High-order semi-Lagrangian numerical method for large-eddy simulations of reacting flows

Author(s): Becerra Sagredo, Julián Tercero

Publication Date: 2007

Permanent Link: https://doi.org/10.3929/ethz-a-005538138

Rights / License: In Copyright - Non-Commercial Use Permitted
High-order Semi-Lagrangian Numerical Method for Large-Eddy Simulations of Reacting Flows

A dissertation submitted to the
ETH Zurich

for the degree of
Doktor der Wissenschaften

presented by
Julián Tercero Becerra Sagredo
Dipl. Phys. Universidad Nacional Autónoma de México
born April 28, 1973
citizen of Mexico

accepted on the recommendation of

Prof. Dr. R. Jeltsch, examiner
Prof. Dr. W. P. Petersen, co-examiner
Dr. J. Gass, co-examiner

June, 2007
Acknowledgements

This work was developed at the Seminar for Applied Mathematics under the supervision of Prof. Dr. Rolf Jeltsch and Prof. Dr. Wesley P. Petersen as part of the project BFE-43099.

I am grateful to both my advisors in ways that cannot be expressed with words, principally for the opportunity to do this research and develop my own ideas, but also for the great conversations I had with them, from which I learned a big deal about all the different human relations that shape the world, in general, and in particular the world of science. With my advisors I enjoyed being part of the great human family. I am also thankful for their patience and interest during the review of my work. In particular I am very impressed by the help of Prof. Jeltsch in rewriting Chapter 7.

I also want to thank my co-advisor Dr. J. Gass and his student, my friend Sevket Baykal not only for all the knowledge and libraries provided for the combustion part, but also for all their respect, trust and patience.

My biggest thanks to all my colleagues in the Institute of Fluid Dynamics and in the Seminar for Applied Mathematics. Specially to Prof. Dr. Thomas Rösgen, Prof. Dr. Leonhard Kleiser, Dr. Michael Fey, Dr. Dirk Wilhelm, Dr. Marina Savelieva, Harish Kumar, Paolo Corti, Kersten Schmidt and Dr. Sebastian Tordeaux.

I want to acknowledge the help, moral and computational, of my
former physics colleagues in Mexico: Dr. Francisco Mandujano, Dr. Carlos Malaga and MSc. Ernesto González.

Finally I want to thank all my friends in Switzerland for all those great years.

This work is dedicated to my mother Margarita, my sister Cecy and all the *banda pesada*, including my friends Anna, Nina and Luz.
We develop a software package for the high-resolution simulation of a subsonic methane/air flame. This software is useful for future research in mechanisms to reduce the hazardous gas emissions from combustion turbines.

First we choose the mathematical models. The fluid model is given by the Navier-Stokes equations for a mixture of ideal gases. The model for the turbulence is a Large-eddy Simulation (LES) dynamical model that uses a transport equation for the subgrid kinetic energy. The model of the chemical reactions is an unsteady flamelet model based on transport equations for the mixture fraction and a reaction progress variable. This last model allows for extinctions and reignitions using transient flamelets based on finite rate chemistry.

The resulting model is a set of coupled transport equations that we need to solve numerically. To this end we have designed a non-diffusive and conservative transport method based on the forward Semi-Lagrangian (FSL) numerical method. The interpolations from the Lagrangian mesh points to a fix mesh are the most important and time consuming procedures. Interpolations that are conservative and create low or zero numerical diffusion and dispersion are desirable. We have found a family of compact support, convolution splines that conserve discrete moments of the data during the interpolations. We call them Z-splines and they were known before this work only up to the cubic Z-spline (equivalent to Bessel interpolation). They can be seen as compact support, piecewise polynomial approximations.
of a perfect filter, the $sinc$ function. The approximations of derivatives produced by the Z-splines are equivalent to the finite difference operators found by inverting the Taylor series expansions from the neighboring points.

Basic tests are performed on our Z-spline FSL method, showing its conservation properties and high accuracy compared to classic Eulerian and Semi-Lagrangian techniques. Then the method is used to simulate acoustic waves and implement non-reflecting outflow and inflow boundaries.

Finally the Z-spline FSL method is used to simulate a methane/air flame called Sandia flame D. First we simulate a cold jet of air and later we include the chemical model variables to simulate the flame. For these simulations we use a spherical coordinates mesh where singular derivatives and high frequencies at the polar axis are controlled.
Resumen

Desarrollamos un paquete informático para la simulación de alta resolución de una llama de metano y aire en régimen subsónico. Este programa es útil para la investigación futura de diversos mecanismos para reducir las emisiones de gases nocivos para la salud producidas por turbinas de combustión de gas.

Primero escogemos los modelos matemáticos. El modelo del fluido está dado por las ecuaciones de Navier-Stokes para una mezcla de gases ideales. El modelo de la turbulencia es un modelo dinámico de simulación de grandes remolinos (LES) basado en una ecuación de transporte para la energía cinética no resuelta por la malla computacional. El modelo de las reacciones químicas es un modelo de llamículas (elementos de llama) variables en el tiempo, basado en ecuaciones de transporte para la fracción de mezcla y una variable de progreso de la reacción. Este último modelo permite extinciones y re-igniciones usando llamículas transientes basadas en reacciones químicas de velocidad finita.

El modelo conjunto es un sistema acoplado de ecuaciones de transporte que debemos resolver con un método numérico. Para ello hemos diseñado un método numérico de transporte que produce difusión y dispersión numérica nula o muy baja basado en el método numérico semi-Lagrangiano hacia delante (FSL). Las interpolaciones desde la malla Lagrangiana a la malla fija son los procesos más importantes y que consumen la mayorías del tiempo de cómputo. Es deseable que el proceso de interpolación sea conservativo y que produzca mínima o
muy baja difusión y dispersión numérica. Durante este trabajo hemos encontrado una familia de splines de convolución y de soporte compacto que conservan los momentos estadísticos discretos de los datos durante la interpolación. Los nombramos splines-Z y eran conocidos únicamente hasta el cúbico (equivalente a la interpolación de Bessel). Pueden entenderse como aproximaciones polinomiales a pedazos y con soporte compacto de un filtro perfecto, la función sinc. Las aproximaciones de derivadas producidas por los splines-Z son equivalentes a los operadores de diferencias finitas obtenidas al invertir el sistema de expansiones en serie de Taylor desde los puntos vecinos.

El método FSL splines-Z es sometido a pruebas básicas donde se muestran sus propiedades de conservación y su alta precisión comparado con los métodos clásicos Eulerianos y semi-Lagrangianos. Más adelante, el método es usado para la simulación de ondas acústicas y para poner en práctica condiciones de frontera que no reflejan las ondas en las entradas o salidas del fluido.

Finalmente, el método numérico FSL splines-Z es usado para simular una llama de metano y aire llamada Sandia D. Primero simulamos un chorro frío de aire y más tarde incluimos las variables químicas para simular la llama. En estas simulaciones usamos una malla dada en coordenadas esféricas donde derivadas singulares y altas frecuencias generadas en el eje polar pueden ser controladas.
Zusammenfassung


Contents

List of Figures xviii
List of Tables xix
Nomenclature xxi

I Introduction 1

II The Equations for Turbulent and Reacting Flows 7

1 The Conservation Laws 9

2 Large-Eddy Simulation Technique 13

3 The Turbulence Model 17

4 The Combustion Model 21
5 The Lagrangian Form of the Governing Equations | 27

III Numerical Method | 29

6 The Forward-trajectory Semi-Lagrangian Method | 31
6.1 A High-order FSL Method | 33
6.2 Review of Semi-Lagrangian Methods | 40
6.3 Stability and Accuracy of the FSL | 41

7 High-order Discrete Representation of Functions | 43
7.1 The Z-splines | 43
7.1.1 Examples of general Z-splines | 51
7.2 Cardinal Z-splines | 54
7.2.1 Approximation of Functions by cardinal Z-splines | 58
7.3 Multi-dimensional Z-splines | 66
7.3.1 Interpolation over a smoothly deformed Cartesian mesh | 67

IV Results | 69

8 Fundamental Test Cases | 71
8.1 Two-dimensional Scalar transport | 71
8.1.1 Solid body rotation | 72
8.1.2 Taylor-Green vortices | 75
8.1.3 Z-spline filters | 77
List of Figures

6.1 The one-dimensional flux splitting technique (a) and the method of transport (b). .................. 32

6.2 The forward-trajectory semi-Lagrangian (FSL) scheme using a bilinear deformed mesh and forward-Euler trajectory integration. .................. 35

7.1 Cubic Z-spline interpolation (equivalent to cubic Bessel interpolation). .................. 46

7.2 The arbitrarily spaced cubic Z-spline basis function (Example 7.1.8) for \( a_1 = a_2 = 1 \) and \( a_3 = a_4 = 0.5 \). . . 52

7.3 One-sided, cubic, cardinal Z-spline basic functions with a boundary on the left. .................. 53

7.4 The first four cardinal Z-spline basis functions \( \tilde{Z}_m \) and their Fourier transforms \( Z'_m \). .................. 55

7.5 The absolute value of the cardinal Z-spline basis functions for \( m = 0, 1, 2, 3, 4, 9, 29 \) and the bounding function \( |1/(\pi x)| \). .................. 62

7.6 \( L_2 \) error between \( sinc(x) \) and the first 100 \( Z \)-splines for \( x \in [-5,5] \). .................. 64
7.7 The sinc function and the cardinal Z-spline basis functions for \( m = 0, 1, 2, 4, 9 \) and 19. ............................ 65

7.8 Two-dimensional deformed Cartesian mesh. ............... 67

8.1 Contours of the function \( \alpha \), equation (8.3), for the solid body rotation test. Initial condition (top left). Results after one revolution using the FSL scheme with \( Z_0 \) (top right), \( Z_1 \) (bottom left) and \( Z_2 \) (bottom right) interpolations. ......................... 73

8.2 \( L_2 \)-error for the Taylor-Green vortices using linear \( Z_0 \) (solid lines) and cubic \( Z_1 \) (dashed lines) volume elements. \( Z_0 \), \( Z_1 \) and \( Z_2 \) interpolations are compared. . . 76

8.3 Vector field of the Taylor-Green vortices (8.9) (left) and the scalar initial condition (8.11) (right). ............ 76

8.4 Contours of the scalar field (8.11) transported by the velocity field (8.9) using the FSL numerical method with \( Z_1 \) volume elements and \( Z_2 \) interpolation. The results are shown from top to bottom after 250, 500 and 1000 time steps with \( \Delta t = 1/1600 \) and 128 x 128 cells. The plots show the results using the FSL method (left) and the improved results using a Z-spline filter (right). ....................... 78

8.5 Numerical solution to the one-dimensional Burgers’ equation with sinusoidal initial condition, mesh spacing \( \Delta x = 0.01 \) and time step \( \Delta t = 0.001 \). The solution is shown as a function of time (top) and after 300 time steps (bottom) using the FSL with cubic \( Z_1 \) interpolation (left) and with the model for discontinuities (right). ......................... 80
8.6 Waves in a doubly periodic domain using the FSL with linear $Z_0$ (top), cubic $Z_1$ (middle) and quintic $Z_2$ bidimensional interpolations (bottom). The left column shows contours of the altitude for shallow-water waves (21 contours from 0.99 to 1.03). The right column shows contours of the density for inviscid Navier-Stokes waves (21 contours from 1 to 1.06). The results are obtained after 45 time steps with $\Delta t = 0.01$ and $64 \times 64$ cells. .......................................................... 82

8.7 Density waves in an open field using the FSL with cubic $Z_1$ interpolation. The conditions of the simulation and the plot are the same as for the doubly periodic Navier-Stokes waves of the previous section. The figure shows the initial acoustic wave leaving the domain without reflections. The snapshots are taken after 5 (top-left), 25 (top-right), 35 (bottom-left) and 70 time steps (bottom-right). ................................. 84

9.1 Spherical polar coordinates mesh used for the simulation of the cold jet and flame. ................................. 89

9.2 Ordering of the ghost cells at the pole. ...................... 90

9.3 Jet velocity at the inflow boundary condition, obtained from the experimental data reported in [2]. ............. 91

9.4 Subgrid kinetic energy of the cold jet showing the turbulent structure at $y = 0$ and $t = 0.01$ s. ................. 93

9.5 Velocity magnitude of the cold jet at $y = 0$ and $t = 0.01$ s. ......................................................... 94

9.6 Temperature field of the Sandia flame D at $y = 0$ and $t = 0.002$ s. ..................................................... 95

9.7 Mixture fraction $Z$ field of the Sandia flame D at $y = 0$ and $t = 0.002$ s. .................................................. 96
9.8 Reaction progress variable $c$ of the Sandia flame D at $y = 0$ and $t = 0.002$ s. ........................................ 97

9.9 Averaged velocity magnitude of the Sandia flame D along the centerline. Lines (simulation), symbols (experiment). ........................................ 98

9.10 Mean values of scalars along the centerline. Lines (simulation), symbols (experiment). ......................... 99

9.11 Mean values of scalars along the centerline. Lines (simulation), symbols (experiment). ......................... 100
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Range of the input variables in the flamelet library.</td>
<td>26</td>
</tr>
<tr>
<td>7.1</td>
<td>Interpolation error of the Z-splines for the test function $\sin(\pi x)$.</td>
<td>65</td>
</tr>
<tr>
<td>8.1</td>
<td>Errors of the FSL method for the bidimensional solid body rotation after one period. The linear and cubic methods are results obtained for traditional SL methods using B-spline interpolation and exact trajectories [7].</td>
<td>74</td>
</tr>
<tr>
<td>8.2</td>
<td>Discrete moments of the pulse used in the solid body rotation test after one period.</td>
<td>74</td>
</tr>
<tr>
<td>9.1</td>
<td>Inlet boundary conditions for the Sandia flame D, the species fractions are given by their specific mole numbers ($Y_m/W_m$).</td>
<td>92</td>
</tr>
</tbody>
</table>
### Latin symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_k)</td>
<td>(J s^2/m^3)</td>
<td>Kolmogorov constant</td>
</tr>
<tr>
<td>(a_i)</td>
<td>–</td>
<td>Arbitrary intervals for the Z-spline (\tilde{Z}_1)</td>
</tr>
<tr>
<td>(a_{ij})</td>
<td>–</td>
<td>Constants in the Runge-Kutta ODE solvers</td>
</tr>
<tr>
<td>(A_m)</td>
<td>–</td>
<td>Finite differences matrix</td>
</tr>
<tr>
<td>(b_i)</td>
<td>–</td>
<td>Constants in the Runge-Kutta ODE solvers</td>
</tr>
<tr>
<td>(c)</td>
<td>–</td>
<td>Reaction progress variable</td>
</tr>
<tr>
<td>(c_f)</td>
<td>–</td>
<td>Reaction progress variable at fuel inlet</td>
</tr>
<tr>
<td>(c_{p,m})</td>
<td>(J/kgK)</td>
<td>Specific heat capacity at constant pressure</td>
</tr>
<tr>
<td>(c_{V,m})</td>
<td>(J/kgK)</td>
<td>Specific heat capacity at constant volume</td>
</tr>
<tr>
<td>(C_m)</td>
<td>–</td>
<td>Proportionality constant of convergence</td>
</tr>
<tr>
<td>(C_v)</td>
<td>–</td>
<td>Eddy viscosity proportionality constant</td>
</tr>
<tr>
<td>(C_\varepsilon)</td>
<td>–</td>
<td>Subgrid kinetic energy dissipation constant</td>
</tr>
<tr>
<td>(D)</td>
<td>(m)</td>
<td>Diameter of fuel jet</td>
</tr>
<tr>
<td>(\mathcal{D})</td>
<td>–</td>
<td>Diagonal matrix of factorials</td>
</tr>
<tr>
<td>(D/Dt)</td>
<td>(1/s)</td>
<td>Total time derivative</td>
</tr>
<tr>
<td>(D_{m})</td>
<td>(m^2/s)</td>
<td>Molecular diffusion of the (m – th) species</td>
</tr>
<tr>
<td>(D_c)</td>
<td>(m^2/s)</td>
<td>Diffusion coefficient for (c)</td>
</tr>
<tr>
<td>(D_Z)</td>
<td>(m^2/s)</td>
<td>Molecular diffusion coefficient of (Z)</td>
</tr>
<tr>
<td>(D_{\text{sub}})</td>
<td>(J/(m^3 s))</td>
<td>Subgrid kinetic energy dissipation</td>
</tr>
<tr>
<td>(D_\chi)</td>
<td>(m^2/s)</td>
<td>Diffusion coefficient for (\chi)</td>
</tr>
<tr>
<td>Symbol</td>
<td>Units</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>$c$</td>
<td>J/kg</td>
<td>Internal energy per unit mass</td>
</tr>
<tr>
<td>$err(Z_m)$</td>
<td>–</td>
<td>$L_2$-error of the Z-spline of a given function</td>
</tr>
<tr>
<td>$E$</td>
<td>J</td>
<td>Total energy</td>
</tr>
<tr>
<td>$f$</td>
<td>–</td>
<td>Scalar function</td>
</tr>
<tr>
<td>$f_i$</td>
<td>–</td>
<td>One-dimensional scalar data</td>
</tr>
<tr>
<td>$f_{i,j,k}$</td>
<td>–</td>
<td>Scalar data over a 3D mesh</td>
</tr>
<tr>
<td>$G_j$</td>
<td>–</td>
<td>Filter kernel</td>
</tr>
<tr>
<td>$h$</td>
<td>J/kg</td>
<td>Enthalpy per unit mass</td>
</tr>
<tr>
<td>$h$</td>
<td>m</td>
<td>Water surface level</td>
</tr>
<tr>
<td>$H_{s\text{ub}}$</td>
<td>J / (m$^2$s)</td>
<td>Subgrid heat flux vector</td>
</tr>
<tr>
<td>$I$</td>
<td>–</td>
<td>Integer number</td>
</tr>
<tr>
<td>$I_{el}$</td>
<td>–</td>
<td>Interpolation (Eulerian to Lagrangian mesh)</td>
</tr>
<tr>
<td>$I_{le}$</td>
<td>–</td>
<td>Interpolation (Lagrangian to Eulerian mesh)</td>
</tr>
<tr>
<td>$J$</td>
<td>–</td>
<td>Integer number</td>
</tr>
<tr>
<td>$K$</td>
<td>–</td>
<td>Integer number</td>
</tr>
<tr>
<td>$k$</td>
<td>1/m</td>
<td>Wave number</td>
</tr>
<tr>
<td>$k_c$</td>
<td>1/m</td>
<td>Cut-off wave number</td>
</tr>
<tr>
<td>$k_{s\text{ub}}$</td>
<td>J/kg</td>
<td>Subgrid kinetic energy per unit mass</td>
</tr>
<tr>
<td>$l_i$</td>
<td>–</td>
<td>Vectors in the Runge-Kutta ODE solvers</td>
</tr>
<tr>
<td>$L_{e\text{m}}$</td>
<td>–</td>
<td>Lewis number of the $m$-th species</td>
</tr>
<tr>
<td>max</td>
<td>–</td>
<td>Maximum value</td>
</tr>
<tr>
<td>min</td>
<td>–</td>
<td>Minimum value</td>
</tr>
<tr>
<td>$N_I$</td>
<td>–</td>
<td>Total number of nodes in direction $I$</td>
</tr>
<tr>
<td>$p$</td>
<td>N/m$^2$</td>
<td>Pressure</td>
</tr>
<tr>
<td>$p_{m,j}$</td>
<td>–</td>
<td>Interpolation polynomial</td>
</tr>
<tr>
<td>$P$</td>
<td>(m, m, m)</td>
<td>Point in a 3D space</td>
</tr>
<tr>
<td>$P$</td>
<td>–</td>
<td>Taylor expansion polynomial</td>
</tr>
<tr>
<td>$P_{s\text{ub}}$</td>
<td>J / m$^3$s</td>
<td>Subgrid kinetic energy production</td>
</tr>
<tr>
<td>$Pr_t$</td>
<td>–</td>
<td>Turbulent Prandtl number</td>
</tr>
<tr>
<td>$q$</td>
<td>–</td>
<td>Polynomial</td>
</tr>
<tr>
<td>$q_i$</td>
<td>J/kg/m$^2$s</td>
<td>Heat flux vector</td>
</tr>
<tr>
<td>$Q$</td>
<td>(m, m, m)</td>
<td>Point in a 3D space</td>
</tr>
<tr>
<td>$r$</td>
<td>m</td>
<td>Radial coordinate</td>
</tr>
<tr>
<td>$R$</td>
<td>–</td>
<td>Last interpolation (or remeshing)</td>
</tr>
<tr>
<td>$R_u$</td>
<td>J / (K mol)</td>
<td>Universal gas constant</td>
</tr>
<tr>
<td>$R_n$</td>
<td>–</td>
<td>Remainder of the Taylor series</td>
</tr>
<tr>
<td>Symbol</td>
<td>Units</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>$Re$</td>
<td></td>
<td>Reynold’s number</td>
</tr>
<tr>
<td>$S$</td>
<td></td>
<td>Right hand side vector of a transport equation</td>
</tr>
<tr>
<td>$S_i$</td>
<td></td>
<td>Vector components of $S$</td>
</tr>
<tr>
<td>$S_{i,j}$</td>
<td>$1/s$</td>
<td>Rate of strain</td>
</tr>
<tr>
<td>$Sc_t$</td>
<td></td>
<td>Turbulent Schmidt number</td>
</tr>
<tr>
<td>$t$</td>
<td>$s$</td>
<td>Time</td>
</tr>
<tr>
<td>$T$</td>
<td>$K$</td>
<td>Temperature</td>
</tr>
<tr>
<td>$T_r$</td>
<td>$s$</td>
<td>Rotation period</td>
</tr>
<tr>
<td>$u$</td>
<td>$m/s$</td>
<td>Velocity vector</td>
</tr>
<tr>
<td>$u_i$</td>
<td>$m/s$</td>
<td>Velocity components</td>
</tr>
<tr>
<td>$V$</td>
<td>$m^3$</td>
<td>Volume</td>
</tr>
<tr>
<td>$V$</td>
<td></td>
<td>Vandermonde matrix</td>
</tr>
<tr>
<td>$V_{i,m}$</td>
<td>$m/s$</td>
<td>Diffusion velocity vector of the $m-th$ species</td>
</tr>
<tr>
<td>$w_m$</td>
<td>$kg/m^3$</td>
<td>Mass reaction source per unit volume</td>
</tr>
<tr>
<td>$w_m$</td>
<td>$kg$</td>
<td>Molecular weight of the $m-th$ species</td>
</tr>
<tr>
<td>$x$</td>
<td>$m$</td>
<td>Position vector</td>
</tr>
<tr>
<td>$x$, $y$, $z$</td>
<td>$m$</td>
<td>3D Cartesian coordinates</td>
</tr>
<tr>
<td>$x_j$</td>
<td>$m$</td>
<td>Cartesian coordinates</td>
</tr>
<tr>
<td>$Y_m$</td>
<td></td>
<td>Chemical species mass fractions</td>
</tr>
<tr>
<td>$Y_F$</td>
<td></td>
<td>Fuel mass fraction</td>
</tr>
<tr>
<td>$Y_{F,u}$</td>
<td></td>
<td>Fuel mass fraction at fuel inlet</td>
</tr>
<tr>
<td>$Y_O$</td>
<td></td>
<td>Oxidizer mass fraction</td>
</tr>
<tr>
<td>$Z$</td>
<td></td>
<td>Mixture fraction</td>
</tr>
<tr>
<td>$Z_m$</td>
<td></td>
<td>$Z$-spline of continuity $m$</td>
</tr>
<tr>
<td>$\hat{Z}$</td>
<td></td>
<td>Fourier transform of $Z$</td>
</tr>
<tr>
<td>$\hat{Z}_m$</td>
<td></td>
<td>$Z$-spline basis function of continuity $m$</td>
</tr>
</tbody>
</table>

