The JHU Turbulence Database Cluster

DOCUMENTATION OF DATABASE FUNCTIONS

1 Spatial differentiation inside database: equidistant grid

In this section, f denotes any one of the three components of velocity, magnetic field, or vector potential in the x, y and z directions (u_x, u_y or u_z; b_x, b_y or b_z; a_x, a_y or a_z), or pressure (p), depending on which function is called. Δx and Δy are the width of grid in x and y direction.

1.1 Options for GetVelocityGradient, GetMagneticFieldGradient, GetVectorPotentialGradient, and GetPressureGradient

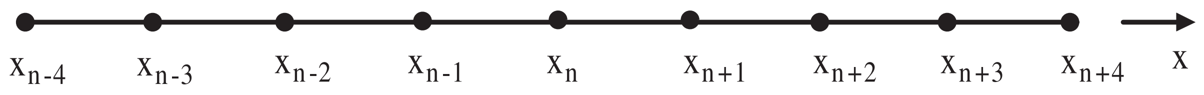

Figure 1: Illustration of data points along x direction. The same approach is used in the y and z directions.

FD4: 4th-order centered finite differencing

With the edge replication of 4 data-points on each side, this option can be spatially interpolated using 4th-order Lagrange Polynomial interpolation.

$$\left.\frac{df}{dx}\right|_{x_n} = \frac{2}{3\Delta x}[f(x_{n+1}) - f(x_{n-1})] - \frac{1}{12\Delta x}[f(x_{n+2}) - f(x_{n-2})] + o(\Delta x^4) \tag{1}$$

FD6: 6th-order centered finite differencing

$$\left.\frac{df}{dx}\right|_{x_n} = \frac{3}{4\Delta x}[f(x_{n+1}) - f(x_{n-1})] - \frac{3}{20\Delta x}[f(x_{n+2}) - f(x_{n-2})] + \frac{1}{60\Delta x}[f(x_{n+3}) - f(x_{n-3})] + o(\Delta x^6) \tag{2}$$

FD8: 8th-order centered finite differencing

With the edge replication of 4 data-points on each side, this is the highest-order finite difference option available.

$$\begin{aligned}\left.\frac{df}{dx}\right|_{x_n} = & \frac{4}{5\Delta x}[f(x_{n+1}) - f(x_{n-1})] - \frac{1}{5\Delta x}[f(x_{n+2}) - f(x_{n-2})] \\ & + \frac{4}{105\Delta x}[f(x_{n+3}) - f(x_{n-3})] - \frac{1}{280\Delta x}[f(x_{n+4}) - f(x_{n-4})] \\ & + o(\Delta x^8) \end{aligned} \tag{3}$$

1.2 Options for GetVelocityLaplacian, GetMagneticFieldLaplacian, GetVectorPotentialLaplacian, GetVelocityHessian, GetMagneticFieldHessian, GetVectorPotentialHessian, and GetPressureHessian

In this section, second derivatives finite difference evaluations are shown. The expressions are given for derivatives along single directions in terms of the x-direction, and mixed derivatives are illustrated on the x-y plane. The same approach is used in the y and z directions, as well as in the x-z and y-z planes for the other mixed derivatives.

