# A study of the numerical schemes in Zephyr

Rong Wang[*], Raymond J. Spiteri[†], and Rosette Tuy[‡]

June 28, 2005

**Abstract**

Zephyr is a software package designed by Martec, Ltd. that is designed to solve the Euler equations for the ideal (or perfect) gas. It is a derivative of Martec's proprietary flow solver, Chinook. Zephyr was created as a research tool to examine the key numerical features of Chinook. Zephyr employs a Godunov-type scheme, where the Godunov flux is computed by solving the corresponding Riemann problem with the approximate Riemann solver known as HLLC. Instead of applying the traditional Godunov scheme for the time integration, Zephyr uses explicit Runge-Kutta (ERK) schemes. In this report we describe the schemes in Zephyr for the spatial and time discretizations. Numerical results are presented for the classical shocktube problem using the ERK methods originally in Zephyr as well as other ERK methods that we have added. Computational results show that the stability restriction is the major barrier in terms of the allowable time step. That is, the stability requirement forces the software to use tiny time steps, consequently leading to exceedingly small temporal errors. Thus, there is no reason to employ anything beyond the forward Euler method for the time integration. Indeed we see that it is the most efficient in terms of both computer time and memory. Better efficiencies will only be possible using an implicit time integrator such as the implicit HLLC or a high-order spatial scheme.

Keywords: Euler equations, HLLC, explicit Runge–Kutta methods

## 1 Properties of the 1-D Euler equations

The one-dimensional Euler equations for an incompressible gas take the form

$$\mathbf{U}_t + \mathbf{f}_x(\mathbf{U}) = \mathbf{0}, \tag{1}$$

[*]Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, S7N 5A9, Canada. The work of this author was partially supported by MITACS and Martec, Ltd. `rong@cs.usask.ca`.

[†]Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, S7N 5A9, Canada. The work of this author was partially supported by MITACS and Martec, Ltd. `spiteri@cs.usask.ca`.

[‡]Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, S7N 5A9, Canada. The work of this author was partially supported by MITACS and Martec, Ltd.. `tuy@cs.usask.ca`.

where

$$\mathbf{U} := \begin{pmatrix} u^{(1)} \\ u^{(2)} \\ u^{(3)} \end{pmatrix} = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}, \quad \mathbf{f}(\mathbf{U}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u H \end{pmatrix},$$

where $\rho$ is the density, $u$ is the velocity, $p$ is the pressure, $E = e + \frac{1}{2}u^2$, and $H = E + p/\rho$. These equations respectively represent the conservation of mass, momentum, and energy. We assume the gas is ideal; the equation of state is then given by

$$p = (\gamma - 1)\rho e,$$

where $\gamma$ is a constant, which represents the ratio of specific heats.

If $\mathbf{U}_x$ exists, equation (1) can be rewritten as

$$\mathbf{U}_t + \mathbf{f}'(\mathbf{U})\mathbf{U}_x = \mathbf{0}. \tag{2}$$

In numerical computations, the characteristic speeds of the gas are needed to calculate the time-step stability restriction. These speeds are given by the eigenvalues of $\mathbf{f}'(\mathbf{U})$. It is not easy to obtain to obtain these eigenvalues directly from $\mathbf{f}'(\mathbf{U})$. However, we can more easily compute the eigenvalues by using the following transformation of variables. We generate new dependent variables by the transformation

$$\begin{aligned} w^{(1)} &= u^{(1)}, \\ w^{(2)} &= \frac{u^{(2)}}{u^{(1)}}, \\ w^{(3)} &= (\gamma - 1)\left(u^{(3)} - \frac{(u^{(2)})^2}{u^{(1)}}\right). \end{aligned}$$

That is,

$$\mathbf{W} := \begin{pmatrix} w^{(1)} \\ w^{(2)} \\ w^{(3)} \end{pmatrix} = \begin{pmatrix} \rho \\ u \\ p \end{pmatrix}.$$

Now $\mathbf{U}_t = (\partial\mathbf{U}/\partial\mathbf{W})\mathbf{W}_t := \mathbf{Q}\mathbf{W}_t$, and $\mathbf{U}_x = (\partial\mathbf{U}/\partial\mathbf{W})\mathbf{W}_x := \mathbf{Q}\mathbf{W}_x$, equation (2) can be rewritten as

$$\mathbf{Q}\mathbf{W}_t + \mathbf{f}'(\mathbf{U}(\mathbf{W}))\mathbf{Q}\mathbf{W}_x = \mathbf{0}.$$

The Jacobian $\mathbf{Q} = \partial\mathbf{U}/\partial\mathbf{W}$ and its inverse are found to be

$$\mathbf{Q} = \begin{pmatrix} 1 & 0 & 0 \\ u & \rho & 0 \\ \frac{1}{2}u^2 & \rho u & \frac{1}{\gamma-1} \end{pmatrix}, \quad \mathbf{Q}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -u/p & 1/\rho & 0 \\ \frac{\gamma-1}{2}u^2 & (1-\gamma)u & \gamma - 1 \end{pmatrix}.$$

Left multiplication by $\mathbf{Q}^{-1}$ gives the Euler equations in nonconservative form,

$$\mathbf{W}_t + \tilde{\mathbf{f}}'(\mathbf{W})\mathbf{W}_x = \mathbf{0},$$

where

$$\tilde{\mathbf{f}}'(\mathbf{W}) = \mathbf{Q}^{-1}\mathbf{f}'(\mathbf{U}(\mathbf{W}))\mathbf{Q} = \begin{pmatrix} u & \rho & 0 \\ 0 & u & 1/\rho \\ 0 & \rho c^2 & u \end{pmatrix}.$$

Here $c = \sqrt{\gamma p/\rho}$ is the speed of sound. The eigenvalues of $\tilde{\mathbf{f}}'(\mathbf{W})$ (and hence of $\mathbf{f}'(\mathbf{U})$) are easily found to be

$$\lambda_1 = u - c, \quad \lambda_2 = u, \quad \lambda_3 = u + c. \tag{3}$$

The corresponding eigenvectors of $\tilde{\mathbf{f}}'(\mathbf{W})$ are

$$\tilde{\mathbf{R}}_1 = \frac{1}{2}\begin{pmatrix} -\rho/c \\ 1 \\ -\rho c \end{pmatrix}, \quad \tilde{\mathbf{R}}_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \tilde{\mathbf{R}}_3 = \frac{1}{2}\begin{pmatrix} \rho/c \\ 1 \\ \rho c \end{pmatrix}.$$

