ABSTRACT

The objective of this technical report is to document mathematical and physical modelling efforts that calculate the transient behaviour of buoyant thermals. Buoyant thermals form a part of a source term for plume dispersion that can be generated by extreme events in the environment such as blasts, nuclear detonations, volcanic eruptions or bushfires. While plume dispersion is relatively well understood, the source term remains often unclear. Numerical and analytical models presented here provide a mathematical understanding and can predict the displacement and temperature evolutions of buoyant thermal cavities. Verification of the models shows an agreement with experimental data found in literature.
Modelling of Buoyant Thermals

Executive Summary

Understanding the characteristics of turbulent convective cavities generated by external energy releases in the environment is of significant importance for a number of military applications and national security problems. Examples in the military domain include an estimation of the wind dispersion of hazardous tracers, toxic industrial chemicals/materials (TICs/TIMs), nuclear fallout, and signatures of rocket launches. In the national security domain and for the community of first responders the relevant problems include chemical improvised explosive devices, large scale industrial incidents (e.g. nuclear reactor incidents), and bushfires. Increasing attention to these problems has been stimulated by recent significant disruptions to the world air-traffic by volcanic ash (leading to the establishment of a special centre within the Australian Bureau of Meteorology). An estimation of the likelihood of such events and their effects on air quality (visibility, background tracer concentrations, detection threshold, false alarms etc.) may also become important for the planning of military and rescue operations.

The dispersion of disseminated hazardous chemical, biological and radiological agents is relatively well understood, however the source term remains largely unknown. In this technical report part of the source term, in particular the process of buoyant thermals after an explosive energy release, is addressed. Mathematical and physical modelling that predicts the displacement and temperature evolutions of buoyant thermal cavities is documented. Numerical and analytical mathematical models are presented. While a numerical model allows for a more accurate solution of the transient buoyant thermal processes, the analytical model offers a quick approximation of their behaviour. Predictions made using these models were compared to the available experimental data generated by explosive energy releases. The numerical model agreed with the experimental data from the literature and the analytical model confirmed the existing scaling laws valid for weak thermal gradients reported in the literature and has extended the understanding of the transient behaviour of buoyant thermal cavities to the region of short timescales and dominant thermal gradients.
This page is intentionally blank
Authors

Martin Kocan
Land Division

Martin Kocan obtained his PhD from the University of Aachen, Germany in 2003 and subsequently worked as a research scientist at the University of Goettingen, Germany, at the University of Western Australia and at Solar Systems Pty Ltd. Martin has been working in the Land Division at DSTO since 2010.

Milan Jamriska
Land Division

Dr Jamriska is a Senior Research Scientist with 15 years of academia experience in aerosol physics. He has published more than thirty research papers in peer reviewed scientific journals and attracted research funding totalling close to one million dollars. Dr Jamriska joined DSTO in 2005 to establish and support CBRN aerosol R&D capabilities. His current work is focused on the characterisation of protective barrier fabrics and respirators against airborne CBRN threats and development of new IPE materials. Other areas of his research include T&E and R&D of aerosol sampling devices and development of new detection methods for aerosols of biological and non-biological origin.

Alex Skvortsov
Land Division

Dr Skvortsov has been working at DSTO since 2005 in the Hazard Assessment Group at Land Division. He is the Science Team Leader for Hazard Modelling. His areas of research include turbulent dispersion, CBRN source backtracking and mathematical biology. Alex’s educational background is in theoretical physics. He holds his PhD, which was devoted to the problem of sound-vortex interaction, from the Moscow University of Applied Physics and Technology. Alex has significant R&D experience in defence sponsored projects (submarine signatures, new materials, artificial intelligence) on which he worked in academia and industry. He has published more than 50
papers. Before joining DSTO Alex worked as an IT Consultant/Architect in the area of Data Mining/Decision Support Systems.

Timothy Dubois
Land Division

Timothy DuBois is a PhD candidate at RMIT University in Melbourne, Australia with the Chemical and Quantum Physics group. His current research focuses on microscopic models of strongly coupled two-level defects in Josephson junctions. Timothy has also contracted for the Defence Science and Technology Organisation since 2009 specialising in statistical analysis and optimisation, as well as research in the areas of atmospheric turbulence and blast dynamics.
## Contents

1. INTRODUCTION ............................................................................................................... 1

2. MODELLING APPROACH .............................................................................................. 2  
   2.1 Numerical model ....................................................................................................... 2  
   2.2 Analytical model ....................................................................................................... 5

3. SIMULATION RESULTS ................................................................................................ 11  
   3.1 Numerical model for buoyant thermals ............................................................. 11  
      3.1.1 Comparison with published experimental data ............................................. 11  
      3.1.2 Simulation results for a generic initial temperature profile .............. 15  
   3.2 Comparison of predicted scaling laws with experimental data ..................... 17

4. CONCLUSIONS ................................................................................................................ 21

5. REFERENCES .................................................................................................................... 21

6. ACKNOWLEDGEMENTS .............................................................................................. 22

APPENDIX A: MATLAB CODE - NUMERICAL MODEL ................................................... 23  
   A.1. Programming code instructions ........................................................................... 23  
   A.2. Programming code for buoyant thermal cavity model .............................. 25
1. Introduction

A thermal is a finite parcel of fluid consisting of the same fluid as its surroundings but at a different temperature. Because of its buoyancy, a cold thermal sinks (negative buoyancy), while a warm thermal rises (positive buoyancy). The name was given by glider pilots to what they perceived as regions of warm air rising above a heated ground in which they could soar. Convection in the atmosphere does indeed proceed by means of rising thermals [Priestley 1959]. Thermals can also be called thermal cavities or thermal balloons due to their different temperature and density compared to their surroundings. Buoyant thermal cavities, depicted in Fig. 1, have traditionally been studied in the context of geophysical fluid dynamics (atmospheric thermals, volcano eruption, hydrothermal fountains) [Batchelor 1954, Kaye 2008, Meerson 1989, Turner 1969].

Figure 1  Schematic diagram of a spherically symmetric buoyant thermal cavity where $z(t)$ is the time dependent vertical coordinate, $r(t)$ is the time dependent radial coordinate, $Z(t)$ is the vertical elevation of the buoyant thermal cavity, and $R(t)$ is the radius of the buoyant thermal cavity.

Since the pioneering work of Batchelor it has been well-recognised that buoyant thermal cavities have remarkable scaling properties. All parameters of the cavity are determined by a single quantity, the amount of energy released in the environment [Batchelor 1954, Turner 1969]. This manifests itself in the strong self-similarity evolution of thermals:

$$Z(t) \sim \left( E_0^{\alpha^*} t \right)^{\alpha^*}, \quad R(t) \sim \left( E_0^{\beta^*} t \right)^{\beta^*}, \quad \lambda(t) = \frac{Z(t)}{R(t)} \sim t^{\alpha^*-\beta^*},$$

where $Z(t)$ is the elevation distance of the thermal cavity, $R(t)$ is its radius, $E_0$ is approximately equal to the total amount of the energy released into the environment, $t$ is the time and $\alpha^*$ and $\beta^*$ are power law coefficients. For weak thermal gradients and larger time scales $\alpha^*=\beta^*=1/2$ and $\lambda(t)$ becomes a constant independent of time, $t$, and energy, $E_0$.

The scaling law shown in Equation 1 is supported by significant experimental evidence (see e.g. [Yaar 2008]). Often scaling is used to provide a simple empirical data fit. A great variety of models have been proposed to generalise this simple self-similar model.
They include various physical processes associated with the rapid energy release (chemical kinetics, radiation, condensation etc.), different atmospheric conditions (stratification, stability, background turbulence) and different flow regimes (turbulent, laminar, buoyancy dominated). Many of these models provide reliable agreements with experimental observations. Unfortunately, in spite of their relative simplicity (system of coupled ordinary differential equations (ODEs)) these models often allow only numerical treatment. Such complexity makes it difficult to formulate results in terms of simple scaling laws and to capture the entire complexity of thermal cavity evolution with the two scaling parameters $\alpha^*$ and $\beta^*$.

A blast generated thermal cavity has a strong buoyancy effect due to high temperatures inside the cavity that can reach 2500$-$3000 °C. This often violates the initial assumptions of underlying scaling (low deviation from ambient density, low thermal gradients). For more powerful energy releases the scaling laws need to be refined especially at short time scales after the explosion. The scaling laws presented in Equation 1 become an asymptote at large time scales for thermals generated by an explosion.

