Nota Scientifica e Tecnica N. 186

Sviluppo di un codice spettrale per le equazioni di Navier-Stokes incompressibili

Michele Iovieno, Daniela Tordella

Dipartimento di Ingegneria Aeronautica e Spaziale, Politecnico di Torino
Corso duca degli Abruzzi 24, 10129 Torino

Febbraio 2001

Sommario

This small report is intended as a short manual for the pseudospectral code `dns4` (fourth version) by myself and is mainly a translation of the report N. 186 (2001) which describes, in italian, the first version of the code (written during 1998-1999). The code solves spectrally the Navier-Stokes equations within a parallelepiped domain with sides $L_1 = L_2 = 2\pi, L_3 = 2d\pi$. The code uses a Fourier-Galerkin spatial discretization coupled with a low storage fourth order Runge-Kutta scheme for time integration. Boundary conditions are periodic in each direction. Computation of non linear terms is carried out pseudospectrally with the 3/2 dealiasing rule.

Contenuto

1 Physical problem

This code solves the incompressible Navier-Stokes equations for an incompressible fluid, that is

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$
$$\partial_t \mathbf{u} = -\nabla\cdot(\mathbf{u}\otimes\mathbf{u}) - \nabla p + \nu\nabla^2\mathbf{u} \tag{2}$$

within a parallelepiped domain with dimensione $L_1 = L_2 = 2\pi$, $L_3 = 2d\pi$, with preriodicity condition on each face:

$$\mathbf{u}(\mathbf{x} + L_i\,\mathbf{e}_i, t) = \mathbf{u}(\mathbf{x}, t) \quad i = 1, 2, 3. \tag{3}$$

All variables are made dimensionless so that $L_1 = L_2 = 2\pi$. Equations (1) ed (2) are then put in dimensionless form and equation (1) is replaced by the Poisson equation for the pressure (take the divergence of (2) and insert (1):

$$\partial_t \mathbf{u} = -\nabla\cdot(\mathbf{u}\otimes\mathbf{u}) - \nabla p + \nu\nabla^2\mathbf{u} \tag{4}$$
$$\nabla^2 p = -\nabla\cdot\nabla\cdot(\mathbf{u}\otimes\mathbf{u}) \tag{5}$$

This second formulation is actually used by the code.

2 Numerical method

2.1 Spatial discretization

Problem (3-4) is adapt for the use of a spectral method, due to the simple shape of the domain, and in particular for the implementation of Fourier-Galerkin spectral method (given the periodic boundary conditions). For details about spectral method, please refer to Canuto et al., 1988. *[omissis]* The test and base functions for such method are trigonometric polynomials, so that the test function space is the space of all trigonometric polynomial of degree $\leq N/2$, and the approximate solution of (4-5) is then represented as a truncated Fourier series

$$\mathbf{u}^N(\mathbf{x}, t) = \sum_{k_1,k_2,k_3=-\frac{N}{2}-1}^{\frac{N}{2}-1} \hat{\mathbf{u}}^N_{\mathbf{k}}(t)\mathrm{e}^{i\mathbf{k}\cdot\mathbf{x}} \tag{6}$$

$$p^N(\mathbf{x}, t) = \sum_{k_1,k_2,k_3=-\frac{N}{2}-1}^{\frac{N}{2}-1} \hat{p}^N_{\mathbf{k}}(t)\mathrm{e}^{i\mathbf{k}\cdot\mathbf{x}} \tag{7}$$

where $\mathbf{k}$ is the three-dimensional wave number vector, $\mathbf{k} = (k_1, k_1, k_3)$. Expansion in a orthogonal base generates a linear transformation between the functions $\mathbf{u}$, p and the coefficients $\hat{\mathbf{u}}^N_{\mathbf{k}}$, $\hat{p}_{\mathbf{k}}$ (which are the unknown of the numerical problem). A dicrete transform generates a linear invertible relation between the values of function of the grid points $\mathbf{x_j} = \frac{2\pi}{N}(j_1, j_2, j_3)$ and the coefficients $\hat{\mathbf{u}}^N_{\mathbf{k}}$, $\hat{p}_{\mathbf{k}}$ (the unknown of the numerical problem), which for a generic function f are defined as:

$$\hat{f}_{\mathbf{k}} = \frac{1}{N^3}\sum_{j_1,j_2,j_3=0}^{N-1} f(\mathbf{x_j})\mathrm{e}^{-i\mathbf{k}\cdot\mathbf{x_j}}, \quad \mathbf{x_j} = \frac{2\pi}{N}(j_1, j_2, j_3).$$

Their computation is performed by means of the socalled Fast Fourier Transform (FFT) (Brigham, 1974), which allows the computation of one one-dimensional transform at the

cost of $\sim \frac{5}{2} N \log_2 N$ operations instead of the $O(N^3)$ operations requiored by a straightforward implementation of the definition. Three- dimensional transform required by our problem can be computed thorugh successive iteration of one-dimensional tranforms. This means $3N^2$ transforms for a total of approsimately $\frac{15}{2} N^3 \log_2 N$ operazions.

[omissis]

We get:

$$\partial_t \hat{u}^N_{i,\mathbf{k}} = -ik_j (\widehat{u_j u_i})^N_{\mathbf{k}} - ik_i \hat{p}^N_{\mathbf{k}} - \nu k^2 \hat{u}^N_{i,\mathbf{k}} \tag{8}$$

$$k^2 \hat{p}^N_{\mathbf{k}} = -k_i k_j (\widehat{u_i u_j})^N_{\mathbf{k}} \tag{9}$$

which can be rewritten by eliminating the pressure

$$\partial_t \hat{u}^N_{i,\mathbf{k}} = -ik_j (\widehat{u_j u_i})^N_{\mathbf{k}} + ik_i \frac{k_i k_j}{k^2} (\widehat{u_i u_j})^N_{\mathbf{k}} - \nu k^2 \hat{u}^N_{i,\mathbf{k}} \tag{10}$$

2.2 Aliasing error

As it is evident from (8-10), the main element of the application of the Fourier-Galerkin method to Navier-Stokes equations id the computation of convective terms in (10). Given that the Fourier coefficients are the variables, it is necessary to define a procedure which gives the Fourier coefficients of the product of two functions from the coefficients of the Fourier coefficients of the factors.

