LMC Code Changes Writeup

Will Pazner

July 17, 2015

1 LMC Code

1.1 Previous results

Running a simple test using the unmodified `1D_MISDC` code for a hydrogen flame results in the following results.

$Y(\mathrm{H_2})$	2.89E-07	1.81	8.24E-08	1.66	2.61E-08
$Y(\mathrm{O_2})$	8.55E-06	1.85	2.38E-06	1.74	7.10E-07
$Y(\mathrm{N_2})$	1.07E-06	1.83	3.00E-07	1.72	9.11E-08
$Y(\mathrm{H_2O})$	8.47E-06	1.85	2.34E-06	1.75	6.97E-07
$Y(\mathrm{H_2O_2})$	6.20E-08	2.25	1.30E-08	2.62	2.12E-09
$Y(\mathrm{HO_2})$	5.03E-08	2.17	1.12E-08	2.28	2.31E-09
ρ	3.83E-08	1.84	1.07E-08	1.77	3.13E-09
T	8.83E-02	1.84	2.47E-02	1.74	7.42E-03
ρh	6.19E+01	1.82	1.75E+01	1.76	5.15E+00
U	1.65E-02	1.76	4.87E-03	1.43	1.81E-03

1.2 Piecewise linear advection term

Instead of using the Godunov solver to find a time-centered velocity, we use the velocities at the endpoints. In 1-D this is particularly easy, because the velocity is entirely determined by the divergence condition

$$\nabla \cdot U^n = \widehat{S}^n, \tag{1}$$

which is accomplished through the `macproject` subroutine.

This is then used to compute the advection term A^n. Then, for every MISDC interation k, we can use the scalar quantities to compute $\widehat{S}^{n+1,(k)}$, which then determines the velocity $U^{n+1,(k)}$ by enforcing the same divergence condition (**??**). We can then compute the time-endpoint advection term $A^{n+1,(k)}$.

We store both A^n as the variable `aofs_old` and $A^{n+1,(k)}$ as `aofs_new`. Then, updating the thermodynamic variables, we use the averaged quantity

$$\texttt{aofs}_{\texttt{avg}} := \frac{A^n + A^{n+1,(k)}}{2} \tag{2}$$

instead of the time-centered value $A^{n+1/2}$ obtained using the Godunov solver. Finally, when calling VODE, instead of using the piecewise constant source term $A^{n+1/2}$, we use the piecewise linear term given by A^n and $A^{n,(k)}$ at the endpoints.

These changes give rise to the following results.

$Y(H_2)$	2.71E-07	1.84	7.58E-08	1.67	2.38E-08
$Y(O_2)$	8.25E-06	1.87	2.26E-06	1.76	6.68E-07
$Y(N_2)$	9.89E-07	1.85	2.74E-07	1.72	8.30E-08
$Y(H_2O)$	8.16E-06	1.87	2.23E-06	1.77	6.55E-07
$Y(H_2O_2)$	6.16E-08	2.25	1.29E-08	2.68	2.02E-09
$Y(HO_2)$	5.04E-08	2.16	1.13E-08	2.25	2.36E-09
ρ	3.60E-08	1.86	9.91E-09	1.78	2.88E-09
T	8.49E-02	1.86	2.35E-02	1.76	6.95E-03
ρh	5.75E+01	1.83	1.62E+01	1.78	4.72E+00
U	1.98E-02	1.69	6.13E-03	1.45	2.24E-03

When the source term is chosen to be a piecewise constant given by the average (**??**) of the advection terms, very similar results are observed.

$Y(H_2)$	2.71E-07	1.83	7.59E-08	1.65	2.41E-08
$Y(O_2)$	8.24E-06	1.87	2.26E-06	1.75	6.74E-07
$Y(N_2)$	9.89E-07	1.85	2.74E-07	1.71	8.38E-08
$Y(H_2O)$	8.16E-06	1.87	2.23E-06	1.75	6.63E-07
$Y(H_2O_2)$	6.13E-08	2.26	1.28E-08	2.65	2.03E-09
$Y(HO_2)$	5.02E-08	2.16	1.12E-08	2.27	2.33E-09
ρ	3.60E-08	1.86	9.93E-09	1.77	2.92E-09
T	8.48E-02	1.85	2.35E-02	1.74	7.04E-03
ρh	5.74E+01	1.83	1.62E+01	1.76	4.78E+00
U	1.94E-02	1.67	6.07E-03	1.43	2.25E-03

1.3 Piecwise linear diffusion term

Now we will try to use piecewise linear term for diffusion. Making the required change in the code results in blow-up (i.e. instability). We see that for 256 gridpoints, halving the time step remedies the instability, but for 512 gridpoints, even dividing the time step by a quarter still results in instability.

As an attempt to understand the problem, I examined the results from the VODE integration more carefully. Subdiving the time-interval $[t_n, t_{n+1}]$ into a fixed number N subintervals of size Δt_{loc}, I used VODE to successively solve the ODE on the intervals $[t_n + k\Delta t_{loc}, t_n + (k+1)\Delta t_{loc}]$, where $0 \leq k < N$. Examining the solution produce by VODE over the subintervals revealed no apparent instability or spurious osciallations.

In the hope that increasing the number of MISDC iterations would help reduce the splitting error, I ran the code for varying values of `misdc_iterMAX`. I found that the more MISDC iterations, the faster the solution became unstable. In factor, when performing 15 MISDC iterations, the method cannot successfully complete even one time-step.

When I ran the code with only one MISDC iteration, instability was observed only after space-time refinement (i.e. with 2048 gridpoints).

Additionally, I tried doing one MISDC iteration with piecewise constants and one with piecewise linear. When the first iteration is constant, and the second linear, the method is unstable. When the first iteration is linear, and the second constant, the method appears to be stable.

2 MISDC-style Backward Euler step

Instead of solving the correction ODE using the VODE integrator, we instead solve the correction integral equation using a Backward Euler step. Results (using a grid of 256 points, and timestep

$\Delta t = 0.1 \times 10^{-5}$, refining both by a factor of 2 each run) are:

$Y(H_2)$	7.96E-08	2.02	1.96E-08	2.01	4.89E-09
$Y(O_2)$	1.16E-06	1.99	2.91E-07	1.99	7.34E-08
$Y(N_2)$	1.08E-07	1.98	2.76E-08	2.00	6.91E-09
$Y(H_2O)$	1.18E-06	2.00	2.96E-07	1.99	7.44E-08
$Y(H_2O_2)$	4.27E-09	1.94	1.11E-09	1.97	2.83E-10
$Y(HO_2)$	1.60E-08	1.98	4.05E-09	1.99	1.02E-09
ρ	1.04E-08	1.99	2.62E-09	1.99	6.62E-10
T	1.20E-02	2.00	3.01E-03	2.00	7.51E-04
ρh	1.00E+01	1.95	2.59E+00	1.99	6.51E-01
U	1.82E-02	2.09	4.30E-03	2.09	1.01E-03