Events such as blasts leave behind a strongly heated and rarefied cavity in the atmospheric gas after the generation and rapid expansion of the gaseous detonation products. This is followed by further expansion of the thermal cavity coupled with a buoyancy-driven cavity. Hence, the thermal cavity is often called a thermal balloon. In the next stage, the cavity continues to rise, due to buoyancy and inertia and is accompanied by cooling until the temperature within the cavity approaches external ambient conditions. Finally, the cavity expands horizontally and vertically as a result of diffusion and turbulence caused by micro-meteorological conditions [Cao 2011]. Compared to the blast process, which occurs on a time scale of milliseconds, the thermal cavity related processes take place on the scale of seconds.

In the current study we present: (i) a numerical model and (ii) a simple analytical and physically consistent model of a buoyant thermal cavity generated by explosive energy releases. Both models lead to predications that were validated with available experimental data.

### 2. Modelling Approach

#### 2.1 Numerical model

The following section describes numerical modelling of the physical processes of buoyant thermals in a three-dimensional spherically symmetric system following a blast. The model includes heat conduction (thermodynamics) and buoyant thermal cavity motion (Buoyant-Lagrangian model).

The time, $t$, and space, $r$, dependent temperature of the gas, $T(r,t)$, can be modelled by solving the following nonlinear partial differential equation (PDE) [Meerson 1989]:

\[
\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial r^2} + \beta T
\]
\[ \frac{\partial T(r,t)}{\partial t} = \frac{T^2(r,t)}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\beta(T)}{T(r,t)} \frac{\partial T(r,t)}{\partial r} \right), \] (2)

where \( r \) is the radial coordinate of a spherically symmetric system and \( \beta(T) \) is the normalised thermal diffusivity of air that can be expressed as

\[ \beta(T) = \frac{k(T)}{c_p(T)\rho_o T_o}, \] (3)

where \( k(T) \) is the temperature dependent heat conductivity, \( c_p(T) \) is the temperature dependent specific heat capacity at constant pressure, \( \rho_o \) is the density of ambient air and \( T_o \) is the temperature of ambient air. If the normalised thermal diffusivity of air, \( \beta(T) \), is written as a power law function of the temperature, i.e. \( \beta(T) = \beta_o T^n \), then the Equation 2 becomes

\[ \frac{dT(r,t)}{dt} = \beta_o \frac{T^2(r,t)}{r^2} \frac{\partial}{\partial r} \left( r^2 T^{n-1}(r,t) \frac{\partial T(r,t)}{\partial r} \right). \] (4)

The exponent \( n \) is equal to 1/2 for the molecular heat conduction of an ideal gas [Heiser 1994]. The partial differential Equation 4 requires one initial and two boundary conditions. The boundary condition at the centre of the thermal cavity, \( r=0 \), follows from spherical symmetry:

\[ \left. \frac{dT}{dt} \right|_{r=0} = 0. \] (5)

At the boundary far away from the centre of the buoyant thermal cavity, \( r=R_b \), the temperature must be equal to the ambient temperature

\[ T\big|_{r=R_b} = T_o, \] (6)

where \( R_b \) is the right boundary of the radial coordinate. The initial temperature profile for the thermal cavity process can be chosen as the final state of the blast process, i.e. at the time when the blast pressure and the blast velocity are equal to the ambient conditions. An example of a temperature profile at these conditions is shown in Figure 2. This temperature profile can be generated by running a gas dynamics computational solver developed at DSTO [Antanovski 2008]. Alternatively, the initial temperature profile for the thermal cavity can generally be described by a power law, \( T(r,t=0)=Ar^k \), where \( A \) and \( k \) are constants [Meerson 1989].
Figure 2  Temperature profile at 100 ms after the detonation of a spherical charge of 0.5 kg of TNT. The dashed line is an indication of a power law.

The density profiles of the thermal cavity can be calculated using an expression for an ideal gas:

$$\rho(r,t)T(r,t) = \rho_oT_o = \frac{pm}{N_M k_B} = \text{const},$$  \hspace{1cm} (7)

where $\rho(r,t)$ is the time, $t$, and radial coordinate, $r$, dependent density of the thermal cavity, $p$ is the pressure, $m$ is the mass of the gas, $N_M$ is the number of molecules in the gas and $k_B$ is the Boltzmann constant.

The radius of the thermal cavity can be determined from the calculated space dependent temperature profiles, where the temperature of the thermal balloon is equal to

$$T(r = R,t) = \left(1 + \frac{FMK}{100}\right) T_o.$$  \hspace{1cm} (8)

$FMK$ is a multiplication factor that determines the deviation from the ambient temperature. A very low value of $FMK$ ($FMK<<1\%$) cannot be used due to the second boundary condition defined in Equation 6 and the initial temperature profile with temperature approaching the ambient temperature at the right boundary of the radial coordinate.

---

1 The temperature profile was generated by using a gas dynamics computational solver developed by L.K. Antanovskii [Antanovskii 2008].
The thermal cavity motion is given by the balance between the buoyancy and the drag force [Werne 1994]:

\[
\frac{dv(t)}{dt} = g \left( \rho_v(t) - \bar{\rho}(t) \right) - \frac{C_D}{2} \rho_v \frac{A_B}{\bar{m}(t)} v^2(t),
\]

where \( v(t) \) is the upward thermal velocity of the buoyant thermal cavity, \( g \) is the gravitational acceleration, \( \bar{\rho}(t) \) is the average density of the thermal cavity at the time \( t \), \( \bar{m}(t) \) is the average mass of the thermal cavity at the time \( t \), \( A_B \) is the cross sectional area of the thermal cavity (on a plane perpendicular to the direction of motion) and \( C_D \) is the drag coefficient. The calculation of the drag coefficient can be found in reference [Kocan 2012]. The first and second terms on the right-hand-side of Equation 9 are the buoyancy and drag terms, respectively.

Particles that are present in the thermal cavity after the blast may be displaced and may change its temperature. The associated mathematical models for particle dispersion and temperature can be found in reference [Kocan 2012].

The numerical predictions of a buoyant thermal cavity were performed using a computer simulation in the Matlab® programming environment based on solving the equations described above. The differential equations for temperature and velocity of the buoyant thermal cavity were solved using the Matlab internal PDE and ODE solver. The Matlab programming code is attached in Appendix A.

2.2 Analytical model

In this section we introduce a physics-based model for a buoyant thermal cavity’ size and vertical position that would allow for a simple analytical treatment.

The following equations describe conservation laws of volume, momentum and heat (buoyancy) [Morton 1956]:

(a) \( \frac{dV(t)}{dt} = S(t)v_e(t) \),

(b) \( \frac{d[V(t)v(t)]}{dt} = gV(t)[\rho_0 - \rho(t)] \),

(c) \( \frac{d}{dt} \left[ gV(t) \frac{\rho_0 - \rho(t)}{\rho_{ref}} \right] = -V(t)v(t)N \),

where \( V(t) \) is the volume of the buoyant thermal cavity, \( S(t) \) is its surface area, \( v_e(t) \) is the gas entrainment velocity into the buoyant thermal cavity and \( v(t) \) is the upward thermal velocity of the thermal cavity given by \( dZ/dt \), \( \rho_{ref} \) is the reference density and \( N \) is the stratification parameter of the environmental fluid corresponding to [Morton 1956]:

UNCLASSIFIED
The effect of meteorological conditions can be neglected if the maximum elevation of the thermal cavity is lower than \( Z_{\text{MAX}}(t) \sim \alpha^{\gamma/2} F_0^{\gamma/4} N^{-\gamma/4} \), where \( \alpha \) is an experimentally fitted parameter \((\alpha=0.1-0.25)\) and \( \pi F_0 = \pi g \frac{\rho_0 - \rho(t)}{\rho_{\text{ref}}} v(t) R^2(t) \) is the source buoyancy flux [Carazzo 2008, Mehaddi 2013, Cushman-Roisin 2014].

Assuming in the first instance a uniform environment where \( d\rho/dz=0 \) and thus \( N=0 \), then the Equation 10c becomes:

\[
\frac{d}{dt} \left[ gV(t) \frac{\rho_0 - \rho(t)}{\rho_{\text{ref}}} \right] = 0 .
\]  

(12)

Solving the above equation we get:

\[
gV(t) \frac{\rho_0 - \rho(t)}{\rho_{\text{ref}}} = \gamma E_o = \text{const} ,
\]  

(13)

where the normalised density difference between the buoyant thermal cavity and the ambient air multiplied by the volume of the buoyant thermal cavity is proportional to the total amount of the energy released into the environment. \( \gamma \) is a constant with units of \([\text{m}^4/(\text{s}^2\text{J})]\).