**Greek symbols**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td></td>
<td>Vector of transport variables</td>
</tr>
<tr>
<td>$\alpha_1$</td>
<td>$K^{0.5}$</td>
<td>Constant parameter in Sutherland’s law</td>
</tr>
<tr>
<td>$\alpha_2$</td>
<td>$K$</td>
<td>Constant parameter in Sutherland’s law</td>
</tr>
<tr>
<td>$\beta_m$</td>
<td></td>
<td>Convergence exponent</td>
</tr>
<tr>
<td>$\delta_{ij}$</td>
<td></td>
<td>Kronecker’s delta</td>
</tr>
</tbody>
</table>

xxiii
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta$</td>
<td>$m$</td>
<td>Computational grid spacing</td>
</tr>
<tr>
<td>$\Delta \alpha$</td>
<td></td>
<td>Discrete increment of the variable $\alpha$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>$m^2/s^3$</td>
<td>Dissipation rate of kinetic energy</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>$m$</td>
<td>Curvilinear coordinate</td>
</tr>
<tr>
<td>$\zeta_{i,\text{sub}}$</td>
<td>$kg/(sm^2)$</td>
<td>Subgrid mixture fraction mass flux</td>
</tr>
<tr>
<td>$\eta$</td>
<td>$m$</td>
<td>Kolmogorov length scale</td>
</tr>
<tr>
<td>$\eta$</td>
<td></td>
<td>Curvilinear coordinate</td>
</tr>
<tr>
<td>$\theta$</td>
<td>$rad$</td>
<td>Polar angle</td>
</tr>
<tr>
<td>$\theta_{i,m}^{\text{sub}}$</td>
<td>$kg/(sm^2)$</td>
<td>Diffusive mass flux vector of the $m$ th species</td>
</tr>
<tr>
<td>$\mu$</td>
<td>$kg/(ms)$</td>
<td>Molecular viscosity</td>
</tr>
<tr>
<td>$\mu_m$</td>
<td>$kg/(ms)$</td>
<td>Chemical species molecular viscosities</td>
</tr>
<tr>
<td>$\nu$</td>
<td>$m^2/s$</td>
<td>Kinematic viscosity</td>
</tr>
<tr>
<td>$\nu_t$</td>
<td>$m^2/s$</td>
<td>Eddy viscosity</td>
</tr>
<tr>
<td>$\nu_m$</td>
<td></td>
<td>Molar stoichiometric coefficients</td>
</tr>
<tr>
<td>$\xi$</td>
<td>$m$</td>
<td>Curvilinear coordinate</td>
</tr>
<tr>
<td>$\rho$</td>
<td>$kg/m^3$</td>
<td>Density</td>
</tr>
<tr>
<td>$\sigma_{i,\text{sub}}$</td>
<td>$Nm$</td>
<td>Subgrid viscous work</td>
</tr>
<tr>
<td>$\tau$</td>
<td>$s$</td>
<td>Local residence time</td>
</tr>
<tr>
<td>$\tau_{ij}$</td>
<td>$kg/(ms^2)$</td>
<td>Stress tensor</td>
</tr>
<tr>
<td>$\tau_{ij}^{\text{sub}}$</td>
<td>$kg/(ms^2)$</td>
<td>Subgrid stress tensor</td>
</tr>
<tr>
<td>$\phi$</td>
<td>$rad$</td>
<td>Azimuthal angle</td>
</tr>
<tr>
<td>$\phi_i$</td>
<td></td>
<td>Generic mesh function</td>
</tr>
<tr>
<td>$\phi_{i,m}^{\text{sub}}$</td>
<td>$kg/(sm^2)$</td>
<td>Subgrid species mass flux</td>
</tr>
<tr>
<td>$\chi$</td>
<td>$1/s$</td>
<td>Scalar dissipation rate of the mixture fraction</td>
</tr>
</tbody>
</table>

### Subscripts

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>Integer indices</td>
</tr>
<tr>
<td>$j$</td>
<td>Integer indices</td>
</tr>
<tr>
<td>$k$</td>
<td>Integer indices</td>
</tr>
<tr>
<td>$m$</td>
<td>Integer index</td>
</tr>
<tr>
<td>$p$</td>
<td>Integer index</td>
</tr>
<tr>
<td>$s$</td>
<td>Integer index</td>
</tr>
<tr>
<td>$t$</td>
<td>Turbulent</td>
</tr>
</tbody>
</table>
### Superscripts

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pure fuel or pure oxidizer streams</td>
</tr>
<tr>
<td>(n)</td>
<td>Current number of time steps</td>
</tr>
<tr>
<td>(sub)</td>
<td>Subgrid</td>
</tr>
<tr>
<td>((p))</td>
<td>(p)-th derivative</td>
</tr>
<tr>
<td>(T)</td>
<td>Transposed</td>
</tr>
<tr>
<td>(*)</td>
<td>Intermediate variable to reduce storage</td>
</tr>
</tbody>
</table>

### Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>Absorbing boundary condition</td>
</tr>
<tr>
<td>CFD</td>
<td>Computational Fluid Dynamics</td>
</tr>
<tr>
<td>CFL</td>
<td>Courant-Friedrichs-Lewy number or condition</td>
</tr>
<tr>
<td>ETH</td>
<td>Swiss Federal Institute of Technology</td>
</tr>
<tr>
<td>FLATRA</td>
<td>Transient Flamelet solver developed by Ferreira</td>
</tr>
<tr>
<td>FSL</td>
<td>Forward Semi-Lagrangian Transport Method</td>
</tr>
<tr>
<td>LES</td>
<td>Large-Eddy Simulation</td>
</tr>
<tr>
<td>LTNT</td>
<td>Laboratory of Thermodynamics in Emerging Technologies</td>
</tr>
<tr>
<td>MoT</td>
<td>Method of Transport</td>
</tr>
<tr>
<td>MPI</td>
<td>Message Passing Interface</td>
</tr>
<tr>
<td>ODE</td>
<td>Ordinary differential equation</td>
</tr>
<tr>
<td>PML</td>
<td>Perfectly matched layer</td>
</tr>
<tr>
<td>RANS</td>
<td>Reynold’s averaged Navier-Stokes</td>
</tr>
<tr>
<td>RHS</td>
<td>Right hand side of the equation or sources</td>
</tr>
<tr>
<td>RK</td>
<td>Runge-Kutta ODE numerical solver</td>
</tr>
<tr>
<td>RK3</td>
<td>Runge-Kutta 3rd order ODE numerical solver</td>
</tr>
<tr>
<td>SAM</td>
<td>Seminar of Applied Mathematics</td>
</tr>
<tr>
<td>SL</td>
<td>Semi-Lagrangian Transport Method</td>
</tr>
<tr>
<td>TLFM</td>
<td>Transient Laminar Flamelet Model</td>
</tr>
<tr>
<td>VODPK</td>
<td>Variable-coefficient Ordinary Differential Equation solver</td>
</tr>
</tbody>
</table>
Part I

Introduction
This work presents the theory and the test results produced during the development of a high-order computational tool for the simulation of turbulent and reactive flows.

The reason for creating this software is for future research in mechanisms that can be implemented to diminish the hazardous gas emissions from combustion turbines. Gas and oil turbines produce most of the electricity consumed in societies nowadays and therefore the reduction of these emissions is directly related to improving the standard of life of the general population. Moreover, new hazardous emissions limits are being imposed in Europe, among other regions of the world, and it is obligatory for the next generation of combustion turbines to meet the new norms.

The reduction of the pollutants in the combustion process is a function of several quantities that are local and highly affected by the dynamics of the flow inside the combustion chamber. Available analytical tools are limited in their ability to solve the complicated dynamics of turbulent flows and therefore computer software must be developed in order to solve the problem numerically.

Several numerical techniques have been developed by many scientists to advance towards the solution of this problem. Nevertheless, there is much room for improvement. Some of the things that must be improved are the accuracy, the order of convergence, and the fully coupled multi-dimensionality of the numerical method, among others. The main achievement of this work is to provide a simple, fast, high-order, truly multi-dimensional, fluid solver. The solver is based on the forward-trajectory semi-Lagrangian numerical method. The interpolation from the deformed Lagrangian mesh to the original regular mesh is the most important and time-consuming numerical procedure. Our interpolation is based on a high-order discrete representation of functions by a family of spline curves developed during this work. The new splines are named Z-splines: they are convolution Hermite splines of compact support constructed from the Taylor finite difference approximations of derivatives. The Z-splines have the desirable property of preserving discrete moments of the interpolated data. The preservation of the moments is directly related to
the conservation of important physical properties like the total mass
and the reduction of spurious forces or torques. For equispaced data,
the Z-splines are compact approximations to the sinc function, i.e.,
the Fourier transform of a perfect low-pass filter.

The direct simulation of all the features of a realistic turbulent and
reactive flow seems impossible for current computers and it will likely
be the same in the foreseeable future. The problem is that very small
spatial and temporal structures are produced by the turbulence and
reactions, and breathtaking amounts of memory and computational
times are necessary to solve for these features numerically.

In order to avoid this problem, a technique known as large-eddy si-
mulation (LES) has been developed and tested during the last two
decades. The idea behind LES is to separate the spatial structures
into supergrid structures represented in a coarse numerical grid, and
subgrid structures that are replaced by subgrid models. The success
of the LES can be expected from the fact that interactions of large
structures of the fluid motion, namely the interaction of large vor-
tices, dominates the dynamics of the fluid motion, while the subgrid
scales are expected to have less specific effects and a more universal
behavior. This is quite close to the reality of highly turbulent flows
observed by experimentalists. But LES should be expected to fail
for problems near to the transition to turbulence where the dynamics
are highly dependent on the interaction between the coarse and the
subgrid structures.

LES is implemented in this work using recently developed numeri-
cal methods. As previously mentioned, a novel high-order forward-
trajectory semi-Lagrangian fluid solver is used to simulate the dyna-
mics of the supergrid structures, while the subgrid structures are
represented by a dynamical model taken from the literature. This
model for the turbulence is based on a transport equation for the sub-
grid kinetic energy and it is constructed assuming the Kolmogorov
energy cascade observed in three-dimensional, well-developed, tur-
bulent flows.

Realistic combustion processes are also extremely difficult to simu-
late with current computers. Hundreds of reactions must be solved simultaneously for hundreds of species in order to simulate real gas or oil combustion. These processes consume even more computational time and memory than the turbulence. Therefore it is necessary to create a reduced model that can reproduce the most important features of the coupled system of reactive equations. To this end, this work adopts the research done by Sevket Baykal from the Laboratory of Thermodynamics in Emerging Technologies (LTNT) of the Swiss Federal Institute of Technology (ETH-Zurich), who created computational software for modelling the combustion process by a reasonable number of transport equations. These libraries are coupled with the three-dimensional high-order, truly multi-dimensional LES code to produce the software necessary for the accurate simulation of turbulent and reactive flows.
Part II

The Equations for Turbulent and Reacting Flows
Chapter 1

The Conservation Laws

The governing equations of motion for mass, momentum, energy and species fractions in a compressible, reacting fluid, known as the Navier-Stokes equations, are written in conservative form as follows:

\[
\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0, 
\]

\[
\frac{\partial \rho u_i}{\partial t} + \frac{\partial }{\partial x_j} \left[ \rho u_i u_j + p \delta_{ij} - \tau_{ij} \right] = 0, 
\]

\[
\frac{\partial \rho E}{\partial t} + \frac{\partial }{\partial x_i} \left[ (\rho E + p) u_i + q_i - u_j \tau_{ji} \right] = 0, 
\]

\[
\frac{\partial \rho Y_m}{\partial t} + \frac{\partial }{\partial x_i} \left[ \rho Y_m (u_i + V_{i,m}) \right] = \dot{w}_m, \quad m = 1:N. 
\]

In these equations, Einstein’s summation convention \( a_i b_i = \sum_i a_i b_i \) is used for 3-vectors and tensors; \( x_i \) and \( t \) are the Cartesian coordinates vector and the time, respectively; \( \rho \) is the mass density; \( u_i \) is the velocity vector; \( p \) is the pressure; \( \tau_{ij} \) is the viscous stress tensor; \( E \) is the total energy per unit mass; \( q_i \) is the heat flux vector; and \( Y_m, V_{i,m} \) and \( \dot{w}_m \) are the individual species mass fraction, diffusion velocities and mass reaction rate per unit volume, respectively.
The sum of the mass fractions must be equal to one, i.e.,

\[ \sum_{m=1}^{N} Y_m = 1. \]  \hfill (1.5)

Therefore, the sum of the \( N \) conservation laws for the mass fractions (1.4) reduces to the conservation of mass (1.1). Although one equation can be eliminated, we keep them in this form because it is useful to explain the combustion model of Chapter 5 where the \( N \) equations (1.4) are reduced even further.

The viscous stress tensor is

\[ \tau_{ij} = \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \left( \frac{\partial u_k}{\partial x_k} \right) \delta_{ij} \]  \hfill (1.6)

where \( \mu \) is the molecular viscosity coefficient, in general a function of the temperature and species fractions.

In our model, \( \mu \) is approximated by Sutherland’s viscosity law [57]

\[ \mu(T) = \frac{\alpha_1 T^{1.5}}{\alpha_2 + T} \sum_{m=1}^{N} \mu_m Y_m \]  \hfill (1.7)

where \( T \) is the temperature, \( \mu_m \) is the \( m \)-th species molecular viscosity at normal conditions of temperature and pressure, and the parameters \( \alpha_1 = 0.085 \ K^{0.5} \) and \( \alpha_2 = 110.4 \ K \) are chosen as for the results found for air [57]. This law results from a kinetic theory by Sutherland (1893) using an idealized intermolecular-force potential.

The diffusion velocities for the mass fractions are approximated by Fick’s law

\[ V_{i,m} = -\frac{D_m}{Y_m} \left( \frac{\partial Y_m}{\partial x_i} \right) \]  \hfill (1.8)

where \( D_m \) is the averaged molecular diffusion coefficient of the \( m \)-th species in the mixture.

The pressure is determined from an equation of state that, in the case
of a perfect gas mixture, is given by

\[ p = \rho R_u T \sum_{m=1}^{N} \frac{Y_m}{W_m}. \] (1.9)

Here, \( R_u \) is the universal gas constant and \( W_m \) is the \( m \)-th species molecular weight.