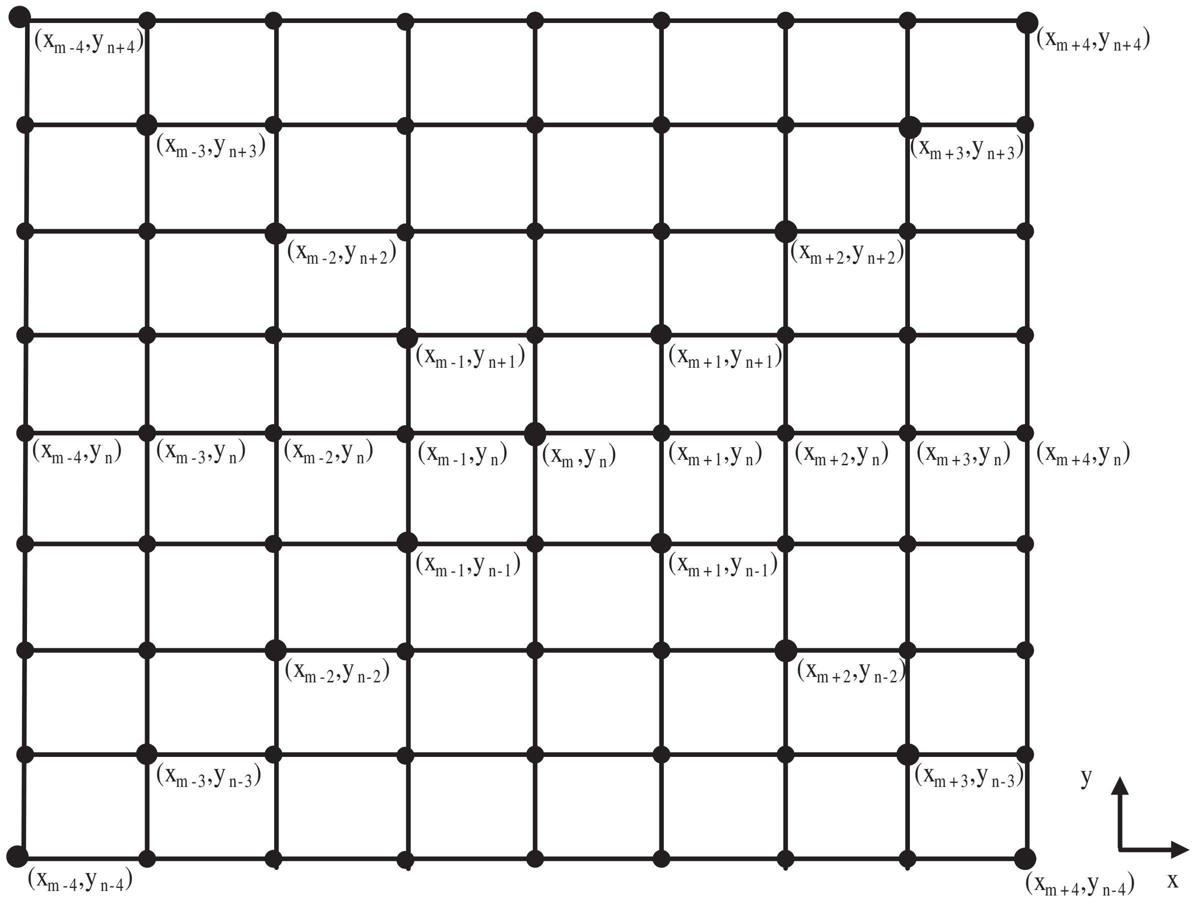

Figure 2: Illustration of data points on $x-y$ plane. The same approach is used in the $x-z$ and $y-z$ planes.

FD4: 4th-order centered finite differencing (can be spatially interpolated using 4th-order Lagrange Polynomial interpolation

$$\begin{aligned}\left.\frac{d^2f}{dx^2}\right|_{(x_m,y_n)} &= \frac{4}{3\Delta x^2}[f(x_{m+1},y_n)+f(x_{m-1},y_n)-2f(x_m,y_n)]\\ &-\frac{1}{12\Delta x^2}[f(x_{m+2},y_n)+f(x_{m-2},y_n)-2f(x_m,y_n)]\\ &+o(\Delta x^4)\end{aligned} \tag{4}$$

$$\begin{aligned}\left.\frac{d^2f}{dxdy}\right|_{(x_m,y_n)} &= \frac{1}{3\Delta x\Delta y}[f(x_{m+1},y_{n+1})+f(x_{m-1},y_{n-1})\\ &-f(x_{m+1},y_{n-1})-f(x_{m-1},y_{n+1})]\\ &-\frac{1}{48\Delta x\Delta y}[f(x_{m+2},y_{n+2})+f(x_{m-2},y_{n-2})\\ &-f(x_{m+2},y_{n-2})-f(x_{m-2},y_{n+2})]\\ &+o(\Delta x^4)\end{aligned} \tag{5}$$

FD6: 6th-order centered finite differencing

$$\begin{aligned}\left.\frac{d^2f}{dx^2}\right|_{(x_m,y_n)} &= \frac{3}{2\Delta x^2}[f(x_{m+1},y_n)+f(x_{m-1},y_n)-2f(x_m,y_n)]\\ &-\frac{3}{20\Delta x^2}[f(x_{m+2},y_n)+f(x_{m-2},y_n)-2f(x_m,y_n)]\\ &+\frac{1}{90\Delta x^2}[f(x_{m+3},y_n)+f(x_{m-3},y_n)-2f(x_m,y_n)]\\ &+o(\Delta x^6)\end{aligned} \tag{6}$$

$$\begin{aligned}\left.\frac{d^2f}{dxdy}\right|_{(x_m,y_n)} &= \frac{3}{8\Delta x\Delta y}[f(x_{m+1},y_{n+1})+f(x_{m-1},y_{n-1})\\ &-f(x_{m+1},y_{n-1})-f(x_{m-1},y_{n+1})]\\ &-\frac{3}{80\Delta x\Delta y}[f(x_{m+2},y_{n+2})+f(x_{m-2},y_{n-2})\\ &-f(x_{m+2},y_{n-2})-f(x_{m-2},y_{n+2})]\\ &+\frac{1}{360\Delta x\Delta y}[f(x_{m+3},y_{n+3})+f(x_{m-3},y_{n-3})\\ &-f(x_{m+3},y_{n-3})-f(x_{m-3},y_{n+3})]\\ &+o(\Delta x^6)\end{aligned} \tag{7}$$