Inserting the solution of Equation 12 into the Equation 10b we get:

\[
\frac{d}{dt} [\rho(t) V(t) v(t)] = \rho_{\text{ref}} \gamma E_o = \text{const} .
\]  

(14)

Hence, the thermal cavity momentum \([\rho(t) V(t) v(t)]\) becomes a linear function of time, \( t \):

\[
\rho(t) V(t) v(t) = \rho_{\text{ref}} \gamma E_o t .
\]  

(15)

Equation 10a allows to extract the transient behaviour of the volume of the thermal cavity. As proposed in earlier studies the entrainment velocity, \( v_e \), shown on the right-hand-side of this equation, can be expressed as being proportional to the elevation velocity and heat flux [Batchelor 1954, Meerson 1989]:

\[
v_e(t) = \alpha v(t) + \beta(T) \frac{\partial^2 T}{\partial r^2} .
\]  

(16)

The first term on the right-hand-side of Equation 16 was proposed by Batchelor and is valid for low gradients of temperature and high elevation velocities [Batchelor 1954]. The second term in the equation is valid when the thermal cavity is close to rest and when the
thermal gradients are dominant [Meerson 1989]. The second term on the right-hand-side of the Equation 16 can be approximated as
\[ \beta(T) \frac{\partial T}{\partial r} \approx \frac{\beta(T)}{\rho_0 R(t)} \frac{b(t) T(t)}{\rho(t)} , \]  
(17)
by replacing \( \partial T/\partial r \) with \([T(t) - T_0]/R(t) \) and by expressing \([T(t) - T_0] \) as \( \frac{\rho_{ref}}{\rho_0} b(t) T(t) \), where \( b(t) \) is the buoyant perturbation given by [Morton 1956]:
\[ b(t) = \frac{\rho_0 - \rho(t)}{\rho_{ref}} . \]  
(18)
Note that the buoyant perturbation is non-zero whenever the density in the buoyant thermal cavity is different from the ambient density. By applying the ideal gas law shown in Equation 7, the buoyant perturbation can be also expressed as:
\[ b(t) = \frac{\rho_0}{\rho_{ref}} \frac{T(t) - T_0}{T(t)} . \]  
(19)
Applying Equations 17 and 19 to Equation 16 we can express the entrainment velocity as:
\[ v_e(t) = \alpha v(t) + \frac{\beta(T) \rho_{ref} b(t) T(t)}{\rho_0 R(t)} . \]  
(20)
Furthermore, expressing the following terms on the right-hand-side of the above equation: the elevation velocity, \( v(t) \), and the buoyant perturbation, \( b(t) \), using the solution of the equation for the momentum conservation (Equation 15) and for the heat conservation (Equation 13), respectively, we can write the entrainment velocity as:
\[ v_e(t) = \alpha \frac{\rho_{ref} \gamma E_0}{\rho(t) \gamma V(t)} + \frac{\beta(T) \rho_{ref} \gamma E_0}{\rho_0 g} \frac{T(t)}{R(t) \gamma V(t)} . \]  
(21)
Applying the above expression for the entrainment velocity into the equation of volume conservation (Equation 10a) we get:
\[ \frac{dV(t)}{dt} = S(t) v_e(t) = S(t) \left[ \alpha \frac{\rho_{ref} \gamma E_0}{\rho(t) \gamma V(t)} + \frac{\beta(T) \rho_{ref} \gamma E_0}{\rho_0 g} \frac{T(t)}{R(t) \gamma V(t)} \right] . \]  
(22)
Similarly as stated above for Equation 16, the first term on the right-hand-side of Equation 22 is valid at low thermal gradients and high elevation velocities and the second term in the equation is valid when the thermal gradients dominate and the buoyant thermal cavity is close to rest.

**Region of short time scales and strong thermal gradients**
Let us now assume a region of short time scales and strong thermal gradients only, then the first term on the right-hand-side of Equation 22 can be neglected. Expressing the radius
of the buoyant thermal cavity in terms of its volume (assuming a spherical buoyant thermal cavity), the second term on the right-hand-side of Equation 22 then becomes:

\[
\frac{dV(t)}{dt} = 3^{\frac{3}{2}} (4\pi)^{\frac{3}{2}} \beta(T) \frac{\rho_{\text{ref}}}{\rho_0} \frac{\gamma E_0}{g} \frac{T(t)}{V(t)^{\frac{3}{2}}}.
\]

The analytical solution of the above differential equation can be written as:

\[
V(t) = \left(\frac{5}{3} \xi E_0 t\right)^{\frac{3}{5}},
\]

where \(\xi = 3^{\frac{3}{5}} (4\pi)^{\frac{3}{2}} \beta(T_i) T_i \rho_{\text{ref}} / (\rho_0 g)\) is a constant assuming that at short time scales \((t \rightarrow t_i)\), the temperature of the thermal buoyant cavity is approximately equal to an initial temperature, \(T(t=ti) \sim T_i\); and the thermal diffusivity is approximately equal to thermal diffusivity at the initial temperature, \(\beta(T) \sim \beta(T_i)\). This is correct at times close to \(t_i\) when the higher order terms in the Taylor expansion series of the buoyant thermal cavity temperature and thermal diffusivity can be neglected, i.e. for example, in the case of temperature we could approximate it as:

\[
T(t = t_i) = T(t_i) + \frac{1}{1!} \frac{dT(t)}{dt}(t-t_i) + \frac{1}{2!} \frac{d^2T(t)}{dt^2}(t-t_i)^2 + \ldots \approx T(t_i).
\]

Similarly, it would be also the case for the thermal diffusivity, \(\beta(T=Ti)\).

Assuming a spherical buoyant thermal cavity, its radius can be determined by

\[
R(t) = \left(\frac{3}{4\pi} V(t)\right)^{\frac{1}{3}} = \frac{3^{\frac{2}{3}}}{(4\pi)^{\frac{1}{3}}} \left(\frac{5}{3} \xi E_0 t\right)^{\frac{1}{3}},
\]

where the volume of the buoyant thermal cavity is taken from Equation 24.

The elevation of the buoyant thermal cavity can be sourced from the definition of the upward thermal velocity:

\[
\frac{dZ(t)}{dt} = v(t) = \frac{\rho_{\text{ref}}}{\rho(t)} \frac{\gamma E_0}{V(t)} t = \left(\frac{3}{5}\right)^{\frac{3}{5}} \frac{\rho_{\text{ref}}}{\rho(t)} \frac{\gamma}{\xi} (E_0 t)^{\frac{3}{5}},
\]

where the upward thermal velocity, \(v(t)\), is expressed from the thermal cavity momentum (Equation 15) and the cavity volume, \(V(t)\), is given in Equation 24. Note that at short time scales when \(t \rightarrow t_i\) the density of the buoyant thermal cavity in Equation 27 can be approximated to be constant by neglecting higher order terms in the Taylor expansion series: \(\rho(t) \rightarrow \rho(t_i)\).

The solution of the first-order ordinary differential equation shown in Equation 27 is:
From Equations 26 and 28 we can derive new scaling laws valid at small time scales and strong thermal gradients:

\[ Z(t) \sim E_0^{7/3} t^{7/3}, \quad R(t) \sim (E_0 t)^{1/2}, \quad \lambda(t) = \frac{Z(t)}{R(t)} \sim E_0^{1/3} t^{1/3}. \] 

(29)

Scaling laws presented in the previous equation and in Equation 1 imply that there is a transition point where the buoyant thermal cavity becomes weaker. The following helps to graphically visualise two distinct regions (weak and strong thermal gradients) where the radius of the thermal cavity allows simple analytical treatment through scaling laws. If we describe the volume of the thermal cavity shown in Equation 22 as a function of the radius, \( R(t) \), assuming a spherical buoyant thermal cavity, we arrive to the following first-order ordinary differential equation:

\[ \frac{dR(t)}{dt} = \frac{\gamma E_0}{3R^3(t)} \left[ c_1 t + \frac{c_2}{R(t)} \right]. \]

(30)

A graphical interpretation of the numerical solution to Equation 30 (solved by applying a Runge-Kutta iterative method) for some specific values of parameters \( c_1 = 3, c_2 = 100 \text{ ms}, \gamma E_0 = 1000 \text{ m}^2/\text{s}^2, R(t \rightarrow 0) = 10 \text{ cm} \) is shown in Figure 3 (blue solid line). The two dashed lines represent regions of a power law dependence of the buoyant thermal cavity radius on time. The red dashed line indicates a region of low temperature gradients for buoyant thermals (Batchelor regime) and the green dashed line shows the case of strong thermal gradients for buoyant thermals.
At larger time scales, the first term in Equation 22 dominates and leads to Batchelor scaling (not derived here). On the other hand, at initial time when the volume of a buoyant thermal cavity is small the second term in Equation 22 dominates.

The following table summarises the scaling laws for buoyant thermals.