[omissis]

The 3/2-rule for a dealiased product of any functions u and v consist in the following steps:

- Fourier coefficients $\{\hat{u}_{\mathbf{m}}\}$ ed $\{\hat{v}_{\mathbf{n}}\}$ are "expanded" from N points to $M = 3N/2$ points in each direction by padding with zeros the additional wavenumbers:

$$\tilde{u}_{\mathbf{k}} = \begin{cases} \hat{u}_{\mathbf{k}} & \text{if } \mid k_i \mid \leq \frac{N}{2} \\ 0 & \text{altrimenti} \end{cases} \qquad \tilde{v}_{\mathbf{k}} = \begin{cases} \hat{v}_{\mathbf{k}} & \text{if } \mid k_i \mid \leq \frac{N}{2} \\ 0 & \text{altrimenti} \end{cases} \tag{11}$$

- Inverse transforms of the "expanded" factors are computed with FFT algorithm.
- The product is then computed in physical spase on the "finer" M-points grid.
- The result is transformed (again FFT) with M points in each direction.
- The transform of the product is truncated from M to N wavenumbers in each direction.

Computational cost with $M = 3N/2$ is given by three 3D-FFT on a finer grid, thus in principle it is about 27/8 more expensive than the pseudospectral computation with aliasing error. In reality it is 1/3 less because some of the one-dimensional FFT in point 2 are not needed because only zeros have to be transformed.

[omissis]

2.3 Time integration

Ordinary differental equations (**??-??**) of the semidiscrete problem are integrated in time by means of a low-storage Runge-Kutta method. Let us consider a general autonomous equation or sistem of equations

$$\partial_t v = F(v). \tag{12}$$

Given the "spectral accuracy" of spatial discretization, it can be useful to implement a high order method (Moreover, explicit high order methods are more stable than the corresponding lower order ones...). I used the fourth order low-sorage RK by Jameson, Schmidt e Turkel (*AIAA*, 1981), that is

$$\begin{aligned} v_1 &= v^n + \frac{1}{4}F(v^n) \\ v_2 &= v^n + \frac{1}{3}F(v_1) \\ v_3 &= v^n + \frac{1}{2}F(v_2) \\ v^{n+1} &= v^n + F(v_3) \end{aligned}$$

It requires four evaluation of F (that is, of right hand side of our system) as all RK-4 schemes, but needs to store only one additional variable (other than v^n and F):

$$\begin{aligned} v &\leftarrow v^n + \frac{1}{4}F(v^n) \\ v &\leftarrow v^n + \frac{1}{3}F(v) \\ v &\leftarrow v^n + \frac{1}{2}F(v) \\ v^{n+1} &\leftarrow v^n + F(v); \end{aligned}$$

[omissis] It can be practically implemented as

$$\begin{aligned} v_1 &\leftarrow v_0 + \frac{1}{4}F(v_0) \\ v_1 &\leftarrow v_0 + \frac{1}{3}F(v_1) \\ v_1 &\leftarrow v_0 + \frac{1}{2}F(v_1) \\ v_0 &\leftarrow v_0 + F(v_1); \end{aligned}$$

(v_0 is the current time step, v_1 is the additional variable to execute RK).
[omissis]

3 Structure of the code

3.1 Files

The program (version 4) consistes in three files:

- `param.h` This two-line file sets the number of points N and $N3$. It is included in the main program and in many subroutines, so that it is necessary to recompile the code when the resolution changes. *Note: this is new since version 4, it turned to be the most convenient way notwihstanding this small inconvenient.*

- `dns4g77.f` This files contains the main program and all subroutines, except those specified below.

- `conv.f` contains the subroutine `conv` which computes the convective terms with the dealiased psudospectral method

- `nag.f` contains all the routines needed for FFT and inverse FFT (they are NAG Mark 11-13, chapter C06).

To compile the code: it is enough to write

```
g77 -o dns4.x dns4_g77.f conv.f nag.f
```

(even if it is better to use the Makefile). It has been tested with g77, gfortran, PGI fortran compiler, Intel fortran compiler, IBM fortran compiler. For resolutions higher than 128^3 is better to use the 64-bit version of the compilers.

3.2 Data format and FFT

All data used by the code - except for the pseudospectral computation of convective terms - are Fourier coefficients of the variables, stored with the hermitian convention. The transform of a real function $u(x)$ satisfies $\hat{u}_{-k} = \overline{\hat{u}_k}$, so that it is necessary to compute (and store) only positive wavenumbers, which can be organized in a real vector of size N. If we indicate with $\hat{u}_k = a_k + ib_k$, the Fourier coefficients, we store them as

$$(a_0, a_1, a_2, ..., a_{\frac{N}{2}-1}, b_{\frac{N}{2}}, ..., b_2, b_1).$$

(note: $b_0 = 0$ because we consider real valued functions, Canuto et al. suggest to set $b_N = 0$). This is in 1D, but for a general function in out three-dimensional domain coefficients a_k and b_k are real after the first FFT, so that this data structure can be used also for the successive FFT in the other directions. The advantage is that it is possibile to use only *real-to-real* FFT, the drawback is that it is not that immediate to rexonstruct the Fourier coefficient for a given vector wavenumber $\mathbf{k}$.

The code uses N points in directions 1 and 2, N_3 points in direction 3. All variables are declared as `REAL*8` and organized as follow:

- Scalar variables: `A(L1,L2,L3)`, the three indices defines the components of the vectorial wavenumber (or the spatial position), they are declared as `A(0:N-1,0:N-1,0:N3-1)`

- Vector variables: `U(L1,L2,L3,J)`. Fourth index defines the component of the vector: they are declared as `A(0:N-1,0:N-1,0:N3-1,3)`

- Symmetric tensors: `S(L1,L2,L3,J)`. Last index spans from 1 to 6: only the diagonl and upper diagonal terms are stored line by line:
  $11, 12, 13 \to 1, 2, 3, 22, 23 \to 4, 5, 33 \to 6$. They are decalred as `A(0:N-1,0:N-1,0:N3-1,3)`.
  Note: there is only one symmetric tensor in the code: tensor $u_i u_j$.