FD8: 8th-order centered finite differencing

$$\begin{aligned}\left.\frac{d^2f}{dx^2}\right|_{(x_m,y_n)} &= \frac{792}{591\Delta x^2}[f(x_{m+1},y_n)+f(x_{m-1},y_n)-2f(x_m,y_n)]\\ &-\frac{207}{2955\Delta x^2}[f(x_{m+2},y_n)+f(x_{m-2},y_n)-2f(x_m,y_n)]\\ &-\frac{104}{8865\Delta x^2}[f(x_{m+3},y_n)+f(x_{m-3},y_n)-2f(x_m,y_n)]\\ &+\frac{9}{3152\Delta x^2}[f(x_{m+4},y_n)+f(x_{m-4},y_n)-2f(x_m,y_n)]\\ &+o(\Delta x^8)\end{aligned} \tag{8}$$

$$\begin{aligned}\left.\frac{d^2f}{dxdy}\right|_{(x_m,y_n)} &= \frac{14}{35\Delta x\Delta y}[f(x_{m+1},y_{n+1})+f(x_{m-1},y_{n-1})\\ &-f(x_{m+1},y_{n-1})-f(x_{m-1},y_{n+1})]\\ &-\frac{1}{20\Delta x\Delta y}[f(x_{m+2},y_{n+2})+f(x_{m-2},y_{n-2})\\ &-f(x_{m+2},y_{n-2})-f(x_{m-2},y_{n+2})]\\ &+\frac{2}{315\Delta x\Delta y}[f(x_{m+3},y_{n+3})+f(x_{m-3},y_{n-3})\\ &-f(x_{m+3},y_{n-3})-f(x_{m-3},y_{n+3})]\\ &-\frac{1}{2240\Delta x\Delta y}[f(x_{m+4},y_{n+4})+f(x_{m-4},y_{n-4})\\ &-f(x_{m+4},y_{n-4})-f(x_{m-4},y_{n+4})]\\ &+o(\Delta x^8)\end{aligned} \tag{9}$$

2 Spatial interpolation inside database: equidistant grid

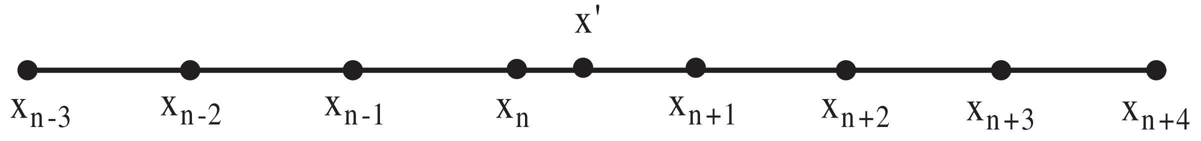

Figure 3: Illustration of Lagrangian interpolation.

2.1 Interpolation Options for GetVelocity, GetMagneticField, GetVectorPotential, GetPressure and GetVelocityAndPressure

In this section, f denotes any one of the three components of velocity, u_x, u_y or u_z, magnetic field b_x, b_y or b_z, vector potential a_x, a_y or a_z, or pressure, p, depending on which function is called. Δx, Δy and Δz are the width of grid in x, y and z direction. $\mathbf{x}' = (x', y', z')$.

NoSInt: No spatial interpolation

In this case, the value at the datapoint closest to each coordinate value is returned, rounding up or down in each direction.

$$f(\mathbf{x}') = f(x_n, y_p, z_q) \tag{10}$$

where $n = int(\frac{x'}{\Delta x} + \frac{1}{2})$, $p = int(\frac{y'}{\Delta y} + \frac{1}{2})$, $q = int(\frac{z'}{\Delta z} + \frac{1}{2})$.

Lag4: 4th-order Lagrange Polynomial interpolation

In this case, 4th-order Lagrange Polynomial interpolation is done along each spatial direction.

$$\begin{aligned} f(\mathbf{x}') &= \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} f(x_{n-2+i}, y_{p-2+j}, z_{q-2+k}) \\ &\cdot l_x^{n-2+i}(x') \cdot l_y^{p-2+j}(y') \cdot l_z^{q-2+k}(z') \end{aligned} \tag{11}$$

$$l_\theta^i(\theta') = \frac{\prod\limits_{j=n-1,j\neq i}^{n+2} (\theta' - \theta_j)}{\prod\limits_{j=n-1,j\neq i}^{n+2} (\theta_i - \theta_j)} \tag{12}$$

where θ can be x, y, or z.

Lag6: 6th-order Lagrange Polynomial interpolation

In this case, 6th-order Lagrange Polynomial interpolation is done along each spatial direction.

$$\begin{aligned} f(\mathbf{x}') &= \sum_{i=1}^{6}\sum_{j=1}^{6}\sum_{k=1}^{6} f(x_{n-3+i}, y_{p-3+j}, z_{q-3+k}) \\ &\cdot l_x^{n-3+i}(x') \cdot l_y^{p-3+j}(y') \cdot l_z^{q-3+k}(z') \end{aligned} \tag{13}$$

$$l_\theta^i(\theta') = \frac{\prod\limits_{j=n-2,j\neq i}^{n+3} (\theta' - \theta_j)}{\prod\limits_{j=n-2,j\neq i}^{n+3} (\theta_i - \theta_j)} \tag{14}$$

where θ can be x, y, or z.

