A Computer–assisted Proof of the Existence of Traveling Wave Solutions to the Scalar Euler Equations with Artificial Viscosity

Oswald Fogelklou and Warwick Tucker
Department of Mathematics
Uppsala University
P.O. Box 480, SE–751 06 Uppsala, Sweden.
oswald@math.uu.se
warwick@math.uu.se
Gunilla Kreiss
Department of Information Technology
Uppsala University
Box 337, SE–751 05 Uppsala, Sweden.
gunilla.kreiss@it.uu.se

November 18, 2010

Abstract

We establish the existence and local uniqueness of traveling wave solutions to the one–dimensional Euler equations with artificial viscosity. The equations are expressed as a fixed–point problem, which is solved by a computer–assisted method based on Yamamoto's application of the Banach fixed–point theorem.

Key words: computer–assisted proof, numerical verification, Euler equations, enclosure, existence, boundary value problems, fixed–point problems
AMS subject classification: 65L10, 65G20, 34B15, 47H10

1 Introduction

It is known that a scalar hyperbolic system of the type

$$u_t + f(u)_x = 0,\ x \in \mathbb{R},\ t > 0$$

with data

$$u(x,0) = \begin{cases} u_l,\ x < 0, \\ u_r,\ x > 0, \end{cases}$$

has a unique solution under some circumstances (e.g. $|u_r - u_l|$ must be sufficiently small) according to [Sm83]. For a specific $f(u)$, the two–dimensional Euler equations are known to have a weak solution under some conditions [Qu96] which is the limit of the solution of the Navier–Stokes equations as the viscosity vanishes. The Navier–Stokes equations themselves are known to have solutions in two dimensions [La69], and weak solutions in three dimensions [Le34]. Establishing the existence of smooth solutions in three dimensions, however, is still an open problem on Smale's list of 18 challenging problems for the twenty-first century [Sm98].

Regarding the one–dimensional Euler equations with boundary conditions, much numerical work has been done. Examples include the implicit finite difference technique of beam and warming [Du88], higher order schemes [Ra92] and linearizations [Jo02]. A comparison between three schemes is found in [Hu06]. From a numerical point of view, it is in many situations "obvious" that a solution exists and is unique. From a

mathematical point of view, however, this is not the case. Partial results exist [BF09], but there is still much left to do. We will employ a computer–assisted method as an analytical tool. Reformulating the original system as a fixed–point problem, the existence and local uniqueness are established via an application of the Banach fixed–point theorem. To avoid discontinuities we introduce artificial viscosity as in [Br02] and [Za05].

2 Problem (re-)formulation

In what follows, we will work with the one–dimensional Euler equations,

$$\begin{aligned} \frac{\partial \rho}{\partial \tilde{t}} + \frac{\partial(\rho q)}{\partial \tilde{x}} &= 0, \\ \frac{\partial(\rho q)}{\partial \tilde{t}} + \frac{\partial(\rho q^2)}{\partial \tilde{x}} + \frac{\partial p}{\partial \tilde{x}} &= 0, \\ \frac{\partial E}{\partial \tilde{t}} + \frac{\partial((E+p)q)}{\partial \tilde{x}} &= 0 \end{aligned} \tag{1}$$

or, in vector notation,

$$y_{\tilde{t}} + f(y)_{\tilde{x}} = 0, \tag{2}$$

where $y = (\rho, \rho q, E)^T$ and $f(y) = (\rho q, \rho q^2 + p, (E+p)q)^T$. If we introduce artificial viscosity in Euler equations (2) as in [Br02] and [Za05] we get

$$y_{\tilde{t}} + f(y)_{\tilde{x}} = \varepsilon y.$$

The parameter ε can be absorbed via the scaling $\hat{x} = \tilde{x}/\varepsilon$ and $\hat{t} = \tilde{t}/\varepsilon$, resulting in

$$y_{\hat{t}} + f(y)_{\hat{x}} = y_{\hat{x}\hat{x}}.$$

Our Ansatz will be traveling wave solutions $y(\hat{x}, \hat{t}) = u(\hat{x} - s\hat{t})$, where s is the shock speed. By setting $x = \hat{x} - s\hat{t}$, we arrive at

$$(f(u) - su)_x = u_{xx}. \tag{3}$$

The goal of this article is to establish existence and local uniqueness of solutions to equation (3) on some interval $[-L, L]$. For simplicity we use Dirichlet boundary conditions. The three unknowns are the density ρ, the density multiplied by the speed ρq and the energy density E. The boundary values are computed by the Rankine–Hugoniot conditions given the pressure, density and the speed on the left–hand side and the pressure on the right–hand side. The boundary values on the left–hand side are $\rho_L = 1$, $q_L = 5.17$ and $p_L = 1$ just as in [Za05]. We will vary the pressure on the right–hand side p_R. To satisfy the entropy conditions, however, we always let $p_R > p_L$.

2.1 The Rankine–Hugoniot conditions

For an ideal polytropic gas we have

$$E_L = \frac{p_L}{\gamma - 1} + \frac{1}{2}\rho_L q_L^2, \tag{4}$$

where $\gamma = 7/5 = 1.4$, since air is mainly diatomic. With the notation $[x] \stackrel{\text{def}}{=} x_R - x_L$ we can express the Rankine–Hugoniot conditions as

$$s[\rho] = [\rho q], \tag{5}$$
$$s[\rho q] = [p + \rho q^2], \tag{6}$$
$$s[E] = [(E+p)q]. \tag{7}$$

Now we define

$$\mu^2 \stackrel{\text{def}}{=} \frac{\gamma - 1}{\gamma + 1} = 1/6,\ c^2 \stackrel{\text{def}}{=} \gamma\frac{p}{\rho},\ v \stackrel{\text{def}}{=} q - s,\ m \stackrel{\text{def}}{=} \rho v. \tag{8}$$

From (5) we get $[m] = 0$, which together with (6) yields $[p + mv] = 0$. The equations $[m] = 0$, $[p + mv] = 0$, (7) and the fact that $pv = (\gamma p/\rho)(\rho v/\gamma) = mc^2/\gamma$ yield

$$[\frac{2c^2}{\gamma - 1} + v^2] = 0. \tag{9}$$

From $[m] = 0$, $[p + mv] = 0$ and (9) one can show

$$\rho_R = \rho_L \frac{p_R + \mu^2 p_L}{p_L + \mu^2 p_R} \tag{10}$$

and

$$v_L = \sqrt{\gamma \frac{p_R + \mu^2 p_L}{\rho_L(1 + \mu^2)}}.$$

Hence the shock speed is obtained:

$$s = q_L - v_L = q_L - \sqrt{\gamma \frac{p_R + \mu^2 p_L}{\rho_L(1 + \mu^2)}}. \tag{11}$$

We have the following equations for the critical speed c_*

$$(1 - \mu^2)c_L^2 + \mu^2 v_L^2 = (1 - \mu^2)c_R^2 + \mu^2 v_R^2 = c_*^2.$$

With the Prandtl's relation $v_L v_R = c_*^2$, we therefore get

$$\begin{aligned} v_R &= \frac{c_*^2}{v_L} = \frac{(1 - \mu^2)c_L^2 + \mu^2 v_L^2}{v_L} = (1 - \mu^2)\frac{\gamma p_L}{\rho_L v_L} + \mu^2 v_L \\ &= p_L(1 - \mu^2)\sqrt{\frac{\gamma(1 + \mu^2)}{\rho_L(p_R + \mu^2 p_L)}} + \mu^2 \sqrt{\frac{\gamma(p_R + \mu^2 p_L)}{\rho_L(1 + \mu^2)}}. \end{aligned} \tag{12}$$

Hence, according to (11) and (12),

$$\begin{aligned} q_R &= v_R + s \\ &= p_L(1 - \mu^2)\sqrt{\frac{\gamma(1 + \mu^2)}{\rho_L(p_R + \mu^2 p_L)}} + \mu^2 \sqrt{\frac{\gamma(p_R + \mu^2 p_L)}{\rho_L(1 + \mu^2)}} + q_L - \sqrt{\frac{\gamma(p_R + \mu^2 p_L)}{\rho_L(1 + \mu^2)}} \\ &= q_L + (1 - \mu^2)\left(p_L \sqrt{\frac{\gamma(1 + \mu^2)}{\rho_L(p_R + \mu^2 p_L)}} - \sqrt{\frac{\gamma(p_R + \mu^2 p_L)}{\rho_L(1 + \mu^2)}}\right). \end{aligned} \tag{13}$$

Finally, we have

$$E_R = \frac{p_R}{\gamma - 1} + \frac{1}{2}\rho_R q_R^2, \tag{14}$$

which can be computed using (13) and (10). Again, for an ideal polytropic gas, we have

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho q^2.$$

Hence

$$p = (\gamma - 1)E - \frac{1}{2}(\gamma - 1)\rho q^2. \tag{15}$$

2.2 The fixed–point formulation

If we insert equation (15) into equation (3) we get

$$(f(u) - su)_x = u_{xx}, \tag{16}$$

where $u = (\rho, \rho q, E)^T$ and

$$\begin{aligned} f(u) &= (u_2, u_2^2/u_1 + u_1, (u_3 + u_1)u_2/u_1) = (\rho q, \rho q^2 + p, (E + p)q)^T \\ &= (\rho q, \rho q^2 + (\gamma - 1)E - \frac{1}{2}(\gamma - 1)\rho q^2, (\gamma E - (\gamma - 1)\frac{1}{2}\rho q^2)q)^T \\ &= \left(\rho q, \frac{(3-\gamma)(\rho q)^2}{2\rho} + (\gamma - 1)E, \frac{\gamma E \rho q}{\rho} + \frac{(1-\gamma)(\rho q)^3}{\rho^2}\right)^T. \end{aligned}$$

We now integrate taking the boundary conditions into account:

$$\begin{aligned} \rho(x) &= \rho_L + \int_{-L}^{x} (\rho q(\tau) - s\rho(\tau))d\tau \\ &+ \frac{x+L}{2L}\left(\rho_R - \rho_L - \int_{-L}^{L} (\rho q(\tau) - s\rho(\tau))d\tau\right), \end{aligned} \tag{17}$$

$$\begin{aligned} \rho q(x) &= (\rho q)_L + \int_{-L}^{x} \left(\frac{3-\gamma}{2}\frac{(\rho q(\tau))^2}{\rho(\tau)} + (\gamma - 1)E(\tau) - s\rho q(\tau)\right)d\tau \\ &+ \frac{x+L}{2L}\left((\rho q)_R - (\rho q)_L - \int_{-L}^{L} \left(\frac{3-\gamma}{2}\frac{(\rho q(\tau))^2}{\rho(\tau)} + (\gamma - 1)E(\tau) - s\rho q(\tau)\right)d\tau\right), \end{aligned} \tag{18}$$

$$\begin{aligned} E(x) &= E_L + \int_{-L}^{x} \left(\frac{\gamma E(\tau)\rho q(\tau)}{\rho(\tau)} + \frac{1-\gamma}{2}\frac{(\rho q(\tau))^3}{(\rho(\tau))^2} - sE(\tau)\right)d\tau \\ &+ \frac{x+L}{2L}\left(E_R - E_L - \int_{-L}^{L} \left(\frac{\gamma E(\tau)\rho q(\tau)}{\rho(\tau)} + \frac{1-\gamma}{2}\frac{(\rho q(\tau))^3}{(\rho(\tau))^2} - sE(\tau)\right)d\tau\right). \end{aligned} \tag{19}$$

Writing $u = (\rho, \rho q, E)^T$, the equations (17)–(19) can be written as the fixed–point equation

$$u = F(u) = (F_1(u), F_2(u), F_3(u))^T, \tag{20}$$

with

$$\begin{aligned} F_1(u(x)) &= \rho_L + \int_{-L}^{x} (\rho q(\tau) - s\rho(\tau))d\tau \\ &+ \frac{x+L}{2L}\left(\rho_R - \rho_L - \int_{-L}^{L} (\rho q(\tau) - s\rho(\tau))d\tau\right), \end{aligned} \tag{21}$$

$$\begin{aligned} F_2(u(x)) &= (\rho q)_L + \int_{-L}^{x} \left(\frac{3-\gamma}{2}\frac{(\rho q(\tau))^2}{\rho(\tau)} + (\gamma - 1)E(\tau) - s\rho q(\tau)\right)d\tau \\ &+ \frac{x+L}{2L}\left((\rho q)_R - (\rho q)_L - \int_{-L}^{L} \left(\frac{3-\gamma}{2}\frac{(\rho q(\tau))^2}{\rho(\tau)} + (\gamma - 1)E(\tau) - s\rho q(\tau)\right)d\tau\right), \end{aligned} \tag{22}$$

$$\begin{aligned} F_3(u(x)) &= E_L + \int_{-L}^{x} \left(\frac{\gamma E(\tau)\rho q(\tau)}{\rho(\tau)} + \frac{1-\gamma}{2}\frac{(\rho q(\tau))^3}{(\rho(\tau))^2} - sE(\tau)\right)d\tau \\ &+ \frac{x+L}{2L}\left(E_R - E_L - \int_{-L}^{L} \left(\frac{\gamma E(\tau)(\rho q(\tau)}{\rho(\tau)} + \frac{1-\gamma}{2}\frac{(\rho q(\tau))^3}{(\rho(\tau))^2} - sE(\tau)\right)d\tau\right). \end{aligned} \tag{23}$$

3 Main Result

We are now prepared to state the main result of this paper:

Theorem 3.1. *Let $L = 0.04$, $\gamma = 1.4$, $\rho_L = 1$, $q_L = 5.17$, $p_L = 1$ and $p_R = 10$. Given the remaining boundary conditions computed by equations (4), and (10)–(14), there exists a solution to*

$$(f(u) - su)_x = u_{xx}, \quad -L \le x \le L, \tag{24}$$

where $u = (\rho, \rho q, E)^T$ *and*

$$\begin{aligned} f(u) &= (\rho q, \rho q^2 + p, (E+p)q)^T \\ &= (\rho q, \rho q^2 + (\gamma-1)E - \frac{1}{2}(\gamma-1)\rho q^2, (\gamma E - (\gamma-1)\frac{1}{2}\rho q^2)q)^T \\ &= \left(\rho q, \frac{(3-\gamma)(\rho q)^2}{2\rho} + (\gamma-1)E, \frac{\gamma E \rho q}{\rho} + \frac{(1-\gamma)(\rho q)^3}{\rho^2}\right)^T, \end{aligned}$$

The parameter values are taken from [Za05]. We will call these values the *standard* parameter values. The existence of solutions for some other parameter values is presented in section 9.