- General tensor`T(L1,L2,L3,J1,J2)`. Last two indices denote the component cartesian of the tensor. They are declared as `A(0:N-1,0:N-1,0:N3-1,3,3)`. *Note: since version 3 of the code, no variable of this kind is used..*

3.3 Main program: time integration of Navier-Stokes equations

The main program executes the time integration of equations (8-9). It is made by three main blocks:

1. Initialization
2. Time integration with RK4
3. periodic data saving

3.4 Data

Main program has the following variables:

- `U0` (vector) the velocity field (current time step)
- `A0` (vector) right hand side of equations
- `U1` (vector) additional vector for RK-4
- `KAPPA`, *KAPPAE* one-dimensional double precision vectors for wavenumbers
- `KAPPA3`, *KAPPAE3* one-dimensional double precision vector for wavenumbers

Variables `KAPPA` etc. contain the wavenumbers and are used only by subroutines which compute the derivatives or which solve the Poisson equation for the pressure.

3.5 Parameters

The physical (and numerical, except the resolution N and N_3) parameters are read from file param0. It and the executable must be in the same directory. It is a simple text file which contains:

1st line: nominal Reynolds number ($1/R$ is the coefficient of diffusive terms)
2nd line: Time step Δt
3rd line: Total number of time steps N_{tot}
4th line: Number of time steps between different savings
5th line: domain size parameter d in direction 3 ($L_3 = 2d\pi$).
6th line: name of the file with initial conditions (16 characters)

3.6 Data input/output

The code reads this file and then the initial conditions from the specified file through a call to subroutine `LEGGI`. (*Note: if the data are not FFT coefficients but are in physical space, it is enough uncomment the call the FFT subroutine* `TRASF3S` *which transforms and overwrites the data.*).

All data are stored in unformatted direct access binary files. Each file contain all the three components of velocity field, starting from last index. They are read as

```
OPEN(1,FILE=...,FORM='UNFORMATTED',ACCESS='DIRECT',RECL=8*3*N*N*N3)
READ(1,REC=1)((((U(L1,L2,L3,J),L1=0,N-1),L2=0,N-1),L3=0,N3-1),J=1,3)
```

(and saved in the same way).

3.7 Time integration

The code solves N_{tot}/N_{salva} cicles of N_{salva} time steps. After each cicle the current velocity field is saved. Saved data are in the same format as initial condition and can be used to prosecute the simulation. Saved files have names `vel`xy`.txt`, where xy is the number of the cicle (*Note: after 99 cicles data will be overwritten and so the program stops. It should be changed to a three-digit format to have the possibility to store 999 files without problems.* The right-hand side of equation (**??**) is computed by subroutine `AVANTI`, which contain the following steps:

1. Computation of tensor $u_i u_j$, stored in variable `MG`. It calls subroutine `CONV`).
2. Computation of its divergence, stored in output variable `A`. It calls subroutine `DIVT2`).
3. Computation of diffusive terms, which are added to `A`
4. Projection onto a divergence-free space (computazion of pressure and addiction of $-\nabla p$ to `A`.

All operations are made by to specialized subroutines, so that `AVANTI` is mainly a sequence of `CALL` statements.

4 List of subdiroutines

4.1 Discrete FFT

NAG subroutines `C06FAF e C06FBF` perform one-dimensional real-to-real direct and inverse transform. Subroutines `TRASF1` and `ATRASF1` are an interface to them. More information about such subroutines is freely available in NAG website. Three dimensional transforms are made iterating one-dimensional tranforms:

$$\begin{aligned}\tilde{u}_{\mathbf{k}} &= \sum_{j_1=0}^{N-1}\sum_{j_2=0}^{N-1}\sum_{j_3=0}^{N-1} u(\frac{2\pi i}{N}j_1, \frac{2\pi i}{N}j_2, \frac{2\pi i}{N}j_3)\mathrm{e}^{\frac{2\pi i}{N}(k_1 j_1 + k_2 j_2 + k_3 j_3)} = \\ &= \sum_{j_3=0}^{N-1} \mathrm{e}^{\frac{2\pi i}{N}k_3 j_3}\left(\sum_{j_2=0}^{N-1} \mathrm{e}^{\frac{2\pi i}{N}k_2 j_2}\left(\sum_{j_1=0}^{N-1} u(\frac{2\pi i}{N}j_1, \frac{2\pi i}{N}j_2, \frac{2\pi i}{N}j_3)\mathrm{e}^{\frac{2\pi i}{N}k_1 j_1}\right)\right)\end{aligned}$$

so that it is possibile to compute $\tilde{u}(k_1, k_2, k_3)$ with a three step procedure:

$$\begin{aligned} f(k_1, j_2, j_3) &\leftarrow \sum_{j_1=0}^{N-1} u(j_1, j_2, j_3)\mathrm{e}^{\frac{2\pi i}{N}k_1 j_1} \\ f(k_1, k_2, j_3) &\leftarrow \sum_{j_2=0}^{N-1} f(k_1, j_2, j_3)\mathrm{e}^{\frac{2\pi i}{N}k_2 j_2} \\ \tilde{u}(k_1, k_2, k_3) &\leftarrow \sum_{j_1=0}^{N-1} f(k_1, k_2, j_3)\mathrm{e}^{\frac{2\pi i}{N}k_3 j_3}. \end{aligned}$$

In each step N^2 one dimensional transforms are computed, for a total of $3N^2$ FFT.
Note: for an odd reason, NAG miss a $N^{1/2}$ coefficient in FFT and $N^{-1/2}$ in inverse FFT. This is normally not a problem (even if the FFT coefficients depend from N and I personally dislike that), unless in pseudospctral computation of convective terms, where the number of

direct FFT is not equal to the number of inverse FFT. This is the reason for the socalled "correzione" (correction) in the last loop of three dimensional subroutines.

