Burning\ area\ in\ \#i\ at\ time\ t &= \frac{n}{N} \times A_{grain} \\ &= \frac{1}{N} \times \frac{M_{charge}}{\tfrac{1}{4}\pi\rho_c d_0^2 l_0} \times \pi\sqrt{1 - f_i^t}\, d_0 l_0 \\ &= \frac{4M_i^{P,t=0}\sqrt{1 - f_i^t}}{\rho_c d_0} \end{aligned} \quad (33)$$

If the burn rate inside element #i during a particular time step is B_i^t, then the volume of solid propellant burned in a short time interval ΔT will be equal to:

$$\begin{aligned} Volume\ burned\ in\ \#i\ during\ \Delta T &= B_i^t \times Burning\ area\ in\ \#i\ at\ time\ t \times \Delta T \\ &= B_i^t \frac{4M_i^{P,t=0}\sqrt{1 - f_i^t}}{\rho_c d_0}\Delta T \end{aligned} \quad (34)$$

Since the density of the unburned propellant is ρ_c, then the mass of solid propellant burned inside element #i during time interval ΔT is equal to:

$$\begin{aligned} Mass\ burned\ in\ \#i\ during\ \Delta T &= \rho_c \times Volume\ burned\ in\ \#i\ during\ \Delta T \\ &= B_i^t \frac{4M_i^{P,t=0}\sqrt{1 - f_i^t}}{d_0}\Delta T \end{aligned} \quad (35)$$

Since the heat given off during the conversion is Q_0 Joules per unit mass, then the heat released inside element #i during time interval ΔT is:

$$\begin{aligned} Heat\ released\ in\ \#i\ during\ \Delta T &= Q_0 \times Mass\ burned\ in\ \#i\ during\ \Delta T \\ &= Q_0 B_i^t \frac{4M_i^{P,t=0}\sqrt{1 - f_i^t}}{d_0}\Delta T \end{aligned} \quad (36)$$

Notice that the "heat released in element #i during time interval ΔT" is exactly the same as the quantity $\Delta Q_i^{t,t+\Delta T}$ which we talked about in the previous section, and which is one of the variables required to evaluate Equation (23).

$$\Delta Q_i^{t,t+\Delta T} = Q_0 B_i^t \frac{4M_i^{P,t=0}\sqrt{1-f_i^t}}{d_0}\Delta T \quad (37)$$

This is the principal equation we will use to describe the burning process. In the computer code, we will need to advance the burn fraction from one time step f_i^t to the next $f_i^{t+\Delta T}$. To do that, we will have to use a couple of the preceding equations in their discretized form.

Part 5 – Thermal equation of state of the gas

The "state" of a gas is the collection of values which describe its physical characteristics. In the simplest models, the variables which are of interest for some fixed quantity of gas at some time t are its pressure P^t, volume V^t, temperature T^t and quantity. The quantity of gas can be measured in different ways. The one I will use is the cardinal number of molecules. However, I will not report the number of molecules simply as 8×10^{24} molecules, say, but will first divide the number of molecules by the constant 6.022141×10^{23}. The constant is called Avogadro's number and the quotient after the division is said to be the number of moles of gas. It is customary to use the symbol N, or N^t at time t, for the number of moles. I will do so even at the risk of possible confusion with the number of elements into which I discretized the gas.

One of the most valuable and widely-used mathematical models for the state of a gas is:

$$P^t V^t = N^t R_{IGC} T^t$$

where $R_{IGC} = 8.314462\ J/°K-mole$ is the Ideal Gas Constant. A gas which behaves like this is an Ideal Gas. Conceptually, and briefly, an ideal gas is one in which the individual molecules are small compared with the distance they travel between collisions. Since this expression relates the pressure-times-volume product to the temperature, it is called the thermal equation of state.

At and around standard ambient conditions, virtually all gases can be treated as ideal. At extreme temperatures and pressures, virtually all gases depart from the ideal. Messr. van der Waals proposed a form of correction to the ideal which greatly expanded the envelope of applicability. His correction (Nobel in 1910 for it) is often written as:

$$\left[P^t + \left(\frac{N^t}{V^t}\right)^2 a\right](V^t - N^t b) = N^t R_{IGC} T^t$$

The first correction term, the one with the coefficient a, arises from the existence of a pairwise attraction force between molecules. It is this pairwise attraction which leads to the condensation of the gas into a liquid at a sufficiently low temperature. Basically, the correction term arises from the asymmetry experienced by molecules as they get nearer to the boundary walls of the container.

The second correction term, the one with the coefficient b, is needed when the assumption that the molecules are mathematical points becomes unrealistic. When subjected to high pressure, for example, the size of the molecules becomes a more significant fraction of the total available volume, and cannot be

neglected. Van der Waals handled this by assuming that individual molecules were hard spheres with radius r and volume $\frac{4}{3}\pi r^3$. The absolute volume of space occupied by substance would therefore be this molecular volume multiplied by the number of molecules. Note, though, that the center-to-center distance between two colliding molecules at their point of closest approach is not r, but twice r. This means that the "zone of exclusion" between hard molecular centers has radius of $2r$ and a volume of $8 \times \frac{4}{3}\pi r^3$. Since molecules collide in pairs, the $2r$ center-to-center distance can be shared by two molecules, with the result that the volume occupied by two colliding molecules is $\frac{1}{2} \times 8 \times \frac{4}{3}\pi r^3$. If there are N^t moles of molecules in the container, then the volume of space which is "occupied" (possible positions which are denied to molecular centers) is reduced by $N^t \times 4 \times \frac{4}{3}\pi r^3$. This denied volume is called the "co-volume". Van der Walls recognized that molecules are not hard spheres. He treated the factor 4 in his correction $b = 4 \times \frac{4}{3}\pi r^3$ as an upper bound.

The gas in our gun will be under very high pressure and temperature. The inter-molecular attraction force correction will be insignificant compared with the co-volume correction. I will write the equation of state for the gas in the i^{th} element in the following way. Since this is a "state" equation, which should apply at every instant in time.

$$P_i^t\left(\tilde{V}_i^t - b_i^t\right) = N_i^t R_{IGC} T_i^t \quad (38)$$

The volume $\tilde{V}_i^t$ is the volume inside element #i which is free space available for gas. This is the total volume of element #i less he volume of unburned propellant.

b_i^t is the correction for co-volume. In Equation (38), the volume $\tilde{V}_i^t$ is an absolute volume, which could be measured in cubic meters, for example. However we go about quantifying it, the co-volume b_i^t will also be an absolute volume.

It is worth noting that almost all of the byproducts from the combustion of nitrocellulose are diatomic gases. Although the molecules of nitrocellulose are complex, what remains after buring is not. A commonly used co-volume correction factor for simple gases is $b = 0.00095\ m^3/kg$. If this is multiplied by the mass of gas inside a container, it yields an absolute volume. Note that b has units that are the reciprocal of density. The effective denisty which corresponds to this value of b is 1053 kg/m^3.

Often, the value of b is assumed to be constant. That is not a good assumption in our case. The gases in the barrel are going to be under tremendous pressure. Under high pressure, molecules are able to penerate more deeply into one another when they interact. As the ambient pressure increases, the volume of space which is denied to molecules of gas decreases. Usually, this effect is taken into account by treating b as a value which depends on the density of the gas. I am going to use the following dependence:

$$b\left(\rho_{gas}\right) = \frac{b}{1 + 2\left(\frac{\rho_{gas}}{500}\right)} \quad (39)$$

At small densities, this function is close to the usual value $b = 0.00095\ m^3/kg$. The function decreases inversely with the density. When the density reaches $500\ kg/m^3$, the evaluated value is one-third of the low-density value. Note that the density of air under standard conditions is $1.225\ kg/m^3$, so this adjustment to b is definitely a high-density phenomenon.

Since the mass of gas inside element #i is the sum $M_i^{Gt} + M_i^{Air}$, and the free space volume available to the gas is $\tilde{V}_i^t$, we can carry out the follwing algebra:

$$Density\ \ of\ the\ gas: \qquad \rho_{gas} = \frac{M_i^{Gt} + M_i^{Air}}{\tilde{V}_i^t} \qquad (40A)$$

$$Absolute\ covolume: \qquad b_i^t = \frac{b\left(M_i^{Gt} + M_i^{Air}\right)}{1 + \left(\frac{2}{500}\frac{M_i^{Gt} + M_i^{Air}}{\tilde{V}_i^t}\right)} \qquad (40B)$$

$$Thermal\ equation\ of\ state: \quad P_i^t\left(\tilde{V}_i^t - b_i^t\right) = \left(N_i^{Gt} + N_i^{Air}\right)R_{IGC}T_i^t \qquad (40C)$$

Part 6 – Calorific equation of state of the gas

In the previous section, we related the pressure-times-volume product to the temperature of a fixed quantity of gas. In this section, we will do something similar, but relate the pressure-times-volume product to the internal energy of the gas. Since the related state variable is energy, this new relationship is called the calorific equation of state. Let me state the result first:

$$U_i^t = \frac{P_i^t\left(\tilde{V}_i^t - b_i^t\right)}{\gamma - 1} \qquad (41)$$

The numerator on the right-hand side is the pressure inside the i^{th} at time t multiplied by the volume of gas (with the co-volume corresction) inside the element at that time. The denominator is the Adiabatic Index γ of the gas, less one. A gas whose energy can be expressed in this way is called a gamma-law gas.

I will give a rough justification for this relationship to shed some light on the assumptions which are involved. Start by considering element #i at a temperature of absolute zero. The pressure will be zero and the internal energy will be zero. It does not matter what volume the gas occupies; the pressure and internal energy will still be zero. It is convenient to start things off with the element's volume being $\tilde{V}_i^t$ and the volume occupied by the gas being $\tilde{V}_i^t - b_i^t$.

Now, let's add some quantity of heat ΔQ, which raises the temperature from zero to T_i^t and the pressure of from zero to P_i^t. The first Law of Thermodynamics tells us that the amount of heat added must be equal to the sum of: (i) the work done by the gas during the process and (ii) the internal energy added to it. In this particular thought experiment, the gas does not do any work. Its volume stays constant. Work would have been done by the pressure if the volume had expanded, but that did not happen. It follows that all of the heat added is absorbed by the gas as internal energy. Since the internal energy was initially zero, we can say that the gas's internal energy after this process is:

$$U_i^t = \Delta Q$$

The specific heat of a substance is defined as the rate at which the addition of heat ΔQ raises the temperature by ΔT.

$$specific\ heat = \frac{\Delta Q}{\Delta T}$$

The specific heat depends on the details of the process. For example, if the pressure of a gas is allowed to change as the heat is being added, the specific heat will be different from the measurement made if the pressure is held constant. In our thought experiment, we held the volume of the gas constant while the heat was added. The specific heat which is measured in this constant-volume process is called the "specific heat at constant volume" and is usually represented by the symbol C_V. In our case, then:

$$C_V = \frac{\Delta Q}{\Delta T} = \frac{\Delta Q}{T_i^t - 0} = \frac{\Delta Q}{T_i^t}$$

$$\rightarrow \qquad U_i^t = \Delta Q = C_V T_i^t \qquad (42)$$

The specific heat $\Delta Q/\Delta T$ is not necessarily the same at all temperatures. In fact, usually it is not. It varies as the temperature changes. An important assumption we are going to make is that the specific heat C_V is constant and does not vary with temperature.

Next, we are going to invoke Meyer's Law. For an Ideal Gas, and for some other models of gases as well:

$$N_i^t R_{IGC} = C_P - C_V \quad (43)$$

where $N_i^t = N_i^{Gt} + N_i^{Air}$ is the number of moles of gas, R_{IGC} is the Ideal Gas Constant (both as used in the previous section) and C_P is the specific heat at constant pressure. Substituting this relationship into the thermal equation of state from Part 5 above, we get:

$$P_i^t\left(\tilde{V}_i^t - b_i^t\right) = (C_P - C_V)T_i^t \quad (44)$$

We can combine Equations (42) and (44) to write:

$$\begin{aligned} U_i^t &= C_V \frac{P_i^t\left(\tilde{V}_i^t - b_i^t\right)}{(C_P - C_V)} \\ &= \frac{P_i^t\left(\tilde{V}_i^t - b_i^t\right)}{\left(\frac{C_P}{C_V} - 1\right)} \\ &= \frac{P_i^t\left(\tilde{V}_i^t - b_i^t\right)}{\gamma - 1} \end{aligned} \qquad (45)$$

where the ratio of the specific heats C_P/C_V is called the Adiabatic Index of the gas, and is usually represented by the symbol γ. Although we have assumed that C_V is constant with temperature, it does not follow that the Adiabatic Index γ is also constant. It depends on C_P. It is possible to investigate how γ changes separately with temperature and with pressure. Often, the effects are combined and γ is said to be a function of the density.