We note that the wave speed is the maximum of the absolute value of the eigenvalues in equation (3). That is

$$S_w = \begin{cases} u + c, & \text{if } u \geq 0; \\ -u + c, & \text{if } u < 0; \end{cases} \tag{4}$$

## 2 Conservative Spatial Discretization Scheme

We assume the spatial domain is $[0, 1]$ and that a uniform mesh with $J$ subintervals is used; i.e., the mesh has grid points

$$x_j = j/J, \quad i = 0, 1, \ldots, J.$$

A conservative scheme is of the following form:

$$\dot{\mathbf{U}}_j = -\frac{1}{\Delta x}\left(\hat{\mathbf{f}}_{j+\frac{1}{2}} - \hat{\mathbf{f}}_{j-\frac{1}{2}}\right),$$

where $\Delta x = x_{j+1} - x_j$, and $\hat{\mathbf{f}}_{j+1/2}$ is the solution of the following Riemann problem in the cell $[x_j, x_{j+1}]$; i.e., equation (1) with the initial condition

$$\mathbf{U}(x, t_n) = \mathbf{U}_L = \mathbf{U}(x_i, t_n), \quad \text{if } x_i \leq x \leq x_{i+\frac{1}{2}};$$

$$\mathbf{U}(x, t_n) = \mathbf{U}_R = \mathbf{U}(x_{i+1}, t_n), \quad \text{if } x_{i+\frac{1}{2}} \leq x \leq x_{i+1}.$$

We now describe the HLLC Riemann solver [1, 7] implemented in Zephyr. Let the local left and the right wave speeds be

$$\begin{aligned} s_i^L &= \min\left(\bar{u} - \bar{c}, u_i - c_i\right), \\ s_i^R &= \max\left(\bar{u} + \bar{c}, u_{i+1} + c_{i+1}\right), \end{aligned}$$

where $u_i$ is the velocity, $c_i$ is the speed of sound at $x_i$, $\bar{u}$ and $\bar{c}$ are given as follows

$$\begin{aligned}\bar{u} &= \frac{\sqrt{\rho_i}u_i + \sqrt{\rho_{i+1}}u_{i+1}}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}},\\ \bar{c} &= \sqrt{\frac{\sqrt{\rho_i}c_i^2 + \sqrt{\rho_{i+1}}c_{i+1}^2}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}}}.\end{aligned}$$

Another parameter $u_i^{\text{HLL}}$ is then computed according to

$$u_i^{\text{HLL}} = \frac{s_i^R \rho_{i+1} u_{i+1} - s_i^L \rho_i u_i + (f_2)_i - (f_2)_{i+1}}{\rho_i^{\text{HLL}}(s_i^R - s_i^L)},$$

where $(f_2)_i$ is the second component of $\mathbf{f}(\mathbf{U})$, i.e., $\rho u^2 + p$, evaluated at $x_i$. The parameter $\rho_i^{\text{HLL}}$ is obtained from

$$\rho_i^{\text{HLL}} = \frac{s_i^R \rho_{i+1} - s_i^L \rho_i + (f_1)_i - (f_1)_{i+1}}{s_i^R - s_i^L},$$

where $(f_1)_i$ is the first component of $\mathbf{f}(\mathbf{U})$, i.e., $\rho u$, evaluated at $x_i$.

The HLLC flux is now written as

$$\hat{\mathbf{f}}_{j+\frac{1}{2}} = \begin{cases} \mathbf{f}_i, & \text{if} \quad 0 \le s_i^L, \\ \mathbf{f}_i + s_i^L(\mathbf{U}_\star^L - \mathbf{U}_i) & \text{if} \quad s_i^L \le 0 \le u_i^{\text{HLL}}, \\ \mathbf{f}_{i+1} + s_i^R(\mathbf{U}_\star^R - \mathbf{U}_{i+1}) & \text{if} \quad u_i^{\text{HLL}} \le 0 \le s_i^R \\ f_i, & \text{if} \quad 0 \le s_i^L, \end{cases}$$

where

$$\mathbf{U}_\star^L = \rho_i \left(\frac{s_i^L - u_i}{s_i^L - u_i^{\text{HLL}}}\right) \begin{bmatrix} 1 \\ u_i^{\text{HLL}} \\ \frac{E_i}{\rho_i} + (u_i^{\text{HLL}} - u_i)\left(u_i^{\text{HLL}} + \frac{p_i}{\rho_i(s_i^L - u_i)}\right) \end{bmatrix},$$

and

$$\mathbf{U}_\star^R = \rho_{i+1} \left(\frac{s_i^R - u_{i+1}}{s_i^R - u_i^{\text{HLL}}}\right) \begin{bmatrix} 1 \\ u_i^{\text{HLL}} \\ \frac{E_{i+1}}{\rho_{i+1}} + (u_i^{\text{HLL}} - u_{i+1})\left(u_i^{\text{HLL}} + \frac{p_{i+1}}{\rho_{i+1}(s_i^R - u_{i+1})}\right) \end{bmatrix}.$$

## 3 Low-storage ERK methods in Zephyr

Let us consider the following ordinary differential equation:

$$\mathbf{Y}_t = \mathbf{f}(t, \mathbf{Y}). \tag{5}$$

An $m$-stage low-storage ERK method has the form

$$\begin{aligned}
\mathbf{Y}^{(0)} &= \mathbf{Y}_n \\
\mathbf{Y}^{(1)} &= \mathbf{Y}_n + \alpha_1 \Delta t\, \mathbf{f}(t_n, \mathbf{Y}^{(0)}) \\
\mathbf{Y}^{(2)} &= \mathbf{Y}_n + \alpha_2 \Delta t\, \mathbf{f}(t_n + \alpha_1 \Delta t, \mathbf{Y}^{(1)}) \\
&\vdots \\
\mathbf{Y}^{(k+1)} &= \mathbf{Y}_n + \alpha_{k+1} \Delta t\, \mathbf{f}(t_n + \alpha_k \Delta t, \mathbf{Y}^{(k)}) \\
&\vdots \\
\mathbf{Y}_{n+1} = \mathbf{Y}^{(m)} &= \mathbf{Y}_n + \alpha_m \Delta t\, \mathbf{f}(t_n + \alpha_{m-1} \Delta t, \mathbf{Y}^{(m-1)}).
\end{aligned}$$