The total energy per unit volume is composed of the internal energy per unit mass and the kinetic energy as follows:

\[ \rho E = \rho \left( e + \frac{1}{2} u_k u_k \right). \] (1.10)

The internal energy per unit mass is given by

\[ e = \sum_{m=1}^{N} Y_m h_m - \frac{p}{\rho} \] (1.11)

where \( h_m \) is the \( m \)-th species enthalpy per unit mass, determined by the caloric equation of state

\[ h_m = \Delta h_{f,m}^0 + \int_{T_0}^{T} c_{p,m}(T)dT, \] (1.12)

where \( \Delta h_{f,m}^0 \) is the heat of formation at temperature \( T_0 \) and \( c_{p,m} \) is the specific heat capacity at constant pressure of the \( m \)-th species.

In our model, the specific heat capacity at constant pressure \( c_{p,m} \) and the specific heat capacity at constant volume \( c_{V,m} \) are approximated as constants and therefore

\[ h_m = c_{p,m} T \] (1.13)

and

\[ e = \sum_{m=1}^{N} Y_m c_{V,m} T. \] (1.14)
Chapter 2

Large-Eddy Simulation Technique

The governing equations (1.1)-(1.4) contain most of the physics of compressible turbulent flows with chemical reactions in a gaseous mixture. No turbulence or combustion model would be needed if an analytical solution were available. Unfortunately, this solution is not known and only a few simple cases can be studied by analytical techniques. Numerical methods are necessary to study the solutions of the governing equations in most of the applications that are important in practice.

A major difficulty in computing a direct numerical solution of the governing equations (1.1)-(1.4) is that its structures usually contain an extremely wide range of scales that are difficult to represent accurately with the computer. Most of the turbulent flows of practical importance are impossible to simulate directly with the current memory sizes and computational times. Some averaging techniques, like RANS (Reynold’s averaged Navier-Stokes), have been used to obtain a simplified system of equations that account for averaged turbulent effects but they provide little to no dynamics. Large-eddy simulation (LES) is a widely accepted numerical technique to deal
with turbulent flows where dynamical effects are important.

The idea behind LES (e.g. see [16], [28], [41] and [29]) is to separate each of the physical variables into a supergrid (i.e., resolved) component, represented by smooth functions over a coarse computational grid, and a subgrid (i.e., unresolved) component. The flow is mainly simulated using supergrid structures while the subgrid structures are modelled. The filtered governing equations provide the mathematical interaction between the subgrid and the supergrid structures.

A spatial filtering using filter kernel $G_f$ is defined by the integral

$$\overline{f(x,t)} = \int f(x',t)G_f(x,x')dx'dx'. \quad (2.1)$$

The supergrid component of a flow variable $f$ is determined by the Favre filtering

$$\tilde{f} = \frac{\overline{f}}{\overline{\rho}}. \quad (2.2)$$

By applying the Favre filtering operation to the governing equations (1.1-1.4), the following LES equations are obtained:

$$\frac{\partial \tilde{\rho}}{\partial t} + \frac{\partial \rho \tilde{u}_i}{\partial x_i} = 0, \quad (2.3)$$

$$\frac{\partial \rho \tilde{u}_i}{\partial t} + \frac{\partial}{\partial x_j} \left[ \tilde{\rho} \tilde{u}_i \tilde{u}_j + \tilde{\rho} \delta_{ij} - \tau_{ij} + \tau_{ij}^{sub} \right] = 0, \quad (2.4)$$

$$\frac{\partial \rho \tilde{E}}{\partial t} + \frac{\partial}{\partial x_i} \left[ \left( \tilde{\rho} \tilde{E} + \tilde{p} \right) \tilde{u}_i + \tilde{q}_i - \tilde{u}_j \tilde{r}_{ji} + H_i^{sub} + \sigma_i^{sub} \right] = 0, \quad (2.5)$$

$$\frac{\partial \rho \tilde{Y}_m}{\partial t} + \frac{\partial}{\partial x_i} \left[ \tilde{\rho} \tilde{Y}_m \tilde{u}_i - \tilde{p} D_i^{m} \frac{\partial \tilde{Y}_m}{\partial x_i} + \Phi_{i,m}^{sub} + \theta_{i,m}^{sub} \right] = \tilde{w}_m. \quad (2.6)$$

These equations describe the evolution of the filtered density and the Favre filtered velocity, energy and mass fractions. The filtered stress tensor $\tau_{ij}$ and the filtered heat flux $\tilde{q}_i$ are defined in terms of the Favre filtered velocity. The $\text{sub}$ superscript subgrid terms are defined as follows: the subgrid stress tensor

$$\tau_{ij}^{sub} = \tilde{\rho} \left[ \tilde{u}_i \tilde{u}_j - \tilde{u}_i \tilde{u}_j \right], \quad (2.7)$$
the subgrid heat flux
\[ H_{i}^{\text{sub}} = \bar{\rho} \left[ \bar{E} u_i - \bar{E} \bar{u}_i \right] + \left[ \bar{\mu} u_i - \bar{\mu} \bar{u}_i \right], \tag{2.8} \]
the subgrid viscous work
\[ \sigma_i^{\text{sub}} = - \left[ u_j \bar{r}_{ji} - \bar{u}_j \bar{r}_{ji} \right], \tag{2.9} \]
the subgrid species mass flux
\[ \Phi_{i,m}^{\text{sub}} = \bar{\rho} \left[ \bar{u}_i \bar{Y}_m - \bar{u}_i \bar{Y}_m \right], \tag{2.10} \]
the subgrid diffusive mass flux
\[ \theta_{i,m}^{\text{sub}} = \bar{\rho} \left[ \bar{V}_{i,m} \bar{Y}_m - \bar{V}_{i,m} \bar{Y}_m \right], \tag{2.11} \]
and the filtered mean reaction rate \( \bar{w}_m \). These terms need to be closed by modelling.
Chapter 3

The Turbulence Model

Modelling the turbulence in LES is equivalent to finding a closure to the subgrid stress tensor $\tau_{ij}^{\text{sub}}$. The model should produce most of the effects expected from the theory of turbulence.

The theory of turbulence is based on the ideas of Kolmogorov [25], [26]. He deduced in 1941, using a similarity hypothesis and dimensional analysis, an energy spectrum for isotropic turbulence that can be observed in experiments. The fact is that when there exists a range of scales (the inertial range) in which effects of viscosity, boundary conditions, and large-scale structures are not important, dimensional analysis leads to the universal power-law spectrum

$$E(k) = a_k \epsilon^{2/3} k^{-5/3}. \quad (3.1)$$

Here, $a_k$ is the Kolmogorov constant, $k$ is the wavenumber magnitude and $\epsilon$ is the dissipation rate of kinetic energy caused by the kinematic viscosity $\nu = \mu/\rho$ (see equation (1.6)). Equation (3.1) is well supported by a large body of experimental data (e.g. see [42] and [45]). Recent experiments using high resolution techniques for particle tracing [55] have shown that Kolmogorov’s theory is basically correct, giving a new breath to the ideas of the Russian mathematician.
In the present work, the subgrid stress tensor is modelled using a dynamic procedure which uses the subgrid kinetic energy per unit mass $k^{sub}$ to define a characteristic velocity scale and the local grid size $\Delta$ as the characteristic length scale (see [60] and [36]). Drawing an analogy between the subgrid stresses in the filtered Navier-Stokes equations and the viscous stresses in unfiltered equations, the subgrid stress tensor $\tau^{sub}$ is modelled using a resolved rate of strain

$$\tau_{ij} = \frac{1}{2} \left( \frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right)$$  \hspace{1cm} (3.2)

and the subgrid kinetic energy per unit mass

$$k^{sub} = \frac{1}{2} \left( \tilde{u}^2 - \bar{u}^2 \right)$$  \hspace{1cm} (3.3)

as follows:

$$\tau_{ij}^{sub} = -2\rho v_t \left( \tilde{S}_{ij} - \frac{1}{3} \tilde{S}_{kk} \delta_{ij} \right) + \frac{2}{3} \tilde{p} k^{sub} \delta_{ij},$$  \hspace{1cm} (3.4)

where $\nu_t$ is the eddy viscosity.

An expression for the eddy viscosity can be deduced using the theory of Kolmogorov. Spectral closure theories by Kraichnan [24] and Lesieur [30] predict that the eddy viscosity at large scales is given by

$$\nu_t = 0.261 \sqrt{\frac{E(k_c)}{k_c}}$$  \hspace{1cm} (3.5)

where $k_c \approx 1/\Delta$ is the cut-off wave number of the LES. Menon and Chakravarthy [11] have used the following form of the Kolmogorov energy spectrum

$$E(k) = a_k \varepsilon^{3/2} k^{-5/3} \exp\left[-\frac{3}{2} a_k (k\eta)^{2/3}\right]$$  \hspace{1cm} (3.6)

valid in the inertial and dissipation range to obtain the following expression of the eddy viscosity:

$$\nu_t = C_\nu \Delta \sqrt{k^{sub}} \exp\left[-\frac{3}{2} a_k (k\eta)^{2/3}\right].$$  \hspace{1cm} (3.7)
Here, \( a_k = 1.4 \) is the Kolmogorov constant, \( \eta = \left( \nu^3 / \epsilon \right)^{1/4} \) is the Kolmogorov length scale, and \( \epsilon \) is the total dissipation rate. If \( k_c \) lies in the inertial range, the contribution of the exponential term is negligible and an approximate expression for the eddy viscosity is given by

\[
\nu_t = C_\nu \Delta \sqrt{k_{sub}}. 
\]

The subgrid kinetic energy can be obtained by solving the transport equation

\[
\frac{\partial \rho k_{sub}}{\partial t} + \frac{\partial}{\partial x_i} (\rho \tilde{u}_i k_{sub}) = P_{sub} - D_{sub} + \frac{\partial}{\partial x_i} \left( \frac{\nu_t}{Pr_t} \frac{\partial k_{sub}}{\partial x_i} \right) 
\]

where \( P_{sub} \) and \( D_{sub} \) are the production and dissipation of the subgrid kinetic energy, respectively, and \( Pr_t \) is the turbulent Prandtl number [12].

The subgrid kinetic energy production term is

\[
P_{sub} = -\tau_{ij}^{sub} \left( \frac{\partial \tilde{u}_i}{\partial x_j} \right) 
\]

and the subgrid kinetic energy dissipation term

\[
D_{sub} = C_\epsilon \tilde{\rho} \left( k_{sub} \right)^{3/2} \frac{\Delta}{\nu_t} 
\]

The behavior of this dynamic model for the turbulence has been investigated and compared to other subgrid models in the work of Menon et al., [36]. They conclude that this dynamic subgrid kinetic energy model (or one-equation model) consistently showed a higher correlation for a range of Reynolds numbers when compared to the dynamic eddy viscosity model. They advise the values for the constants \( C_\nu = 0.067 \) and \( C_\epsilon = 0.916 \), for three-dimensional homogeneous turbulence.

The rest of the subgrid terms also need modeling. The closure of the subgrid heat flux is achieved using the gradient-diffusion model

\[
H_{i}^{sub} = -\tilde{\rho} \frac{\nu_t}{Pr_t} \frac{\partial \tilde{h}}{\partial x_i} 
\]
where $\tilde{h}$ is the filtered total mixture enthalpy per unit mass (see (1.12) and (1.13))

$$\tilde{h} = \sum_{m=1}^{N} c_{p,m} Y_m T.$$  \hfill (3.13)

Analogous to the closure for the subgrid heat flux, the closure of the subgrid species mass flux is achieved using the gradient-diffusion model

$$\Phi_{i,m}^{sub} = -\rho \nu_t \frac{\partial \tilde{Y}_m}{\partial x_i},$$  \hfill (3.14)

where $Sc_t$ is the turbulent Schmidt number

$$Sc_t = Pr_t Le_m$$  \hfill (3.15)

and $Le_m$ is the species Lewis number [27].

In this work, the subgrid viscous work $\sigma_i^{sub}$ and the diffusive mass flux $\theta_i^{sub}$ are neglected.
Chapter 4

The Combustion Model

One of the most common ways to classify combustion is due to how the fuel and oxidizer are supplied into the reaction zone. In this way, combustion can be classified into premixed, nonpremixed and partially premixed.

Premixed combustion and partially premixed combustion means that the fuel and oxidizer are mixed completely or partially before entering the combustion chamber. Nonpremixed combustion means that the fuel and the oxidizer are separated before entering the combustion zone.

Nonpremixed combustion is of practical importance in liquid fueled gas turbines, diesel engines, furnaces and fires. In these problems, combustion takes place simultaneously with the turbulent mixing process and therefore, the rate of chemical reactions is controlled essentially by molecular mixing of fuel and oxidizer, at the dissipative turbulent scales. These scales are the smallest of the subgrid level and therefore the chemical reactions must be entirely modelled.

The methane/air flame that we want to simulate, named Sandia flame D, is of the partially premixed type. This flame is part of the 1997 Sandia data set of scalar measurements in piloted methane-air jet
flames [2] where four piloted flames, named C, D, E and F, with increasing velocity of the main jet and pilot, and increasing probability of localized extinctions are studied experimentally. In these flames the mixing rates are high enough that they burn as diffusion flames, with a single reaction zone near the stoichiometric mixture fraction. In these flames the jet fluid consists of a mixture of three parts air and one part methane by volume, surrounded by a pilot jet of hot air. Flame D is of particular importance because the velocity of the jet is in the limit of lift-off, when the flame detaches from the burner lip. The name Sandia is a reference to the Sandia National Laboratories in Livermore, California, U.S.A.

The main difficulty associated with the model of the chemical reactions is the large number of species and therefore the large number of transport equations (see (1.4)) that must be solved simultaneously in realistic combustion processes.

In this work, the large number of species equations are reduced using the transient laminar flamelet model (TLFM) developed by the group of J. Gas and S. Baykal of the Laboratory of Thermodynamics in Emerging Technologies (LTNT) of the ETH, Zurich. This particular model was designed by Ferreira [20] and it has been implemented together with low order Large Eddy Simulations (LES) by Baykal [3]. We refer the reader to their work for a complete discussion of the TLFM, including its derivation and comparison to other alternative combustion models. Our work aims to couple their combustion model to a high order LES code.

Flamelet models assume that a turbulent flame is an ensemble of thin, locally laminar flamelets. They are applicable when the local flamelet thickness is smaller than the turbulent eddies, acoustic perturbations and radii of flame curvature. In such cases, the flamelets are strained and stretched by the turbulent flow field, but maintain their inner structure. The TLFM of Ferreira [20] is one of the earliest unsteady flamelet models. It is applicable for the nonpremixed flames with local extinction and aims to add the partial premixing effects and reignition phenomenon to the classical steady flamelet models.
First of all, flamelet models assume that the chemical state and thereby the species mass fractions $Y_m$, can be related to a conserved scalar, namely the mixture fraction $Z$. The mixture fraction is defined as

$$Z = \frac{sY_F - Y_O + Y^0_O}{sY^0_F + Y^0_O},$$

(4.1)

where $Y_F$ and $Y_O$ are the species mass fractions of the fuel and the oxidizer, respectively. $Y^0_F$ and $Y^0_O$ are the fuel and oxidizer mass fractions in pure fuel and oxidizer streams, respectively. And the mass stoichiometric ratio $s$ is defined as

$$s = \frac{\nu_O W_O}{\nu_F W_F},$$

(4.2)

where $W_F$ and $W_O$ are the molecular weights of fuel and oxidizer species, respectively, and $\nu_F$ and $\nu_O$ are the molar stoichiometric coefficients of fuel and oxidizer, respectively.

$Z$ is equal to one in the pure fuel stream and zero in the pure oxidizer stream. The $N$ equations for the species mass fractions (1.4) are compressed into one transport equation for the mixture fraction $Z$. The iso-surface of the stoichiometric mixture fraction defines the location of the flame surface.

The LES filter is used to obtain the following transport equation for the filtered mixture fraction:

$$\frac{\partial \tilde{Z}}{\partial t} + \frac{\partial}{\partial x_i} \left( \rho \tilde{u}_i \tilde{Z} \right) = \frac{\partial}{\partial x_i} \left[ \bar{\rho} D_Z \frac{\partial \tilde{Z}}{\partial x_i} - \zeta^{sub}_i \right],$$

(4.3)

where $D_Z$ is the molecular diffusion coefficient of the mixture fraction and the $\zeta^{sub}_i$ is the subgrid mixture fraction mass flux

$$\zeta^{sub}_i = \bar{\rho} \left[ \tilde{u}_i \tilde{Z} - \tilde{u}_i \tilde{Z} \right].$$

(4.4)

The model for the subgrid mixture fraction mass flux is the extension of the gradient-diffusion model for the subgrid species mass flux $\Phi^{sub}_{i,m}$. 
Assuming a constant or averaged Schmidt number (3.15), the model is given by

$$\xi_i^{\text{sub}} = -\frac{\nu}{Sc_i} \frac{\partial \tilde{Z}}{\partial x_i}. \quad (4.5)$$

The mixture fraction does not contain any information about the chemical reactions, for example, if the mixture is ignited or not, and because of its scalar nature, it cannot provide information about chemical variations in the directions perpendicular to its gradients. In a chemistry model solely based on the mixture fraction, fast chemistry (also known as the Burke-Schumann limit [8]) and steady flamelet state are necessary assumptions to associate the mixture fraction with a chemical state.

One additional transported scalar is introduced to improve the model of the chemical reactions in the TLFM, describing non-equilibrium states using finite rate chemical reactions, allowing for ignitions and extinctions. This variable is called the reaction progress variable $c$, as the one popularly used in premixed combustion computations. It is independent from the mixture fraction and it should not be conserved. Its change in time is given by the transport equation

$$\frac{\partial \rho \tilde{c}}{\partial t} + \frac{\partial}{\partial x_i} (\rho \tilde{u}_i \tilde{c}) = \frac{\partial}{\partial x_i} \left[ \rho D_c \frac{\partial \tilde{c}}{\partial x_i} \right] + S_c. \quad (4.6)$$

The source term $S_c$ is given by

$$S_c = \begin{cases} 
(c_f - \tilde{c})/\tau & \text{if } c_f - \tilde{c} > 0, \\
0 & \text{if } c_f - \tilde{c} \leq 0, 
\end{cases} \quad (4.7)$$

so as to simulate the competition between extinction and reignition states. It has the function of making the value of $\tilde{c}$ equal to the local state of reaction progress

$$c_f = 1 - \frac{Y_F}{Y_{F,u}}. \quad (4.8)$$

Here, $Y_F$ is the local fuel mass fraction, to be computed from the transient flamelet model, and $Y_{F,u}$ is the maximum unburned fuel mass
fraction at the inlet of the jet. The time $\tau$ is chosen as the residence
time of a fluid particle (with velocity vector $\mathbf{u}$) in a computational
grid cell (of volume $V$)

$$\tau = \frac{V^{1/3}}{||\mathbf{u}||}. \quad (4.9)$$

$D_c$ is the diffusion coefficient for the reaction progress variable, the
sum of the molecular and turbulent diffusivities.