For the purposes of this paper, I am going to that γ varies linearly with density. As a starting point, let's consider the gas as if it was an ideal diatomic gas. The gas products resulting from the burning of nitrocellulose are mainly CO_2, CO, H_2O, H_2 and N_2. Recall that CO_2 is a colinear molecule (the three atoms lie on a straight line), so carbon dioxide has the same dynamic characteristics as a diatomic molecule. The ambient air trapped inside the barrel before firing is mostly N_2 and O_2 so it, too, is virtually all diatomic. The Adiabatic Index of an ideal diatomic gas at room temperature and pressure has a theoretical value of $(5 + 2)/5 = 1.4$, arising from three translational and two rotational degrees of freedom. At high temperatures, a vibrational mode of motion begins to absorb energy, so the number of degrees of freedom increases to six, and the Adiabatic Index of an ideal diatomic gas falls to $(6 + 2)/6 = 1.333$ at high temperatures. I have seen a report that suggests the Adiabatic Index rises to 1.9 when the gas density is $1200\ kg/m^3$. The straight line which connects these two data points is:

$$\gamma = 1.333 + 0.567\left(\frac{\rho_{Gas}}{1200}\right) \quad (46)$$

This is the expression I have used in the numerical simulation for the Adiabatic Index.

Part 7 – Equation of motion for the shell

I am going to treat the shell as a rigid body being accelerated in one direction by a uniform pressure exerted on its rear face. We have already defined the mass and speed of the shell as M_{shell} and v_{shell}, respectively. Newton's Law for the acceleration of the shell down the axis of the barrel is:

$$M_{shell}\frac{dv_{shell}}{dt} = Net\ accelerating\ force \quad (47)$$

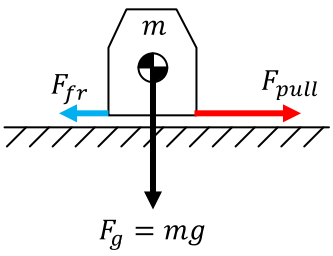
Writing down Newton's Law is not the hard part. The hard part is trying to model the various effects which contribute to the net accelerating force. I have included some of the usual suspects in the following expression.

$$Net\ accelerating\ force = (P_{shell} - P_{atm})A - F_{friction} - F_{rifling} - F_{melting} \quad (48)$$

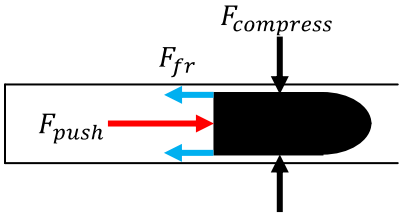
Here, P_{shell} is the pressure force acting on the rear face of the shell. P_{atm} is the ambient air pressure of the day, which is a reasonable first approximation for the pressure which acts on the projected frontal area of the shell. P_{atm} is only a first approximation because the air initially present in the barrel in front of the shell will be compressed and heated as the progress of the shell drives it out of the barrel.

There will be friction $(F_{friction})$ as the metal projectile slides through the metal barrel. The rifling $(F_{rifling})$ will increase the metal-to-metal friction, but will have a second retarding effect as well. Even if there was no friction at all between the rifling and the projectile, the rifling will give rise to a retarding force. The rifling extracts mechanical power from the forward motion of the shell and diverts it into rotational acceleration around the shell's spin axis. This diversion of energy from the forward motion is a consequence of simple dynamics of motion, and is not related to friction. The metals in contact will also be heated by the friction, and some deformation $(F_{melting})$ is to be expected.

For the purposes of this paper, I am going to collect together all of the retarding forces under the heading "friction". I am going to use a simple model for the friction not unlike the one often used to describe frictional forces due to gravitational weight. Let me review the traditional model, which is illustrated here.



Gravity pulls mass m down against a flat surface with weight F_g. The mass is pulled towards the right by some other force F_{pull}. The interface between the mass and the flat surface resists the pulling force with a frictional force F_{fr}. Surprisingly, the magnitude of the frictional force is not related to the magnitude of the pulling force. Instead, it is proportional to the transverse force – the weight. A coefficient of friction μ is often used as the constant of proportionality, so that $F_{fr} = \mu F_g$.



I have shown to the left how one can apply the traditional model to the shell inside the barrel. The frictional force F_{fr} is, once again, opposite in direction to the velocity of the shell. One would expect the magnitude of the frictional force to be proportional, not to the pushing force F_{push}, but to the transverse force $F_{compress}$ with which

the barrel seizes the shell. If so, then the expression would be $F_{fr} = \mu F_{compress}$, where the constant of proportionality might or might not be the same coefficient of friction as above.

This interpretation makes sense, but what to do about $F_{compress}$ is far from clear. Let me describe three alternatives, each of which makes sense in its own right, but all of which have quite different consequences.

1. In the traditional gravitational case, the mass of the object is very often a constant. It follows that the frictional force is also constant. Often, one uses two versions of the coefficient of friction, a "static" coefficient of friction which applies while the weight is at rest, and another, but also constant, "dynamic" coefficient which applies once the weight begins moving. If this type of behaviour applies to the shell in the barrel, and if the compression force is constant, then the result would be a frictional force which remains constant while the shell accelerates down the barrel.

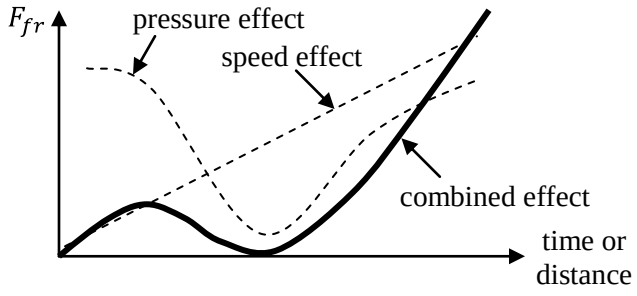
2. An alternative would be to focus on the very thin gap which may exist between the circumference of the shell and the barrel. Perhaps there was no gap at all when the shell was rammed into the breech. But, under the enormous pressure of the expanding gas, the barrel will be forcibly enlarged. That will create a gap which will quickly be filled by the high pressure propellant gas. If this is what happens, then an appropriate model would be to treat the thin gap as a thin layer of viscous fluid between two moving surfaces. The frictional force would then have the characteristics of a viscous force. It would be proportional to three variables and inversely proportional to a fourth. The frictional force would be proportional to: (i) the surface area of contact, which we would estimate as the circumference of the inside of the barrel multiplied by the length of the shell, (ii) the relative speed of the two surfaces which, in our case, would be the speed of the shell, and (iii) the dynamic viscosity of the high temperature high pressure gases from combustion. As for the fourth variable, the frictional force would be inversely proportional to the distance between the surfaces. The formula we would use for the frictional force would be something like this:

$$F_{fr} = \mu \frac{2\pi R_{shell} L_{shell} \times v_{shell}}{d_{gap}}$$

The pressure driving the shell P_{shell} enters into this relationship in two ways. Firstly, it affects the viscosity. The viscosity will increase with pressure, but only gradually. Secondly, the distance d_{gap} between the surfaces will increase in step with the increase in pressure. Basically, the frictional force will be driven in the following way:

$$F_{fr} \propto \frac{v_{shell}}{P_{shell}}$$

The frictional force will vary as the shell travels down the barrel as shown in the following figure. (I have run a few runs already, and can say that the pressure peaks when the shell is about half-way down the barrel. The reciprocal of the pressure has a dip.) The frictional force in this alternative is not constant at all.



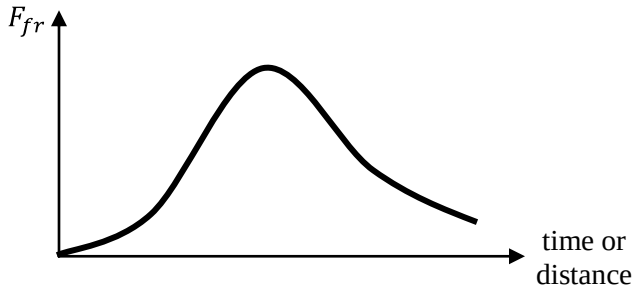
3. A third alternative is to imagine that the thin gap between the shell and the barrel behaves like a rubber or plastic seal between two moving parts. If that is the case, then the pressure P_{shell} acting on the rear face of the shell also acts on the end-face of this "seal" where it peeks out through the gap around the circumference of the rear end of the shell. As hydraulic pressure does, the pressure on the end-face would be transmitted throughout the volume of the "seal". In particular, it would act radially outwards, and would be the main cause of the compression force $F_{compress}$. Taking into account the contact area $2\pi R_{shell}L_{shell}$ between the surfaces, we could write:

$$F_{compress} = 2\pi R_{shell}L_{shell} \times P_{shell}$$
$$\rightarrow \quad F_{fr} = \mu \times 2\pi R_{shell}L_{shell} \times P_{shell}$$

Noting that the area of the barrel is A_B, and that it is equal to πR_{shell}^2, we can express the main dependence as follows:

$$F_{fr} \propto P_{shell} \times A_B \quad (49)$$

Then, the frictional force as the shell travels down the barrel looks like the following. In this alternative, the frictional force is greatest when the pressure peaks.



In my opinion, none of these three alternatives is satisfactory. I will explore in a subsequent paper an approach I believe is more credible. It leads to a frictional force which is more like the third alternative above than the first two. Because of that, I will use a frictional force having the characteristics of Equation (49). I will make just two changes. I will use C_{fr} as the constant of proportionality. I will also use the pressure difference $P_{shell} - P_{atm}$, rather than just P_{shell}, as the dependent variable. That is a very minor change, but greatly simplifies the equation for the shell's dynamics, as I will now demonstrate.

Let me substitute this form of frictional force into Equation (48) for Newton's Law. We get:

$$M_{shell}\frac{dv_{shell}}{dt} = (P_{shell} - P_{atm})A_B - C_{fr}(P_{shell} - P_{atm})A_B$$
$$\rightarrow \quad M_{shell}\frac{dv_{shell}}{dt} = (1 - C_{fr})(P_{shell} - P_{atm})A_B$$
$$\rightarrow \quad \left(\frac{M_{shell}}{1 - C_{fr}}\right)\frac{dv_{shell}}{dt} = (P_{shell} - P_{atm})A_B \quad (50)$$

This has exactly the same form as Newton's Law without any frictional force at all. Friction has been taken into account by increasing the mass of the shell from M_{shell} to a larger mass $M_{shell}/(1 - C_{fr})$. This "effective" mass, being larger, causes the "effective" mass to accelerate more slowly in response to the driving pressure than it would in the absence of any friction. We can express the effective mass as follows:

$$M_{shell}|_{effective} = \frac{M_{shell}}{1 - C_{fr}}$$
$$= M_{shell} + \frac{C_{fr}}{1 - C_{fr}} M_{shell} \quad (51)$$

Calculations done by some investigators show that the effective mass can be as much as 20% greater than the rest mass. In the computer code for this paper, I have set $C_{fr} = 0.167$, which gives a 20% increase in effective mass.

Implementation of the numerical procedure

Suppose we have just completed calculating the speeds of all the boundary faces at some half-time step, say time $t + \Delta T/2$. We therefore know $v_i^{t+\Delta T/2}$ for all boundary faces $i = 0,1,2 \cdots N$. Speed $v_0^{t+\Delta T/2}$ will be equal to zero, since it is the breech plate. Speed $v_N^{t+\Delta T/2}$ will be the speed of the shell. The other speeds will have been calculated using the equations for gas dynamics.

Step #1 – Advance the locations of the boundary faces
Advance the locations of all boundary faces from time t through one whole time step to time $t + \Delta T$.

$$X_i^{t+\Delta T} = X_i^t + \left(v_i^{t+\Delta T/2} \Delta T\right) \text{ for all } i = 0,1,2 \cdots N \quad (A)$$

Step #2 – Calculate the absolute volumes of the elements
Calculate the absolute volumes of all the elements at time $t + \Delta T$.

$$V_i^{t+\Delta T} = A_B\left(X_i^{t+\Delta T} - X_{i-1}^{t+\Delta T}\right) \text{ for all } i = 1,2 \cdots N \quad (B)$$

Step #3 – Analyze the rate at which burning occurs
Calculate the burn rate $B_i^{\Delta T}$ inside each element during this time step.

$$\ln\left(1000 B_i^{\Delta T}\right) = \begin{bmatrix} 0.046696597 \left(\ln \dfrac{P_i^t}{101{,}300}\right)^2 + \cdots \\ \cdots + 0.34808898 \left(\ln \dfrac{P_i^t}{101{,}300}\right) - 0.572295873 \end{bmatrix} \text{ for all } i = 1,2 \cdots N \quad (C)$$

Calculate the mass of propellant burned inside each element during this time step.

$$\Delta M_i^{\Delta T} = B_i^{\Delta T} \frac{4 M_i^{P,t=0}}{d_0} \sqrt{1 - f_i^t} \Delta T \text{ for all } i = 1,2 \cdots N \quad (D)$$

Advance the burned and unburned masses in each element, and the burned fraction.

$$\left.\begin{aligned} M_i^{P,t+\Delta T} &= M_i^{Pt} - \Delta M_i^{\Delta T} \\ M_i^{G,t+\Delta T} &= M_i^{Gt} + \Delta M_i^{\Delta T} \\ f_i^{t+\Delta T} &= \frac{M_i^{Gt+\Delta T}}{M_i^{P,t=0}} \end{aligned}\right\} \text{ for all } i = 1,2 \cdots N \quad (E)$$

Step #4 – Calculate the heat generated

Calculate the heat released inside each element by the burning which takes place during this time step. This heat by will absorbed by the gas.