Details about subroutines which compute direct and inverse FFT:

- `TRASF1(F,N)`. One dimensional FFT. `F(0:N-1)` : on input, data to be tranformed, on output, transformed data (data overwritten). Essentially this subroutine is an interface to to `C06FAF` (generates control and additional variables). No "correction"
- `TRASF3X(U,UT,N,N3)`. Three dimensional FFT. `U(0:N-1,0:N-1,0:N3-1)`: input, data in physical space; `UT(0:N-1,0:N-1,0:N3-1)` : output, transformed data. It calls `TRASF1`. "Correction" applied in the last loop. No more used since version 2, replaced by `TRASF3S`.
- `TRASF3S`. Three dimensional FFT, but unlike tt TRASF3X data are overwritten in variable `U(0:N-1,0:N-1,0:N3-1)`. Moreover it is more efficient.
- `ATRASF1(F,N)`. One dimensional inverse FFT. `F(0:N-1)` : on input Fourier coefficients, on output data in physical space (data overwritten). It is an interface to NAG subroutines `C06FBF` and`C06GBF`. No "correction".
- `ATRASF3X(UT,U,N,N3)`. Inverse FFT. `UT(0:N-1,0:N-1,0:N3-1)`: input, Fourier coefficients. `U(0:N-1,0:N-1,0:N3-1)`: output, data in physical space. It calls `ATRASF1`. "Correction" applied in the last loop. No more used since version 2, replaced by `ATRASF3S`.
- `ATRASF3S(U,N,N3)` Three dimensional inverse tranform, but data are overwritten in variable `UT(0:N-1,0:N-1,0:N3-1)`. It can be used only for inverse FFT of Fourier coefficient expanded with the 3/2 rule adding zeros in the additional wavenumber, because it assumes that last third of coefficient is zero and skip consequently some one dimensional FFT, sparing a lot of time.

4.2 Pseudospectral computation of convective terms

Computation of products is done with the dealiased (3/2 rule) pseudospectral, in the following steps:

1. Expansion of Fourier coefficients from N^3 to M^3, with $M = 3N/2$.
2. Inverse FFT (M^3).
3. Product in physical space.
4. FFT of the result (M^3).
5. Truncation of the FFT coefficient from M^3 to N^3.

There is a general purpouse subroutine `PRODOTTO` and a specialized subroutine `CONV` for convective terms. Subroutine `PRODOTTO` performs the above mentioned five steps. As regard the expansion and truncation of Fourier coefficients, please note that high wavenumbers are in the centre of the data due to the implementation of the hermitian format.

[omissis]

- `PRODOTTO(U,V,W)`. Computes the Fourier coefficients of $w = uv$ from the coefficients of u and v. Input: `U` and `V`, scalars. Output: `W`, scalar. Note that `CALL PRODOTTO(A,B,B,N,N)` write the result in variable B, thus overwriting the input (but it works). It is no more used in the DNS code.

- `MODULO(F,FM)`. Simply computes the Fourier coefficients of the modulus $|\mathbf{f}|$ a vector $\mathbf{f}$. Input: `F`, vector. Output `FM`, scalar. Never used in DNS code.

- `SEL(J,JJ,N,M)`. Simply subroutine which select the index to expand and truncate Fourier coefficients. Input: `J` index within a N-points array. Output: `JJ` corresponding index in the same array, expanded to M points. (N, M should be specified with $M > N$ otherwise it fails. In short it does:

$$JJ = \begin{cases} J \text{ if } J < N/2 \\ M - N + J \text{ if } J \geq N/2 \end{cases} \tag{13}$$

- `TRONCA(FE,FT,N,N3,M,M3)`. Truncates Fourier coefficients from $M, M3$ to $N, N3$. Input: `FE(0:M-1,0:M-1,0:M3-1)` scalar data to be truncated. Output: `FT(0:N-1,0:N-1,0:N3-1)` truncated data. It must be $N < M$, $N3 < M$, otherwise it fails. It calls `SEL`.

- `ESPANDI(FT,FE,N,N3,M,M3)`. Expands data from $N, N3$ to $M, M3$. Input: `FT(0:N-1,0:N-1,0:N3-1)`. Output: `FE(0:M-1,0:M-1,0:M3-1)` expanded Fourier coefficients padded with zeros in the additional wavenumbers. It must be $N < M$, $N3 < M$, otherwise it fails. It calls `SEL`.

- `CONV(U,MG)`. This subroutines computes the convective terms. Input: `U` (vector). Output: `MG` symmetric tensor. All three velocity components are expanded, inverse transformed and then all the six products $u_i u_j$ ($j \geq i$ are computed, transformed and truncated. It costs three inverse FFT and six direct FFT and thus it is more efficient than six calls to `PRODOTTO`. It uses `ATRASF3S` and `TRASF3S`.

4.3 Differential operators

The code operates on Fourier coefficients, so that derivativs are just multiplication fot wavenumbers, to be performed considering the hermitian form of data. In one dimension the coefficients $\tilde{f}_k = a_k + ib_k$, withn $k = 0, \ldots, N/2 - 1$ of the discrete FFT of f are organized in a real vector $f = (f_k)$ as

$$f = \left(a_0, a_1, \ldots, a_{\frac{N}{2}-1}, 0, b_{\frac{N}{2}-1}, \ldots, b_1\right),$$

so that the coefficients of its derivative, given by $\tilde{f}' = k(-b_k + ia_k)$, must be organized in a vector g as

$$g = \left(0, f_{N-1}, 2f_{N-2}, \ldots, \left(\frac{N}{2} - 1\right) f_{\frac{N}{2}-1}, 0, \left(\frac{N}{2} - 1\right) f_{\frac{N}{2}-1}, \ldots, f_1\right).$$

In three dimension nothing changes (for partial derivative that operation is done on one iondex only)

- `DERIVATA(F,J,DF,N,N3,KAPPA,KAPPA3)`. It computes the partial derivative $Df = \partial f / \partial x_j$.
  Input: `F(0:N-1,0:N-1,0:N3-1)`, scalar to be derived; N, $N3$, $KAPPA(0 : N-1)$, $KAPPA3(0 : N3-1)$ as defined in the main program; J direction of the derivative.
  Output: `DF(0:N-1,0:N-1,0:N3-1)` scalar, the derivative of `F` in the direction x_j defined by J.

- `DIVERGENZA(U,DIVU,KAPPA,KAPPA3)`. It computes the divergence of vector $\mathbf{u}$. Input: `U(0:N-1,0:N-1,0:N3-1,3)` vector; $KAPPA(0 : N-1)$, $KAPPA3(0 : N3-1)$ as defined in the main program.
  Output: `DIVU(0:N-1,0:N-1,0:N3-1)` scalar, the divergence of `U`. It calls `DERIVATA`.

