

The JHU Turbulence Database Cluster

DOCUMENTATION OF DATABASE FUNCTIONS

1 Spatial differentiation inside database

In this section, f denotes any one of the three components of velocity in the x , y and z directions (u_x , u_y or u_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 and GetPressureGradient

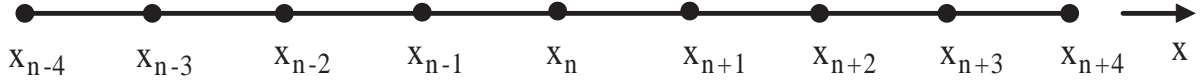


Figure 1: Illustration of data points along x direction. The same approach is used in the y and z directions.

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.

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

FD6: 6th-order centered finite differencing

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

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.

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

1.2 Options for GetVelocityLaplacian, GetVelocityHessian 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.

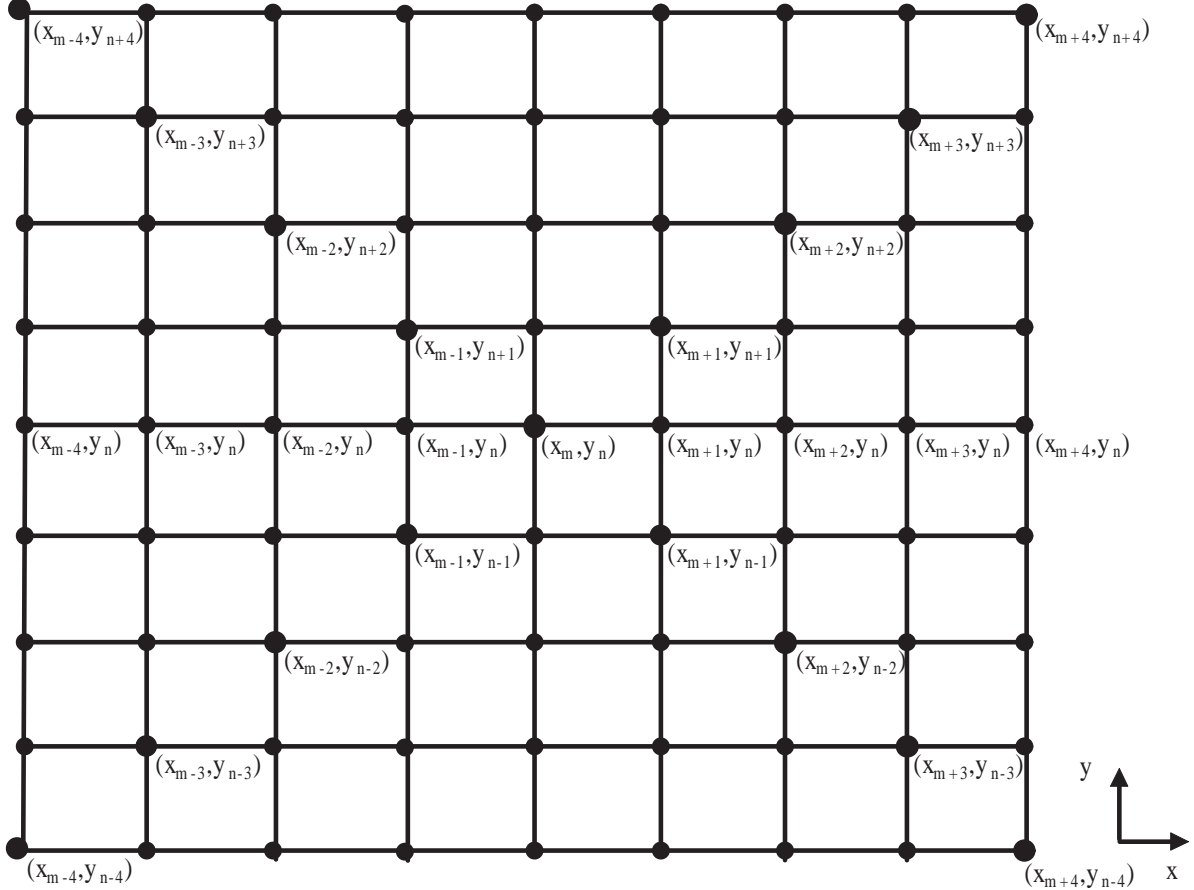


Figure 2: Illustration of data points on $x - y$ plane. The same approach is used in the $x - z$ and $y - z$ planes.