In order to be consistent, it is easy to see that we must have $\alpha_m = 1$. Unlike usual ERK schemes, only the approximate solution $\mathbf{Y}_n$ at $t_n$ and the current stage $\mathbf{Y}^{(k)}$ are stored in order to reduce the memory requirement. It is easy to show that these low-storage ERK schemes are at most first-order (though other low-storage schemes of higher order are possible; see e.g., [2]). The stage coefficients can be tuned to increase the maximum step size and to improve the stability for the upwind spatial discretization [8]. In Zephyr, the coefficients are chosen from [8] even though the spatial discretization scheme is based on HLLC. Table 3 gives the CFL number and the stage coefficients for the first- and second-order upwind spatial scheme.

| | first-order HLLC scheme | | | second-order HLLC scheme | | |
|---|---|---|---|---|---|---|
| stages | 2 | 4 | 6 | 2 | 4 | 6 |
| $\sigma$ | 1 | 2 | 3 | 0.4693 | 0.9214 | 3 |
| $\alpha_1$ | 0.3333 | 0.0833 | 0.0370 | 0.4242 | 0.1084 | 0.0370 |
| $\alpha_2$ | 1 | 0.2069 | 0.0851 | 1 | 0.2602 | 0.0851 |
| $\alpha_3$ | | 0.4265 | 0.1521 | | 0.5052 | 0.1521 |
| $\alpha_4$ | | 1 | 0.2562 | | 1 | 0.2562 |
| $\alpha_5$ | | | 0.4512 | | | 0.4512 |
| $\alpha_6$ | | | 1 | | | 1 |

Table 1: Stage coefficients ($\alpha$) and CFL number ($\sigma$).

Besides the above ERK methods, the forward Euler method is also implemented in Zephyr with $\sigma = 1$.

## 4 Algorithm for the time integration

The time integration in Zephyr is implemented as follows:

```
dt_ramp= $10^{-12}$
t = $t_0$
while ($t < t_{out}$)
```

```
{
    Calculate the wave speed in the $i$-th cell, $(S_w)_i$, using (4).
    Calculate dt_global, the largest step size based on the CFL condition.
    (See below for details.)
    Set dt_global= min(dt_global, dt_ramp).
    Set dt_ramp= 2 * dt_ramp.
    Set dt_employ= $\sigma$ * dt_global, where $\sigma$ is the CFL number from Table 1.
    ($\sigma$ depends on the choice of spatial and ERK schemes.)
    Take one step using ERK methods with the step size dt_employ.
    Set t = t + dt_employ.
}
```

We note that the initial step size is at most $10^{-12}$ even though the largest step size based on CFL condition may be much larger in practice than $10^{-12}$. At each time step, the largest time stable step size is calculated using $C\,\frac{\Delta x}{(S_w)_i}$, where $(S_w)_i$ is the wave speed in the $i$-th cell, and $C$ is the Courant coefficient supplied by the user. The recommended value of $C$ is 0.5. The step size `dt_global` is then chosen as the minimum value over all the cells; i.e.,

$$\texttt{dt\_global} = \min_i C\,\frac{\Delta x}{(S_w)_i}. \tag{6}$$

¿From (6) we note that `dt_global` does not depend on which ERK method is used. Because `dt_employ`= $\sigma$ * `dt_global`, we see from Table 3 that if the first-order HLLC method is employed for the spatial discretization,

- `dt_employ`= `dt_global` when the forward Euler method is used for the time integration;
- `dt_employ`= `dt_global` when the 2-stage ERK scheme is used for time integration;
- `dt_employ`= $2 \cdot$ `dt_global` when the 4-stage ERK scheme is used for time integration;
- `dt_employ`= $3 \cdot$ `dt_global` when the 6-stage ERK scheme is used for time integration.

In each time step, the $m$-stage ERK method takes essentially $m$ times as expensive as the forward Euler method. Therefore, we expect that the forward Euler method will be the least expensive method in terms of cost per step. The other three methods have similar costs per step. Specifically, the forward Euler method costs essentially half as much per step as the other methods. This becomes particularly relevant in situations where the spatial error is the dominant source of error in a given problem. In such cases, the forward Euler method will be the most computationally efficient.

Similarly, if the second-order HLLC method is employed for the spatial discretization,

- `dt_employ`= `dt_global` when the forward Euler method is used for the time integration;
- `dt_employ`= $0.4693 \cdot$ `dt_global` when the 2-stage ERK scheme is used for time integration;
- `dt_employ`= $0.9214 \cdot$ `dt_global` when the 4-stage ERK scheme is used for time integration;
- `dt_employ`= $3 \cdot$ `dt_global` when the 6-stage ERK scheme is used for time integration.

As before, the forward Euler method is the most computationally efficient. However, now the 6-stage ERK method is the second most efficient. The performance of 2-stage and 4-stage are still similar; they are the least efficient in this case.

## 5 Numerical experiments

The notation used and the statistics collected in this section include:

| | |
|---|---|
| $p$: | the pressure; |
| $\rho$: | the density; |
| $u$: | the velocity; |
| $CPU$: | the CPU time in seconds; |
| $r$: | the order of the spatial schemes; |
| $m$: | the number of stages for the ERK methods. |

Here $m = 1$ is the forward Euler method, $m = 2, 4, 6$, are the ERK methods in Table 3.

### 5.1 Problem 1

The first problem is the 1-D Euler equations with the following initial conditions.

$$p = \begin{cases} 10^5, & \text{if } 0 \le x \le 0.5; \\ 10^3, & \text{if } 0.5 \le x \le 1; \end{cases}$$

$$\rho = \begin{cases} 1, & \text{if } 0 \le x \le 0.5; \\ 0.01, & \text{if } 0.5 \le x \le 1; \end{cases}$$

$$u = 0, \qquad 0 \le x \le 1.$$

$\gamma$ is chosen to be 1.4.