$$\Delta Q_i^{t,t+\Delta T} = Q_0 \Delta M_i^{\Delta T} \text{ for all } i = 1,2 \cdots N \quad (F)$$

Step #5 – Calculate the density of the gas

It is necessary first to calculate the volume which is available as free space for the gas. This is Equation (24), which adjusts the absolute volumes of the elements for the volume occupied by the unburned solid propellant.

$$\tilde{V}_i^{t+\Delta T} = V_i^{t+\Delta T} - \frac{M_i^{P,t+\Delta T}}{\rho_c} \text{ for all } i = 1,2 \cdots N \quad (G)$$

Calculation of the density does not require any co-volume correction. That correction is needed for the equations of state, but the definition of density is straight-forward.

$$\rho_i^{t+\Delta T} = \frac{mass\ of\ gas}{volume\ of\ gas} = \frac{M_i^{G,t+\Delta T} + M_i^{Air}}{\tilde{V}_i^{t+\Delta T}} \text{ for all } i = 1,2 \cdots N \quad (H)$$

Step #6 – Calculate the co-volume correction

Calculate the co-volume correction.

$$b_i^{t+\Delta T} = \frac{0.00095\left(M_i^{G,t+\Delta T} + M_i^{Air}\right)}{1 + \left(\frac{2}{500}\rho_i^{t+\Delta T}\right)} \text{ for all } i = 1,2 \cdots N \quad (I)$$

This is probably a good place to update the number of moles of gas inside each element. Since gas is added at the rate of 40 moles per kilogram of solid propellant burned, the number of moles added by combustion during the ime step is:

$$Moles\ added = 40 \Delta M_i^{\Delta T} \text{ for all } i = 1,2 \cdots N \quad (J)$$

Advance the total number of moles of gas inside each element. Do do forget that the total number of moles includes the original air.

$$\left.\begin{aligned} N_i^{G,t+\Delta T} &= N_i^{Gt} + Moles\ added \\ N_i^{t+\Delta T} &= N_i^{G,t+\Delta T} + N_i^{Air} \end{aligned}\right\} \text{ for all } i = 1,2 \cdots N \quad (K)$$

Step #7 – Calculate the Adiabatic Index inside each element

We can calculate the Adiabatic Index using Equation (46). I did not say so above, but a separate index is needed for each element.

$$\gamma_i^{t+\Delta T} = 1.333 + 0.567\left(\frac{\rho_i^{t+\Delta T}}{1200}\right) \text{ for all } i = 1,2 \cdots N \quad (L)$$

Step #8 – Calculate the pressure and internal energy

I am going to use an iterative procedure to calculate the pressure $P_i^{t+\Delta T}$ and internal energy $U_i^{t+\Delta T}$ inside each element. In effect, this is a simultaneous solution of the equation ensuring conservation of internal energy and the calorific equation of state. The following procedure, for element $\#i$, can only be carried out after the procedure has been completed for element $\#i-1$, to the left of this element. To begin the process, assume that the pressure is the same as it was in this element during the last time step. That is, $P_i^{t+\Delta T} = P_i^t$. Then, the sub-steps are:

Step #8A

Calculate the speed of sound, which is one of the ingredients of the artificial viscosity.

$$S_i^{t+\Delta T} = \sqrt{\gamma_i^{t+\Delta T} \frac{P_i^{t+\Delta T}}{\rho_i^{t+\Delta T}}} \quad (M)$$

Note that this formulation for the speed of sound is accurate only for ideal gases at modest temperatures and pressures. At high temperatures and pressures, something more complex should be used. The only reason this approximation is adequate for our purpose is that the speed of sound is not a required result in its own right, but is only used as an estimating parameter in the calculation of the artificial viscosity.

Step #8B

Calculate the artificial viscosity.

$$W_i^{t+\Delta T} = +C_1 \rho_i^{t+\Delta T} \left(v_i^{t+\Delta T/2} - v_{i-1}^{t+\Delta T/2} \right)^2 + C_2 \rho_i^{t+\Delta T} S_i^{t+\Delta T} \left| v_i^{t+\Delta T/2} - v_{i-1}^{t+\Delta T/2} \right| \quad (N)$$

This calculation only matters if the relative velocity condition is met. If the velocity condition is not met, and the element is expanding, then the artificial viscosity will be zero.

Step #8C

Calculate the internal energy using the equation ensuring conservation of internal energy.

$$U_i^{t+\Delta T} = U_i^t + \Delta Q_i^{t,t+\Delta T} - \tfrac{1}{2}\left[\left(P_i^{t+\Delta T} + W_i^{t+\Delta T} \right) + \left(P_i^t + W_i^t \right) \right] \left(V_i^{t+\Delta T} - V_i^t \right) \quad (O)$$

Step #8D

Use the calorific equation of state to calculate a new, revised, pressure.

$$P_i^{t+\Delta T} = \left(\gamma_i^{t+\Delta T} - 1 \right) \frac{U_i^{t+\Delta T}}{\tilde{V}_i^{t+\Delta T} - b_i^{t+\Delta T}} \quad (P)$$

Step #8E

Using this new value for the pressure, go back and start over at Step #8A. Terminate the iterations when the changes in $P_i^{t+\Delta T}$ and $U_i^{t+\Delta T}$ during one iteration become acceptably small, either in absolute terms or in relative terms. Note that, once element $\#i$ has been sorted out, it will be necessary to move on to element $\#i+1$.

Step #9

As a loose end for the gas dynamics, the temperature inside element $\#i$ can be computed using the thermal equation of state.

$$T_i^{t+\Delta T} = \frac{P_i^t\left(\tilde{V}_i^{t+\Delta T} - b_i^{t+\Delta T}\right)}{N_i^t R_{IGC}} \text{ for all } i = 1,2 \cdots N \quad (Q)$$

Step #10
Calculate the acceleration of the shell.

$$\left.\frac{dv_{shell}}{dt}\right|^t = \frac{(1 - C_{fr})}{M_{shell}}(P_{shell}^t - P_{atm})A_B \; where \; P_{shell}^t = P_N^{t+\Delta T} \quad (R)$$

Step #11
We are now ready to advance the speeds of the boundary faces through the next time step. For the boundary faces which are enclosed within gas, the relevant equation is Equation (11).

$$v_i^{t+3\Delta T/2} = v_i^{t+\Delta T/2} - \left(\frac{2\Delta T}{M_i + M_{i+1}}\right)\left[\left(P_{i+1}^{t+\Delta T} - P_i^t\right) + \left(W_{i+1}^{t+\Delta T} - W_i^t\right)\right]A_B \text{ for all } i = 1,2 \cdots N-1 \quad (S)$$

The quantities $P_{i+1}^{t+\Delta T}$ and $W_{i+1}^{t+\Delta T}$ will have been finalized during Step #6 and can be used here without further ado. The speed of the last boundary face $v_N^{t+3\Delta T/2}$ is the speed of the shell, advanced using:

$$v_{shell}^{t+3\Delta T/2} = v_{shell}^{t+\Delta T/2} + \left(\left.\frac{dv_{shell}}{dt}\right|^t \Delta T\right) \quad (T)$$

Equation (T) completes the calculations done during one time step. The new speeds, at time $t + 3\Delta T/2$, are the starting point for the next time step.

The use of adaptive time steps

I quickly discovered that it is not practical to use a constant time step. There are times during the process when the time step needs to be very small, on the order of tens of nanoseconds. Running the entire simulation at this time scale is not practical. I handled the problem in the following way. At the end of every time step, I searched through the pressures in all the elements to identify which element experienced the greatest relative change in pressure during the time step just finished. If this percentage change exceeded a certain preset threshold, I then reduced the length of the time step to be used during the next iteration. On the other hand, if this percentage change in pressure was less than another certain preset threshold, then I increased the length of the time step for the next iteration. The parameters I used, which seemed to generate consistent and reasonably quick simulations, were the following:

$$\left.\begin{array}{ll} \text{If } \left|\max_{all\ i}\left(\frac{P_i^{end} - P_i^{start}}{P_i^{start}}\right)\right| > 0.001, & \text{then } \Delta T_{next} = 0.9\Delta T_{last} \\ \text{If } \left|\max_{all\ i}\left(\frac{P_i^{end} - P_i^{start}}{P_i^{start}}\right)\right| < 0.00025, & \text{then } \Delta T_{next} = 1.01\Delta T_{last} \end{array}\right\} \quad (U)$$

In the final version of the code, I applied this same test, with these same preset thresholds, to the density of the gas inside each element as well as to the static pressure. I insisted on the reduction in the duration of the next time step if the maximum relative change in either pressure or density required it. On the other hand, I permitted the duration of the time step to be increased only if both the pressure test and the density test were satisfactory.

Other features of the code

I have listed in Appendix "A" the source code for the numerical simulation of the base case, whose parameters have been discussed above. The code was developed in Visual Basic 2010 Express. The number of elements is set to 2000. A limited selection of key values is displayed on the screen every 250 time steps.

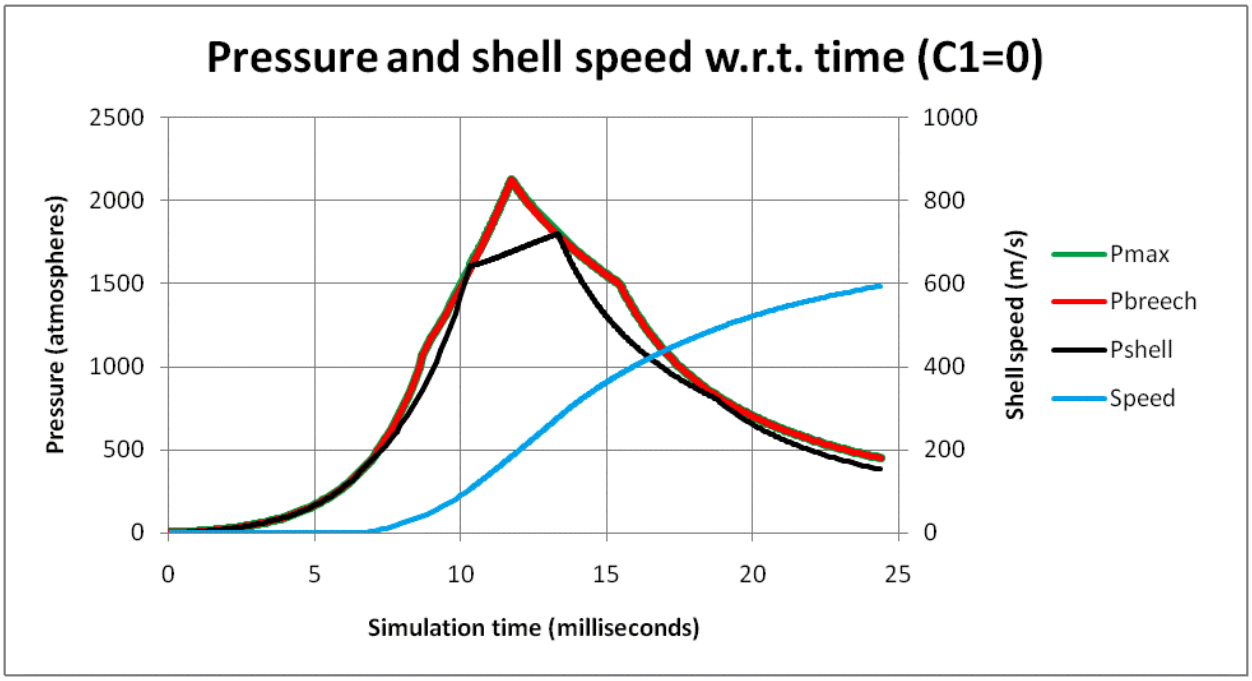
The principal variables are written into text files for later analysis. As an example, the variable B_i^t is written into a text file with the name "Naval_gun_simulation_Bi.txt". The user can set the time interval between successive writes to the output text files. Similarly, the user can set the interval between element indices to be saved, so that all 2000 values are not written. The numbers are written as comma-separated values ("csv") to make it easy to import into Excel or other analysis programs.

Note this. In the base case, I set the artificial viscosity constant to zero, $C_1 = C_2 = 0$. Despite my expectation, the run concluded with no mathematical errors.