Lag8: 8th-order Lagrange Polynomial interpolation

In this case, 8th-order Lagrange Polynomial interpolation is done along each spatial direction.

$$\begin{aligned} f(\mathbf{x}') &= \sum_{i=1}^{8}\sum_{j=1}^{8}\sum_{k=1}^{8} f(x_{n-4+i}, y_{p-4+j}, z_{q-4+k}) \\ &\cdot l_x^{n-4+i}(x') \cdot l_y^{p-4+j}(y') \cdot l_z^{q-4+k}(z') \end{aligned} \tag{15}$$

$$l_\theta^i(\theta') = \frac{\prod_{j=n-3, j\neq i}^{n+4} (\theta' - \theta_j)}{\prod_{j=n-3, j\neq i}^{n+4} (\theta_i - \theta_j)} \tag{16}$$

where θ can be x, y, or z.

2.2 Interpolation Options for GetVelocityGradient, GetVelocityLaplacian, GetVelocityHessian, GetMagneticFieldGradient, GetVectorPotentialGradient, GetMagneticFieldLaplacian, GetVectorPotentialLaplacian, GetMagneticFieldHessian, GetVectorPotentialHessian, GetPressureGradient, and GetPressureHessian

In this section, f denotes gradients of velocity, magnetic field, vector potential, or pressure gradient, Laplacian of velocity, magnetic field, vector potential, or Hessian of velocity, magnetic field, vector potential or pressure, depending on which function is called.

FD4NoInt, FD6NoInt and FD8NoInt: No spatial interpolation

In this case, the value of the 4th, 6th, or 8th order finite-difference evaluation of the derivative at the datapoint closest to each coordinate value is returned, rounding up or down in each direction.

$$f(\mathbf{x}') = f(x_n, y_p, z_q) \tag{17}$$

where $n = int(\frac{x'}{\Delta x} + \frac{1}{2})$, $p = int(\frac{y'}{\Delta y} + \frac{1}{2})$, $q = int(\frac{z'}{\Delta z} + \frac{1}{2})$.

FD4Lag4: 4th-order Lagrange Polynomial interpolation of 4th-order finite diff.

In this case, the values of the 4th order finite-difference evaluation of the derivative at the data points are interpolated using 4th-order Lagrange Polynomials.

$$\begin{aligned} f(\mathbf{x}') = & \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} f(x_{n-2+i}, y_{p-2+j}, z_{q-2+k}) \\ & \cdot l_x^{n-2+i}(x') \cdot l_y^{p-2+j}(y') \cdot l_z^{q-2+k}(z') \end{aligned} \tag{18}$$

$$l_\theta^i(\theta') = \frac{\prod_{j=n-1, j\neq i}^{n+2} (\theta' - \theta_j)}{\prod_{j=n-1, j\neq i}^{n+2} (\theta_i - \theta_j)} \tag{19}$$

where θ can be x, y, or z.

3 Spatial interpolation inside the database: non-uniform grid

Spatial interpolation for domains with non-uninform grid spacing (e.g. the channel flow domain) is applied using multivariate polynomial interpolation of the barycentric Lagrange form from

Ref. [1]. Using this approach, we are interested in interpolating the field f at point $\boldsymbol{x}'$. The point $\boldsymbol{x}'$ is known to exist within the grid cell at location (x_m, y_n, z_p) where (m, n, p) are the cell indices. The cell indices are obtained for the x and z directions, which are uniformly distributed, according to

$$\begin{aligned} m &= \text{floor}(x'/dx) \\ p &= \text{floor}(z'/dz)\ . \end{aligned} \tag{20}$$

In the y direction the grid is formed by Marsden-Schoenberg collocation points which are not uniformly distributed. Along this direction we perform a search to obtain n such that

$$\begin{aligned} y_n \le y' < y_{n+1} &\quad \text{if } y' \le 0 \\ y_{n-1} < y' \le y_n &\quad \text{if } y' > 0 \end{aligned} \tag{21}$$

The cell indices are also assured to obey the following:

$$\begin{aligned} 0 \le m \le N_x - 2 & \\ 0 \le n \le N_y/2 - 1 &\quad \text{if } y' \le 0 \\ N_y/2 \le n \le N_y - 1 &\quad \text{if } y' > 0 \\ 0 \le p \le N_z - 2 & \end{aligned} \tag{22}$$

where N_x, N_y, and N_z are the number of grid points along the x, y, and z directions, respectively. In the case that $x' = x_{N_x - 1}$ the cell index set to be $m = N_x - 2$; likewise for the z direction.