4 Notation

In the proof of Theorem 3.1, we will work with meshes on the interval $I_L = [-L, L]$. Such a mesh is defined by a set of mesh points $-L = x_0 < x_1 < \ldots < x_{N-1} < x_N = L$. The set $\{[x_0, x_1], [x_1, x_2], \ldots, [x_{N-1}, x_N]\}$ is called a *partition* of the interval I_L. Given a partition and a function u: $I_L \to \mathbb{R}^3$, we define the samples $u_{k,i} = u_k(x_i)$ over the mesh points. Let $\Pi^h u$: $I_L \to \mathbb{R}^3$ be the function that coincides with u on the mesh points, and is linear in each component on each element of the partition. Since $\Pi^h u$ is uniquely defined by its values at the mesh points, it can be described by a vector with the mesh point samples as components, written $\overline{\Pi^h u}$. We will also sometimes use the notation $\rho_i = \rho(x_i) = u_1(x_i)$, $\rho q_i = \rho q(x_i) = u_2(x_i)$ and $E_i = E(x_i) = u_3(x_i)$. To shorten some complicated expressions, we use the notation $h_i = x_{i+1} - x_i$ together with

$$P_j(\overline{f}) \equiv f_{j-1}/f_j, \quad \text{and} \quad Q_j(\overline{f}) \equiv f_{j-1}/(f_j - f_{j-1})$$

for vectors $\overline{f}$ with indices.

5 Spaces and Operators

In order to study the fixed–point problem (20), we assume that ρ_L, q_L, p_L and p_R are given so that all the boundary values ρ_L, ρq_L, E_L, ρ_R, ρq_R and E_R are either known or can be computed by the equations (4)–(14).

Consider the space of continuous functions with appropriate boundary conditions:

$$\begin{aligned} V_1 = \{v \in C(I_L; \mathbb{R}^3) \quad : \quad & v_1(-L) = \rho_L, v_2(-L) = (\rho q)_L, v_3(-L) = E_L, \\ & v_1(L) = \rho_R, v_2(L) = (\rho q)_R, v_3(L) = E_R\}. \end{aligned} \tag{25}$$

Then a solution of (20) must belong to V_1. Let S_1^h be the intersection of V_1 and the space of all piecewise linear functions on some fixed partition of I_L, and let u^h be a piecewise linear, continuous, approximate solution of (20) in S_1^h. If we introduce the following space

$$V_2 = \{v \in C(I_L; \mathbb{R}^3) \colon v_i(\pm L) = 0 \ \forall i \in \{1, 2, 3\}\},$$

then the error $\hat{w} = u - u^h$ belongs to V_2.

Let S_2^h be the space of all piecewise linear functions in V_2 on the same partition of I_L, which was used for S_1^h. Then w^h, the piecewise linear, continuous part of the error $\hat{w}$, belongs to S_2^h.The remainder $w^\infty = \hat{w} - w^h$ is continuous and vanishes at each grid point of the partition.

We now define the Banach space $X = S_2^h \times X^\infty$, where $X^\infty = (I - \Pi^h)V_2$. This gives the fixed–point equation

$$w = T(w), \quad w \in X, \tag{26}$$

where $w = (w^h, w^\infty) = (w_1^h, w_2^h, w_3^h, w_1^\infty, w_2^\infty, w_3^\infty) \in X$ and $T(w) = (T_h(w), T_\infty(w))$ with

$$T_h(w) = (I - A^h)^{-1}(\Pi^h F(u^h + w^h + w^\infty) - (u^h + A^h w^h)), \tag{27}$$

$$T_\infty(w) = (I - \Pi^h)F(u^h + w^h + w^\infty), \tag{28}$$

where A^h describes the Fréchet derivative of $\Pi^h F(v)$ at $v = u^h$, derived in Section 7.2.

6 General Statement of Convergence Conditions

For $w = (w^h, w^\infty) \in X$ we will use the notation

$$\begin{aligned}(w)_{k,i} &= |w_k^h(x_i)| \ \forall k \in \{1,2,3\} \text{ and } \forall i \in \{1,\dots,N-1\} \text{ and} \\ (w)_{k,N} &= ||w_k^\infty||_\infty \ \forall k\{1,2,3\}.\end{aligned}$$

Note that this will rearrange our representation of the Banach space X. Suppose we want to prove the existence of a unique solution of (26) in a set W, referred to as a *candidate set*. Taking a vector $\overline{W} \in \mathbb{R}^{3N}$ with positive components

$$\overline{W} = (W_{1,1},\dots,W_{1,N},W_{2,1},\dots,W_{2,N},W_{3,1},\dots,W_{3,N})^T, \tag{29}$$

a candidate set W is defined by

$$W = \{w \in X | (w)_{k,i} \le W_{k,i}, \ \forall k \in \{1,2,3\} \text{ and } \forall i \in \{1,\dots,N\}\}. \tag{30}$$

In [Ya98], sufficient conditions on W to prove the existence and local uniqueness of a solution to (26) are derived. The proof involves assumptions on bounds of $T(0)$ and T' respectively, where T' is the Fréchet derivative of T. We now formulate the assumptions and the theorem in [Ya98] in three dimensions.

Assumption 1. *There is a vector* $\overline{Y} = (Y_{1,1},\dots,Y_{1,N},Y_{2,1},\dots,Y_{2,N},Y_{3,1},\dots,Y_{3,N})^T$ *with positive components, such that the conditions*

$$(T(0))_{k,i} \le Y_{k,i} \ \forall k \in \{1,2,3\} \text{ and } \forall i \in \{1,\dots,N\}$$

hold.

Assumption 2. *The operator* T *has a Fréchet derivative* T' *with the following property. For any* $\overline{W}$ *there exists a vector* $\overline{Z} = (Z_{1,1},\dots,Z_{1,N},Z_{2,1},\dots,Z_{2,N},Z_{3,1},\dots,Z_{3,N})^T$ *with non–negative components such that the conditions*

$$(T'(\tilde{w})w)_{k,i} \le Z_{k,i} \ \forall k \in \{1,2,3\} \text{ and } \forall i \in \{1,\dots,N\}$$

hold for any w, $\tilde{w} \in W$.

Since the Z_i's satisfying the above inequality depend on $\overline{W}$ in general, we write them as $Z_i(\overline{W})$. Now, define the set K in X by

$$K = \{v \in X | (v)_{k,i} \le Y_{k,i} + Z_{k,i}(\overline{W}) \ \ \forall k \in \{1,2,3\} \text{ and } \forall i \in \{1,\dots,N\}\}$$

Theorem 6.1. *If* $K \subset W$ *holds for the candidate set* W *defined by (30), then there exists a solution to (29) in* K. *Moreover, the solution is unique within the set* W.

The straightforward proof is based on Banach's fixed–point theorem. In the proof, it is shown that the set K includes the image $T(W)$. In the next section we will outline in detail how these bounds can be rigorously verified in computations, and show one approach to obtain a suitable candidate set W.

7 Convergence Conditions for Euler Equations

In this section we derive Y and Z such that Assumption 1 and Assumption 2 hold for the Euler equations. In order to facilitate comparison with the numerical implementation, key expressions are labeled with names corresponding to the names used in the code.

In order to compute candidate vectors satisfying Assumption 1 and 2 respectively, we start with estimations of the operator F and later the Fréchet derivative of F.

7.1 The Operator F

Let us first consider the evaluation of $\Pi^h F(u^h)$. On the interval $[x_i, x_{i+1}]$ we have

$$\begin{aligned}
\rho^h(\tau) &= \rho_i + \frac{\rho_{i+1}-\rho_i}{h_{i+1}}(\tau - x_i),\\
(\rho q)^h(\tau) &= (\rho q)_i + \frac{(\rho q)_{i+1}-(\rho q)_i}{h_{i+1}}(\tau - x_i),\\
E^h(\tau) &= E_i + \frac{E_{i+1}-E_i}{h_{i+1}}(\tau - x_i).
\end{aligned}$$

Of course $u^h = (\rho^h, (\rho q)^h, E^h)^T$, so that

$$\begin{aligned}
\Pi^h F_1(u^h(x_i)) &= \rho_L + \int_{-L}^{x_i}((\rho q)^h(\tau) - s\rho^h(\tau))d\tau \qquad (31)\\
&+ \frac{x_i+L}{2L}\left(\rho_R - \rho_L - \int_{-L}^{L}((\rho q)^h(\tau) s\rho^h(\tau))d\tau\right)\\
&= \rho_L + \sum_{j=1}^{i} h_j \frac{(\rho q)_j + (\rho q)_{j-1}}{2} - s\sum_{j=1}^{i} h_j \frac{\rho_j + \rho_{j-1}}{2}\\
&+ \frac{x_i+L}{2L}\left(\rho_R - \rho_L - \sum_{j=1}^{N} h_j \frac{(\rho q)_j + (\rho q)_{j-1}}{2} + s\sum_{j=1}^{N} h_j \frac{\rho_j + \rho_{j-1}}{2}\right),
\end{aligned}$$

$$\begin{aligned}
\Pi^h F_2(u^h(x_i)) &= (\rho q)_L + \int_{-L}^{x_i}\left(\frac{3-\gamma}{2}\frac{((\rho q)^h(\tau))^2}{\rho^h(\tau)} + (\gamma-1)E^h(\tau) - s(\rho q)^h(\tau)\right)d\tau \qquad (32)\\
&+ \frac{x_i+L}{2L}\left((\rho q)_R - (\rho q)_L - \int_{-L}^{L}\left(\frac{3-\gamma}{2}\frac{((\rho q)^h(\tau))^2}{\rho^h(\tau)} + (\gamma-1)E^h(\tau) - s(\rho q)^h(\tau)\right)d\tau\right)\\
&= (\rho q)_L + \frac{3-\gamma}{2}\sum_{j=1}^{i} a_j + (\gamma-1)\sum_{j=1}^{i} h_j \frac{E_j + E_{j-1}}{2} - s\sum_{j=1}^{i} h_j \frac{(\rho q)_j + (\rho q)_{j-1}}{2} + \frac{x_i+L}{2L}\\
&\times\left((\rho q)_R - (\rho q)_L - \frac{3-\gamma}{2}\sum_{j=1}^{N} a_j - (\gamma-1)\sum_{j=1}^{N} h_j \frac{E_j + E_{j-1}}{2} + s\sum_{j=1}^{N} h_j \frac{(\rho q)_j + (\rho q)_{j-1}}{2}\right),
\end{aligned}$$

where

$$\begin{aligned}
a_j &= \int_{x_{j-1}}^{x_j}\frac{((\rho q)^h(\tau))^2}{\rho^h(\tau)}d\tau = \frac{h_j((\rho q)_j - (\rho q)_{j-1})^2}{\rho_j - \rho_{j-1}}\\
&\times \left(\frac{1}{2} + 2Q_j(\rho q) - Q_j(\rho) - \log(P_j(\rho))[Q_j(\rho q)^2 - Q_j(\rho)(2Q_j(\rho q) - Q_j(\rho))]\right)
\end{aligned}$$

and

$$\begin{aligned}
\Pi^h F_3(u^h(x_i)) &= E_L + \int_{-L}^{x_i}\left(\frac{\gamma E^h(\tau)(\rho q)^h(\tau)}{\rho^h(\tau)} + \frac{1-\gamma}{2}\frac{((\rho q)^h(\tau))^3}{(\rho^h(\tau))^2} - sE^h(\tau)\right)d\tau\\
&+ \frac{x_i+L}{2L}\left(E_R - E_L - \int_{-L}^{L}\left(\frac{\gamma E^h(\tau)(\rho q)^h(\tau)}{\rho^h(\tau)} + \frac{1-\gamma}{2}\frac{((\rho q)^h(\tau))^3}{(\rho^h(\tau))h_j^2} - sE^h(\tau)\right)d\tau\right)\\
&= E_L + \gamma\sum_{j=1}^{i} b_j + \frac{1-\gamma}{2}\sum_{j=1}^{i} c_j - s\sum_{j=1}^{i} h_j \frac{E_j + E_{j-1}}{2}\\
&+ \frac{x_i+L}{2L}\left(E_R - E_L - \gamma\sum_{j=1}^{N} b_j - \frac{1-\gamma}{2}\sum_{j=1}^{N} c_j + s\sum_{j=1}^{N} h_j \frac{E_j + E_{j-1}}{2}\right), \qquad (33)
\end{aligned}$$

where

$$b_j = \int_{x_{j-1}}^{x_j} \frac{E^h(\tau)(\rho q)^h(\tau)}{\rho^h(\tau)} d\tau = \frac{h_j(E_j - E_{j-1})((\rho q)_j - (\rho q)_{j-1})}{\rho_j - \rho_{j-1}}$$
$$\times \left(\frac{1}{2} + Q_j(E) + Q_j(\rho q) - Q_j(\rho) - \log(P_j(\rho))[Q_j(\rho q)Q_j(E) - Q_j(\rho)(Q_j(E) + Q_j(\rho q) - Q_j(\rho))]\right)$$

and

$$c_j = \int_{x_{j-1}}^{x_j} \frac{((\rho q)^h(\tau))^3}{(\rho^h(\tau))^2} d\tau = \frac{h_j((\rho q)_j - (\rho q)_{j-1})^3}{(\rho_j - \rho_{j-1})^2}$$
$$\times \left(\frac{1}{2} + 3Q_j(\rho q) - 2Q_j(\rho) + (Q_j(\rho q) - Q_j(\rho))^2\left(- 3\log(P_j(\rho)) + \left(Q_j(\rho q) - Q_j(\rho)\right)\frac{(\rho_j - \rho_{j-1})^2}{\rho_j \rho_{j-1}}\right)\right).$$