Two more scalars are used to define a chemical state in the TLFM:
the residence time of a fluid particle $\tau$ and the scalar dissipation rate
of the mixture fraction $\chi$. The residence time of a fluid particle $\tau$
defined in (4.9), provides a measure of the evolution of the flamelet
or the response to the unsteadiness imposed by the turbulence. The
scalar dissipation rate of the mixture fraction

$$\chi = 2D_\chi ||\nabla \tilde{Z}||^2 \quad (4.10)$$

is a measure of the strain on the flame and equivalent to the dissi-
pation rate of scalars inside a turbulent flow field. Therefore, it
provides a measure of the influence of the turbulent flow field on
the inner reaction zone and the dependence of the chemical state on
the flame structure. The diffusion coefficient for the scalar dissipa-
tion rate $D_\chi$ is equivalent to the diffusion coefficient for the reaction
progress variable.

A flamelet library with tabulated values, together with multilinear
interpolations, are used to compute in every node the filtered, or
averaged, individual species mass fractions $\bar{Y}_i$ and the temperature
$T$, from the local values of the filtered mixture fraction $\tilde{Z}$, its scalar
dissipation rate $\chi$, the filtered reaction progress variable $\tilde{c}$ and the
local turbulent residence time $\tau$. Following the idea of Baykal [3] we
have loaded the tables into memory at the start of our computations
reducing the computational time considerably. We have also included
the tables for the values of $\sum_m Y_m/W_m$ necessary to compute the
pressure in (1.9), the viscosity of the mixture at normal conditions
$\sum_m \mu_m Y_m$ and the specific heat capacities of the mixture $\sum_m c_{p,m} Y_m$
and $\sum_m c_{\nu,m} Y_m$. The range of the input variables in the flamelet
library is presented in the Table 4.1.
Table 4.1: Range of the input variables in the flamelet library.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Range</th>
<th>Discrete values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixture Fraction (Z)</td>
<td>[0.0, 0.1563]</td>
<td>70</td>
</tr>
<tr>
<td>Progress Variable (c)</td>
<td>[0.0, 1.0]</td>
<td>9</td>
</tr>
<tr>
<td>Scalar Dissipation Rate (χ)</td>
<td>[1.0 × 10⁻², 30.0]</td>
<td>6</td>
</tr>
<tr>
<td>Residence Time (τ)</td>
<td>[1.0 × 10⁻⁵, 0.01]</td>
<td>4</td>
</tr>
</tbody>
</table>

The flamelet library provided to us has been created by the FLATRA code [20], where the flamelet equations (not shown here) are solved for a counterflow diffusion flame, which consists of opposed, axisymmetric fuel and oxidizer jets. The flamelet equations are discretized in time and space, and solved using VODPK finite differences.

The mass fractions tabulated in the flamelet library are given for the following molecular species: methane (CH₄), oxygen (O₂), carbon dioxide (CO₂), water (H₂O), carbon monoxide (CO), hydrogen (H₂) and nitrogen (N₂).
The Lagrangian Form of the Governing Equations

The Lagrangian form of a transport equation is obtained by changing to a particle-tracking frame of reference and substituting the partial time derivative with the material, or total, time derivative. The total time derivative is given by the chain rule

\[ \frac{D}{Dt} = \frac{\partial}{\partial t} + \frac{dx_i}{dt} \frac{\partial}{\partial x_i}, \]

where Einstein’s summation convention \( a_i b_i = \sum_i a_i b_i \) is used for 3-vectors \( (i = 1:3) \). Notice that \( dx_i/dt \) is the velocity vector component \( u_i \) and therefore

\[ \frac{D}{Dt} = \frac{\partial}{\partial t} + u_i \frac{\partial}{\partial x_i}. \] (5.1)

The Lagrangian form of the equations for the transport of the filtered density and the Favre filtered velocities, energy, subgrid kinetic energy and mixture fraction are:

the conservation of mass

\[ \frac{D\rho}{Dt} = -\rho \frac{\partial u_i}{\partial x_i}, \] (5.2)
Newton’s second law

\[ \rho \frac{D\tilde{u}_i}{Dt} = \frac{\partial}{\partial x_j} \left[ -\tilde{p}\delta_{ij} + \tilde{r}_{ij} - \tau_{ij}^{sub} \right], \quad (5.3) \]

the first law of thermodynamics (conservation of energy)

\[ \rho \frac{D\tilde{e}}{Dt} = -\tilde{p} \frac{\partial \tilde{u}_i}{\partial x_i} - \frac{\partial}{\partial x_i} \left[ q_i - \tilde{w}_j^i j_i + H_i^{sub} + \sigma_i^{sub} \right], \quad (5.4) \]

the subgrid kinetic energy transport, from (3.3),

\[ \rho \frac{D\tilde{k}^{sub}}{Dt} = p_{sub} - \tilde{D}_{sub} + \frac{\partial}{\partial x_i} \left[ \rho \frac{\nu_i}{Pr_i} \frac{\partial \tilde{k}^{sub}}{\partial x_i} \right], \quad (5.5) \]

the mixture fraction transport

\[ \rho \frac{D\tilde{Z}}{Dt} = \frac{\partial}{\partial x_i} \left[ \tilde{p} D_Z \frac{\partial \tilde{Z}}{\partial x_i} - \xi_i^{sub} \right], \quad (5.6) \]

and the reaction progress variable transport

\[ \rho \frac{D\tilde{c}}{Dt} = \frac{\partial}{\partial x_i} \left[ \tilde{p} D_c \frac{\partial \tilde{c}}{\partial x_i} \right] + S_c. \quad (5.7) \]
Part III

Numerical Method
Chapter 6

The Forward-trajectory Semi-Lagrangian Method

We want to develop a high-order, non-diffusive numerical method for the solution of a system of non-linear transport equations written in Lagrangian form

$$\frac{D\alpha}{Dt} = S(\alpha, \mathbf{x}, t). \quad (6.1)$$

Here, $\mathbf{x}$ is the position vector with components $x_i$ for $i = 1:3$; $t$ is the time; $\alpha = \alpha(\mathbf{x}, t)$ is a vector of transported variables (frequently the density, velocity and energy); $S(\alpha, \mathbf{x}, t)$ is the right hand side (RHS) vector of the transport equations; and $D/Dt$ is the material or total time derivative defined along a trajectory given by the solution of the ODEs that define the velocity vector components

$$u_i = \frac{dx_i}{dt}. \quad (6.2)$$

The system of equations (6.1) can be written, using (5.1), as the conservation law

$$\frac{\partial \alpha_i}{\partial t} + \frac{\partial}{\partial x_i} (u_i \alpha_i) = S_i(\alpha, \mathbf{x}, t) + \alpha_i \frac{\partial u_i}{\partial x_i} \quad (6.3)$$
We want to solve the system of equations (6.1) over the parallelepiped $\Omega$ such that

$$\Omega = [x_1(1), x_1(N_I)] \times [x_2(1), x_2(N_J)] \times [x_3(1), x_3(N_K)] \in \mathbb{R}^3.$$ 

We discretize $\Omega$ using an equispaced Cartesian mesh given by the nodes $x(I, J, K) = [x_1(I), x_2(J), x_3(K)]^T$, such that the number of nodes $I = 1 : N_I$, $J = 1 : N_J$ and $K = 1 : N_K$, the corner node $x(1, 1, 1)$, and the mesh spacings $\Delta x_1, \Delta x_2, \text{and} \Delta x_3$ are given.

A wide collection of conservative, finite volume schemes have been developed to solve a system of conservation laws over Cartesian domains and we refer the reader to the book of LeVeque [32] for an overview of this topic.

Nevertheless, research produced in our group by Michael Fey [21] has proven that the one-dimensional splitting of the finite volume schemes produces a large amount of numerical diffusion when the direction of transport is not aligned with the coordinate lines. The problem is that the divergence in the conservation laws (6.3) is approximated using Gauss’ theorem for each finite volume and the resulting fluxes
do not take into consideration the fluxes into corner cells as sketched in part (a) of Figure 6.1.

The second-order, multi-dimensional solution proposed by Fey [21] is called the Method of Transport (MoT) and it is sketched in part (b) of Figure 6.1. In this method, an individual advection equation is found for each corner of the finite volume and each flux is computed using a linear reconstruction of the faces of the deformed finite volume. He found significant improvement in his numerical solutions and lowered the direction dependence of the numerical diffusion. Further reduction of the numerical diffusion requires higher-order schemes.

We noticed that implementing the MoT for higher-orders of convergence is a complicated task. Therefore, we decided to keep the basic idea of the MoT but implement it using a Lagrangian representation of the equations. This allows us to solve transport equations for the corners of the cells using high-order integrators in time. Although we could compute the flux to the neighboring cells, we decided instead, for simplicity, to reconstruct the transported variables using multi-dimensional interpolations. Given this, we need a conservative, non-diffusive and inexpensive interpolation scheme. We were successful in this quest and refer the reader to Chapter 7 for a complete discussion of the interpolations.

6.1 A High-order FSL Method

The proposed high-order extension of the multi-dimensional transport achieved in the MoT, independently of the interpolation scheme and applied to the Lagrangian representation of conservation laws, is known in the literature as the forward-trajectory, semi-Lagrangian method (FSL).

In general, the FSL consists of an explicit multistage ODE numerical integration in time and multi-dimensional spatial interpolations. The interpolations are used during intermediate stages and at the end of the time step. They are used to compute the values of the variables
on the original mesh and to compute the RHS on the new positions of the nodes.

We illustrate the basic concept behind the FSL for the simplest case, using a first order explicit (forward) Euler integration in time and multi-linear interpolations over a bidimensional Cartesian mesh with equidistant nodes. Although this method in particular is highly diffusive, it is useful to introduce the idea and then take it further to higher order integrations and interpolations.

**Integration of trajectories and variables:** Figure 6.2 shows a bidimensional (2D) forward-Euler trajectory of a grid cell. The trajectory is obtained from (6.2) by

\[ x^{n+1}(I, J) = x^0(I, J) + \Delta t \, u^n(x^0(I, J)), \quad (6.4) \]

where \( x^0(I, J) \) are the positions of the nodes in the Cartesian mesh, \( x^{n+1}(I, J) \) are the positions of the nodes at time \( t^{n+1} = (n+1)\Delta t \) and \( \Delta t \) is the discrete time interval. The transport variables are updated in a similar way by

\[ \alpha^{n+1}(I, J) = \alpha^n(I, J) + \Delta t \, S^n(x^0(I, J)). \quad (6.5) \]

**Interpolation:** The set of arrival points form a flow-distorted Lagrangian mesh (see Figure 6.2), where two-dimensional interpolations must be used to reconstruct the flow variables \( \alpha^{n+1}(x^{n+1}(I, J)) \) over the entire domain \( \Omega \) (in this case 2D) and evaluate them on the positions of the Cartesian mesh points to get \( \alpha^{n+1}(x^0(I, J)) \). The multidimensional interpolation of the transport variables to the Cartesian mesh is necessary to avoid high distortions of the moving mesh and therefore the need for very small time steps \( \Delta t \). The interpolation over a distorted mesh is the topic of Section 7.3.1. For this simple case we use multilinear interpolations given by the affine area weights of the four corner points of the distorted cell relative to the corresponding Cartesian mesh point inside the cell. If the interpolation operator is denoted by \( R \) then \( \alpha^{n+1}(x^0) = (R(\alpha^{n+1}(x^{n+1}))(x^0). \)

Notice that because the nodes move, they could move out of \( \Omega \). Every time step the Lagrangian nodes form a new computational domain
6.1 A High-order FSL Method

Figure 6.2: The forward-trajectory semi-Lagrangian (FSL) scheme using a bilinear deformed mesh and forward-Euler trajectory integration.

\( \Omega_{n+1} \) that is not necessarily contained in \( \Omega \). To avoid this, we use a few layers of extra nodes (known as ghost cells) in every direction to get the extended computational domain \( \Omega_0 \), such that \( \Omega_{n+1} \subset \Omega_0 \) always. The ghost cells are used to impose the corresponding boundary conditions and assure the necessary number of neighboring points for the interpolations.

Now, we explain the FSL method in the case of explicit Runge-Kutta numerical integration in general.

**Integration of trajectories:** The explicit Runge-Kutta method for
the ODE $x' = u(x(t), t)$ is given by

$$x^{n+1} = x^n + \Delta t \sum_{i=1}^{s} b_i k_i,$$

\[ k_1 = u(x^n, t^n), \]
\[ k_2 = u(x^n + a_{21} k_1 \Delta t, t^n + c_2 \Delta t), \]
\[ k_3 = u(x^n + a_{31} k_1 \Delta t + a_{32} k_2 \Delta t, t^n + c_3 \Delta t), \]
\[ \vdots \]
\[ k_s = u(x^n + a_{s1} k_1 \Delta t + \ldots + a_{s,s-1} k_{s-1} \Delta t, t^n + c_s \Delta t), \]

where $\Delta t$ is the discrete time step, $t^n$ is the time at step $n$ and $a_{ij}$, $b_i$ and $c_i$ are constants from the numerical method. The velocity vector $u$ is known at the Lagrangian nodes at every stage of the RK method because either is given or it is computed inside the vector of variables $\alpha$.

**Integration of transport variables:** The evolution of the vector of transport variables $\alpha = \alpha(x(t), t)$ is given over the Lagrangian nodes that, while moving, form a distorted Lagrangian mesh where the computation of the RHS $S$ (containing differential operators) can become in general quite complicated or of low numerical accuracy. We have decided to compute $S$ always on the Cartesian mesh and use interpolations to obtain the value of $S$ at the moving nodes. At the same time, for the computation of $S$ we need the values of the vector $\alpha$ at the Cartesian mesh, for which we use interpolations from the Lagrangian nodes. Finally, the vector $\alpha$ is placed back into the original mesh at the end of the time step once again using an interpolation. This final interpolation is the most important and sometimes is known as remeshing $R$.

If the interpolation operator from the Eulerian mesh to the Lagrangian nodes is denoted by $I_{el}$, the interpolation from the Lagrangian nodes to the Eulerian mesh denoted by $I_{le}$, and the remeshing denoted by $R$, then the FSL method for $\alpha = \alpha(x(t), t)$ is given in every node by

$$\alpha^{n+1}(x^0) = \left( R(\alpha^n + \Delta t \sum_{i=1}^{s} b_i k_i) \right)(x^0),$$
6.1 A High-order FSL Method

\[ l_1 = S(\alpha^n, t^n), \]
\[ l_2 = I_{el}(S(I_{le}(\alpha^n + a_{21} l_1 \Delta t), t^n + c_2 \Delta t)), \]
\[ l_3 = I_{el}(S(I_{le}(\alpha^n + a_{31} l_1 \Delta t + a_{32} l_2 \Delta t), t^n + c_3 \Delta t)), \]
\[ \vdots \]
\[ l_s = I_{el}(S(I_{le}(\alpha^n + a_{s1} l_1 \Delta t + \ldots + a_{s,s-1} l_{s-1} \Delta t), t^n + c_s \Delta t)). \]

We have abused the notation to keep it simple, omitting the evaluation of the interpolation operators. The notation for the interpolation from the Lagrangian nodes to the Eulerian mesh \( I_{le} \) should include the evaluation of the function at \( x^0 \). While the interpolation operator from the Eulerian mesh to the Lagrangian nodes \( I_{el} \) should include the evaluation of the function at \( x^{n+j/s} = x^n + \Delta t \sum_{i=1}^{j-1} a_{j+1,i} k_i \) such that \( l_j = (I_{el} S)(x^{n+(j-1)/s}) \).

Our particular high-order FSL for the case of a three-dimensional Cartesian mesh is given by the following algorithm:

I. Initial State. The transported variables are given on the fixed mesh

\[ \alpha^n(I, J, K) = \alpha(x_1(I), x_2(J), x_3(K), t^n). \]

Here, the RHS \( S^n(I, J, K) = S(\alpha, x, t^n) \) is computed over the mesh from the given variables \( \alpha^n(I, J, K) \). If \( S^n \) contains derivatives or integrals, those should be computed with a numerical method corresponding to the chosen interpolation scheme.

II. Integration in Time. The system of transport ODEs (6.1) and the equation for the trajectories (6.2) are solved using a low-storage, third-order Runge-Kutta (RK3) scheme that is described in detail later on (Algorithm 6.1.1).

During the first stage we update the variables \( \alpha \) and the positions of the nodes \( x \), using the RHS \( S^n \) and the velocity vector \( u \) (given at the Cartesian mesh), respectively. The positions of the computational nodes at the mesh \( x(I, J, K) = [x_1(I), x_2(J), x_3(K)]^T \) change and define a new distorted mesh \( x^*(I, J, K, t^*) \) where the intermediate value of the transported variables \( \alpha^*(I, J, K, t^*) \) is known.
IIa. **Interpolation from new nodes to the fixed mesh.** Note that the computation of the RHS over a non-orthogonal mesh is complicated and, for large deformations, of low accuracy. Therefore the RHS $S^*$ is computed over the Cartesian mesh at every intermediate stage of the integration in time. In these cases, interpolations of the required accuracy $I_{el}$ are used to obtain the values of the variables $\alpha^*$ at the Cartesian mesh nodes and from those values be able to compute the RHS $S^*$. See Chapter 7 for the description of the multidimensional interpolations that we use.

IIb. **Interpolation of the RHS.** Once the RHS $S^*$ has been computed over the Cartesian mesh, another interpolation $I_{el}$ is necessary to bring the RHS from the Cartesian mesh to the Lagrangian positions $x^*(I, J, K, t^*)$ where it must be evaluated to complete the integration in time at every intermediate stage.

III. **Remeshing.** At the end of every time step, the value of the transported variables $\alpha^{n+1}(I, J, K, t^{n+1})$ is known at the deformed mesh, given by the updated positions of the nodes $x^{n+1}(I, J, K, t^{n+1})$ and another interpolation, the remeshing $R$, is necessary to reposition the transported variables onto the Cartesian mesh and be ready for the next integration time step.

As we mentioned, we use a Cartesian mesh with the nodes at the corners of the elements. But any other coordinate system with Cartesian topology could be used. For example, we use spherical coordinates to simulate a turbulent and reactive jet in Chapter 9.

We give now the details of the integration in time. We chose to do it using an explicit, low-storage, third-order Runge-Kutta (RK3) scheme [58] that, when applied to the system of ODEs (6.1)-(6.2)

\[
\frac{d}{dt}x(t) = u(x, t),
\]

\[
\frac{D}{Dt} \alpha(x(t), t) = S(\alpha, x(t), t),
\]

is given by the following pseudo-code.
Algorithm 6.1.1. (RK3-FSL):

\[ x_i^* = x_i^n + \frac{\Delta t}{3} u_i(x^n, t) \quad \text{for } i = 1:3, \]

\[ \alpha^* = \alpha^n + \frac{\Delta t}{3} S(\alpha^n, x^n, t), \]

\[ u_i^* = -\frac{5}{9} u_i(x^n, t) + u_i(x^*, t + \Delta t/3) \quad \text{for } i = 1:3, \]

\[ S^* = -\frac{5}{9} S(\alpha^n, x^n, t) + I_{el} S(I_{el} \alpha^*, x^n, t + \Delta t/3), \]

\[ x_i^* = x_i^* + \frac{15\Delta t}{16} u_i^* \quad \text{for } i = 1:3, \]

\[ \alpha^* = \alpha^* + \frac{15\Delta t}{16} S^*, \]

\[ x_i^{n+1} = x_i^* + \frac{8\Delta t}{15} \left( -\frac{153}{128} u_i^* + u_i(x^*, t + 3\Delta t/4) \right) \quad \text{for } i = 1:3, \]

\[ \alpha^{n+1} = \mathcal{R}(\alpha^* + \frac{8\Delta t}{15} \left( -\frac{153}{128} S^* + I_{el} S(I_{el} \alpha^*, x^n, t + 3\Delta t/4) \right)), \]

where the variables depend on the nodes of the mesh \( x_i = x_i(I, J, K) \), \( u_i = u_i(I, J, K) \) (for \( i = 1:3 \)) and \( \alpha = \alpha(I, J, K) \). The superscript \( * \) is given to an intermediate variable that is used to reduce storage, the superscript \( n \) is given to initial values during the time step and \( n + 1 \) is given to the updated values. At the end of the time step \( \alpha^{n+1} \) is given on the original mesh \( x^n = x^0 = x(t^0) \).