**Table 1  Scaling laws for buoyant thermals**

<table>
<thead>
<tr>
<th>Region</th>
<th>Strong thermals</th>
<th>Weak thermals</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha^<em>$ in $Z(t)\sim t^{\alpha^</em>}$</td>
<td>2.00³</td>
<td>7/5=1.40</td>
</tr>
<tr>
<td>$\beta^<em>$ in $R(t)\sim t^{\beta^</em>}$</td>
<td>4/53=0.08</td>
<td>1/5=0.20</td>
</tr>
<tr>
<td>$\alpha^<em>\beta^</em>$ in $Z(t)/R(t)\sim t^{\alpha^<em>\beta^</em>}$</td>
<td>102/53=1.92</td>
<td>6/5=1.20</td>
</tr>
</tbody>
</table>

² Assuming an initial temperature profile $T(r,t=0)=Ar^{-k}$ where $A$ is a temperature parameter and $k=7.5$. If $k=3$ the scaling law for the radius of the buoyant thermal cavity predicted by the numerical model of Meerson will become the same as for the analytical model (not derived here) [Meerson 1989].

³ Power of two follows from pure accelerated motion of the buoyant thermal cavity when buoyancy exceeds the drag force.
3. Simulation Results

3.1 Numerical model for buoyant thermals

3.1.1 Comparison with published experimental data

The DSTO developed programming code, documented in Appendix A, was tested against experimental blast measurements published in literature [Yaar 2008]. The calculated data were compared to the published results and an agreement was found.

Figure 4 shows a comparison of the top cloud height between the modelled and the experimental data from Yaar for 0.5 kg and 2 kg of TNT [Yaar 2008]. Using $\beta_o = 8 \cdot 10^{-5} \, m^2/(sK^{3/2})$ and $\nu=0.5$ for the thermal diffusivity of the buoyant thermal cavity ($\beta(t) = \beta_o T^\nu$) and a parameter $FMK=1\%$ in the numerical simulation, a good agreement between the modelled and experimental data was achieved. $\beta_o$, $\nu$ in the Equation 4 and $FMK$ in Equation 8 were the only fitting parameters in the model. The exponent $\nu$ was kept constant at $\nu=0.5$ valid for the molecular heat conduction of an ideal gas. The fitted value for $\beta_o$ of $8.10^{-5} \, m^2/(sK^{3/2})$ reflects the thermal properties of the thermal cavity in the experiment. Note that the chosen $FMK$ parameter has a significant influence on the calculated radius of the spherical thermal cavity.

![Graph showing comparison between modelled and experimental data for different TNT blast masses.](image-url)

Figure 4  Buoyant thermal cavity top cloud height for blast of 0.5 kg and 2.0 kg TNT explosives. The modelled data match well the observed experimental data [Yaar 2008].
Figure 5 shows the calculated upward thermal velocity of the buoyant thermal cavity for 0.5 kg and 2 kg of TNT. No experimental data of upward thermal velocity were available for comparison with the model from Yaar et al. [Yaar 2008]. The slope of the reported top cloud height versus time could be an indicator of an estimated velocity but unfortunately, the calculated slope provides strongly scattered data and thus could not be used for comparison with the model.

\[ \beta(t) = 8 \times 10^{-5} t^{1.2} \]

Figure 5  Calculated upward thermal velocity of the buoyant thermal cavity for blasts of 0.5 kg and 2 kg TNT explosives.
Figure 6 shows scaling laws for modelled buoyant thermal cavity elevation $Z(t)\sim t^{\alpha^*}$. For larger time scales ($t>1$ s) the model did not reach $Z(t)\sim t^{1/2}$ [Batchelor 1954] but showed lower $\alpha^*$ coefficients due to neglected first term on the right-hand-side of Equation 16 in the numerical model. For lower time scales the buoyant thermal cavity elevation was proportional to $t^2$ as predicted by pure accelerated motion when buoyancy exceeds the drag force.

![Figure 6](image_url)  
**Figure 6** Extracted power law exponent for time dependent elevation of the buoyant thermal cavity, $Z(t)\sim t^{\alpha^*}$.

Figure 7 shows time dependent thermal cavity over radius, $\lambda(t)$, and its associated extracted power law exponent, $\alpha^*-\beta^*$. $\lambda(t)$ shows that this quantity is increasing with time and saturates at a certain value depending on the strength of the source. At small time scales, $\lambda(t)$ is proportional to $t^{102/53}$ and at larger time scales $\lambda(t)$ is proportional to a constant which agrees with Meerson [Meerson 1989] and Batchelor [Batchelor 1954] predictions, respectively.

![Figure 7](image_url)
Figure 7  (A) Time dependent elevation over radius of the buoyant thermal cavity, $\lambda(t)\sim t^{\alpha^* - \beta^*}$, and (B) its associated extracted power law exponent, $\alpha^* - \beta^*$. 
3.1.2 Simulation results for a generic initial temperature profile

This section shows scaling laws calculated by solving the numerical model for an initial temperature profile defined as a power law $T(r,0)=Ar^{-k}$, where $A$ and $k$ are positive constants [Meerson 1989]. This power law describes a decrease of the initial temperature with the distance from the centre of the buoyant thermal cavity. The equation is invalid at the periphery of the thermal cavity where the temperature must approach the ambient temperature. Furthermore, at the centre ($r=0$) of the buoyant thermal cavity there is a singularity which disappears away from the centre ($r>0$).

Figure 8 and 9 present an elevation of the buoyant thermal cavity and calculated scaling laws for an initial temperature profile, respectively. The following constants were used for the initial temperature profile $A=10^{-3}$ Km$^3$ and $k=3$. The remaining parameters were defined as: $\beta_0 = 8 \times 10^{-5}$ m$^2$/(sK$^{3/2}$), $\nu = 0.5$ [$\beta(T) = \beta_0 T^\nu$] and $FMK=1\%$.

A scaling law between 1 and 2 for the elevation was observed at smaller time scales as predicted by the numerical and analytical models. Please note that a scaling of 1 corresponds to a velocity of the buoyant cavity being constant, i.e. when the buoyant driving force is equal to the resistance drag force. The scaling of 2 indicates pure accelerated motion of the buoyant thermal cavity when buoyancy exceeds the drag force. The scaling of $1/2$ as predicted by Batchelor was not recovered due to neglecting the first term on the right-hand-side of Equation 16 in our numerical model described in Section 2.2 [Batchelor 1954, Meerson 1989]. The scaling for the radius (exponent $\beta^*$) matched the value as predicted by the simplified analytical model.

![Figure 8](image-url)  
**Figure 8** Calculated elevation of the buoyant thermal cavity for an initial temperature profile defined by $T(r,0)=Ar^{-k}$, where $A=10^{-3}$ Km$^3$ and $k=3$. 

UNCLASSIFIED
Figure 9 Extracted power law exponent for time dependent elevation, $Z(t) \sim t^{\alpha^*}$, and radius, $R(t) \sim t^{\beta^*}$, of the buoyant thermal cavity for an initial temperature profile defined as $T(r,t=0) = Ar^{-k}$, where $A=10^{-3}$ Km$^3$ and $k=3$. Asymptotes indicate the predicted scaling laws. The position of asymptotes in relation to the time scale is indicative only.
3.2 Comparison of predicted scaling laws with experimental data

This section shows verification of predicted scaling laws by numerical and analytical models with various experimental data available from the literature.

![Graph showing experimental data and scaling laws](image)

**Figure 10** Extracted power law exponent for time dependent elevation of the buoyant thermal cavity, $Z(t)\sim t^{\alpha^*}$ [Yaar 2008]. Asymptotes indicate the predicted scaling laws. The position of asymptotes in relation to the time scale is indicative only.

Figure 10 displays the extracted power law exponent for experimental data of time dependent elevation of a buoyant thermal, $Z(t)\sim t^{\alpha^*}$, collected by Yaar et.al. [Yaar 2008]. The power law exponent matched with the Batchelor prediction of $\alpha^*=1/2$ for weak thermal gradients [Batchelor 1954]. The data did not agree with the scaling valid for strong thermal gradients where $\alpha^*$ was expected at 2 or 7/5 as predicted by pure accelerated motion or the analytical model, respectively. The reason may be unavailability of data that cover short time scales of strong thermal gradients.

Similar scaling was observed for experimental data published by Dobranich for the elevation of the buoyant thermal cavity as shown in Figure 11 [Dobranich 1997]. In the case of the radius of the buoyant thermal cavity, at larger time scales, the radius followed the scaling law of $\beta^*=1/2$ for weak thermal gradients as predicted by Batchelor [Batchelor 1954] and at lower time scales, the radius approached low scaling of 4/53 as predicted by Meerson et.al. [Meerson 1989].

Experimental data collected by DSTO [Jamriska 2010], pictured in Figure 12, show scaling of $\alpha^*$ and $\beta^*$ to be scattered around 1/2 corresponding to weak thermal gradients as predicted by Batchelor [Batchelor 1954]. The published data did not seem to cover shorter time scales which would reach asymptotes of $\alpha^*=2$ or 7/5 (as predicted by pure accelerated...
motion or by analytical model, respectively) and of $\beta^* = 4/53$ or $1/5$ (as predicted by the numerical or analytical model, respectively).