A reference solution, $\mathbf{U}_{\text{ref}}$ is obtained by applying Zephyr with $J = 50000$, $r = 1$ and $m = 1$. We compute the solution at the output time $t = 4 \times 10^{-4}$ and plot $p$, $\rho$, and $u$.

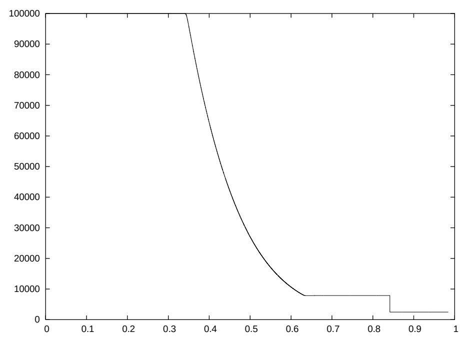

Figure 1: $p$ for Problem 1.

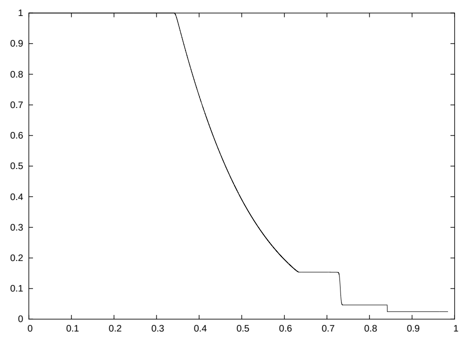

Figure 2: $\rho$ for Problem 1.

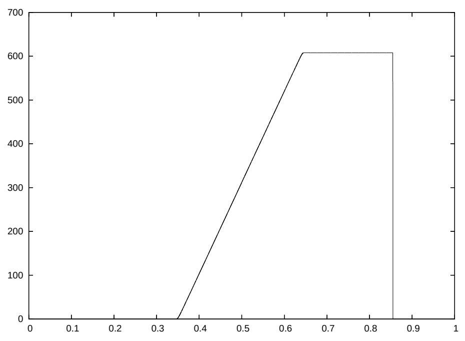

Figure 3: $u$ for Problem 1.

The error of a given numerical solution is computed from

$$\frac{\sqrt{\int_0^1 (\mathbf{U} - \mathbf{U}_{\text{ref}})^2 \, dx}}{\sqrt{\int_0^1 \mathbf{U}_{\text{ref}}^2 \, dx}}$$

We applied Zephyr with different time step sizes for testing. The default step size is given by `dt_employ`= $\sigma$ * `dt_global`. We further modified the formula to `dt_employ`= $\theta$ * $\sigma$ * `dt_global`in order to examine the effect of the Courant coefficient. Therefore, $\theta = 1$ means the default step size is used, while $\theta = 2$ means twice the default step size is used. Similarly, $\theta = 0.5$ means that only half the default step size is used. The following tables report the errors and the $CPU$ time when we apply Zephyr with $J = 400$ and $r = 1$.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.88 \cdot 10^{-2}$ | $1.82 \cdot 10^{-2}$ | $1.16 \cdot 10^{-1}$ | 10.67 |
| 1 | $1.81 \cdot 10^{-2}$ | $1.76 \cdot 10^{-2}$ | $1.14 \cdot 10^{-1}$ | 5.35 |
| 2 | $1.65 \cdot 10^{-2}$ | $1.58 \cdot 10^{-2}$ | $1.01 \cdot 10^{-1}$ | 2.91 |
| 4 or 8 | unstable | | | |

Table 2: The error with $J = 400$, $r = 1$, and $m = 1$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.90 \cdot 10^{-2}$ | $1.85 \cdot 10^{-2}$ | $1.16 \cdot 10^{-1}$ | 20.41 |
| 1 | $1.84 \cdot 10^{-2}$ | $1.82 \cdot 10^{-2}$ | $1.13 \cdot 10^{-1}$ | 10.68 |
| 2 | $1.76 \cdot 10^{-2}$ | $1.75 \cdot 10^{-2}$ | $1.17 \cdot 10^{-1}$ | 5.59 |
| 4 | $1.56 \cdot 10^{-2}$ | $1.60 \cdot 10^{-2}$ | $1.02 \cdot 10^{-1}$ | 3.11 |
| 8 | unstable | | | |

Table 3: The error with $J = 400$, $r = 1$, and $m = 2$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.86 \cdot 10^{-2}$ | $1.84 \cdot 10^{-2}$ | $1.15 \cdot 10^{-1}$ | 19.61 |
| 1 | $1.78 \cdot 10^{-2}$ | $1.78 \cdot 10^{-2}$ | $1.19 \cdot 10^{-1}$ | 10.50 |
| 2 | $1.67 \cdot 10^{-2}$ | $1.69 \cdot 10^{-2}$ | $1.28 \cdot 10^{-1}$ | 5.54 |
| 4 | $1.59 \cdot 10^{-2}$ | $1.60 \cdot 10^{-2}$ | $1.42 \cdot 10^{-1}$ | 3.73 |
| 8 | unstable | | | |

Table 4: The error with $J = 400$, $r = 1$, and $m = 4$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.83 \cdot 10^{-2}$ | $1.82 \cdot 10^{-2}$ | $1.16 \cdot 10^{-1}$ | 19.51 |
| 1 | $1.68 \cdot 10^{-2}$ | $1.73 \cdot 10^{-2}$ | $1.10 \cdot 10^{-1}$ | 10.46 |
| 2 | $1.46 \cdot 10^{-2}$ | $1.61 \cdot 10^{-2}$ | $8.57 \cdot 10^{-2}$ | 6.27 |
| 4 | $1.98 \cdot 10^{-2}$ | $1.66 \cdot 10^{-2}$ | $2.16 \cdot 10^{-1}$ | 4.24 |
| 8 | unstable | | | |

Table 5: The error with $J = 400$, $r = 1$, and $m = 6$ for Problem 1.

We make the following observations.