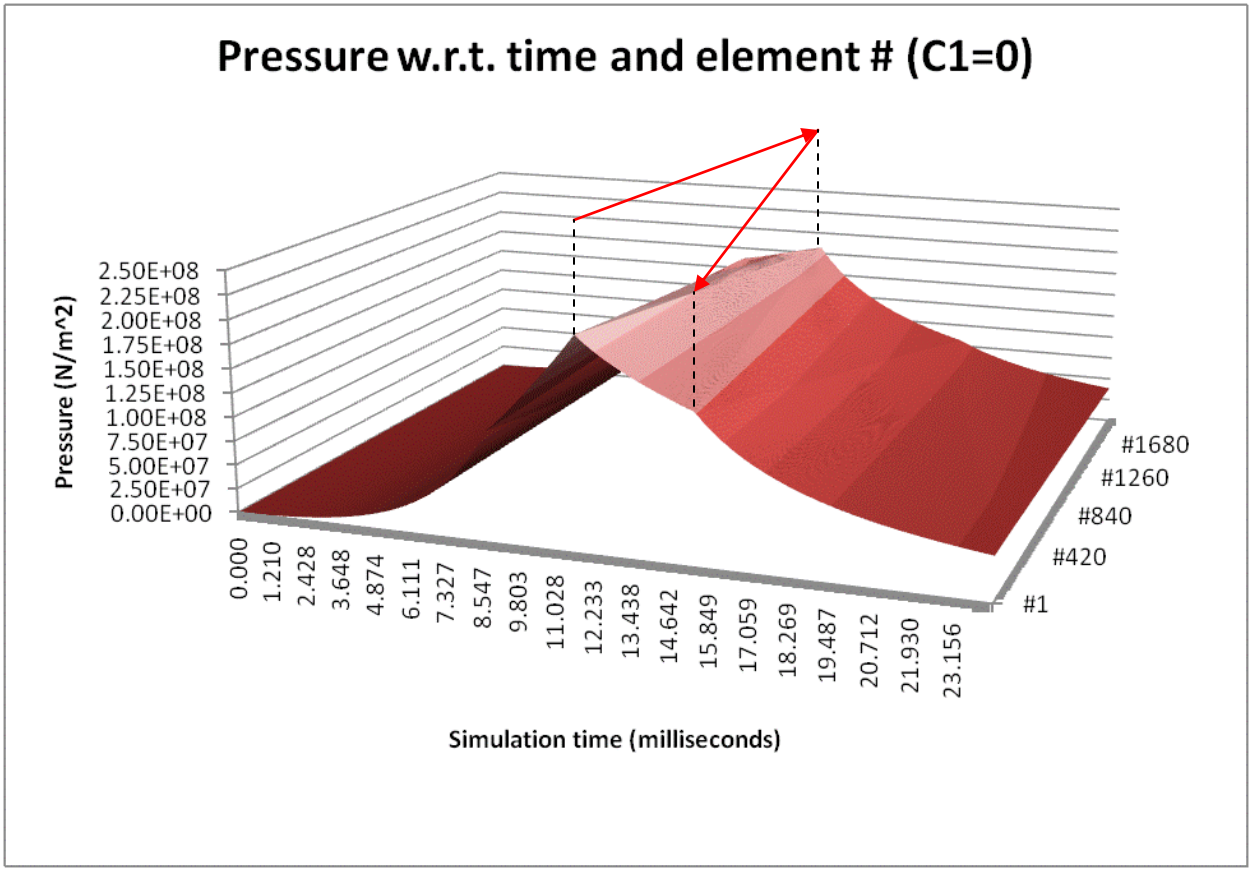
Results from the simulation of the base case

The following graph shows three pressures and the shell's speed with respect to time. The red curve is the pressure at the breech, inside element #1. The black curve is the pressure inside the last element, element #2000, which is the one which actually presses on the rear face of the shell. The green curve, which is almost completely hidden by the breech pressure, is the maximum of the pressures inside all the elements. The maximum pressure experienced inside the barrel during firing is just under 2100 atmospheres.

Since the shell is held in place by its engraving band until the pressure inside the chamber reaches 400 atmospheres, the shell's speed (blue curve) is zero until that time. This occurs 6.70 milliseconds into the simulation. The shell accelerates in proportion to the pressure, and reaches a speed of about 595 meters per second when it exits the barrel, 24.36 milliseconds after ignition.

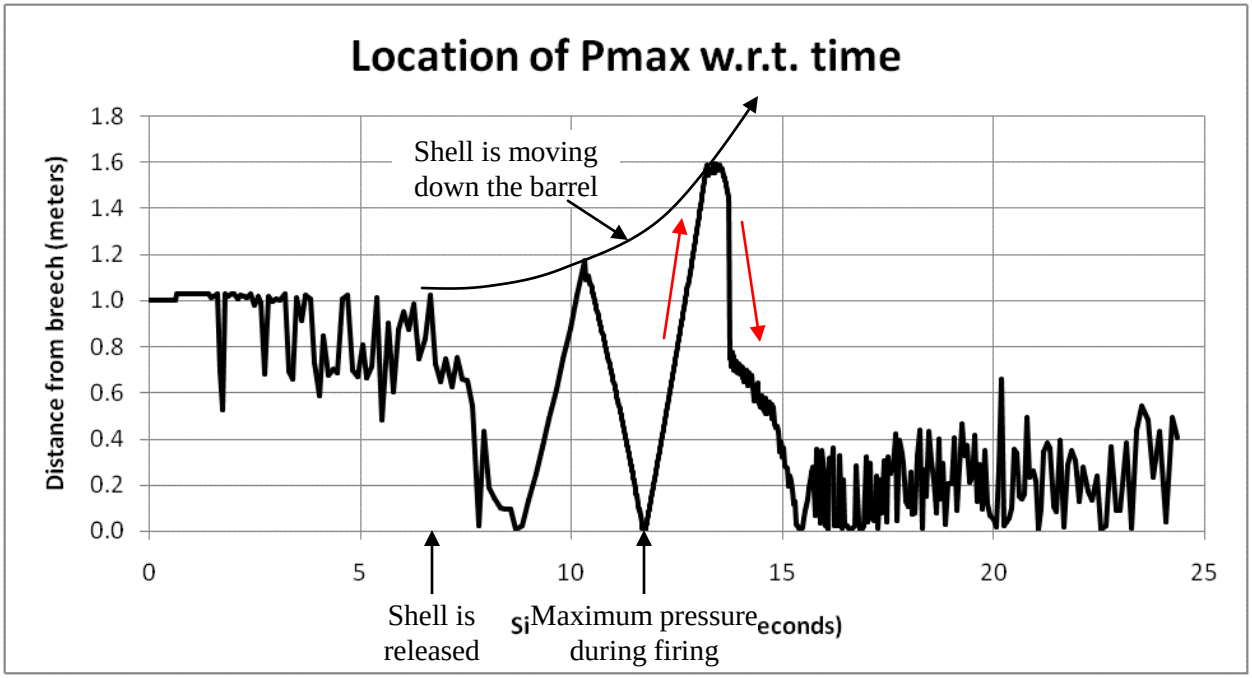


The cusps in the pressure curves represent mild shock waves. As I stated above, the numerical integration ran without a hitch even though I set the artificial viscosity to zero. The shock waves are a little clearer in the following surface plot. It shows the pressure (vertical axis) as a function of time (horizontal axis) and element number (depth axis, projected into the page). The colour bands are 100 bar, or approximately 100 atmospheres, high. The numbers along the depth axis are element numbers, with the breech element #1 being at the front side of the graph. The depth axis corresponds to the distance between the breech and the shell, but it is not the actual distance in meters. As time passes and the shell moves further down the barrel, the elements get longer in the x-direction. This increase in their length is not shown in the graph.



The pressure is always a little higher at the breech-end than at the shell-end. I have indicated with arrows raised above the surface, the progress of a shock wave from the breech to the shell, where it is reflected back to the breech. That particular shock wave starts at the moment when the pressure reaches its peak value at the breech. There is a similar shock wave on the "front" side of this hill, while the pressure is still rising, but that wave is not so clear with the graph oriented as it is.

I have tried in the following graph to track the progress of the shock wave as it travels up and down the barrel. To prepare this graph, I first looked through the elements along the barrel to find out which one had the highest pressure. I repeated this at every time step. If a shock wave front is well-defined, and moving smartly through the gas, then the peak pressure will pass smoothly and uniformly along successive neighbouring elements. Next, I converted elements #'s into distances in meters. I did this by multiplying the breech-to-shell distance at every time by the appropriate fraction of tube length, calculated by dividing the element # by the total number of elements. The following graph shows the location of the instantaneous peak-pressure element with respect to simulation time.



The two red arrows identify the same pair of shock waves that were identified with red arrows in the surface plot above. It looks like the shock wave that took place while the gas was still being compressed was much better defined than the one (marked with the red arrows) which took place while the gas was starting to expand. The noise in the element locations which happens before the shell is released (6.70 ms) and again after about 15 ms indicates that there is no shock front present, so the location of the instantaneous peak pressure is set randomly by hot spots in the burning propellant.

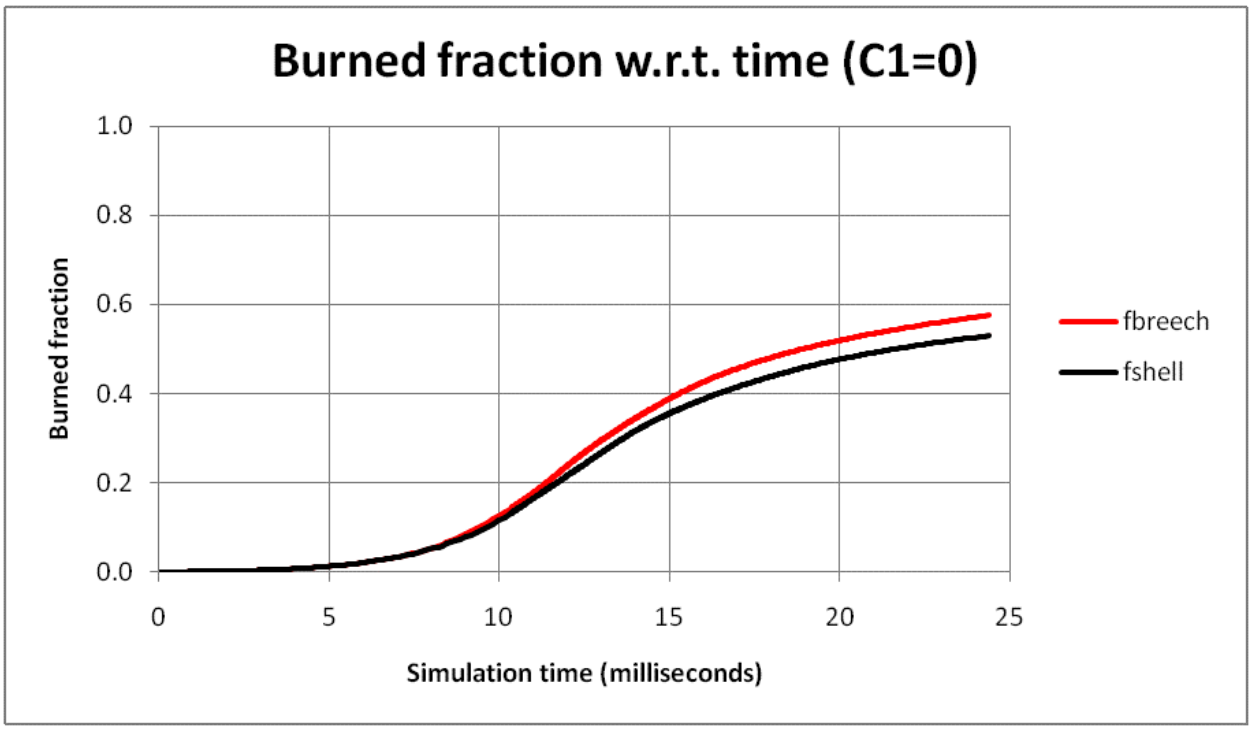
We can calculate the speed of these shock waves. Take the first one, which leaves the breech at about 9.0 ms into the simulation. It reaches the shell at about 10.5 ms, by which time the rear face of the shell has moved out to about 1.2 meters from the breech. The shock wave travelled 1.2 meters in 1.5 ms, which is a speed of 800 meters per second.

Look at the shock which is reflected back from the shell about 13 ms into the run. At first blush, it looks like the shock front made a sudden leap, at nearly infinite speed. No, that is the wrong interpretation. What is happening is that the shock wave is starting to dissipate, and the identity of the element which had the highest instantaneous pressure changed. It was out-of-sequence, and the shock wave front is no longer well-defined.

This simulation is not a good representation of what one would expect for this gun. Here is what one would expect.

- The peak pressure should be around 3,750 atmospheres, almost twice what is predicted here.
- The shell's exit speed should be around 750 meters per second, at which speed it would have $(750/595)^2 = 1.59$ times as much kinetic energy as it has when it exits the gun at 595 meters per second, as predicted here.

There is a reason for this shortfall. It is illustrated in the following graph, which shows that only about one-half of the propellant was burned. Slightly more propellant was burned at the breech (57.6%) than inside the last element (53.0%)., but the fact is that burning was too slow.