- `DIVT(T,DT,N)`. It was used to compute the divergence of a generic tensor. Removed since version 4.

- `DIVT2(T,DT,KAPPA,KAPPA3)`. It computes the divergence of a symmetric tensor.
  Input: `T(0:N-1,0:N-1,0:N3-1,6)` a symmetric tensor; $KAPPA(0:N-1)$, $KAPPA3(0:N3-1)$ as defined in the main program.
  Output: *T(0:N-1,0:N-1,0:N3-1,3)* vector, it contains the divergence of `T`.
  It calls `DIVERGENZA`.

- `GRAD(P,GP,KAPPA,KAPPA3)`. It computes the gradient of a scalar function.
  Input: `P(0:N-1,0:N-1,0:N3-1)` a scalar; $KAPPA(0:N-1)$, $KAPPA3(0:N3-1)$ as defined in the main program.
  Output: *GP(0:N-1,0:N-1,0:N3-1,3)* vector, it contains the gradient of `P`.
  It calls `DERIVATA`.

- `GRADT(U,GU,KAPPA,KAPPA3,N,N3)`. It computes the tensorial gradient of a vector. Input: `U(0:N-1,0:N-1,0:N3-1,3)` a vector; $KAPPA(0:N-1)$, $KAPPA3(0:N3-1)$ as defined in the main program.
  Output: `U(0:N-1,0:N-1,0:N3-1,3,3)` a scalar, it contains the gradient of `U`
  The subroutines calls three times subrotine `GRAD`.
  Note: subroutine non more used since version 3, because non simmetric tensor variables are no more used.

- `LAPLACE(F,LF,N,KAPPA)`. Calcola il laplaciano di una funzione scalare `F(L1,L2,L3)`; il risultato è posto in `LF(L1,L2,L3)`. Il programma esegue la moltiplicazione per i quadrati dei numeri d'onda. La variabile intera `K(L)`, dichiarata con limiti 0:N-1, deve essere introdotta come ingresso e pari a $(0, 1, 2, \ldots, N/2, \ldots, 2, 1)$.

- `LAPLACEV(U,LU,N,KAPPA)`. Calcola il laplaciano di una data funzione vettoriale in ingresso `U(L1,L2,L3,J)`; il risultato è posto in `LU(L1,L2,L3,J)`. Il sottoprogramma richiama `LAPLACE` per eseguire il laplaciano delle componenti di U. La variabile intera `K(L)`, dichiarata con limiti 0:N-1, deve essere introdotta come ingresso e pari a $(0, 1, 2, \ldots, N/2, \ldots, 2, 1)$; viene usata come ingresso per `LAPLACE`.

4.4 Elimination of the divergence from a vector field

Subroutine `ELIMINAD` projects a vectorial field $\mathbf{u}_*$ in the space of solenoidal functions (zero divergence). This operation is performed by solving a Poisson equation: ...
Questa operazione viene effettuata risolvendo un'equazione di Poisson: per il teorema di Ladyzhenskaja lo spazio delle funzioni vettoriali può essere decomposto come somma diretta dello spazio delle funzioni a divergenza nulla e di quello delle funzioni a rotore nullo, pertanto si può scrivere

$$\mathbf{u}_* = \mathbf{u} + \nabla\phi, \quad \nabla \cdot \mathbf{u} = 0.$$

Prendendone la divergenza abbiamo

$$\nabla^2\phi = \nabla \cdot \mathbf{u}_*$$

Risolvendo questa equazione abbiamo ϕ e quindi possiamo calcolare il campo a divergenza nulla $\mathbf{u} = \mathbf{u}_* - \nabla\phi$. In dettaglio il sottoprogramma `ELIMINAD(U,N,KAPPA,KAPPAE)` riceve in ingresso una variabile `U(L1,L2,L3,J)` di tipo vettoriale, con indici `L1`, `L2`, `L3`, dichiarati fra 0 ed $N-1$ e restituisce in uscita nella stessa variabile un campo vettoriale a divergenza nulla. I vettori di interi `KAPPA(0:N/2-1)` e `KAPPAE(0:N-1)` sono definiti nel sottoprogramma `AVANTI`. Come tutti gli altri sottoprogrammi, sia i dati in ingresso che quelli in uscita riportano in coefficienti di Fourier della trasformata discreta delle varibili fisiche.

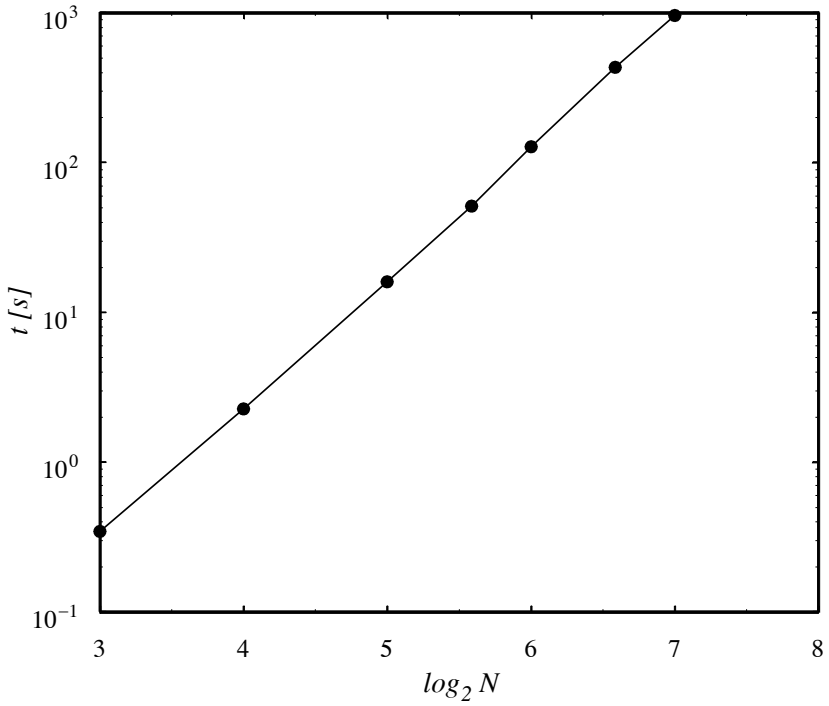

Figura 1: Tempi di calcolo per il codice `DNS1` in funzione del numero di modi di Fourier utilizzati su processore1Pentium 700 MHz.