7.2 The Fréchet Derivative of F

We need to compute A^h, which describes the Fréchet derivative of $\Pi^h F(v)$ at $v = u^h$. Since A^h is a linear operator on the finite dimensional space S^h, there is a matrix $\tilde{B}$ such that the coefficient vector of $A^h v^h$ can be expressed as $\tilde{B}\overline{v^h}$. The Fréchet derivative of the operator F is

$$\begin{aligned} F'(u)v = & \left(\int_{-L}^{x} (v_2 - sv_1)d\tau - \frac{x+L}{2L}\int_{-L}^{L}(v_2 - sv_1)d\tau, \int_{-L}^{x}\left[(\gamma-3)\left[\frac{u_2^2 v_1}{2u_1^2} - \frac{u_2 v_2}{u_1}\right] + (\gamma-1)v_3 - sv_2\right]d\tau \right. \\ & - \frac{x+L}{2L}\int_{-L}^{L}\left((\gamma-3)\left(\frac{u_2^2 v_1}{2u_1^2} - \frac{u_2 v_2}{u_1}\right) + (\gamma-1)v_3 - sv_2\right)d\tau, \int_{-L}^{x}\left(\gamma\left(\frac{u_2 v_3}{u_1} + \frac{u_3 v_2}{u_1} - \frac{u_3 u_2 v_1}{u_1^2}\right)\right. \\ & + (\gamma-1)\left(\frac{u_2^3 v_1}{u_1^3} - \frac{3u_2^2 v_2}{2u_1^2}\right) - sv_3\Big)d\tau - \frac{x+L}{2L}\int_{-L}^{L}\left(\gamma\left(\frac{u_2 v_3}{u_1} + \frac{u_3 v_2}{u_1} - \frac{u_3 u_2 v_1}{u_1^2}\right)\right. \\ & \left. + (\gamma-1)\left(\frac{u_2^3 v_1}{u_1^3} - \frac{3u_2^2 v_2}{2u_1^2}\right) - sv_3\Big)d\tau\right). \end{aligned} \tag{34}$$

By replacing v by hat functions and u by $\Pi^h u$ in (34), one can show that

$$\tilde{B} = \begin{pmatrix} -s\alpha_{ij}^* & \alpha_{ij}^* & 0 \\ \frac{\gamma-3}{2}\eta_{ij}^* & -(\gamma-3)\beta_{ij}^* - s\alpha_{ij}^* & (\gamma-1)\alpha_{ij}^* \\ -\gamma\kappa_{ij}^* + (\gamma-1)\lambda_{ij}^* & \gamma\delta_{ij}^* - \frac{3}{2}(\gamma-1)\eta_{ij}^* & \gamma\beta_{ij}^* - s\alpha_{ij}^* \end{pmatrix}_{i,j=1,\ldots,N-1},$$

where

$$\chi_{ij}^* = \chi_{ij} - \frac{x_i + L}{2L}\chi_{Nj}, \qquad \chi_{ij} = \begin{cases} 0 & i < j \\ \chi_j & i = j \\ \chi_j + \hat{\chi}_j & i > j \end{cases}$$

and $\hat{\chi_j}$ is χ_j with h_j replaced by h_{j+1} and the index $j-1$ replaced by the index $j+1$ $\forall i, j \in \{1, \ldots, N-1\}$ and $\forall \chi \in \{\alpha, \beta, \delta, \kappa, \eta, \lambda\}$, where

$$\begin{aligned}
\alpha_j &= h_j/2 \\
\beta_j &= h_j \frac{(\rho q)_j - (\rho q)_{j-1}}{\rho_j - \rho_{j-1}} \left(\frac{1}{2} + (Q_j(\rho q) - Q_j(\rho))[1 + Q_j(\rho) \log(P_j(\rho))] \right) \\
\delta_j &= h_j \frac{E_j - E_{j-1}}{\rho_j - \rho_{j-1}} \left(\frac{1}{2} + (Q_j(E) - Q_j(\rho))[1 + Q_j(\rho) \log(P_j(\rho))] \right) \\
\kappa_j &= h_j \frac{(E_j - E_{j-1})((\rho q)_j - (\rho q)_{j-1})}{(\rho_j - \rho_{j-1})^2} \left(\frac{1}{2} + (Q_j(\rho q) + Q_j(E))(1 + 2\log(P_j(\rho))Q_j(\rho) + P_j(\rho)) \right. \\
&- \left. Q_j(E)Q_j(\rho q)(\log(P_j(\rho)) + 1 - P_j(\rho)) - Q_j(\rho)(3Q_j(\rho)\log(P_j(\rho)) + 2 + P_j(\rho)) \right) \\
\eta_j &= h_j \frac{((\rho q)_j - (\rho q)_{j-1})^2}{(\rho_j - \rho_{j-1})^2} \left(\frac{1}{2} + 2Q_j(\rho q)(1 + 2\log(P_j(\rho))Q_j(\rho) + P_j(\rho)) \right. \\
&- \left. Q_j(\rho q)^2(\log(P_j(\rho)) + 1 - P_j(\rho)) - Q_j(\rho)(3Q_j(\rho)\log(P_j(\rho)) + 2 + P_j(\rho)) \right) \\
\lambda_j &= h_j \frac{((\rho q)_j - (\rho q)_{j-1})^3}{(\rho_j - \rho_{j-1})^3} \left(\frac{1}{2} + (Q_j(\rho q) - Q_j(\rho)) \left[3 - 3\log(P_j(\rho))(Q_j(\rho q) - 2Q_j(\rho)) \right. \right. \\
&+ \left. \left. (Q_j(\rho q) - 7Q_j(\rho) - 6Q_j(\rho)^2) \frac{(\rho_j (\rho q)_{j-1} - \rho_{j-1}(\rho q)_j)(\rho_j - \rho_{j-1})^2}{2\rho_{j-1}\rho_j^2((\rho q)_j - (\rho q)_{j-1})} \right] \right).
\end{aligned}$$

7.3 A Candidate Vector for Satisfying Assumption 1

Consider the construction of Y, which requires bounds on $T(0)$. From (27) we have

$$T_h(0) = \left(I - A^h\right)^{-1} \left(\Pi^h F(u^h) - u^h\right). \tag{35}$$

From (35) we can now obtain bounds on $T(0)$ by solving

$$(I - \tilde{B})\overline{T_h(0)} = \overline{\Pi^h F(u^h)} - \overline{u^h}, \tag{36}$$

and from (28) we have

$$T_\infty(0) = \left(I - \Pi^h\right) F\left(u^h\right). \tag{37}$$

Thus, we can get a bound on $T_\infty(0)$ via the following proposition.

Proposition 7.1. *Suppose $v \in C^1([x_i, x_{i+1}])$, and let v^h be the straight line between the points $(x_i, v(x_i))$ and $(x_{i+1}, v(x_{i+1}))$. Then*

$$\max_{x \in [x_i, x_{i+1}]} |v^h(x) - v(x)| \leq \frac{x_{i+1} - x_i}{2} \max_{x \in [x_i, x_{i+1}]} |v'(x)|.$$

Proof. We have

$$v^h(x) = v(x_i) + \frac{x - x_i}{x_{i+1} - x_i}(v(x_{i+1}) - v(x_i))$$

and

$$v(x_i) = v(x) + (x_i - x)v'(\xi_1) \text{ and } v(x_{i+1}) = v(x) + (x_{i+1} - x)v'(\xi_2),$$

where $\{x, \xi_1, \xi_2\} \subset [x_i, x_{i+1}]$, so that

$$\begin{aligned}
|v^h(x) - v(x)| &= \left| (x_i - x)v'(\xi_1) + \frac{x - x_i}{x_{i+1} - x_i} \left((x_{i+1} - x)v'(\xi_2) - (x_i - x)v'(\xi_1) \right) \right| \\
&\leq \max_{x \in [x_i, x_{i+1}]} |v'(x)| \frac{|(x_i - x)(x_{i+1} - x_i) + (x - x_i)^2| + |(x_{i+1} - x)(x - x_i)|}{x_{i+1} - x_i} \\
&\leq \max_{x \in [x_i, x_{i+1}]} \frac{2|v'(x)|}{x_{i+1} - x_i} \max_{x \in [x_i, x_{i+1}]} (x - x_i)(x_{i+1} - x) = \frac{x_{i+1} - x_i}{2} \max_{x \in [x_i, x_{i+1}]} |v'(x)|.
\end{aligned}$$

□

According to Proposition 7.1 and (37)

$$||(T_\infty(0))_k||_\infty \leq \max_{i\in\{1,\ldots,N\}} \frac{h_i}{2} \max_{x\in[x_{i-1},x_i]} |\frac{d}{dx}F_k(u^h(x))|.$$

According to (21)

$$\frac{d}{dx}F_1(u^h(x)) = \rho q^h(x) - s\rho^h(x) + \frac{1}{2L}\Big(\rho_R - \rho_L - \int_{-L}^{L}(\rho q^h(\tau) - s\rho^h(\tau))d\tau\Big),$$

from which it follows that

$$\max_{x\in[x_{i-1},x_i]} |\frac{d}{dx}F_1(u^h(x))| \leq \max_{k\in\{i-1,i\}} \left|\rho q_k - s\rho_k + \frac{1}{2L}\Big(\rho_R - \rho_L - \int_{-L}^{L}(\rho q^h(\tau) - s\rho^h(\tau))d\tau\Big)\right| \stackrel{\text{def}}{=} g_{1,i}. \quad (38)$$

According to (22), we have

$$\begin{aligned}\frac{d}{dx}F_2(u^h(x)) &= \frac{3-\gamma}{2}\frac{(\rho q^h(x))^2}{\rho^h(x)} + (\gamma-1)E^h(x) - s\rho q^h(x)\\ &+ \frac{1}{2L}\Big((\rho q)_R - (\rho q)_L - \int_{-L}^{L}\Big(\frac{3-\gamma}{2}\frac{(\rho q^h(\tau))^2}{\rho^h(\tau)} + (\gamma-1)E^h(\tau) - s\rho q^h(\tau)\Big)d\tau\Big),\end{aligned}$$

from which it follows that

$$\begin{aligned}\max_{x\in[x_{i-1},x_i]} |\frac{d}{dx}F_2(u^h(x))| &\leq \max_{j,m,n\in\{i-1,i\}} \left|\frac{3-\gamma}{2}(\rho q)_m^2 + \rho_j\Big[(\gamma-1)E_n - s(\rho q)_m\right.\\ &+ \left.\frac{1}{2L}\Big((\rho q)_R - (\rho q)_L - \int_{-L}^{L}\Big(\frac{3-\gamma}{2}\frac{(\rho q^h(\tau))^2}{\rho^h(\tau)} + (\gamma-1)E^h(\tau) - s\rho q^h(\tau)\Big)d\tau\Big)\Big]\right|\\ &/ \quad \min(\rho_{i-1},\rho_i) \stackrel{\text{def}}{=} g_{2,i}. \quad (39)\end{aligned}$$

Finally, following (23), we have

$$\begin{aligned}\frac{d}{dx}F_3(u^h(x)) &= \frac{\gamma E^h(x)\rho q^h(x)}{\rho^h(x)} + \frac{1-\gamma}{2}\frac{(\rho q^h(x))^3}{(\rho^h(x))^2} - sE^h(x)\\ &+ \frac{1}{2L}\Big(E_R - E_L - \int_{-L}^{L}\Big(\frac{\gamma E^h(\tau)\rho q^h(\tau)}{\rho^h(\tau)} + \frac{1-\gamma}{2}\frac{(\rho q^h(\tau))^3}{(\rho^h(\tau))^2} - sE^h(\tau)\Big)d\tau\Big),\end{aligned}$$

so that

$$\begin{aligned}\max_{x\in[x_{i-1},x_i]} |\frac{d}{dx}F_3(u^h(x))| &\leq \max_{j,m,n\in\{i-1,i\}} \left|\frac{1-\gamma}{2}(\rho q)_m^3 + \rho_j\Big(\gamma E_n(\rho q)_m + \rho_j\Big[-sE_n + \frac{1}{2L}\Big(E_R - E_L\right.\\ &- \left.\left.\int_{-L}^{L}\Big(\frac{\gamma E^h(\tau)\rho q^h(\tau)}{\rho^h(\tau)} + \frac{1-\gamma}{2}\frac{(\rho q^h(\tau))^3}{(\rho^h(\tau))^2} - sE^h(\tau)\Big)d\tau\Big)\Big]\Big)\right|\\ &/ \quad \min(\rho_{i-1}^2,\rho_i^2) \stackrel{\text{def}}{=} g_{3,i}. \quad (40)\end{aligned}$$

Note that the integral parts of equations (38), (39) and (40) have already been computed in equations (31), (32) and (33) so they are not written explicitly again.