Figures 13 to 15 show experimental data that approach the scaling law determined by the analytical model $[Z(t) \sim t^{7/5}]$ for strong thermal gradients at short time scales. Possible reason that the agreement was not better may be the lack of availability of experimental data for the early stage of the buoyant process.

![Experimental data and scaling laws](image-url)

*Figure 11 Extracted power law exponent for time dependent elevation, $Z(t) \sim t^{\alpha^*}$, and radius, $R(t) \sim t^{\beta^*}$, of the buoyant thermal cavity [Dobranich 1997]. Asymptotes indicate the predicted scaling laws. The position of asymptotes in relation to the time scale is indicative only.*
Figure 12 Extracted power law exponent for time dependent elevation, $Z(t)\sim t^\alpha$, and radius, $R(t)\sim t^\beta$, of the thermal cavity [Jamriska 2010]. Asymptotes indicate the predicted scaling laws. The position of asymptotes in relation to the time scale is indicative only.

Figure 13 Extracted power law exponent for time dependent elevation, $Z(t)\sim t^\alpha$, and radius, $R(t)\sim t^\beta$, of the thermal cavity [Jodoin 1994]. Asymptotes indicate the predicted scaling laws. The position of asymptotes in relation to the time scale is indicative only.
Figure 14 Extracted power law exponent for time dependent elevation, $Z(t) \sim t^\alpha$, and horizontal radius, $R(t) \sim t^\beta$, of the thermal cavity [Sandman 2005]. Asymptotes indicate the predicted scaling laws. The position of asymptotes in relation to the time scale is indicative only.

Figure 15 Extracted power law exponent for time dependent elevation of the buoyant thermal cavity, $Z(t) \sim t^\alpha$, [Basket 1991]. Asymptotes indicate the predicted scaling laws. The position of asymptotes in relation to the time scale is indicative only.
4. Conclusions

Mathematical and physical modelling that predicts the displacement of buoyant thermal cavities which form a part of a source term for plume dispersion is reported. An improved mathematical understanding of the transient evolution of buoyant thermals after explosive energy releases has been achieved. Numerical and analytical models are documented. A comparison of the numerical model with the published experimental data revealed a good agreement between the simulation and reported experimental results. The only fitting parameters used were the thermal parameters describing buoyant thermals and a constant used to determine the radius of the thermal cavity. The Matlab programming code for the numerical model is published here. In addition to weak thermal gradients (predicted by Batchelor), a simplified analytical model, presented here, describes scaling laws for a region when thermal gradients dominate, usually at an early stage in the buoyant process. Experimental data are shown to match the predicted scaling laws for low thermal gradients and for large thermal gradients the data approach the predicted values.

5. References

Jamriska M., Muir B., Muir W., Teo A. and Piggott E. (2010), DSTO-CR-2010-0064


6. Acknowledgements

This work has been conducted as a part of Task 07/285 (Support to DIO-HPPD). Support, assistance and guidance received from the DIO client throughout the project are being acknowledged.

The authors wish to thank Leonid Antanovskii (WCMD at DSTO) and Michael Roberts (LD at DSTO) for their valuable feedback, comments and fruitful discussions.
Appendix A: Matlab code – Numerical model

A.1. Programming code instructions

The programming code shown in the next section was tested in Matlab version 7.10.0.499 (R2010a) 32-bit (win32). The following bullets points show a top level instructions manual.

- Copy all m files into one working directory.
- Change parameters in the file “InputParameters.m” if required.
- Run the file “mainFireball.m” in Matlab.
- Choose the initial temperature profile from the following three options through the GUI menu (Fig. A1):
  - “Load from MAT file” – this option loads initial gas state data for the buoyant thermal cavity simulation from a gas dynamics simulation of a blast.
    - This option requires to copy the MAT file into a subdirectory “\input” and to change the MAT file name in the file “InputParameters.m”, accordingly.
    - Please note that the MAT file can be generated using the mathematical solver reported in [Kocan 2012]. The Matlab MAT file includes the temperature, pressure and density profiles after a blast as well as all parameters for the buoyant thermal cavity numerical model. ‘After a blast’ means at the time when the air velocity and the air pressure are equal to ambient conditions.
  - “Use generic T(r)=T_0+A.r^-k” – this option uses generic initial gas state conditions.
    - The initial temperature profile is defined by $T(r)=T_0+A.r^k$, where $A$ and $k$ are positive constants, $T$ is the thermal cavity temperature in Kelvin, $T_0$ is the ambient temperature and $r$ is the radial coordinate in metres [Meerson 1989].
  - “Cancel” – alternatively, an exit option is available.
- Four plot windows are generated (Fig. A2): (1.) a plot of an initial temperature profile for the buoyant thermal cavity model, (2.) plots of temperature and density profiles and plots of the time dependent mean temperature and mean density of the buoyant thermal cavity, (3.) a plot of velocity, elevation, radius and mass of a buoyant thermal cavity and (4.) a plot of a extracted power law exponent for time dependent elevation and radius of the thermal cavity.
Figure A1  GUI menu options for the numerical model of the buoyant thermal cavity.

Figure A2  Example of generated figures by the numerical modelling.
A.2. Programming code for buoyant thermal cavity model

`mainFireball.m`

```matlab
function [] = mainFireball % (InputFileName,beta0,ni)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Model for buoyant thermal cavity rise after blast
% The model calculates temperature profiles, density,
% position and velocity and estimates the cavity radius during
% the rising process

% Validation: the model was validated with experimental data published
% scenario”, S.Apikyan et al. (eds.) “Prevention, detection and
% response to nuclear and radiological threats”, Springer, pp.193-205
% author: Martin Kocan
% Defence Science and Technology Organisation (DSTO)
% Land Division (LD)
% 506 Lorimer Street, Fishermens Bend VIC 3207
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;

% global variables
global x;
global R;
global p0; global r0;
global InitialFireballTemperature;
global FireballRadialCoordinate;
global FMK;
global g;
global FireballRadialCoordinate; % required for ParticleTemperatureODE
global gamma;
global beta0; global ni;
global choice;

disp('*** Buoyant thermal cavity model ***');
disp('Choose an option from the GUI menu:');
choice = menu('Initial temperature profile:','Load from MAT file','<html>Use generic T(r)=T<sub>0</sub>+A.r^{-k}</html>','Cancel');
pause(0.1);

switch choice
    case 1
        disp('Initial conditions based on gas state after a blast.');
        % load all parameters
        InputParameters;
    case 2
        disp('Initial conditions based on generic temperature profile
T(r)=T<sub>0</sub>+A.r^{-k}');
        % load all parameters
        InputParameters;
    case 3
        return;
end
```

UNCLASSIFIED
end

% plot initial fireball temperature profile
figure; semilogy(x,InitialFireballTemperature,'.','...'
'MarkerEdgeColor','r','MarkerFaceColor','r');
title('Initial temperature profile for thermal cavity modelling');
xlabel('Radial coordinate (m)'); ylabel('Temperature just after the blast (K)');
pause(0.5);

% time step optimisation for temperature PDE - while loop
% with a condition that three closest points Temp(t) lie on a linear
% within a defined tolerance (eg.10%) in FireballTempTimeStepCheck.m
FireballTimeStepCheck = zeros(Nt,1); % time vector monitoring if temp.
% is linear across three time points
FireballTimeStepCheck(1:3)=1; % initialisation
MF = 2; % multiplication factor for increasing the number of time points
CounterWhileLoop = 0; % counting number of while loops

while sum(FireballTimeStepCheck) > 2 % Temp(t) must be non-linear at
least on 2 time points
    if CounterWhileLoop>0
        hmm=sprintf('%g closest points of mean fireball temperature are
not linear with time within the tolerance. Increasing number of points by
%g.',sum(FireballTimeStepCheck),MF);
        disp(hmm);
    end
    % distribution of time step via exponential function:
    % many time steps at small time scales
    % and few steps at larger time scales
    % thermal equation requires many points
    % at the start of the time interval
    if tstart == 0  tstart = 10^(-5); end
    % time span for fireball temperature PDE
    t.fireball=expspace(reallog(tstart),reallog(tstop),Nt)';

    % fireball solution
    tic; disp('Solving temperature equation...');
    fireball.temperature = FireballTempPDEMeerson(FireballRadialCoordinate,t.fireball); % vector
    (time,space)
    toc;
    tic; disp('Solving density, radius, mean density and mean
temperature...');
    fireball.density = FireballDensity(fireball); % vector(time,space)
    fireball.radius = FireballRadius(fireball); % vector(time)
    fireball.densityMean = FireballDensityMean(fireball); % vector(time);
    average density in fireball
    fireball.temperatureMean = FireballTempMean(fireball); %
    vector(time); average temperature in fireball
toc;

