TITAN CODE

USER'S GUIDE

by

MICHAEL GEHMEYR
and
DIMITRI MIHALAS

LABORATORY for COMPUTATIONAL ASTROPHYSICS

DEPARTMENT of ASTRONOMY
UNIVERSITY OF ILLINOIS

and

NATIONAL CENTER for SUPERCOMPUTING
APPLICATIONS

Version 1.0, February 1994

DISCLAIMER

Although every attempt has been made to provide accurate and error-free data and software, the University of Illinois makes no warranty, express or implied, for the accuracy, completeness, or usefulness of the information and data provided. Further, it assumes no legal liability or responsibility whatsoever, for any damage or loss users may sustain as a result of using them. In no event shall the University of Illinois be liable for any special, indirect, or consequential damages, or any damages whatsoever resulting from loss of use, data, or profits, whether in an action of contract, negligence, or other tortious action, or any action whatsoever, arising out of, or in connection with, the use or performance of this software. The University of Illinois makes no representations about the suitability of this software for any purpose; users use these data and codes entirely at their own risk.

ACKNOWLEDGEMENTS

We thank K.-H. Winkler for introducing us to, and teaching us about, adaptive-grid techniques in radiation hydrodynamics. D.M. thanks the National Solar Observatory for generous support of his sabbatical visit to Tucson during the Spring semester of 1993. M.G. thanks C. Keller, the director of the Institute for Geo- and Planetary Physics at the Los Alamos National Laboratory, for his warm hospitality during which part of the user's guide has been written. This work has been supported in part by National Science Foundation grants AST-8914143 and AST-920113, National Aeronautics and Space Administration grant 90-NASA-M-1665, and research funds made available by the University of Illinois.

Contents

I. INTRODUCTION

This user's guide describes how to use TITAN which is a modern computational tool for generating accurate solutions of the one-dimensional radiation hydrodynamical equations in a broad variety of initial and boundary conditions. It employs an implicit adaptive grid procedure to deal with the multiple length and time scales inherent in such problems. The grid is designed to detect, resolve, and track nonlinear features of the flow. TITAN is intended for use by the astrophysical community both as a research and a teaching tool. TITAN comes with a suite of test problems that cover purely hydrodynamical and radiation hydrodynamical calculations as well as time dependent radiation transport through static media. An abbreviated version of this guide is published in:

Adaptive Mesh Radiation Hydrodynamics with TITAN, M. Gehmeyr and D. Mihalas, *Physica D,* **??,** ??, 1994

In what follows, the one-dimensional equations of radiation hydrodynamics are briefly reviewed, the essential numerical methods of the code are highlighted, the suite of test problems, and how to run and modify them, is described, and ways how to set up new problems are suggested. Further details about the equations used in the code are given in the TITAN Code Reference Manual.

To assist the reader we write all ordinary physical and mathematical variables in *italic type*, and `FORTRAN` quantities in `typewriter type`. Further, we have highlighted all parts of the code that are explicitly discussed, as well as all examples that could be viewed on a monitor, with a gray background.

II. RADIATION HYDRODYNAMICS

A. The Equations

Reference

D. Mihalas and B. W. Mihalas, "Foundations of Radiation Hydrodynamics", Oxford University Press, New York, 1984

The equations of Newtonian gray radiation hydrodynamics for a single fluid in the comoving-frame of reference taking into account terms of $O\left(\frac{v}{c}\right)$ are formulated according to Mihalas & Mihalas. The two meaningful geometries for radiation transfer are slab symmetry, designated by the parameter $\mu = 0$, and spherical symmetry, $\mu = 2$. The hydrodynamical equations by themselves can also be solved in cylindrical symmetry, $\mu = 1$.

The mass within a shell of radius r (a slab of distance r) is given by by the density:

$$m(r) = \int^r \rho(x) \frac{d(x^{\mu+1})}{\mu + 1} \tag{1}$$

where ρ is the density of the gas. The set of governing equations are, defining the comoving time derivative as $D_t \equiv \frac{\partial}{\partial t} + u\frac{\partial}{\partial r}$:

continuity

$$D_t(\rho) + \rho \frac{\partial(r^\mu u)}{r^\mu \partial r} = 0 \tag{2}$$

momentum

$$\rho D_t(u) + \frac{\partial(p)}{\partial r} + \frac{\partial\left(r^{3\mu/2} P_Q\right)}{r^{\mu/2}\partial r} + \rho\Big[\Big(1 - \frac{\mu}{2}\Big)g + 2\pi\mu\frac{Gm}{r^\mu}\Big] - \frac{1}{c}\rho\chi_F F = 0 \tag{3}$$

radiative momentum

$$\rho D_t\Big(\frac{F}{\rho c^2}\Big) + \Big[\frac{\partial(P)}{\partial r} + \frac{\mu}{2}\frac{(3P - E)}{r}\Big] + \frac{F}{c^2}\frac{\partial u}{\partial r} + \frac{1}{c}\rho\chi_F F = 0 \tag{4}$$

fluid plus radiative energy

$$\rho D_t\Big(e+\frac{E}{\rho}\Big)+\frac{\partial(r^\mu F)}{r^\mu\partial r}+(p+P)\frac{\partial(r^\mu u)}{r^\mu\partial r}+P_Q\Big[\frac{\partial u}{\partial r}-\frac{\mu}{2}\frac{u}{r}\Big]$$
$$+\frac{\mu}{2}(E-3P)\frac{u}{r}=0 \qquad (5)$$

radiative energy

$$\rho D_t\Big(\frac{E}{\rho}\Big)+\frac{\partial(r^\mu F)}{r^\mu\partial r}+P\frac{\partial(r^\mu u)}{r^\mu\partial r}+\frac{\mu}{2}(E-3P)\frac{u}{r}$$
$$+c\rho\Big[\kappa_E E-\kappa_P a_r T^4\Big]=0 \qquad (6)$$

In equation (5) e, the specific internal energy of the gas, and the radiative energy E, are combined to form a "total internal" energy. Correspondingly there are two different contributions to the work: that of the gas pressure p and that of the radiative stress $P\frac{\partial(r^\mu u)}{r^\mu\partial r}+\frac{\mu}{2}(E-3P)\frac{u}{r}$. In addition we provide for the work done by an artificial stress P_Q. The radiant energy is transported via the radiative flux F. We have neglected terms that combine radiation quantities with the gas acceleration.

The gas momentum equation (3) allows for gravitational forces in two geometries, with G standing for the Newtonian gravitational constant, and g for a constant acceleration. The radiation force, $\frac{1}{c}\rho\chi_F F$, the gradient of the gas pressure, and the artificial stress are exerted on the fluid elements, which move with velocity u.

The equation for the radiative flux (4) can be interpreted as the photon momentum equation which couples to the gas momentum via the last term.

The radiative energy equation (6) describes transport and work of the photons and constitutes, together with equation (4), the first two frequency and solid angle integrated moments of the transport equation. The last term in (6) can be viewed approximately as $\frac{1}{\tau_r}\Big(E-a_r T^4\Big)$, a_r denoting the radiation constant, and then be interpreted as a heat exchange on a time scale $\tau_r\equiv\frac{1}{c\,\kappa\rho}$, which is the lifetime of a photon traveling a mean free path $\frac{1}{\kappa\rho}$ between two scattering or absorption events. If $E>a_r T^4$ then radiative energy is converted into heat, and *vice versa*. By assuming such a form for the radiative coupling term we restrict ourselves to a radiating fluid in LTE.

Equations (4) & (6) are referred as "full transport" which is a correct description in all regimes for the one-dimensional case. In particular, the

"diffusion regime" is recovered when (4) becomes equivalent to

$$F = -\frac{c}{3\rho\chi}\,\frac{\partial E}{\partial r}. \tag{7}$$

Substituting formula (7) into equation (6) yields the non-equilibrium diffusion because $a_r T_r^4 \equiv E$ defines a radiative temperature field which is allowed to differ from the fluid temperature T. Imposing the additional condition that $\kappa_E E = \kappa_P a_r T^4$, or $T_r = T$ leads back to equilibrium diffusion.

In spherical symmetry it is customary to define the luminosity as the surface brightness $\mathcal{L} = 4\pi r^2 F$ of a shell with radius r.

B. Adaptive Mesh Transformation

Reference

Adaptive–Mesh Radiation Hydrodynamics. I. The Radiation Transport Equation in a Completely Adaptive Coordinate System, K.–H. Winkler, M.L. Norman, and D. Mihalas, *J.Q.S.R.T.*, **31,** 473, 1984

W. M. Tscharnuter and K.–H. A. Winkler, *Comput. Phys. Commun.*, **18,** 171, 1979

The mass definition (1) can be immediately rewritten in differential form as:

$$dm = \rho\; d\Big[\frac{r^{\mu+1}}{\mu+1}\Big] \tag{8}$$

The partial differential equations (2)–(6) should be represented in finite difference form on an arbitrary moving grid distribution. This distribution can be obtained by uniquely subdividing the whole computational domain into cells around each grid point k with $k = 1, \ldots N$. If (2)–(6) are integrated over those cells each encompassing a volume (length) $dV_k = \frac{r^{\mu+1}}{\mu+1}\Big|_k$, they reduce to a set of N ordinary differential equations (in time). The proper transformation is achieved by the adaptive–mesh Reynolds transport theorem as described in Winkler *et al.*Hereby the cells are assumed to move with a relative velocity $u_{rel}|_k$ with respect to the fluid flow.

For each cell ($k = 1, \ldots N$) the following equations hold:

continuity

$$\frac{d}{dt}\int_V \rho\ dV + \oint_{\partial V}\ \rho u_{rel}\ dS = 0 \tag{9}$$

fluid momentum

$$\frac{d}{dt}\int_V u\rho\ dV - \oint_{\partial V}\ u\rho u_{rel}\ dS + \int_V r^{\mu} d(p) + \int_V r^{-\mu/2} d[r^{3\mu/2}\ P_Q]$$

$$+ \int_V \Big[\Big(1-\frac{\mu}{2}\Big)g + 2\pi\mu\frac{Gm}{r^{\mu}} - \chi_F\frac{F}{c}\Big]\ \rho dV = 0 \tag{10}$$

radiative momentum

$$\frac{d}{dt}\int_V \frac{F}{c^2}\ dV - \oint_{\partial V}\ \frac{F}{c^2}u_{rel}\ dS + \int_V r^{\mu} d(P) + \int_V\ \Big[\frac{\mu}{2}\frac{(3P-E)}{r}\Big]\ dV$$

$$+ \int_V \frac{F}{c^2} r^{\mu} d(u) + \int_V \Big[\chi_F\frac{F}{c}\Big]\ \rho dV = 0 \tag{11}$$

fluid plus radiative energy

$$\frac{d}{dt}\int_V \Big(e+\frac{E}{\rho}\Big)\rho\ dV - \oint_{\partial V}\ \Big(e+\frac{E}{\rho}\Big)\rho u_{rel}\ dS + \int_V d[r^{\mu}\ F]$$

$$+ \int_V (p+P)\ d[r^{\mu}\ u] + \int_V P_Q\Big[\frac{\partial u}{\partial r} - \frac{\mu}{2}\frac{u}{r}\Big]\ dV + \int_V \Big[\frac{\mu}{2}(E-3P)\frac{u}{r}\Big]\ dV = 0 \tag{12}$$

radiative energy

$$\frac{d}{dt}\int_V E\ dV - \oint_{\partial V}\ E u_{rel}\ dS + \int_V d[r^{\mu}\ F]$$

$$+ \int_V P\ d[r^{\mu}\ u] + \int_V \Big[\frac{\mu}{2}(E-3P)\frac{u}{r}\Big]\ dV + \int_V\ c\ \Big[\kappa_E E - \kappa_P a_r T^4\Big]\ \rho dV = 0 \tag{13}$$

Here ∂V denotes the surface of the volume element V. In the transformation onto the adaptive grid Neumann-Richtmyer's idea of introducing an artificial viscosity to treat shocks in the flow has been implemented according to Tscharnuter & Winkler. A tensor formulation is used which introduces the artificial stress:

$$P_Q = -\frac{4}{3}\rho\mu_Q\Big[\frac{\partial u}{\partial r} - \frac{\mu}{2}\frac{u}{r}\Big] \tag{14}$$

in conjunction with a linear and quadratic artificial viscosity, ℓ_Q giving the shock width:

$$\mu_Q = C_1 \ell_Q \sqrt{\gamma p/\rho} - C_2 \ell_Q^2 \min\Big[\, 0, \frac{\partial(r^\mu u)}{r^\mu \partial r} \,\Big] \tag{15}$$

C. Symbolic Notation

Reference

WH80s: Numerical Radiation Hydrodynamics, K.–H. Winkler and M.L. Norman,in *Astrophysical Radiation Hydrodynamics,* (Dordrecht: Reidel), 71, 1986

The volume (length) integrals in the equations above (9)–(13) can be approximated with finite differences in the simplest way if one assumes that the integrands remain constant. In general this is valid as long as the linear dimensions of the integral volumes are small compared to any physical length scale in the problem. In the case of smooth variations of the physical variables this can be achieved by decomposing the physical domain into a sufficiently large number of numerical zones. In the case of (*in praxi*) discontinuities the demand is that the physical variables still are not allowed to vary significantly between neighboring zones. This can be satisfied only if the zones become very small; or in other words if the numerical grid *resolves* such nonlinear features. Both situations are taken care of by the implicit nature of the adaptive mesh.

In a like spirit the spatial differential operators can interpreted in the simplest form by spatial differences denoted by Δ in symbolic form. Equation (8) valid for all zones $k = 1, \ldots, N$ on the numerical domain then can be straightforwardly expressed as:

$$\Delta m = \rho \Delta V \tag{16}$$

The temporal derivatives remain intact, the surface integrals are resolved as differences of the surface terms, and the spatial derivatives approximated as spatial differences the symbolic form of the equations (9)–(15) are again valid for all zones $k = 1, \ldots, N$:

continuity

$$\frac{d(\rho\Delta V)}{dt} + \Delta(\rho r^{\mu} u_{rel}) = \Delta\left(r^{\mu}\sigma_{\rho}\frac{\Delta\rho}{\Delta r}\right) \tag{17}$$

fluid momentum

$$\frac{d}{dt}[u\Big\langle\rho\Delta V\Big\rangle] - \Delta\left(\Big\langle\frac{dm}{dt}\Big\rangle u\right) + r^{\mu}\Delta p + \left(\frac{2-\mu}{2}g + \frac{2\pi\mu G m}{r^{\mu}}\right)\Big\langle\rho\Delta V\Big\rangle$$

$$= \phi_{Q}\Delta V + \frac{1}{c} <\chi_{F}> F\Big\langle\rho\Delta V\Big\rangle \tag{18}$$

radiative momentum

$$\frac{d}{dt}\left[\frac{F}{c^2}\Big\langle\Delta V\Big\rangle\right] - \Delta\left[\Big\langle\frac{dm}{dt}\Big\rangle\left(\frac{F}{c^2 <\rho>}\right)\right] + r^{\mu}\left(\Delta P + \frac{F\Delta u}{c^2}\right) + \frac{(3P-E)}{r}\Big\langle\Delta V\Big\rangle$$

$$= -\frac{1}{c} <\chi_{F}> F\Big\langle\rho\Delta V\Big\rangle \tag{19}$$

fluid plus radiative energy

$$\frac{d}{dt}\left[\left(e + \frac{E}{\rho}\right)\rho\Delta V\right] - \Delta\left[\frac{dm}{dt}\left(e + \frac{E}{\rho}\right) - r^{\mu}F\right]$$

$$+(p+P)\Delta(r^{\mu}u) + (E-3P)\frac{u}{r}\Delta V = \Delta\left(r^{\mu}\sigma_{e}\frac{\Delta e}{\Delta r}\right) + \epsilon_{Q}\Delta V \tag{20}$$

radiative energy

$$\frac{d}{dt}(E\Delta V) - \Delta\left[\frac{dm}{dt}\left(\frac{E}{\rho}\right) - r^{\mu}F\right] + P\Delta(r^{\mu}u) + (E-3P)\frac{u}{r}\Delta V$$

$$= c\Big[\kappa_{P}a_{r}T^{4} - \kappa_{E}E\Big]\rho\Delta V \tag{21}$$

The bracketed quantities are averaged in this staggered mesh representation, *cf.* the discussion in section III A. This set of equations corresponds to the equations $M1$, $C1$, $GM1$, $RM1$, $FE1$, $RE1$ in the TITAN Code Reference Manual. The original advection operator $\Delta(\rho r^{\mu} u_{rel})$ has been retained only in the continuity equation (17). In all other equations, for reasons discussed by Winkler and Norman, the combination $\rho r^{\mu} u_{rel}$ is replaced by $-\frac{dm}{dt}$. This is possible because the motion of the grid relative to the fluid results in an apparent mass transfer rate that can be computed according to:

$$\frac{dm}{dt} = -\rho r^{\mu} u_{rel} \tag{22}$$

Since the correct amount of mass per zone is already determined by (16) the advection operator in all other ordinary differentail equations (17)–(21) can be expressed through $-\Delta\left(\frac{dm}{dt}\right)$.

The artificial stress enters into the momentum equation (18) in the form of a force:

$$\phi_Q \Delta V \equiv r^{-(\mu/2)} \Delta \left[\frac{4}{3} \rho \mu_Q r^{(3\mu/2)} \left(\frac{\Delta u}{\Delta r} - \frac{\mu}{2} \left\langle \frac{u}{r} \right\rangle \right) \right] \tag{23}$$

and into the fluid plus radiative energy equation (20) in the form of a source:

$$\epsilon_Q \equiv \frac{4}{3} \mu_Q \rho \left(\frac{\Delta u}{\Delta r} - \frac{\mu}{2} \left\langle \frac{u}{r} \right\rangle \right)^2 \tag{24}$$

The artificial viscosity coefficient is evaluated according to:

$$\mu_Q \equiv C_1 \ell_Q a_s - \min[C_2 \ell_Q^2 \Delta(r^\mu u)/\Delta V, 0] \tag{25}$$

The shock width can be assumed as a constant $\ell_Q = \ell_0$ or proportional to the radius $\ell_Q = \ell_1 < r >$. Additional diffusion operators appear in the continuity and fluid plus radiative energy equations. Their purpose is to treat numerical artifacts that occur in contact continuities and when shock reflect from walls (wall heating). The constants σ_ρ and σ_e control the length scale over which density or internal energy is allowed to diffuse.

D. Boundary Conditions

TITAN is equipped with a broad variety of boundary conditions, allowing for everything but very special-purpose boundaries. They naturally fall into two categories which describe the behavior of the radiation field and of the fluid ath the boundaries. The inner and outer (left and right) boundary conditions obey similar laws; hence is suffices to give just the generic formulation. More details are given in the TITAN Code Reference Manual.

(1) radiative boundary:

In general the radiative boundary can be posed by requiring the radiative energy to remain constant: $\Delta E = 0$. We provide for the optically transmitting and reflecting cases. The first is defined through a flux which is computed from the free streaming formula. In addition the flux is allowed to be modified by a constant intensity field $I^{\pm}$. The second case is given through a

zero flux condition. Furthermore we have the possibility of imposing a time dependent net flux.

transmitting

$$F = \pm c g_E E \mp (2g_E + 1)\pi I^{\pm}. \tag{26}$$

reflecting flux

$$F = 0. \tag{27}$$

imposed flux

$$F = \Phi(t). \tag{28}$$

In equation (26) g_E is a surface Eddington factor connecting E to F.

(2) Eulerian boundary:

The Eulerian boundary is given by the fixed grid condition $\dot{r} = 0$. We allow for three different cases. The transmitting condition demands that temperature, density, and velocity all remain constant across the boundary. The zero flux condition necessitates that the temperature and density stay constant while the velocity vanishes. For the nonzero flux condition density and temperature can be functions of time.

transmitting

$$\Delta T = 0 \ ; \ \Delta \rho = 0 \ ; \ \Delta u = 0. \tag{29}$$

zero flux

$$\Delta T = 0 \ ; \ \Delta \rho = 0 \ ; \ u = 0. \tag{30}$$

nonzero flux

$$T = \Theta(t) \ ; \ \rho = P(t) \ ; \ u = 0. \tag{31}$$

(3) Lagrangean boundary:

The Lagrangean boundary is given by the fixed mass condition $\dot{m} = 0$. We allow for two different forcings. In the piston driven case a time dependent velocity is assumed to move the boundary. Otherwise we allow for a forcing through an external pressure.

external velocity

$$u = U_{ext}(t). \tag{32}$$

external pressure

$$p = P_{ext}(t). \tag{33}$$

In addition to the boundaries we need to provide for phantom zones so that the difference stencil stays well defined. These phantom zones have to be chosen to reflect the physical properties of the boundary conditions.

E. Constitutive Relations

References

D. Mihalas, W. Däppen, D. Hummer, and B. W. Mihalas,
Ap. J., **331,** 794, 1988; *Ap. J.*, **331,** 815, 1988;
Ap. J., **332,** 261, 1988; *Ap. J.*, **350,** 350, 1990

(1) fluid equation of state:

Arbitrary equations of state as functions of the density and gas temperature are permitted:

caloric

$$e = e(\rho, T), \tag{34}$$

thermal

$$p = p(\rho, T). \tag{35}$$

They can be given in form of tables or as analytical formulae. In the first case monotonic bicubic hermite polynomials are employed to perform the interpolations. Important thermodynamical properties for the fluid are derived through the Gibbs form $de(\rho,T) = Tds(\rho,T) + p(\rho,T)d(\frac{1}{\rho})$, where s denotes the specific entropy. Among those of particular interest is the adiabatic index $\gamma = \frac{\partial \ln p}{\partial \ln \rho}\Big|_s$.

For the purpose of the test problems a perfect gas is assumed so that: $p = \frac{R}{\overline{\mu}}\rho T$, with R standing for the universal gas constant and $\overline{\mu}$ for the mean molecular weight. Prescribing a constant adiabatic index γ then relates the gas pressure to the gas energy: $e = \frac{1}{(\gamma-1)}\frac{p}{\rho}$.