The interpolation stencil also contains q points in each direction for an order q interpolant (with degree $q-1$). The resulting interpolated value is expressed as:

$$f(\boldsymbol{x}') = \sum_{i=i_s}^{i_e} \sum_{j=j_s}^{j_e} \sum_{k=k_s}^{k_e} l_x^i(x') l_y^j(y') l_z^k(z') f(x_i, y_j, z_k) \tag{23}$$

where the starting and ending indices are given as

$$\begin{aligned} i_s &= m - \text{ceil}(q/2) + 1 \\ i_e &= i_s + q - 1 \\ j_s &= \begin{cases} n - \text{ceil}(q/2) + 1 + j_o & \text{if } n \le N_y/2 - 1 \\ n - \text{floor}(q/2) + j_o & \text{otherwise} \end{cases} \\ j_e &= j_s + q - 1 \\ k_s &= p - \text{ceil}(q/2) + 1 \\ k_e &= k_s + q - 1 \end{aligned} \tag{24}$$

and j_o is the index offset for the y direction depending on the distance from the top and bottom walls. The ceil() function ensures that stencil remains symmetric about the interpolation point when q is odd. In the case for j_s, the separate treatments for the top and bottom halves of the channel is done to ensure that the one-sided stencils remain symmetric with respect to the channel center. The value for j_o may be evaluated based upon the y cell index and the interpolation order as

$$j_o = \begin{cases} \max(\text{ceil}(q/2) - n - 1, 0) & \text{if } n \le N_y/2 - 1 \\ \min(N_y - n - \text{ceil}(q/2), 0) & \text{otherwise} \end{cases}\ . \tag{25}$$

The interpolation weights, l_x, l_y, and l_z, are given as

$$l_\theta^\xi(\theta') = \frac{\frac{w_\xi}{\theta' - \theta_\xi}}{\sum_{\eta=\xi_s}^{\xi_e} \frac{w_\eta}{\theta' - \theta_\eta}} \tag{26}$$

where θ may either be x, y, or z. The barycentric weights, w_ξ, in (26) are given as

$$w_\xi = \frac{1}{\prod_{\eta=\xi_s, \eta\neq\xi}^{\xi_e} \theta_\xi - \theta_\eta} \tag{27}$$

The weights are computed by applying a recursive update procedure as in Ref.[1]. A slightly modified version of the algorithm in Ref. [1] is given below:

```
for $\xi = \xi_s$ to $\xi_e$ do
  $w_\xi = 1$
end for
for $\xi = \xi_s + 1$ to $\xi_e$ do
  for $\eta = \xi_s$ to $\xi - 1$ do
    $w_\eta = (\theta_\eta - \theta_\xi) w_\eta$
    $w_\xi = (\theta_\xi - \theta_\eta) w_\xi$
  end for
end for
for $\xi = \xi_s$ to $\xi_e$ do
  $w_\xi = 1/w_\xi$
end for
```

To account for the periodic domain along the x and z directions we adjust the i and k indices when referencing f in (23) such that

$$f(\boldsymbol{x}') = \sum_{i=i_s}^{i_e} \sum_{j=j_s}^{j_e} \sum_{k=k_s}^{k_e} l_x^i(x') l_y^j(y') l_z^k(z') f(x_{i\%Nx}, y_j, z_{k\%Nz}) \tag{28}$$

and % is the modulus operator. The indices for the interpolation coefficients remain the same, however, we use the fact that the grid points are uniformly spaced such that (26) becomes

$$l_\theta^\xi(\theta') = \frac{\frac{w_\xi}{\theta' - \xi\Delta\theta}}{\sum_{\eta=\xi_s}^{\xi_e} \frac{w_\eta}{\theta' - \eta\Delta\theta}} \tag{29}$$

and similarly for the barycentric weights, (27) becomes

$$w_\xi = \frac{1}{\prod_{\eta=\xi_s, \eta\neq\xi}^{\xi_e} (\xi - \eta)\Delta\theta} \tag{30}$$

for the x and z directions. The computation of the barycentric weights for the x and z directions are carried out once (for a given interpolation order) for all grid points using (30); for the y direction the barycentric weights are computed for each point using (27).

4 Spatial differentiation inside the database: non-uniform grid

Spatial differentiation for grids with non-uniform spacing is performed using the barycentric method of the interpolating polynomial. In one dimension (assuming the x direction; the same applies for the y and z directions), the interpolant for the field f is given as

$$f(x) = \sum_{j=i_s}^{i_e} l_x^j(x) f(x_j) \ . \tag{31}$$

It follows that the r^{th} derivative may be computed as

$$\frac{d^r f}{dx^r}(x) = \sum_{j=i_s}^{i_e} \frac{d^r l_x^j}{dx^r}(x) f(x_j) \ . \tag{32}$$

Within the database we compute the derivatives at the grid sites for the $FD4NoInt$, $FD6NoInt$, and $FD8NoInt$ differencing methods where no interpolation is performed. If a sample point is given that does not coincide with a grid point, the derivative at the *nearest* grid point is computed and returned. For the $FD4Lag4$ method we compute the derivatives with the $FD4NoInt$ method (at the grid sites) and then these data are interpolated to the interpolation point using the $Lag4$ interpolation method presented in §3.