    % check if any three closest points in fireball.temperatureMean
    % are linear with time within a chosen tolerance
    % if not -> reduce the selected time step
    tic; disp('Checking time step...');
FireballTimeStepCheck =
FireballTempTimeStepCheck(fireball.temperatureMean,t.fireball,10);
toc;
Nt = round(Nt*MF); % increase number of time points
CounterWhileLoop = CounterWhileLoop+1;
end
fireball.massMean = fireball.densityMean.* (4/3 * pi * fireball.radius.^3); % vector(time); average mass in fireball
t.fireball = t.fireball(1:length(fireball.radius)); % fireball.radius determines the length of the time interval
[t.ODEfireball FireballVelocityAndPosition] = FireballVelocityAndPositionODE(fireball,t);
fireball.velocity = FireballVelocityAndPosition(:,2); % vector(time); average velocity of the fireball
fireball.position = FireballVelocityAndPosition(:,1); % vector(time); average position of the fireball

if length(fireball.radius) == length(t.fireball)
    disp('Time interval appears small. Increase variable "tstop" in file 
"Inputparameters.m".');
end

% A solution profile can also be illuminating.
FireballPlot(fireball,t);
% Input parameters
% 1. InputFileName is the mat file originating from Rotationally Symmetric Blast Model with Particles [Kocan M. et al. (2012) "Particle Motion in a Blast wave" DSTO TR-2764] (run main3Dsym.m) mat file must provide field density and pressure after blast at following conditions:
% field velocity is zero
% field pressure is equal to ambient pressure
% 2. beta0 & ni originate from thermal equation
% \[ \beta(T) = \frac{k(T)}{c(T)/r_0/T_0} = \beta_0 T^{\text{ni}} \] (power law temp dependency)
% where \( k \) [J/s/m/K] is the thermal conductivity of air
% \( c \) [J/K/kg] is the specific heat capacity
% \( r_0 \) [kg/m3] is the density of the ambient air
% \( T_0 \) [K] is the temperature of the ambient air
% \( \beta_0 \) [m^2/s/K^{3/2} if ni=0.5] is the temp.parameter
% ni is the temperature parameter for the environment
% 3. FMK [%] is the Fireball multiplication factor FMK used to estimate the fireball radius
% Position of the fireball radius is the position where the difference between ambient temperature and fireball temperature is less than FMK.
% 4. tstart [s] is the starting time for hot balloon modelling
% 5. tstop [s] is the end time for hot balloon modelling
% 6. Nt is the number of points for the time interval <tstart,tstop>
% 7. p0 [Pa], r0 [kg/m3], T0 [K] ambient values for pressure, density & temperature, respectively

if choice == 1
    % Option 1
    % Initial conditions based on gas state after a blast
    % define and load the MAT file
    InputFileName='input/RotationallySymmetric3DBlastWithParticles';
    load(InputFileName);
    % input data for fireball temperature PDE
    tstart=0; % [s] starting time
    tstop=1000; % [s] maximum time to be solved
    Nt=601; % number of timesteps in pde
    % ambient values for pressure (p0) and density (r0) remaining after shock
    p0 = blast.pressures(end,end);
    r0 = blast.densities(end,end);
    % initial profiles
    InitialFireballDensity = blast.densities(:,end); % space vector
    InitialFireballTemperature = blast.pressures(:,end);/
    InitialFireballDensity /R; % space vector
FireballRadialCoordinate = x; % [m] radial coordinate vector

else
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Option 2 %%%%%%%%%%%%%%%%%
  %%% Initial conditions based on generic temperature %%%
  %%% profile T(r)=T0+A.r^(-k) %%%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %
  % input data for fireball temperature PDE
  tstart=0; % [s] starting time
  tstop=2000; % [s] maximum time to be solved
  Nt=1000; % number of timesteps in pde

  % ambient conditions
  p0 = 101325; % [Pa] standard pressure at sea level
  T0 = 278.15; % [K] e.g. this temperature

  % constants not present in the input file
  R=287; % [J/kg/K] gas constant of dry air

  % define radial coordinate
  x = linspace(1e-3,2,3000); % [m] space coordinate vector
  FireballRadialCoordinate = x; % [m] space coordinate vector

  % define initial temperature profile T(x)=T0+A.x^(-k)
  A=1e-3; k=3;
  InitialFireballTemperature = T0+A.*FireballRadialCoordinate.^(-k); % artificially implemented
  T0=InitialFireballTemperature(end); % need to redefine ambient temperature in the case the space mesh is short.

  % ambient density follows the equation of state
  r0=p0/T0/R; % [kg/m3]
end

% temperature characteristics of the thermal cavity
beta0 = 8e-5;
ni = 0.5;
FMK = 1;

% constants not present in the input file
g = 9.80665; % [m/s2] gravitational acceleration
gamma = 1.4; % ratio of the specific heat at constant pressure to the specific heat at constant volume

% write to command window
hm = sprintf('I use following thermal properties beta(T)=%g*T^%g K^%g',beta0,ni,ni);
disp(hm);
hm = sprintf('I use following Fireball multiplication factor FMK=%g%%',FMK);
disp(hm);
hm = sprintf('Start time %gs; Stop time %gs; Number of points %g',tstart,tstop,Nt);
disp(hm);
hm = sprintf('Start mesh %g; Stop mesh %g; Number of points %g',x(1),x(end),length(x));
disp(hm);
function y = expspace(d1, d2, n)
% EXPSPACE exponentially spaced vector.
% EXPSPACE(X1, X2) generates a row vector of 50 logarithmically
% equally spaced points between decades 10^X1 and 10^X2. If X2
% is pi, then the points are between 10^X1 and pi.
% EXPSPACE(X1, X2, N) generates N points.
% For N < 2, LOGSPACE returns 10^X2.
% Class support for inputs X1,X2:
  float: double, single
% See also LINSPACE, :.

% Copyright 1984-2005 The MathWorks, Inc.
% $Revision: 5.11.4.2 $  $Date: 2005/10/03 16:12:49 $

if nargin == 2
  n = 50;
end
n = double(n);

if d2 == pi || d2 == single(pi)
  d2 = exp(d2);
end

y = exp(1).^ [d1+(0:n-2)*(d2-d1)/(floor(n)-1), d2];
FireballTempPDEMeerson.m

function u = FireballTempPDEMeerson(x,t)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Temperature of a fireball as function of radial coordinate and %
% solved via PDE (partial differential equation) solver in Matlab %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m = 2;

sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t); % solve pde
% Extract the first solution component as u.
u = sol(:,:,1); % fireball temperature u(time,space)

% --------------------------------------------------------------
function [c,f,s] = pdex1pde(x,t,u,DuDx)
global beta0; % [m2/s] beta(T)=beta0 * T^ni;
beta(T)=ThermalDiffusivityGas/TempAmbient*DensityGas/DensityAmbient
global ni; % [ ] power coefficient
c = 1/beta0/(u^2);
f = (u^(ni-1)) * DuDx;
s = 0;

% --------------------------------------------------------------
function u0 = pdex1ic(x)
global InitialFireballTemperature; % [K] temperature space vector
global FireballRadialCoordinate; % [m] radial coordinate vector
u0 = interp1(FireballRadialCoordinate,InitialFireballTemperature,x);
% initial temperature profile

% --------------------------------------------------------------
function [pl,ql,pr,qr] = pdex1bc(xl,ul,xr,ur,t) % boundary conditions
global p0; global r0; % ambient pressure [Pa] and density [kg/m3]
global R; % 287 J/kg/K specific gas constant of dry air
pl = 0; % dT/dt(x=1)=0
ql = 1;
pr = ur-p0/r0/R; % T(x=r)=p0/r0/R
qr = 0;
FireballDensity.m

```matlab
function FireballDensity = FireballDensity(fireball)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Density of a fireball 
% calculated from Density*Temp=Density0*Temp0=const 
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global p0; global r0; % ambient pressure [Pa] and density [kg/m3]
global R; % 287 J/kg/K specific gas constant of dry air

for i = 1:size(fireball.temperature,1) % loop over number of times
    FireballDensity(i,:) = (r0*(p0/r0/R)) ./ fireball.temperature(i,:);
end
```