- As discussed in Section 4, the forward Euler method is the most efficient among all the methods when the default setting is used in Zephyr; i.e., when $\theta = 1$. The other schemes have similar performance, as expected.
- The forward Euler method is unstable when $\theta = 4$ or 8, whereas the other ERK methods are unstable only when $\theta = 8$. It seems that the Courant coefficient, $C$, can be set to 1.
- It is possible to improve efficiency without loss of accuracy by using values of $\theta$ that are greater than 1.
- The error does not increase when the time step size is increased provided that a stable solution is obtained. This means that the spatial error dominates the temporal error no matter which ERK scheme is used. In other words, the CFL restriction forces Zephyr to use tiny time steps that lead to unnecessarily small temporal errors. Because the CFL condition forces all the methods to take small time steps, they produce solutions with small temporal errors. Thus the total errors are dominated by the spatial component, and hence *there is no advantage to using higher-order ERK schemes.*

We now present the results for $J = 400$ and $r = 2$. As shown in the following tables, better errors are obtained because the second-order spatial scheme is employed.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $6.08 \cdot 10^{-3}$ | $6.63 \cdot 10^{-3}$ | $9.40 \cdot 10^{-2}$ | 19.54 |
| 1 | $5.16 \cdot 10^{-3}$ | $5.08 \cdot 10^{-3}$ | $9.25 \cdot 10^{-2}$ | 9.96 |
| 2 | $6.33 \cdot 10^{-3}$ | $4.82 \cdot 10^{-3}$ | $1.13 \cdot 10^{-1}$ | 5.32 |
| 4 or 8 | unstable | | | |

Table 6: The error with $J = 400$, $r = 2$, and $m = 1$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $6.74 \cdot 10^{-3}$ | $8.01 \cdot 10^{-3}$ | $9.25 \cdot 10^{-2}$ | 71.81 |
| 1 | $6.46 \cdot 10^{-3}$ | $7.78 \cdot 10^{-3}$ | $9.28 \cdot 10^{-2}$ | 36.67 |
| 2 | $5.93 \cdot 10^{-3}$ | $7.37 \cdot 10^{-3}$ | $9.28 \cdot 10^{-2}$ | 18.66 |
| 4 | $4.95 \cdot 10^{-3}$ | $6.66 \cdot 10^{-3}$ | $8.95 \cdot 10^{-2}$ | 9.80 |
| 8 | unstable | | | |

Table 7: The error with $J = 400$, $r = 2$, and $m = 2$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $6.62 \cdot 10^{-3}$ | $7.99 \cdot 10^{-3}$ | $9.45 \cdot 10^{-2}$ | 64.81 |
| 1 | $6.30 \cdot 10^{-3}$ | $7.81 \cdot 10^{-3}$ | $9.78 \cdot 10^{-2}$ | 33.05 |
| 2 | $6.37 \cdot 10^{-3}$ | $7.64 \cdot 10^{-3}$ | $1.11 \cdot 10^{-1}$ | 17.28 |
| 4 | $6.37 \cdot 10^{-3}$ | $7.78 \cdot 10^{-3}$ | $1.04 \cdot 10^{-1}$ | 10.04 |
| 8 | $1.08 \cdot 10^{-2}$ | $1.18 \cdot 10^{-2}$ | $1.35 \cdot 10^{-1}$ | 5.80 |

Table 8: The error with $J = 400$, $r = 2$, and $m = 4$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $6.22 \cdot 10^{-3}$ | $7.26 \cdot 10^{-3}$ | $1.07 \cdot 10^{-1}$ | 29.57 |
| 1 | $4.95 \cdot 10^{-3}$ | $6.76 \cdot 10^{-3}$ | $8.33 \cdot 10^{-2}$ | 16.88 |
| 2 | $9.31 \cdot 10^{-3}$ | $9.26 \cdot 10^{-3}$ | $1.13 \cdot 10^{-1}$ | 9.71 |
| 4 | $2.85 \cdot 10^{-2}$ | $1.95 \cdot 10^{-2}$ | $2.24 \cdot 10^{-1}$ | 6.44 |
| 8 | unstable | | | |

Table 9: The error with $J = 400$, $r = 2$, and $m = 6$ for Problem 1.

We have the same results as before regarding performance. There are, however, two notable differences.

- We are able to obtain a solution with $m = 4$ and $\theta = 8$ even though the error is larger than others. As we know, at each time step `dt_employ`$=$ $0.9214 \cdot$ `dt_global` for the 4-stage ERK scheme. In other words, it takes a smaller step than even the forward Euler method. Recall that the value of $\sigma$ and the coefficients of the ERK schemes employed in Zephyr are chosen from [8], where a different spatial discretization is employed. As for the HLLC scheme, a larger step size could be allowed, especially when the spatial scheme is second order.

- For the 6-stage ERK scheme, when we choose $\theta = 2$ and $\theta = 4$, the error is doubled; i.e., the temporal error now dominates the spatial error. This implies that it is not as accurate as the other methods with the same $\theta$; i.e., it has a larger error constant for this problem.

¿From the above experiments, we conclude that *when the same mesh and order of spatial discretization $r$ are used, the forward Euler method is the most efficient choice in default mode ($\theta = 1$).* We now present results using $J = 2000$.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $6.97 \cdot 10^{-3}$ | $8.27 \cdot 10^{-3}$ | $6.47 \cdot 10^{-2}$ | 274.51 |
| 1 | $6.65 \cdot 10^{-3}$ | $7.81 \cdot 10^{-3}$ | $6.56 \cdot 10^{-2}$ | 157.55 |
| 2 | $6.04 \cdot 10^{-3}$ | $6.72 \cdot 10^{-3}$ | $5.94 \cdot 10^{-2}$ | 73.38 |
| 4 or 8 | unstable | | | |

Table 10: The error with $J = 2000$, $r = 1$, and $m = 1$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $7.05 \cdot 10^{-3}$ | $8.51 \cdot 10^{-3}$ | $6.44 \cdot 10^{-2}$ | 539.12 |
| 1 | $6.90 \cdot 10^{-3}$ | $8.33 \cdot 10^{-3}$ | $6.62 \cdot 10^{-2}$ | 306.95 |
| 2 | $6.51 \cdot 10^{-3}$ | $7.93 \cdot 10^{-3}$ | $6.53 \cdot 10^{-2}$ | 140.47 |
| 4 | $5.54 \cdot 10^{-3}$ | $7.14 \cdot 10^{-3}$ | $5.41 \cdot 10^{-2}$ | 74.05 |
| 8 | unstable | | | |