For evaluating derivatives at the grid sites we follow the method presented in Ref. [1] such that

$$\frac{d^r f}{dx^r}(x_i) = \sum_{j=i_s}^{i_e} D_{x,ij}^{(r)} f(x_j) \ . \tag{33}$$

where $D_{x,ij}^{(r)} = \frac{d^r l_x^j}{dx^r}(x_i)$ is the differentiation matrix [1]. The differentiation matrices for $r = 1$ and $r = 2$ are given, respectively, as

$$D_{x,ij}^{(1)} = \frac{w_j/w_i}{x_i - x_j} \tag{34}$$

$$D_{x,ij}^{(2)} = -2\frac{w_j/w_i}{x_i - x_j}\left[\sum_{k\neq i} \frac{w_k/w_i}{x_i - x_k} + \frac{1}{x_i - x_j}\right] \tag{35}$$

for $i \neq j$ and

$$D_{x,jj}^{(r)} = -\sum_{i\neq j} D_{x,ji}^{(r)} \tag{36}$$

when $i = j$ for all $r > 0$. We note that in (35) and (36) fixes have been applied to the respective equations presented in Ref. [1], i.e., (9.4) and (9.5). As with the interpolation schemes, the grid point locations for the uniformly distributed directions are expressed as $\theta_\xi = \xi\Delta\theta$, where θ may either be x or z.

For second order mixed derivatives (such as for the pressure Hessian) we compute the derivatives at the grid sites within the respective plane. When computing the mixed partials along x and y we have

$$\frac{d^2 f}{dxy}(x_m, y_n) = \sum_{i=i_s}^{i_e} \sum_{j=j_s}^{j_e} D_{x,mi}^{(1)} D_{y,nj}^{(1)} f(x_i, y_j) \ . \tag{37}$$

Similar formulae exist for mixed partials along x and z, and y and z.

The differencing stencil size depends on the required order of the differencing method and the derivative order, r. In general, the resulting stencil size is determined as

$$q = \begin{cases} q' + r & \text{for non–symmetric grid distribution about evaluation point} \\ q' + r - (r+1)\%2 & \text{for symmetric grid distribution about evalutation point} \end{cases} \tag{38}$$

where q' is the order of the differencing method. For example, to obtain a 6^{th} order differencing method for the first derivative of f along the x, y, and z directions, a value of $q = 7$ is required. For the second derivative, the x or z directions require a value of $q = 7$ where the y direction requires $q = 8$ to acheive a 6^{th} order differencing method.

5 Temporal interpolation inside database

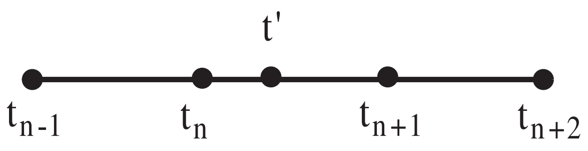

Figure 4: Illustration of data points for time.

In this section, f denotes velocity, magnetic field, vector potential, pressure, their gradient, Laplacian or Hessian, or force, depending on which function is called. Δt is the time increment between consecutive times stored in the database.

NoTInt: No temporal interpolation

In this case, the value at the datapoint closest to the time value is returned, rounding up or down.

$$f(t') \quad = \quad f(t_n) \tag{39}$$

where $n = int(\frac{t'}{\Delta t} + \frac{1}{2})$.

PCHIP: Cubic Hermite Interpolation in time

The value from the two nearest time points is interpolated at time t' using Cubic Hermite Interpolation Polynomial, with centered finite difference evaluation of the end-point time derivatives - i.e. a total of four temporal points are used.

$$f(t') = a + b(t' - t_n) + c(t' - t_n)^2 + d(t' - t_n)^2(t' - t_{n+1}) \tag{40}$$

where

$$a \quad = \quad f(t_n)$$

$$b \quad = \quad \frac{f(t_{n+1}) - f(t_{n-1})}{2\Delta t}$$

$$c = \frac{f(t_{n+1}) - 2f(t_n) + f(t_{n-1})}{2\Delta t^2}$$

$$d = \frac{-f(t_{n-1}) + 3f(t_n) - 3f(t_{n+1}) + f(t_{n+2})}{2\Delta t^3}$$

6 Spatial filtering inside database

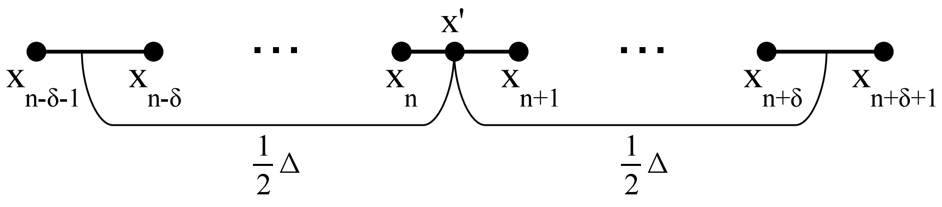

Figure 5: Illustration of data points for filtering.