Using (36) and (38)–(40) we set the candidate vector for Assumption 1 to be

$$\overline{Y} = \begin{pmatrix} |(I-\tilde{B})^{-1}(\overline{\Pi^h F(u^h)} - \overline{u^h})|_{1,1} \\ \vdots \\ |(I-\tilde{B})^{-1}(\overline{\Pi^h F(u^h)} - \overline{u^h})|_{1,N-1} \\ \max_{i\in\{1,\dots,N\}} \frac{h_i}{2} g_{1,i} \\ |(I-\tilde{B})^{-1}(\overline{\Pi^h F(u^h)} - \overline{u^h})|_{2,1} \\ \vdots \\ |(I-\tilde{B})^{-1}(\overline{\Pi^h F(u^h)} - \overline{u^h})|_{2,N-1} \\ \max_{i\in\{1,\dots,N\}} \frac{h_i}{2} g_{2,i} \\ |(I-\tilde{B})^{-1}(\overline{\Pi^h F(u^h)} - \overline{u^h})|_{3,1} \\ \vdots \\ |(I-\tilde{B})^{-1}(\overline{\Pi^h F(u^h)} - \overline{u^h})|_{3,N-1} \\ \max_{i\in\{1,\dots,N\}} \frac{h_i}{2} g_{3,i} \end{pmatrix}, \tag{41}$$

where the expressions for $\Pi^h F$ and $\tilde{B}$ from the preceding subsections are inserted.

7.4 A Candidate Vector for Satisfying Assumption 2

We now derive Z, i.e., we bound $(T'(\tilde{w})w)_i$. From (27) and (28) we have

$$T'_h(\tilde{w})w = (I-A^h)^{-1}(\Pi^h F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty) - A^h w^h), \tag{42}$$
$$T'_\infty(\tilde{w})w = (I-\Pi^h)F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty). \tag{43}$$

Let us start by bounding the expression (42). For A^h, we have already derived the corresponding matrix $\tilde{B}$. From (34) and from using $A^h w^h = \Pi^h F'(u^h)w^h$, together with $w^h, \tilde{w}^h \in S_2^h$ and $(w)_{k,i}, (\tilde{w})_{k,i} \le W_{k,i}$, we have

$$\begin{aligned}
&(\Pi^h F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty) - A^h w^h)_{1,i} \\
=\ & \left| \int_{-L}^{x_i} (w_2^h + w_2^\infty - w_2^h - s(w_1^h + w_1^\infty - w_1^h))d\tau - \frac{x_i+L}{2L}\int_{-L}^{L} (w_2^h + w_2^\infty - w_2^h - s(w_1^h + w_1^\infty - w_1^h))d\tau \right| \\
=\ & \left| \int_{-L}^{x_i} (w_2^\infty - s w_1^\infty)d\tau - \frac{x_i+L}{2L}\int_{-L}^{L} (w_2^\infty - s w_1^\infty)d\tau \right| \le 2(x_i+L)(W_{2,N} + |s|W_{1,N}) \overset{\text{def}}{=} q_{1,i},
\end{aligned} \tag{44}$$

$$\begin{aligned}
&(\Pi^h F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty) - A^h w^h)_{2,i} \\
=\ & \left| \int_{-L}^{x_i} \left[\frac{(\gamma-3)(u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)}{2(u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty)^2} \Big((u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)(w_1^h+w_1^\infty) - 2(u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty)(w_2^h+w_2^\infty) \Big) \right. \right. \\
&+ \left. (\gamma-1)w_3^\infty - s w_2^\infty \right] d\tau - \frac{x_i+L}{2L}\int_{-L}^{L} \left[\frac{(\gamma-3)(u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)}{2(u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty)^2} \Big((u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)(w_1^h+w_1^\infty) \right. \\
&- \left. 2(u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty)(w_2^h+w_2^\infty) \Big) + (\gamma-1)w_3^\infty - s w_2^\infty \right] d\tau - (\gamma-3)\Big(\int_{-L}^{x_i} \frac{u_2^h}{2(u_1^h)^2}(w_1^h u_2^h - 2w_2^h u_1^h)d\tau \\
&- \left. \frac{x_i+L}{2L}\int_{-L}^{L} \frac{u_2^h}{2(u_1^h)^2}(w_1^h u_2^h - 2w_2^h u_1^h)d\tau \Big) \right| \\
\le\ & 2(x_i+L)(|\gamma-1|W_{3,N} + |s|W_{2,N}) + |\gamma-3|\Big(\sum_{j=1}^{i} h_j \Big(\frac{A_{2,j}^2 B_{1,j}}{2F_j^2} + \frac{A_{2,j}B_{2,j}}{F_j} + \frac{C_{2,j}^2 D_{1,j}}{2G_j^2} + \frac{C_{2,j}D_{2,j}}{G_j} \Big) \\
&+ \frac{x_i+L}{2L}\sum_{j=1}^{N} h_j \Big(\frac{A_{2,j}^2 B_{1,j}}{2F_j^2} + \frac{A_{2,j}B_{2,j}}{F_j} + \frac{C_{2,j}^2 D_{1,j}}{2G_j^2} + \frac{C_{2,j}D_{2,j}}{G_j} \Big) \Big) \overset{\text{def}}{=} q_{2,i}
\end{aligned} \tag{45}$$

and

$$
\begin{aligned}
&(\Pi^h F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty)-A^h w^h)_{3,i}\\
&= \Big|-s\Big(\int_{-L}^{x_i} w_3^\infty d\tau-\frac{x_i+L}{2L}\int_{-L}^{L} w_3^\infty d\tau\Big)\\
&\quad+\gamma\Big(\int_{-L}^{x_i}\Big(\frac{(u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)(\tilde{w}_3^h+\tilde{w}_3^\infty)}{u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty}+\frac{(u_3^h+\tilde{w}_3^h+\tilde{w}_3^\infty)(\tilde{w}_2^h+\tilde{w}_2^\infty)}{u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty}\\
&\quad-\frac{(u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)(u_3^h+\tilde{w}_3^h+\tilde{w}_3^\infty)(\tilde{w}_1^h+\tilde{w}_1^\infty)}{(u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty)^2}\Big)d\tau-\frac{x_i+L}{2L}\int_{-L}^{L}\Big(\frac{(u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)(\tilde{w}_3^h+\tilde{w}_3^\infty)}{u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty}\\
&\quad+\frac{(u_3^h+\tilde{w}_3^h+\tilde{w}_3^\infty)(\tilde{w}_2^h+\tilde{w}_2^\infty)}{u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty}-\frac{(u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)(u_3^h+\tilde{w}_3^h+\tilde{w}_3^\infty)(\tilde{w}_1^h+\tilde{w}_1^\infty)}{(u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty)^2}\Big)d\tau\\
&\quad+\int_{-L}^{x_i}\Big(\frac{u_2^h w_3^h}{u_1^h}+\frac{u_3^h w_2^h}{u_1^h}-\frac{u_3^h u_2^h w_1^h}{(u_1^h)^2}\Big)d\tau-\frac{x_i+L}{2L}\int_{-L}^{L}\Big(\frac{u_2^h w_3^h}{u_1^h}+\frac{u_3^h w_2^h}{u_1^h}-\frac{u_3^h u_2^h w_1^h}{(u_1^h)^2}\Big)d\tau\Big)\\
&\quad+(\gamma-1)\Big(\int_{-L}^{x_i}\frac{(u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)^2}{(u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty)^3}\Big((u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)(w_1^h+w_1^\infty)-\frac{3}{2}(u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty)(w_2^h+w_2^\infty)\Big)d\tau\\
&\quad-\frac{x_i+L}{2L}\int_{-L}^{L}\frac{(u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)^2}{(u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty)^3}\Big((u_2^h+\tilde{w}_2^h+\tilde{w}_2^\infty)(w_1^h+w_1^\infty)-\frac{3}{2}(u_1^h+\tilde{w}_1^h+\tilde{w}_1^\infty)(w_2^h+w_2^\infty)\Big)d\tau\\
&\quad+\int_{-L}^{x_i}\frac{(u_2^h)^2}{(u_1^h)^3}\Big(u_2^h w_1^h-\frac{3}{2}u_1^h w_2^h\Big)d\tau-\frac{x_i+L}{2L}\int_{-L}^{L}\frac{(u_2^h)^2}{(u_1^h)^3}\Big(u_2^h w_1^h-\frac{3}{2}u_1^h w_2^h\Big)d\tau\Big)\Big|\\
&\leq 2(x_i+L)|s|W_{3,N}\\
&\quad+\sum_{j=1}^{i}h_j\Big(\gamma\Big(\frac{A_{2,j}B_{3,j}+A_{3,j}B_{2,j}}{F_j}+\frac{A_{2,j}A_{3,j}B_{1,j}}{F_j^2}+\frac{C_{2,j}D_{3,j}+C_{3,j}D_{2,j}}{G_j}+\frac{C_{2,j}C_{3,j}D_{1,j}}{G_j^2}\Big)\\
&\quad+|\gamma-1|\Big(\frac{A_{2,j}^3B_{1,j}}{F_j^3}+\frac{3A_{2,j}^2B_{2,j}}{2F_j^2}+\frac{C_{2,j}^3D_{1,j}}{G_j^3}+\frac{3C_{2,j}^2D_{2,j}}{2G_j^2}\Big)\Big)\\
&\quad+\frac{x_i+L}{2L}\Big(\sum_{j=1}^{N}h_j\Big(\gamma\Big(\frac{A_{2,j}B_{3,j}+A_{3,j}B_{2,j}}{F_j}+\frac{A_{2,j}A_{3,j}B_{1,j}}{F_j^2}+\frac{C_{2,j}D_{3,j}+C_{3,j}D_{2,j}}{G_j}+\frac{C_{2,j}C_{3,j}D_{1,j}}{G_j^2}\Big)\\
&\quad+|\gamma-1|\Big(\frac{A_{2,j}^3B_{1,j}}{F_j^3}+\frac{3A_{2,j}^2B_{2,j}}{2F_j^2}+\frac{C_{2,j}^3D_{1,j}}{G_j^3}+\frac{3C_{2,j}^2D_{2,j}}{2G_j^2}\Big)\Big)\Big)\stackrel{\text{def}}{=} q_{3,i},
\end{aligned}
\tag{46}
$$

where

$$
\begin{aligned}
A_{k,j} &= \max(|u_{k,j-1}|+W_{k,j-1},|u_{k,j}|+W_{k,j})+W_{k,N},\\
B_{k,j} &= \max(W_{k,j-1},W_{k,j})+W_{k,N},\\
C_{k,j} &= \max(|u_{k,j-1}|,|u_{k,j}|),\\
D_{k,j} &= \max(W_{k,j-1},W_{k,j}),\\
F_j &= \min(u_{1,j-1},u_{1,j})-\max(W_{1,j-1},W_{1,j})-W_{1,N},\\
G_j &= \min(u_{1,j-1},u_{1,j}).
\end{aligned}
\tag{47}
$$

We now move on to (43). Using Proposition 7.1 again to estimate the interpolation error, we get according to (34)

$$
\begin{aligned}
&||[(I-\Pi^h)F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty)]_1||_\infty\\
&\quad\leq \max_{j\in\{1,\ldots,N-1\}}\frac{h_j}{2}\max_{x_{j-1}\leq x\leq x_j}|\frac{d}{dx}[F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty)]_1|\\
&\quad\leq \max_{j\in\{1,\ldots,N-1\}}\frac{h_j}{2}\max_{x_{j-1}\leq x\leq x_j}|w_2^h+w_2^\infty-s(w_1^h+w_1^\infty)-\frac{1}{2L}\int_{-L}^{L}(w_2^h+w_2^\infty-s(w_1^h+w_1^\infty))d\tau|\\
&\quad\leq \max_{j\in\{1,\ldots,N-1\}}\frac{h_j}{2}(B_{2,j}+|s|B_{1,j}+\frac{1}{2L}\sum_{j=1}^{N}h_j(B_{2,j}+|s|B_{1,j}))\stackrel{\text{def}}{=} Z_{1,N}.
\end{aligned}
$$

$$
\begin{aligned}
&||[(I-\Pi^h)F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty)]_2||_\infty \\
&\quad \le \max_{j\in\{1,\ldots,N-1\}} \frac{h_j}{2} \max_{x_{j-1}\le x\le x_j} |\frac{d}{dx}[F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty)]_2| \\
&\quad \le \max_{j\in 1,\ldots,N-1} \frac{h_j}{2}\left(|\gamma-3|\left(\frac{A_{2,j}^2B_{1,j}}{2F_j^2}+\frac{A_{2,j}B_{2,j}}{F_j}\right)+|\gamma-1|B_{3,j}+|s|B_{2,j}\right. \\
&\qquad + \left.\frac{1}{2L}\sum_{j=1}^{N} h_j\left(|\gamma-3|\left(\frac{A_{2,j}^2B_{1,j}}{2F_j^2}+\frac{A_{2,j}B_{2,j}}{F_j}\right)+|\gamma-1|B_{3,j}+|s|B_{2,j}\right)\right) \stackrel{\text{def}}{=} Z_{2,N}.
\end{aligned}
$$

$$
\begin{aligned}
&||[(I-\Pi^h)F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty)]_3||_\infty \\
&\quad \le \max_{j\in\{1,\ldots,N-1\}} \frac{h_j}{2} \max_{x_{j-1}\le x\le x_j} |\frac{d}{dx}[F'(u^h+\tilde{w}^h+\tilde{w}^\infty)(w^h+w^\infty)]_3| \\
&\quad \le \max_{j\in 1,\ldots,N-1} \frac{h_j}{2}\left(\gamma\left(\frac{A_{2,j}B_{3,j}+A_{3,j}B_{2,j}}{F_j}+\frac{A_{2,j}A_{3,j}B_{1,j}}{F_j^2}\right)+|\gamma-1|\left(\frac{A_{2,j}^3B_{1,j}}{F_j^3}+\frac{3A_{2,j}^2B_{2,j}}{2F_j^2}\right)+|s|B_{3,j}\right. \\
&\qquad + \left.\frac{1}{2L}\sum_{j=1}^{N} h_j\left(\gamma\left(\frac{A_{2,j}B_{3,j}+A_{3,j}B_{2,j}}{F_j}+\frac{A_{2,j}A_{3,j}B_{1,j}}{F_j^2}\right)+|\gamma-1|\left(\frac{A_{2,j}^3B_{1,j}}{F_j^3}+\frac{3A_{2,j}^2B_{2,j}}{2F_j^2}\right)+|s|B_{3,j}\right)\right) \\
&\quad \stackrel{\text{def}}{=} Z_{3,N}.
\end{aligned}
$$

Now define

$$
\overline{q} \stackrel{\text{def}}{=} (q_{1,1},\ldots,q_{1,N-1},q_{2,1},\ldots,q_{2,N-1},q_{3,1},\ldots,q_{3,N-1})^T.
$$

Then

$$
\overline{Z}(\overline{W}) = \begin{pmatrix}
(|(I-\tilde{B})^{-1}|\overline{q})_{1,1} \\
\vdots \\
(|(I-\tilde{B})^{-1}|\overline{q})_{1,N-1} \\
Z_{1,N} \\
(|(I-\tilde{B})^{-1}|\overline{q})_{2,1} \\
\vdots \\
(|(I-\tilde{B})^{-1}|\overline{q})_{2,N-1} \\
Z_{2,N} \\
(|(I-\tilde{B})^{-1}|\overline{q})_{3,1} \\
\vdots \\
(|(I-\tilde{B})^{-1}|\overline{q})_{3,N-1} \\
Z_{3,N}
\end{pmatrix}
$$

satisfies Assumption 2.