The interpolations at intermediate stages of the RK3, from the Eulerian to the Lagrangian nodes \( I_{el} \), and from the Lagrangian to the Eulerian nodes \( I_{le} \), are done with a tensor product of cubic Z-splines (see (7.43) in Section 7.2). The final interpolation (or remeshing) \( \mathcal{R} \) is done with a tensor product of quintic Z-splines (see Example 7.2.1 of Section 7.2). We have observed that this combination of interpolations provides the best balance of accuracy and computational time. It shows that high accuracy during the remeshing is the crucial operation to obtain overall high accuracy.

The distorted mesh at intermediate stages of the time integration is represented using multi-dimensional linear Z-splines (7.42) and cubic
Z-splines (7.43). The details of the multi-dimensional interpolations are given in Section 7.3 and interpolations from a deformed Cartesian mesh are described in Section 7.3.1. Except for a description of the Z-splines in Chapter 7, these are all the basic elements of our algorithm.

It is important to remark that the Z-splines are conservative interpolations (see Proposition 7.2.1). This is a desirable property, important to achieve conservation of the transported variables like mass and energy in the FSL numerical method. Conservation of higher moments of the transported variables during the interpolations also eliminates spurious forces and torques. The conservation of moments of the Z-spline-FSL scheme is shown in Section 8.1.1, while its numerical convergence is analyzed in Section 8.1.2.

6.2 Review of Semi-Lagrangian Methods

FSL schemes have been previously studied in the literature. Interesting results have been produced by Purser and Leslie [31] that have extended the FSL to a mass-conserving scheme. Also, a globally conservative second and third-order FSL scheme using cascade interpolation has been presented by Nair et al., [38]. In this case, cascade interpolation [49] is an efficient procedure compared to tensor products of cubic spline interpolation (like our Z-spline interpolation) but it is not truly multi-dimensional nor conservative.

There are some numerical methods in the literature that seem to be closely related to the FSL method but in fact there are clear differences. For example, traditional particle methods represent the fluid as a field of particles that carry a non-deforming basis function with them. These methods have problems around convergent or divergent points in the flow where there is a high or low density of particles, respectively. They are therefore often used to simulate incompressible flows. The FSL method does not have this problem because the nodes are not particles but the corners of more complicated volume elements.
In general, semi-Lagrangian (SL) methods are relatively new. Semi-Lagrangian advection combined with semi-implicit time-stepping was introduced in meteorology by Robert [51], [52] nearly two decades ago. Many current operational weather prediction software implementations use SL advection because of the truly multi-dimensionality of the model and have less restrictive time stepping compared to Eu-
erian grid methods. But traditional SL methods solve the trajectory equation backward in time, from the arrival point, by using a second-order accurate implicit midpoint method requiring several iterations [46]. This iterative procedure is computationally expensive [34], [50]. Moreover, backward-trajectory schemes do not easily extend to high-
order accuracy [31].

6.3 Stability and Accuracy of the FSL

Forward semi-Lagrangian schemes do not necessarily have a traditional CFL (Courant, Friedrichs, Lewy) [13] stability restriction in the time step. In general they are known to have less restrictive time step bounds than Eulerian methods [22].

We do not pretend to give an stability estimate for our method applied to the governing equations of turbulent and reactive flows (1.1)-(1.4) but to give some general statements found in the literature about SL stability and accuracy.

A necessary condition for stability of the convective part is that the computational fluid elements do not compress too much during a time step, regardless of the number of computational elements that they’ve moved through during the step. The computational elements cannot be allowed to collapse; neither should their sides come close. The necessary condition to avoid compression is given by the gradients of the velocity field as follows:

$$\Delta t \ll \min_i \left[ \frac{\Delta x_i}{\max_j \Delta j u_i} \right] \approx \frac{1}{\max_i(\|\nabla u_i\|)},$$  \hspace{1cm} (6.6)

where the index counts the spatial directions $i = 1:3$, $\Delta x_i$ is the
discretization in space and $\Delta_j u_i$ is the change in the direction $j$ of the velocity component $i$. In case of constant velocity in time and space, the time step doesn’t have a bound.

The stability region of the FSL scheme is contained in the stability region of the time integration scheme. In the present work, this corresponds to the stability region of the low-storage, third-order Runge-Kutta method [58] applied to the turbulent and reactive governing equations (1.1)-(1.4). We don’t give an estimate for this but it is known that for the conditions of our flows, the fastest sound wave of the mixture of gases (331.4 m/s for air) is going to dominate the stability of these equations like a CFL condition.

An analysis of the convergence of semi-Lagrangian advection schemes has been done by Falcone and Ferretti [18]. One of their main results can be summarized as follows:

**Theorem 6.1.** Assuming that the trajectory approximation method is $O(\Delta t^p)$, that the error in the spatial interpolation procedure is $\varepsilon(\Delta x)$, and that the time interval $[0, T]$ considered is such that $\Delta t = T/N_t$; under mild regularity assumptions

$$\max_{n=1,N} \max_{I,J,K} ||\alpha(x,t^n) - \alpha^n|| \leq C[O(\Delta t^p) + N_t \varepsilon(\Delta x)]. \quad (6.7)$$

Here $\alpha^n = \alpha^n(I,J,K)$ is the numerical approximation to $\alpha(x,t^n)$, $x = (x_1(I), x_2(J), x_3(K))$ is the position 3-vector at mesh indices $(I,J,K)$ and $t^n$ is the time at step $n$.

Therefore, the total error of SL schemes is an accumulation of the spatial interpolation errors and the time integration errors. Thus, a necessary condition for convergence is that the interpolation error should vanish quickly for $N_t$ large. This property is fulfilled by the Z-spline interpolation as we show in the convergence studies of Section 8.1.2.
Chapter 7

High-order Discrete Representation of Functions

7.1 The Z-splines

In the multi-dimensional case we use tensor products of one-dimensional basis functions, so we first want to interpolate the one-dimensional real data

\[(x_i, f_i) \text{ for } i = 1:n, \quad (7.1)\]

with \(x_i \neq x_j\) for \(i \neq j\).

Obviously we can order the data points such that

\[x_1 < x_2 < ... < x_{n-1} < x_n.\]

We will construct a piecewise polynomial function which is \(m\)-times differentiable, and assume that \(2m + 1 \leq n\).

In order to obtain the required smoothness at the interpolation points \(x_i\), we construct derivatives in the following way. Let \(p_{m,j}(x)\) be the unique polynomial interpolation of the data at \(2m + 1\) consecutive
values of $x_i$. Clearly, the degree of $p_{m,j}(x)$ is $2m$. Let $I_j$ be the set of indices of the interpolation points. Depending on the location of $x_j$ we distinguish the following three cases:

I.1) Interpolations in between the boundaries:
$I_j = \{j - m, j - m + 1, \ldots, j, \ldots, j + m - 1, j + m\}$ for $j$ such that $m + 1 \leq j \leq n - m$.

I.2) Interpolations near the left boundary:
$I_j = \{1, 2, \ldots, j, \ldots, 2m + 1\}$ for $j$ such that $1 \leq j \leq m$.

I.3) Interpolations near the right boundary:
$I_j = \{n - 2m, n - 2m + 1, \ldots, j, \ldots, n - 1, n\}$ for $j$ such that $n - m + 1 \leq j \leq n$.

We will need the first $m$ derivatives of this polynomial $p_{m,j}(x)$ at $x_j$. To simplify notation, we use the abbreviation

$$f_{m,j}^p := \left. \frac{d^p}{dx^p} p_{m,j}(x) \right|_{x_j} \text{ for } p = 0 : m, \quad j = 1 : n, \quad (7.2)$$

and clearly,

$$f_{m,j}^0 = f_j. \quad (7.3)$$

Observe that $p_{m,j}(x)$ depends linearly on the interpolation values $f_i$, $i = 1 : n$. Since differentiation is a linear process this gives us the following remark.

**Remark 7.1.1.** The derivatives $f_{m,j}^p$ in equation (7.2) are linear functions of the interpolation values $f_i$, $i = 1 : n$.

**Definition 7.1.2.** The Z-spline $Z_m(x)$ interpolating the data $(x_i, f_i)$, $i = 1 : n$ with $x_1 < x_2 < \ldots < x_n$, is a piecewise polynomial function which satisfies the conditions:

$$Z_m(x) \in C^m([x_1, x_n]), \quad (7.4)$$

$$\left. \frac{d^p}{dx^p} Z_m(x) \right|_{x_j} = f_{m,j}^p \quad \text{for } p = 0 : m, \quad j = 1 : n, \quad (7.5)$$
7.1 The Z-splines

\[ Z_m(x) \in \pi_{2m+1}([x_i, x_{i+1}]) \quad \text{for} \quad i = 1:n - 1, \quad (7.6) \]

where \( \pi_n \) stands for the class of polynomials of degree not exceeding \( n \), over the field \( \mathbb{R} \) of real numbers.

Clearly the conditions (7.3) and (7.5) state that \( Z_m(x) \) interpolates the data i.e.,

\[ Z_m(x_j) = f_{m,j} = f_j \quad \text{for} \quad j = 1:n. \quad (7.7) \]

**Note:** The Z-spline space defined in (7.4)-(7.6) is not the classical spline space where the B-splines are the basis functions [44]. For the same degree of the piecewise polynomial, the B-spline is twice as continuous as the Z-spline.

The linear interpolation case \( m = 0 \) is well known and equivalent to the piecewise linear spline space with the so-called roof- or hat-function (7.42) as the basis function. For equidistant data points, it produces a second order accurate interpolation that conserves up to the first discrete moment of the interpolated data. The following results are valid for \( m > 0 \).

**Proposition 7.1.3.** The interpolating Z-spline \( Z_m(x) \) exists and is unique. Each function value \( Z_m(x) \) is a linear function of the interpolation values \( f_j \) for \( j = 1:n \).

**Proof:** The interpolation polynomials \( p_{m,j}(x) \) are uniquely determined by the data given by (7.1) and I.1, I.2, I.3. Hence the values \( f_{m,j} \) exist and are unique. By the conditions (7.6) and (7.5), we have that

\[ q(x) = Z_m(x)|_{x \in [x_i, x_{i+1}]} \quad (7.8) \]

is a polynomial of degree \( 2m + 1 \) which should satisfy the Hermite interpolation conditions,

\[ \frac{d^p}{dx^p} q(x) \bigg|_{x = x_i} = f_{m,i}^p \quad \text{and} \quad (7.9) \]

\[ \frac{d^p}{dx^p} q(x) \bigg|_{x = x_{i+1}} = f_{m,i+1}^p \quad (7.10) \]
for $p = 0 : m$. From Hermite interpolation theory [9] we know that $q(x)$ exists and is unique. In fact, explicit formulas can be given [9]. In addition $q(x)$ depends linearly on $f^p_{m,i}$ and $f^p_{m,i+1}$. Using Remark 7.1.1 thus says that $Z_m(x)$ depends for any fixed $x$ linearly on $f_1, f_2, ..., f_n$. $\square$

Figure 7.1 shows the basic idea behind the Z-splines for the case of $m = 1$ and the set of points in I.1. The notation $p_{1,j}$ is the central quadratic interpolating polynomial for node $j$. The $Z_1$ spline is a $C^1$ piecewise polynomial equivalent to cubic Bessel interpolation.

**Proposition 7.1.4.** The interpolating Z-spline $Z_m(x)$ reproduces any polynomial $q(x) \in \pi_{2m}$. By this we mean:

If $f_i = q(x_i)$, $i = 1 : n$ and $2m + 1 \leq n$ then the Z-spline $Z_m(x)$ interpolating the data

$$(x_i, f_i) \quad \text{for } i = 1 : n,$$

satisfies

$$Z_m(x) = q(x). \quad (7.11)$$

**Proof:** Clearly $q(x)$ satisfies the conditions (7.6), (7.4) and the inter-
7.1 The Z-splines

Interpolation condition, i.e., (7.3) for \( p = 0 \). It remains to show that

\[
\frac{d^p}{dx^p} q(x) \bigg|_{x=x_j} = f_{m,j}^p, \quad \text{for } p = 1 \cdot m. \tag{7.12}
\]

Observe that \( q(x) \in \pi_{2m} \) and satisfies

\[
q(x_i) = f_i = p_{m,j}(x_i) \quad \text{for } i \in I_j. \tag{7.13}
\]

Since the number of interpolation points is \( |I_j| = 2m + 1 \), the interpolation is unique. Hence, \( q(x) = p_{m,j}(x) \). Therefore (7.12) holds and by Proposition 7.1.3 we have \( Z_m(x) = q(x) \). \( \Box \)

**Remark 7.1.5.** Proposition 7.1.4 is optimal in the sense that polynomials of degree \( 2m + 1 \) are not reproduced.

To see this let \( q(x) \in \pi_{2m+1} \setminus \pi_{2m} \) that is \( q(x) = \alpha x^{2m+1} + \ldots \), with \( \alpha \neq 0 \). We show now that (7.12) does not hold already for \( p = 1 \).

Observe that \( q - p_{m,j} \in \pi_{2m+1} \). From (7.13) follows that \( q - p_{m,j} \) has zeros at \( x_i \) for every \( i \in I_j \). Hence

\[
q(x) - p_{m,j}(x) = \alpha \prod_{i \in I_j} (x - x_i). \tag{7.14}
\]

Already for the first derivative at \( x_j \) we find

\[
\frac{d}{dx} q(x_j) = f_{m,j}^1 + \alpha \prod_{\substack{i \in I_j \atop i \neq j}} (x_j - x_i). \tag{7.15}
\]

Hence we see that even though \( q(x) \) satisfies (7.6) and (7.4), it does not satisfy (7.5) for \( p = 1 \). \( \Box \)

In order to present formulas for the Z-splines, we make use of the linear dependence of \( Z_m(x) \) on \( f_i \). To do this we introduce the Z-spline basis functions.

**Definition 7.1.6.** \( \tilde{Z}_{m,i}(x) \) is called the Z-spline basis function associated with \( x_i \) if it is the Z-spline interpolating the data

\[
(x_j, \delta_{ij}) \quad \text{for } j = 1 : n, \tag{7.16}
\]
where the Kronecker delta is
\[
\delta_{ij} = \begin{cases} 
1 & \text{if } i = j, \\
0 & \text{otherwise.}
end{cases}
\] (7.17)

Using the linear dependence of \( Z_m \) on the interpolation values \( f_i \), we can write the Z-spline interpolating the data given in (7.1) in the following way
\[
Z_m(x) = \sum_{i=1}^{n} f_i \tilde{Z}_{m,i}(x).
\] (7.18)

Therefore the Z-spline basis functions are discrete convolution splines. Note that \( \tilde{Z}_{m,i}(x) \) are by definition independent of the interpolation values \( f_j, j = 1 : n \). Hence we can construct and analyze the Z-spline basis functions independent of the interpolation values.

**Proposition 7.1.7.** \( \tilde{Z}_{m,i}(x) = 0 \) for any \( x \) which is not in the neighborhood of \( x_i \), i.e., \( \tilde{Z}_{m,i} \) has compact support. More precisely, we distinguish three cases:

i) If \( i \) satisfies \( m + 2 \leq i \leq n - m - 1 \) then
\[
\tilde{Z}_{m,i}(x) = 0 \quad \text{for} \quad x \in [x_1, x_n] \setminus (x_{i-m-1}, x_{i+m+1}).
\] (7.19)

ii) If \( i \) satisfies \( 1 \leq i \leq m + 1 \) then
\[
\tilde{Z}_{m,i}(x) = 0 \quad \text{for} \quad x \in [x_1, x_n] \setminus (x_1, x_{i+m+1}).
\] (7.20)

iii) If \( i \) satisfies \( n - m \leq i \leq n \) then
\[
\tilde{Z}_{m,i}(x) = 0 \quad \text{for} \quad x \in [x_1, x_n] \setminus (x_{i-m-1}, x_n).
\] (7.21)

**Proof:** First observe that by (7.16) in Definition 7.1.6 we have
\[
\tilde{Z}_{m,i}(x_j) = 0 \quad \text{for} \quad j \neq i.
\] (7.22)

We distinguish the three cases:
7.1 The Z-splines

i) Let $i$ satisfy $m+2 \leq i \leq n-m-1$. For $j < i-m$ or $j > i+m$ we have that $\tilde{Z}_{m,i}(x_k) = 0$ for $k \in I_j$ given in I.1, I.2 or I.3. Hence $p_{m,j} \equiv 0$ and thus all its derivatives are zero. In particular this is the case for $j = k, k+1$ if one has

$$[x_k, x_{k+1}] \subset [x_1, x_N] \setminus (x_{i-m-1}, x_{i+m+1}).$$

As $\tilde{Z}_{m,i}(x)$ for $x \in [x_k, x_{k+1}]$ is the Hermite interpolation polynomial interpolating this zero data, we have

$$\tilde{Z}_{m,i}(x)|_{[x_k, x_{k+1}]} \equiv 0. \quad (7.23)$$

ii) Let $i$ satisfy $1 \leq i \leq m+1$. For $j > i+m$ we have that $\tilde{Z}_{m,i}(x_k) = 0$ for $k \in I_j$ given in I.1 or I.3. Hence $p_{m,j} \equiv 0$ and thus all its derivatives are zero. In particular this is the case for $j = k, k+1$ if one has

$$[x_k, x_{k+1}] \subset [x_1, x_N] \setminus (x_1, x_{i+m+1}).$$

As $\tilde{Z}_{m,i}(x)$ for $x \in [x_k, x_{k+1}]$ is the Hermite interpolation polynomial interpolating this zero data, we have

$$\tilde{Z}_{m,i}(x)|_{[x_k, x_{k+1}]} \equiv 0. \quad (7.24)$$

iii) Let $i$ satisfy $n-m \leq i \leq n$. This is analogous to the case ii) with $i$ close to $n$ instead of 1. Therefore we omit the details. $\square$

We explicitly construct the Z-spline basis functions in each subinterval. This is done in two steps. We first derive how one obtains the needed derivatives of the polynomials $p_{m,j}(x)$. To simplify the notation for this procedure, assume $m$ to be fixed and denote $p_{m,j}(x)$ by $P(x)$ and the interpolation data by

$$(x_i, f_i) \text{ for } i = 1:2m+1. \quad (7.25)$$

We need formulas for

$$\frac{d^p}{dx^p} P(x)|_\Xi =: \bar{f}^p, \quad (7.26)$$
where \( \bar{x} \in \{x_1, \ldots, x_{2m+1}\} \).