Table 11: The error with $J = 2000$, $r = 1$, and $m = 2$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $6.96 \cdot 10^{-3}$ | $8.45 \cdot 10^{-3}$ | $6.65 \cdot 10^{-2}$ | 492.79 |
| 1 | $6.60 \cdot 10^{-3}$ | $8.21 \cdot 10^{-3}$ | $6.59 \cdot 10^{-2}$ | 282.55 |
| 2 | $5.98 \cdot 10^{-3}$ | $7.72 \cdot 10^{-3}$ | $6.50 \cdot 10^{-2}$ | 130.88 |
| 4 | $5.98 \cdot 10^{-3}$ | $7.20 \cdot 10^{-3}$ | $7.21 \cdot 10^{-2}$ | 80.18 |
| 8 | unstable | | | |

Table 12: The error with $J = 2000$, $r = 1$, and $m = 4$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $6.66 \cdot 10^{-3}$ | $8.37 \cdot 10^{-3}$ | $6.15 \cdot 10^{-2}$ | 480.56 |
| 1 | $6.40 \cdot 10^{-3}$ | $8.09 \cdot 10^{-3}$ | $6.85 \cdot 10^{-2}$ | 276.66 |
| 2 | $5.57 \cdot 10^{-3}$ | $7.55 \cdot 10^{-3}$ | $6.32 \cdot 10^{-2}$ | 140.17 |
| 4 | $4.51 \cdot 10^{-2}$ | $3.25 \cdot 10^{-2}$ | $2.17 \cdot 10^{-1}$ | 88.63 |
| 8 | unstable | | | |

Table 13: The error with $J = 2000$, $r = 1$, and $m = 6$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.53 \cdot 10^{-3}$ | $9.07 \cdot 10^{-4}$ | $3.81 \cdot 10^{-2}$ | 466.70 |
| 1 | $1.67 \cdot 10^{-3}$ | $1.66 \cdot 10^{-3}$ | $3.63 \cdot 10^{-2}$ | 234.76 |
| 2 | $3.18 \cdot 10^{-3}$ | $2.78 \cdot 10^{-3}$ | $4.14 \cdot 10^{-2}$ | 121.85 |
| 4 or 8 | unstable | | | |

Table 14: The error with $J = 2000$, $r = 2$, and $m = 1$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.78 \cdot 10^{-3}$ | $1.99 \cdot 10^{-3}$ | $4.11 \cdot 10^{-2}$ | 1795.01 |
| 1 | $1.65 \cdot 10^{-3}$ | $1.85 \cdot 10^{-3}$ | $3.96 \cdot 10^{-2}$ | 868.92 |
| 2 | $1.62 \cdot 10^{-3}$ | $1.61 \cdot 10^{-3}$ | $3.96 \cdot 10^{-2}$ | 433.24 |
| 4 | $1.90 \cdot 10^{-3}$ | $1.40 \cdot 10^{-3}$ | $3.93 \cdot 10^{-2}$ | 220.16 |
| 8 | unstable | | | |

Table 15: The error with $J = 2000$, $r = 2$, and $m = 2$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.75 \cdot 10^{-3}$ | $2.08 \cdot 10^{-3}$ | $4.08 \cdot 10^{-2}$ | 1775.02 |
| 1 | $1.81 \cdot 10^{-3}$ | $2.07 \cdot 10^{-3}$ | $4.31 \cdot 10^{-2}$ | 861.99 |
| 2 | $2.09 \cdot 10^{-3}$ | $2.16 \cdot 10^{-3}$ | $4.52 \cdot 10^{-2}$ | 397.83 |
| 4 | $3.06 \cdot 10^{-3}$ | $2.86 \cdot 10^{-3}$ | $3.60 \cdot 10^{-2}$ | 209.32 |
| 8 | $9.95 \cdot 10^{-3}$ | $7.84 \cdot 10^{-3}$ | $5.94 \cdot 10^{-2}$ | 112.69 |

Table 16: The error with $J = 2000$, $r = 2$, and $m = 4$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.95 \cdot 10^{-3}$ | $1.67 \cdot 10^{-3}$ | $4.40 \cdot 10^{-2}$ | 685.98 |
| 1 | $2.23 \cdot 10^{-3}$ | $1.99 \cdot 10^{-3}$ | $2.58 \cdot 10^{-2}$ | 410.00 |
| 2 | $2.06 \cdot 10^{-2}$ | $8.88 \cdot 10^{-3}$ | $3.34 \cdot 10^{-1}$ | 249.68 |
| 4 or 8 | unstable | | | |

Table 17: The error with $J = 2000$, $r = 2$, and $m = 6$ for Problem 1.

Comparing the tests between $J = 400$ and 2000, we see that in order to obtain the same accuracy with $m = 1$, using $r = 2$ is more efficient. For example, if we set $\theta = 1$, similar errors are obtained by using either $J = 400$, $r = 2$, and $m = 1$, or $J = 2000$, $r = 1$, and $m = 1$. However, the former combination of settings leads to much more efficient program than the latter.

In addition to the ERK schemes originally coded in Zephyr, we tested two classical second-order ERK methods. We refer to the first one as $m = 2M$. It is based on the well known mid-point rule and has the form

$$\begin{aligned}\mathbf{Y}^{(1)} &= \mathbf{Y}_n + 0.5\,\Delta t\,\mathbf{f}(t_n,\mathbf{Y}_n),\\ \mathbf{Y}_{n+1} &= \mathbf{Y}_n + \Delta t\,\mathbf{f}(t_n + 0.5\,\Delta t, \mathbf{Y}^{(1)}).\end{aligned}$$

We refer the second one as $m = 2T$. It is based on the well known trapezoidal rule and has the form

$$\begin{aligned}\mathbf{Y}^{(1)} &= \mathbf{Y}_n + \Delta t\,\mathbf{f}(t_n,\mathbf{Y}_n),\\ \mathbf{Y}_{n+1} &= \mathbf{Y}_n + 0.5\,\Delta t\,\mathbf{f}(t_n,\mathbf{Y}_n) + 0.5\,\Delta t\,\mathbf{f}(t_n + \Delta t, \mathbf{Y}^{(1)}).\end{aligned}$$