In this section, f denotes any one of the three components of velocity, u_x, u_y or u_z, magnetic field b_x, b_y or b_z, vector potential a_x, a_y or a_z, or pressure, p, depending on the user supplied parameter for the GetBoxFilter and GetBoxFilterGradient functions called $field$. If the GetBoxFilterSGS function is called, f denotes any one of the six components of the sub-grid stress tensor for the respective field (e.g. u_x^2, u_y^2, u_z^2, u_xu_y, u_xu_z, u_yu_z). The valid values for the $field$ parameter are $velocity, magnetic, potential, pressure$. Δx is the resolution of the grid (it is equal in all directions). The user also supplies the desired filter width Δ, which is given as a floating point number and is required to be an odd multiple of the grid resolution (to eliminate the need for interpolating at the edges). $\mathbf{x}' = (x', y', z')$.

6.1 GetBoxFilter and GetBoxFilterSGS

The box filter around the target location and with the given filter width is computed as follows:

$$\overline{f(\mathbf{x}')} = \frac{1}{\Delta^3} \cdot \sum_{i=n-\delta}^{n+\delta} \sum_{j=p-\delta}^{p+\delta} \sum_{k=q-\delta}^{q+\delta} f(x_i, y_j, z_k), \tag{41}$$

where $n = int(\frac{x'}{\Delta x} + \frac{1}{2})$, $p = int(\frac{y'}{\Delta y} + \frac{1}{2})$, $q = int(\frac{z'}{\Delta z} + \frac{1}{2})$, and $\delta = int(\frac{1}{2}\frac{\Delta}{\Delta x})$.

6.2 GetBoxFilterGradient

In this case, the computation of the gradient is done by means of second order finite differencing of the filtered values. In addition to the $field$ and $filter\ width$ parameters described above the user can request a particular spacing for the finite differencing computation, $\Delta x'$. The filtered values computed as above are used to compute the filtered value of the gradient according to:

$$\overline{\frac{df}{dx}}\bigg|_{x_n} = \frac{1}{2\Delta x'}[\overline{f(x_{n+h})} - \overline{f(x_{n-h})}] + o(\Delta x'^2), \tag{42}$$

where $h = int(\frac{\Delta x'}{\Delta x})$.

7 Evaluating the applied force inside database using GetForce for hydrodynamic isotropic turbulence dataset ("isotropic1024")

Information about the forcing term $f_i(x, y, z, t)$ (force per unit mass, $i = x, y, z$) applied during the DNS has been stored inside the database and can be retrieved using the function GetForce. During DNS, an effective forcing is applied in Fourier space by rescaling low-k Fourier modes (with magnitudes $0.5 \le k \le 2.5$, $k = \sqrt{k_x^2 + k_y^2 + k_z^2}$) to maintain their kinetic energy to prescribed values consistent with a $-5/3$ spectrum. The forcing region is divided into two shells, $0.5 \le k \le 1.5$ and $1.5 < k \le 2.5$, and the spectrum is fixed at a value of 0.3 in shell $0.5 \le k \le 1.5$ shell and 0.13 in shell $1.5 < k \le 2.5$ shell (these values are obtained empirically so that the simulated spectrum is close to $k^{-5/3}$ at low k).

In order to represent the rescaling in terms of a forcing term, we interpreting the time-advancement in terms of a first-order time-advancement and write the discretized Navier-Stokes equation (NSE) in Fourier space as follows

$$\hat{u}_i^{n+1}(k_x, k_y, k_z) = \hat{u}_i^{n+}(k_x, k_y, k_z) + \hat{f}_i(k_x, k_y, k_z)dt \tag{43}$$

in which $\hat{u}_i^{n+} = \hat{u}_i^n + (\cdots)dt$ with $(\cdots)$ for terms on the right-hand side of NSE excluding the forcing term, and dt is the time-step of the DNS.

In the DNS, the rescaling induces a difference between $\hat{u}_i^{n+}$ and $\hat{u}_i^n$ in the wave-number range $0.5 \le k \le 2.5$ that is equivalent to a force-term defined in the two shells as follows

$$\hat{f}_i^n(k_x, k_y, k_z) = \frac{1}{dt}\left(\sqrt{\frac{0.3}{\sum_{0.5 \le k \le 1.5}[(\hat{u}_x^{n+})^2 + (\hat{u}_y^{n+})^2 + (\hat{u}_z^{n+})^2)]/2}} - 1\right)\, \hat{u}_i^{n+}(k_x, k_y, k_z) \tag{44}$$

for shell $0.5 \le k \le 1.5$ and

$$\hat{f}_i^n(k_x, k_y, k_z) = \frac{1}{dt}\left(\sqrt{\frac{0.13}{\sum_{1.5 \le k \le 2.5}[(\hat{u}_x^{n+})^2 + (\hat{u}_y^{n+})^2 + (\hat{u}_z^{n+})^2)]/2}} - 1\right)\, \hat{u}_i^{n+}(k_x, k_y, k_z) \tag{45}$$

for shell $1.5 < k \le 2.5$, where $\hat{u}_x$, $\hat{u}_y$, $\hat{u}_z$ denote the three velocity components in Fourier space and $k = \sqrt{k_x^2 + k_y^2 + k_z^2}$ is the magnitude of wavenumber vector $\mathbf{k}$. In this way, the energy in these shells $E(k = 1) = \sum_{0.5 \le k \le 1.5}(\hat{u}_x^2 + \hat{u}_y^2 + \hat{u}_z^2)/2$ and $E(k = 2) = \sum_{1.5 < k \le 2.5}(\hat{u}_x^2 + \hat{u}_y^2 + \hat{u}_z^2)/2$ is maintained at 0.3 and 0.13.