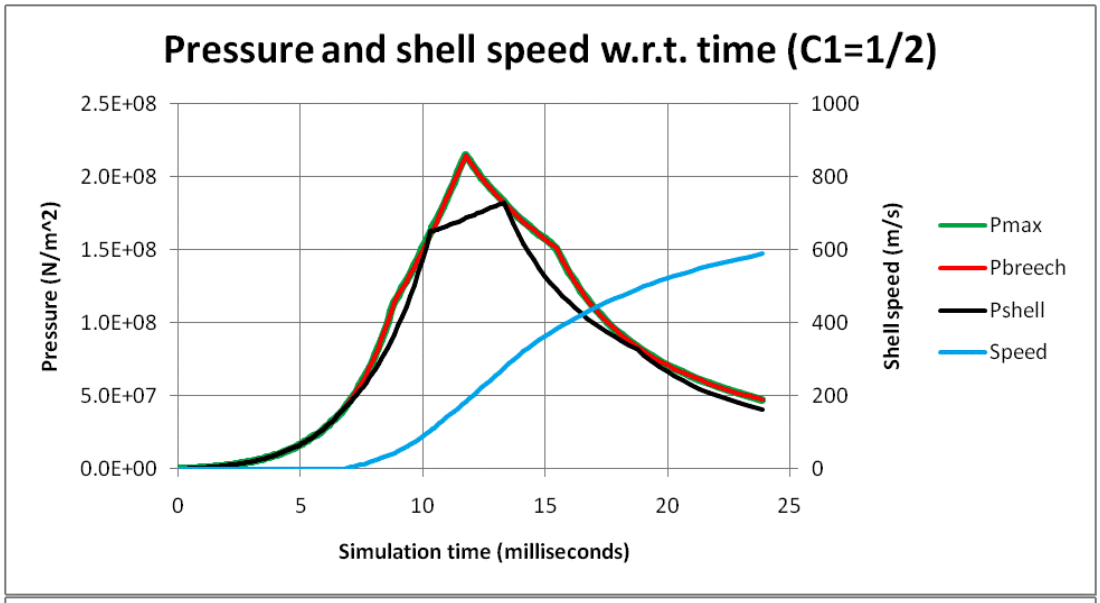


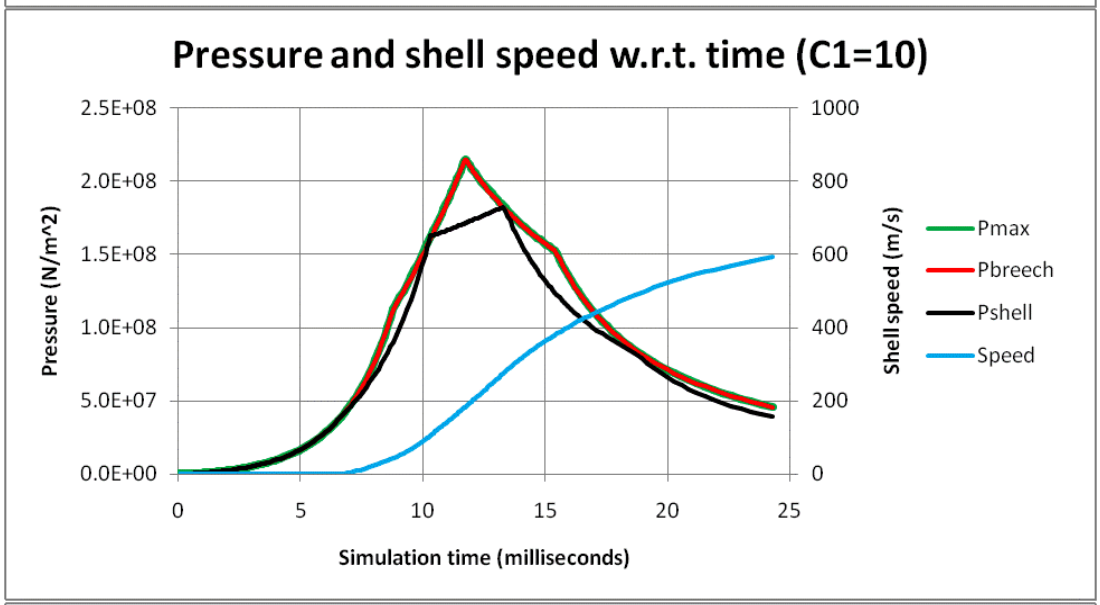
And, there is a good reason why more propellant was not consumed. In this paper, I modeled the grains of propellant as solid cylinders. I ignored the fact that the grains are pierced by several holes drilled through from one end-face to the other. These holes provide additional surface area on which burning takes place. The holes allow the grains to be burned from the inside out, as well as from the outside in. The net result is that more propellant is consumed, more heat is generated more quickly than from solid grains, and there is higher pressure and greater shell speed. I will look at the effect of these holes in the next paper.

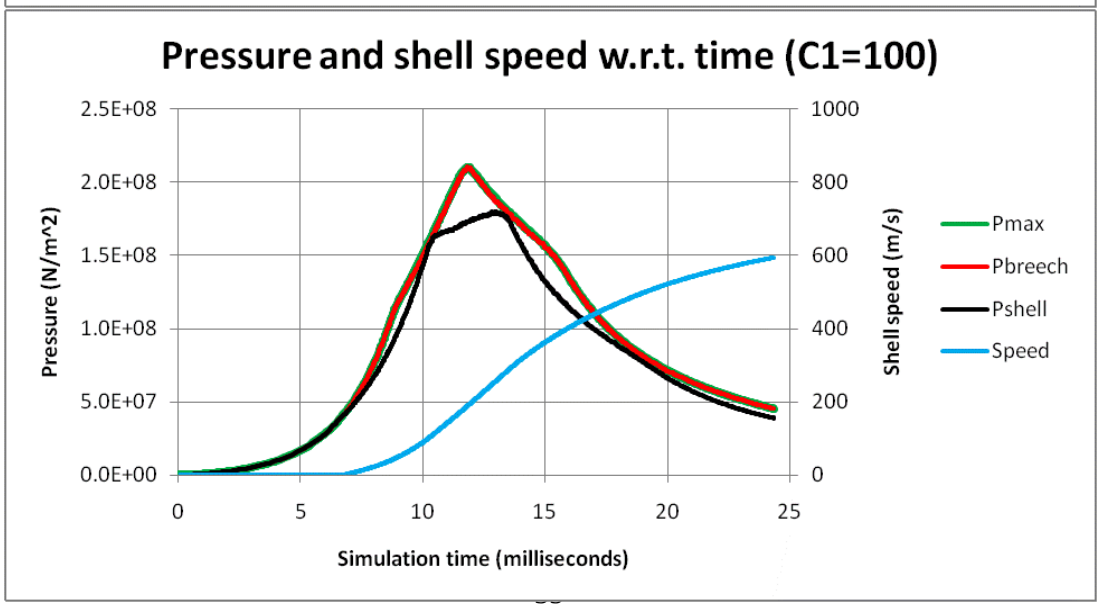
As I have said, the numerical procedure did not require the use of artificial viscosity. Apparently, the time steps were short enough that the pressures calculated using the normal unadjusted equations of state were able to change quickly enough to withstand on-coming shocks. The time steps got as low as 60 ns (nanoseconds) during the run.

Even though artificial viscosity was not necessary, it is still interesting to look at its effects. I have placed on the following page three "Pressure and shell speed w.r.t. time" graphs, all of which are comparable to the one above with that caption. The three graphs show the results of using non-zero artificial viscosity. The C_1 artificial viscosity coefficient is set to ½, 10 and 100, respectively, in the three cases. In all cases, the second artificial viscosity coefficient C_2 (which is based on a speed-of-sound factor), is set to one-tenth of C_1.

The main observation is this. The $C_1 = 100$ case shows some softening of the sharpness at the shock wave fronts. If one looks closely, there is slight evidence of this kind of softening at $C_1 = 10$. At $C_1 = ½$, the shock fronts look as sharp as they were with no artificial viscosity at all. It is worth bearing in mind that the purpose of the artificial viscosity is not to make the shock waves somehow disappear, but only to soften them just enough to permit the numerical integration to survive.


Pressure and shell speed w.r.t. time (C1=1/2)
Pressure (N/m^2)
2.5E+08
2.0E+08
1.5E+08
1.0E+08
5.0E+07
0.0E+00
Shell speed (m/s)
1000
800
600
400
200
0
0
5
10
15
20
25
Simulation time (milliseconds)
Pmax
Pbreech
Pshell
Speed


Pressure and shell speed w.r.t. time (C1=10)
Pressure (N/m^2)
2.5E+08
2.0E+08
1.5E+08
1.0E+08
5.0E+07
0.0E+00
Shell speed (m/s)
1000
800
600
400
200
0
0
5
10
15
20
25
Simulation time (milliseconds)
Pmax
Pbreech
Pshell
Speed


Pressure and shell speed w.r.t. time (C1=100)
Pressure (N/m^2)
2.5E+08
2.0E+08
1.5E+08
1.0E+08
5.0E+07
0.0E+00
Shell speed (m/s)
1000
800
600
400
200
0
0
5
10
15
20
25
Simulation time (milliseconds)
Pmax
Pbreech
Pshell
Speed

Also note that the use of artificial viscosity, and the dissipation which accompanies it, did not cause any change in the speed at which the shell exits from the barrel.

In the next paper, I will add holes to the web of the grains.

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 numerical simulation

The program consists of a Windows Forms application (Form1) and two modules, Variables and Procedures.

Listing of form Form1

```
Option Strict On
Option Explicit On

Public Class Form1
    Inherits System.Windows.Forms.Form

    Public Sub New()
        InitializeComponent()
        With Me
            Text = "Naval gun simulation"
            FormBorderStyle = Windows.Forms.FormBorderStyle.None
            Size = New Drawing.Size(1000, 700)
            CenterToScreen()
            MinimizeBox = True
            MaximizeBox = True
            FormBorderStyle = Windows.Forms.FormBorderStyle.Fixed3D
            With Me
                Controls.Add(buttonSimulate) : buttonSimulate.BringToFront()
                Controls.Add(buttonExit) : buttonExit.BringToFront()
                Controls.Add(labelResult) : labelResult.BringToFront()
            End With
            Visible = True
            PerformLayout()
            BringToFront()
        End With
    End Sub

    '////////////////////////////////////////////////////////////////////////////////
    '////////////////////////////////////////////////////////////////////////////////
    '// Controls for MainForm.

    Private WithEvents buttonSimulate As New Windows.Forms.Button With _
        {.Size = New Drawing.Size(120, 30), _
         .Location = New Drawing.Point(5, 5), _
         .Text = "Simulate", .TextAlign = ContentAlignment.MiddleCenter}

    Private WithEvents buttonExit As New Windows.Forms.Button With _
        {.Size = New Drawing.Size(120, 30), _
         .Location = New Drawing.Point(5, 40), _
         .Text = "Exit", .TextAlign = ContentAlignment.MiddleCenter}

    Public labelResult As New Windows.Forms.Label With _
        {.Size = New Drawing.Size(950, 600), _
         .Location = New Drawing.Point(125, 5), _
         .Text = "", .TextAlign = ContentAlignment.TopLeft, .Visible = True}
```

```
    '////////////////////////////////////////////////////////////////////////////////
    '////////////////////////////////////////////////////////////////////////////////
    '// Handlers for controls for MainForm.

    Private Sub buttonSimulate_Click() Handles buttonSimulate.MouseClick
        RunCompleteSimulation()
    End Sub

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

End Class
```