We now present the results by using the two ERK schemes with $\sigma = 1$.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $7.08 \cdot 10^{-3}$ | $8.62 \cdot 10^{-3}$ | $6.44 \cdot 10^{-2}$ | 537.62 |
| 1 | $6.97 \cdot 10^{-3}$ | $8.56 \cdot 10^{-3}$ | $6.59 \cdot 10^{-2}$ | 270.58 |
| 2 | $6.65 \cdot 10^{-3}$ | $8.40 \cdot 10^{-3}$ | $6.50 \cdot 10^{-2}$ | 135.68 |
| 4 or 8 | unstable | | | |

Table 18: The error with $J = 2000$, $r = 1$, and $m = 2M$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $6.97 \cdot 10^{-3}$ | $8.27 \cdot 10^{-3}$ | $6.47 \cdot 10^{-2}$ | 489.13 |
| 1 | $6.65 \cdot 10^{-3}$ | $7.81 \cdot 10^{-3}$ | $6.56 \cdot 10^{-2}$ | 246.31 |
| 2 | $6.04 \cdot 10^{-3}$ | $6.72 \cdot 10^{-3}$ | $5.94 \cdot 10^{-2}$ | 124.77 |
| 4 or 8 | unstable | | | |

Table 19: The error with $J = 2000$, $r = 1$, and $m = 2T$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.79 \cdot 10^{-3}$ | $2.07 \cdot 10^{-3}$ | $4.20 \cdot 10^{-2}$ | 819.13 |
| 1 | $1.54 \cdot 10^{-3}$ | $2.07 \cdot 10^{-3}$ | $3.78 \cdot 10^{-2}$ | 408.77 |
| 2 | $2.07 \cdot 10^{-3}$ | $2.11 \cdot 10^{-3}$ | $4.37 \cdot 10^{-2}$ | 209.06 |
| 4 or 8 | unstable | | | |

Table 20: The error with $J = 2000$, $r = 2$, and $m = 2M$ for Problem 1.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.53 \cdot 10^{-3}$ | $9.07 \cdot 10^{-4}$ | $3.81 \cdot 10^{-2}$ | 737.96 |
| 1 | $1.67 \cdot 10^{-3}$ | $1.66 \cdot 10^{-3}$ | $3.63 \cdot 10^{-2}$ | 371.49 |
| 2 | $3.14 \cdot 10^{-3}$ | $2.75 \cdot 10^{-3}$ | $4.17 \cdot 10^{-2}$ | 191.53 |
| 4 | unstable | | | |

Table 21: The error with $J = 2000$, $r = 2$, and $m = 2T$ for Problem 1.

As expected, these added methods offer no computational advantage compared to the forward Euler method.

## 5.2 Problem 2

The second problem we consider is the 1-D Euler equations with the following initial conditions:

$$p = \begin{cases} 10^4, & \text{if } 0 \leq x \leq 0.5; \\ 10^5, & \text{if } 0.5 \leq x \leq 1; \end{cases}$$

$$\rho = \begin{cases} 0.125, & \text{if } 0 \leq x \leq 0.5; \\ 1, & \text{if } 0.5 \leq x \leq 1; \end{cases}$$

$$u = 0, \qquad 0 \leq x \leq 1.$$

$\gamma$ is chosen to be 1.4.

The reference solution $\mathbf{U}_{\text{ref}}$ is obtained by applying Zephyr with $J = 50000$, $r = 2$, and $m = 1$. We compute the solution at the output time $t = 6 \times 10^{-4}$.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $2.25 \cdot 10^{-2}$ | $2.37 \cdot 10^{-2}$ | $7.23 \cdot 10^{-2}$ | 8.26 |
| 1 | $2.08 \cdot 10^{-2}$ | $2.25 \cdot 10^{-2}$ | $6.62 \cdot 10^{-2}$ | 4.24 |
| 2 | $1.58 \cdot 10^{-2}$ | $1.94 \cdot 10^{-2}$ | $3.88 \cdot 10^{-2}$ | 2.28 |
| 4 or 8 | unstable | | | |

Table 22: The error with $J = 400$, $r = 1$, and $m = 1$ for Problem 2.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $2.30 \cdot 10^{-2}$ | $2.41 \cdot 10^{-2}$ | $7.46 \cdot 10^{-2}$ | 16.09 |
| 1 | $2.19 \cdot 10^{-2}$ | $2.35 \cdot 10^{-2}$ | $7.07 \cdot 10^{-2}$ | 8.32 |
| 2 | $2.11 \cdot 10^{-2}$ | $2.27 \cdot 10^{-2}$ | $7.30 \cdot 10^{-2}$ | 4.48 |
| 4 | $2.48 \cdot 10^{-2}$ | $2.36 \cdot 10^{-2}$ | $8.65 \cdot 10^{-2}$ | 2.56 |
| 8 | unstable | | | |

Table 23: The error with $J = 400$, $r = 1$, and $m = 2$ for Problem 2.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $2.24 \cdot 10^{-2}$ | $2.38 \cdot 10^{-2}$ | $7.39 \cdot 10^{-2}$ | 15.46 |
| 1 | $2.23 \cdot 10^{-2}$ | $2.35 \cdot 10^{-2}$ | $7.96 \cdot 10^{-2}$ | 8.09 |
| 2 | $1.67 \cdot 10^{-2}$ | $2.08 \cdot 10^{-2}$ | $4.65 \cdot 10^{-2}$ | 4.56 |
| 4 | $2.41 \cdot 10^{-2}$ | $2.30 \cdot 10^{-2}$ | $7.76 \cdot 10^{-2}$ | 3.19 |
| 8 | unstable | | | |

Table 24: The error with $J = 400$, $r = 1$, and $m = 4$ for Problem 2.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $2.14 \cdot 10^{-2}$ | $2.33 \cdot 10^{-2}$ | $6.95 \cdot 10^{-2}$ | 15.36 |
| 1 | $2.21 \cdot 10^{-2}$ | $2.33 \cdot 10^{-2}$ | $8.05 \cdot 10^{-2}$ | 8.43 |
| 2 | $1.58 \cdot 10^{-2}$ | $2.04 \cdot 10^{-2}$ | $4.09 \cdot 10^{-2}$ | 5.23 |
| 4 or 8 | unstable | | | |