(2) radiative equation of state:

The two-equation formalism for the radiative transport, equations (6) & (4), is closed via the variable Eddington factor f_E. The radiative pressure P in this description is simply proportional to E:

$$P = f_E E. \tag{36}$$

In order to evaluate f_E we integrate the static transport equation for the radiative intensity as a function of angle ϕ, $I(x = \cos\phi)$, using a ray tracing technique. The Eddington factor is then computed as the ratio of the second to the zeroth moment:

$$f_E = \frac{\int_{-1}^{+1} I(x) x^2 dx}{\int_{-1}^{+1} I(x) dx}. \tag{37}$$

At the boundaries of the radiation field a corresponding factor, called g_E, is evaluated. f_E describes the geometric properties of the radiation field. For an isotropic intensity distribution, $I(\phi) = I_o$, we obtain $f_E = \frac{1}{3}$. This corresponds to the diffusion limit of the two-equation formalism. If the intensity distribution peaks along the line of sight, *e.g.*, $I(\phi) \propto \cos^2\phi$, we get $f_E = \frac{3}{5} > \frac{1}{3}$. The stronger the intensity field is pointed along the line of sight the closer $f_E \to 1$. This corresponds to the free streaming limit of the full transport formalism. On the other hand, if the intensity distribution peaks sideways, *e.g.*, $I(\phi) \propto \sin^2\phi$, we find $f_E = \frac{3}{15} < \frac{1}{3}$. This situation is often found in radiating shock fronts.

(3) opacities:

In general the frequency dependent opacity $\chi(\nu)$ is given by contributions from absorption, $\chi^a(\nu)$, and scattering, $\chi^s(\nu)$. The gray formulation of the radiation hydrodynamics equations introduces three different opacity means: the flux mean, the absorption (or energy) mean, and the Planck mean:

flux mean

$$\chi_F = \int_0^\infty [\chi^a(\nu) + \chi^s(\nu)] \, \frac{F(\nu)}{F} \, d\nu. \tag{38}$$

absorption mean

$$\kappa_E = \int_0^\infty \chi^a(\nu) \, \frac{E(\nu)}{E} \, d\nu. \tag{39}$$

Planck mean

$$\kappa_P = \int_0^\infty \chi^a(\nu)\ \frac{B(\nu,T)}{\sigma T^4/\pi}\ d\nu. \tag{40}$$

The flux and the energy mean both depend on the spectral distribution of the radiation field, whereas the Planck mean is given by the local gas temperature and density via the Planck distribution B. The evaluation of the first two opacity means therefore can be carried out correctly only if one also computes the frequency dependent, *i.e.,* non-gray, radiation field. This is often done with a multigroup approach. For the moment TITAN is restricted to the gray transport.

Defining the Rosseland mean opacity

$$\chi_{Ross}^{-1} = \int_0^\infty [\chi^a(\nu) + \chi^s(\nu)]^{-1}\ \frac{\partial B(\nu,T)/\partial T}{4\sigma T^3/\pi}\ d\nu, \tag{41}$$

one can show that in diffusion approximation it equals to the flux mean, *i.e.,* $\chi_F = \chi_{Ross}$. For the purpose of our test problems we assume a constant χ_{Ross}. In order to simulate the effects of a scattering or absorption dominated fluid we set both the energy mean and the Planck mean equal to a fraction $0 \leq \zeta \leq 1$ of the Rosseland mean, *i.e.,* $\kappa_E = \kappa_P = \zeta\chi_{Ross}$.

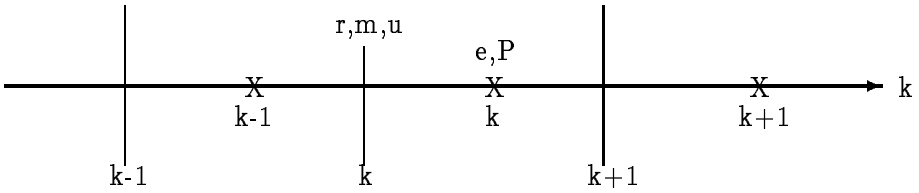
Both the fluid equation of state and the opacities are available in table lookup form. The source is described by Mihalas *et al.*in a series of articles. In this case monotonized bicubic hermite interpolation is used to extract the desired quantity $x(T,\rho)$ along with its partial derivatives $\frac{\partial \ln x}{\partial \ln T}$ and $\frac{\partial \ln x}{\partial \ln \rho}$.

III. CODE DESCRIPTION

A. Some Numerical Details

(1) staggered grid:

The physical equations (17)–(21) are discretized with finite differences. The physical quantities are defined on a staggered grid.



This means that the scalar-like quantities ρ, e, E, p, P and the tensor-like quantities P_Q are defined on the grid centers whereas the vector-like quantities r, u, m, F are defined on the grid interfaces. The grid indexing convention is such that k increases from the inner (left) to the outer (right) boundary while the center index always lies outside (to the right) of the interface index. Since temperature and density are defined on the grid centers all constitutive relations (equations of state, opacities) are also.

The staggered grid allows one to write down the spatial operators in a straightforward manner: $(\Delta p)_k = (p)_k - (p)_{k-1}$ is interface centered and hence combines directly with $(r^\mu)_k$ to form the pressure force in the momentum equation (18). Similarly, $(\Delta r^\mu u)_k = \left(r^\mu_{k+1} u_{k+1} - r^\mu_k u_k\right)$ is cell centered and hence combines directly with $(p)_k$ to express the pressure work done in the energy equation (20). Furthermore, an appropriate averaging can be given so that grid centered quantities become centered on the interfaces, and *vice versa*. For instance, $<u/r>_k = \frac{1}{2}\left((u/r)_{k+1} + (u/r)_k\right)$ to be multiplied with the radiative energy $(E_r)_k$ in the radiative energy equation (21).

(2) advection scheme:

Some attention must be paid to the advection terms. Here we have employed the second-order, upwind Van Leer advection scheme but allow also for the first-order Donor Cell scheme. To compute the advection of a quantity x, first its up- and downwind values are estimated. For scalar-like quantities

these lie at grid interfaces and for vector-like quantities they lie at the cell centers. For the continuity equation (17) the up/downwind density is then multiplied by $r^{\mu}u_{rel}$ before the spatial difference operator in employed to obtain the advected amounts. In the continuity equation the advection term therefore expands to:

$$\Delta(\rho r^{\mu}u_{rel}) = (r_{k+1}^{n+\theta})^{\mu}\left[u_{k+1}^{n+\theta} - \frac{\delta r}{\delta t}\Big|_{k+1}\right]\overline{\rho}_{k+1} - (r_{k}^{n+\theta})^{\mu}\left[u_{k}^{n+\theta} - \frac{\delta r}{\delta t}\Big|_{k}\right]\overline{\rho}_{k} \tag{42}$$

The exact formula of $\overline{\rho}$ can be found in equation $C4$, TITAN Code Reference Manual. For all other equations (18)–(21) the up/downwind quantities $\overline{x}$ are multiplied by $-\frac{\delta m}{\delta t}$, possibly spatially averaged, and then the spatial difference operator is applied.

(3) time centering:

The temporal evolution is achieved with a simple formula to advance from the old to the new time level. As a consequence all variables are functions of the integer pair (k, n), *i.e.,* of zone number and time index. The time derivative of the quanitity x, $\frac{dx}{dt}$, is represented as the difference between its value at the old (n) and the new $(n+1)$ time level, δt denoting the time step:

$$\frac{\delta x}{\delta t}\Big|_{k} \equiv \frac{\left(x_{k}^{n+1} - x_{k}^{n}\right)}{\delta t} \tag{43}$$

In the equations all quantities x, aside from the temporal operator, are defined in between the old and the new time step with the aid of the time centering parameter θ

$$x_{k} = \theta x_{k}^{n+1} + (1-\theta)x_{k}^{n}. \tag{44}$$

The algorithm becomes fully implicit for $\theta = 1$ and assumes backward-Euler differencing for $\theta = \frac{1}{2}$. In general we adopt $\frac{1}{2} < \theta \leq 1$.

(4) convergence criteria:

The physical equations are formulated in conservation form using a discrete volume scheme, *cf.* section II.C. One has to provide for the initial solution of the fundamental quantities for this set of algebraic equations. Their linearization results in a block pentadiagonal matrix. This system is inverted by the Henyey scheme and then the solution is advanced in time by Newton's method. The data on the new time level are said to be converged when

the maximum of the relative change of the variables is less than a specified tolerance, typically 10^{-5}, and the number of iterations lies within a specified range, say 3 and 15. The new time step is evaluated by a nonlinear control which incorporates accuracy considerations such that the temporal change of the variables is less than a given limit, typically 10%. A more detailed account of the solution procedure and convergence control is given in the TITAN Code Reference Manual chapters XVI and XVII.

B. The Grid Equation

Reference

Simple Adaptive Grids for Initial Value Problems,
E. A. Dorfi and L. O'C. Drury, *J. Comput. Phys.,* **69,** 175, 1987

The heart of TITAN is the adaptive grid algorithm. Its purpose is to detect, resolve, and track nonlinear features (such as shock waves or ionization fronts) of the radiating fluid. The algorithm couples the physical equations (17)–(21) with a grid equation nonlinearly in a twofold manner. First the physical equations are transformed according to the adaptive grid Reynolds transport theorem, *cf.* section II.B. This changes the equations to a particular frame of reference for each grid point. Second the grid equation is added. It has only numerical significance and determines which frame of reference for each grid point at every time index is chosen. The grid equation needs input from physical quantities in order to find the features one wishes to resolve.

In TITAN the grid equation according to Dorfi & Drury is adopted. The basic idea is to balance the grid concentration defined as

$$\mathcal{C}_k^n = \frac{\mathcal{X}(r_{k+1}^n, r_k^n)}{(r_{k+1}^n - r_k^n)} \tag{45}$$

with the resolution function

$$\mathcal{R}_k^n = \sqrt{1 + \left[\mathcal{C}_k^n\right]^2 \sum_j \left[\frac{f_{j\,k+1}^n - f_{j\,k}^n}{\mathcal{X}(f_{j\,k+1}^n, f_{j\,k}^n)}\right]^2} \tag{46}$$

of the problem under consideration. The latter measures the arc length of various physical quantities f_k^n which are typically the density, gas energy, radiative energy, or gas pressure. Demanding the grid equation

$$\mathcal{C}_k^n \propto \mathcal{R}_k^n \tag{47}$$

under the constraint

$$\frac{\alpha}{\alpha+1} \leq \frac{\mathcal{C}_{k+1}^n}{\mathcal{C}_k^n} \leq \frac{\alpha+1}{\alpha} \tag{48}$$

with $\alpha > 1$ one forces the grid to distribute in such a way that steep features in the fluid are resolved while maintaining approximately the same number of grid points within the unit arc length over the whole domain. The detailed derivation and formulation of the grid equation can be found in Dorfi & Drury. The exact implementation in TITAN is documented in detail in the TITAN Code Reference Manual section II.C. The function $\mathcal{X}$ renders the expressions in (45) & (46) dimensionless and serves as a scaling factor. In TITAN it is defined in three different ways:

(1) linear scaling:

$$\mathcal{X}(x_{k+1}^n, x_k^n) = x_{scale} \tag{49}$$

(2) logarithmic scaling:

$$\mathcal{X}(x_{k+1}^n, x_k^n) = (x_{k+1}^n + x_k^n) \tag{50}$$

(3) harmonic scaling:

$$\mathcal{X}(x_{k+1}^n, x_k^n) = (x_{k+1}^n / x_k^n) - (x_k^n / x_{k+1}^n) \tag{51}$$

The grid equation itself contains further a time scale τ which sets the time step δt below which the grid distribution remains frozen in.

In addition to the adaptive grid the user can choose between two fixed grids:

(1) Lagrangean frame:

$$\frac{r_k^{n+1} - r_k^n}{\delta t} - u_k = 0 \tag{52}$$

The grid velocity, *i.e.,* the first term, equals to the time centered fluid velocity. Applying (22) this means that the grid is fixed on a given mass shell distribution. Find equation (52) also as $LG1$ in the TITAN Code Reference Manual.

(2) Eulerian frame:

$$r_k^{n+1} - r_k^n = 0 \tag{53}$$

The grid velocity is zero and therefore the grid is frozen into a given radial distribution. Find equation (53) also as $EG1$ in the TITAN Code Reference Manual.

For the discussion of actual computations there is a very useful quantity which characterizes the performance of the adaptive grid is the grid resolution. In general this is the the grid concentration in logarithmic scaling: $(r_{k+1}^n + r_k^n)/(r_{k+1}^n - r_k^n)$. For an Eulerian grid with N grid points the grid spacing $(r_{k+1}^n - r_k^n)$ is equidistant and proportional to N^{-1} times the size of the computational domain. Hence the grid resolution for it is $\propto N$, the number of grid points. For an adaptive grid the resolution of a particular feature indicates the number of Eulerian grid points that would be needed to resolve it equivalently well.

C. Flow Chart

The following flow chart lists the subroutine names as well as their mutual dependencies through the calling sequence within TITAN:

```
titan                                                    main
      opnfil                                    opens various files
           readeos                              reads tabulated eos
           readopac                          reads tabulated opacities
      start                                        selects problem
           start{01,02,03,...}                   initializes model
                eos                                   evaluates eos
                opac                              evaluates opacities
                eddfac                    evaluates Eddington factors
                     eddf
                     eddg
                gridinit                    initializes grid equation
                     grid                             grid equation
      step                                                  driver
           peq                        pepares quantities and derivatives
                peq{r,h,rh}
                     eos
```

```
          opac
          eddfac
     iter =< niter ?                          convergence check
          matgen              generates rhs and Jacobian matrix
               mass                                 mass equation
               contin                                 continuity
                    advectc              advection at cell centers
               viscous               evaluates artificial viscosity
               gasmom{  h,rh}                         momentum
                    advecti            advection at cell interfaces
               gasnrg{r,h,rh}          fluid plus radiative energy
                    advectc              advection at cell centers
                    diffuse           evaluates artificial diffusion
               radnrg{r,  rh}                    radiative energy
                    advectc              advection at cell centers
               radmom{r,  rh}                radiative momentum
                    advecti            advection at cell interfaces
               grid                                   grid equations
          matslv          inverts matrix and solves for corrections
               scopy
               sgemm
                    sgemv
               sgefad or sgefaj
                    sscal     trslbl
                    saxpy
               sgemv
                    saxpy
               sgesld or sgesl(m)j
                    sdot     trslbl     sgeslj
                        sgemv
                        sscal
               sgbfad or bglsdc
                    idamax
                    sscal     sscal
                    saxpy     bglsrd
               sgbsld or bglssl
                    saxpy     bglsrd
                    sdot
```

```
          maxdel                          evaluates biggest correction
          update                                updates all quantities
               update{r,h,rh}
                    eos
                    opac
                    eddfac
                         eddf
                         eddg
     dmax =< conv ?                                 convergency check
          maxdel
          update
               update{r,h,rh}
                    eos
                    opac
                    eddfac
                         eddf
                         eddg
          timstp                           nonlinear time step control
          smax < 2 stol ?
               totnrg
               return
          jback > nback ?
               clzfil
               stop
          reset                      resets quantities to old time level
     jtry > ntry ?                                  convergency check
          stop
archive                             makes a dump of the current model
     jmod = 0 or jstep = jstepe ?
          ier > 0 ?
               stop
writout                    writes the current model in formatted form
writsty                      tracks the evolution of selected quantities
clzfil                                                 closes all files
```

This flow chart is available in the source code as the file `titan.flow`. The reader should note that TITAN's subroutine `matslv` has the option to be executed for a Cray and a non-Cray machine which is manifested in the "or"

option in the flow chart.

Nomenclature: Most of the subroutine names are self-explanatory: `gas` and `rad` standing for gas and radiation, `mom` and `nrg` standing for momentum and energy equation. The suffixes `r, h, rh` indicate that the subroutine refers to either radiation transfer in static media, or pure hydrodynamics, or radiation hydrodynamics. `peq` initiates all variables for a new time step and `update` corrects them at each iteration step until the model is converged. `eos`, `opac`, and `eddfac` compute the equation of states, the opacities, and the Eddington factors either analytically or via tables (which are available throuh `readeos` and `readopac`). `matgen` and `matslv` stand for generating of the Jacobian matrix together with the right hand sides and solving them for the correction vectors (via the Newton-Raphson scheme). `archive` makes unformatted dumps of the model at specified intervals, the output file ends always in ##.dmp. `writout` and `writsty` write a model snap shot at a given time or the evolution of selected quantities into formatted files, ending in ##.out and ##.sty.

D. Comment Cards

Each of these subroutines contains a short header that describes its action or purpose and also gives its relative position in the calling sequence within TITAN. As an illustration may serve the header of the subroutine `gasnrgrh`:

```
      subroutine gasnrgrh
c
c*************************************************************
c     radiating fluid energy equation
c.............................................................
c     calling sequence:  matgen > gasnrgrh > advectc, diffuse
c*************************************************************
c
```

The first text line(s) in the comment always state what the subroutine does. In this case it says that `gasnrgrh` sets up the radiating fluid energy equation. The dotted line is followed by the calling sequence, here: `matgen` calls `gasnrgrh` which in turn calls `advectc`, and `diffuse`. Since the calling sequence is also reflected in the flow chart the user can locate immediately

any subroutine within TITAN.

In all subroutines that set up the physical and grid equations a detailed cross referencing to the documentation in the TITAN Code Reference Manual is provided for in form of comment cards. The general structure of those routines is that first some dummy variables and the advected (diffused) quantities are defined, then the entries into the Jacobian matrix are evaluated in the sequence time derivative, advection, other terms from the equation, followed by the definition of the right hand side itself, next the inner and outer boundary conditions are given, and finally the phantom zones are filled. At each step the equation numbers from the TITAN Code Reference Manual are used to allow the user to keep track of mathematical expressions and their derivation. This is expected to be particularly useful if the user wishes to edit the equations or to add to them or to replace them with other equations.

As an illustration for the structure of the equation subroutines may serve the previous example again, `radnrgrh`. It begins with `define various quantities and their derivatives`, followed by `set interface energy densities for advection`, and `set gas energy diffusion terms`. In the main part we evaluate `all terms except advection and rhs` on the `entire domain, including boundaries`. As documented in the TITAN Code Reference Manual it is divided into the sections `time derivative` (VI.C.1), `flux divergence` (VI.C.3), `work` (VI.C.4), `radiation anisotropy` (VI.C.5), `(artificial) energy diffusion` (VI.C.6), and `(artificial energy dissipation` (VI.C.7). Next we define `advection and right-hand-side` (VI.C.2 and VI.C.8) in the sections `interior zones`, `inner boundary condition (all cases)`, and `outer boundary condition (all cases)`. In the final part we set the `phantom zones` in the sections `eulerian inner boundary`, `lagrangean inner boundary` (XIII.A), and `eulerian outer boundary`, `lagrangean outer boundary` (XIII.B).

IV. A BRIEF TUTORIAL

IV.A. Installing TITAN

In order to build TITAN one needs all its source files. They come in 2 packages: one comprises all the subroutines as given in the flow chart, the other contains various mathematical software in particular from BLAS and LINPACK for the Newton-Raphson solver. In order to obtain the first release versions of TITAN one has to file a registration with the LCA (laboratory for computational astrophysics). The user will be given access privileges to a machine where the code resides. Begin by making the directory `titan` in your local file space. Then signon and change into the directory `/lca/codes/titan`. The source code is located there. Transfer the compressed tar-file `titan.tar.Z` in binary mode from the directory `source`. A protocol of these actions looks like:

```
% mkdir titan ; cd titan
% ftp landrew.ncsa.uiuc.edu
Connected to landrew.ncsa.uiuc.edu.
220 landrew.ncsa.uiuc.edu FTP server
Name (landrew.ncsa.uiuc.edu:username):  titan
331 Password required for titan.
Password:
230 User titan logged in.
ftp> cd /lca/codes/titan
250-Please read the file README
250- it was last modified on Wed Feb 2 17:22:30 1994 - ...
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
.index
README
physica_paper
ref_manual
user_guide
source
```

```
226 Transfer complete.
55 bytes received in 0.0057 seconds (9.4 Kbytes/s)
ftp> cd source
250 CWD command successful.
ftp> binary
200 Type set to I.
ftp> get titan.tar.Z
200 PORT command successful.
150 Binary data connection for titan.tar.Z
226 Binary Transfer complete.
local:  titan.tar.Z remote:  titan.tar.Z
150367 bytes received in 13 seconds (12 Kbytes/s)
ftp> bye
221 Goodbye.
```

Now the source code resides on your local directory `titan`. The individual `FORTRAN` files need to be extracted. This is achieved by the following command:

```
%ls
titan.tar.Z
%zcat titan.tar.Z | tar xvf -
x advectc.f, 2419 bytes, 5 tape blocks
         ⋮
x Makefile, 3158 bytes, 7 tape blocks
```

There is a `Makefile` which installs TITAN on a UNIX system. TITAN can be compiled by issuing:

```
%make
f77 -sun4 -c advectc.f
advectc.f:
       advectc:
         ⋮
f77 -O advectc.o ... -o titan.x
```

TITAN is now ready to run. Its executable is called `titan.x`. TITAN has been compiled on the following platforms:

SUN	Sparc 2	SunOS 4.1.1
SUN	Sparc 10	SunOS 4.1.3
HP	Apollo 735	HP-UX 9.1
SGI	IRIS 4D/35	IRIX 4.0.5
SGI	Indigo 2 E	IRIX 4.0.5

The code needs the Cray vector merge function `cvmgt` (and its relatives). If the user runs TITAN on a Cray these will automatically be provided. Otherwise the user must provide them or load `craypack` which we supply. If the code is run on a Cray machine, the code requires some CAL-coded linear algebra subroutines (`trslbl`, `bglsdc`, `bglsrd`) which are available on Cray only. Again the source code for these routines can be provided where necessary. The code also uses the standard LINPACK routines such as `sgemv`, `sgemm`, etc. This package is generally available on most systems. If not we can provide for the source code. Finally, the code utilizes BLAS (basic linear algebra routines) like `saxpy`, `sdot`, etc. extensively. Most systems will provide these automatically. If they are not present we have the `FORTRAN` code.