8 The main algorithm

The main routine of the algorithm is called `maineuler`. It begins by calling the subroutine `semidiseuler`, in which the time–dependent version of equation (3), i.e.,

$$
y_t + (f(y)-sy)_x = y_{xx}
$$

is solved up to some user–defined stop time, at which steady–state is approximately reached. Let u^h denote the approximate solution obtained from `semidiseuler`. Using splines, the values of u^h are computed on a user–defined mesh.

Next, the subroutine `Yeuler` computes Y. This is done by calling `Beuler` – which constructs the matrix $\tilde{B}$, followed by `Feuler` – which computes $\Pi^h F(u^h)$. Following this, the main routine `maineuler` attempts to find a suitable candidate set W via the following bootstrap method:

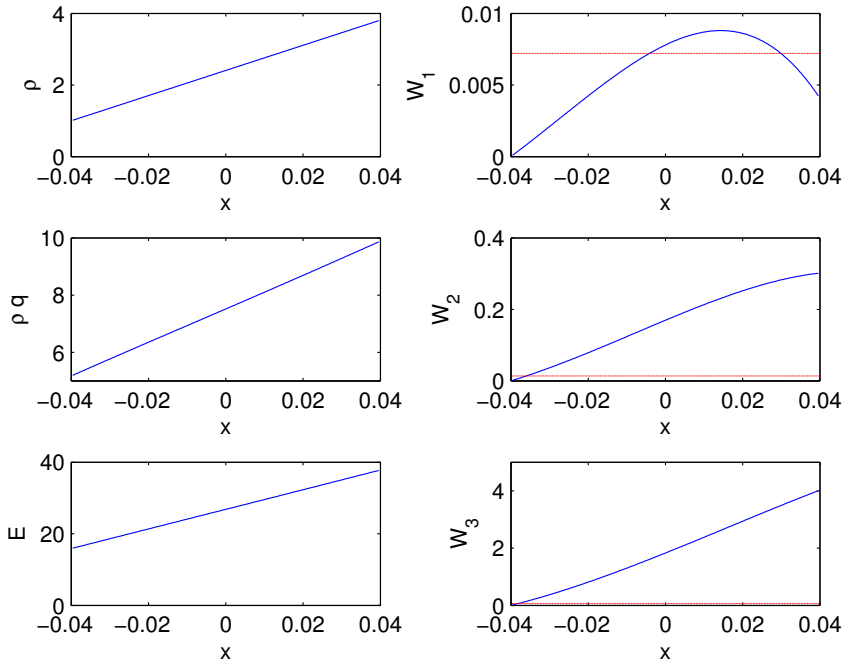

Figure 1: The three u and W components as functions of x at time $t = 5$ for a uniform mesh with 200 intervals, $L = 0.04$ and the standard parameter values. The red lines shows the W^∞ components.

$m \Leftarrow 0$
$W_{k,i}^{(m)} \Leftarrow Z_{k,i}(\bar{0})\ \forall k \in \{1,2,3\}$ and $i \in \{1,\ldots,N\}$
while $Y_{k,i} + Z_{k,i}(\overline{W}^{(m)}) \geq W_{k,i}^{(m)}$ for some $k \in \{1,2,3\}$ and $i \in \{1,\ldots,N\}$
  $m \Leftarrow m + 1$
  $W_{k,i}^{(m)} \Leftarrow (1+\delta)(Y_{k,i} + Z_{k,i}(\overline{W}^{(m-1)}))\ \forall k \in \{1,2,3\}$ and $i \in \{1,\ldots,N\}$
end.

In each iteration W is inflated (we use $\delta = 0.01$), and $Z = Z(W)$ is recomputed by the subroutine `Zeuler`, which also calls `Beuler`. If the convergence criterion $K = Y + Z \subset W$ is reached, then the existence and local uniqueness of a solution to the Euler equations have been proved, according to Theorem 6.1. If the maximal number of iterations is reached, the main routine returns an error message. This also happens when the components of W_1 become too large, rendering the estimates including F_j (see (47)) invalid. All interval computations are performed using the free MATLAB package INTLAB [In].

9 Result and Discussion

The computations described above were performed using the standard parameter values ($L = 0.04$, $\gamma = 1.4$, $\rho_L = 1$, $q_L = 5.17$, $p_L = 1$ and $p_R = 10$) over a uniform mesh with 200 intervals. The approximate steady–state solution was obtained by solving the time dependent system up to time $t = 5$. The main routine established the necessary bounds for Theorem 3.1 after 12 iterations in the bootstrap subroutine. Some of the obtained information is presented in Figure 1.

Keeping all other parameter values fixed, we perform the same computations for decreasing values of p_R. Note that, as this parameter approaches $p_L = 1$, we get closer to violating the entropy condition. This makes

the necessary bounds harder to establish, which is indicated by the increasing number of iterations needed for the bootstrap subroutine to succeed, see Table 1.

p_R	15	13	12	10	8
iterates	12	15	17	25	42*

Table 1: The number of bootstrap iterations needed for different values of p_R. For $p_R = 8$, the mesh size was increased to 800.

We have only been successful in proving the existence and local uniqueness to equation (16) for small values of L. This corresponds to having a large artificial viscosity, and therefore the solutions are very far away from displaying shocks, as is clear from Figure 1. In Figure 2, we illustrate the solution for the more interesting value $L = 5$. Unfortunately, for this value of L, we have no rigorous results.

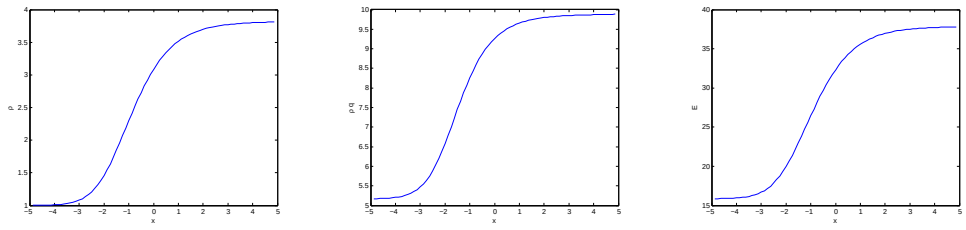

Figure 2: Plots of ρ, ρq and E, for the standard parameter values except that $L = 5$. The values of the number of grid intervals is 80, and the flow time is $t = 30$.

There are several ways to improve our algorithm. The function spaces do not necessarily have to be spanned by hat functions; perhaps other basis functions are more appropriate. This change would probably increase the complexity of each step of the calculations, but it should also increase speed and accuracy. The bottleneck of our computations is the inversion of the matrix $I - \tilde{B}$. Here $\tilde{B}$ consists of nine blocks where each block is a sum of a lower triangular matrix and an outer product. Utilizing the Sherman–Morrison formula [Pr02], we may thus increase the speed of the inversion.

In future work we will study time–dependent problems. These are more interesting seeing that they can describe more complicated physical phenomena. As an example one could consider the time–dependent version of this problem with the method described in [MZ01].

References

[BF09] M. Bernot, A. Figalli and F. Santambrogio, Generalized solutions for the Euler equations in one and two dimensions, *Journal de Mathématiques Pures et Appliqués* **91** (2) (2009), pp. 137–155.

[BM98] M. Berz, K. Makino, Verified Integration of ODEs and Flows using Differential Algebraic Methods on High–Order Taylor Models, *Reliable Computing* **4** (4) (1998), pp. 361–369.

[Br00] A. Bressan, Hyperbolic Systems of Conservation Laws, Oxford University Press, (2000).

[Br02] A. Bressan, Viscosity Solutions of Nonlinear Hyperbolic Systems, *In T. Hou and E. Tadmor, editors. Hyperbolic Problems: Theory, Numerics, Applications* (2002), pp. 19–41.

[BS94] S. Brenner and R. Scott, The Mathematical Theory of Finite Element Methods, Springer–Verlag, (1994).

[Bu74] J.M. Burgers, The Nonlinear Diffusion Equation: Asymptotic Solutions and Statistical Problems, Reidel, (1974).

[CF48] R. Courant and K.O. Friedrichs, Supersonic Flow and Shock Waves, Interscience Publishers, (1948).

[CM92] A. J. Chorin and J. E. Marsden, A Mathematical Introduction to Fluid Mechanics, Third edition, Springer–Verlag, (1992).

[Du88] V. Dutta, Solution of one–dimensional euler–equations by the implicit finite difference technique of beam and warming, Technical Report, National Aeronautical Laboratory, (1988).

[Fa95] R. Farzaneh, A Computer Generated Proof for the Existence of Periodic Orbits for Three–Dimensional Vector Fields, Ph.D. Thesis, Cornell, (1995).

[Gl65] J. Glimm, Solutions in the large for certain nonlinear hyperbolic systems of equations, *Comm. Pure Appl. Math.* **18** (1965), pp. 679–715.

[HR02] H. Holden and N.H. Risebro, Front Tracking for Hyperbolic Conservation Laws, Springer–Verlag, (2002).

[Hu06] J. Hudson, A Review on the Numerical Solution of the 1D Euler equations, MIMS EPrint: 2006.9, (2006).

[In] INTLAB – INTerval LABoratory Version 5.3.
Available from `http://www.ti3.tu--harburg.de/rump/intlab/`.

[Jo02] S. Johansson, Numerical Solutions of the Linearized Euler Equations Using High Order Finite Difference Operators with the Summation by Parts Property, Tech. Rep. 2002–034, Department of Information Technology, Uppsala University, (2002).

[KK86] G. Kreiss and H–O Kreiss, Convergence to Steady State of Solutions of Burgers' Equations, *Appl. Numer. Math.* **2** (1986), pp. 161–179.

[KK98] G. Kreiss and H–O Kreiss, Stability of Steady State Solutions of Burgers' Equation, *SIAM J. Numer. Anal.* **35** (6) (1998), pp. 2329–2349.

[KK04] G. Kreiss, H–O Kreiss and J.Lorenz, Stability of Viscous Shocks on Finite Intervals, Proceedings of the Tenth International Conference on Hyperbolic Problems: Theory, Numerics, Applications, (2004).

[KL89] H–O. Kreiss and J. Lorenz, Initial–Boundary Value Problems and the Navier–Stokes Equations, *Pure and Applied Mathematics* **136**, Academic Press, (1989).

[KS01] G. Kreiss and M. Siklosi, Proving Existence of Solutions of Nonlinear Differential Equations Using Numerical Approximations, *In H. Freistühler and G. Warnecke, editors. Hyperbolic Problems: Theory, Numerics and Applications Eight International Conference in Magdeburg February/March 2000* Birkhäuser, (2001).

[La69] O. Ladyzhenskaya, The Mathematical Theory of Viscous Incompressible Flow, 2nd Edition, Gordon and Breach, (1969).

[La73] P. D. Lax, Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves, Society for Industrial and Applied Mathematics, (1973).

[Le34] J. Leray, Sur le mouvement d'un liquide visqueux emplissant l'espace, *Acta Mathematica* **63** (1) (1934), pp. 193–248.

[MZ01] K. Mischaikow and P. Zgliczyński, Rigorous Numerics for Partial Differential Equations: the Kuramoto–Sivashinsky Equation, *Found. Comput. Math.* **1** (3) (2001), pp. 255–288.

[Na92] M. T. Nakao, A numerical verification method for the existence of weak solutions for nonlinear boundary value problems, *J. Math. Anal. and Appl.* **164** (2) (1992), pp. 489–507.

[Pl01] M. Plum, Computer–assisted enclosure methods for elliptic differential equations, *Linear Algebra and its Applications* **324** (2001), pp. 147–187.

[Pr02] W. H. Press et. al., Numerical Recipes in C++: The Art of Scientific Computing, Second Edition, Cambridge University Press, (2002).

[Qu96] J. Quansen, Existence of weak solutions of 2–D Euler equations with initial vorticity, *Acta Mathematicae Applicatae Sinica* **12** (2) (1996), pp. 121–129.

[Ra92] K. S. Ravichandran, Numerical experiments in 1-D Euler Equations Using Higher Order Schemes, *Aeronautical Society of India* **44** (3) (1992), pp. 233–243.

[SA04] M. Siklosi and P–O. Åsén, On a Computer–Assisted Method for Proving Existence of Solutions of Boundary Value Problems, KTH–Numerical Analysis and Computer Science, October, (2004).

[Sm83] J. Smoller, Shock Waves and Reaction–Diffusion Equation, Springer–Verlag, (1983).

[Sm98] S. Smale, Mathematical Problems for the Next Century, Math. Intelligencer **20** (2) (1998), pp. 7–15.