Clearly we have
\[
P(x) = \sum_{p=0}^{2m} \frac{f^p}{p!} (x - \bar{x})^p. \tag{7.27}
\]

The interpolation conditions can be written as
\[
f_i = P(x_i) = \sum_{p=0}^{2m} \frac{f^p}{p!} (x_i - \bar{x})^p \quad \text{for } i = 1:2m. \tag{7.28}
\]

This can be written in matrix notation by introducing
\[
f = (f_1, f_2, \ldots, f_{2m+1})^T, \tag{7.29}
\]
\[
\bar{f} = (\bar{f}^0, \bar{f}^1, \ldots, \bar{f}^{2m})^T, \tag{7.30}
\]
\[
D = \text{diag} \left( 1, \frac{1}{1!}, \frac{1}{2!}, \ldots, \frac{1}{(2m + 1)!} \right), \tag{7.31}
\]

and the Vandermonde matrix \( V \) with
\[
[V]_{i,p} = (x_i - \bar{x})^{p-1} \quad \text{for } i, p = 1:2m + 1. \tag{7.32}
\]

Hence (7.28) is equivalent to
\[
f = VD\bar{f}. \tag{7.33}
\]

This leads to the needed formulas for the derivatives
\[
\bar{f} = D^{-1}V^{-1}f. \tag{7.34}
\]

The inverse of the Vandermonde matrix \( V^{-1} \) is given by the elements
\[
[V]_{r,s}^{-1} = \frac{(-1)^{2m+1-r}}{(2m + 1 - r)!(r-1)!} v_{r,s} \quad \text{for } r, s = 1:2m + 1. \tag{7.35}
\]
where $v_{r,s}$ is the coefficient of $x^{r-1}$ in the polynomial

\[ l(x) = \frac{(x - \xi_1)(x - \xi_2)(x - \xi_3)...(x - \xi_{2m+1})}{x - \xi_s}, \quad (7.36) \]

given that the $(r, s)$, element of the Vandermonde matrix is $\xi_r^{-1}$.

We construct the Z-spline basis functions in every subinterval $x \in [x_i, x_{i+1}]$ using the Hermite interpolation formula [9]

\[ \tilde{Z}_m(x) = \sum_{p=0}^{m} \left( \tilde{f}_p(x_i)B_{p0}(x) + \tilde{f}_p(x_{i+1})B_{p1}(x) \right), \quad (7.37) \]

where

\[ B_{p0}(x) = \frac{1}{p!}(x - x_i)^p \left( \sum_{\nu=0}^{m-p-1} (x - x_i)^\nu b_{\nu 0} \right) l_0^m(x), \quad (7.38) \]

\[ B_{p1}(x) = \frac{1}{p!}(x - x_{i+1})^p \left( \sum_{\nu=0}^{m-p-1} (x - x_{i+1})^\nu b_{\nu 1} \right) l_1^m(x), \quad (7.39) \]

with

\[ b_{\nu k} = \frac{1}{\nu!} \left( \frac{1}{l_k^m(x_i + k)} \right)^{(\nu)}, \quad l_0(x) = \frac{x - x_{i+1}}{x_i - x_{i+1}}, \quad l_1(x) = \frac{x - x_i}{x_{i+1} - x_i}. \]

### 7.1.1 Examples of general Z-splines

In this section we present explicit formulas for the cubic Z-splines given arbitrary intervals and near boundaries.
Figure 7.2: The arbitrarily spaced cubic Z-spline basis function (Example 7.1.8) for $a_1 = a_2 = 1$ and $a_3 = a_4 = 0.5$.

**Example 7.1.8.** The centered, cubic Z-spline basis function for arbitrary intervals $a_1, a_2, a_3$ and $a_4$ is given by

$$
\tilde{Z}_1(x) = \begin{cases} 
0 & x < -a_1 - a_2 \\
\left( \frac{a_2+a_1}{a_1} \right) + \left( \frac{3a_2+a_1}{a_2a_1} \right)x + \frac{3a_2+2a_1}{a_2a_1(a_2+a_1)}x^2 + \frac{1}{a_2a_1(a_2+a_1)}x^3 & -a_1 - a_2 \leq x \leq -a_2, \\
1 - \left( \frac{1}{a_3} - \frac{1}{a_2} \right)x - \frac{a_3+2(a_2+a_1)}{a_3a_2(a_2+a_1)}x^2 - \frac{a_3+a_2+a_1}{a_3a_2^2(a_2+a_1)}x^3 & -a_2 \leq x \leq 0, \\
1 + \left( \frac{1}{a_2} - \frac{1}{a_3} \right)x - \frac{a_2+2(a_3+a_2)}{a_2a_3(a_3+a_4)}x^2 - \frac{a_2+a_3+a_4}{a_2a_3^2(a_3+a_4)}x^3 & 0 \leq x \leq a_3, \\
\left( \frac{a_3+a_4}{a_4} \right) - \left( \frac{3a_3+a_4}{a_3a_4} \right)x + \frac{3a_4+2a_3}{a_3a_4(a_3+a_4)}x^2 - \frac{1}{a_3a_4(a_3+a_4)}x^3 & a_3 \leq x \leq a_3 + a_4, \\
0 & x > a_3 + a_4.
\end{cases}
$$
Figure 7.3: One-sided, cubic, cardinal Z-spline basic functions with a boundary on the left.
Example 7.1.9. The one-sided cubic Z-spline basis functions with a boundary on the left and intervals $a_1$, $a_2$, $a_3$ and $a_4$ from left to right:

$$
\tilde{Z}_{1,1}(x) = \begin{cases} 
0 & x \leq 0, \\
1 - \frac{2a_3+a_4}{a_3(a_3+a_4)}x + \frac{1}{a_3(a_3+a_4)}x^2 & 0 \leq x \leq a_3, \\
\tilde{Z}_1(x) & a_3 \leq x.
\end{cases}
$$

$$
\tilde{Z}_{1,2}(x) = \begin{cases} 
0 & x \leq -a_2, \\
1 - \frac{2a_3+a_4}{a_3(a_3+a_4)}x + \frac{1}{a_3(a_3+a_4)}x^2 & -a_2 \leq x \leq 0, \\
\tilde{Z}_1(x) & 0 \leq x.
\end{cases}
$$

$$
\tilde{Z}_{1,3}(x) = \begin{cases} 
0 & x \leq -a_1 - a_2, \\
1 + \frac{a_1+2a_2}{a_2(a_1+a_2)}x - \frac{1}{a_2(a_1+a_2)}x^2 & -a_1 - a_2 \leq x \leq -a_2, \\
\tilde{Z}_1(x) & -a_2 \leq x.
\end{cases}
$$

7.2 Cardinal Z-splines

Cardinal splines, in general, are special cases of spline interpolations given the data points $x_j = j$ for $j \in \mathbb{Z}$. In this situation the formulas for the Z-splines simplify.

In our application Z-splines are used to interpolate functions given on a grid. In regions away from boundaries and drastic function changes one will use an equidistant grid. To model this situation we assume that the domain is $\mathbb{R}$ and without loss of generality we can assume the grid width to be 1.

Hence all basis functions have the same shape and differ only by integer shifts. We introduce as $\tilde{Z}_m(x)$ the basis function for cardinal Z-splines as the Z-spline interpolating the data

$$
x_j = j, \quad f_j = \delta_{0j} \quad \text{for } j \in \mathbb{Z}.
$$

(7.40)
With this definition the general interpolation formula (7.18) for gen-
eral data can be written as

\[ Z_m(x) = \sum_{i=\infty}^{\infty} f_i Z_m(x - x_i). \] (7.41)

Figure 7.4: The first four cardinal \( Z \)-spline basis functions \( \tilde{Z}_m \) and
their Fourier transforms \( \tilde{Z}_m \).

We give the values of the derivatives \( \tilde{f} = A_m f \) in matrix form (see
(7.34)). Observe that given the equivalence of the unique interpo-
High-order Discrete Representation of Functions

Integrating polynomial (7.27) with the truncated Taylor expansion, the results are the well known finite difference matrices

\[ A_0 = [1], \]

\[ A_1 = \begin{bmatrix} 0 & 1 & 0 \\ \frac{-1}{2} & 0 & \frac{1}{2} \\ 1 & -2 & 1 \end{bmatrix}, \]

\[ A_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ \frac{1}{12} & \frac{-2}{3} & 0 & \frac{2}{3} & \frac{-1}{12} \\ \frac{-1}{12} & \frac{4}{3} & \frac{-5}{2} & \frac{4}{3} & \frac{-1}{12} \\ \frac{-1}{2} & 1 & 0 & -1 & \frac{1}{2} \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}, \]

and

\[ A_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{-1}{60} & \frac{3}{20} & \frac{-3}{4} & 0 & \frac{3}{4} & \frac{-3}{20} & \frac{1}{60} \\ \frac{1}{90} & \frac{-3}{20} & \frac{3}{2} & \frac{-49}{3} & \frac{3}{2} & \frac{-3}{20} & \frac{1}{90} \\ \frac{1}{8} & -1 & \frac{13}{8} & 0 & \frac{-13}{8} & 1 & \frac{-1}{8} \\ \frac{-1}{6} & 2 & \frac{-13}{2} & \frac{28}{3} & \frac{-13}{2} & 2 & \frac{-1}{6} \\ \frac{-1}{2} & 2 & \frac{-5}{2} & 0 & \frac{5}{2} & -2 & \frac{1}{2} \\ 1 & -6 & 15 & -20 & 15 & -6 & 1 \end{bmatrix}. \]

Three cardinal Z-spline basis functions were known previously. These are:

the linear interpolation

\[ \tilde{Z}_0(x) = \begin{cases} 1 - |x| & |x| \leq 1, \\ 0 & |x| > 1, \end{cases} \quad (7.42) \]

which was already used in Babylon times [35].

The cubic convolution interpolation (equivalent to cubic Bessel inter-
polation [15])

\[ \tilde{Z}_1(x) = \begin{cases} 
1 - \frac{5}{2}x^2 + \frac{3}{2}|x|^3 & |x| \leq 1, \\
\frac{1}{2} (2 - |x|)^2 (1 - |x|) & 1 \leq |x| \leq 2, \\
0 & |x| > 2; 
\end{cases} \] (7.43)

used by Karup and King [35] around 1900.

And the perfect reconstruction filter (or sinc function)

\[ \tilde{Z}_\infty(x) = \frac{\sin \pi (x - j)}{\pi (x - j)}, \] (7.44)

studied by Whittaker [59] in 1915.

We give two high-order cardinal Z-spline basis functions of practical value for our purposes:

**Example 7.2.1.** The $C^2$, quintic cardinal Z-spline basis function

\[ \tilde{Z}_2(x) = \begin{cases} 
1 - \frac{15}{12}x^2 - \frac{35}{12}|x|^3 + \frac{63}{12}|x|^4 - \frac{25}{12}|x|^5 & |x| \leq 1, \\
-4 + \frac{75}{4}|x| - \frac{245}{8}x^2 \\
+ \frac{545}{24}|x|^3 - \frac{63}{8}|x|^4 + \frac{25}{24}|x|^5 & 1 \leq |x| \leq 2, \\
18 - \frac{153}{4}|x| + \frac{255}{8}x^2 \\
- \frac{315}{24}|x|^3 + \frac{21}{8}|x|^4 - \frac{5}{24}|x|^5 & 2 \leq |x| \leq 3, \\
0 & |x| > 3.
\]
Example 7.2.2. The $C^3$, seventh-degree cardinal Z-spline basis function

$$\tilde{Z}_3(x) = \begin{cases} 
1 - \frac{49}{36} x^2 - \frac{959}{144} x^4 + \frac{2569}{144} |x|^5 - \frac{727}{48} x^6 + \frac{623}{144} |x|^7 & |x| \leq 1, \\
\frac{138}{5} - \frac{8617}{60} |x| + \frac{12873}{40} x^2 - \frac{791}{2} |x|^3 \\
+ \frac{4557}{16} x^4 - \frac{9583}{80} |x|^5 + \frac{21811}{80} x^6 - \frac{623}{240} |x|^7 & 1 \leq |x| \leq 2, \\
-440 + \frac{25949}{20} |x| - \frac{17131}{72} x^2 + \frac{2247}{2} |x|^3 \\
-\frac{66437}{144} x^4 + \frac{81109}{720} |x|^5 - \frac{727}{48} x^6 + \frac{623}{720} |x|^7 & 2 \leq |x| \leq 3, \\
\frac{3632}{5} - \frac{7456}{5} x^2 + \frac{58786}{45} x^2 - 633 |x|^3 \\
+ \frac{26383}{144} x^4 - \frac{22807}{720} |x|^5 + \frac{727}{240} x^6 - \frac{89}{720} |x|^7 & 3 \leq |x| \leq 4, \\
0 & |x| > 4.
\end{cases}$$

7.2.1 Approximation of Functions by cardinal Z-splines

We derive the interpolation error produced by the cardinal Z-splines when used to approximate a function $f = f(x)$ given as the samples $f_i = f(i)$ for $i \in \mathbb{Z}$.

The approximation of a function by Z-splines makes sense only if the conditions given by Shannon’s theorem are fulfilled.

Shannon’s Theorem (Kolmogorov’s Theorem). Part I. If $f$ is bandlimited, i.e., its spectrum

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x)e^{-ikx} dx$$

is limited to a finite range of wavenumbers $|k| \leq k_c$, then it is not necessary to sample it at every point $x$ to specify it completely. The function $f$ can be recovered completely from its samples $f_i$ if the sample interval is equal or smaller to the Nyquist interval, i.e.,

$$\Delta x \leq \pi/k_c.$$  \hspace{1cm} (7.46)

If the sampling interval $\Delta x > \pi/k_c$, this situation is known as undersampling and if $\Delta x < \pi/k_c$, is known as oversampling.
Shannon’s Theorem. Part II. Let \( f \) be a bandlimited function whose samples \( f_i = f(i) \), for \( i \in \mathbb{Z} \), are such that \( 1 \leq \pi / k_c \). The function can be simply reconstructed from the samples using a pulse of the type \( \sin(\pi x)/(\pi x) \).

We shall prove now that the accuracy of the \( Z \)-splines is related to the exact reproduction of polynomials \( q(x) \in \pi_{2m} \) (Proposition 7.1.4). This property, combined with the compact support (Proposition 7.1.7) and the orthonormality (Definition 7.1.6) of the \( Z \)-spline basis functions, produce the following equivalent results.

**Proposition 7.2.1.** The first \( 2m+1 \) discrete moments of the cardinal \( Z \)-spline basis function \( \tilde{Z}_m \) are conserved, i.e., for \( n = 0:2m \) we have

\[
\sum_{j=-\infty}^{\infty} (x - j)^n \tilde{Z}_m(x - j) = \delta_{0n} = \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{7.47}
\]

*Proof:* Applying Proposition 7.1.4 to cardinal \( Z \)-splines gives

\[
x^n = Z_m(x) = \sum_{j=-\infty}^{\infty} j^n \tilde{Z}_m(x - j) \quad \text{for } n = 0:2m. \tag{7.48}
\]

Hence

\[
\sum_{j=-\infty}^{\infty} (x - j)^n \tilde{Z}_m(x - j) = \sum_{j=-\infty}^{\infty} \sum_{s=0}^{n} x^s \binom{n}{s} (-j)^{n-s} \tilde{Z}_m(x - j)
\]

\[
= \sum_{s=0}^{n} x^s \binom{n}{s} (-1)^{n-s} \sum_{j=-\infty}^{\infty} j^{n-s} \tilde{Z}_m(x - j)
\]

\[
= \sum_{s=0}^{n} x^s \binom{n}{s} (-1)^{n-s} x^{n-s}
\]

\[
= (x - x)^n = \delta_{0n}.
\]

Notice that the second equality sign holds since for each fixed value of \( x \) the infinite sum is in fact only a finite sum as by Proposition 7.1.7, \( \tilde{Z}_m(x) \) has compact support. \( \square \)
**Proposition 7.2.2.** For sufficiently smooth functions, the cardinal Z-spline basis functions are $L_2$ accurate to order $2m + 1$.

**Proof:** The interpolation error for equidistant data taken at intervals $\Delta x = x_{j+1} - x_j$ is computed as the difference

$$f(x) - Z_m(x) = f(x) - \sum_{j=-\infty}^{\infty} f(x_j) \frac{Z_m(x_j)}{\Delta x}.$$ 

(7.49)

Using Proposition 7.2.1 for $n = 0$ and substituting $f(x_j)$ by its Taylor expansion leads to

$$f(x) - Z_m(x) = \sum_{j=-\infty}^{\infty} \left( \sum_{p=1}^{2m} \frac{f^{(p)}(x)}{p!} (x - x_j)^p + R_{2m+1}(x, x_j) \right) \frac{Z_m(x_j)}{\Delta x},$$

(7.50)

where $R_{2m+1}(x, x_j)$ is the remainder of the series, given by the mean value theorem as

$$R_n(x, x_j) = \frac{(x - x_j)^n}{n!} f^{(n)}(\xi)$$

(7.51)

for $\xi$ in between $x$ and $x_j$ or viceversa.

Exchanging the order of the sums in (7.50) leads to

$$f(x) - Z_m(x) = \sum_{p=1}^{2m} \frac{f^{(p)}(x)}{p!} \sum_{j=-\infty}^{\infty} (x - x_j)^p \frac{Z_m(x_j)}{\Delta x} + \sum_{j=-\infty}^{\infty} R_{2m+1}(x, x_j) \frac{Z_m(x_j)}{\Delta x}.$$ 

(7.52)

Proposition 7.2.1 for the case of $\Delta x$ constant and $p \leq 2m$, can be written as

$$\sum_{j=-\infty}^{\infty} (x - x_j)^p \frac{Z_m(x_j)}{\Delta x} = \delta_{p0}.$$ 

(7.53)
Using the conservation of moments (7.53) in (7.52) leads to

\[
f(x) - Z_m(x) = \sum_{j=-\infty}^{\infty} \frac{(x - x_j)^{2m+1}}{(2m+1)!} f^{(2m+1)}(\xi_j) \bar{Z}_m \left( \frac{x - x_j}{\Delta x} \right),
\]

for some \( \xi_j \)'s in between \( x \) and \( x_j \) or vice versa.