Table 25: The error with $J = 400$, $r = 1$, and $m = 6$ for Problem 2.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $8.29 \cdot 10^{-3}$ | $1.15 \cdot 10^{-2}$ | $4.19 \cdot 10^{-2}$ | 14.86 |
| 1 | $7.49 \cdot 10^{-3}$ | $1.01 \cdot 10^{-2}$ | $4.20 \cdot 10^{-2}$ | 7.61 |
| 2 | $1.56 \cdot 10^{-2}$ | $1.28 \cdot 10^{-2}$ | $5.77 \cdot 10^{-2}$ | 4.25 |
| 4 or 8 | unstable | | | |

Table 26: The error with $J = 400$, $r = 2$, and $m = 1$ for Problem 2.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $1.01 \cdot 10^{-2}$ | $1.31 \cdot 10^{-2}$ | $4.74 \cdot 10^{-2}$ | 54.63 |
| 1 | $9.59 \cdot 10^{-3}$ | $1.28 \cdot 10^{-2}$ | $4.63 \cdot 10^{-2}$ | 27.67 |
| 2 | $8.78 \cdot 10^{-3}$ | $1.24 \cdot 10^{-2}$ | $4.47 \cdot 10^{-2}$ | 14.38 |
| 4 | $1.14 \cdot 10^{-2}$ | $1.31 \cdot 10^{-2}$ | $5.66 \cdot 10^{-2}$ | 7.65 |
| 8 | $7.41 \cdot 10^{-2}$ | $5.85 \cdot 10^{-2}$ | $1.66 \cdot 10^{-1}$ | 4.45 |

Table 27: The error with $J = 400$, $r = 2$, and $m = 2$ for Problem 2.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $9.75 \cdot 10^{-3}$ | $1.30 \cdot 10^{-2}$ | $4.65 \cdot 10^{-2}$ | 49.49 |
| 1 | $1.02 \cdot 10^{-2}$ | $1.32 \cdot 10^{-2}$ | $5.17 \cdot 10^{-2}$ | 25.54 |
| 2 | $1.00 \cdot 10^{-2}$ | $1.31 \cdot 10^{-2}$ | $4.99 \cdot 10^{-2}$ | 13.32 |
| 4 | $1.38 \cdot 10^{-2}$ | $1.50 \cdot 10^{-3}$ | $5.11 \cdot 10^{-2}$ | 7.72 |
| 8 | $3.16 \cdot 10^{-2}$ | $2.65 \cdot 10^{-2}$ | $6.36 \cdot 10^{-2}$ | 4.79 |

Table 28: The error with $J = 400$, $r = 2$, and $m = 4$ for Problem 2.

| $\theta$ | $p$ | $\rho$ | $u$ | $CPU$ |
|---|---|---|---|---|
| 0.5 | $8.74 \cdot 10^{-3}$ | $1.23 \cdot 10^{-2}$ | $4.52 \cdot 10^{-2}$ | 22.82 |
| 1 | $9.58 \cdot 10^{-3}$ | $1.25 \cdot 10^{-2}$ | $3.72 \cdot 10^{-2}$ | 13.2 |
| 2 | $7.14 \cdot 10^{-2}$ | $4.67 \cdot 10^{-2}$ | $2.76 \cdot 10^{-1}$ | 8.32 |
| 4 | $1.22 \cdot 10^{-1}$ | $8.38 \cdot 10^{-2}$ | $3.28 \cdot 10^{-1}$ | 6.41 |
| 8 | unstable | | | |

Table 29: The error with $J = 400$, $r = 2$, and $m = 6$ for Problem 2.

Once again we see that *in the default mode ($\theta = 1$) and for a given level of accuracy, the most efficient combination of settings for Zephyr is the second-order spatial discretization combined with the forward Euler method in time.*

## 6 Conclusions and future directions

We have performed several experiments using Zephyr for the 1-D Euler equations. The main conclusion is that, based on the current spatial discretization schemes, the stability requirement forces the software to use extremely small time steps, and hence the temporal errors are exceedingly small. High-order ERK methods are not able to take large enough step sizes to overcome the increased cost per step. The most efficient combination of settings is to use the second-order scheme in space and the forward Euler method in time. In order to improve the efficiency for 1-D problems, we could use high-order spatial discretization schemes, such as ENO [3, 4] or WENO [5] schemes. This can increase the numerical domain of integration and hence ameliorate the CFL restriction. At the same time, a given accuracy can be achieved by smaller number of mesh points in space, also leading to the possibility of employing larger time steps. Another approach to improve the efficiency of Zephyr is to switch to the use of an implicit solver.

## References

[1] P. Batten, N. Clarke, C. Lambert, and D.M. Causon, On the choice of wavespeeds for the HLLC Riemann solver, SIAM J. Sci. Comput., 18

(1997), 1553–1570.

[2] S. Gottlieb, C.-W. Shu, and E. Tadmor, Strong stability-preserving high-order time discretization methods, SIAM Review, 43 (2001), 89–112.

[3] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy, Uniformly high order Essentially Non-Oscillatory schemes I, SIAM J. Numer. Anal., 24 (1987), 270–309.

[4] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy, Uniformly high order Essentially Non-Oscillatory schemes III, J. Comput. Phys., 71 (1987), 231–303.

[5] G.-S. Jiang and C.-W. Shu, Efficient implementation of Weighted ENO schemes, J. Comput. Phys., 126 (1996), 202–228.

[6] B. Van Leer, C.H. Tai, and K.G. Powell, Design of optimally smoothing multi-stage schemes for the Euler equations, AIAA Paper 89–1933, 1989.

[7] E.F. Toro, Riemann solvers and numerical methods for fluid dynamics, a practical introduction, Springer-Verlag, New York, second edition, 1999.

[8] C.H. Tai, J.H. Sheu, and B. Van Leer, Optimal multistage schemes for Euler equations with residual smoothing, AIAA Journal, 33 (1995), 1008–1016.