[Tu02] W. Tucker, A Rigorous ODE Solver and Smale's 14th Problem, *Foundations of Computational Mathematics*, 2:1 (2002), pp. 53–117.

[Ya98] N. Yamamoto, A Numerical Verification Method for Solutions of Boundary Value Problems with Local Uniqueness by Banach's Fixed Point Theorem, *SIAM Journal on Numerical Analysis*, **35** (5) (1998), pp 2004–2013.

[Za05] S. Zahedi, A Fixed Point Method for Proving Existence of Viscous Shock Waves, TRITA–NA–E05171, KTH, (2005)

[Zg02] P. Zgliczyński, Attracting Fixed Points for the Kuramoto–Sivashinsky Equation: A Computer Assisted Proof, *SIAM Journal on Applied Dynamical Systems* **1** (2) (2002), pp. 215–235.

[As04] P–O. Åsén, A Proof of a Resolvent Estimate for Plane Couette Flow by New Analytical and Numerical Techniques, TRITA–NA–0427, KTH, (2004).

Appendix

Here we list all MATLAB code used in implementing the algorithm. The tilde sign means that a row is terminated in the listing, but not in the corresponding M–file.

maineuler.m

```
function [K,W]=maineuler(pl,rl,ql,pr,g,t,x,k,delta,iter)
% Proves existence and local uniqueness of solutions to the equation (f(u)-su)_x=u_{xx}.
% Input:  pl    = pressure at the left boundary
%         rl    = density (\rho) at the right boundary
%         ql    = speed at the left boundary
%         pr    = pressure at the right boundary
%         g     = thermodynamic constant \gamma=C_p/C_v=7/5 for diatomic gases
%         t     = time that the semi-discretized boundary value problem will
%                 run until an approximate steady-state is reached
%         x     = starting mesh for the main algorithm
%         k     = changes the uniform mesh to a mesh with more points in the
%                 middle for k>1 and less points in the middle if k<1 or
%                 more points on the right if k<1 and on the left if k>1.
%                 The user may change this by flipping the % between the
%                 lines which begin with xnu.
%         delta = algorithm parameter describing how much the search domain
%                 should be increased in every step.
%         iter  = number of iterations
%         Typical values 1,1,5.17,10,1.4,10,[-5:0.2:5],1,0.01,5
% Output: W     = candidate set written in vector form
%         K     = a set which must be contained in the candidate set written in
%                 vector form
% There is an output if and only if we have a proof.

format long
convergence=0;
s=size(x);
if s(1)<s(2)
  x=x';
end
disp('Solving the boundary value differential equation...')
L=x(end);
N=length(x)-1;
xshort=x(2:end-1);
u=semidiseuler(pl,rl,ql,pr,g,N,L,t); % Approximate solution through semi-discretization of a boundary value~
problem which runs until steady-state.
subplot(3,2,1)
plot(x(2:end-1),u(1:end/3))
xlabel('x')
ylabel('\rho')
subplot(3,2,3)
plot(x(2:end-1),u(1+end/3:2*end/3))
xlabel('x')
ylabel('\rho q')
subplot(3,2,5)
plot(x(2:end-1),u(2*end/3+1:end))
xlabel('x')
ylabel('E')
delta=intval(delta);
% Put % in front of exactly one of the following two lines.
xnu=sign(x).*abs(x).^k/L^(k-1); % For k>1 finer mesh in the middle
%xnu=((x+L)/2/L).^k*2*L-L; % For 0<k<1 finer mesh on the right
u=[spline(x(2:end-1),u(1:end/3),xnu(2:end-1));spline(x(2:end-1),u(1+end/3:2*end/3),xnu(2:end-1));~
spline(x(2:end-1),u(1+2*end/3:end),xnu(2:end-1))];
m=(g-1)/(g+1);
rr=rl*(pr+m*pl)/(pl+m*pr); % The right boundary of \rho
r=[rl;u(1:end/3);rr];
disp('Computing Y...')
g=intval(g);
pl=intval(pl);
rl=intval(rl);
ql=intval(ql);
pr=intval(pr);
delta=intval(delta);
```

```
x=intval(x);
L=intval(L);
k=intval(k);
% Put % in front of exactly one of the following two lines.
xnu=sign(mid(x)).*abs(x).^k/L^(k-1); % For k>1 finer mesh in the middle
%xnu=((x+L)/2/L).^k*2*L-L; % For 0<k<1 finer mesh on the right
u=intval(u);
Y=Yeuler(xnu,u,L,g,pl,rl,ql,pr);       % For Assumption 1
max_of_Y1h=ma(Y(1:end/3-1))
Y1inf=Y(end/3)
max_of_Y2h=ma(Y(end/3+1:2*end/3-1))
Y2inf=Y(2*end/3)
max_of_Y3h=ma(Y(2*end/3+1:end-1))
Y3inf=Y(end)
Z=zeros(3*N,1);
for m=1:iter
  disp(['Iteration ' int2str(m) '...'])
  W=(1+delta)*(Y+Z); % W is increased.???
  W1h=W(1:end/3-1);
  W1h=W1h.sup;
  max_of_W1h=max(W1h)
  W1inf=W(end/3)
  subplot(3,2,2)
  plot(xshort,W1h)
  xlabel('x')
  ylabel('W_1')
  hold on
  plot(xshort,W1inf.sup,'r')
  hold off
  W2h=W(end/3+1:2*end/3-1);
  W2h=W2h.sup;
  max_of_W2h=max(W2h)
  W2inf=W(2*end/3)
  subplot(3,2,4)
  plot(xshort,W2h)
  xlabel('x')
  ylabel('W_2')
  hold on
  plot(xshort,W2inf.sup,'r')
  hold off
  W3h=W(2*end/3+1:end-1);
  W3h=W3h.sup;
  max_of_W3h=max(W3h)
  W3inf=W(end)
  subplot(3,2,6)
  plot(xshort,W3h)
  xlabel('x')
  ylabel('W_3')
  hold on
  plot(xshort,W3inf.sup,'r')
  hold off
  drawnow
  time=cputime;
  Z=Zeuler(xnu,u,L,g,pl,rl,ql,pr,W); % For Assumption 2
  if ~sum((Y.sup+Z.sup)>W.inf) % Is Assumption 2 fulfilled?
    convergence=1;
    break
  end
  time_for_iteration=cputime-time
end
if(convergence)
  K=Y.sup+Z.sup;
  W=W.sup;
else
  error('No convergence')
end
```

semidiseuler.m

```
function u=semidiseuler(pl,rl,ql,pr,g,N,L,t)
% Solves u_t+(f(u))_x=u_{xx} with the unknowns
```

```
% u=(u1,u2,u3)=(Density,Density*Speed,Energy density)=(\rho,\rho q,E),
% discretizing in space and solving the corresponding ode system using the
% MATLAB solvers ode15s, ode45, ode23, ode113, ode23t, ode15s, ode23s or ode23tb.
% Input:  pl    = pressure at the left boundary
%         rl    = density (\rho) at the right boundary
%         ql    = speed at the left boundary
%         pr    = pressure at the right boundary
%         g     = thermodynamic constant \gamma=C_p/C_v=7/5 for diatomic gases
%         N     = number of intervals
%         L     = the half length of the domain
%         t     = time that the semi-discretized boundary value problem will
%                 run until an approximate steady-state is reached
% Output: u     = approximate solution of the equation (f(u)-su)_x=u_{xx}

global plg rlg qlg prg gg Ng Lg
gg=g;
plg=pl;
rlg=rl;
qlg=ql;
prg=pr;
Ng=N;
Lg=L;

rql=rlg*qlg % Speed*Density at left boundary
El=plg/(gg-1)+rlg*qlg^2/2 % Energy density at the left boundary
m=(gg-1)/(gg+1);
rr=rlg*(prg+m*plg)/(plg+m*prg) % Density at the right boundary
qr=qlg+(1-m)*(plg*sqrt(gg*(1+m)/rlg/(prg+m*plg))-sqrt(gg*(prg+m*plg)/rlg/(1+m)));
rqr=rr*qr % Speed*Density at the right boundary
Er=prg/(gg-1)+rr*qr^2/2 % Energy at the right boundary

h=2*Lg/Ng;
% Third degree polynomial initial guess
% Arrangement: (r1,rq1,E1,r2,rq2,E2,...)
u0(1:3:3*Ng-5)=(rr+rlg)/2+(rr-rlg)*([-L+h:h:L-h]/L).*(3-([-L+h:h:L-h]/L).^2)/4;
u0(2:3:3*Ng-4)=(rqr+rql)/2+(rqr-rql)*([-L+h:h:L-h]/L).*(3-([-L+h:h:L-h]/L).^2)/4;
u0(3:3:3*Ng-3)=(Er+El)/2+(Er-El)*([-L+h:h:L-h]/L).*(3-([-L+h:h:L-h]/L).^2)/4;

z=ode15s(@geuler,[0 t],u0'); % Solves the ode system dy/dt=geuler(t,y). Choose
%ode15s, ode45, ode23, ode113, ode23t, ode15s, ode23s or ode23tb.
u=z.y(:,end);                % Chooses the last time step

% Arrangement: (r1,...,rn,rq1,...,rqn,E1,...,En)
u=[u(1:3:length(u))' u(2:3:length(u))'  u(3:3:length(u))'];
```

geuler.m

```
function f=geuler(t,y)
% The function semidiseuler solves the ode system dy/dt=geuler(t,y).
% Input:  t = time
%         y = u
% Output: f = right-hand side of dy/dt=geuler(t,y).

global plg rlg qlg prg gg Ng Lg

rql=rlg*qlg; % Speed*Density at left boundary
El=plg/(gg-1)+rlg*qlg^2/2; % Energy density at the left boundary
m=(gg-1)/(gg+1);
s=qlg-sqrt(gg*(prg+m*plg)/rlg/(1+m));
rr=rlg*(prg+m*plg)/(plg+m*prg); % Density at the right boundary
qr=qlg+(1-m)*(plg*sqrt(gg*(1+m)/rlg/(prg+m*plg))-sqrt(gg*(prg+m*plg)/rlg/(1+m)));
rqr=rr*qr; % Speed*Density at the right boundary
Er=prg/(gg-1)+rr*qr^2/2; % Energy density at the right boundary

h=2*Lg/Ng;
f=zeros(1,3*Ng-3); % Allocation
% Inner \rho
f(4:3:3*Ng-8)=(y(1:3:3*Ng-11)-2*y(4:3:3*Ng-8)+y(7:3:3*Ng-5))/h^2-(y(8:3:3*Ng-4)-y(2:3:3*Ng-10))/2/h+~
s*(y(7:3:3*Ng-5)-y(1:3:3*Ng-11))/2/h;

% Inner \rho q
```

```
f(5:3:3*Ng-7)=(y(2:3:3*Ng-10)-2*y(5:3:3*Ng-7)+y(8:3:3*Ng-4))/h^2+(gg-3)*(y(8:3:3*Ng-4).^2./y(7:3:3*Ng-5)~
-y(2:3:3*Ng-10).^2./y(1:3:3*Ng-11))/4/h+(1-gg)*(y(9:3:3*Ng-3)-y(3:3:3*Ng-9))/2/h+s*(y(8:3:3*Ng-4)~
-y(2:3:3*Ng-10))/2/h;

% Inner E
f(6:3:3*Ng-6)=(y(3:3:3*Ng-9)-2*y(6:3:3*Ng-6)+y(9:3:3*Ng-3))/h^2+(gg-1)*(y(8:3:3*Ng-4).^3./y(7:3:3*Ng-5).^2~
-y(2:3:3*Ng-10).^3./y(1:3:3*Ng-11).^2)/4/h-gg*(y(9:3:3*Ng-3).*y(8:3:3*Ng-4)./y(7:3:3*Ng-5)-y(3:3:3*Ng-9)~
.*y(2:3:3*Ng-10)./y(1:3:3*Ng-11))/2/h+s*(y(9:3:3*Ng-3)-y(3:3:3*Ng-9))/2/h;

% Left boundary \rho
f(1)=(rlg-2*y(1)+y(4))/h^2-(y(5)-rql)/2/h+s*(y(4)-rlg)/2/h;

% Left boundary \rho q
f(2)=(rql-2*y(2)+y(5))/h^2+(gg-3)*(y(5)^2/y(4)-rql^2/rlg)/4/h+(1-gg)*(y(6)-El)/2/h+s*(y(5)-rql)/2/h;

% Left boundary E
f(3)=(El-2*y(3)+y(6))/h^2+(gg-1)*(y(5)^3/y(4)^2-rql^3/rlg^2)/4/h-gg*(y(6)*y(5)/y(4)-El*rql/rlg)/2/h~
+s*(y(6)-El)/2/h;

% Right boundary \rho
f(3*Ng-5)=(y(3*Ng-8)-2*y(3*Ng-5)+rr)/h^2-(rqr-y(3*Ng-7))/2/h+s*(rr-y(3*Ng-8))/2/h;

% Right boundary \rho q
f(3*Ng-4)=(y(3*Ng-7)-2*y(3*Ng-4)+rqr)/h^2+(gg-3)*(rqr^2/rr-y(3*Ng-7)^2/y(3*Ng-8))/4/h+(1-gg)~
*(Er-y(3*Ng-6))/2/h+s*(rqr-y(3*Ng-7))/2/h;

% Right boundary E
f(3*Ng-3)=(y(3*Ng-6)-2*y(3*Ng-3)+Er)/h^2+(gg-1)*(rqr^3/rr^2-y(3*Ng-7)^3/y(3*Ng-8)^2)/4/h~
-gg*(Er*rqr/rr-y(3*Ng-6)*y(3*Ng-7)/y(3*Ng-8))/2/h+s*(Er-y(3*Ng-6))/2/h;

f=f';
```