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

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

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

FD6: 6th-order centered finite differencing

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

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

FD8: 8th-order centered finite differencing

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

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

2 Spatial interpolation inside database

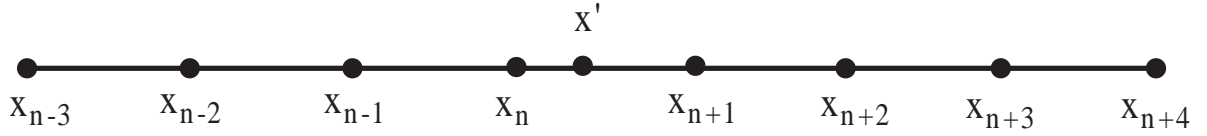


Figure 3: Illustration of Lagrangian interpolation.

2.1 Interpolation Options for GetVelocity, GetVelocityAndPressure, and GetForce

In this section, f denotes any one of the three components of velocity, u , v or w , or pressure, p , or force components f_x , f_y or f_z , 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 $n = \text{int}(\frac{x'}{\Delta x} + \frac{1}{2})$, $p = \text{int}(\frac{y'}{\Delta y} + \frac{1}{2})$, $q = \text{int}(\frac{z'}{\Delta z} + \frac{1}{2})$.

Lag4: 4th-order Lagrange Polynomial interpolation

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

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

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

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.

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

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

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.

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

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

where θ can be x, y, or z.

2.2 Interpolation Options for GetVelocityGradient, GetPressureGradient, GetVelocityLaplacian, GetVelocityHessian and GetPressureHessian

In this section, f denotes velocity or pressure gradient, velocity or pressure Hessian, or velocity Laplacian, 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.

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

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

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.

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

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

where θ can be x, y, or z.

3 Temporal interpolation inside database

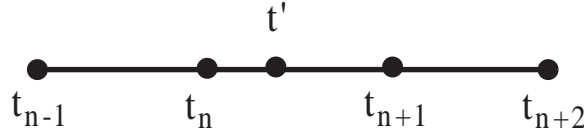


Figure 4: Illustration of data points for time.

In this section, f denotes velocity, pressure, force, velocity or pressure gradient, velocity or pressure Hessian, or velocity Laplacian, 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.

$$f(t') = f(t_n) \quad (20)$$

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

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.

$$f(t') = a + b(t' - t_n) + c(t' - t_n)^2 + d(t' - t_n)^3 \quad (21)$$

where

$$a = f(t_n)$$

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

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

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

4 Evaluating the applied force inside database using GetForce

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 $0.5 \leq k \leq 2.5$, $k = \sqrt{k_x^2 + k_y^2 + k_z^2}$) to maintain their kinetic energy to prescribed values consistent with a $-5/3$ spectrum. The forcing region is divided into two shells, $0.5 \leq k \leq 1.5$ and $1.5 < k \leq 2.5$, and the spectrum is fixed at a value of 0.3 in shell $0.5 \leq k \leq 1.5$ shell and 0.13 in shell $1.5 < k \leq 2.5$ 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

$$\hat{u}_i^{n+1}(k_x, k_y, k_z) = \hat{u}_i^{n+}(k_x, k_y, k_z) + \hat{f}_i(k_x, k_y, k_z)dt \quad (22)$$

in which $\hat{u}_i^{n+} = \hat{u}_i^n + (\dots)dt$ with (\dots) 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 $0.5 \leq k \leq 2.5$ that is equivalent to a force-term defined in the two shells as follows

$$\hat{f}_i^n(k_x, k_y, k_z) = \frac{1}{dt} \left(\sqrt{\frac{0.3}{\sum_{0.5 \leq k \leq 1.5} [(\hat{u}_x^{n+})^2 + (\hat{u}_y^{n+})^2 + (\hat{u}_z^{n+})^2] / 2}} - 1 \right) \hat{u}_i^{n+}(k_x, k_y, k_z) \quad (23)$$

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

$$\hat{f}_i^n(k_x, k_y, k_z) = \frac{1}{dt} \left(\sqrt{\frac{0.13}{\sum_{1.5 \leq k \leq 2.5} [(\hat{u}_x^{n+})^2 + (\hat{u}_y^{n+})^2 + (\hat{u}_z^{n+})^2] / 2}} - 1 \right) \hat{u}_i^{n+}(k_x, k_y, k_z) \quad (24)$$

for shell $1.5 < k \leq 2.5$, 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 $E(k=1) = \sum_{0.5 \leq k \leq 1.5} (\hat{u}_x^2 + \hat{u}_y^2 + \hat{u}_z^2) / 2$ and $E(k=2) = \sum_{1.5 < k \leq 2.5} (\hat{u}_x^2 + \hat{u}_y^2 + \hat{u}_z^2) / 2$ 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 $k_x \geq 0$ (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

$$f_i(x, y, z, t_n) = \sum_{k_x, k_y, k_z} e^{i(k_x x + k_y y + k_z z)} \hat{f}_i^n(k_x, k_y, k_z) \quad (25)$$

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