From this expression we can compute the exact \( L_2 \) error over the interval \( x \in [x_i, x_i + \Delta x] = [x_i, x_{i+1}] \) as

\[
\|f - Z_m\|_{L_2} = \frac{1}{\sqrt{\Delta x}} \left( \int_{x_i}^{x_{i+1}} (f(x) - Z_m(x))^2 \, dx \right)^{1/2}.
\]

We want to get a simplified bound for the error and therefore we take the absolute value inside the sum to get

\[
\|f - Z_m\|_{L_2}^2 \leq 
\]

\[
\frac{1}{\Delta x} \int_{x_i}^{x_{i+1}} \left[ \sum_{j=-\infty}^{\infty} |f^{(2m+1)}(\xi_j)| \left| \frac{(x - x_j)^{2m+1}}{(2m+1)!} \bar{Z}_m \left( \frac{x - x_j}{\Delta x} \right) \right| \right]^2 \, dx.
\]

We bound the derivative by its maximum value to find

\[
\|f - Z_m\|_{L_2} \leq 
\]

\[
\frac{\max |f^{(2m+1)}|}{\sqrt{\Delta x} (2m+1)!} \left( \int_{x_i}^{x_{i+1}} \left( \sum_{j=-\infty}^{\infty} |(x - x_j)^{2m+1} \bar{Z}_m \left( \frac{x - x_j}{\Delta x} \right)| \right)^2 \, dx \right)^{1/2}.
\]

Here, the maximum of the derivative \( f^{(2m+1)} \) and the sum inside the integral are defined only over the neighboring points \( x_j \) of \( x_i \) because the Z-splines have compact support (Proposition 7.1.7).

We bound the Z-spline by the functions \( 1 \) and \( 1/(\pi x) \) as shown in Figure 7.5. Then we bound the distances \( (x - x_j) \) inside the sum by the closest bigger multiple of \( \Delta x \).
Finally, we obtain the bound for the $L_2$ error
\[
\|f - Z_m\|_{L_2} \leq C_m \max |f^{(2m+1)}| (\Delta x)^2m+1
\] (7.58)
where $C_m$ is a real constant independent of $f$. The same expression is obtained for the error in the maximum or $L_\infty$ norm.

The constant $C_m$ is approximated in both norms by
\[
C_m = \frac{2(\pi + \sum_{j=2}^{m+1} j^{2m})}{\pi (2m + 1)!},
\] (7.59)
such that $C_1 = 0.757$, $C_2 = 0.531$, $C_3 = 0.617$, $C_4 = 0.812$, $C_5 = 1.137$, $C_6 = 1.664$, ... Tighter approximations for $C_m$ should be found if the approximation in (7.56) is improved. \(\square\)

**Proposition 7.2.3.** The cardinal Z-spline basis functions are orthogonal with respect to the monomials $x^n$ for $n = 0 : 2m$. By this we mean
\[
\int_{-\infty}^{\infty} x^n \tilde{Z}_m(x) dx = \delta_{n0}.
\] (7.60)
Proof: We divide the integral into intervals of length 1 and use Proposition 7.2.1 to get

$$\int_{-\infty}^{\infty} x^n \tilde{Z}_m(x) dx = \sum_{j=-\infty}^{\infty} \int_0^1 (\xi-j)^n \tilde{Z}_m(\xi-j) d\xi; \quad (7.61)$$

$$= \int_0^1 \sum_{j=-\infty}^{\infty} (\xi-j)^n \tilde{Z}_m(\xi-j) d\xi = \delta_{n0}$$

for $n = 0: 2m$. □

Proposition 7.2.4. The Fourier transform of the cardinal Z-spline basis function, $\tilde{Z}_m$, is unity at zero and has zeroes of order $2m$ at the multiples of $2\pi$.

Proof: The Fourier transforms of the Z-spline basis functions and their derivatives are

$$\hat{Z}^{(p)}_m(k) = \int_{-\infty}^{\infty} (-ix)^p \tilde{Z}_m(x) e^{-ikx} dx. \quad (7.62)$$

We divide the integral as in Proposition 7.2.3 to get

$$\int_{-\infty}^{\infty} (-ix)^p \tilde{Z}_m(x) e^{-ikx} dx = \sum_{j=-\infty}^{\infty} (-i)^p \int_0^1 (\xi-j)^p \tilde{Z}_m(\xi-j) e^{-ik(\xi-j)} d\xi$$

$$= (-i)^p \int_0^1 e^{-ik\xi} \sum_{j=-\infty}^{\infty} (\xi-j)^p \tilde{Z}_m(\xi-j) e^{ikj} d\xi$$

for $p = 0: 2m$. Notice that inside the sum $e^{ikj} = 1$ for $k$ multiple of $2\pi$. In this case we can use Proposition 7.2.1 and therefore, for $p \leq 2m$,

$$\hat{Z}^{(p)}_m(0) = \delta_{p0},$$

$$\hat{Z}^{(p)}_m(2\pi l) = 0, \quad l \in \mathbb{Z}, \quad l \neq 0. \quad (7.63)$$

Proposition 7.2.5. The cardinal Z-spline basis functions converge to the perfect reconstruction filter (also called cardinal function or sinc function)

$$\tilde{Z}_\infty(x) = \frac{\sin(\pi x)}{\pi x}.$$
Figure 7.6: $L_2$ error between $sinc(x)$ and the first 100 Z-splines for $x \in [-5, 5]$.

*Proof:* We lack of a clear and simple proof to this proposition. A convergent $C_m$ for large $m$ in Proposition 7.2.2, would lead to this result trivially, given that from Shannon’s Theorem we know that the sinc function gives the unique basis function for the perfect reconstruction of a bandlimited function $f$.

Instead, we have computed the $L_2$ norm of the difference between the first 100 Z-splines and the sinc function numerically, shown in Figure 7.6. The Z-splines converge to the sinc function. $\square$

A direct comparison of some Z-splines with the sinc function is shown in Figure 7.7.

We have implemented the first four cardinal Z-splines with explicit formulas and have computed their numerical interpolation error. This is given in the Table 7.1. The error is computed for the test function $\sin(\pi x)$ using the $L_2$ norm in integral form. The samples have a constant phase equal to zero. The numerical convergence of the cardinal
Figure 7.7: The sinc function and the cardinal Z-spline basis functions for $m = 0, 1, 2, 4, 9$ and 19.

<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>$\text{err}(Z_0)$</th>
<th>$\text{err}(Z_1)$</th>
<th>$\text{err}(Z_2)$</th>
<th>$\text{err}(Z_3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.500</td>
<td>$1.26 \times 10^{-1}$</td>
<td>$5.32 \times 10^{-2}$</td>
<td>$2.27 \times 10^{-2}$</td>
<td>$1.01 \times 10^{-2}$</td>
</tr>
<tr>
<td>0.250</td>
<td>$3.30 \times 10^{-2}$</td>
<td>$3.60 \times 10^{-3}$</td>
<td>$4.3 \times 10^{-4}$</td>
<td>$5.63 \times 10^{-5}$</td>
</tr>
<tr>
<td>0.166</td>
<td>$1.46 \times 10^{-2}$</td>
<td>$7.22 \times 10^{-4}$</td>
<td>$4.01 \times 10^{-5}$</td>
<td>$2.34 \times 10^{-6}$</td>
</tr>
<tr>
<td>0.125</td>
<td>$8.20 \times 10^{-3}$</td>
<td>$2.29 \times 10^{-4}$</td>
<td>$7.24 \times 10^{-6}$</td>
<td>$2.40 \times 10^{-7}$</td>
</tr>
<tr>
<td>0.100</td>
<td>$5.24 \times 10^{-3}$</td>
<td>$9.43 \times 10^{-5}$</td>
<td>$1.91 \times 10^{-6}$</td>
<td>$4.08 \times 10^{-8}$</td>
</tr>
</tbody>
</table>

Table 7.1: Interpolation error of the Z-splines for the test function $\sin(\pi x)$. 
Z-splines for this Fourier component is found using the least-squares method and the form of the error $err(Z_m) = C_m \Delta x^{\beta_m}$. It is found that the constants of proportionality have the values: $C_0 = 0.54$, $C_1 = 0.87$, $C_2 = 1.48$ and $C_3 = 2.67$. The numerical convergence exponents are: $\beta_0 = 2.01$, $\beta_1 = 3.97$, $\beta_2 = 5.89$ and $\beta_3 = 7.83$.

### 7.3 Multi-dimensional Z-splines

During this work we do not consider the study of Z-splines for an arbitrary set of points in multiple dimensions. In principle, the one-dimensional definition could be generalized to an arbitrary set of points using interpolating surfaces and constructing the respective multi-dimensional Hermitian splines.

In our applications we use orthogonal grids and extend the Z-spline basis functions to higher dimensions using the tensor product of one-dimensional Z-spline basis functions

$$\tilde{Z}_m(x_1, x_2, ..., x_n) = \tilde{Z}_m(x_1)\tilde{Z}_m(x_2)...\tilde{Z}_m(x_n).$$

If the three-dimensional fields or variables $f = f(x, y, z)$ are represented from its discrete values $f_{i,j,k} = f(x_i, y_j, z_k)$ in Cartesian coordinates, then the interpolation to a point $(x, y, z)$ such that, for some integers $r, s$ and $t$, $x_r \leq x \leq x_{r+1}$, $y_s \leq y \leq y_{s+1}$ and $z_t \leq z \leq z_{t+1}$, is given by

$$Z_m(x, y, z) = \sum_{i=r-m}^{r+m+1} \sum_{j=s-m}^{s+m+1} \sum_{k=t-m}^{t+m+1} f_{i,j,k} \tilde{Z}_m(x-x_i, y-y_j, z-z_k).$$

Data points given in other orthogonal coordinates systems can be interpolated using the analytical mapping between the orthogonal coordinates and the Cartesian coordinates. The mapping is used to find the location of the interpolation point in the Cartesian coordinates. Then the interpolation is calculated over the Cartesian mesh.
Interpolations using a deformed Cartesian grid are possible using an algebraic non-linear mapping. This procedure is particularly important for the FSL where the volume elements move and deform in time and it is described in the next Section.

### 7.3.1 Interpolation over a smoothly deformed Cartesian mesh

Let the three-dimensional data points \( f_{i,j,k} \) be given over a Cartesian mesh \( Q_{i,j,k} = (x_i, y_j, z_k) \). Let the locations of the Cartesian nodes be changed slightly by a smooth vector function. The new mesh \( P_{i,j,k} = (x_{ijk}, y_{ijk}, z_{ijk}) \) is a deformation of the original Cartesian mesh. The interpolations from the deformed mesh \( P \) to the regular Cartesian mesh \( Q \) are possible if the volume elements of the deformed mesh have not collapsed or crossed coordinate lines during the deformation.
The interpolations are performed using an exact, non-linear, algebraic mapping. The idea is to map every deformed volume element to a unit cube using the Z-spline representation for the coordinate lines. The mapping is given by the Taylor expansions from the corner point $P = (x_{ijk}, y_{ijk}, z_{ijk})$ of the volume element shown in Figure 7.8. If the point $Q$ is inside this element and the vector from point $P$ to point $Q$ is given in $(x, y, z)$ coordinates as $PQ = (\Delta x, \Delta y, \Delta z)$, then it is mapped to the vector $P'Q' = (\Delta \xi, \Delta \eta, \Delta \zeta)$, in the $(\xi, \eta, \zeta)$ coordinates of the mapped (Cartesian) space. The relation between the coordinate systems is given by the Taylor expansions from point $P'$ to point $Q'$:

\[
\begin{align*}
\Delta x &= \frac{\partial x}{\partial \xi} (P') \Delta \xi + \frac{\partial x}{\partial \eta} (P') \Delta \eta + \frac{\partial x}{\partial \zeta} (P') \Delta \zeta + \ldots, \\
\Delta y &= \frac{\partial y}{\partial \xi} (P') \Delta \xi + \frac{\partial y}{\partial \eta} (P') \Delta \eta + \frac{\partial y}{\partial \zeta} (P') \Delta \zeta + \ldots, \\
\Delta z &= \frac{\partial z}{\partial \xi} (P') \Delta \xi + \frac{\partial z}{\partial \eta} (P') \Delta \eta + \frac{\partial z}{\partial \zeta} (P') \Delta \zeta + \ldots.
\end{align*}
\] (7.66)-(7.68)

All the geometric partial derivatives different than zero are computed numerically at the point $P'$ using the Z-spline representation of $x = x(\xi, \eta, \zeta)$, $y = y(\xi, \eta, \zeta)$ and $z = z(\xi, \eta, \zeta)$. Partial derivatives of multi-dimensional Z-splines are products of the respective derivatives of the one-dimensional Z-spline basis functions, evaluated at the nodes. The derivatives of the Z-spline basis functions at the nodes are given by the finite differences matrices (see Section 7.2).

The non-linear mapping (7.66)-(7.68) is exact because Taylor expansions are computed using all the finite number of non-vanishing derivatives. The mapped coordinates $P'Q' = (\Delta \xi, \Delta \eta, \Delta \zeta)$ are obtained by inverting the non-linear mappings using Newton’s method. The initial condition for Newton’s method is obtained by solving the same problem ignoring all the non-linear terms. Finally, the interpolation is done over a Cartesian mesh in the mapped space using (7.65).
Part IV

Results
8.1 Two-dimensional Scalar transport

The multi-dimensional advection of a scalar is the most basic test case for the SL transport scheme. The velocity field is given constant in time and an initial scalar field is transported along the trajectories of the fluid elements. This process is described by the ODEs

\[ \frac{D}{Dt} \alpha(x, t) = 0, \]  

where \( x = [x_1, x_2]^T \) and

\[ \frac{d}{dt} x_i(t) = u_i \quad \text{for } i = 1:2. \]  

The purpose of this test is to analyze the numerical errors introduced by the advection of the fluid variables. Our high-order forward-trajectory semi-Lagrangian (FSL) scheme is compared to the traditional linear and cubic spline SL schemes.
8.1.1 Solid body rotation

The best known test case in the literature about SL advection is the solid body rotation test. This test is studied in detail for traditional SL schemes and finite volumes in the work of Bonaventura [7]. The rotation takes place around the center of the domain \([-1, 1] \times [-1, 1]\). The rotation period is \(T_r = 1000\), the spatial resolution \(\Delta x = 1/25\) and the time step \(\Delta t = 5.0\). The simulation is run up to one revolution.

The initial condition is given by the compactly supported function

\[
\alpha(x_1, x_2) = \begin{cases} 
5 \left[1 + \cos \left( \frac{\pi \sqrt{2}}{0.2} \right) \right] & \text{for } \beta \leq 0.04, \\
0 & \text{otherwise,}
\end{cases}
\]

where \(\beta = (x_1 + 0.5)^2 + x_2^2\).

Figure 8.1 shows the contours of the initial condition and the results for bidimensional \(Z_0, Z_1\) and \(Z_2\) interpolations. The result for \(Z_2\) shows minimum numerical dissipation.

A quantitative analysis of the FSL is done by calculating the relative \(L_1\)-error

\[
e_{rel}^1 = \frac{\sum_{i=1}^{N} |\alpha_i^n - \alpha(x_i, t^n)|}{\sum_{i=1}^{N} |\alpha(x_i, t^n)|},
\]

the relative \(L_2\)-error

\[
e_{rel}^2 = \sqrt{\frac{\sum_{i=1}^{N} (\alpha_i^n - \alpha(x_i, t^n))^2}{\sum_{i=1}^{N} \alpha(x_i, t^n)^2}},
\]

the dissipation error

\[
e_{diss} = [\sigma(\alpha^n) - \sigma(\alpha(x_i, t^n))]^2 + (\alpha^n - \alpha(x_i, t^n))^2,
\]

and the minimum of the function. Here, \(i\) is a global index for the nodes in the mesh, \(\alpha^n_i\) is the numerical approximation to \(\alpha(x_i, t^n)\),
Figure 8.1: Contours of the function $\alpha$, equation (8.3), for the solid body rotation test. Initial condition (top left). Results after one revolution using the FSL scheme with $Z_0$ (top right), $Z_1$ (bottom left) and $Z_2$ (bottom right) interpolations.
Table 8.1: Errors of the FSL method for the bidimensional solid body rotation after one period. The linear and cubic methods are results obtained for traditional SL methods using B-spline interpolation and exact trajectories [7].

<table>
<thead>
<tr>
<th>Interpolation</th>
<th>$e^{rel}_2$</th>
<th>$e^{rel}_1$</th>
<th>$e_{diss}$</th>
<th>minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>0.89</td>
<td>1.51</td>
<td>0.75</td>
<td>0.0</td>
</tr>
<tr>
<td>Cubic</td>
<td>0.21</td>
<td>0.35</td>
<td>2.76 x 10^{-3}</td>
<td>-0.98</td>
</tr>
<tr>
<td>FSL-Z0</td>
<td>0.87</td>
<td>1.50</td>
<td>0.27</td>
<td>0.0</td>
</tr>
<tr>
<td>FSL-Z1</td>
<td>0.39</td>
<td>0.67</td>
<td>1.13 x 10^{-2}</td>
<td>-0.81</td>
</tr>
<tr>
<td>FSL-Z2</td>
<td>4.75 x 10^{-2}</td>
<td>0.10</td>
<td>9.64 x 10^{-6}</td>
<td>-0.25</td>
</tr>
</tbody>
</table>

Table 8.2: Discrete moments of the pulse used in the solid body rotation test after one period.

$$x_i = (x_1(i), x_2(i)),$$ and the definitions of the mean and the standard deviation for a generic mesh function $\phi_i$ are

$$\bar{\phi} = \frac{1}{N} \sum_{i=1}^{N} \phi_i \quad \sigma(\phi)^2 = \frac{1}{N} \sum_{i=1}^{N} (\phi_i - \bar{\phi})^2. \quad (8.7)$$

The Table 8.1 shows the comparison of our FSL schemes with the traditional SL schemes for linear and cubic B-spline interpolation. The simulation using Z2 interpolation is a significant improvement from the traditional results. It produces almost three orders of magnitude less numerical dissipation.

The sum of the discrete moments

$$m_p = \sum_i \alpha_i^p \left(x_1(i)^p + x_2(i)^p\right) \quad (8.8)$$

is calculated after one complete solid body rotation. Here, $x_k$ is the position 3-vector, $x_k(i)$ is the $k-th$ component of the position vector
8.1 Two-dimensional Scalar transport

at mesh position \( i \) and \( p = 0, 1, 2, \ldots \). The results for \( Z_0, Z_1 \) and \( Z_2 \) interpolations are presented in the Table 8.2.

8.1.2 Taylor-Green vortices

The convergence of the FSL method is studied in the doubly-periodic domain \([0, 1] \times [0, 1]\). The velocity field is given by the bidimensional Taylor-Green vortices:

\[
\begin{align*}
  u_1(x_1, x_2) &= -2\pi \cos(2\pi x_1) \sin(2\pi x_2), \\
  u_2(x_1, x_2) &= 2\pi \sin(2\pi x_1) \cos(2\pi x_2).
\end{align*}
\] (8.9)

In the first test, the scalar field is an invariant of motion: the vorticity field

\[
\alpha(x_1, x_2) = 8\pi^2 \cos(2\pi x_1) \cos(2\pi x_2).
\] (8.10)

The test includes RK3 integration in time; bidimensional linear \( Z_0 \) and cubic \( Z_1 \) volume elements; and bidimensional linear \( Z_0 \), cubic \( Z_1 \) and quintic \( Z_2 \) interpolations. The solution is compared after ten time steps for \( \Delta t = 0.001 \). The numerical errors produced by the FSL are shown in Figure 8.2.

The numerical convergence is found to be mostly related to the spatial interpolation error. The time integration error is minimal and there is a very small difference in the results between the linear and the cubic volume elements.