TITAN's `Makefile` is written for the Unix operating systems on workstations. For the SUN it works straightforwardly and the command "make" generates the executable. For the HP and SGI the user has to comment out line 42

```
#$(RL) $(BOX) $(LIB)
```

to suppress the "ranlib" command which is automatically issued by the archive command.

B. The Input Files and How to Run a Problem

B.1. The Input Files

TITAN is ready to run all test problems with the help of input files called `titan.in##`, the hash marks referring to the problem number `lprob`. The code assumes the default name `titan.in`. For instance, the input file for Sod's shock tube, `titan.in03`, must therefore be copied into this file before

TITAN can be started. We have constructed the input files in a particular way. The input file `titan.in03` looks like this:

```
   2                                                                    i5
   6    output  SOD.out      new                  i5, 2x, a8, 2x, a8
   7      dump  SOD.dmp      new                      i5, 2(2x, a8)
   3    1                                                         2i5
SOD SHOCK TUBE PROBLEM                                            a80
      1.0       1.0       0.0                                 3f10.0
    0.125       0.1       0.0                                 3f10.0
   1    1   10 1.000e+00                                   3i5, e10.3
      ydl       ypl       yul
      ydr       ypr       yur
c
c-------------------------------------------------- titan.in03
```

The rows are read formatted in the specific format as indicated. The last line always identifies the name of the input file. The problem number `lprob=3` appears in the fourth row. The following row gives a problem header which appears in the output and dump files. Their names are defined in lines following the first one. The rows following the header give particular physical parameters and job control parameters for the current run followed by comments eluding to the meaning of the physical ones.

The number in the first row states how many files (aside from the `tty`) should be opened, in this case two. In the second and third row they are defined by a unit number (`iunit = 2` or `3`), file type (`output` or `dump`), file name (`SOD.out` or `SOD.dmp`), and file status (`new`). In the fourth row the problem number is specified as defined in the discussion of the test problems. It is followed by the initialization flag (`linit = 1`). The problem header is given in the fifth row. The eighth row lists the dump file index and time step index from where the computation should be started (`jdump = 1` and `jstep = 1`), the number of time steps that should be executed (`nstep = 10`), and the maximum time (age) until when the calculation should be evolved (`tmax = 1.0`). The contents of these rows is standardized and necessary for a successful start of `titan.x`. What is listed between the header and the job control line is optional and defined according to the needs of the particular problem. In our example the sixth and seventh row contain sets of three parameters which set the left and right density, pressure, and velocity.

The following table lists the problem number, the default file name, and the problem header for each test:

lprob	name	header
3	SOD	SOD SHOCK TUBE PROBLEM
4	BLST	WOODWARD-COLLELA BLAST WAVE PROBLEM
5	SDOV	TAYLOR-SEDOV BLAST WAVE PROBLEM
6	HEAT	HEATING TOWARDS RADIATIVE EQUILIBRIUM
7	COOL	COOLING FROM RADIATIVE EQUILIBRIUM
8	SUBC	SUBCRITICAL SHOCK PROBLEM
9	SUPC	SUPERCRITICAL SHOCK PROBLEM
10	RLST	RADIATING BLAST WAVE PROBLEM

B.2. Starting and Restarting a Problem

The user has to select a test problem and copy the respective input file into the default one. In our example of Sod's shock tube we type:

```
%cp titan.in03 titan.in
overwrite titan.in?  y
```

If we issued the command `cat titan.in03` we would see the listing of the previous page.

Now we are ready to run TITAN:

```
%titan.x
iter=  2 time= 0.00E+00 d_r/ro(42)= 1.03E-02 rhs(97)= 5.65E-03
iter=  3 time= 1.00E-02 d_r/ro(42)= 9.74E-03 rhs(97)= 3.92E-03
           ⋮
iter= 77 time= 7.50E-01 d_r/ro(10)= 1.61E-08 rhs(73)= 4.50E-11
iter= 78 time= 7.60E-01 d_r/ro(10)= 9.06E-09 rhs(73)= 2.29E-11
model jdump = 1 jstep =  1 archived
```

```
jstep =  2 dtim = 1.00E-05 tim = 1.00E-05 iter = 1
kmax =        73        72        71        70        69
dmax =  3.69E-07  5.65E-08  9.43E-04  1.17E-02  7.08E-04
cmax =  2.84E-03
              ⋮
jstep = 11 dtim = 1.90E-05 boost iteration
kmax =        74        73        72        71        70
dmax = -1.41E-09 -1.42E-10  3.08E-06  4.52E-06  1.66E-06
cmax =  1.40E-04

total energy check
total = 1.3749758E+00 tee = 1.3749758E+00 tes = 0.0000000E+00
  tew = 0.0000000E+00 tel = 0.0000000E+00 teq = 0.0000000E+00

model jdump = 2 jstep = 11 archived
```

TITAN starts out by relaxing the adaptive grid from an equidistant distribution to one that resolves the initial (density and temperature) discontinuity. This it achieves in 78 iteration steps as indicated in the first column. The second one lists a non-physical time which advances by 10^{-2} per iteration step. In the third column the greatest relative temporal change of the radius variable, $\left(r^{n+1} - r^n\right)/r^n$, at the corresponding grid index is given. We see that it improves from an initial 10^{-2} to less than 10^{-8} at which point we say the relaxation scheme to be converged. In the last column we list the magnitude of the right hand side of the grid equation which ideally is zero when solved by the Newton-Raphson scheme. TITAN makes use of this preiteration to redistribute the grid points in order to resolve the desired features for the problems with `lprob` $\in (2, 4, 5, 6)$, *i.e.,* for all hydrodynamic problems and the first radiative transport problem. The grid relaxation procedure is completed with writing the resulting model into the output and dump file.

The model with `jdump = 1` and `jstep = 1` serves as the initial condition from which the partial differential (rather, finite difference) equations evolve. This is documented in the listing by the fact that the first time step begins at `jstep = 2`. The initial time step, here `dtim` $= 10^{-5}$, is set by the user. It should be considerably smaller than any physical time scales of the problem in order to allow the code to compute beyond transients that could stem from the initialization procedure. Each time step takes at least 2 iteration

cycles and at most 10 to converge. The convergence threshold for `dmax` has been set to 10^{-6}. In our example the eighth and ninth time step needed only three and the eleventh time step six iteration cycles.

Each iteration step lists the biggest value of the right hand side of each equation in the row beginning with `dmax =` and the corresponding grid index `kmax` in the row above. The last number `cmax =` gives the maximum cell size as evaluated in `maxdel`. When the code has converged for a particular time step it performs one boost iteration to improve the accuracy of the solution. The convergence control is described in the TITAN Code Reference Manual, chapter XVII. At the end of each time step the cumulative energy conservation and balancing is listed under `total energy check`. This is done as described in the TITAN Code Reference Manual, chapter XI, with the correspondences as follows: total energy `total`, sum of internal, radiation, kinetic, and potential energy `tee` $= \mathcal{E}^{n+1}$, the net energy loss by transport through the boundary surfaces `tes` $= \mathcal{S}^{n+1}$, the work done at the boundary surfaces `tew` $= \mathcal{W}^{n+1}$, the luminosity lost through the boundary surfaces `tel` $= \mathcal{L}^{n+1}$, and the energy dissipation by the artificial viscosity `teq` $= \mathcal{Q}^{n+1}$. These quantities are evaluated in the routine `totnrg`.

TITAN stops the computation either when the desired number of time steps (here: `nstep = 10`) have been executed or when the desired model age (here: `tmax = 1.0`) has been reached. It terminates properly by archiving the last model into the dump file and writing it into the output file while stating the corresponding `jdump` and `jstep`.

Now we want to restart TITAN at `jdump = 2` and `jstep = 11` and run Sod's shock tube until the time equals unity. In order to do that we have to edit the input file `titan.in` and change the third, fourth, and eighth line in the following manner:

```
        ⋮
7     dump  S0D.dmp      old
3   2
        ⋮
2   11  310 1.000e+00
        ⋮
```

First we changed the status of the dump file from `new` to `old` so that TITAN can read the last model from the previous run. By setting `linit = 2` we tell

the code to bypass the initialization procedure and continue straight with the evolution of the equations. In the eighth line we replace the first two integers with the appropriate settings for `jdump` and `jstep` as given at the end of the previous run, and change `nstep` to `310` and set `tmax = 1.0`. Now we type `titan.x` again so see the following listing at our monitor:

```
jstep =  12 dtim = 2.30E-04 tim = 9.77E-04 iter = 1
kmax =        58        57        56        55        54
dmax =  7.06E-05  3.16E-06 -5.85E-03 -2.70E-03  3.41E-02
cmax =  1.47E-01
          ⋮
jstep = 306 dtim = 1.88E-02 tim = 1.01E+00 iter = 5
kmax =        76        75        74        73        72
dmax =  4.35E-06  5.10E-06  7.02E-06  8.66E-06  2.83E-06
cmax =  3.00E-04

jstep = 306 dtim = 1.88E-02 boost iteration
kmax =        76        75        74        73        72
dmax = -9.34E-08 -1.09E-07 -1.32E-07 -1.78E-07 -6.18E-08
cmax =  5.17E-06

total energy check
total = 1.3593973E+00 tee = 1.3593973E+00 tes = 0.0000000E+00
  tew = 0.0000000E+00 tel = 0.0000000E+00 teq = 1.6190176E-19

model jdump = 8 jstep = 306 archived
```

TITAN completed the calculation in 306 time steps and made 8 dumps. The last time step has converged after 5 iteration cycles followed by the boost iteration. Whenever convergence is reached quickly it is achieved quadratically as expected from the Newton-Raphson algorithm. This is always true for a fixed grid. For the adaptive grid this cannot happen all the time especially if the correct grid distribution is not found right away. A look at the total energies at `jstep = 11` and `306` tells that we have conserved it to within one percent. We save the output file to `S0D.out1`. TITAN wrote a model every five time steps in the format as given in `writout`. There are eight columns: the grid index `k`, the radius, grid resolution, density,

artificial viscous stress, gas energy, and gas pressure. At four different times we extract the density profile and show them as snap shots in figure 1 (+'s indicate grid points).

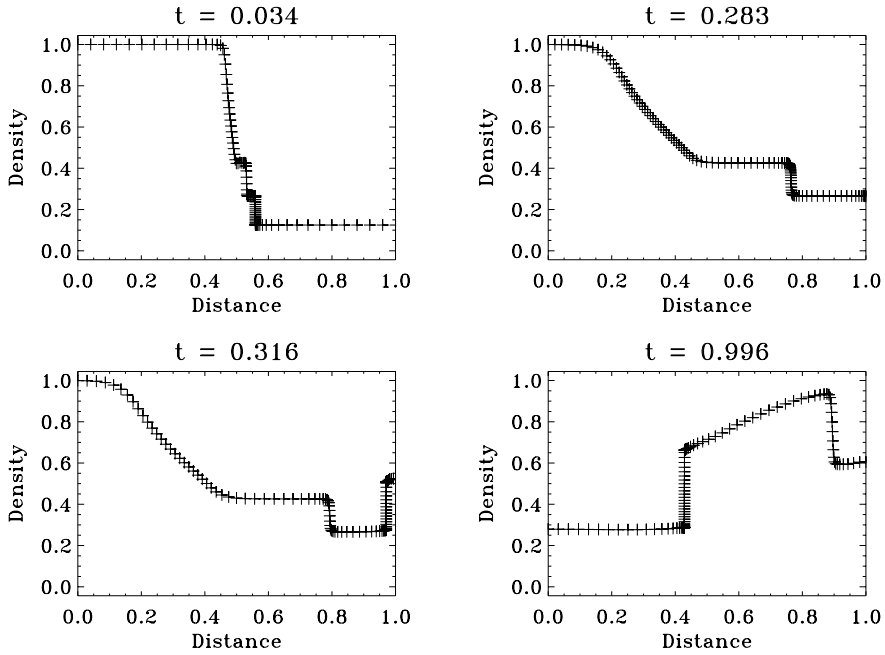


Fig. 1: Density profiles of classical shock tube with adaptive grid

The user can design new shock tube problems by changing the physical parameters in the input file. Let the density jumps be twice as large as in Sod's case. We have to change the first number in the sixth line of `titan.in` to `ydl = 2`. The following figure illustrates the result in form of density profiles at four different times (+'s indicating again individual grid points). Comparing this with figure 1 we see that the new run completes at a later time. The sound speed, proportional to $\sqrt{p/\rho}$, is a factor $\sqrt{2}$ larger thus making the shock to propagate slower by the inverse factor.

B.3. Hints for Running the Test Problems

In the spirit of the previous section the user can run all the other test problems. It is useful to break up the computation into several runs in order to avoid the output files `####.out` to become too big. This is especially

important if one wishes to write out many models, every fifth time step say, to produce space-time diagrams or movies. The dump and output frequencies are set in the `start##` files.

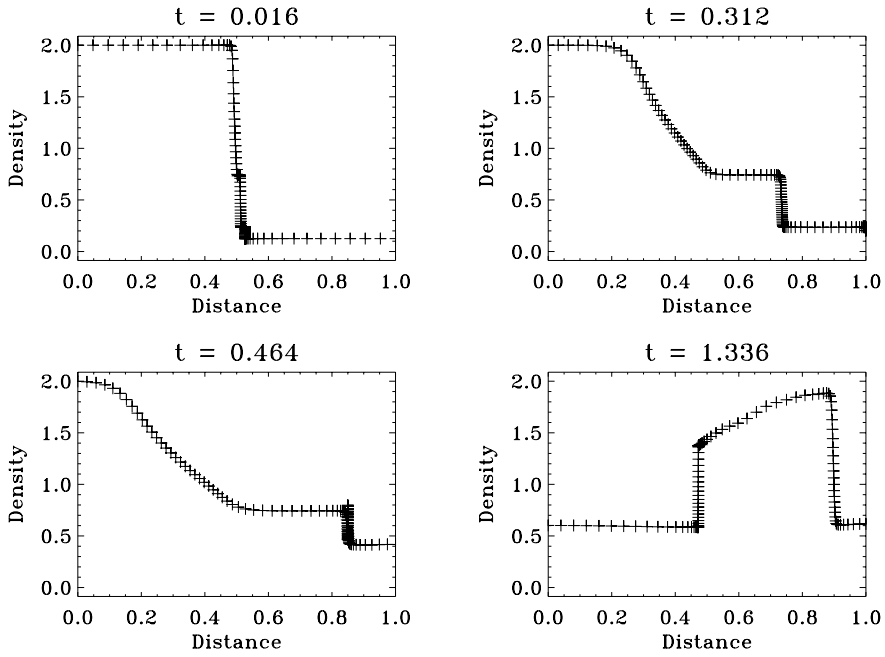


Fig. 2: Density profiles of modified shock tube with adaptive grid

The input files of the problems `lrpob = 8` and `9` have only one difference in the seventh line which states the physical parameters. For the subcritical test we have set `upstn` $= 6 \times 10^5$ whereas it is 2×10^6 for the supercritical test. The start files for both tests are identical and a duplicate exists only for completeness reasons. Either test could be run with the problem number of the other.

Some care must be taken when running the two radiative transfer tests. The heating towards radiative equilibrium needs only `nstep = 150` for the relaxation computation. The results for the different opacities are obtained by varying the last physical parameter `ratio` in the input file. The cooling tests require the final model as an input. Since this is the same for all heating runs the user can take any such dump file called `HEAT.dmp` and copy it into `COOL.dmp`. The cooling test then must start at the last dump (`jdump = 31`) and time step index (`jstep = 151`). It proceeds for a default `nstep`

= 400 and can be restarted by changing the linit flag and the job control parameters jdump and jstep.

C. I/O-Files

TITAN permits access to the nine input/output files. All of the unit numbers and file types are defined in opnfil. Only itty and iin are assigned to non-zero and hardwired to 4 and 5 there. The other unit numbers can be assigned to any different integer. The file names are all eight character long with the default ********. The file type (format a8) is used to identify the unit number and file status with the file name. All files are formatted, with file status being either new or old, except the dump file which is unformatted. The available files are given in the following table:

unit	file type	file name	access
itty	monitor	/dev/tty	
iin	input	titan.in	
iout	output	####.out	sequential
idump	dump	####.dmp	direct
ieos	eos		sequential
iopac	opac		sequential
ihist	history	####.sty	sequential
iinit	initial	####.ini	sequential
idoc	doc	####.doc	sequential

The files for eos and opac define the constitutive relations in tabular form and come either from OP or OPAL. They are opened in opnfil and called by their respective subroutines. In a future release some will be available and carry the file names as follows:

file name	contents
#####.pg	gas pressure
#####.eg	internal energy
#####.pe	electron pressure
#####.cf	Rosseland mean opacity
#####.ke	Planck mean opacity

The user can add further files if so desired by editing opnfil and clzfil and copying the structure of these routines.

D. The Start Routines and How to Modify a Problem

D.1. The Start Routines

All start routines are constructed in the same fashion. At the top are comment cards containing the name, purpose, and some additional information. These are followed by the three major segments:

```
c=============================================================
c     general setup
c=============================================================
          ⋮
c=============================================================
c     generate initial solution
c=============================================================
          ⋮
c=============================================================
c     recover model from dump file
c=============================================================
          ⋮
```

In `general setup` all logical switches and all job control and physical parameters are specified as defined in the TITAN Code Reference Manual, chapter XIV and XV. The general setup has the following layout:

```
c-------------------------------------------------------------
c     read in header and parameters
c-------------------------------------------------------------
          ⋮
c-------------------------------------------------------------
c     geometry:  lgeom = 0 => planar geometry
c-------------------------------------------------------------
          ⋮
c-------------------------------------------------------------
c     gravity:  lgrav = 0 => no gravity
c-------------------------------------------------------------
```

```
           ⋮
c-------------------------------------------------------------
c      radiation transport:  lrad = 0 => no radiation
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c      radiation boundary conditions:
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c      declare equations to be solved:
c-------------------------------------------------------------
⋮
c-------------------------------------------------------------
c      numerics:
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c      set hydro parameters:  lhydr = 1 => hydro
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c      hydrodynamic boundary conditions:
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c      grid specification:  lgrid = 1 => adaptive grid
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c      iteration control & integration control:
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c      output control:
c-------------------------------------------------------------
```

```
           ⋮
c-------------------------------------------------------------
c      eos:  leos = 2 => perfect gas
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c      opacity:  lopac = 4 => constant opacity
c-------------------------------------------------------------
           ⋮
```

First the header and the physical parameters are read from the input file `titan.in`. In `geometry` we specify slab (0), cylindrical (1), or spherical (2) symmetry. The quantity $\mu =$ `lgeom` and its relatives as used in `peq##` and the definition of the equations is set. In `radiation transport` we set the flag `lrad` either on or off. In the first case we need to specify what transfer scheme we would like. That is done by setting `ltran` along with `lam`. We further have to give the number of rays that intersect the core `ncor` for the Eddington factor calculation, *cf.* section X. Finally the radiant boundary conditions have to be defined via `lribc` and `lrobc`. In the case of them being equal to unity the outgoing/incoming specific intensity incident on the inner/outer boundary, `xipl` $= I_L^+$ to `ximr` $= I_R^-$, need to be given. In `declare equations to be solved` the number of grid points in the computational domain `ncell` must be set along with the number of equations at each grid point `neqn`. The include file `titan.par` sets upper limits on these numbers, `ncell` $\leq$ `mgr - 9` and `neqn` $\leq$ `meqn`, as well as for `ncor` $\leq$ `mcor`. Each equation is assigned a dummy index. For adaptive grid computations it is meaningful to set `ir = 1` because the code has to find the coordinate system. In `set hydro parameters` there is a series of hydrodynamical control parameters:

parameter name	symbols	equation
time-centering parameter	theta $= \theta$	(14)
pseudoviscous length	ql0 $= \ell_0$, ql1 $= \ell_1$	(25), FE70
pseudoviscous coefficients	cq1 $= C_1$, cq2 $= C_2$	(25), FE83
pseudoviscous pressure ratio	q0 $= q_0$	AG87
artificial diffusion coeff.	sigd $= \sigma_\rho$, side $= \sigma_e$	C42

Furthermore, the order of the advection scheme has to be defined: for `cadv = 1` we choose Donor-cell and for `cadv = 2` we choose Van Leer advection. When we decide for a specific hydrodynamic boundary condition we have to set:

boundary	switch	constraint
Lagrangean	llibc	uextl $= U_L$
	llobc	uextr $= U_R$, pextr $= \Pi_R$
Eulerian	leibc	phil{0,1,2} $= \Phi_L^{0,1,2}$
	leobc	phir{0,1,2} $= \Phi_R^{0,1,2}$

In `grid specification` the user has to set the grid flag `lgrid`: the choice is among an adaptive grid (1), a fixed Eulerian grid (2), or a fixed Lagrangean grid (3). In the first case a series of grid control parameters need to be specified:

parameter name	symbols	equation
diffusion coefficient	alph $= \alpha$	AG5
time constant	tau $= \tau$	AG4
delay exponent	ibet $= \beta$	AG4
spatial scale	xscale $= R_{scale}$	AG7
ordinate scales	yscl(l) $= (F_{scale})_l$	AG9
grid weights	wt(l) $\in (W_m, W_\rho, W_T, W_E,$ $W_p, W_e, W_\kappa, W_q)$	

The user has a choice between three settings for the scalings of the grid quantities: for `ladx = 1` together with `xscale` $= R_{scale}$ one choses a linear spatial resolution (`AG7`), for `ladx = 2` a logarithmic spatial resolution (`AG8`), for `lady(l) = 1` together with `yscl(l)` $= (F_{scale})_l$ one choses a linear ordinate resolution (`AG9`), for `lady(l) = 2` a logarithmic ordinate resolution (`AG10`), and for `lady(l) = 3` a harmonic ordinate resolution (`AG11`). In general one can reduce the pseudoviscous length scale by at least one order of magnitude in the adaptive grid over the fixed grid case.

For `iteration control & integration control` we recommend the user to stay with the parameters as preset. On the other hand, the user should choose her/his own settings for `output control`: `ndump` is the number of timesteps between dumps of models and to `####.dmp` and `nout` is the number of timesteps between printouts of models to `####.out`. The dump frequency can be chosen generously since it is used to store possible locations for a job

restart. The write frequency depends on whether the user wishes to get a quick overview over the solution or wants to do a detailed study.