Yeuler.m

```
function Yint=Yeuler(xint,uint,L,g,pl,rl,ql,pr)
% Candidate vector for satisfying Assumption 1
% Input:  xint  = mesh
%         uint  = approximate solution
%         L     = the half length of the domain
%         g     = thermodynamic constant \gamma=C_p/C_v=7/5 for diatomic gases
%         pl    = pressure at the left boundary
%         rl    = density (\rho) at the right boundary
%         ql    = speed at the left boundary
%         pr    = pressure at the right boundary
% Output: Yint  = candidate vector for Assumption 1

rql=rl*ql; % Speed*Density at left boundary
El=pl/(g-1)+rl*ql^2/2; % Energy density at the left boundary
m=(g-1)/(g+1);
s=ql-sqrt(g*(pr+m*pl)/rl/(1+m));
rr=rl*(pr+m*pl)/(pl+m*pr); % Density at the right boundary
qr=ql+(1-m)*(pl*sqrt(g*(1+m)/rl/(pr+m*pl))-sqrt(g*(pr+m*pl)/rl/(1+m)));
rqr=rr*qr; % Speed*Density at the right boundary
Er=pr/(g-1)+rr*qr^2/2; % Energy density at the right boundary

hint=xint(2:end)-xint(1:end-1);
r=uint(1:end/3);
if (any(r)<=0)
  error('Negative density')
end
rq=uint(end/3+1:2*end/3);
E=uint(2*end/3+1:end);
n=length(xint);
r=[rl;r;rr];     % Density
rq=[rql;rq;rqr]; % Density*Speed
E=[El;E;Er];     % Energy density

% Auxiliary variables. Copied from Feuler.
Pr=P(r);
lPr=log(Pr);
Qr=Q(r);
```

```
Qrq=Q(rq);
QE=Q(E);

% The first part (Density)
% For the constant integral part. Copied from Feuler.
v1=hint.*(rq(2:end)+rq(1:end-1)-s*(r(2:end)+r(1:end-1)))/2;
s1=intval(zeros(1,n-1)); % Allocation
s1(1)=v1(1);
for j=2:n-1
  s1(j)=s1(j-1)+v1(j); % Cumulative sum
end
c1=(r(end)-r(1)-s1(end))/2/L;
% The rest
F1=zeros(1,n-1); % Allocation
for j=1:n-1
  su=abs(rq(j:j+1)-s*r(j:j+1)+c1);
  F1(j)=max(su.sup);
end
F1=intval(F1);
F1=F1'.*hint/2;
F1=max(F1.sup);
F1=intval(F1);

% The second part (Density*Speed)
% For the constant integral part. Copied from Feuler.
a=1/2+2*Qrq-Qr-lPr.*(Qrq.*Qrq-Qr.*(2*Qrq-Qr));
a=a.*(rq(2:end)-rq(1:end-1)).^2./(r(2:end)-r(1:end-1));
v2=hint.*((g-1)*(E(2:end)+E(1:end-1))-s*(rq(2:end)+rq(1:end-1))+(3-g)*a)/2;
s2=intval(zeros(1,n-1)); % Allocation
s2(1)=v2(1);
for j=2:n-1
  s2(j)=s2(j-1)+v2(j); % Cumulative sum
end
c2=(rq(end)-rq(1)-s2(end))/2/L;
% The rest
F2=zeros(1,n-1); % Allocation
for j=1:n-1
  su(1)=abs((3-g)*rq(j)^2/2+r(j)*((g-1)*E(j)-s*rq(j)+c2));
  su(2)=abs((3-g)*rq(j)^2/2+r(j+1)*((g-1)*E(j)-s*rq(j)+c2));
  su(3)=abs((3-g)*rq(j+1)^2/2+r(j)*((g-1)*E(j)-s*rq(j+1)+c2));
  su(4)=abs((3-g)*rq(j+1)^2/2+r(j+1)*((g-1)*E(j)-s*rq(j+1)+c2));
  su(5)=abs((3-g)*rq(j)^2/2+r(j)*((g-1)*E(j+1)-s*rq(j)+c2));
  su(6)=abs((3-g)*rq(j)^2/2+r(j+1)*((g-1)*E(j+1)-s*rq(j)+c2));
  su(7)=abs((3-g)*rq(j+1)^2/2+r(j)*((g-1)*E(j+1)-s*rq(j+1)+c2));
  su(8)=abs((3-g)*rq(j+1)^2/2+r(j+1)*((g-1)*E(j+1)-s*rq(j+1)+c2));
  in=r(j:j+1);
  F2(j)=max(su.sup)/min(in.inf);
end
F2=intval(F2').*hint/2;
F2=max(F2.sup);
F2=intval(F2);

% The third part (Energy density)
% For the constant integral part. Copied from Feuler.
b=1/2+QE+Qrq-Qr-lPr.*(Qrq.*QE-Qr.*(QE+Qrq-Qr));
b=b.*(rq(2:end)-rq(1:end-1)).*(E(2:end)-E(1:end-1))./(r(2:end)-r(1:end-1));
c=1/2+3*Qrq-2*Qr+(Qrq-Qr).^2.*(-3*lPr+(Qrq-Qr).*(r(2:end)-r(1:end-1)).^2./r(2:end)./r(1:end-1));
c=c.*(rq(2:end)-rq(1:end-1)).^3./(r(2:end)-r(1:end-1)).^2;
v3=hint.*(g*b-((g-1)*c+s*(E(2:end)+E(1:end-1)))/2);
s3=intval(zeros(1,n-1)); % Allocation
s3(1)=v3(1);
for j=2:n-1
  s3(j)=s3(j-1)+v3(j); % Cumulative sum
end
c3=(E(end)-E(1)-s3(end))/2/L;
% The rest
F3=zeros(1,n-1); % Allocation
for j=1:n-1
  su(1)=abs((1-g)*rq(j)^3/2+r(j)*(g*E(j)*rq(j)+r(j)*(c3-s*E(j))));
  su(2)=abs((1-g)*rq(j)^3/2+r(j+1)*(g*E(j)*rq(j)+r(j+1)*(c3-s*E(j))));
  su(3)=abs((1-g)*rq(j+1)^3/2+r(j)*(g*E(j)*rq(j+1)+r(j)*(c3-s*E(j))));
```

```
  su(4)=abs((1-g)*rq(j+1)^3/2+r(j+1)*(g*E(j)*rq(j+1)+r(j+1)*(c3-s*E(j))));
  su(5)=abs((1-g)*rq(j)^3/2+r(j)*(g*E(j+1)*rq(j)+r(j)*(c3-s*E(j+1))));
  su(6)=abs((1-g)*rq(j)^3/2+r(j+1)*(g*E(j+1)*rq(j)+r(j+1)*(c3-s*E(j+1))));
  su(7)=abs((1-g)*rq(j+1)^3/2+r(j)*(g*E(j+1)*rq(j+1)+r(j)*(c3-s*E(j+1))));
  su(8)=abs((1-g)*rq(j+1)^3/2+r(j+1)*(g*E(j+1)*rq(j+1)+r(j+1)*(c3-s*E(j+1))));
  in=r(j:j+1).^2;
  F3(j)=max(su.sup)/min(in.inf);
end
F3=intval(F3)'.*hint/2;
F3=max(F3.sup);
F3=intval(F3);

Y=abs((speye(length(uint))-Beuler(xint,uint,L,g,pl,rl,ql,pr))\(Feuler(xint,uint,L,g,pl,rl,ql,pr)-uint));
Yint=[Y(1:end/3);F1;Y(1+end/3:2*end/3);F2;Y(2*end/3+1:end);F3];
```

P.m

```
function Pf=P(f)
% Easy help function used by Yeuler, Beuler and Feuler.
% Input:  f   = vector
% Output: Pf  = vector with one less component than f
Pf=f(1:end-1)./f(2:end);
```

Q.m

```
function Qf=Q(f)
% Easy help function used by Yeuler, Beuler and Feuler.
% Input:  f   = vector
% Output: Qf  = vector with one less component than f
Qf=f(1:end-1)./(f(2:end)-f(1:end-1));
```

Beuler.m

```
function Bint=Beuler(xint,uint,L,g,pl,rl,ql,pr)
% Constructs a matrix which describes the Fr\'echet derivative.
% Input:  xint  = mesh
%         uint  = approximate solution
%         L     = the half length of the domain
%         g     = thermodynamic constant \gamma=C_p/C_v=7/5 for diatomic gases
%         pl    = pressure at the left boundary
%         rl    = density (\rho) at the right boundary
%         ql    = speed at the left boundary
%         pr    = pressure at the right boundary
% Output: Bint  = the matrix mentioned above

rql=rl*ql; % Speed*Density at left boundary
El=pl/(g-1)+rl*ql^2/2; % Energy density at the left boundary
m=(g-1)/(g+1);
s=ql-sqrt(g*(pr+m*pl)/rl/(1+m));
rr=rl*(pr+m*pl)/(pl+m*pr); % Density at the right boundary
qr=ql+(1-m)*(pl*sqrt(g*(1+m)/rl/(pr+m*pl))-sqrt(g*(pr+m*pl)/rl/(1+m)));
rqr=rr*qr; % Speed*Density at the right boundary
Er=pr/(g-1)+rr*qr^2/2; % Energy density at the right boundary
hint=xint(2:end)-xint(1:end-1);
r=uint(1:end/3);
rq=uint(end/3+1:2*end/3);
E=uint(2*end/3+1:end);
n=length(xint);
r=[rl;r;rr];     % Density
rq=[rql;rq;rqr]; % Density*Speed
E=[El;E;Er];     % Energy density

% Auxiliary variables
Pr=P(r(1:end-1));
Prh=Phat(r(2:end));
lPr=log(Pr);
lPrh=log(Prh);
Qr=Q(r(1:end-1));
Qrh=Qhat(r(2:end));
```

```
Qrq=Q(rq(1:end-1));
Qrqh=Qhat(rq(2:end));
QE=Q(E(1:end-1));
QEh=Qhat(E(2:end));

alpha=intval(zeros(n-2,n-2));
% i=j
alpha(n-2,n-2)=hint(n-2)/2;
% i>j
for j=1:n-3
  alpha(j,j)=hint(j)/2;
  alpha(j+1:n-2,j)=(hint(j)+hint(j+1))/2;
end
alpha=alpha-(xint(2:end-1)+L)*(hint(1:end-1)+hint(2:end))'/4/L;

beta=intval(zeros(n-2,n-2));
% i=j
beta1=1/2+(Qrq-Qr).*(1+Qr.*lPr);
beta1=beta1.*hint(1:end-1).*(rq(2:end-1)-rq(1:end-2))./(r(2:end-1)-r(1:end-2));
% i>j
beta2=1/2+(Qrqh-Qrh).*(1+Qrh.*lPrh);
beta2=beta2.*hint(2:end).*(rq(2:end-1)-rq(3:end))./(r(2:end-1)-r(3:end));
beta2=beta1+beta2;
beta(n-2,n-2)=beta1(n-2);
for j=1:n-3
  beta(j,j)=beta1(j);
  beta(j+1:n-2,j)=beta2(j);
end
beta=beta-(xint(2:end-1)+L)*beta2'/2/L;

delta=intval(zeros(n-2,n-2));
% i=j
delta1=1/2+(QE-Qr).*(1+Qr.*lPr);
delta1=delta1.*hint(1:end-1).*(E(2:end-1)-E(1:end-2))./(r(2:end-1)-r(1:end-2));
% i>j
delta2=1/2+(QEh-Qrh).*(1+Qrh.*lPrh);
delta2=delta2.*hint(2:end).*(E(2:end-1)-E(3:end))./(r(2:end-1)-r(3:end));
delta2=delta1+delta2;
delta(n-2,n-2)=delta1(n-2);
for j=1:n-3
  delta(j,j)=delta1(j);
  delta(j+1:n-2,j)=delta2(j);
end
delta=delta-(xint(2:end-1)+L)*delta2'/2/L;

eta=intval(zeros(n-2,n-2));
% i=j
eta1=1/2+Qrq.*(2+4*lPr.*Qr+2*Pr+Qrq.*(Pr-1-lPr))-Qr.*(3*lPr.*Qr+2+Pr);
eta1=eta1.*hint(1:end-1).*(rq(2:end-1)-rq(1:end-2)).^2./(r(2:end-1)-r(1:end-2)).^2;
% i>j
eta2=1/2+Qrqh.*(2+4*lPrh.*Qrh+2*Prh+Qrqh.*(Prh-1-lPrh))-Qrh.*(3*lPrh.*Qrh+2+Prh);
eta2=eta2.*hint(2:end).*(rq(2:end-1)-rq(3:end)).^2./(r(2:end-1)-r(3:end)).^2;
eta2=eta2+eta1;
eta(n-2,n-2)=eta1(n-2);
for j=1:n-3
  eta(j,j)=eta1(j);
  eta(j+1:n-2,j)=eta2(j);
end
eta=eta-(xint(2:end-1)+L)*eta2'/2/L;

kappa=intval(zeros(n-2,n-2));
% i=j
kappa1=1/2+(Qrq+QE).*(1+2*lPr.*Qr+Pr)-QE.*Qrq.*(lPr+1-Pr)-Qr.*(3*lPr.*Qr+2+Pr);
kappa1=kappa1.*hint(1:end-1).*(E(2:end-1)-E(1:end-2)).*(rq(2:end-1)-rq(1:end-2))./(r(2:end-1)-r(1:end-2)).^2;
% i>j
kappa2=1/2+(Qrqh+QEh).*(1+2*lPrh.*Qrh+Prh)-QEh.*Qrqh.*(lPrh+1-Prh)-Qrh.*(3*lPrh.*Qrh+2+Prh);
kappa2=kappa2.*hint(2:end).*(E(2:end-1)-E(3:end)).*(rq(2:end-1)-rq(3:end))./(r(2:end-1)-r(3:end)).^2;
kappa2=kappa1+kappa2;
kappa(n-2,n-2)=kappa1(n-2);
for j=1:n-3
  kappa(j,j)=kappa1(j);
```

```
  kappa(j+1:n-2,j)=kappa2(j);
end
kappa=kappa-(xint(2:end-1)+L)*kappa2'/2/L;

lambda=intval(zeros(n-2,n-2));
% i=j
lambda1=1/2+(Qrq-Qr).*(3+3*lPr.*(2*Qr-Qrq)+(Qrq-Qr.*(7+6*Qr)).*(r(2:end-1).*rq(1:end-2)-r(1:end-2).*~
rq(2:end-1)).*(r(2:end-1)-r(1:end-2)).^2/2./r(2:end-1).^2./r(1:end-2)./(rq(2:end-1)-rq(1:end-2)));
lambda1=lambda1.*hint(1:end-1).*(rq(2:end-1)-rq(1:end-2)).^3./(r(2:end-1)-r(1:end-2)).^3;
% i>j
lambda2=1/2+(Qrqh-Qrh).*(3+3*lPrh.*(2*Qrh-Qrqh)+(Qrqh-Qrh.*(7+6*Qrh)).*(r(2:end-1).*rq(3:end)-r(3:end).*~
rq(2:end-1)).*(r(2:end-1)-r(3:end)).^2/2./r(2:end-1).^2./r(3:end)./(rq(2:end-1)-rq(3:end)));
lambda2=lambda2.*hint(2:end).*(rq(2:end-1)-rq(3:end)).^3./(r(2:end-1)-r(3:end)).^3;
lambda2=lambda2+lambda1;
lambda(n-2,n-2)=lambda1(n-2);
for j=1:n-3
  lambda(j,j)=lambda1(j);
  lambda(j+1:n-2,j)=lambda2(j);
end
lambda=lambda-(xint(2:end-1)+L)*lambda2'/2/L;

% Concatenation to form the matrix
B1=[-s*alpha,alpha,zeros(n-2,n-2)]; %The first row
B2=[(g-3)*eta/2,-(g-3)*beta-s*alpha,(g-1)*alpha]; %The second row
B3=[-g*kappa+(g-1)*lambda,g*delta-3*(g-1)*eta/2,g*beta-s*alpha]; %The third row

Bint=[B1;B2;B3];
```