5 Utilizzo del codice

In figura 1 si riporta il tempo di calcolo necessario per eseguire un passo temporale utilizzando il codice `DNS1.for` su una macchina dotata di processore Pentium a 700 MHz in funzione del numero di modi di Fourier utilizzati per la discretizzazione spaziale. Approssimativamente si ha la relazione

$$t \sim aN^3$$

con $a \approx 4,50 \cdot 10^{-4} s$. Tale legge di scala rileva una crescita del tempo di calcolo minore di quanto ci si sarebbe atteso da una stima basata sul tempo necessario per eseguire le trasformate, la parte più gravosa del codice, che cresce come $N^3 \log_2 N$. Il tempo di calcolo complessivo per raggiungere in una simulazione un dato tempo finale T è pertanto indicativamente proporzionale ad N^4 poiché la stabilità impone che il passo temporale sia ridotto approssimativamente come N^{-1} per flussi ad alto numero di Reynolds (si veda §2.3).

Il primo esempio che si presenta ha prevalentemente carattere di prova del codice. Il flusso che viene risolto è infatti costituito da una schiera periodica di vortici bidimensionali. Le condizioni iniziali per questo flusso si trovano in Townsend (1976): in termini adimensionali la funzione di corrente è data da

$$\psi_0(x_1, x_2) = \cos \alpha x_1 \cos \alpha x_2. \tag{14}$$

La soluzione di questo problema è

$$\psi(x_1, x_2, t) = \mathrm{e}^{-\frac{2\alpha^2}{Re} t} \psi_0(x_1, x_2) \tag{15}$$

Anche se il problema è in realtà bidimensionale, è stato risolto numericamente usando il codice tridimensionale qui presentato per le equazioni di Navier-Stokes e poi calcolando a

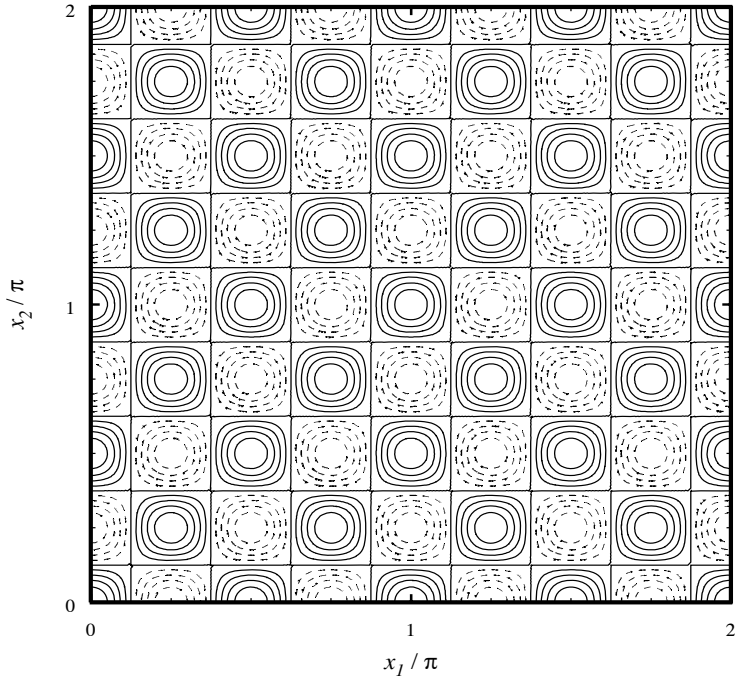

Figura 2: Linee di corrente per il flusso dato da (15), con $\alpha = 4$ e $Re = 20$ per $t = 1$. Simulazione con 48^3 punti di griglia.

posteriori la funzione di corrente risolvendo l'equazione di Poisson

$$-\nabla^2 \psi = \omega_3$$

con condizioni di periodicità. In figura 2 si trova una visualizzazione delle linee di corrente per un tempo adimensionale $t = 1$. Le prove effettuate con diverse risoluzioni (da 16^3 fino a 64^3) su questo flusso mostrano come già con 32^3 punti si abbia un errore relativo dell'ordine di 10^{-9} per $t = 1$.

Il secondo esempio riguarda la simulazione della turbolenza omogenea ed isotropa in decadimento, con $R_\ell = 350$. L'evoluzione della turbolanza dietro griglie viene simulata risolvendo le equazioni in un dominio cubico con condizioni din periodicità. In questo caso Questo flusso è stato risolto sia col codice per le equazioni di Navier-Stokes su 128^3 punti, sia usando la simulazione "large-eddy" con 32^3 punti-griglia. La figura 3 riporta un esempio di spettro unidimensionale calcolato usando il codice Navier-Stokes, mentre in figura è riprodotto l'andamento del decadimento dell'energia cinetica di questo flusso valutato con entrambi i metodi. Si trova un decadimento rappresentabile secondo la legge $E/E_0 \approx (t/\tau)^{-1.23}$, in accordo sia con la misure sperimentali (Comte-Bellot & Corrsin, 1971) sia con i risultati delle simulazioni numeriche presenti in letteratura.