---

**FireballDensity.m**

```matlab
function FireballDensity = FireballDensity(fireball)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Density of a fireball 
% calculated from Density*Temp=Density0*Temp0=const 
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global p0; global r0; % ambient pressure [Pa] and density [kg/m3]
global R; % 287 J/kg/K specific gas constant of dry air

for i = 1:size(fireball.temperature,1) % loop over number of times
    FireballDensity(i,:) = (r0*(p0/r0/R)) ./ fireball.temperature(i,:);
end
```
FireballRadius.m

```matlab
function FireballRadius = FireballRadius(fireball)
% Radius of a fireball
% estimated from fireball temperature profile
% fireball radius is equal to position where T(x)=Tambient*(1+FMK/100)
% parameter FMK can be e.g. 50% or 10% or 1% or 0.1% etc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% error message if any of the calculated T(t,x) is not close to initial %
% temperature within the defined tolerance (FMK [%]) at right boundary
if max(fireball.temperature(:,end)) > ( (1+FMK/100)*
fireball.temperature(1,end) )
    hm = sprintf('Error in calculation of T(t,x). T(t,x=xR) @ right
    boundary is not close initial temperature (probably ambient). /n Check
    initial T(t=0,x) or extend the space mesh.');
    disp(hm);
end
% error message if the initial T(t=t0,x) is not close to ambient %
% temperature within the defined tolerance (FMK [%])
if fireball.temperature(1,end) > ( (1+FMK/100)* p0/r0/R )
    hm = sprintf('Initial temperature T(t0,x) @ right boundary is not
    close ambient temperature within tolerance defined in FMK. Check the
    initial T(t=t0,x).');
    disp(hm);
end
for i = 1:size(fireball.temperature,1)
    [~,FireballRadiusIndex] = min(...
        abs(...
            fireball.temperature(i,:)...
            - (1+FMK/100)*fireball.temperature(1,end) )...)
    FireballRadius(i) = x(FireballRadiusIndex);
end
% if fireball.temperature(x) @ certain t starts to get below %
% (1+FMK/100)*Tambient %
% which will be visible as decreasing radius with time %
% then keep as fireball radius the maximum reached value %
% index for maximum radius %
% [~,FireballMaxRadiusIndex] = min( abs( FireballRadius-
% max(FireballRadius) ) ) ; % does not work if FireballRadius has more than
% one local maximum
```
CounterWhileLoop = 3;
RadiusDifference = FireballRadius(2)-FireballRadius(1);
while ( RadiusDifference >= 0 && CounterWhileLoop<=length(FireballRadius) )
    RadiusDifference = FireballRadius(CounterWhileLoop) - FireballRadius(CounterWhileLoop-1);
    CounterWhileLoop = CounterWhileLoop+1;
end
FireballMaxRadiusIndex = CounterWhileLoop-2;

if FireballMaxRadiusIndex < size(fireball.temperature,1)
    for i = FireballMaxRadiusIndex+1 : size(fireball.temperature,1)
        FireballRadius(i) = FireballRadius(FireballMaxRadiusIndex);
    end
end
function FireballDensityMean = FireballDensityMean(fireball)

% Average temperature of the fireball
% estimated from fireball temperature profile (within fireball radius)

global x; % [m] radial coordinate
global FMK; % [%] fireball multiplication factor that determines fireball radius R: T(x=R)=K/100*(Tmax-Tambient)

for i = 1:length(fireball.radius)
    [~,FireballRadiusIndex] = min(abs(fireball.radius(i)-x)); % finds index for vector x that corresponds to value of FireballRadius
    FireballDensityMean(i) = mean(fireball.density(i,1:FireballRadiusIndex)); % vector(time); average fireball density over space at given time
end

% finds index for vector t that corresponds to ambient density within the % given tolerance: (1+FMK/100)*Densityambient
[~,FireballMaxDensityIndex] = min(abs( FireballDensityMean - (1- FMK/100)*fireball.density(1,end) ) ); % for time larger than this point (i.e. for t>t(FireballMaxDensityIndex)
% approximate mean fireball density is equal to ambient density
if FireballMaxDensityIndex < size(fireball.temperature,1)
    for i = FireballMaxDensityIndex+1 : size(fireball.temperature,1)
        FireballDensityMean(i) = fireball.density(i,end);
    end
end
FireballTempMean.m

function FireballTempMean = FireballTempMean(fireball)

% Average temperature of the fireball estimated from fireball temperature profile (within fireball radius)

global x; % [m] radial coordinate
global p0; global r0; % ambient pressure [Pa] and density [kg/m3]
global FMK; % [%] fireball multiplication factor that determines fireball radius R: T(x=R)=K/100*(Tmax-Tambient)

for i = 1:length(fireball.radius)
    [~,FireballRadiusIndex] = min(abs(fireball.radius(i)-x)); % finds index for vector x that corresponds to value of FireballRadius
    FireballTempMean(i) = mean(fireball.temperature(i,1:FireballRadiusIndex)); % vector(time); average fireball temperature over space at given time
end

% finds index for vector t that corresponds to ambient temperature within given the tolerance: (1+FMK/100)*Tambient
[~,FireballMinTempIndex] = min(abs(FireballTempMean - (1+FMK/100)*fireball.temperature(1,end)));
% for time larger than this point (i.e. for t>t(FireballMinTempIndex)
% approximate mean fireball temp is equal to ambient temperature
if FireballMinTempIndex < size(fireball.temperature,1)
    for i = FireballMinTempIndex+1 : size(fireball.temperature,1)
        FireballTempMean(i) = fireball.temperature(1,end);
    end
end
FireballTempTimeStepCheck.m

function message = FireballTempTimeStepCheck(vector,t, tolerance)
% Function FireballTempTimeStepCheck.m
% checks if dataset t & vector is linear across three
% closest data points within a tolerance
% Input:
%   vector
%   - fireball.temperature(:,1) vector at the centre of the fireball
%   - fireball.temperatureMean(:) column vector
%   t time vector
%   tolerance percentage
% Output:
%   message showing that the three data points are not linear
%   across within tolerance
%   message is empty if three closest data points are linear
%   within tolerance
if (size(vector,1)==1) && (size(vector,2)>1)
    vector = vector';
end
% create linear interpolation between closest two points
for TimeIndex = 3:size(vector,1)
    slope = (vector(TimeIndex,1)-vector(TimeIndex-1,1))... / (t(TimeIndex)-t(TimeIndex-1));
    intercept = vector(TimeIndex,1)-slope*t(TimeIndex);
    TempExpected = slope * t(TimeIndex-2) + intercept;
    if abs( (vector(TimeIndex-2,1)-TempExpected) / TempExpected ) >
        tolerance/100 % difference larger than tolerance
        message(TimeIndex) = 1; % 3rd point not within tolerance for linear function
    else
        message(TimeIndex) = 0; % 3rd point within tolerance for linear function
    end
end
function [T Y] = FireballVelocityAndPositionODE(fireball,t)
% FireballVelocityAndPositionODE calculates velocity and position of the fireball via Matlab's ODE solver ode45
% Matlab's ODE solver ode45
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% global timeODE; global timeODEindex;
% global FireballMassMean;

timeODEindex = 1;
timeODE(1) = 0;
FireballMassMean(1) = 0;
FireballMassMeanGradient(1) = 0;

% define the initial condition
IC = [0 0]; % initial condition for [FireballPosition FireballVelocity]
% calculate v(T) by integrating dv/dt for each particle (T=time)
% Y=particle velocity
[T,Y] = ode23( @(tt,y) ...
    FireballEquation(tt,y,t,fireball),...
    tspan,...
    IC );
% Y 1st column particle position, 2nd column particle velocity; rows =
% T time corresponding to rows in Y
end

function dy = FireballEquation(tt,y,t,fireball)
% Ordinary differential equations
% Calculation of fireball velocity and fireball position as dependent on
% fluid velocity and density
% y(1) fireball position [m]
% y(2) fireball velocity [m/s]
% dy(1,1) dxp/dt derivation of fireball position over time [m/s]
% dy(2,1) dvp/dt derivation of fireball velocity over time [m/s2]

global r0; % ambient density [kg/m3]
global g; % [m/s2] gravitational acceleration
global timeODE; global timeODEindex;
global FireballMassMean;

% define the if statement
if ( y(2) < 0 )
    y(2)=0; % hard constrain: if fireball velocity is negative then it
    % must equal to 0
    % negative velocity is unrealistic - model may be incorrect
    dy(1,1)=0;
    hm = sprintf('Fireball height is decreasing (velocity < 0) at %g s.
    Model for velocity may be incorrect.\',tt);
    disp(hm);
end

timeODEindex = timeODEindex + 1;
timeODE(timeODEindex) = tt;
FireballDensityMean = interp1(t.fireball,fireball.densityMean,tt); %
interp1(x,y,x0) interpolate data set (x,y) at positions x0
FireballRadius = interp1(t.fireball,fireball.radius,tt); % interp1(x,y,x0)
interpolate data set (x,y) at positions x0
FireballSurface = 4 * pi * FireballRadius.^2; % projected area of a
sphere
FireballVolume = 4/3 * pi * FireballRadius.^3;
if y(1) < FireballRadius % fireball is a hemisphere (not full sphere yet)
    FireballSurfaceUnderground = 2*pi*FireballRadius * (FireballRadius-y(1)); % need to substract the area under ground
    FireballSurface = FireballSurface - FireballSurfaceUnderground;
    FireballVolumeUnderground = 1/6*pi*(FireballRadius-y(1))^3 * (3*(FireballRadius^2-y(1)^2) + (FireballRadius-y(1))^2); % need to
    substract the volume under ground
    FireballVolume = FireballVolume - FireballVolumeUnderground;
end
FireballMassMean(timeODEindex) = FireballDensityMean * FireballVolume;