Finally there are a few parameters to set in `eos` and `opac`. In general the switch settings `leos = 1` and `lopac = 1` give access to the constitutive relations in tabular form. The user has to provide for her/his own tables until we give out some OP data. For the equation of state one has also the option of an ideal gas (`leos = 2`), in case of which the polytropic exponent `gam` $= \gamma$ and the mean molecular weight `gmu` $= \mu_m$ have to be given, or of the Stellingwerf fit formula for variable star envelopes `leos = 3`, in case of which the chemical abundancies `xabun` $= X$, `yabun` $= Y$, `zabun` $= Z$ have to be set (X, Y, Z = 1 - X - Y are not allowed to be arbitrary numbers). For the opacities one also has the option of a constant law `lopac` ≤ 3, in case of which the ratio between the Rosseland and Planck mean opacity `ratio` $= \zeta$ must be defined (see II.E.3), or of the Stellingwerf fit formula for variable star envelopes `lopac = 2`, in case of which chemical abundancies together with the electron number density `xnen(k)` $= N_e(\rho, T)$ must be specified.

The part `generate initial solution` is executed for the setting `linit = 1`. It is particular to the individual test problem. Accordingly the various sections differ with `lprob`. In them the initial model for the time dependent equations is constructed in a consistent fashion. For our example `start03` this is achieved in the following steps under the heading `initialize variables`: we begin with setting up the `radial grid`, then define the r-dependent physical quantities `density and temperature`, and `mass and velocity`, along with the `phantom zones`, followed by the evaluation of the `eos`, after which we are ready to `initiate old time solution`, to `initialize terms in total energy equation`, to `zero unneeded variables`, to `zero initial errors`, to `initialize time and timestep`, and to `initialize grid equation` optionally for `lgrid = 1`. In the case that we opted for the adaptive grid we have to reset various quantities which is done in `zero unneeded variables` again, `initialize terms in total energy equation`, `mass and velocity`, and finally `initialize time and timestep` again. The initial model is now archived in `write initial model on dump file` and assigned the numbers `jstep = 1` and `jdump = 1`. The user is asked to study the various initialization procedures for the test problems.

The start routines are always concluded by `recover model from dump file`. This section is executed for `linit` $\neq 1$. It begins with reading the problem

header and parameters from the dump file and comparing them with those specified in the input file. If they agree (otherwise resulting in an error message) the new dump number and timestep for the starting model are read in from `titan.in`. It is checked that the specified starting model exists. The model is read in from the dump file and it is verified that the code starts from the correct one. Here is the generic layout:

```
c-------------------------------------------------------------
c     compare model header
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c     read dump number and timestep number of initial model;
c     check that specified starting model exists
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c     read model
c-------------------------------------------------------------
           ⋮
c-------------------------------------------------------------
c     check that we got the right one
c-------------------------------------------------------------
           ⋮
```

D.2. Modifying a Problem

We can change the default job control and physical parameter settings of all test problems in the respective `start##` routines. Again the hash marks refer to the problem number. For illustrative purposes we employ again Sod's shock tube as our example and `start03` is the start routine under consideration. For instance, we might be interested in repeating the complete run with a fixed (Eulerian) grid to compare it with the adaptive result. We edit `start03` and forward to line 180 where the section of the grid specification begins. There we see that the logical switch `lgrid` is used to choose between the adaptive (`1`) and Eulerian (`2`) grid. Since we are now interested in the

last one we have to modify line 206 to `lgrid = 2`. We recompile this new setting by typing `make titan`. Before we can run the Eulerian version of the problem we first have to reset the input file `titan.in`. First the status of the dump file is changed back to `new`, then `linit = 1`, and finally we set `jdump = 1` and `jstep = 1` in the correct formatting. We leave the number of time steps `nstep = 310` because we wish to run the evolution until `tmax = 1`. Now we can carry out the computation:

```
model jdump = 1 jstep =  1 archived

jstep =   2 dtim = 1.00E-05 tim = 1.00E-05 iter = 1
kmax =        99        98        97        96        95
dmax =  0.00E+00  8.01E-16 -5.37E-21  1.95E-17  4.44E-14
cmax =  1.00-300

jstep =   2 dtim = 1.00E-05 boost iteration
kmax =        99        98        97        96        95
dmax =  0.00E+00 -5.34E-17 -5.37E-21  4.16E-20  3.47E-16
cmax =  1.00-300

total energy check
total = 1.3750000E+00 tee = 1.3750000E+00 tes =  0.0000000E+00
  tew = 0.0000000E+00 tel = 0.0000000E+00 teq = -9.5764228E-34
          ⋮
jstep = 311 dtim = 1.89E-03 tim = 3.49E-01 iter = 1
kmax =       100        99        98        97        96
dmax =  0.00E+00 -1.93E-05  1.57E-02  1.01E-01 -6.62E-03
cmax =  1.00-300

jstep = 311 dtim = 1.89E-03 tim = 3.49E-01 iter = 2
kmax =       100        99        98        97        96
dmax =  0.00E+00  4.43E-08  7.39E-05  8.21E-04 -1.87E-03
cmax =  1.00-300

jstep = 311 dtim = 1.89E-03 tim = 3.49E-01 iter = 3
kmax =        85        84        83        82        81
dmax =  0.00E+00 -7.61E-13 -3.12E-10  2.34E-12 -1.83E-11
cmax =  1.00-300
```

```
jstep = 311 dtim = 1.89E-03 boost iteration
kmax =        77        76        75        74        73
dmax =  0.00E+00  5.73E-17  3.26E-17  3.62E-16  4.66E-15
cmax =  1.00-300

total energy check
total = 1.3687542E+00 tee = 1.3687548E+00 tes = 0.0000000E+00
  tew = 0.0000000E+00 tel = 0.0000000E+00 teq = 1.3583605E-22

model jdump = 8 jstep = 311 archived
```

At the beginning TITAN takes only two iteration steps per time step until it increases to about 10^{-3}. The Newton-Raphson then converges strictly quadratically in three to four iterations. We see that `cmax` = 10^{-300} at every time step which is the default setting in `maxdel`. This comes from the fact that the right hand side of the grid equation now is always identical to zero as seen in the first entry of the `dmax` row. The Eulerian version has reached the time 0.349 in `311` time steps. We therefore want to continue the calculation. For this we have to change `titan.in` again in the way as described in section IV.B. but this time we set `nstep = 999` to make sure that the run will be completed. We execute `titan.x` and after another 354 time steps the code reaches `tmax = 1` and terminates with `jdump = 16` and `jstep = 665`. We now can look at `SOD.out` containing the Eulerian models and compare them with `SOD.out1` where we saved the adaptive grid models.

In figure 3 we show the density profiles of this fixed grid computation. The four times were chosen to be close to the ones in figure 1. We observe that the density jumps lack the sharpness of this figure (or figure 2 for that matter). Already at early times the shock, contact discontinuity, and rarefaction fan are not well resolved. About 2 grid points define the shock and about 6 the contact discontinuity. In contrast to that the propagating shock front contains between 10 and 20 grid points and the contact discontinuity about 15 in the adaptive grid computation. The "extra" grid points were drawn from the other regions and from the rarefaction fan in particular, which is represented by considerably fewer grid points in the adaptive grid computation. Our Eulerian calculation proceeds satisfactory (upper right plot) until the shock reflects from the wall. At this instant (lower left plot)

the numerical effects from wall heating are visible. The corresponding spike stays

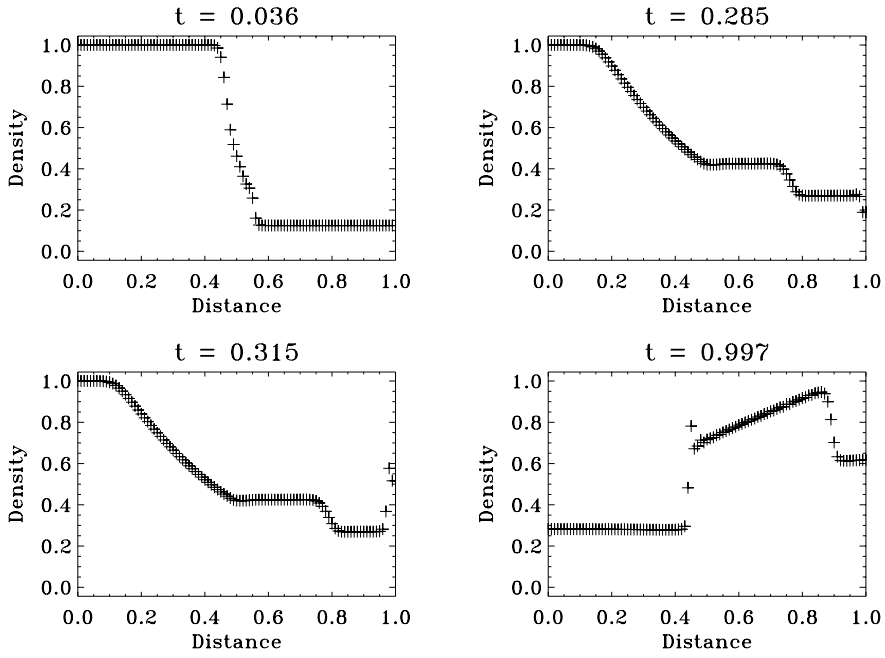


Fig. 3: Density profiles of classical shock tube with fixed grid

through the interaction with the contact discontinuity. The matter could be helped by increasing the coefficients for artificial mass and energy diffusion σ_ρ and σ_e in the lines before the definition of the hydrodynamic boundary conditions.

E. Rules for Writing Your Own Problem

The radiation hydrodynamics equations in TITAN can be modified, augmented, or completely replaced with your own set of equations. Examples for this are: addition of conductive terms for heat transport, a complete $O(v/c)$ formulation of the radiation hydrodynamical equations allowing fluid flows $v \approx 0.1c$, a general relativistic formulation, addition of species equations for multi fluid problems for chemical or nuclear reaction networks, addition of transfer equations for non-photonic energy fluxes, replacement

by models for cosmic ray acceleration or any other set of nonlinear partial differential equations. Corresponding to the changes is the amount of work that goes into adapting TITAN for your problem of choice. In the simplest case,*i.e.,* using the equations as provided, one has be concerned with the initial model only. In the case of adding terms or equations one also needs to do coding of the right hand sides and entries into the Jacobian matrix. The modular nature of TITAN's architecture allows one to achieve this with the smallest effort possible. In general, the code itself should always be refered to for guidance.

E.1. Writing Your Own Start Routine

A new initial model has to be generated in different ways depending on whether it can be formulated analytically via formulae or whether it can be defined by a set of static equations or whether is has to be obtained from mapping a provided model onto TITAN's finite difference scheme. Examples for the first are all test problems, for the second are the stellar structure equations, and for the third is a machine readable model from a colleague.

We have provided for two dummy start routines, `start01` and `start02`, where the user can state her/his problem of choice. They contain a template for the first segment, `general setup`. The user has to fill in the appropriate numbers. For the proper choices one can refer to the examples in the other start routines and the TITAN Code Reference Manual, chapters XIV and XV. The user also has to make a decision on which parameters should be read in. This is done in the section `read in header and parameters`. There the problem header must be read and it is written onto the monitor and output file.

The section where the initial model is archived carrying the heading `write initial model on dump file` and the last segment `recover model from dump file` exist already. The user can optionally write to/read from the dump file as the first record (`irec = 1`) a desired selection of parameters together with the `header`. In the test problems the parameters are those which are read in from the input file. Under the section `compare model header` the parameters are again user defined. Those from the restart model are compared with those from the inital model to check the correctness of the restart model. Suppose that we have read in the variables `xmass`, `xlum`,

xrad, teff, from titan.in. In the start routine we therefore have to define:

```
              ⋮
c=============================================================
c     write initial model on dump file
c=============================================================
c
      if ( idump .le.  0 ) return
c
      irec = 1
      write(idump, rec = irec, iostat = ier)
     .                         header, xmass, xlum, xrad, teff
              ⋮
c=============================================================
c     recover model from dump file
c=============================================================
c
      if ( linit .ne.  1 ) then
c
c-------------------------------------------------------------
c     compare model header
c-------------------------------------------------------------
c
      irec = 1
      read (idump, rec = irec, iostat = ier)
     .                        yheader, ymass, ylum, yrad, yteff
              ⋮
      if ( yheader .ne.  header .or.  ymass .ne.  xmass .or.
     .     ylum    .ne.  xlum   .or.  yrad  .ne.  xrad  .or.
     .     yteff   .ne.  teff      )  then
c
         write(itty, '(a80/a80/1p4e15.7/1p4e15.7)')
     .                  header, yheader,
     .                  xmass, xlum, xrad,  teff,
     .                  ymass, ylum, yrad, yteff,
          stop 'start21'
       end if
```

In the main section, `generate inital solution`, the part of the coding already exists which is necessary for a proper start of the evolution equations. This encompasses the pieces `zero unneeded variables`, `radial grid`, `mass and velocity`, `radiation field`, `phantom zones`, `eos`, `opacity`, `eddington factors`, `initialize terms in total energy equation`, `initialize old time solution`, `initialize time and timestep`, and `zero initial errors`. These sections are intended to follow the user's specification of the inital model including the grid relaxation. Their main purpose is to reset the physical quantities properly after an optional grid relaxation and therefore must be executed in the order given. We recommend to follow this procedure in any case. In `radial grid` and `mass and velocity` some important quantities like the cell volume `dvoln(k)` and cell mass `xmn(k)` are re-evaluated and the velocity field is set to zero, `u(k) = 0`. For radiative problems the `radiation field` is re-evaluated. So are the `eos`, `opacity`, and `eddington factors` which are also needed for the evaluation of the `total energy` of the initial model. In `initialize time and timestep` the beginning timestep `dtime` for the evolution needs to be set. In general we advise to make it a very small number because the code quickly relaxes to the physical time scale.

E.2. Hints For Generating the Initial Model

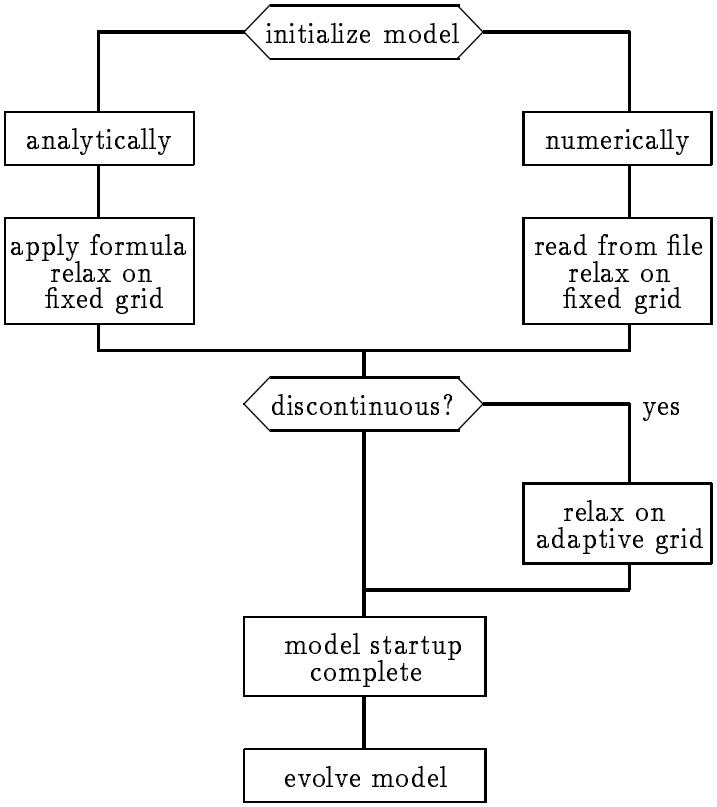
Reference

M. Steffen, *Astron. Astrophys.*, **239,** 443, 1990

Most of the main section of the start routine is devoted to the definition of the initial model. For all practical purposes it can be described either analytically by formulae or by a set of ordinary differential equations or is given in numerical form. All test problems belong to the first category whereby the last one (`lprob = 10`) employs a model relaxation. In general the user should formulate the initial model first on a fixed (Eulerian or Lagrangean) grid and relax it there. In the second step, if necessary or desirable, this solution is mapped onto the adaptive grid and relaxed on that. In all hydrodynamic test cases we had to relax the adaptive grid to resolve the prescribed discontinuities (see V.B). On the other hand, in all radiation hydrodynamic tests we stated the initial model on the Eulerian

grid and used that to start up the evolution. This was possible because the initial models were smooth. Their evolution induced steep features which were resolved through the parabolic nature of the adaptive grid equation.

The following flow diagram summarizes the initialization procedure:



Some recommendations on how to proceed if the initial model is given

"by formulae":

Define either the radius or the mass variable equidistantly. Express as many physical quantities as possible in terms of the radius or mass, and all others as functions of the first. As an example may serve $\rho(r) = \rho_0 \left(r/r_0\right)^7$ which translates immediately into $\rho_k^n = \rho_0 \left(r_k^n/r_0\right)^7$. The user should pay close attention to the functional dependencies if the same formula occurs in the evolution equations. For instance, the mass in a shell is written in the form $\Delta m = 4\pi\rho\Delta \left(r^{\mu+1}/\mu + 1\right)$,

cf. chapter III.B of the TITAN Code Reference Manual, which has a different truncation error than $\Delta m = 4\pi\rho r^{\mu}\Delta(r)$! Coding the proper formulae constitutes already a good starting model for a fixed grid computation. If it contains sharp features then the adaptive mesh should be allowed to resolve them in a relaxation procedure with a selection of grid parameters. As described in the previous section (E.1) some important quantities need to be re-evaluated on the new non-uniform grid distribution. The model is ready to evolve with that set of grid parameters.

"by a set of equations":

Define either the radius or the mass variable equidistantly. Write down the equations with one of them as the independent variable. In the case of ordinary differential equations it is crucial to formulate them in a way that is consistent with the formulation of the partial differential equations. *In praxi* this means that one uses the identical finite difference approximations. The resulting expressions might look somewhat awkward but they admit exactly the same truncation errors as the evolution equations. Being extremely attentive to this point saves a lot of debugging hassle later on. As an example may serve the radiative flux in diffusion approximation. Its formula is given in equation (7), see chapter II.A. Our finite difference approximation, as found in chapter IX.A of the TITAN Code Reference Manual, is derived from

$$F = -\frac{c}{3}\frac{1}{\rho\chi_F}\frac{\partial E}{\partial\left(\frac{r^{\mu+1}}{\mu+1}\right)} \tag{54}$$

which is analytically equivalent to (7) but not numerically! Our rule of thumb in formulating the finite difference expressions is to leave them as compact as possible. Relax or integrate the equations to find the fixed grid solution of the initial model. The integration is best carried out with the Newton-Raphson scheme. If one needs to resolve physical structures then the adaptive grid has to be relaxed. This means that now the independent variable is viewed as a dependent one and all quantities become functions of the grid index k. The grid equation with a set of grid parameters must be solved together with the given equations. As soon as a desired grid distribution is obtained the initial model can serve as an input into the evolution equations.

"in numerical form":

The physical quantities are given in dependency on the grid index. They must be transformed into the set of fundamental variables (r, m, ρ, u, F, T, E) that is used by TITAN. Further, they must be mapped onto the finite difference approximations employed by TITAN (see the examples above). Finally, the need to map between a different amount of grid points in the computational domain can arise. For this purpose we provide for an interpolation routine which utilized monotonized piecewise cubic polynomials according to M. Steffen. If the given numerical description of the initial model is steady-state then the mapped model should be relaxed first on a fixed grid and then, if necessary, together with the grid equation. If the given numerical description of the initial model is a snap shot from an evolution process, *e.g.,* containing a particular velocity field, then a relaxation of the mapped model might not be possible and it has to be fed into TITAN's evolution equations "as is". In this case one should try to find the best possible mapping satisfying the above criteria. A test for the quality of the mapping is how well the model evolves physically as well as numerically.

E.3. Relaxing the Adaptive Grid

There are several ways to relax the adaptive grid if the initial model contains sharp features or discontinuities. Here we shall describe a method that works reliable and well. This relaxation scheme is employed in our test problems.

The basic principle is as follows: Decide by which physical quantities the discontinuity can be described most easily. They will become part of our selection of the grid parameters. Express the discontinuity in terms of them as a function of the radius or mass variable. Write this down in form of one or several steady-state equations, possibly by introducing dummy quantities. These equations are to be coupled with the grid equation. Since it possesses a time derivative one can use the standard Newton-Raphson iteration to perform a relaxation of the adaptive grid. Once the grid distribution is satisfactory the relaxation is terminated.

We illustrate this method using Sod's shock tube as an example. The complete relaxation scheme has been coded outside of `start03` in the file

gridinit. The first routine there, grid3nit, is called by start03. It contains the parts:

```
c=============================================================
c     modify original code parameters
c=============================================================
              ⋮
c=============================================================
c     newton-raphson iteration
c=============================================================
              ⋮
c=============================================================
c     restore original code parameters
c=============================================================
              ⋮
```

In the first part we select temperature and density to express the initial discontinuity in terms of the radius variable. Together with the grid equation we therefore want to iterate three equations. Since the grid equation is implicit we have to solve for r(k) via a matrix inversion. TITAN provides already for a complete linear solution scheme which we can employ here as well. To do this we set neqn = 3 and assign three equation indices ir = 1, id = 2, it = 3. Then we choose as grid parameters the density, pressure, and internal energy in logarithmic scaling with weights equal unity. The grid diffusion coefficient α is left as set in start03 but the grid time constant τ is arbitrarily set to tau = 10^{-2}. So is the time step, dtime =10^{-2}, which serves only as a counter in the relaxation method. Their ratio $\tau/\delta t$ thus becomes equal to unity which forces the grid equation to redistribute the zones. In the last part the particular selection of code parameters are nullified by resetting them to the original ones.