Listing of module Variables

```
Option Strict On
Option Explicit On

Public Module Variables

    ' Simulation parameters:
    '   NE = Number of gas elements
    '   MaxSimTime = Maximum length of simulation, in seconds
    '   deltaT = Initial duration of a time step, in seconds
    '   Time = Simulation time, in seconds
    '   deltaTSave = Time interval between writes to output text files
    '   TimeOfNextSave = Time of next write to output text files
    '   deltaNESave = Spatial interval between gas elements which will be saved
    Public NE As Int32 = 2000
    Public MaxSimTime As Double = 0.05
    Public deltaT As Double = 0.0000001
    Public Time As Double
    Public deltaTSave As Double = 0.00001
    Public TimeOfNextSave As Double
    Public deltaNESave As Int32 = 20

    ' Variables used to calculate the length of a time step
    '   MaxChangeAllowed = If per-step change in Pi() or RHOi() exceeds this, reduce TS
    '   MinChangeAllowed = If per-step change in Pi() and RHOi() are less, increase TS
    '   DecreaseInTS = Fraction to decrease time step if pressure change is too high
    '   IncreaseInTS = Fraction to increase time step is pressure change is too low
    '   MaxPressure = Maximum of pressure in all elements, for display purposes only
    '   MaxPressureIndex = Index of element with maximum pressure, for display only
    Public MaxChangeAllowed As Double = 0.001
    Public MinChangeAllowed As Double = 0.00025
    Public DecreaseInTS As Double = 0.9
    Public IncreaseInTS As Double = 1.01
    Public MaxPressure As Double
    Public MaxPressureIndex As Int32

    ' Important times:
    '   TimeOfShellStart = Time at which the shell starts to move
    '   TimeOfShellExit = Time at which the shell leaves the barrel
    Public TimeofShellStart As Double
    Public TimeOfShellExit As Double
```

```
' Physical parameters of the gun:
'   Lchamber = Length of chamber
'   Lbarrel = Shell travel distance
'   Abarrel = Area of open barrel
'   Mshell = Mass of the shell
'   Pengband = Engraving band pressure
Public Lchamber As Double = 1.03        ' meters
Public Lbarrel As Double = 5.97         ' meters
Public Abarrel As Double = 0.0127       ' square meters
Public Mshell As Double = 31.8          ' kilograms
Public Pengband As Double = Val("4E7")  ' Newtons per square meter

' Physical parameters of the propellant:
'   Mcharge = Total mass of propellant
'   Dgrain = Diameter of a grain of propellant
'   Lgrain = Original length of a grain of propellant
'   RHOgrain = Crystalline density of solid propellant
'   RHOLoad = Loading denisty of solid propellant
'   Q0 = Heat released from burning one kilogram of propellant
'   Rgrain = Original radius of a grain of propellant
'   LgrainEff = Effective length of one grain, per element
'   AgrainEff = Effective cross-sectional area of one grain, per element
Public Mcharge As Double = 8.85         ' kilograms
Public Dgrain As Double = 0.0069        ' meters
Public Lgrain As Double = 0.012         ' meters
Public RHOgrain As Double = 1660        ' kilograms per cubic meter
Public RHOLoad As Double = 680          ' kilograms per cubic meter
Public Q0 As Double = 3430000           ' Joules per kilogram
Public Rgrain As Double = Dgrain / 2    ' meters
Public LgrainEff As Double
Public AgrainEff As Double

' Physical parameters of the gas:
'   StoichRatio = Moles of gas produced per kilogram of propellant
'   Ridc = R, the Ideal Gas Constant
'   C1, C2 = Artificial viscosity coefficients
'   Cfr = Mass compensation coefficient for frictional forces
'   Bcovolume = Basic ideal gas co-volume correction "b"
Public StoichRatio As Double = 40       ' moles per kilogram
Public Rigc As Double = 8.31446         ' Joules per mole-degK
Public C1 As Double = 0                 ' Artificial viscosity coefficient #1
Public C2 As Double = C1 / 10           ' Artificial viscosity coefficient #2
Public Cfr As Double = 0.167            ' Mass compensation for frictional forces
Public Bcovolume As Double = 0.00095    ' cubic meters per kilogram

' Initial conditions of the ambient air inside the barrel:
'   Patm = Atmospheric pressure
'   Tatm = Temperature inside the chamber
'   AirMW = Molecular weight of dry air
'   AirNi = Number of moles of original air inside each element
'   AirMi = Mass of original air inside each element
Public Patm As Double = 101300          ' Newtons per square meter
Public Tatm As Double = 100             ' degC
Public AirMW As Double = 0.02897        ' kilograms per mole
Public AirNi As Double                  ' moles
Public AirMi As Double                  ' kilograms
```

```
' Gas element variables:
' Values for all elements and for two consecutive time steps are stored
' in the following variables.
'   MiT = Total mass of "stuff" inside element #i (constant)
'   MiP0 = Original mass of propellant inside element #i
'   MiP(NE) = Mass of unburned propellant inside element #i at time T
'   MiG(NE) = Mass of propellant gas inside element #i at time T
'   Ni(NE) = Total moles of gas inside element #i at time T
'   Xi(NE) = Location of boundary faces at time T (At breech, Xi(0)=0 always)
'   Vi(NE) = Speed of boundary faces at time T-deltaT/2 (Vi(0)= 0 always)
'   Pi(NE) = Static pressure inside element #i at time T
'   Wi(NE) = Artificial viscosity inside element #i at time T
'   VOLi(NE) = Total volume of element #i at time T
'   VOLGasi(NE) = Gas volume of element #i at time T
'   BCoVoli(NE) = Co-volume correction inside element #i at time T
'   Ti(NE) = Temperature inside element #i at time T
'   Ui(NE) = Internal energy of element #i at time T
'   RHOi(NE) = Density inside element #i at time T
'   GAMMAi(NE) = Ratio of specific heats inside element #i at time T
'   Si(NE) = Speed of sound inside element #i at time T
'   deltaQi(NE) = Heat added to element #i during one time step
'   deltaNi(NE) = Moles of gas added to element #i during one time step
Public MiT As Double                                ' kilograms
Public MiP0 As Double                               ' kilograms
Public MiP(NE), MiP_previous(NE) As Double          ' kilograms
Public MiG(NE), MiG_previous(NE) As Double          ' kilograms
Public Ni(NE), Ni_previous(NE) As Double            ' moles
Public Xi(NE), Xi_previous(NE) As Double            ' meters
Public Vi(NE), Vi_previous(NE) As Double            ' meters per second
Public Pi(NE), Pi_previous(NE) As Double            ' Newtons per square meter
Public Wi(NE), Wi_previous(NE) As Double            ' Newtons per square meter
Public VOLi(NE), VOLi_previous(NE) As Double        ' cubic meters
Public VOLGasi(NE), VOLGasi_previous(NE) As Double  ' cubic meters
Public BCoVoli(NE), BCoVoli_previous(NE) As Double  ' cubic meters
Public Ti(NE), Ti_previous(NE) As Double            ' degK
Public Ui(NE), Ui_previous(NE) As Double            ' Joules
Public RHOi(NE), RHOi_previous(NE) As Double        ' kilograms per cubic meter
Public GAMMAi(NE), GAMMAi_previous(NE) As Double
Public Si(NE), Si_previous(NE) As Double            ' meters per second
Public deltaQi(NE), deltaQi_previous(NE) As Double  ' Joules
Public deltaNi(NE), deltaNi_previous(NE) As Double  ' moles

' Propellant variables:
' Values for all elements and for two consecutive time steps are stored in the
' following variables.
'   Bi(NE) = Burn rate inside element #i at time T
'   Rgraini(NE) = Radius of grains in element #i at time T
'   PropVOLi(NE) = Volume of propellant in element #i at time T
'   deltaMi(NE) = Mass of propellant burned during one time step
'   fi(NE) = Fraction of propellant burned in element #i at time T
Public Bi(NE), Bi_previous(NE) As Double               ' meters per second
Public Rgraini(NE), Rgraini_previous(NE) As Double     ' meters
Public PropVOLi(NE), PropVOLi_previous(NE) As Double   ' cubic meters
Public deltaMi(NE), deltaMi_previous(NE) As Double     ' kilograms
Public fi(NE), fi_previous(NE) As Double
```

```
    ' Projectile variables:
    ' Values for the projectile for two consecutive time steps are stored in the
    ' following variables.
    '   Pshell = Pressure on rear face of shell at time T
    '   Xshell = Location of rear face of shell at time T
    '   Vshell = Speed of shell at time T-deltaT/2
    '   ACCshell = Acceleration of shell at time T
    '   CanShellMove is True when the shell has broken free from its band
    Public Pshell, Pshell_previous As Double            ' Newtons per square meter
    Public Xshell, Xshell_previous As Double            ' meters
    Public Vshell, Vshell_previous As Double            ' meters per second
    Public ACCshell, ACCshell_previous As Double        ' meters per second^2
    Public CanShellMove As Boolean

    ' Text file processing
    Public ThisDirectory As String = FileSystem.CurDir.ToString
    Public TextOutputFileName As String = "Naval_gun_simulation_"
    ' Log of screen display
    Public FileWriterMaster As System.IO.StreamWriter
    ' Gas element variables
    Public FileWriterMiP As System.IO.StreamWriter
    Public FileWriterMiG As System.IO.StreamWriter
    Public FileWriterNi As System.IO.StreamWriter
    Public FileWriterXi As System.IO.StreamWriter
    Public FileWriterVi As System.IO.StreamWriter
    Public FileWriterPi As System.IO.StreamWriter
    Public FileWriterWi As System.IO.StreamWriter
    Public FileWriterVOLi As System.IO.StreamWriter
    Public FileWriterVOLGasi As System.IO.StreamWriter
    Public FileWriterBCoVoli As System.IO.StreamWriter
    Public FileWriterTi As System.IO.StreamWriter
    Public FileWriterUi As System.IO.StreamWriter
    Public FileWriterRHOi As System.IO.StreamWriter
    Public FileWriterGAMMAi As System.IO.StreamWriter
    Public FileWriterSi As System.IO.StreamWriter
    Public FileWriterdeltaQi As System.IO.StreamWriter
    Public FileWriterdeltaNi As System.IO.StreamWriter
    ' Propellant variables
    Public FileWriterBi As System.IO.StreamWriter
    Public FileWriterRgraini As System.IO.StreamWriter
    Public FileWriterPropVOLi As System.IO.StreamWriter
    Public FileWriterdeltaMi As System.IO.StreamWriter
    Public FileWriterfi As System.IO.StreamWriter
    ' Projectile variables
    Public FileWriterShell As System.IO.StreamWriter

End Module
```