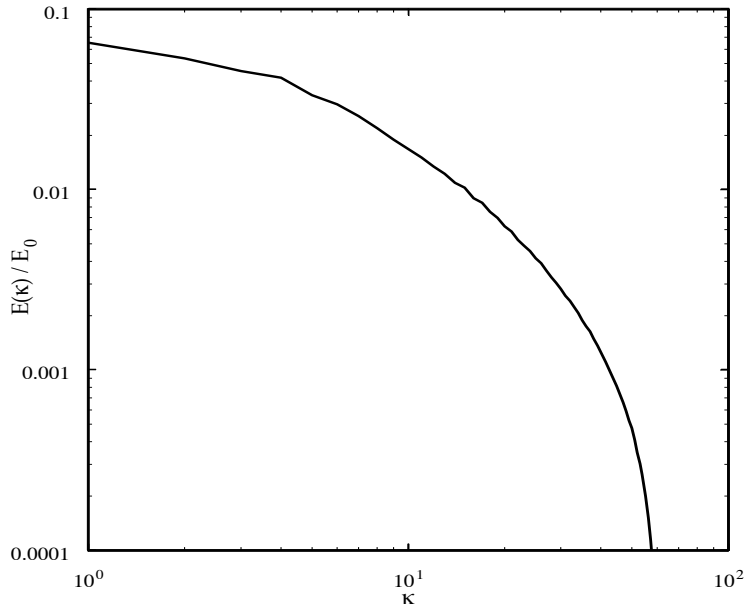

Figura 3: Spettri unidimensionali da simulazione numerica diretta turbolenza omogenea con 128^3 punti di griglia, $t/\tau = 0.9$.

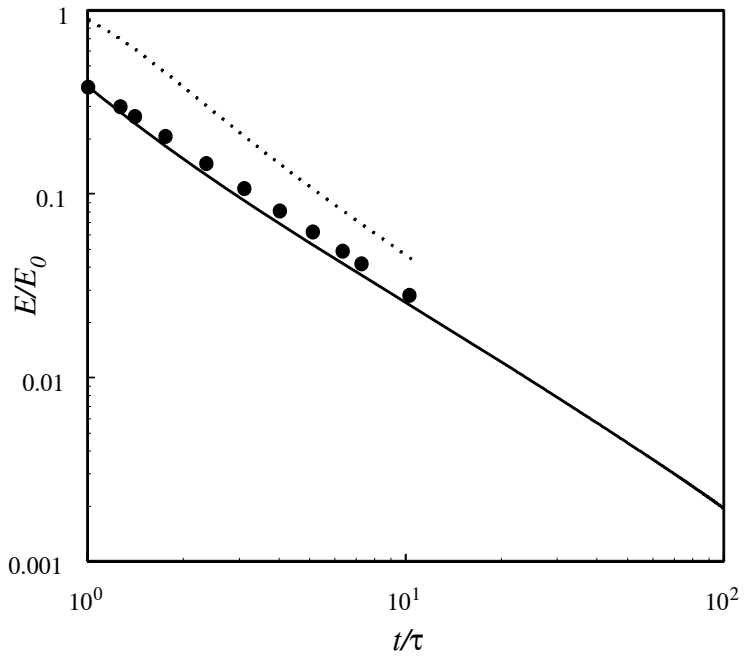

Figura 4: Decadimento dell'energia: $(\cdots\cdots)$ simulazione numerica diretta 128^3, $(---)$ simulazione "large-eddy" 32^3 modello momento angolare, $\circ$ dati di simulazione numerica diretta filtrati per permettere il confronto diretto fra i due metodi.

Bibliografia

[1] Akselvoll K., Moin P. 1996 Large-eddy simulation of turbulent confined coannular jets. *J. fluid Mech.*, **315**, 387–411.

[2] Bardina J., Ferziger J. H., Reynolds. W. C. 1983 Improved turbulence models based on large eddy simulation of homogeneous, incompressible, turbulent flows. *Report Tf-19*, Stanford University.

[3] Brigham E. O. 1974 *The fast Fourier transform*. Prentice-Hall, Englewood Cliffs.

[4] Canuto C., Hussaini. M. Y., Quarteroni. A., Zang. T. A. 1988 *Spectral methods in fluid dynamics.* Springer Verlag, Berlin.

[5] Comincioli V. 1995 *Analisi numerica* McGraw-Hill, Milano.

[6] Comte-Bellot G., Corrsin. S. 1971 Simple eulerian time correlation of full and narrow-band velocity signals in grid-generated isotropic turbulence. *J.Fluid Mech.* **48**, 273–337.

[7] Frisch U. 1995 *Turbulence*, Cambridge University Press.

[8] Iovieno M., Tordella D. 1999 Shearless turbulence mixings by means of the angular momentum large eddy model, *American Physical Society - 52th DFD Annual Meeting.*

[9] Iovieno M. 2001 *Angular momentum applications in fluid mechanics*, Tesi di Dottorato, Politecnico di Torino.

[10] Jameson A., Schmidt H., Turkel E. 1981 Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta stepping schemes. *AIAA Paper* N. 81-1259.

[11] Moin P., Kim J. 1982 Numerical investigation of turbulent channel flow. *J.Fluid Mech.* **118**, 341–377.

[12] AA.VV. 1992 *Nag Fortran Routine Manual, Mark 11 Release.*

[13] Peyret R. 1999 Introduction to high-order approximation methods for computational fluid dynamics. *Prèpublication n^o 543*, Universite de Nice-Sophia Antipolis .

[14] Rudin W. 1974 *Analisi reale e complessa*. Bollati-Boringhieri, Torino.

[15] Townsend A.A. 1976 *The structure of turbulent shear flow*, Cambridge University Press.

Listato

Si riportano i listati dei codici descritti nella presente nota. Copia dei codici e dei sottoprogrammi Nag utilizzati per l'esecuzione delle FFT si trovano nel disco allegato.