Phat.m

```
function Phatf=Phat(f)
% Easy help function used by Beuler.
% Input:  f      = vector
% Output: Phatf  = vector with one less component than f
Phatf=f(2:end)./f(1:end-1);
```

Qhat.m

```
function Qhatf=Qhat(f)
% Easy help function used by Beuler.
% Input:  f      = vector
% Output: Qhatf  = vector with one less component than f
Qhatf=f(2:end)./(f(1:end-1)-f(2:end));
```

Feuler.m

```
function Fint=Feuler(xint,uint,L,g,pl,rl,ql,pr)
% Computes \Pi^hF(u^h), where u=F(u) is the fixed-point equation.
% Input:  xint  = mesh
%         uint  = approximate solution
%         L     = the half length of the domain
%         g     = thermodynamic constant \gamma=C_p/C_v=7/5 for diatomic gases
%         pl    = pressure at the left boundary
%         rl    = density (\rho) at the right boundary
%         ql    = speed at the left boundary
%         pr    = pressure at the right boundary
% Output: Fint  = F mentioned above

rql=rl*ql; % Speed*Density at left boundary
El=pl/(g-1)+rl*ql^2/2; % Energy density at the left boundary
m=(g-1)/(g+1);
s=ql-sqrt(g*(pr+m*pl)/rl/(1+m));
rr=rl*(pr+m*pl)/(pl+m*pr); % Density at the right boundary
qr=ql+(1-m)*(pl*sqrt(g*(1+m)/rl/(pr+m*pl))-sqrt(g*(pr+m*pl)/rl/(1+m)));
rqr=rr*qr; % Speed*Density at the right boundary
Er=pr/(g-1)+rr*qr^2/2; % Energy density at the right boundary

hint=xint(2:end)-xint(1:end-1);
r=uint(1:end/3);
```

```
rq=uint(end/3+1:2*end/3);
E=uint(2*end/3+1:end);
n=length(xint);
r=[rl;r;rr];     % Density
rq=[rql;rq;rqr]; % Density*Speed
E=[El;E;Er];     % Energy density

% Auxiliary variables
Pr=P(r);
lPr=log(Pr);
Qr=Q(r);
Qrq=Q(rq);
QE=Q(E);

% The first part of F (Density)
v1=hint.*(rq(2:end)+rq(1:end-1)-s*(r(2:end)+r(1:end-1)))/2;
s1=intval(zeros(1,n-1)); % Allocation
s1(1)=v1(1);
for j=2:n-1
  s1(j)=s1(j-1)+v1(j); % Cumulative sum
end
s1=s1';
Fint(1:n-2)=r(1)+s1(1:end-1)+(xint(2:end-1)+L)*(r(end)-r(1)-s1(end))/2/L;

% The second part of F (Density*Speed)
a=1/2+2*Qrq-Qr-lPr.*(Qrq.*Qrq-Qr.*(2*Qrq-Qr));
a=a.*(rq(2:end)-rq(1:end-1)).^2./(r(2:end)-r(1:end-1));
v2=hint.*((g-1)*(E(2:end)+E(1:end-1))-s*(rq(2:end)+rq(1:end-1))+(3-g)*a)/2;
s2=intval(zeros(1,n-1)); % Allocation
s2(1)=v2(1);
for j=2:n-1
  s2(j)=s2(j-1)+v2(j); % Cumulative sum
end
s2=s2';
Fint(n-1:2*n-4)=rq(1)+s2(1:end-1)+(xint(2:end-1)+L)*(rq(end)-rq(1)-s2(end))/2/L;

% The third part of F (Energy density)
b=1/2+QE+Qrq-Qr-lPr.*(Qrq.*QE-Qr.*(QE+Qrq-Qr));
b=b.*(rq(2:end)-rq(1:end-1)).*(E(2:end)-E(1:end-1))./(r(2:end)-r(1:end-1));
c=1/2+3*Qrq-2*Qr+(Qrq-Qr).^2.*(-3*lPr+(Qrq-Qr).*(r(2:end)-r(1:end-1)).^2./r(2:end)./r(1:end-1));
c=c.*(rq(2:end)-rq(1:end-1)).^3./(r(2:end)-r(1:end-1)).^2;
v3=hint.*(g*b-((g-1)*c+s*(E(2:end)+E(1:end-1)))/2);
s3=intval(zeros(1,n-1)); % Allocation
s3(1)=v3(1);
for j=2:n-1
  s3(j)=s3(j-1)+v3(j); % Cumulative sum
end
s3=s3';
Fint(2*n-3:3*n-6)=E(1)+s3(1:end-1)+(xint(2:end-1)+L)*(E(end)-E(1)-s3(end))/2/L;

Fint=Fint';
```

Zeuler.m

```
function Z=Zeuler(xint,uint,L,g,pl,rl,ql,pr,W)
% Candidate vector for satisfying Assumption 2
% Input:  xint  = mesh
%         uint  = approximate solution
%         L     = the half length of the domain
%         g     = thermodynamic constant \gamma=C_p/C_v=7/5 for diatomic gases
%         pl    = pressure at the left boundary
%         rl    = density (\rho) at the right boundary
%         ql    = speed at the left boundary
%         pr    = pressure at the right boundary
%         W     = candidate set written in vector form
% Output: Z     = candidate vector for Assumption 2

disp('Finite dimensional part...')
tic % Starts timer for the finite dimensional part
Wr=W(1:end/3);
Wrq=W(1+end/3:2*end/3);
```

```
WE=W(1+2*end/3:end);

rql=rl*ql; % Speed*Density at left boundary
El=pl/(g-1)+rl*ql^2/2; % Energy density at the left boundary
m=(g-1)/(g+1);
s=ql-sqrt(g*(pr+m*pl)/rl/(1+m));
rr=rl*(pr+m*pl)/(pl+m*pr); % Density at the right boundary
qr=ql+(1-m)*(pl*sqrt(g*(1+m)/rl/(pr+m*pl))-sqrt(g*(pr+m*pl)/rl/(1+m)));
rqr=rr*qr; % Speed*Density at the right boundary
Er=pr/(g-1)+rr*qr^2/2; % Energy density at the right boundary

hint=xint(2:end)-xint(1:end-1);
r=uint(1:end/3);
rq=uint(end/3+1:2*end/3);
E=uint(2*end/3+1:end);
xshort=xint(2:end-1);
N=length(xint)-1;

%% Finite dimensional part

% The first part (Density)
Zr=2*(xshort+L)*(Wrq(end)+abs(s)*Wr(end));

% No error on the boundary
% The last component is the error in the infinite dimensional space
Wr=[0;Wr(1:end-1);0;Wr(end)]; % No error on the boundary ~
and the last component is the error in the infinite dimensional space
Wrq=[0;Wrq(1:end-1);0;Wrq(end)];
WE=[0;WE(1:end-1);0;WE(end)];

r=[rl;r;rr];     % Density
rq=[rql;rq;rqr]; % Density*Speed
E=[El;E;Er];     % Energy density

A2=ma([abs(rq(1:end-1))+Wrq(1:end-2) abs(rq(2:end))+Wrq(2:end-1)])+Wrq(end);
A3=ma([abs(E(1:end-1))+WE(1:end-2) abs(E(2:end))+WE(2:end-1)])+WE(end);
C2=ma([abs(rq(1:end-1)) abs(rq(2:end))]);
C3=ma([abs(E(1:end-1)) abs(E(2:end))]);
D1=ma([Wr(1:end-2) Wr(2:end-1)]);
D2=ma([Wrq(1:end-2) Wrq(2:end-1)]);
D3=ma([WE(1:end-2) WE(2:end-1)]);
B1=D1+Wr(end);
B2=D2+Wrq(end);
B3=D3+WE(end);
G=mi([r(1:end-1) r(2:end)]);
F=G-ma([Wr(1:end-2) Wr(2:end-1)])-Wr(end);
if any(F.inf<=0)
  error('Illegal estimation')
end

% The second part (Density*Speed)
Zrqd=abs(g-3)*hint.*(A2.^2.*B1/2./F.^2+A2.*B2./F+C2.^2.*D1/2./G.^2+C2.*D2./G);
Zrq=intval(zeros(N,1)); % Allocation
Zrq(1)=Zrqd(1);
for j=2:N
  Zrq(j)=Zrq(j-1)+Zrqd(j); % Cumulative summation
end
Zrq=Zrq(1:end-1)+(xshort+L)*(Zrq(end)/2/L+2*abs(g-1)*WE(end)+2*abs(s)*Wrq(end));

% The third part (Energy density)
ZEd=hint.*(g*((A2.*B3+A3.*B2)./F+A2.*A3.*B1./F.^2+(C2.*D3+C3.*D2)./G+C2.*C3.*D1./G.^2)+abs(g-1)~
*(A2.^3.*B1./F.^3+3*A2.^2.*B2/2./F.^2+C2.^3.*D1./G.^3+3*C2.^2.*D2./2./G.^2));
ZE=intval(zeros(N,1));  % Allocation
ZE(1)=ZEd(1);
for j=2:N
  ZE(j)=ZE(j-1)+ZEd(j); % Cumulative summation
end
ZE=ZE(1:end-1)+(xshort+L)*(ZE(end)/2/L+2*abs(s)*WE(end));
toc % Stops timer for the finite dimensional part
disp('Matrix construction, inversion and matrix vector multiplication...')
tic % Starts timer for matrix inversion
```

```
Zshort=abs(inv(speye(length(uint))-Beuler(xint,uint,L,g,pl,rl,ql,pr)))*[Zr;Zrq;ZE];
toc % Stops timer for matrix inversion

%% Infinite dimensional part
disp('Infinite dimensional part...')
tic % Starts timer for the infinite dimensional part
% The first part (Density)
Zr=B2+abs(s)*B1;
Zr=ma(hint.*(Zr+hint'*Zr/2/L))/2;

% The second part (Density)
Zrq=abs(g-3)*(A2.^2.*B1/2./F.^2+A2.*B2./F)+abs(g-1)*B3+abs(s)*B2;
Zrq=ma(hint.*(Zrq+hint'*Zrq/2/L))/2;

% The third part (Energy density)
ZE=g*((A2.*B3+A3.*B2)./F+A2.*A3.*B1./F.^2)+abs(g-1)*(A2.^3.*B1./F.^3+3*A2.^2.*B2/2./F.^2)+abs(s)*B3;
ZE=ma(hint.*(ZE+hint'*ZE/2/L))/2;

Z=[Zshort(1:end/3);Zr;Zshort(1+end/3:2*end/3);Zrq;Zshort(2*end/3+1:end);ZE];
toc % Stops timer for the infinite dimensional part
```

ma.m

```
function y=ma(x)
% Rigorous max for nx1 or nx2 matrices
% Used by Zeuler
% Input:  x = nx1 or nx2 intval matrix
% Output: y = maximum of the nx1 matrix or the maximum of each column of the nx2 matrix
s=size(x);
if s(2)==1
  y=intval(max(x.sup));
elseif s(2)==2
  for j=1:s(1)
    a=x(j,1);
    b=x(j,2);
    y(j)=intval(max(a.sup,b.sup));
  end
  y=y';
else
  error('Wrong size')
end
```

mi.m

```
function y=mi(x)
% Rigorous min for nx1 or nx2 matrices
% Used by Zeuler
% Input:  x = nx1 or nx2 intval matrix
% Output: y = minimum of the nx1 matrix or the minimum of each column of the nx2 matrix
s=size(x);
if s(2)==1
  y=intval(min(x.inf));
elseif s(2)==2
  for j=1:s(1)
    a=x(j,1);
    b=x(j,2);
    y(j)=intval(min(a.inf,b.inf));
  end
  y=y';
else
  error('Wrong size')
end
```