In the middle part, newton-raphson iteration, the grid relaxation with our selection of grid parameters is carried out. It follows the general solution procedure as outlined in step. Under the heading calculate derivatives we describe the density and temperature discontinuity by a very steep, yet smooth function. We choose the tanh-function because it allows us to control the location and the slope of the step. Having the left and right density

prescribed as ydl and ydr we can express the density step as a function of the radius in one compact formula:

$$\rho(r) = \mathtt{ydr} + \frac{1}{2}(\mathtt{ydl} - \mathtt{ydr})\left[1 - \tanh\left(\frac{r - \mathtt{rc}}{\mathtt{rw}}\right)\right] \tag{55}$$

The quantities rc and rw give the location and the steepness of the step. From the formula (55) we see that the density approaches a step function with the discontinuity at $r = \mathtt{rc}$ for $\mathtt{rw} \to \infty$. We use an artificial viscosity to describe the emerging shock. It is therefore meaningful to relate rw to the shock width which is given by the pseudoviscous length; in start03 we simply set $\mathtt{rw} = \ell_0$. The temperature step is also expressed in the form (55). Furthermore, (55) admits a straightforward derivative through the cosh-function:

$$\frac{d\rho(r)}{dr} = -\frac{1}{2}(\mathtt{ydl} - \mathtt{ydr})\frac{1}{\mathtt{rw}}\cosh^{-2}\left(\frac{r - \mathtt{rc}}{\mathtt{rw}}\right) \tag{56}$$

The next section, initialize all matrices to zero, the relaxation iteraion begins with resetting all entries of the Jacobian matrix and the auxiliary solution arrays to zero. This is identical to the beginning of matgen. This is followed by defining two equations through their right hand sides, *e.g.,* rhs(k) = dn(k) - ρ(rn(k)), and elements of the Jacobian matrix. Then the grid equation is called. The system of three equations is solved on the computational domain by matslv. We use only the radius array from the solution, contained in rhs(k), and correction vector, δr. We evaluate its largest element, called dmax, and compute the largest grid-spacing, called cmax. If dmax or cmax exceed the preset values dtol or ctol we scale δr by the factor min(1., dtol/dmax, ctol/cmax). The new radial distribution is computed from the solution vector, *i.e.,* rhs(k), and the old radial distribution rn(k). The dummy time step is advanced by dtime and our two equations are re-evaluated. We employ as the convergence criterion the relative change in the grid distribution between two relaxation steps: $\max_k \left(r_k^{n+1}/r_k^n - 1\right) < 10^{-8}$. By non-fulfillment the relaxation cycle is continued with initialize all matrices to zero.

This grid relaxation appears to be somewhat cumbersome because it necessitates at least one more equation in addition to the grid equation. All we care about is to redistribute the grid points to resolve the discontinuity. Thus we actually only have to relax the grid equation by itself. It, however, needs input from the discontinuity via a the physical variables, such

as density and temperature and their spatial derivatives. The description of the discontinuity, however, depends on the specific problem that one wishes to investigate. Iterating only the grid equation thus means that one has to code it differently for each problem. We thought that this is way too complicated and prone to admit many mistakes. In order to avoid this we decided to keep the grid equation as programmed and add a description of the discontinuity in equation form. Thus we only have to specify that for a given problem while having the solution procedure already at hand.

E.4. Coding Your Own Equations

In principle, any number of new terms, and any number of new equations can be implemented in TITAN. An example for the first case is the addition of electronic heat conduction, and for the latter is the addition of an equation for the turbulent kinetic energy. In both cases the user has to define new global quantities and list them as a common block in the include file `titan.com`. Since adding a new term is a special case of adding a new equation we shall discuss some rules that need to be observed in implementing the latter.

Suppose that we wish to add an equation for the turbulent kinetic energy that is of the form

$$\rho D_t(\varepsilon) + \frac{\partial(r^\mu \Psi)}{r^\mu \partial r} + \pi \frac{\partial(r^\mu u)}{r^\mu \partial r} + \rho\Big[K - \Sigma\Big] = 0 \tag{57}$$

with the notation ε for the turbulent kinetic energy, Ψ for the turbulent flux, π for the turbulent pressure, and Σ and K for sink and source terms. Before we can code equation (57) we have to transform it to arbitrary coordinates in the spirit of chapter II B. The adaptive mesh version reads:

$$\frac{d}{dt}\int_V \varepsilon\ \rho dV - \oint_{\partial V} \varepsilon\rho u_{rel}\ dS + \int_V d[r^\mu\ \Psi] + \int_V \pi\ d[r^\mu \rho\ u]$$
$$+ \int_V \Big[K - \Sigma\Big]\ \rho dV = 0 \tag{58}$$

Expressed in symbolic notation:

$$\frac{d}{dt}(\varepsilon\rho\Delta V) - \Delta\left[\frac{dm}{dt}\ \varepsilon - r^\mu\Psi\right] + \pi\Delta(r^\mu u) = \Big[\Sigma - K\Big]\ \rho\Delta V \tag{59}$$

The translation from this to the actual finite difference representation is completely analogous to that of any equation (17)–(21) with the help of the TITAN Code Reference Manual. Let the FORTRAN names of the new quantities be et $= \varepsilon$, ft $= \Psi$, pt $= \pi$, source $= \Sigma$, and sink $= K$.

The new fundamental quantity is ε since we determine it from equation (59). All the other quantities are algebraic combinations of ε with the existing quantities ρ, e, E, u, F, r, and the auxiliary quantity m. In our example we have the formulae $\pi = \frac{2}{3}\rho\varepsilon$ and $\Psi = -\Lambda\frac{\partial(\varepsilon^{3/2})}{\partial r}$. We won't need the definitions of the other quantities for the purpose of our discussion. Thus, instead of seven equations we now deal with eight, and correspondingly we have eight fundamental quantities that are iterated in the Newton-Raphson algorithm. Those and their related quantities are specified in titan.com. Most of them appear in the first columns of the common blocks with the names common /mid/, common /new/, and common /old/, referring to the time-centered quantities (x), the quantities at the new (x^{n+1}) and old (x^n) time level, respectively. In our case we have listed the time-centered et in the row below er and above fr (in line 130 to be precise). Similarly, we add the turbulent kinetic energy at the new time level etn $= \varepsilon^{n+1}$ to common /new/, and eto $= \varepsilon^n$ to common /old/ (lines 145 and 157). Since ε is also advected we denote the advected et by etb. Turbulent kinetic energy is a scalar quantity and therefore cell-centered. The appropriate advection routine is advectc. It employs the advected quantity at both time levels and its monotonized slopes at both time level, see chapter IV, equations $C22$–$C36$, in the TITAN Code Reference Manual. We have to provide for storage space for the monotonized slopes of the advected ε which we call etsn and etso. We enter them also in the common blocks /new/ and /old/. Furthermore, we list ft, pt, sink, and source in /mid/. The new common block /mid/ now reads:

```
      common /mid/
     .       r     (mgr), rmu   (mgr), rmup1 (mgr), rmum1 (mgr),
     .       dvol  (mgr), xm    (mgr), d     (mgr), db    (mgr),
     .       u     (mgr), ub    (mgr), unom  (mgr), as    (mgr),
     .       eg    (mgr), egb   (mgr), pg    (mgr), xne   (mgr),
     .       er    (mgr), erb   (mgr), egr   (mgr), egrb  (mgr),
     .       et    (mgr), etb   (mgr), egt   (mgr), egtb  (mgr),
     .       fr    (mgr), frb   (mgr), egrt  (mgr), egrtb (mgr),
```

```
.      frnom (mgr), avchi (mgr), pt    (mgr), ambda (mgr),
.      ft    (mgr), fc    (mgr), sink  (mgr), source(mgr),
.      ftnom (mgr), fcnom (mgr), floor (mgr), acous (mgr),
.      chif  (mgr), xke   (mgr), xkp   (mgr), plf   (mgr),
.      beta  (mgr), dels  (mgr), cp    (mgr), fedd  (mgr),
.      t     (mgr), geddl      , geddr
```

It might be necessary to dedicate a separate common block for the definition of turbulent quantities. In our case a candidate for this is the turbulent flux `ft` $= \Psi$ since we need this quantity in the turbulent energy routine, in the internal energy routine which contains `egrt` $= e + E/\rho + \varepsilon$, and in the total energy routine `totnrg`. In `titan.com` we demonstrate this in `common /turb1/` which contains all derivatives of `ft` with respect to all fundamental quantities at various grid points, *e.g.*, `dftdld00` $= \partial$`ft`$/\partial \ln($`dn`$)|_k$ or `dftdlrp1` $= \partial$`ft`$/\partial \ln($`rn`$)|_{k+1}$. Finally we need to provide for additional equation indices `ic` and `jc` in `common /index/`.

Now we are ready to code the turbulent energy equation in the new file `turnrg`. An easy way of getting most terms correctly is to copy the contents of `gasnrgh`. We begin by changing all equation indices `it` $\rightarrow$ `ic`, `jt` $\rightarrow$ `jc`. Then we code (59) in the section `right hand side` in the following manner:

```
rhs(ic,k) =
.                        ( dn(k) * etn(k) * dvoln(k)
.                        - do(k) * eto(k) * dvolo(k) )/dtime
.                                    - dmdt(k+1) * etb(k+1)
.                                    + dmdt(k  ) * etb(k  )
.                                     + rmu(k+1) *  ft(k+1)
.                                     - rmu(k  ) *  ft(k  )
.                           + pt(k) * ( rmu(k+1) *   u(k+1)
.                                     - rmu(k  ) *   u(k  ))
.            + ( sink(k) - source(k) ) * d(k)  * dvol(k  )
```

In the section `set interface energy densities for advection` we have to rename the character string `eg` $\rightarrow$ `et`. In the section `time derivative` we repeat the renaming and delete the derivative with respect to the density, *i.e.*, `e00(ic,jd,k)`, since $\partial\varepsilon/\partial\rho = 0$. We proceed analogously in the section `work`: `pg` $\rightarrow$ `pt`. We can keep its derivatives `dptdldn` $= \partial$`pt`$/\partial \ln($`dn`$)$ and

dptdlcn = ∂pt/∂ln(et) if we define them beforehand. It is advisable to include them in the common block common /turb1/ so that they can be referred to in the internal energy equation. Now we have to compute the Jacobian matrix entries from the flux divergence. We could copy the section flux divergence from the file radnrgr and change ie $\rightarrow$ ic. Since ft is a derived variable we have to right down all of its derivatives with respect to all fundamental quantities at all grid points as listed in titan.com under common /turb1/. Again these are best evaluated beforehand at the beginning of turnrg. We conclude the main programming part with the sinks and sources. The section source-sink terms in radnrgr is a good guide to follow. In the section interior zones of our file turnrg we already have defined the right hand side. The advection of et is coded by changing egb into etb, the other derivatives are correct. We conclude the coding of turnrg by setting all inner and outer boundary conditions and the phantom zones.

If the user introduces a new fundamental quantity then it and its relatives must be listed in the following routines:

start01: Add etn, etsn to the write statement to the dump file with record number irec = jdump + 2 in the section write initial model on dump file and in the corresponding read statement in the section read model. Assign the additional equation index ic = 8 and the numbers of equations neqn = 8 in the section declare equations to be solved. Finally we have to define the flag ltur = 1 to admit the turbulence equation.

archive: Add etn, etsn to the write statement to the dump file with record number irec = jdump + 2.

reset: Add etn, etsn to the loop do 2.

totnrg: Add et(k) = 0.5 (etn(k) + eto(k)) to the loop do 2. The formula for ft(k) is already available through the common block /mid/ and need not to be re-computed in loop do 1. The turbulent flux needs also to be included in the computation of the luminosity.

The following two new routines should be written:

peqtrh: Copy the routine peqrh which prepares the radiation hydrodynamics equations for the Newton-Raphson iteration into this new file. Add eto, et, etn, etso to the loop do 2. Define inner and outer

boundary conditions for etn. Evaluate the time-centered quantity et = θ etn + (1 - θ) eto in loop do 12.

updatetrh: Copy the routine updaterh which updates the solution of the radiation hydrodynamics equations after a Newton-Raphson iteration into this new file. Add the line etn(k) = (1 + rhs(ic,k)) etn(k) to the loop do 12. Just like in peqtrh the inner and outer boundary conditions for etn must be defined and the time-centered et must be evaluated.

Before we can test the new equation we have to call it from matgen along with the other equations:

```
   if (lhydr.eq.1 .and. lrad.eq.1 .and. ltur.ge.1) then
   .          call mass
   .          call contin
   .          call viscous
   .          call turnrg
   .          call gasmomtrh
   .          call gasnrgtrh
   .          call radnrgrh
   .          call radmomrh
   .          call grid
   .          return
   end if
```

The internal energy equation gasnrgtrh is the routine gasnrgrh with all the new terms from turnrg. The turbulent sink-source term is omitted just like the turbulent one because we regard the sum egrt in this file. Because of the turbulent pressure term we also get a corresponding pt gradient in the momentum equation which is included in gasmomtrh. The routines peqtrh and updatetrh have to be called from peq and update with the logic as shown in the example above. The implementation of the new equation (57) is now complete.

V. TEST PROBLEMS

A. Brief Description

References

G. A. Sod, *J. Comput. Phys.,* **27,** 1, 1978

P. R. Woodward and P. Collela, *J. Comput. Phys.,* **54,** 115, 1983

L. D. Landau and E. M. Lifshitz, "Fluid Mechanics", Pergamon Press, London, 1959

L. Ensman, Ph. D. Thesis, University of California at Santa Cruz, 1991

Ya. B. Zel'dovich & Yu. P. Raizer, "Physics of Shock Waves and High-Temperature Hydrodynamic Phenomena", Academic Press, New York, 1966

D. Mihalas and B. W. Mihalas, "Foundations of Radiation Hydrodynamics", Oxford University Press, Oxford, 1984

TITAN comes with a suite of test problems that were designed to illustrate how versatile and powerful the code is. The test problems divide naturally into three categories: hydrodynamical tests, radiation transfer tests in static media, and radiation hydrodynamical tests.

In the first class fall Sod's shock tube in which a dense hot gas expands into a dilute cold one, Woodward's problem, in which pressure jumps generate two interacting shocks, and the Sedov-Taylor problem, in which an inital energy deposition creates a self-similar spherical blast wave.

In the second class fall radiative heating problems, in which a light source heats a gas until it settles into an equilibrium, and radiative cooling problems, in which the light source of the previous situation is being turned off. The radiative problems allow the solutions to be obtained through equilibrium and non-equilibrium diffusion as well as through full transport with variable Eddington factors.

In the third class fall the sub- and supercritical shocks, in which a moving piston generates a non-adiabatic shock with radiative features, and a radiative blast wave, in which an energy deposition leads to an explosion with shock fronts and radiative waves.

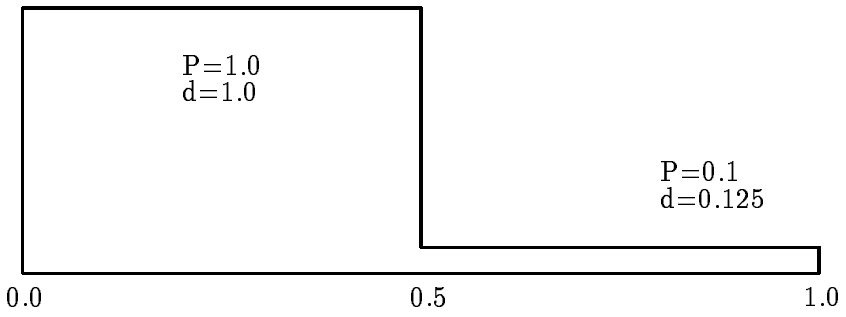
In TITAN the flag `lprob` identifies a particular test problem and selects the appropriate start routine as well as the correct formats of the output. This flag is assigned in the input file as described in section IV.B.

B. Hydrodynamical Test Problems

For the hydrodynamical tests an ideal gas $p = (\gamma - 1)\,\rho e$ is considered to be initally at rest within a cylinder of unit length. The adiabatic index is chosen to be $\gamma = 1.4$.

B.1. Sod's Shock Tube

The first test problem is the classical problem in which a dense hot ideal gas occupying the left half of a cylinder penetrates into a colder dilute gas residing in its right half. It has already been shown by Riemann that a shock wave, a contact discontinuity, and a rarefaction wave are formed. A well-known analyses of corresponding numerical simulations has been carried out by Sod. In his setup the gas is initially separated into two parts by an eightfold density and a tenfold pressure jump.



The computation is carried out with time centering parameter $\theta = 0.55$ on 100 grid points. An artificial mass and energy diffusion of $\sigma_\rho = \sigma_e = 10^{-5}$ is used in order to treat the heating at the right wall from the shock reflection.

The fixed grid (Eulerian) case is run with a linear artificial viscosity $\ell_0 = 10^{-3}$, *i.e.,* one tenth of the grid spacing. Using typically four iterations per time step the time $t = 1$ was reached after 670 time steps. For the

adaptive grid case a $\ell_0 = 10^{-4}$ was chosen. As the grid parameters f_k^n 's were employed the density, pressure and energy in logarithmic scaling together with $\alpha = 1.5$, $\tau = 10^{-15}$. TITAN needed about eight iterations per time step and reached the time $t = 1$ after only 230 time steps. Since one additional equation had to be solved the actual CPU time for completion of the run was approximately the same in both the adaptive and fixed grid version.

The first figure shows snap shots of the density profile on the adaptive grid at four different times $t \in (0.034, 0.283, 0.316, 0.996)$ from top left to bottom right.

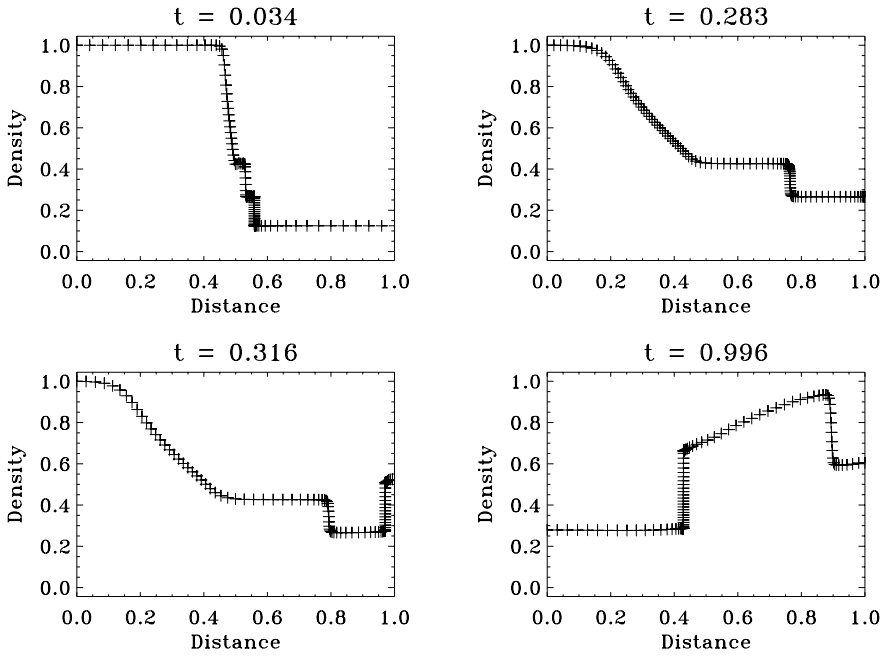


Fig. 4: Density profiles at different times (+'s indicate grid points)

The top left plot shows the formation of Riemann's solution: the right-most step being the shock front, the next step indicating the contact discontinuity, followed by the rarefaction fan. In the top right figure the shock rams into the wall while being followed by the discontinuity at a slower speed. The shock has reflected (bottom left plot) and moves leftward towards to discontinuity. In the last drawing the shock has moved beyond the middle of the shock tube and drags the rarefaction wave behind it which connects it to the contact discontinuity.

One can obtain a complete overview of the actions of all nonlinear waves by drawing a space-time diagram of the density. This is done by stacking many of the its profiles and plotting the contour lines of the result:

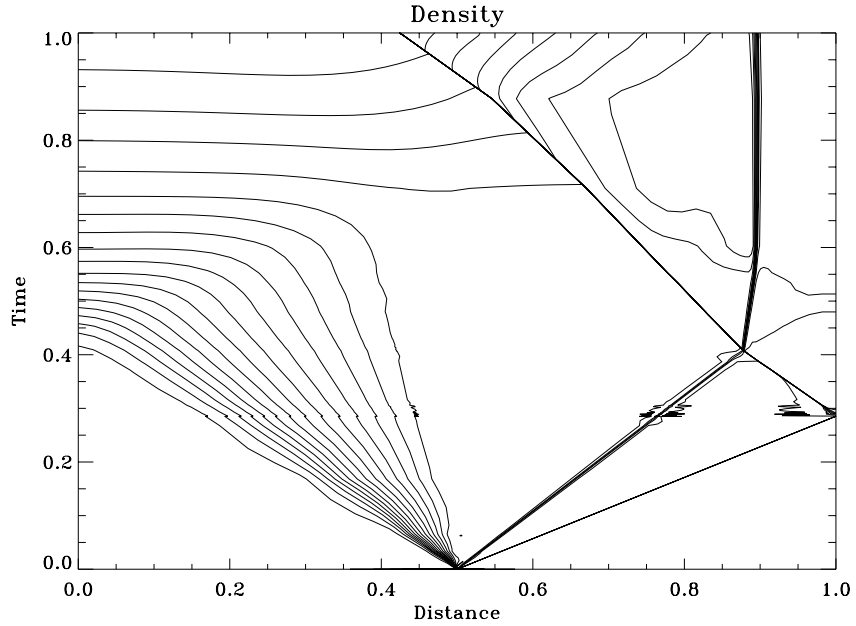


Fig. 5: Density contours

For times $t < 0.3$ there are three nonlinear waves: the right-traveling shock front and contact discontinuity and the left-traveling rarefaction waves. The first two can be identified by their respective speeds which can be read off the slope of the right-going lines. The shock speed is approximately $+1.77$ while the speed of the discontinuity is about $+1.0$. At $t = 0.283$ the shock reflects from the wall and travels backwards with a slower speed of about -1.2. It collides with the contact discontinuity around $t = 0.41$. Both waves interact and penetrate each other. The shock and the contact discontinuity are slowed down considerably and a second right-traveling shock emerges as a new wave. It reflects from the wall around $t = 0.5$ and shortly thereafter interacts with the contact discontinuity. As a result the latter one is slowed down even further to an almost halt but the computation does not tell what happens further. The contact discontinuity and the first, left-propagating shock are connected by a rarefaction fan after their interaction. The primary

left traveling rarefaction waves reflect also about $t = 0.5$, and when they reach the first shock, it starts to speed up.