Listing of module Procedures

```
Option Strict On
Option Explicit On

' List of subroutines:
'   InitializeForSimulation()
'   RunFullSimulation()
'   ExecuteOneTimeStep()
'   ShiftAllValuesToPreviousVariables()
'   FindMaximumPressure()
'   OpenAllOutputTextFiles()
'   CloseAllOutputTextFiles()
'   WriteHeadersToAllOutputTextFiles()
'   WriteDataRowToAllOutputTextFiles()

Public Module Procedures

    Public Sub InitializeForSimulation()
        ' This subroutine sets all quantities to their initial values, so
        ' everything is ready to start the first time step.
        '
        ' ------------------------------------------------------------
        ' Deal with the ambient air in the chamber prior to firing
        Dim TotalVolumeOfAir As Double = _
            Mcharge * ((1 / RHOLoad) - (1 / RHOgrain))
        Dim TotalMolesOfAir As Double = _
            Patm * TotalVolumeOfAir / (Rigc * (Tatm + 273.15))
        Dim TotalMassOfAir As Double = AirMW * TotalMolesOfAir
        AirNi = TotalMolesOfAir / NE
        AirMi = TotalMassOfAir / NE
        ' ------------------------------------------------------------
        '
        ' Set the initial MASS of propellant inside each element
        MiP0 = Mcharge / NE
        MiT = MiP0 + AirMi
        For J As Int32 = 1 To NE Step 1
            MiP(J) = MiP0
            MiG(J) = 0
        Next J
        '
        ' Set the initial VOLUME of propellant inside each element
        For J As Int32 = 1 To NE Step 1
            PropVOLi(J) = MiP(J) / RHOgrain
            Rgraini(J) = Rgrain
        Next J
        '
        ' Set the effective area and length of grains in each element
        ' This is done for the sole purpose of calculating the volume of propellant
        AgrainEff = Math.PI * (Rgrain ^ 2)
        LgrainEff = ((Mcharge / NE) / RHOgrain) / AgrainEff
        '
        ' Set the unburned fractions
        For J As Int32 = 1 To NE Step 1
            fi(J) = 1
        Next J
        '
```

```
    ' Set initial locations of boundary faces
    Xi(0) = 0
    For J As Int32 = 1 To NE Step 1
        Xi(J) = Lchamber * J / NE
    Next J
    Xshell = Lchamber
    '
    ' Set the initial speeds
    For J As Int32 = 1 To NE Step 1
        Vi(J) = 0
    Next J
    Vshell = 0
    '
    ' Set the initial pressure, volume, density and number of moles of air
    For J As Int32 = 1 To NE Step 1
        Pi(J) = Patm
        VOLi(J) = Abarrel * (Xi(J) - Xi(J - 1))
        RHOi(J) = AirMi / (VOLi(J) - PropVOLi(J))
        Ni(J) = AirNi
    Next J
    '
    ' Set the internal energies
    For J As Int32 = 1 To NE Step 1
        Ui(J) = (Ni(J) * Rigc * (Tatm + 273.15)) / (1.333 - 1)
    Next J
End Sub

Public Sub RunCompleteSimulation()
    Form1.labelResult.Text = "Starting simulation ..."
    Form1.labelResult.Refresh()
    '
    ' Initialize the vectors
    InitializeForSimulation()
    '
    ' Open the text output files
    OpenAllOutputTextFiles()
    WriteHeadersToAllOutputTextFiles()
    '
    ' Save the starting values
    Time = 0
    WriteDataRowToAllOutputTextFiles()
    TimeOfNextSave = Time + deltaTSave
    '
    ' Set the control parameters
    CanShellMove = False
    Dim ScreenUpdateInterval As Int32 = 250
    Dim ScreenUpdateCounter As Int32 = 250
    Do
        Time = Time + deltaT
        ' Test to see if the simulation has timed out
        If (Time >= MaxSimTime) Then
            Exit Do
        End If
        ' Shift the present variables into their "previous" variants
        ShiftAllValuesToPreviousVariables()
        ' Run one time step
        ExecuteOneTimeStep()
```

```
' Find the change in maximum pressure
FindMaximumPressure(MaxPressure, MaxPressureIndex)
' Test to see if it is time to write to the output text files
If (Time >= TimeOfNextSave) Then
    WriteDataRowToAllOutputTextFiles()
    TimeOfNextSave = Time + deltaTSave
End If
' Test to see if the shell can start moving
If ((CanShellMove = False) And _
    (Pi(NE) >= Pengband)) Then
    CanShellMove = True
    TimeofShellStart = Time
    Form1.labelResult.Text = Strings.Right( _
        Form1.labelResult.Text & vbCrLf & _
        "Shell starts moving at time = " & Trim(Str(Time)), 6000)
    Form1.labelResult.Refresh()
    FileWriterMaster.WriteLine( _
        "Shell starts moving at time = " & Trim(Str(Time)))
    Threading.Thread.Sleep(2000)
End If
' Test to see if the shell has left the barrel
If (Xshell > (Lchamber + Lbarrel)) Then
    TimeOfShellExit = Time
    Form1.labelResult.Text = Strings.Right( _
        Form1.labelResult.Text & vbCrLf & _
        "Shell exits at time = " & Trim(Str(Time)), 6000)
    Form1.labelResult.Refresh()
    FileWriterMaster.WriteLine( _
        "Shell exits at time = " & Trim(Str(Time)))
    ' Write the final values to the output text files
    WriteDataRowToAllOutputTextFiles()
    Exit Do
End If
' Update the screen display
ScreenUpdateCounter = ScreenUpdateCounter + 1
If (ScreenUpdateCounter >= ScreenUpdateInterval) Then
    Form1.labelResult.Text = Strings.Right( _
        Form1.labelResult.Text & vbCrLf & _
        "T(us)= " & FormatNumber(Time * 1000000, 2) & _
        "  dT(us)= " & FormatNumber(deltaT * 1000000, 6) & _
        "  MaxP= " & FormatNumber(MaxPressure, 0, , TriState.True) & _
        "  MaxPIndex= " & Trim(Str(MaxPressureIndex)) & _
        "  P1= " & FormatNumber(Pi(1), 0, , , TriState.True) & _
        "  Pne= " & FormatNumber(Pi(NE), 0, , , TriState.True) & _
        "  Bne= " & FormatNumber(Bi(NE), 4) & _
        "  fne= " & FormatNumber(fi(NE), 5) & _
        "  f1= " & FormatNumber(fi(1), 5) & _
        "  Vs= " & FormatNumber(Vshell, 2) & _
        "  Wne= " & FormatNumber(Wi(NE), 0, , , TriState.True), _
        6000)
    Form1.labelResult.Refresh()
    ' Save the new line to the Master text output file as well
    FileWriterMaster.WriteLine( _
        "Time= ," & Trim(Str(Time)) & _
        ", with deltaT= ," & Trim(Str(deltaT)) & _
        ", MaxP= ," & Trim(Str(MaxPressure)) & _
        ", MaxPIndex= ," & Trim(Str(MaxPressureIndex)) & _
        ", P1= ," & Trim(Str(Pi(1))) & _
```

```
                ", Pne= ," & Trim(Str(Pi(NE))) & _
                ", Wne= ," & Trim(Str(Wi(NE))) & _
                ", Une= ," & Trim(Str(Ui(NE))) & _
                ", Bne= ," & Trim(Str(Bi(NE))) & _
                ", fne= ," & Trim(Str(fi(NE))) & _
                ", f1= ," & Trim(Str(fi(1))) & _
                ", Xs= ," & Trim(Str(Xshell)) & _
                ", Vs= ," & Trim(Str(Vshell)))
            ScreenUpdateCounter = 0
        End If
        ' Based on change in maximum pressure, change the time step if necessary
        ' Equation (?????*****)
        Dim lRelChangeInP As Double
        Dim lMaxRelChangeInP As Double
        Dim lRelChangeInRHO As Double
        Dim lMaxRelChangeInRHO As Double
        lMaxRelChangeInP = Val("-1E+20")
        lMaxRelChangeInRHO = Val("-1E+20")
        For J As Int32 = 1 To NE Step 1
            lRelChangeInP = _
                Math.Abs((Pi(J) - Pi_previous(J)) / Pi_previous(J))
            lRelChangeInRHO = _
                Math.Abs((RHOi(J) - RHOi_previous(J)) / RHOi_previous(J))
            If (lRelChangeInP > lMaxRelChangeInP) Then
                lMaxRelChangeInP = lRelChangeInP
            End If
            If (lRelChangeInRHO > lMaxRelChangeInRHO) Then
                lMaxRelChangeInRHO = lRelChangeInRHO
            End If
        Next J
        If ((lMaxRelChangeInP > MaxChangeAllowed) Or _
            (lMaxRelChangeInRHO > MaxChangeAllowed)) Then
            deltaT = DecreaseInTS * deltaT
        End If
        If ((lMaxRelChangeInP < MinChangeAllowed) And _
            (lMaxRelChangeInRHO < MinChangeAllowed)) Then
            deltaT = IncreaseInTS * deltaT
        End If
    Loop
    ' Close the output text file
    CloseAllOutputTextFiles()
End Sub

Public Sub ExecuteOneTimeStep()
    ' This subroutine advances the simulation from time t to time t+deltaT once the
    ' speeds of the boundary faces at time t+deltaT/2 have been calculated.
    ' For example, Vi(j) is the speed of boundary face #j at time t+deltaT/2
    '              Pi(j) is the pressure in element #j at time t+deltaT
    ' If the Boolean flag CanShellMove is False, then the shell, and the RHS
    ' boundary face of element #NE, are not permitted to move.
    '
    ' Step #1: Equation (A)
    Xi(0) = 0
    For J As Int32 = 1 To (NE - 1) Step 1
        Xi(J) = Xi_previous(J) + (Vi(J) * deltaT)
    Next J
    If (CanShellMove = True) Then
        Xi(NE) = Xi_previous(NE) + (Vshell * deltaT)
```

```
Else
    Xi(NE) = Xi_previous(NE)
End If
Xshell = Xi(NE)
'
' Step #2: Equation (B)
For J As Int32 = 1 To NE Step 1
    VOLi(J) = (Xi(J) - Xi(J - 1)) * Abarrel
Next J
'
' Step #3: Burning equations
For J As Int32 = 1 To NE Step 1
    ' Equation (C)
    Bi(J) = Math.Exp(( _
        (0.046696597 * ((Math.Log(Pi_previous(J) / Patm)) ^ 2)) + _
        (0.34808898 * Math.Log(Pi_previous(J) / Patm)) + _
        -0.572295873)) / 1000
    ' Direct calculations for Equation (D)
    ' Calculate the new radius of the grain after the burning in this time step
    Rgraini(J) = Rgraini_previous(J) - (Bi(J) * deltaT)
    ' Calculate the new unburned propellant volume
    PropVOLi(J) = Math.PI * (Rgraini(J) ^ 2) * LgrainEff
    ' Calculate the decrease in volume and corresponding mass burned
    deltaMi(J) = RHOgrain * (PropVOLi_previous(J) - PropVOLi(J))
    '
    ' Check that burning does not consume more than 100% of the remaining mass
    If (deltaMi(J) > MiP_previous(J)) Then
        Rgraini(J) = 0
        PropVOLi(J) = 0
        deltaMi(J) = MiP_previous(J)
    End If
    ' Equation (E)
    MiP(J) = MiP_previous(J) - deltaMi(J)
    MiG(J) = MiG_previous(J) + deltaMi(J)
    fi(J) = MiG(J) / MiP0
Next J
'
' Step #4: Equation (F)
For J As Int32 = 1 To NE Step 1
    deltaQi(J) = Q0 * deltaMi(J)
Next J
'
' Step #5: Density
For J As Int32 = 1 To NE Step 1
    ' Equation (G)
    VOLGasi(J) = VOLi(J) - (MiP(J) / RHOgrain)
    ' Equation (H)
    RHOi(J) = (MiG(J) + AirMi) / VOLGasi(J)
Next J
'
' Step #6: Co-volume correction
For J As Int32 = 1 To NE Step 1
    ' Equation (I)
    BCoVoli(J) = Bcovolume * (MiG(J) + AirMi) / (1 + (2 * RHOi(J) / 500))
    ' Equation (J)
    deltaNi(J) = 40 * deltaMi(J)
    ' Equation (K)
    Ni(J) = Ni_previous(J) + deltaNi(J)
```

```
Next J
'
' Step #7: Adiabtaic Index
For J As Int32 = 1 To NE Step 1
    ' Equation (L)
    GAMMAi(J) = 1.333 + (0.567 * RHOi(J) / 1200)
Next J
'
' Step #8: Internal energy and pressure
For J As Int32 = 1 To NE Step 1
    Dim MaxNumOfIterations As Double = 1000
    Dim NumOfIterations As Double = 0
    ' Starting guess
    Pi(J) = Pi_previous(J)
    ' Loop
    Do
        ' Equation (M)
        Si(J) = Math.Sqrt(GAMMAi(J) * Pi(J) / RHOi(J))
        ' Carry out the relative speed test
        Dim deltaV As Double
        If (J = 1) Then
            deltaV = -Vi(J)
        Else
            deltaV = Vi(J - 1) - Vi(J)
        End If
        ' Equation (N)
        If (deltaV > 0) Then
            Wi(J) = RHOi(J) * ( _
                (C1 * deltaV * deltaV) + _
                (C2 * Si(J) * Math.Abs(deltaV)))
        Else
            Wi(J) = 0
        End If
        ' Equation (O)
        Dim lFactor1 As Double
        Dim lFactor2 As Double
        lFactor1 = Pi(J) + Wi(J) + Pi_previous(J) + Wi_previous(J)
        lFactor2 = VOLi(J) - VOLi_previous(J)
        Ui(J) = Ui_previous(J) + deltaQi(J) - (0.5 * lFactor1 * lFactor2)
        ' Equation (P)
        Dim lFactor3 As Double
        Dim P_New As Double
        lFactor3 = VOLGasi(J) - BCoVoli(J)
        P_New = (GAMMAi(J) - 1) * Ui(J) / lFactor3
        ' Test for negative pressure
        If (P_New <= 0) Then
            MsgBox("Error: Negative pressure.  Reduce time step.")
            Exit Sub
        End If
        ' Test for convergence
        If (Math.Abs((P_New - Pi(J)) / Pi(J)) < 0.000000001) Then
            Pi(J) = P_New
            Exit Do
        End If
        ' Restrict the per-iteration adjustment in pressure
        Dim MaxAbsChange As Double
        MaxAbsChange = Math.Min(0.1 * Pi(J), Math.Abs(P_New - Pi(J)))
        If (P_New > Pi(J)) Then
```

```
                Pi(J) = Pi(J) + MaxAbsChange
            Else
                Pi(J) = Pi(J) - MaxAbsChange
            End If
            ' Test for too many iterations
            NumOfIterations = NumOfIterations + 1
            If (NumOfIterations > MaxNumOfIterations) Then
                MsgBox("Error: Too many iterations.")
                Exit Sub
            End If
        Loop
    Next J
    '
    ' Step #9: Temperature
    ' Equation(Q)
    For J As Int32 = 1 To NE Step 1
        Ti(J) = Pi(J) * (VOLGasi(J) - BCoVoli(J)) / (Ni(J) * Rigc)
    Next J
    '
    ' Step #10: Acceleration of the shell
    ' Equation(R)
    Dim EffectiveMshell As Double = Mshell / (1 - Cfr)
    Pshell = Pi(NE)
    ACCshell = (Pshell - Patm) * Abarrel / EffectiveMshell
    '
    ' Step #11: Advance the speeds
    ' Equation (S)
    For J As Int32 = 1 To (NE - 1) Step 1
        Dim lFactor1 As Double
        lFactor1 = Pi(J + 1) + Wi(J + 1) - Pi(J) - Wi(J)
        Vi(J) = Vi_previous(J) - (deltaT * lFactor1 * Abarrel / MiT)
    Next J
    ' Equation (T)