Cd = FireballDragCoefficientODESandia(FireballRadius,y(2)); % y(2) is
FireballVelocity

% dFireballPosition/dt=v
y(2) = round(y(2)*100000)/100000; % accuracy at 3rd decimal point is
sufficient !23/01/2013!

dy(1,1) = y(2); % y(2) is FireballVelocity

% % % dFireballVelocity/dt=....
% J.Werne - Plume model for the boundary layer dynamics in hard
turbulence%
% dy(2,1) = g*( r0/FireballDensityMean - 1 )... %
% - ( Cd/2 * r0 * FireballSurface / FireballMassMean(timeODEIndex) ) * (y(2)^2);

if ( y(2) < 0 ) || ( dy(1,1) < 0 )
    y(2)=0; % hard constrain:if fireball velocity is negative then it
must equal to 0
    % negative velocity is unrealistic - model may be incorrect
    dy(1,1)=0;
    hm = sprintf('Fireball height is decreasing (velocity < 0) at %g s.
Model for velocity may be incorrect.',tt);
    disp(hm);
end
end

function Cd =
FireballDragCoefficientODESandia(FireballRadius,FireballVelocity)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Drag coefficient calculation
% D.Dobranich et al. - The Fireball Integrated Code Package
% Sandia Report, SAND97-1585, UC-505, Sandia National Laboratories
% (1997) p16
global p0; global r0; % ambient pressure [Pa] and density [kg/m3]
global R; % 287 J/kg/K specific gas constant of dry air

ReynoldsNumber = r0 * abs(FireballVelocity-0) * 2*FireballRadius / DynamicViscosity( p0/r0/R );

if ReynoldsNumber == 0 % relative velocity = 0 => no drag i.e. Cd=0
    Cd = 0;
end

if (ReynoldsNumber <= 1.9) && (ReynoldsNumber > 0) % Stokes flow
    a=24; b=-1; Cd = a*(ReynoldsNumber^b);
end

if (ReynoldsNumber > 1.9) && (ReynoldsNumber < 500) % transition region
    a=18.5; b=-0.6; Cd = a*(ReynoldsNumber^b);
end

if ReynoldsNumber >= 500 % Newton's law
    a=0.44; b=0; Cd = a*(ReynoldsNumber^b);
end

end

function DynamicViscosity = DynamicViscosity(gas_temperature)

% Calculation of Dynamic Viscosity for ideal gas
% see also http://en.wikipedia.org/wiki/Viscosity
% Valid for temperatures between 0 < T < 555 K with an error due to pressure less than 10% below 3.45 MPa
% Calculation is made for standard air

Tref = 291.15; % [K] reference temperature
DynamicViscosityRef = 18.27e-6; % [Pa.s] reference viscosity at Tref
C = 120; % [K] Sutherland's constant
T = gas_temperature; % [K] gas temperature

a = Tref + C;
b = T + C;

DynamicViscosity = DynamicViscosityRef * a / b * (T/Tref)^(3/2); % [Pa.s]

end
function [] = FireballPlot(fireball,t)

global x; global p0; global r0; global R;
%------
figure;
subplot(2,2,1);
semilogy(x, fireball.temperature(:,:)); hold on;
semilogy([x(1) x(end)], [p0/r0/R p0/r0/R],'k'); hold on;
title('Thermal cavity temperature');
xlabel('Distance [m]');
ylabel('Thermal cavity temperature (t,x) [K]');

subplot(2,2,3);
plot(x, fireball.density(:,:)); hold on;
plot([x(1) x(end)], [r0 r0],'k'); hold on;
title('Thermal cavity density');
xlabel('Distance [m]');
ylabel('Thermal cavity density (t,x) [kg/m^2]');

subplot(2,2,2);
semilogx(t.fireball, fireball.temperatureMean,'.'); hold on;
semilogx([t.fireball(1) t.fireball(end)], [(p0/r0/R) (p0/r0/R)],'k');
legend('DSTO model','Ambient temperature');
title('Thermal cavity mean temperature');
xlabel('Time [s]'); ylabel('Thermal cavity mean temperature [K]');

subplot(2,2,4);
semilogx(t.fireball, fireball.densityMean,'.'); hold on;
semilogx([t.fireball(1) t.fireball(end)], [r0 r0], 'k');
legend('DSTO model','Ambient density');
title('Thermal cavity mean density');
xlabel('Time [s]'); ylabel('Thermal cavity mean density [kg/m^3]');
%------
figure;
subplot(2,2,1);
plot(t.ODEfireball, fireball.velocity,'.'); hold on;
title('Thermal cavity velocity');
xlabel('Time [s]'); ylabel('Thermal cavity velocity [m/s]');

subplot(2,2,2);
plot(t.ODEfireball, fireball.position,'.'); hold on;
title('Thermal cavity cloud height');
xlabel('Time [s]'); ylabel('Thermal cavity elevation [m]');

subplot(2,2,3);
semilogx(t.fireball, fireball.radius,'.'); hold on;
title('Thermal cavity radius');
xlabel('Time [s]'); ylabel('Thermal cavity radius [m]');

subplot(2,2,4);
semilogx(t.fireball, fireball.massMean,'.'); hold on;
title('Thermal cavity mass');
xlabel('Time [s]'); ylabel('Thermal cavity mass [kg]');
%------

%% -------
% Fig: scaling laws
% CloudHeight/t^(1/2)=constant for weak thermals [Batchelor 1954]
% CloudHeight/t^(7/5)=constant for strong thermals [Skvortsov 2012]
% CloudRadius/t^(1/5)=constant for strong thermals [Skvortsov 2012]
figure;
derivAltitude1 =
diff(log10(fireball.position))./diff(log10(t.ODEfireball));
derivRadius = diff(log10(fireball.radius))./diff(log10(t.fireball'));
% 2 following lines are there to remove high exponent values which are
% artifacts of the simulation as
% they scale with initial time (tstart)
[-tODEfireballIndex] = min( abs( t.ODEfireball - 5e-5 )  ); % ignore all
values at time less < 5e-5 s
loglog(t.ODEfireball(tODEfireballIndex+1:end),derivAltitude1(tODEfireball
Index:end),'ok'); hold on;
loglog(t.fireball(2:end),derivRadius,'squarer'); hold on;
loglog([t.fireball(2) 40], [7/5 7/5],'-.k'); hold on;
loglog([t.fireball(2) 40], [2 2],':.k'); hold on;
loglog([30 t.fireball(end)], [1/2 1/2],'--k'); hold on;
loglog([t.fireball(2) 40], [1/5 1/5],'-.r'); hold on;
loglog([t.fireball(2) 40], [4/53 4/53],':.r'); hold on;
xlabel('TIME (s)'); ylabel('EXPONENT \alpha* in function Z(t)~t^{
\alpha*} & \beta* in function R(t)~t^{ \beta*}');
legend('Numerical model Z(t)~t^{\alpha*}','...'
'Numerical model R(t)~t^{\beta*}','...'
'Asymptote Z(t)~t^{7/5} [DSTO analyt.model]','...'
'Asymptote Z(t)~t^{2} [Meerson 1989]','...'
'Asymptote Z(t)~R(t)~t^{(1/2)} [Batchelor 1954]','...'
'Asymptote R(t)~t^{(1/5)} [DSTO analyt.model]','...'
'Asymptote R(t)~t^{4/53} [Meerson 1989]','...'
'Location', 'NorthEast');
ylim([1e-2 1e3]);
title('Experimental data and scaling laws');
%------
# Modelling of Buoyant Thermals

The objective of this technical report is to document mathematical and physical modelling efforts that calculate the transient behaviour of buoyant thermals. Buoyant thermals form a part of a source term for plume dispersion that can be generated by extreme events in the environment such as blasts, nuclear detonations, volcanic eruptions or bushfires. While plume dispersion is relatively well understood, the source term remains often unclear. Numerical and analytical models presented here provide a mathematical understanding and can predict the displacement and temperature evolutions of buoyant thermal cavities. Verification of the models shows an agreement with experimental data found in literature.