The primary shock front and the contact discontinuity appear as sharp lines in the space-time diagram above. This means that they are well resolved by many grid points which is possible through the action of the adaptive grid. The following space-time diagram shows the motion of every fourth grid point during the calculation:

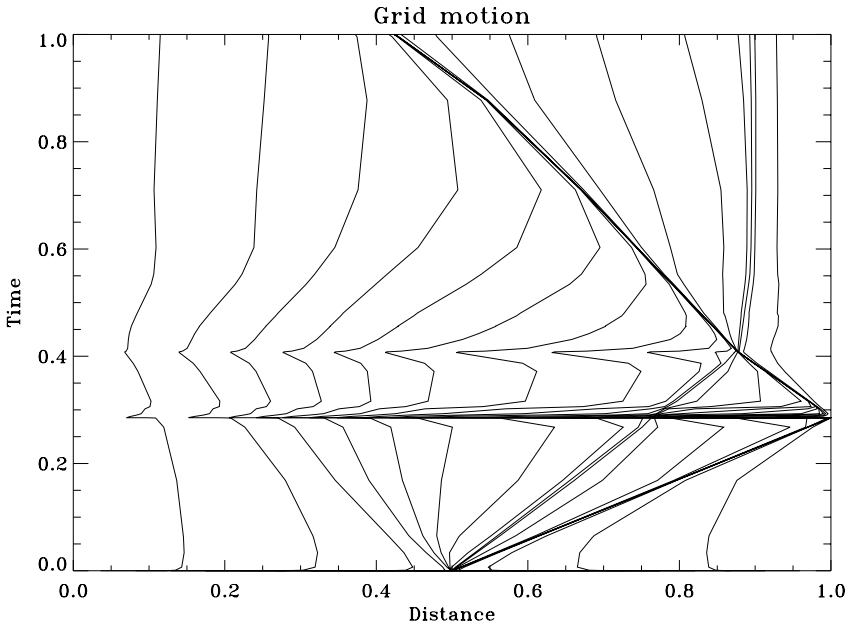


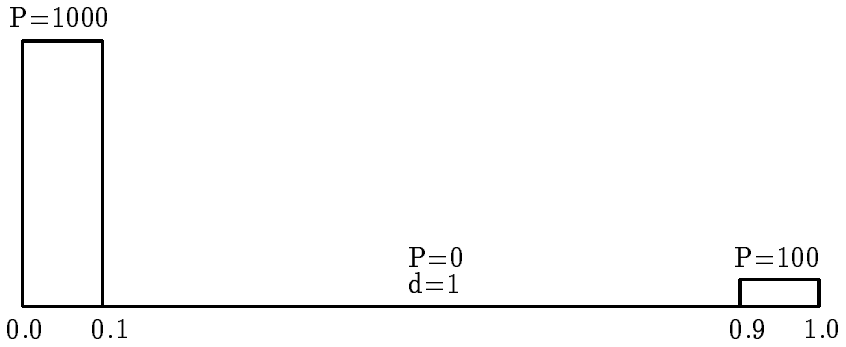
Fig. 6: Action of the adaptive grid

As soon as the nonlinear waves form the adaptive grid detects the two right-propagating waves. This is possible through the choice of grid parameters: density, pressure, and energy. The grid points zoom into the discontinuities on a very short time scale until they are numerically resolved and track them. When the shock reflects at the right wall the adaptive grid tries to relax (because momentarily there is no shock) and begins to move the grid points towards a more uniform (*i.e.*, Eulerian) distribution. This is, however, not done instantanteously because the mesh remembers the previous configuration via the grid time scale τ. When the shock moves away from the wall the adaptive grid recognizes it again and re-distributes the grid points so as to resolve it. The shock and the contact discontinuities remain

resolved and tracked until the run terminates. By contrast, the rarefaction waves and the secondary shock are not detected because they do not show any strong variations in the grid parameters. This situation could be improved if they were chosen differently.

B.2. Woodward's Interacting Blast Waves

Woodward introduced a planar problem of two interacting blast waves. The cylinder is divided into three sections with three different pressure regimes.



In the left part the gas pressure is $P(0 \leq r \leq \frac{1}{10}) = 1000$, in the middle part $P(\frac{1}{10} < r < \frac{9}{10}) = 0$, and in the right part $P(\frac{9}{10} \leq r \leq 1) = 100$. The density is initially constant $\rho = 1$ across the domain $0 \leq r \leq 1$. As grid parameters we adopted the density, pressure and energy in logarithmic scaling, together with $\alpha = 1.5$, $\tau = 10^{-6}$. This time the code is run with 200 grid points. The artificial viscosity is kept linear at $\ell_0 = 10^{-4}$.

The large pressure steps at both ends of the cylinder create two shocks which travel towards the middle. The graph to the right (fig. 7) shows their density profiles at an early stage (+'s marking individual grid points). One can see that the left front is much stronger than the right one. Further, rarefaction waves begin to appear in form of the density dips, which connect to the unit density near the walls.

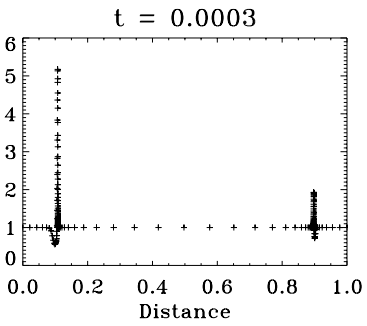


Fig. 7: Density profile

In the following figure, density profiles at four different times $t \in (1.57, 2.94, 3.85, 5.00) \times 10^{-2}$ are plotted. The top left graph shows the approach of the two shocks. Note in particular that the left one is much faster than the right one. The fast shock has a resolution of 10^5 and the slow shock one of several 10^4. The density between the shocks is still at the $\rho = 1$ plateau thus indicating that they move independently. The shocks are followed by contact discontinuities, both being resolved by 10^3. They connect to the rarefaction fans which stretch to the walls. The high resolution of the discontinuities leaves only a crude representation of the rarefaction fans with resolution of about 20. It is by choice that the adaptive grid is allowed to zone only into the shocks and has to neglect the rarefaction waves. At time $t = 0.028$ the shocks collide near position 0.69 and penetrate each other. When they separate a new third contact discontinuity forms in between them (lower right plot). At $t = 0.05$ the fast shock has reflected from the right wall but has not yet run into the third contact discontinuity.

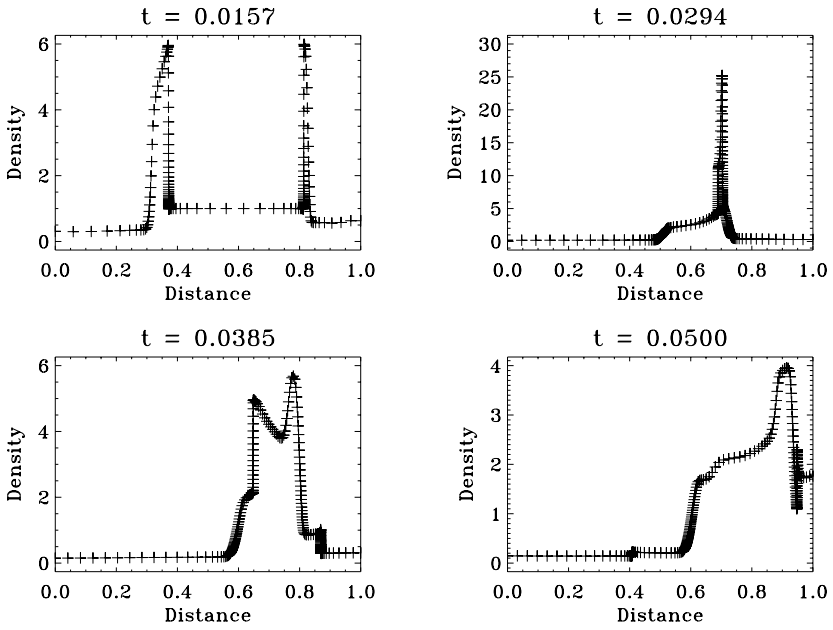


Fig. 8: Shock profiles at different times (+'s indicate grid points)

A complete overview of the actions of all nonlinear waves can be obtained by investigating the space-time diagram of the density. The respective contours in logarithmic scale are drawn in figure 9. The shock fronts (due to

their high resolutions) appear as thick lines, the contact discontinuities as broad lines, and the rarefaction waves as an ensemble of single contour lines.

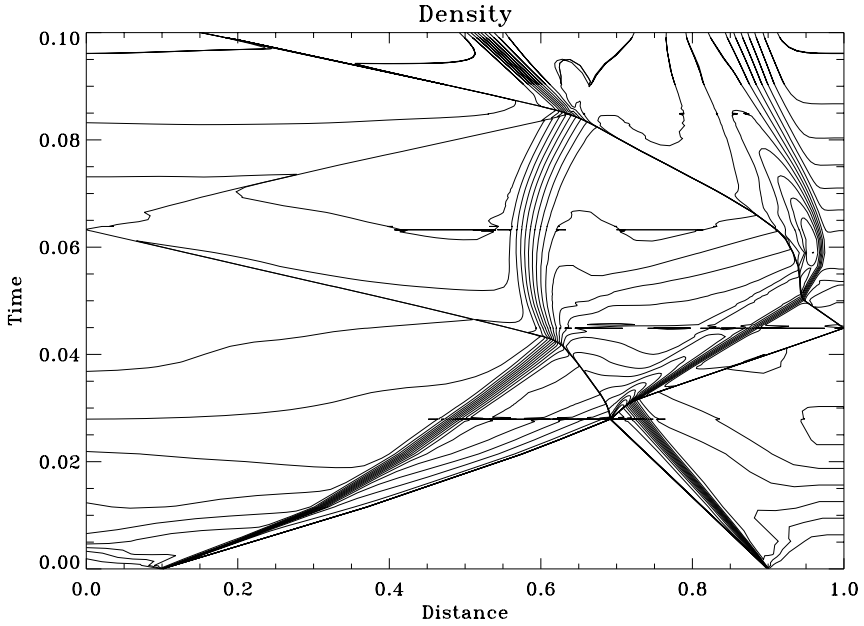


Fig. 9: Density contours (logarithmic scale)

The five density profiles (figures 7 & 8) are horizontal cross sections at the respective times. For times $t < 0.3$ there are the fast left traveling and slow right traveling shocks (speed given by the slope of the thick contour lines) each followed by a contact discontinuity which connects to a rarefaction fan. The fans are reflecting soon from the walls indicating that the region near the walls are being depleted quickly. At $t = 0.028$ the two shocks collide and slow down after the interaction while creating a secondary contact discontinuity. Shortly after the fast shock runs into the primary discontinuity from the right and accelerates again to near its original speed while the discontinuity is reflected. At $t = 0.045$ this shock reflects at the right wall and interacts with the discontinuity at about $t = 0.05$. Both nonlinear waves are slowed down considerably. The situation is comparable Sod's shock tube. New, however, is that the shock interacts with the secondary discontinuity and speeds up again. Shortly before $t = 0.045$ the slower right propagating shock collides with the other primary discontinuity and is greatly accelerated while the discontinuity is reflected. This shock now propagates uninterruptedly to

the left wall. There it reflects at time $t = 0.063$ and moves with similar speed towards the right again until it runs again into the discontinuity. This time, however, the other shock front meets also with them leading to a multiple interaction of nonlinear waves.

A comparison of TITAN's computation with the original one by Woodward and Collela (using a special version of the PPMLR scheme on a domain of 3096 zones) tells that the shocks are well represented but the contact discontinuities lack sharpness. This can be accounted for in part to a fairly large artificial mass and energy diffusion which allowed TITAN to complete the run in a little over 1000 time steps (about a factor 5 less than the number of time steps PPMLR required). For comparison reasons the space-time diagram of the grid motion is presented in the following figure.

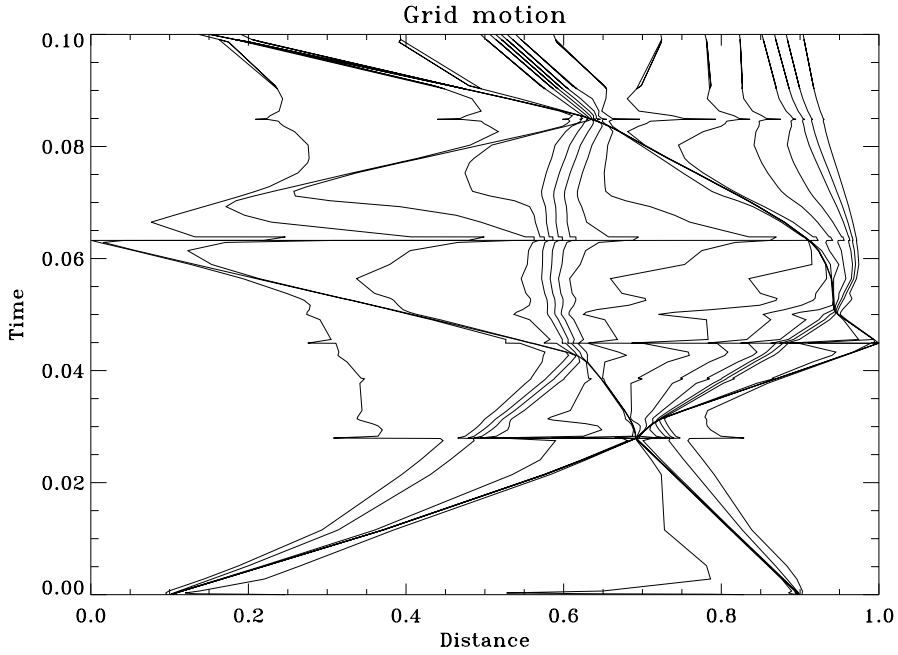


Fig. 10: Action of the adaptive grid

Figure 10 demonstrates well that the adaptive grid keeps track of all discontinuities. Note the strong correlation between the grid motion and the density contours. Its typical behavior at the momentary absence of the shock(s) due to reflection from a wall or shock-shock interaction is also evident.

B.3. Sedov-Taylor Self-Similar Blast Waves

TITAN's hydrodynamics benchmarking is concluded with the Sedov-Taylor blast wave. An energy deposition of 10^{50} *ergs* at the center of a ball with radius $R_{max} = 10^9$ *km* generates a spherical shock that propagates in a self-similar fashion. The gas is assumed to have a polytropic index $\gamma = \frac{5}{3}$ and is initially at rest with a constant density of 10^{-8} g/cm^3 and temperature of 50 K. This time, density and pressure serve as the only grid parameters. Logarithmic scaling is choosen for all quantities involved. The grid α is again kept at 1.5, the shock width is $\ell_1 = 10^{-4}$ and we take 100 zones to cover the ball.

This problem is set up so that the shock wave reaches the surface of the ball after $t = 2 \times 10^5$ *sec*. TITAN completes the calculation in about 1000 time steps. The resolution of the shock front is about 10^6 during the evolution. In the figure to the right the evolution of this blast wave is illustrated. Snapshots of the density and the velocity at five different times $t \in$ (140, 2900, 26590, 93023, 197670) *sec* are overlayed. The upwind density is always 10^{-8} g/cm^3 with the material at rest. At the shock front the density jump remains constant 4×10^{-8} g/cm^3. The velocity jump decreases from 117 km/sec to 1.5 km/sec during these five instances. Both the density and the velocity profiles fall off to zero behind the shock.

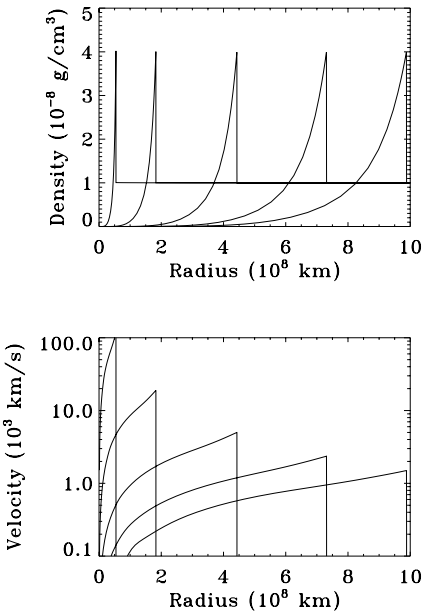


Fig. 11: Density and velocity profiles at five different times

The position, density, pressure, temperature, and velocity of the shock as a function of time are given in next figure. All variables vary strictly monotonically except for the shock density which remains at a steady value $\rho_s = 4 \times 10^{-8}$ g/cm^3 as it should. In particular the expected relations

$$R_s(t) \propto t^{2/5} \tag{54}$$

$$v_s(t) \propto t^{-3/5} \tag{55}$$

$$T_s(t) \propto P_s(t) \propto t^{-6/5} \tag{56}$$

are satisfied. These are the analytical solution as discussed in the textbook of Landau & Lifshitz.

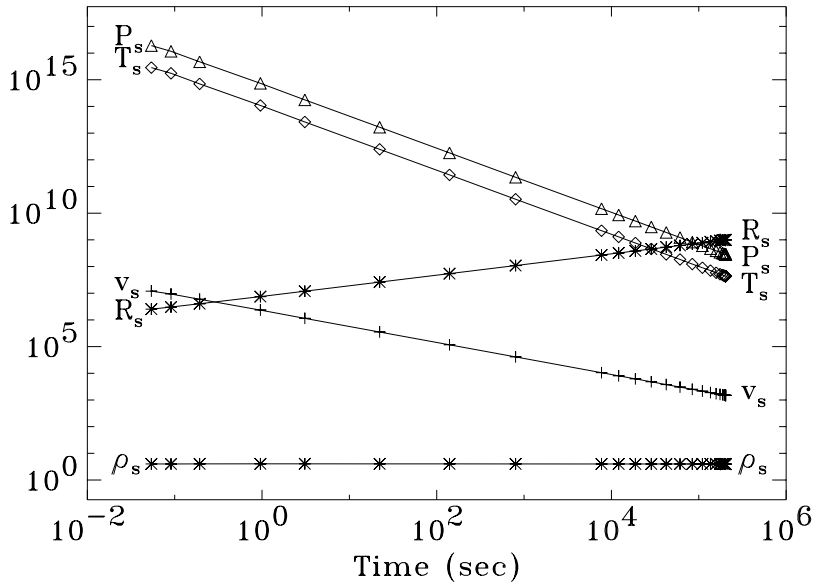


Fig. 12: Power-law dependencies of the shock quantities

The self-similarity of the computed flow is demonstrated in figure 13. It shows the profiles of density, pressure, and velocity in terms of the position, all measured with respect to the shock data. The graphics were generated by overlaying the actual grid values at six different times $t \in$ (140, 799, 12090, 42398, 110070, 174900) *sec*. One can verify that the shocked region obeys the analytical relations:

$$v/v_s \propto R/R_s \tag{57}$$

$$\rho/\rho_s \propto (R/R_s)^{3/(\gamma-1)} \tag{58}$$

and similarly for the pressure. Their functions jump to the preshock values and stay at a constant level. Compare this figure with the one in Landau & Lifshitz!

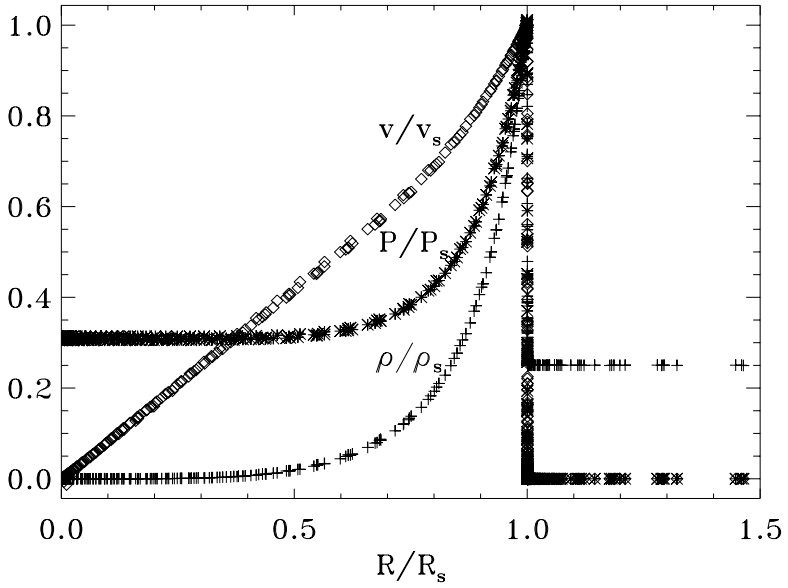


Fig. 13: Self-similar profiles at six different times

C. Radiative Transport Test Problems

In the following section L. Ensman's suggestions for radiative tests are implemented. Consider spheres with mass of about 0.5 $M_\odot$ and radial extension of $R = 2.84 \times 10^3\ R_\odot$ which are filled with pure hydrogen at a constant density $\rho = 2.9677 \times 10^{-11}\ g/cm^3$. Further assume that the hydrogen is fully ionized so that the sole opacity source is electron scattering. The mean flux opacity is then set proportional to the Thomson scattering opacity: $\rho\chi_F = \chi_e = 1.19 \times 10^{-11}\ cm^{-1}$. In addition the energy and Planck mean opacities are equated, $\kappa_E = \kappa_P$, and allowed to differ from the mean flux opacity by a constant factor, $\kappa_E = \zeta\chi_F$. This setup allows us to illustrate the effects of a scattering versus absorption dominated gas.