There exist in total of 80 discrete wave-number modes in these two shells. There are 20 modes for $k_x = 0$, 30 for $k_x > 0$, and another 30 for $k_x < 0$. In the database, the complex Fourier coefficients $\hat{f}_x$, $\hat{f}_y$, $\hat{f}_z$ corresponding to $k_x \ge 0$ (50 modes) are stored, the remaining 30 modes ($k_x < 0$) are the conjugates of the modes $k_x > 0$.

Using the GetForce function, force values at any prescribed position (x, y, z) are evaluated in the database from the Fourier forcing coefficients according to direct summation of the Fourier series according to

$$f_i(x, y, z, t_n) = \sum_{k_x, k_y, k_z} e^{i(k_x x + k_y y + k_z z)}\; \hat{f}_i^n(k_x, k_y, k_z) \tag{46}$$

where i can be x, y, and z. Values of $f_i(x, y, z, t)$ at arbitrary times t can be obtained by specifying PCHIP temporal interpolation.

8 Evaluating the applied force inside database using GetForce for MHD turbulence dataset ("mhd1024")

Using the GetForce function, force values at any prescribed position (x, y, z) are evaluated in the Taylor-Green prescribed force field according to

$$f_x(x, y, z, t_n) = 0.25 \sin(2x) \cos(2y) \cos(2z) \tag{47}$$

$$f_y(x, y, z, t_n) = -0.25 \cos(2x) \sin(2y) \cos(2z) \tag{48}$$

$$f_z(x, y, z, t_n) = 0. \tag{49}$$

9 Tracking fluid particles by GetPosition function

Getposition function tracks arrays of particles simultaneously and returns final particle locations at the end of the trajectory integration time. The function uses second order time Runge-Kutta integration.

Given particle locations $\mathbf{y}$ at a start time (t_{ST}), the function returns all the particle locations at a user defined end time (t_{ET}) with user-specified particle integration time-step (Δt_p). Forward tracking is accomplished by specifying $t_{ET} > t_{ST}$, whereas backward tracking is accomplished by specifying $t_{ET} < t_{ST}$. The time-step Δt_p's sign need not be specified to make the distinction between forward and backward tracking since inside the tracking integration, it is taken to be $\text{sign}[t_{ET} - t_{ST}]|\Delta t_p|$.

Particle tracking is accomplished by integrating between times t_{ST} and t_{ET} the equation

$$\frac{\partial \mathbf{x}^+(\mathbf{y}, t)}{\partial t} = \mathbf{u}^+(\mathbf{y}, t) \tag{50}$$

where $\mathbf{x}^+(\mathbf{y}, t)$ and $\mathbf{u}^+(\mathbf{y}, t)$ denotes the position and velocity at time t of the fluid particle originating from position $\mathbf{y}$ at initial time t_{ST} (superscript + represents Lagrangian quantities following the fluid particle). The Lagrangian velocity $\mathbf{u}^+(\mathbf{y}, t)$ is replaced by the Eulerian velocity from the database $\mathbf{u}(\mathbf{x}, t)$ where the particle is located, namely $\mathbf{u}^+(\mathbf{y}, t) = \mathbf{u}(\mathbf{x}^+(\mathbf{y}, t), t)$.

To advance the particle positions between two successive time instants t_m and $t_{m+1} (= t_m + \Delta t_p)$ the predictor step yields an estimate

$$\mathbf{x}^* = \mathbf{x}^+(\mathbf{y}, t_m) + \Delta t_p \; \mathbf{u}^+(\mathbf{y}, t_m). \tag{51}$$

The corrector step then gives the particle position at t_{m+1} as

$$\mathbf{x}^+(\mathbf{y}, t_{m+1}) = \mathbf{x}^+(\mathbf{y}, t_m) + \Delta t_p \; [\mathbf{u}^+(\mathbf{y}, t_m) + \mathbf{u}^+(\mathbf{x}^*, t_{n+1})]/2. \tag{52}$$

The integration proceeds until t_m reaches the user-specified final time t_{ES}. The last integration time-step is typically done using a smaller time-step so that the integration ends exactly at the specified t_{ES}. GetPosition then returns $\mathbf{x}^+(\mathbf{y}, t_{ES})$ for all particles that were at initial locations $\mathbf{y}$.

For this integration scheme, the time-stepping error is of order $(\Delta t_p)^3$ over one time step. In general, accurate spatial and time interpolations are crucial to obtain the fluid velocities while tracking particles along their trajectories. Spatial interpolation with various optional orders of accuracy can be specified by the user, see §2 above. Time interpolation is done by default using PHCIP (see §5 above).

References

[1] J.P. Berrut and L.N. Trefethen. Barycentric lagrange interpolation. *SIAM*, 46(3):501–517, 2004.