In the second test, a two-color scalar field is used to simulate a mixing process and show, in qualitative form, the generation of numerical errors by the FSL when computing undersampled fields. The initial condition is the periodic hyperbolic tangent

\[
\alpha(x_1, x_2) = \begin{cases} 
  \tanh(-20 x_1) & 0.0 \leq x_1 \leq 0.25, \\
  \tanh(20(x_1 - 0.5)) & 0.25 \leq x_1 \leq 0.75, \\
  \tanh(-20(x_1 - 1)) & 0.75 \leq x_1 \leq 1.0.
\end{cases}
\] (8.11)

The transported scalar field develops spurious oscillations at the regions of high gradients in the velocity field as shown in Figure 8.4 (left
Figure 8.2: $L_2$-error for the Taylor-Green vortices using linear $Z_0$ (solid lines) and cubic $Z_1$ (dashed lines) volume elements. $Z_0$, $Z_1$ and $Z_2$ interpolations are compared.

Figure 8.3: Vector field of the Taylor-Green vortices (8.9) (left) and the scalar initial condition (8.11) (right).
8.1 Two-dimensional Scalar transport

These oscillations are due to interpolation errors produced in those regions where the density of mesh points is not enough to resolve the rapid changes in the scalar field. In this case, the oscillations in the solution come from the low resolution of the scalar field and not from rapid changes in the velocity field. The velocity field is smooth during this simulation. Therefore, smooth velocity fields are not enough to obtain smooth solutions (as sometimes is argued in coarse grid simulations). It is necessary to filter the flow variables to avoid the generation of numerical errors in undersampled simulations.

8.1.3 Z-spline filters

We have seen that the numerical errors from the interpolations produce strong oscillations of the fields in those regions where the fields are undersampled, i.e., where the fields have large variations of their gradients. The elimination of these oscillations in the fields is possible using a Z-spline filter. This filter is designed using the same Z-spline function as during the interpolation. The idea is to double the support of the Z-spline and use the odd and even sets of points to interpolate the function. The linearly weighted average of both values, the original value and the interpolated, is the filtered field. Repeating this process reduces the high frequencies dramatically and keeps the low frequencies practically untouched. Larger supports and averages must be used if the window of the filter should be widen. Cumulative filters are implemented to allow better control over the filtering operation. These types of algorithms are well-known to the computer graphics community, they refer to them as fairing. Normally, they use schemes based on approximations to the Laplacian operator.

The advantage of a Z-spline filter is that it keeps control over the space of functions that can be represented with that particular Z-spline function. The Z-spline filter smears out all the frequency components that cannot be represented by the Z-spline over the given discrete mesh. The Z-spline filters are highly accurate and a particular advantage is that the $Z_1$ and $Z_2$ filters are conservative and
Figure 8.4: Contours of the scalar field (8.11) transported by the velocity field (8.9) using the FSL numerical method with $Z_1$ volume elements and $Z_2$ interpolation. The results are shown from top to bottom after 250, 500 and 1000 time steps with $\Delta t = 1/1600$ and $128 \times 128$ cells. The plots show the results using the FSL method (left) and the improved results using a Z-spline filter (right).
8.2 Burgers’ equation

Burgers’ equation is the simplest non-linear model for fluid motion. The one-dimensional Burgers’ equation is

$$\frac{Du(x, t)}{Dt} = 0, \quad (8.12)$$

where \( u(x, t) = \frac{dx}{dt} \) is the velocity field.

This equation is solved using the FSL with linear volume elements and cubic interpolation in the domain \( x \in [0, 1] \) and with the initial condition

$$u(x, 0) = \sin(2\pi x). \quad (8.13)$$

In this example the interesting feature is the presence of discontinuities in the derivative of the solution. In this case the solution is not band-limited and therefore creates problems with the interpolation near the discontinuities.

We show that the FSL can handle discontinuities. In order to do so, a model for the discontinuities is adopted. A proper definition of discontinuity is implemented in the code and once the discontinuity is detected, the domain should be divided using the point of discontinuity and one-sided Z-splines should be used for the interpolation around the discontinuity.

The model for discontinuities adopted in this work is such that a sufficiently large change in the gradient of the field defines a discon-
Figure 8.5: Numerical solution to the one-dimensional Burgers’ equation with sinusoidal initial condition, mesh spacing $\Delta x = 0.01$ and time step $\Delta t = 0.001$. The solution is shown as a function of time (top) and after 300 time steps (bottom) using the FSL with cubic $Z_1$ interpolation (left) and with the model for discontinuities (right).

The results for the simulation of the one-dimensional Burgers’ equation are shown in Figure 8.5. The plots on the left show the FSL with cubic interpolation. The numerical errors around the discontinuity are due to the lack of resolution. The plots on the right show the solution using the model for discontinuities and one-sided interpolations around the discontinuity.

It is not the same but quite close to the idea of using limiters in finite volume advection schemes (see [32]). The model used in the code for the Burgers’ equation detects a discontinuity when

$$\frac{|u_{i+1} - 2u_i + u_{i-1}|}{\max(|u_{i+1} - u_i|, |u_i - u_{i-1}|)} > 0.25.$$  \hspace{1cm} (8.14)

This is the relative change in the slope of the function $u$. 
8.3 Shallow-water equations

The shallow-water equations

\[
\frac{D}{Dt} h = -h \frac{\partial u_i}{\partial x_i}, \tag{8.15}
\]

\[
\frac{D}{Dt} u_i = - \frac{\partial h}{\partial x_i}, \tag{8.16}
\]

for \( i = 1:2 \), are solved in the bidimensional doubly periodic domain \([0,1] \times [0,1]\).

The initial condition is a Gaussian perturbation in the water surface

\[
h(x_1, x_2, 0) = 1.0 + 0.2 \exp\left(-\frac{x_1^2 + x_2^2}{0.005}\right), \tag{8.17}
\]

and initial constant velocities \( u_1 = u_2 = 1/4 \). The shallow-water wave in the periodic domain is simulated using bicubic Z-spline volume elements. Figure 8.6 shows a comparison between the wave patterns obtained using the FSL with \( Z_0, Z_1 \) and \( Z_2 \) interpolations.

8.4 Inviscid Navier-Stokes equations

A simple model of the inviscid Navier-Stokes equations is given by the equations

\[
\frac{D}{Dt} \rho = -\rho \frac{\partial u_i}{\partial x_i}, \tag{8.18}
\]

\[
\frac{D}{Dt} u_i = - \frac{\partial p}{\partial x_i}, \tag{8.19}
\]

\[
\frac{D}{Dt} e = -p \frac{\partial u_i}{\partial x_i}, \tag{8.20}
\]

for \( i = 1:2 \), where the internal energy is in the role of the temperature and the equation of state is simplified to \( p = e\rho \).
Figure 8.6: Waves in a doubly periodic domain using the FSL with linear $Z_0$ (top), cubic $Z_1$ (middle) and quintic $Z_2$ bidimensional interpolations (bottom). The left column shows contours of the altitude for shallow-water waves (21 contours from 0.99 to 1.03). The right column shows contours of the density for inviscid Navier-Stokes waves (21 contours from 1 to 1.06). The results are obtained after 45 time steps with $\Delta t = 0.01$ and $64 \times 64$ cells.
The equations are solved numerically using the FSL over a doubly periodic domain \([0, 1] \times [0, 1]\). The initial condition is the same Gaussian perturbation as in the case of the shallow-water waves but here the perturbation is given in the density and pressure fields. The initial internal energy is constant and equal to one. The initial velocity field is also the same as in the case of the shallow-water equations. The solution of this problem has an initial solitary wave coming out of the perturbation and an equilibrium is obtained for the remaining part of the perturbation. Figure 8.6 shows a comparison of the results obtained using \(Z_0\), \(Z_1\) and \(Z_2\) interpolations.

This problem is interesting as a direct introduction for the simulation of jets and flames. One problem to solve before being ready for the simulations of the flames is the open field boundary conditions.

### 8.4.1 Open field boundary condition

The open field boundary condition is obtained using the following procedure:

1. Reflect the fields across the open boundary.
2. Filter the solution in a small region (six cells) around the open boundary.

The same Z-spline filter used for the elimination of numerical oscillations is used at the boundary to obtain the non-reflective boundary condition. This type of boundary conditions has been frequently used since the work of Berenger [6] in 1994. He has shown that this is a highly effective absorbing boundary condition (ABC). His algorithm is designated the *perfectly matched layer* or PML.

Figure 8.7 shows the result for the Navier-Stokes wave. The open field boundary conditions are found to eliminate all the outgoing waves and reflections as expected. The use of Z-spline filters to obtain a non-reflecting boundary is a significant improvement compared to
Figure 8.7: Density waves in an open field using the FSL with cubic $Z_1$ interpolation. The conditions of the simulation and the plot are the same as for the doubly periodic Navier-Stokes waves of the previous section. The figure shows the initial acoustic wave leaving the domain without reflections. The snapshots are taken after 5 (top-left), 25 (top-right), 35 (bottom-left) and 70 time steps (bottom-right).
other ABC’s where dissipative schemes are used after the reflection of the fields. The Z-spline filters produce very low dissipation of the supergrid structures.
Chapter 9

Simulation of Turbulent and Reacting Flow

We show preliminary results for the large-eddy simulation of a piloted partially premixed methane/air diffusion flame (called Sandia flame D). The simulation is based on the governing equations and the turbulence and combustion models described in Part I (Chapters 1 to 5). The numerical method used for the computation of an approximate solution of such models is the Z-spline forward semi-Lagrangian (FSL) scheme described in Part II (Chapters 6 and 7).

9.1 Settings for the computation

We are using a mesh in spherical polar coordinates \((r, \theta, \phi)\) with a resolution of \(179 \times 119 \times 48\) cells. The mesh spacings are \(\Delta r = 1/1250, \Delta \theta = 1/1000\) and \(\Delta \phi = 2\pi/48\). The minimum values for the coordinates are \(r_0 = 0.25, \theta_0 = \Delta \theta/2\) and \(\phi_0 = 0\). The mesh is shown in Figure 9.1. The minimum value for \(\theta\) is chosen such that the nodes in the mesh are not placed at the polar axis.
We use the international metric system for the units of the variables during the simulations.

The diameter of the fuel jet in the simulations is $D = 7.2$ mm. The computational domain size compared to $D$ has, approximately, a basis radius of $3.5D$ (inflow boundary), a polar height of $21D$ and a top radius of $7D$ (outflow boundary).

We kept the governing equations and the models in Cartesian coordinates and all the vectors are described by their Cartesian components. The necessary numerical derivatives on the right hand side of the equations are computed over the spherical coordinates mesh and transformed to Cartesian derivatives. One must be careful because the transformation from spherical to Cartesian coordinates is singular at the polar axis and therefore the pole should not be part of the mesh.

The interpolations are computed over the spherical coordinates space. The coordinates of the points in the mesh are updated in Cartesian coordinates and therefore it is necessary to transform the positions of the Lagrangian nodes to spherical coordinates before the interpolations.

Ghost cells are used to handle continuity at the polar axis and the open field boundary condition (Section 8.4.1) at the external boundaries. The ordering of the ghost cells at the pole is explained in Figure 9.2. After the mapping, the pole becomes a line. If a particle gets to the polar line, its azimuthal angle cannot be defined. We choose the last angle that the particle had before it got to the pole. In general, the source terms at the pole can cause problems. For our simulations, the source terms are bounded at the pole because the interpolations with Z-splines using the points as in Figure 9.2 provide smoothing of the singular derivatives. An additional problem is that the high resolution in the azimuthal direction is a source of high frequencies in the fields. An azimuthal Z-spline filter is used to control those frequencies.

At the inflow boundary, all the required mean quantities are specified
Figure 9.1: Spherical polar coordinates mesh used for the simulation of the cold jet and flame.
according to the experimental data presented in [2] for the fuel jet, pilot and co-flowing air stream. The inflow condition is sketched in Figure 9.3. The fluctuations of the velocity field are included via velocity vectors with magnitude corresponding to the experimental fluctuation amplitude presented in [2] and random direction.

The bulk velocity of the jet is 67 m/s. Therefore for a jet of air the Reynolds number is \( Re = 22400 \). A CFL number equal to 2 is used to fix the time step such that \( \Delta t = 2 \min (\Delta x) / \max ||u|| \), where \( \min (\Delta x) \) is the minimum grid spacing in the spherical coordinates mesh.

The open field boundary condition is used at all the external boundaries. This condition is also necessary at the inflow boundary to avoid reflections. The Z-spline filter used for this boundary condition is iterated 12 times with a linear weight of 0.9 favorable to the unfiltered fields.

A single Z-spline Favre filter of one iteration and weight parameter of 0.985 favorable to the unfiltered fields is used over the whole computational domain to reduce numerical oscillations due to the mixing processes.

The code is written in Fortran 95 and run in an Intel Pentium 4 processor of 3.6 GHz where every time step takes around one minute of
9.2 Simulation of a Cold jet

Before the simulation of the flame we test our code with the simulation of the jet composed only by air.

The mixture fraction $Z$ and the progress variable $c$ are not computed.
9.3 Simulation of a Sandia flame D

The Sandia flame D is a 25/75% methane/air mixture. The fuel jet, the pilot and the air are represented at the inlet by three different values for the mixture fraction \( Z \): 0.1563 in the fuel stream, 0.04 in the pilot stream and 0.0 in the air co-flow. The reaction progress variable \( c \) is initialized as 0.0 for the fuel stream and 1.0 for the pilot.
Figure 9.4: Subgrid kinetic energy of the cold jet showing the turbulent structure at $y = 0$ and $t = 0.01$ s.
Figure 9.5: Velocity magnitude of the cold jet at \( y = 0 \) and \( t = 0.01 \) s.
Figure 9.6: Temperature field of the Sandia flame D at $y = 0$ and $t = 0.002$ s.
Figure 9.7: Mixture fraction $Z$ field of the Sandia flame D at $y = 0$ and $t = 0.002$ s.
Figure 9.8: Reaction progress variable $c$ of the Sandia flame D at $y = 0$ and $t = 0.002$ s.
Figure 9.9: Averaged velocity magnitude of the Sandia flame D along the centerline. Lines (simulation), symbols (experiment).

stream and the air co-flow.

The unsteady flamelet software library produced by the FLATRA code [20] was extended to provide us with the values of the temperature $T$, the number of molecules per unit volume $n$, the local fuel mass fraction $Y_F$ and the molecular diffusion coefficients $D_Z$, $D_c$ and $D_\chi$ as a function of the mixture fraction $Z$, its scalar dissipation rate $\chi$, the reaction progress variable $c$ and the local residence time $\tau$. Linear interpolations are used to get intermediate values from the data base.

The values of the individual mass fractions can be obtained from the same four scalars at any desired time step, but they are not needed during the computation itself.

The initial condition for the fluid variables is given by the last fields produced by the cold jet simulation. The initial condition of the mixture fraction $Z$ and reaction progress variable $c$ is an extension of
9.3 Simulation of a Sandia flame D

![Graphs showing temperature and normalized mixture fraction along the centerline.]

Figure 9.10: Mean values of scalars along the centerline. Lines (simulation), symbols (experiment).
Figure 9.11: Mean values of scalars along the centerline. Lines (simulation), symbols (experiment).
9.3 Simulation of a Sandia flame D

the inlet condition along the polar axis.

One of the purposes of this work was not achieved, namely the computation of thermoacoustics. First, we coupled the fluid and combustion models using a source term in the energy equation (1.3) that added the internal energy corresponding to the temperature field obtained from the combustion tables. Then, we were forced to uncouple the fluid model and the ideal gas equation of state from the temperature field provided in the combustion model. An acoustic instability appears probably because in principle the combustion model was not designed to handle thermoacoustic effects. We would need a pressure dependent heat release model for that aim.

The original model used in the work of Baykal [3] is for an incompressible fluid and without energy equation. Therefore, our compressible fluid model is a step forward toward the inclusion of thermoacoustic effects. The high-order and truly multi-dimensional LES compressible fluid model used here provides the turbulent field necessary for a more realistic transport of the scalars.

In a recent publication, Schmitt et al. [43], conclude that in order to obtain realistic predictions of pollutant formation and pressure oscillation levels in flame simulations, not only an accurate chemical scheme and realistic heat transfer models are required but also a proper description of the thermoacoustics. The study of the acoustic instability when our combustion model is fully coupled with the energy equation and the simulation of thermoacoustic effects are objectives of future research.

Now, we show the results for the uncoupled simulations where the temperature of the flame is not included in the energy equation but it is provided directly by the combustion model. Figures 9.6-9.8 show the contour plots of the Temperature, the mixture fraction and the reaction progress variable for \( y = 0 \) and \( t = 0.002s \). The plot of the Temperature shows that the flame has ignited.

One-dimensional plots for the mean values of the scalars along the polar axis are obtained averaging the results over one flow-through
time, that is, the time a particle in the polar axis remains in the flow. They are presented in Figures 9.9-9.11. The results show relatively good agreement with experimental results [2]. The overprediction of fuel, CH$_4$, may cause the underprediction of products like CO and CO$_2$.

The results reported here are just preliminary. Results comparable to those reported in the literature require computational domains at least four times larger in the polar axis direction and at least two orders of magnitude longer averaging time periods.
Part V

Conclusion
We have created software for the high-resolution simulation of subsonic flames.

We were able to design a non-diffusive and conservative transport method based on the forward Semi-Lagrangian numerical method. The main feature of our transport method is the use of high-order Z-splines as basis functions for the interpolations. Only the first two Z-splines were known before this work.

The Z-splines are convolution Hermite splines that preserve the statistical moments of the data after the interpolation. We use the two times continuously differentiable $Z_2$ spline that preserves up to the fifth moment of the data. Numerical diffusion and dispersion are related to the preservation of moments during the interpolations and therefore our $Z_2$-spline FSL numerical method produces practically zero numerical diffusion and low dispersion.

The Z-spline FSL transport method is simple to code for high-orders and straightforward for complicated systems of transport equations. Eight coupled transport equations were solved for the LES of a Sandia flame D.

The Z-splines allowed us to provide simple answers to well known problems in these types of simulations, like the non-reflective boundary condition and the pole singularity.

We have used previously tested models for the turbulence and the chemistry to avoid the need to validate them. Validations of the models are reported in the literature for domains of at least 80 jet-diameters long. A parallel version of the code has been developed for future computations over such computational domains.

The preliminary results presented for the Sandia flame D are in relative good agreement with the experimental results but we were not able to compute thermoacoustic effects. An acoustic instability that needs further study was encountered. Therefore in our model, the temperature in the combustion model is decoupled from the energy equation of the fluid.
Finally, there are some concepts that we have explored but not implemented in the code. Future improvements must allow multi-dimensional discontinuities in the fields, general meshes for complex geometries and a recursive formula for the Z-spline interpolations.
Bibliography


Curriculum Vitae

Julián Tercero Becerra Sagredo

April 28, 1973  Born in Hermosillo Sonora, México
1990 – 1995  Physics studies, UNAM Mexico City
1995 – 1996  Research assistant, CIE UNAM Temixco
Sept. 1996  Title of Physicist with honors
1996 – 1998  Research assistant, UW Seattle
August 1998  MSc. Aeronautics & Astronautics and MSc. Applied Mathematics
1998 – 2006  Research and teaching assistant, IFD and SAM ETH Zürich