C.1. Radiative Heating

The sphere is set up to be in diffusion equilibrium at luminosity $\mathcal{L} = 826\ \mathcal{L}_\odot$. From the available data one computes the radiant energy according to

$$E(r) = \frac{\mathcal{L}}{4\pi c R}\left[\sqrt{3} + 3\chi_e\Big(\frac{R}{r} - 1\Big)\right] \tag{59}$$

and from that the temperature profile via

$$T(r) = \sqrt[4]{E(r)/a_r} \tag{60}$$

At time $t = 0$ a strong light source at the center is turned on which has a $10^{1.5}$ times higher luminosity, $\mathcal{L}_c = 26,130\ \mathcal{L}_\odot$, and illuminates the gas sphere. The sphere now has to evolve to a new radiative equilibrium at this higher luminosity.

One can calculate the relaxation with full transport under the assumption that the absorptive opacity is considerably smaller than the total opacity: $\kappa_E = 10^{-5}\ \chi_F = 10^{-5}\ \chi_e$. This defines the case of a scattering dominated gas. The computation is carried out with the adaptive grid. This time the radiation energy and the luminosity are taken as the grid parameters in logarithmic scaling and $\alpha = 3$, $\tau = 10^3$ is set.

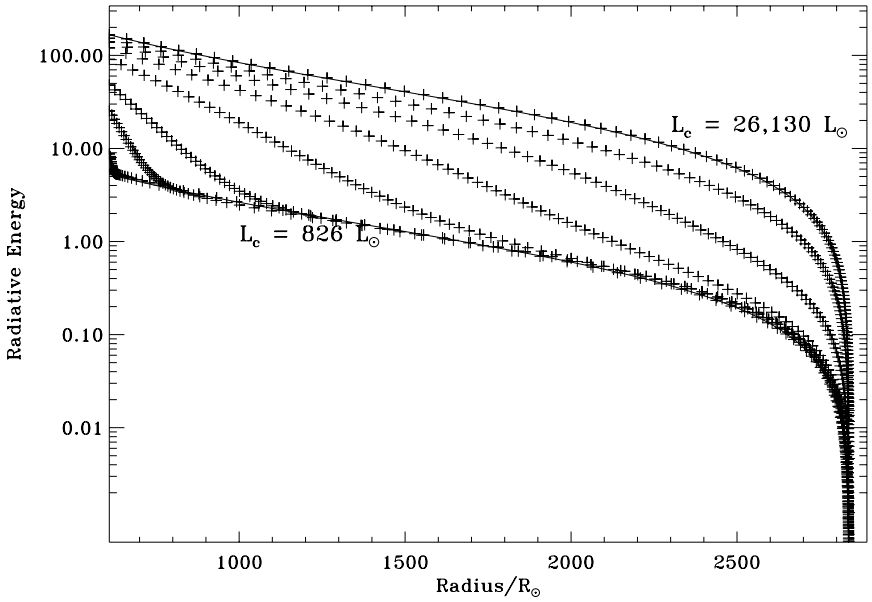


Fig. 14: Relaxation of the radiative energy profile

This graph illustrates how the higher central luminosity changes the profiles of the radiative energy in time. Initially the sphere is at equilibrium at the

lower luminosity, $\mathcal{L}_c = 826\ \mathcal{L}_\odot$, which is represented by curve $E(r)$ at the bottom (+'s indicating individual grid points). This curve relaxes and the radiative energy increases until the new equilibrium at the higher luminosity, $\mathcal{L}_c = 26,130\ \mathcal{L}_\odot$, is reached. For both the initial and the final $E(r)$ curve the analytical solution from equation (59) were overlayed. The agreement between the analytical (assuming diffusion) and numerical (assuming variable Eddington factors) result is remarkable. This should be expected because the shell is optically thick except right underneath the surface and hence the diffusion regime is established everywhere else.

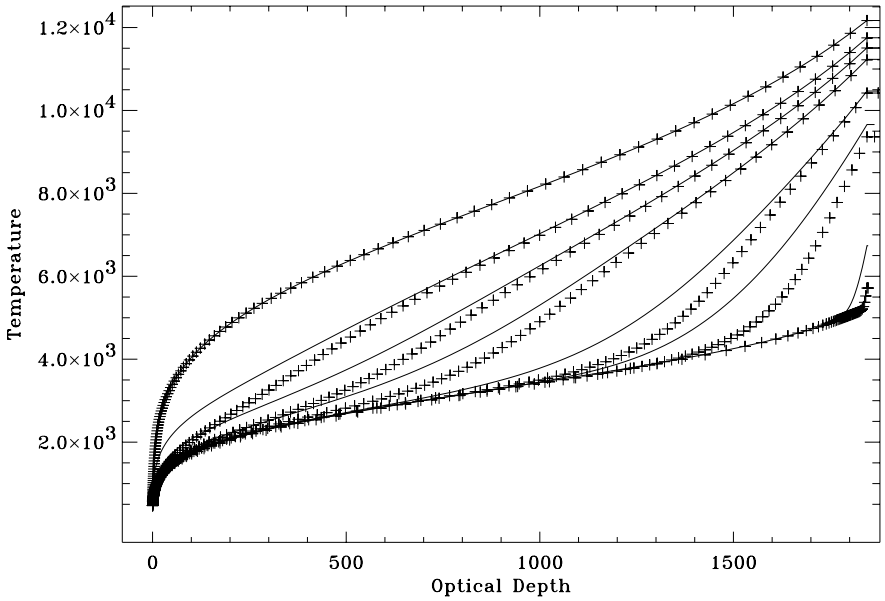


Fig. 15: Relaxation of gas and radiative temperatures towards equilibrium

The heating of the gas (+) and radiative (—) temperature by the central light source is given in figure 15. It shows the relaxation of their profiles as a function of the optical depth at various times. Both bottom curves are the initial condition in which the two temperatures agree because of equilibrium diffusion. When the light source is turned on a radiation wave moves through the sphere thereby heating up the gas. Since a weak coupling between the gas and the radiation field is assumed, $\zeta = 10^{-5}$, the gas temperature rises noticebly slower than the radiative temperature. As soon as the higher luminosity value has reached the surface both temperatures agree once again and the new radiative equilibrium is established. Because most of the gas

is optically thick the diffusion regime prevails and the temperature follows again the profile from equation (59).

This experiment is repeated with different factors ζ. The sphere relaxes to the same equilibrium state every time. Figure 16 depicts the evolution of the surface luminosity for six different choices $0 \leq \zeta \leq 1$. The correspondence between the symbols and the ratios f_r for the gray opacity is as follows: $\zeta = 0$ ($\Box$), 10^{-9} ($\triangle$), 10^{-7} ($\Diamond$), 10^{-5} ($*$), 10^{-2} ($+$), 1 (—). The relaxation of the temperature profiles from the previous figure is represented here by the asterisk. Note that the time the radiation wave takes to reach the surface and hence begins to raise the luminosity is, within a factor of 2, the same in all cases, namely around 6×10^6 *sec*. This number is in reasonable agreement with the classical diffusion time scale $\chi_e R^2/c \approx 1.4 \times 10^7$ *sec*. The spheres with almost pure scattering opacities reach the higher luminosity plateau first. Those spheres with almost pure absorptive opacities attain it later.

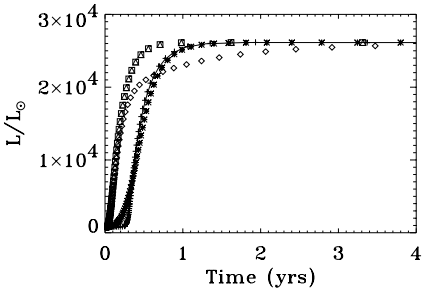


Fig. 16: Relaxation of the surface luminosity for six different opacities

The experimental setup allows us further to investigate the different radiative transport schemes in TITAN. There are three choices:

"equilibrium diffusion":
: The gas and radiative temperature are not distinguished, $E = a_r T^4$, and the radiative flux is evaluated according to equation (7). The differentiation between the mean opacities becomes meaningless and only the total opacity is considered.

"non-equilibrium diffusion":
: The radiative flux is still computed according to equation (7). But now the radiation energy E is computed from equation (6). Two temperatures are possible and the absorptive character of the gas can be taken into account. Both diffusion cases demand a constant Eddington factor $f_E = \frac{1}{3}$.

"full transport":

The radiation field is described by the two equations (4) & (6). Since they necessitate an estimate for f_E one obtains also information about the angular distribution of the intensity field.

The consistency of the three approaches can be checked in the context of the hydrogen spheres. The equilibrium diffusion computation can also be simulated with non-equilibrium diffusion by setting $\kappa_E = \chi_F$. Since the gas is optically thick (aside right underneath the surface) the corresponding full transport calculation should behave in the same manner. Performing the actual runs demonstrates a tight agreement between the three transport schemes.

C.2. Radiative Cooling

The final equilibrium model from above is now taken as the initial condition. At $t = 0$ the light source at the center is turned off, $\mathcal{L}_c = 0$. This time a fixed (Eulerian) grid instead of the adaptive grid is chosen. The runs are carried out with the full radiative transport scheme.

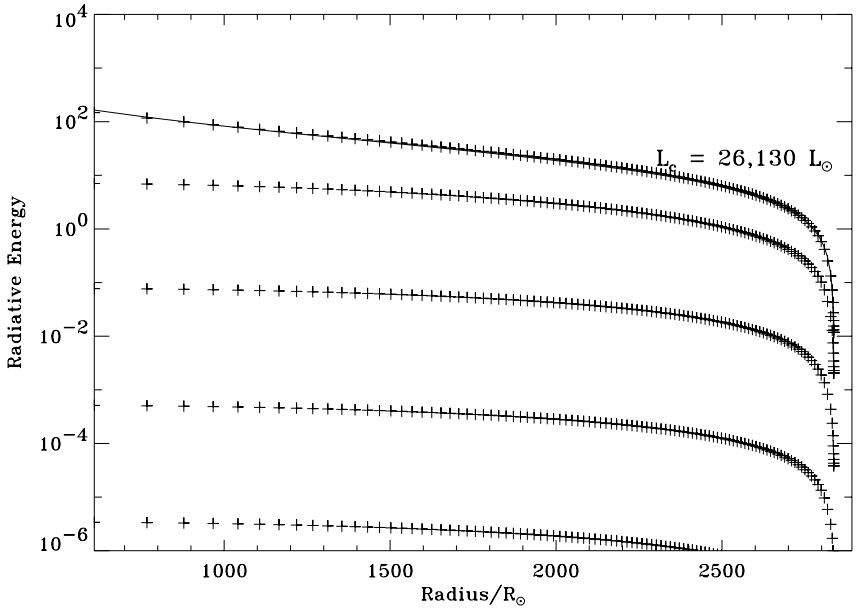


Fig. 17: Cooling of the radiative energy profile

This graph illustrates how the radiative energy decreases in time once the central luminosity is turned off. Initially the sphere is at equilibrium at $\mathcal{L}(0) = 26,130\,\mathcal{L}_\odot$ which is represented by the curve $E(r)$ (+'s again marking the grid points) at the top with the analytical solution overlayed. As time progresses, E decreases throughout the sphere as radiant energy leaks out, and the gradient of E flattens, consistent with $\mathcal{L}(t)$ deminishing to zero. At the surface a sharp gradient continues to drive $\mathcal{L}(t)$ outward. For $t \to \infty$ it goes to zero, $E \to 0$, which represents the new equilibrium when $\mathcal{L}_c = 0$.

Just like before one can consider different strengths of the gas-radiation coupling. In figure 18 the decline of the surface luminosity for various gray opacities defined by the ratios $0 \leq \zeta \leq 1$ are plotted. Again these ratios correspond to the symbols in the following way: $\zeta = 0$ ($\Box$), 10^{-9} ($\triangle$), 10^{-7} ($\diamond$), 10^{-5} ($*$), 10^{-2} ($+$), 1 (—). The curve of the diffusion regime ($\zeta = 1$) is representative of all regimes with $\zeta > 0$. Their analyses tells that the long time behavior of the luminosity decay follows a power law, $\mathcal{L}(t) \propto t^{-s}$, with the power being close to $s = \frac{1}{3}$. In contrast to that the regime of complete decoupling between the gas and radiative temperature, $\zeta = 0$, is characterized by an exponential decay, $\mathcal{L}(t) \propto e^{-t/s}$, with the time scale $s \approx 1.7 \times 10^7$ *sec* being close to the classical diffusion time. This regime behaves differently because the cooling wave alters only the radiative temperature but leaves the profile of the gas temperature unchanged.

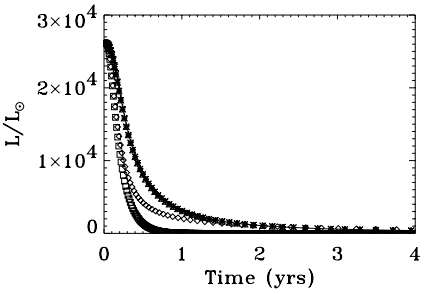


Fig. 18: Relaxation of the surface luminosity for six different opacties

In figure 19 the profiles of the gas (+) and radiative (—) temperature for the case $\zeta = 10^{-7}$ are plotted. Initially they are in equilibrium but the decoupling begins immediately. One observes that the profile of the radiative temperature is merely shifted downward in time. Simultaneously the profile of the gas temperature is modified to become a constant. Since the bulk of the total opacity comes from electron scattering the radiation cools off faster than the gas so that the gas energy contribution to the luminosity rises soon to domination. Note that the radiative temperature is related to the radiative energy via equation (60) and therefore shows the same behaviour as E in figure 17.

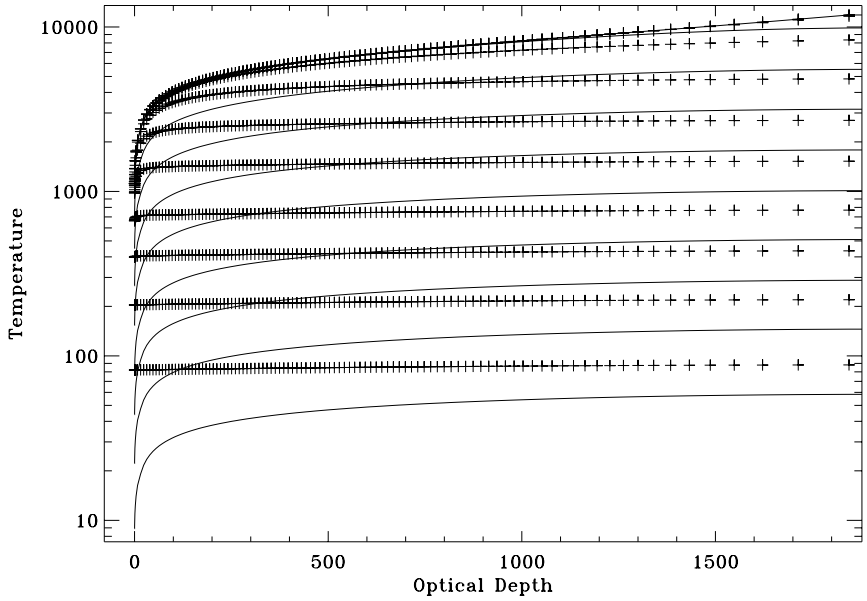


Fig. 19: Relaxation of gas and radiative temperatures towards equilibrium

D. Radiation Hydrodynamics Test Problems

This section describes three radiation hydrodynamics benchmarks. The fundamental understanding of radiating shocks was developed by Zel'dovich & Raizer and additional discussion of the phenomena involved is given by Mihalas & Mihalas. All tests are carried out in spherical geometry. The full transport scheme for the radiative field is employed. L. Ensman's test setups are again followed in detail.

Consider a thin shell with an inner radius $R_i = 8 \times 10^6\ km$ and outer radius $R_o = 8.7 \times 10^6\ km$ filled with a cold gas at constant density $7.78 \times 10^{-10}\ g/cm^3$. A shallow temperature profile

$$T(r) = 10 + 75\ \frac{r - R_o}{R_i - R_o}\ K \tag{61}$$

is assumed . The Rosseland mean opacity is taken to be constant $\rho\chi_{Ross} = 3.115 \times 10^{-10}\ cm^{-1}$, and all of it is absorptive, $\zeta = 1$. The gas is initially at rest. At time $t = 0$ a piston at the inner radius is moved outwards at a constant high speed. A shock front is thus formed which couples strongly to

the radiation field. If the maximum temperature immediately ahead of the shock is less than the downstream temperature one speaks of a "subcritical" shock, the case of equality one speaks of a "supercritical" shock.

In the following calculations the sphere is represented by 100 grid points. Density and mass in logarithmic scaling are selected as the sole grid parameters along with $\alpha = 1.5$ and $\tau = 10^{-20}$. The shock width is taken to be proprotional to the radius, $\ell_1 = 10^{-5}$, to demand a high resolution. All runs are executed fully implicit, *i.e.*, with time centering parameter $\theta = 1$.

D.1. Subcritical Shock

The piston speed is 6 km/sec. Figure 20 shows the radiative shock front propagating towards the surface. The four quantities gas T and radiative temperature T_r, radiative flux F, and density ρ are plotted as functions of the radius at seven different times. The shock front is located where the flux is largest. The temperatures just ahead of the shock are always smaller than those in the shocked region thus defining this run as a subcritical shock.

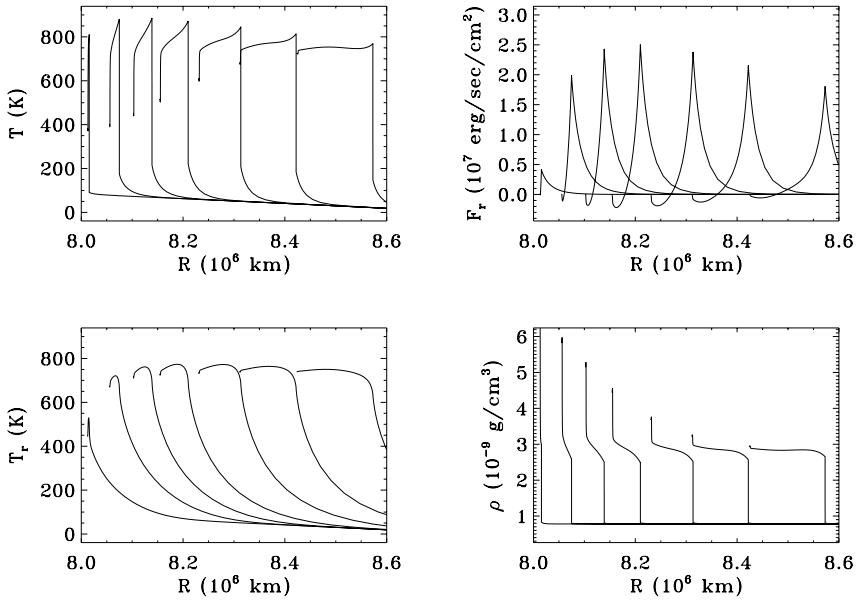


Fig. 20: Propagation of a subcritical shock

It is often useful to look at the dependencies of the physical quantities on other coordinates. The following figures illustrate this for the example of the gas temperature. The same seven snapshots were taken but instead of showing them as a function of the radius they are now plotted as a function of the exterior mass, of the optical depth, and of the grid point (zone index). The dependency $T = T(M_{ext})$ is the Lagrangean representation of the data which emphazises the outermost layers of the gas sphere. The graph $T = T(\tau)$ shows the temperature profile from a radiation transport perspective which stresses how transparent the features are. These functional dependencies help the physical interpretation of the results. On the other hand, the profile $T = \{T_k \, ; k = 1, \ldots, 100\}$ is interesting from a numerical point of view because one can learn from them how well the calculation was carried out. In the given example one can, for instance, see that the temperature jumps in the shock are well resolved by about 10 grid points.

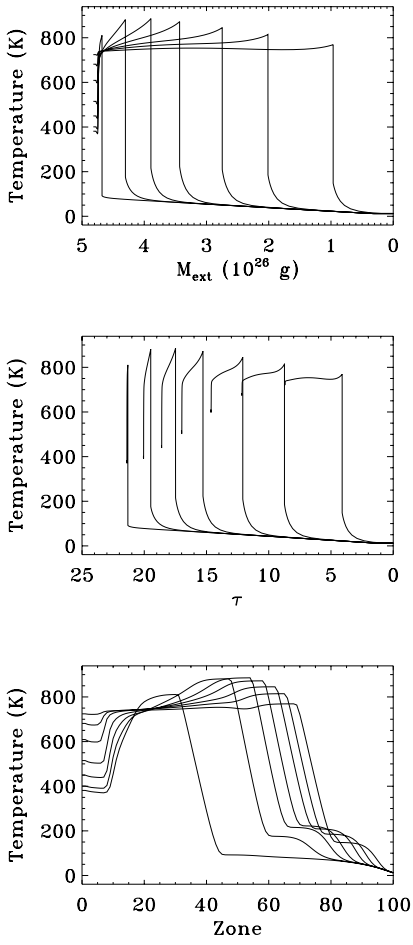


Fig. 21: Temperature profiles

Its characteristic features, as explained in Zel'dovich & Raizer, are an overshoot of the gas temperature, a nearly symmetric profile of the radiative flux, and an extended non-equilibrium region upstream from the shock, which is generated by the effects of radiative preheating from the flux. In figure 22 the density, gas and radiative temperature, flux and Eddington factor ($\cdots$, scaled so that 1.0 corresponds tp $f_E = \frac{1}{3}$), and velocity are plotted in dependency on the optical depth away from the front, $(\tau - \tau_s)$, at the half piston crossing time ($t = 56,581$ *sec* to be exact).