    If (CanShellMove = True) Then
        Vshell = Vshell_previous + (ACCshell * deltaT)
    Else
        Vshell = 0
    End If
    Vi(NE) = Vshell
End Sub

Public Sub ShiftAllValuesToPreviousVariables()
    ' This subroutine is called at the end of every time step.  It moves the values
    ' just calculated into their "previous" variants, in preparation for the next
    ' time step.
    Xi_previous(0) = Xi(0)
    For J As Int32 = 1 To NE Step 1
        ' Gas element variables
        MiP_previous(J) = MiP(J)
        MiG_previous(J) = MiG(J)
        Ni_previous(J) = Ni(J)
        Xi_previous(J) = Xi(J)
        Vi_previous(J) = Vi(J)
        Pi_previous(J) = Pi(J)
        Wi_previous(J) = Wi(J)
        VOLi_previous(J) = VOLi(J)
        VOLGasi_previous(J) = VOLGasi(J)
        BCoVoli_previous(J) = BCoVoli(J)
```

```
            Ti_previous(J) = Ti(J)
            Ui_previous(J) = Ui(J)
            RHOi_previous(J) = RHOi(J)
            GAMMAi_previous(J) = GAMMAi(J)
            Si_previous(J) = Si(J)
            deltaQi_previous(J) = deltaQi(J)
            deltaNi_previous(J) = deltaNi(J)
            ' Propellant variables
            Bi_previous(J) = Bi(J)
            Rgraini_previous(J) = Rgraini(J)
            PropVOLi_previous(J) = PropVOLi(J)
            deltaMi_previous(J) = deltaMi(J)
            fi_previous(J) = fi(J)
        Next J
        ' Projectile variables
        Pshell_previous = Pshell
        Xshell_previous = Xshell
        Vshell_previous = Vshell
        ACCshell_previous = ACCshell
    End Sub

    Public Sub FindMaximumPressure( _
        ByRef MaxP As Double, ByRef MaxPIndex As Int32)
        ' This function is called at the end of every time step.  It looks through the
        ' current values of the pressure in all elements.  It returns the maximum value
        ' and the index of the element with the maximum pressure.
        MaxP = Val("-1E+20")
        For J As Int32 = 1 To NE Step 1
            If (Pi(J) > MaxP) Then
                MaxP = Pi(J)
                MaxPIndex = J
            End If
        Next J
    End Sub

    Public Sub OpenAllOutputTextFiles()
        ' Log of screen display
        FileWriterMaster = New System.IO.StreamWriter( _
            ThisDirectory & "\" & TextOutputFileName & "Master.txt")
        ' Gas element variables
        FileWriterMiP = New System.IO.StreamWriter( _
            ThisDirectory & "\" & TextOutputFileName & "MiP.txt")
        FileWriterMiG = New System.IO.StreamWriter( _
            ThisDirectory & "\" & TextOutputFileName & "MiG.txt")
        FileWriterNi = New System.IO.StreamWriter( _
            ThisDirectory & "\" & TextOutputFileName & "Ni.txt")
        FileWriterXi = New System.IO.StreamWriter( _
            ThisDirectory & "\" & TextOutputFileName & "Xi.txt")
        FileWriterVi = New System.IO.StreamWriter( _
            ThisDirectory & "\" & TextOutputFileName & "Vi.txt")
        FileWriterPi = New System.IO.StreamWriter( _
            ThisDirectory & "\" & TextOutputFileName & "Pi.txt")
        FileWriterWi = New System.IO.StreamWriter( _
            ThisDirectory & "\" & TextOutputFileName & "Wi.txt")
        FileWriterVOLi = New System.IO.StreamWriter( _
            ThisDirectory & "\" & TextOutputFileName & "VOLi.txt")
        FileWriterVOLGasi = New System.IO.StreamWriter( _
            ThisDirectory & "\" & TextOutputFileName & "VOLGasi.txt")
```

```
    FileWriterBCoVoli = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "BCoVoli.txt")
    FileWriterTi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "Ti.txt")
    FileWriterUi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "Ui.txt")
    FileWriterRHOi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "RHOi.txt")
    FileWriterGAMMAi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "GAMMAi.txt")
    FileWriterSi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "Si.txt")
    FileWriterdeltaQi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "deltaQi.txt")
    FileWriterdeltaNi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "deltaNi.txt")
    ' Propellant variables
    FileWriterBi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "Bi.txt")
    FileWriterRgraini = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "Rgraini.txt")
    FileWriterPropVOLi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "PropVOLi.txt")
    FileWriterdeltaMi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "deltaMi.txt")
    FileWriterfi = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "fi.txt")
    ' Projectile variables
    FileWriterShell = New System.IO.StreamWriter( _
        ThisDirectory & "\" & TextOutputFileName & "Shell.txt")
End Sub

Public Sub CloseAllOutputTextFiles()
    ' Log of screen display
    FileWriterMaster.Close()
    ' Gas element variables
    FileWriterMiP.Close()
    FileWriterMiG.Close()
    FileWriterNi.Close()
    FileWriterXi.Close()
    FileWriterVi.Close()
    FileWriterPi.Close()
    FileWriterWi.Close()
    FileWriterVOLi.Close()
    FileWriterVOLGasi.Close()
    FileWriterBCoVoli.Close()
    FileWriterTi.Close()
    FileWriterUi.Close()
    FileWriterRHOi.Close()
    FileWriterGAMMAi.Close()
    FileWriterSi.Close()
    FileWriterdeltaQi.Close()
    FileWriterdeltaNi.Close()
    ' Propellant variables
    FileWriterBi.Close()
    FileWriterRgraini.Close()
    FileWriterPropVOLi.Close()
    FileWriterdeltaMi.Close()
```

```
    FileWriterfi.Close()
    ' Projectile variables
    FileWriterShell.Close()
End Sub

Public Sub WriteHeadersToAllOutputTextFiles()
    ' Log of screen display
    FileWriterMiP.Write("Time, ")
    ' Gas element variables
    FileWriterMiG.Write("Time, ")
    FileWriterNi.Write("Time, ")
    FileWriterXi.Write("Time, ")
    FileWriterVi.Write("Time, ")
    FileWriterPi.Write("Time, ")
    FileWriterWi.Write("Time, ")
    FileWriterVOLi.Write("Time, ")
    FileWriterVOLGasi.Write("Time, ")
    FileWriterBCoVoli.Write("Time, ")
    FileWriterTi.Write("Time, ")
    FileWriterUi.Write("Time, ")
    FileWriterRHOi.Write("Time, ")
    FileWriterGAMMAi.Write("Time, ")
    FileWriterSi.Write("Time, ")
    FileWriterdeltaQi.Write("Time, ")
    FileWriterdeltaNi.Write("Time, ")
    ' Propellant variables
    FileWriterBi.Write("Time, ")
    FileWriterRgraini.Write("Time, ")
    FileWriterPropVOLi.Write("Time, ")
    FileWriterdeltaMi.Write("Time, ")
    FileWriterfi.Write("Time, ")
    ' Projectile variables
    FileWriterShell.WriteLine("Time, Xshell, Vshell, ACCshell, Pshell")
    For K As Int32 = 0 To (NE - 1) Step deltaNESave
        Dim ElementIndex As Int32
        If (K = 0) Then
            ElementIndex = 1
        Else
            ElementIndex = K
        End If
        ' Gas element variables
        FileWriterMiP.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterMiG.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterNi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterXi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterVi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterPi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterWi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterVOLi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterVOLGasi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterBCoVoli.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterTi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterUi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterRHOi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterGAMMAi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterSi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterdeltaQi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterdeltaNi.Write(Trim(Str(ElementIndex)) & ", ")
```

```
        ' Propellant variables
        FileWriterBi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterRgraini.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterPropVOLi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterdeltaMi.Write(Trim(Str(ElementIndex)) & ", ")
        FileWriterfi.Write(Trim(Str(ElementIndex)) & ", ")
    Next K
    ' Gas element variables
    FileWriterMiP.WriteLine(Trim(Str(NE)))
    FileWriterMiG.WriteLine(Trim(Str(NE)))
    FileWriterNi.WriteLine(Trim(Str(NE)))
    FileWriterXi.WriteLine(Trim(Str(NE)))
    FileWriterVi.WriteLine(Trim(Str(NE)))
    FileWriterPi.WriteLine(Trim(Str(NE)))
    FileWriterWi.WriteLine(Trim(Str(NE)))
    FileWriterVOLi.WriteLine(Trim(Str(NE)))
    FileWriterVOLGasi.WriteLine(Trim(Str(NE)))
    FileWriterBCoVoli.WriteLine(Trim(Str(NE)))
    FileWriterTi.WriteLine(Trim(Str(NE)))
    FileWriterUi.WriteLine(Trim(Str(NE)))
    FileWriterRHOi.WriteLine(Trim(Str(NE)))
    FileWriterGAMMAi.WriteLine(Trim(Str(NE)))
    FileWriterSi.WriteLine(Trim(Str(NE)))
    FileWriterdeltaQi.WriteLine(Trim(Str(NE)))
    FileWriterdeltaNi.WriteLine(Trim(Str(NE)))
    ' Propellant variables
    FileWriterBi.WriteLine(Trim(Str(NE)))
    FileWriterRgraini.WriteLine(Trim(Str(NE)))
    FileWriterPropVOLi.WriteLine(Trim(Str(NE)))
    FileWriterdeltaMi.WriteLine(Trim(Str(NE)))
    FileWriterfi.WriteLine(Trim(Str(NE)))
End Sub

Public Sub WriteDataRowToAllOutputTextFiles()
    ' Gas element variables
    FileWriterMiP.Write(Trim(Str(Time)) & ", ")
    FileWriterMiG.Write(Trim(Str(Time)) & ", ")
    FileWriterNi.Write(Trim(Str(Time)) & ", ")
    FileWriterXi.Write(Trim(Str(Time)) & ", ")
    FileWriterVi.Write(Trim(Str(Time)) & ", ")
    FileWriterPi.Write(Trim(Str(Time)) & ", ")
    FileWriterWi.Write(Trim(Str(Time)) & ", ")
    FileWriterVOLi.Write(Trim(Str(Time)) & ", ")
    FileWriterVOLGasi.Write(Trim(Str(Time)) & ", ")
    FileWriterBCoVoli.Write(Trim(Str(Time)) & ", ")
    FileWriterTi.Write(Trim(Str(Time)) & ", ")
    FileWriterUi.Write(Trim(Str(Time)) & ", ")
    FileWriterRHOi.Write(Trim(Str(Time)) & ", ")
    FileWriterGAMMAi.Write(Trim(Str(Time)) & ", ")
    FileWriterSi.Write(Trim(Str(Time)) & ", ")
    FileWriterdeltaQi.Write(Trim(Str(Time)) & ", ")
    FileWriterdeltaNi.Write(Trim(Str(Time)) & ", ")
    ' Propellant variables
    FileWriterBi.Write(Trim(Str(Time)) & ", ")
    FileWriterRgraini.Write(Trim(Str(Time)) & ", ")
    FileWriterPropVOLi.Write(Trim(Str(Time)) & ", ")
    FileWriterdeltaMi.Write(Trim(Str(Time)) & ", ")
    FileWriterfi.Write(Trim(Str(Time)) & ", ")
```

```
For K As Int32 = 0 To (NE - 1) Step deltaNESave
    Dim ElementIndex As Int32
    If (K = 0) Then
        ElementIndex = 1
    Else
        ElementIndex = K
    End If
    ' Gas element variables
    FileWriterMiP.Write(Trim(Str(MiP(ElementIndex))) & ", ")
    FileWriterMiG.Write(Trim(Str(MiG(ElementIndex))) & ", ")
    FileWriterNi.Write(Trim(Str(Ni(ElementIndex))) & ", ")
    FileWriterXi.Write(Trim(Str(Xi(ElementIndex))) & ", ")
    FileWriterVi.Write(Trim(Str(Vi(ElementIndex))) & ", ")
    FileWriterPi.Write(Trim(Str(Pi(ElementIndex))) & ", ")
    FileWriterWi.Write(Trim(Str(Wi(ElementIndex))) & ", ")
    FileWriterVOLi.Write(Trim(Str(VOLi(ElementIndex))) & ", ")
    FileWriterVOLGasi.Write(Trim(Str(VOLGasi(ElementIndex))) & ", ")
    FileWriterBCoVoli.Write(Trim(Str(BCoVoli(ElementIndex))) & ", ")
    FileWriterTi.Write(Trim(Str(Ti(ElementIndex))) & ", ")
    FileWriterUi.Write(Trim(Str(Ui(ElementIndex))) & ", ")
    FileWriterRHOi.Write(Trim(Str(RHOi(ElementIndex))) & ", ")
    FileWriterGAMMAi.Write(Trim(Str(GAMMAi(ElementIndex))) & ", ")
    FileWriterSi.Write(Trim(Str(Si(ElementIndex))) & ", ")
    FileWriterdeltaQi.Write(Trim(Str(deltaQi(ElementIndex))) & ", ")
    FileWriterdeltaNi.Write(Trim(Str(deltaNi(ElementIndex))) & ", ")
    ' Propellant variables
    FileWriterBi.Write(Trim(Str(Bi(ElementIndex))) & ", ")
    FileWriterRgraini.Write(Trim(Str(Rgraini(ElementIndex))) & ", ")
    FileWriterPropVOLi.Write(Trim(Str(PropVOLi(ElementIndex))) & ", ")
    FileWriterdeltaMi.Write(Trim(Str(deltaMi(ElementIndex))) & ", ")
    FileWriterfi.Write(Trim(Str(fi(ElementIndex))) & ", ")
Next K
' Gas element variables
FileWriterMiP.WriteLine(Trim(Str(MiP(NE))))
FileWriterMiG.WriteLine(Trim(Str(MiG(NE))))
FileWriterNi.WriteLine(Trim(Str(Ni(NE))))
FileWriterXi.WriteLine(Trim(Str(Xi(NE))))
FileWriterVi.WriteLine(Trim(Str(Vi(NE))))
FileWriterPi.WriteLine(Trim(Str(Pi(NE))))
FileWriterWi.WriteLine(Trim(Str(Wi(NE))))
FileWriterVOLi.WriteLine(Trim(Str(VOLi(NE))))
FileWriterVOLGasi.WriteLine(Trim(Str(VOLGasi(NE))))
FileWriterBCoVoli.WriteLine(Trim(Str(BCoVoli(NE))))
FileWriterTi.WriteLine(Trim(Str(Ti(NE))))
FileWriterUi.WriteLine(Trim(Str(Ui(NE))))
FileWriterRHOi.WriteLine(Trim(Str(RHOi(NE))))
FileWriterGAMMAi.WriteLine(Trim(Str(GAMMAi(NE))))
FileWriterSi.WriteLine(Trim(Str(Si(NE))))
FileWriterdeltaQi.WriteLine(Trim(Str(deltaQi(NE))))
FileWriterdeltaNi.WriteLine(Trim(Str(deltaNi(NE))))
' Propellant variables
FileWriterBi.WriteLine(Trim(Str(Bi(NE))))
FileWriterRgraini.WriteLine(Trim(Str(Rgraini(NE))))
FileWriterPropVOLi.WriteLine(Trim(Str(PropVOLi(NE))))
FileWriterdeltaMi.WriteLine(Trim(Str(deltaMi(NE))))
FileWriterfi.WriteLine(Trim(Str(fi(NE))))
' Projectile variables
FileWriterShell.WriteLine(Trim(Str(Time)) & ", " & Trim(Str(Xshell)) & ", " & _
```

```
            Trim(Str(Vshell)) & ", " & Trim(Str(ACCshell)) & ", " & Trim(Str(Pshell)))
    End Sub

End Module
```