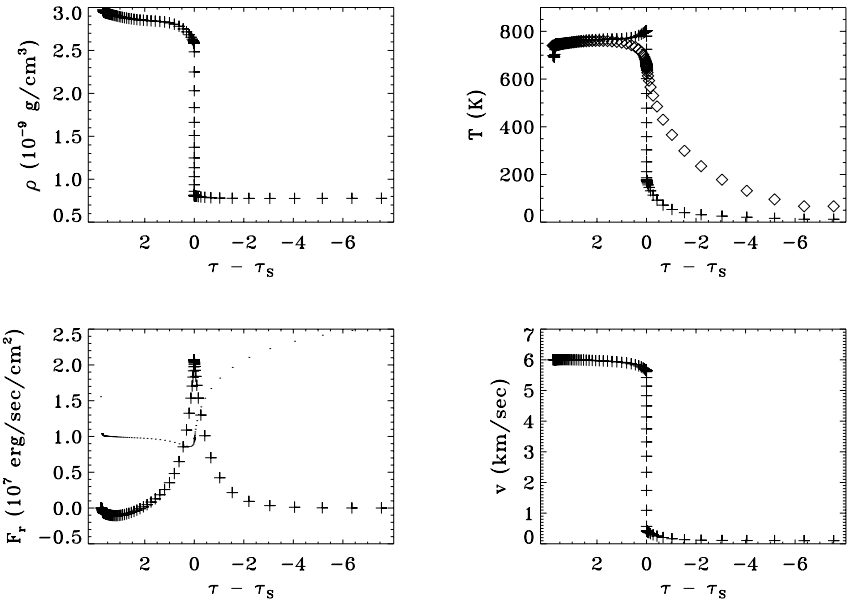


Fig. 22: Profiles as functions of the relative optical depth

A non-equilibrium diffusion analysis, *cf.* Zel'dovich & Raizer, demonstrates that the density, gas pressure, radiative flux and energy all decrease upstream depend exponentially on the optical depth away from the shock:

$$\rho(\tau) \propto p(\tau) \propto F(\tau) \propto E(\tau) \propto e^{-\sqrt{3}|\tau-\tau_s|} \tag{62}$$

One consequence is that the radiative temperature must decrease slower than the gas temperature. This functional dependency on the relative optical depth holds also downstream for the flux and the gas temperature. The maximum flux is determined by the postshock temperature T_2 : $F_s \approx \frac{1}{2\sqrt{3}} a_r c T_2^4$. The analytical values differ a factor less than 2 from the actual numerical data. The behavior of our simulation agrees quite well with the analytical estimates, and the existing deviations can be explained by the use of variable Eddington factors.

D.2. Supercritical Shock

The piston speed is 20 km/sec. Figure 23 shows the radiating shock while it propagates across the shell. The temperatures in the regions immedi-

ately ahead of and trailing the shock front are equal. Therefore this run is classified as a supercritical shock.

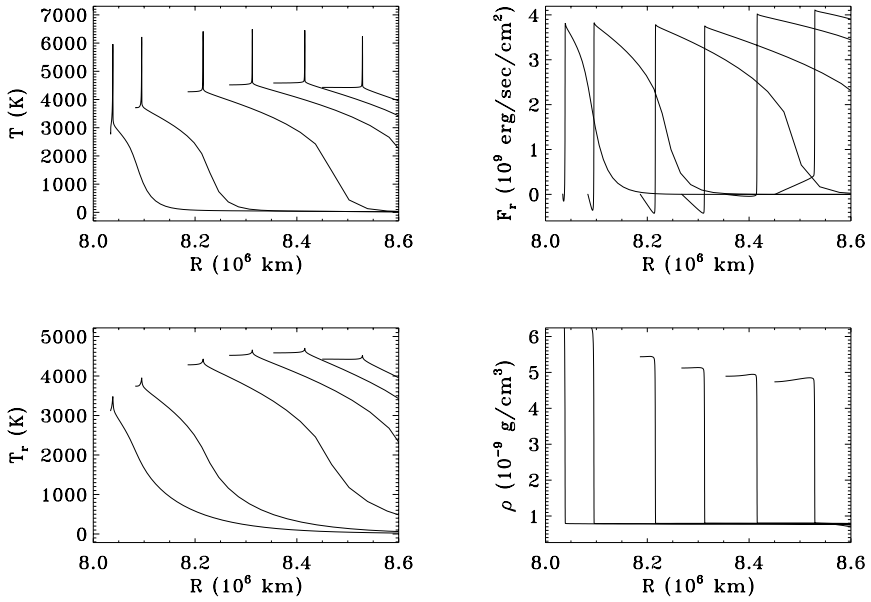


Fig. 23: Propagation of a supercritical shock

Its characteristic features are again a strong overshoot of the gas temperature, this time a highly asymmetric flux profile, and an extended radiative precursor whose head is out of equilibrium with the gas. In contrast to the subcritical case the immediate surroundings of the shock satisfy the condition of radiative equilibrium. The temperature overshoot itself is out of equilibrium and its width is narrower than the unit mean free path of the photons. This implies that the radiation flow from this feature cannot be treated correctly by the diffusion approximation. In figure 24 the density, gas and radiative temperature, flux and Eddington factor ($\cdots$ scaled so that 1 corresponds to $f_E = \frac{1}{3}$), and velocity are plotted in dependency on the optical depth away from the front, $\tau - \tau_s$, at one quarter piston crossing time ($t = 8,541\ sec$).

A non-equilibrium diffusion analysis according to Zel'dovich & Raizer predicts again that the radiative flux and radiative energy decline exponentially and so does the gas temperature in the upstream part of the precursor that is out of equilibrium. In the equilibrium region around the shock one finds

for the gas temperature a dependency on the optical depth away from the front like

$$T(\tau) \propto \sqrt[3]{1 + \frac{3\sqrt{3}}{4}|\tau - \tau_s|} \tag{63}$$

There is also an estimate for the maximum flux which is not as accurate as for the subcritical case. Radiative equilibrium is well established within $(\tau - \tau_s) < -5$ aside of the shock front itself. Density, flux, and velocity undergo jumps from the up- to the downstream regions. Our numerical results reflect the analytical predictions. Because the analytical results are not as reliable as in the previous case a close comparison does not give good results despite the fact we actually have $f_E = \frac{1}{3}$.

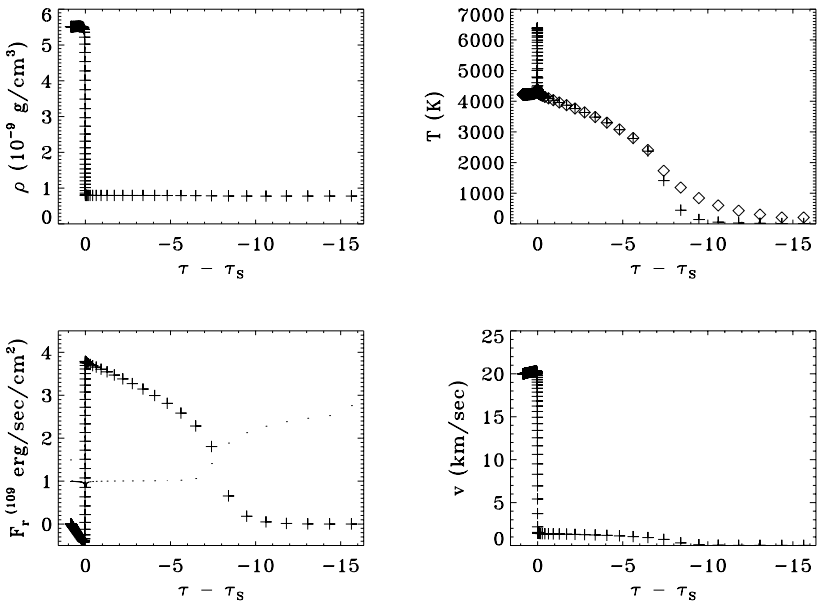


Fig. 24: Profiles as function of the relative optical depth

The spike in the gas temperature is prominent and very narrow. Its width is given by $(\rho\chi_e)^{-1} = 3,210\ km$. This is much larger than the rezoning limitation of $\ell \approx 80\ km$ imposed by the artificial shock width. In figure 25 on the right gas $(+)$ and radiative $(\diamond)$ temperature are plotted in the vicinity of the shock front. Their dependency in the relative optical depth is almost independent on time. The spike exists clearly only for the gas temperature while the radiative temperature varies smoothly. The spike is optically thin

and hence practically transparent to the radiation (although the actual location is mostly at large optical depths). Note that there are many (around 25) grid points in this region, each zone with an optical depth of at most 5×10^{-3}. The spike is therefore numerically well represented and represents the physical, in particular radiative, properties correctly.

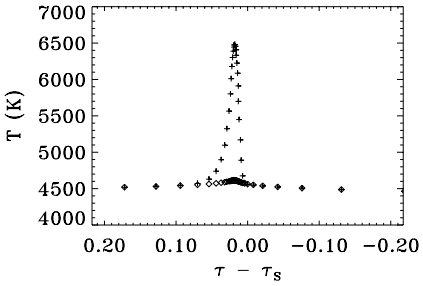


Fig. 25: Temperature spike

D.3. Radiative Blast Wave

Now consider a shell stretching from an inner radius $R_i = 10^7\ km$ to an outer radius of $R_o = 10^8\ km$ containing a mass of $0.16\ M_\odot$. Instead of a constant density profile a power law atmosphere is assumed:

$$\rho(r) = 10^{-6} \left(\frac{R_i}{r}\right)^7 g/cm^3 \tag{64}$$

The gas is purely absorptive and the mean opacity taken to be constant and equal to the Thomson free electron scattering opacity (hence implying the gas is completely ionized hydrogen). The adiabatic index is $\gamma = \frac{5}{3}$. The sphere is illuminated by a central light source with a luminosity $\mathcal{L} = 10^{34}\ ergs/sec$. A self-consistent temperature profile for the initial model is obtained by relaxating the sphere's luminosity to this value using the full transport scheme, *cf.* the discission of radiative heating.

At time $t = 0$ a large amount of energy, corresponding to a central temperature of $10^9\ K$, is deposited inside the sphere. This sets off a radiative blast wave (differing from the Sedov-Taylor blast wave by the fact that now the effects of radiation are included). Its evolution is followed employing the adaptive grid. As grid parameters we choose the mass, density, and radiative energy in logarithmic and the velocity u ($1000\ km/sec$) in linear scaling, together with $\alpha = 1.5$, $\tau = 10^{-20}$, and $\theta = 1$. The shock width is $\ell_1 = 10^{-4}$. After initiating the blast wave it reaches the surface in 700 time steps.

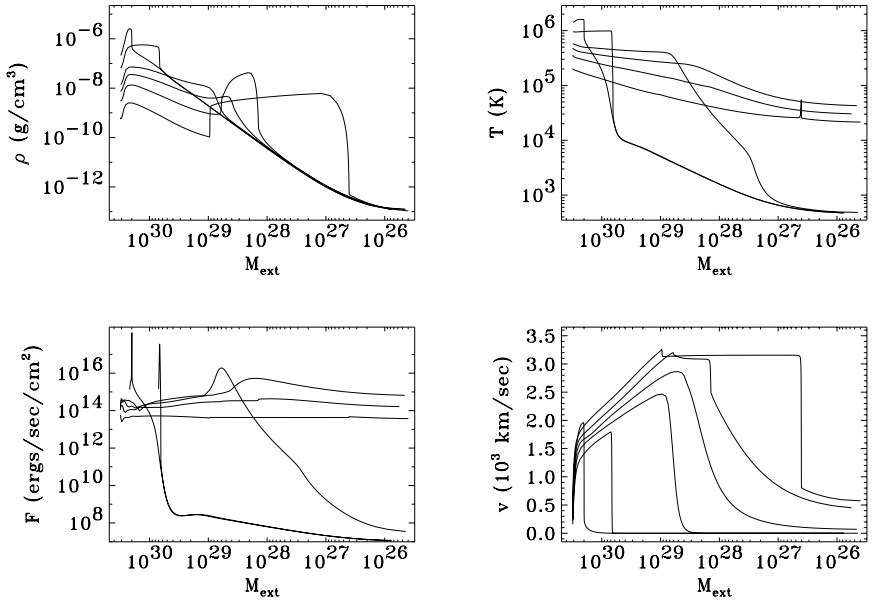


Fig. 26: Evolution of a radiative blast wave

In figure 26 the evolution of the gas sphere is depicted at six different times. Initially the shock moves down the density and temperature gradient. There is a very prominent flux peak and the shock profiles for density, temperature and velocity are sharp. The shock looks like an adiabatic one. In time the radiation preheats an increasing region ahead of it. When the shock has penetrated to an optical depth of about 200 (which occurs at time $t \approx 7000\ sec$) this radiative precursor has reached the surface. The surface luminosity rises very fast and reaches its maximum at about $t = 1.13 \times 10^4\ sec$, see figure 27. The gas and radiative temperatures are out of equilibrium at the head of the precursor. The heating induces a quasi-isothermal wind in the outer shell and the gas equilibrates again. Before reaching an optical depth of 60 the shock has dissipated through the radiative processes. The shocked gas, however, maintains its momentum and begins a "snow plow" phase where it sweeps up wind material. The surface

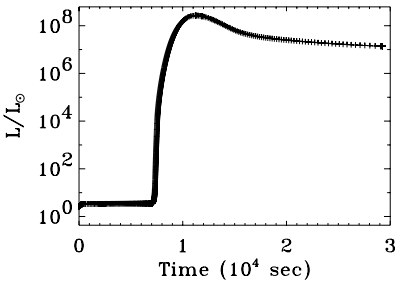


Fig. 27: Light curve of the blast

luminosity declines and an almost constant flux profile becomes established. The sphere begins to cool off. Behind the "snow plow" a very dense shell is formed which is well established by $t \approx 1.5 \times 10^4$ *sec* at optical depth $\tau = 16$. While the plow steepens up at its head to a strong supercritical shock (with the typical temperature spike) another, reverse shock forms at its tail. It propagates slowly into the homologously expanding inner regions. The dense shell expands rapidly and moves at nearly the shock speed outwards. Through the radiation the gas keeps cooling down, and, since the almost constant flux level decreases, the surface luminosity keeps dropping.

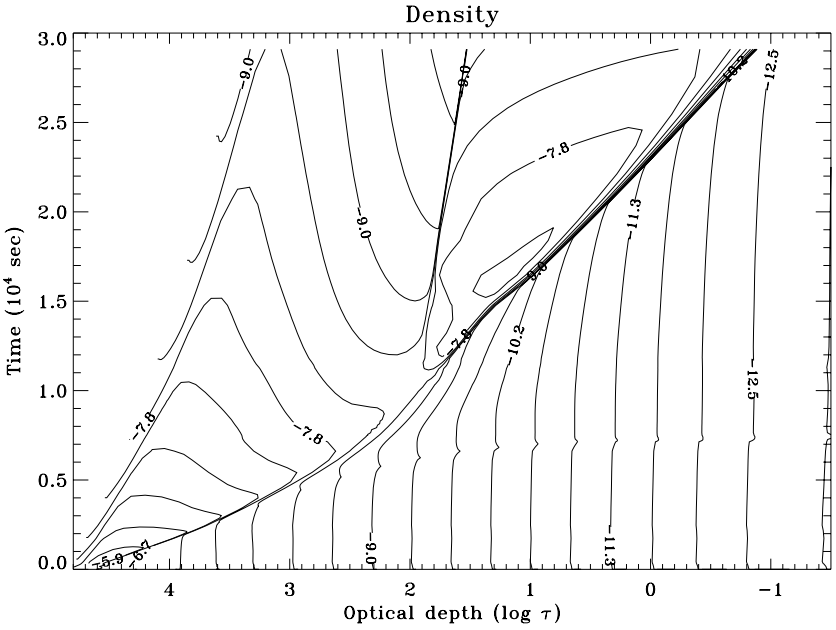


Fig. 28: Density contours (logarithmic scale)

The evolution of the radiative blast simulation can be illustrated via "optical depth-time diagrams" of the density, velocity, and temperature (figures 28 – 30). One can recognize the adiabatic shock during the initial phase up to several 1000 *sec*. While it dissipates through radiative losses the radiative precursor surfaces at about 10^4 *sec* and the luminosity there breaks out. The shocked regions plow into the isothermal wind while the gas keeps cooling off. A new supercritical shock is born and runs fast towards the surface. Simultaneously a reverse adiabatic shock forms. It moves into the inner regions which already undergo a homologous expansion and a high density shell at a constant speed is created.

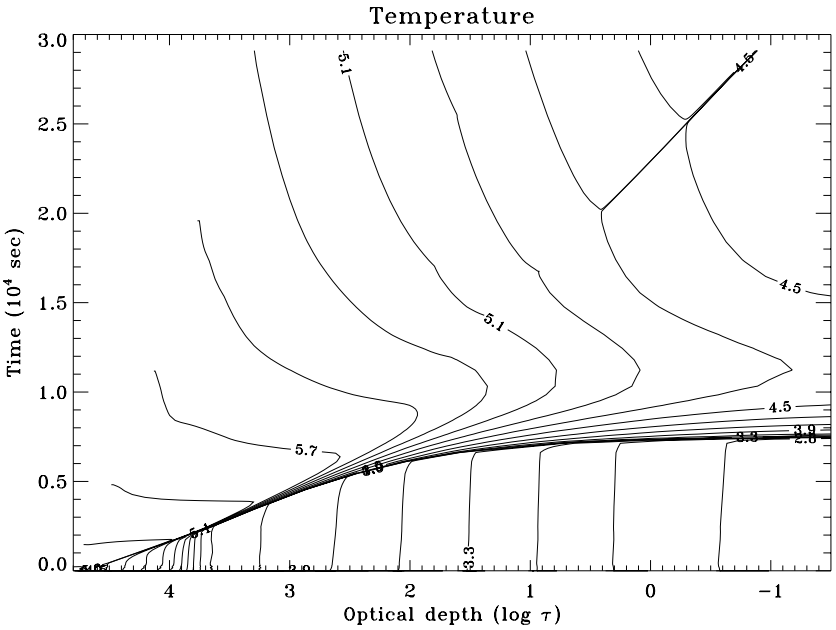


Fig. 29: Temperature contours (logarithmic scale)

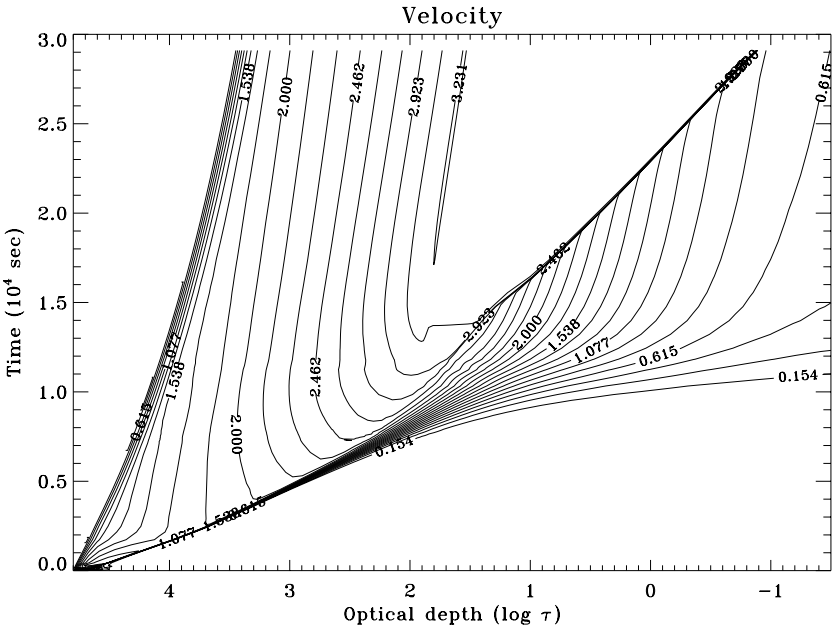


Fig. 30: Velocity contours (linear scale)

The grid motion is depicted in figure 31 in which every fourth zone is plotted as a function of optical depth and time. After a very short relaxation period the grid points begin to zoom into the forming shock, $t \approx 2000$ *sec*. They actually lock into the radiative precursor until it surfaces at $t \approx 7000$ *sec*. The grid then rezones quickly to resolve a new shock system that forms around $t \approx 10^4$ *sec*. It tracks a reverse shock and follows a forward shock all the way to the surface.

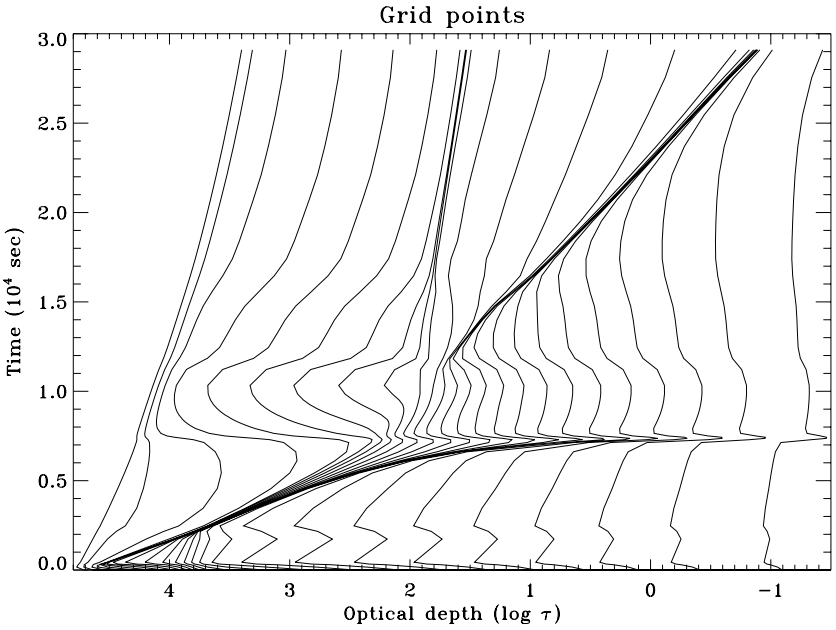


Fig. 31: Action of the adaptive grid

VI. CONCLUSION

We ask the TFUG novice to carefully study this user guide and repeat the given example calculations to gain familiarity with TITAN. The user should be able to reproduce all test problems before attempting to specify his or her own problem. One needs some practise to learn how the adaptive grid behaves. We always advise to first write one's favourite problem for a fixed grid and get TITAN to perform the computation properly and then to see how the adaptive grid can be employed. A great deal of questions regarding the code itself can be answered by inspecting the programming together with the TITAN Code Reference Manual. When setting up your own problem it is beneficial to always refer to the code itself for guidance.

We have set up a mailing list for people using TITAN. It's e-mail address is `dmihalas@altair.astro.uiuc.edu`. The list is primarily intended for discussion of issues relating to TITAN's algorithm, and dissemination of information directly to its usage. The list will also be used to announce any bug fixes, updates and release news. We appreciate the reporting of any bug findings, complaints, suggestions, or comments.

Good Application!