Programma DNS1.for - equazioni di Navier-Stokes

```
      PROGRAM DNS1
C     SIMULAZIONE TURBOLENZA OMOGENEA ED ISOTROPA (COME IL PROGRAMMA
C     LES), MA USANDO LE TRASFORMATE REALI PER RISPARMIARE TEMPO E
C     SOPRATTUTTO MEMORIA. AVANZAMENTO NEL TEMPO CON RK4.
C
C     NOTAZIONI:
C      N=NUMERO DI PUNTI IN OGNI DIREZIONE
C      M=NUMERO DI PUNTI PER IL CALCOLO DEI PRODOTTI SENZA ALYASING
C      IZ2=NUMERO DI PASSI TEMPORALI DELL'INTERA SIMULAZIONE
C      IZ=NUMERO DI PASSI TEMPORALI DOPO CUI SI SALVANO I RISULTATI SU
C         DISCO
C      IZ1=NUMERO DI CICLI DI IZ PASSI TEMPORALI
C
C      SCALARI:F(L1,L2,L3)
C        GLI INDICI INDIVIDUANO IL NUMERO D'ONDA VETTORIALE
C      VETTORI:U(L1,L2,L3,J)
C        I PRIMI TRE INDICI INDIVIDUANO IL NUMERO D'ONDA VETTORIALE, IL
C        QUARTO LA COMPONENTE DEL VETTORE
C      TENSORI:G(L1,L2,L3,J1,J2)
C        I PRIMI TRE INDICI INDIVIDUANO IL NUMERO D'ONDA VETTORIALE,
C        GLI ULTIMI DUE LE COMPONENTI DEL TENSORE.
C      TUTTE LE GRANDEZZE SONO LE TRASFORMATE DI FOURIER IN FORMA
C      HERMITIANA.
C
C     I DATI SI DEVONO TROVARE IN FILE C:\DATI\INIZIO\
C     I RISULTATI VENGONO SALVATI SU DISCO NELLA DIRECTORY D:\DATI\RIS.
C     CHE VIENE SPECIFICATA, CON NOME VEL**.BIN
C     SONO TUTTI MEMORIZZATI IN FILE BINARI AD ACCESSO DIRETTO
C
C     VARIABILI:
C     U(L1,L2,L3,J)=VELOCITA'
      WRITE(6,*)'Numero di punti per direzione'
      READ(5,*)N
      WRITE(6,*)'Numero totale di passi temporali ITOT'
      READ(5,*)ITOT
      WRITE(6,*)'Numero di passi temporali per ciclo IS'
      WRITE(6,*)'(dopo cui si salva su disco)'
      WRITE(6,*)'Suggerimento:ITOT multiplo IS'
      READ(5,*)IS
C     VARIABILI PRINCIPALI
      CALL PRINCIPALE(N,ITOT,IS)
      END
C     ****************************************************************
      SUBROUTINE PRINCIPALE(N,ITOT,IS)
C     VARIABILI PRINCIPALI
      REAL*8 U0(0:N-1,0:N-1,0:N-1,1:3)
      REAL*8 U1(0:N-1,0:N-1,0:N-1,1:3)
      REAL*8 A1(0:N-1,0:N-1,0:N-1,1:3)
C     NOMI DEI FILE
      CHARACTER*32 S
      CHARACTER S1,S2
      CHARACTER*8 NOME1
C     COEFFICIENTI
      REAL*8 DELTAT,RE
C     DEFINIZIONE DEL VETTORE DEI NUMERI D'ONDA
      INTEGER KAPPA(0:N/2),KAPPAE(0:N-1)
C     ASSEGNAZIONE COSTANTI
      WRITE(6,*)'Passo temporale '
```

```
      READ(5,*)DELTAT
      WRITE(6,*)'Numero di Reynolds '
      READ(5,*)RE
C     DIRECTORY RISULTATI
      WRITE(6,*)'(Sotto)directory in cui si salveranno i risultati:'
      WRITE(6,*)'C:\DATI\RIS\...'
      READ(5,*)NOME1
      IC=ITOT/IS
C
C     DEFINIZIONE DEL VETTORE DEI NUMERI D'ONDA
      DO 111 I=0,N/2
      KAPPA(I)=I
111   KAPPAE(I)=I
      DO 112 I=1,N/2-1
112   KAPPAE(N-I)=I
C
      M=3*N/2
C
      CALL LEGGI(U0,N)
C
C     AVANZAMENTO NEL TEMPO CON RUNGE-KUTTA 4
C     IL SOTTOPROGRAMMA AVANTI DEFINISCE LA FUNZIONE A
C     DU/DT=A  (D/DT=DERIVATA PARZIALE NEL TEMPO)
C     ****ADESSO USIAMO LA VERSIONE RK4 "LOW STORAGE" *****
C
      WRITE(6,*)'Terminata la lettura dei dati...'
C
      DO 999 I1=1,IC
C
      DO 99 I2=0,(IS-1)
C
      CALL AVANTI(U0,A1,N,M,RE,KAPPA,KAPPAE)
      DO 15 J1=1,3
      DO 15 L3=0,N-1
      DO 15 L2=0,N-1
      DO 15 L1=0,N-1
      U1(L1,L2,L3,J1)=U0(L1,L2,L3,J1)+DELTAT*A1(L1,L2,L3,J1)*0.25D0
15    CONTINUE
      CALL AVANTI(U1,A1,N,M,RE,KAPPA,KAPPAE)
      DO 20 J1=1,3
      DO 20 L3=0,N-1
      DO 20 L2=0,N-1
      DO 20 L1=0,N-1
      U1(L1,L2,L3,J1)=U0(L1,L2,L3,J1)+DELTAT*A1(L1,L2,L3,J1)*(1.0D0/3)
20    CONTINUE
      CALL AVANTI(U1,A1,N,M,RE,KAPPA,KAPPAE)
      DO 25 J1=1,3
      DO 25 L3=0,N-1
      DO 25 L2=0,N-1
      DO 25 L1=0,N-1
      U1(L1,L2,L3,J1)=U0(L1,L2,L3,J1)+DELTAT*A1(L1,L2,L3,J1)*0.5D0
25    CONTINUE
      CALL AVANTI(U1,A1,N,M,RE,KAPPA,KAPPAE)
      DO 30 J1=1,3
      DO 30 L3=0,N-1
      DO 30 L2=0,N-1
      DO 30 L1=0,N-1
      U0(L1,L2,L3,J1)=U0(L1,L2,L3,J1)+DELTAT*A1(L1,L2,L3,J1)
30    CONTINUE
C      WRITE(6,*)'Terminato passo temporale ',I2,'/',I1
99    CONTINUE
C     ADESSO SALVIAMO SU DISCO LO IS-ESIMO PASSO TEMPORALE,
      J1=MOD(I1,10)
      J2=MOD(I1/10,10)
      S1=CHAR(J2+ICHAR('0'))
      S2=CHAR(J1+ICHAR('0'))
```

```
      S='C:\DATI\RIS\'//NOME1//'\VEL'//S1//S2//'.BIN'
      OPEN(1,FILE=S,FORM='UNFORMATTED',ACCESS='DIRECT',
     >RECL=8,STATUS='NEW')
      DO 50 J1=1,3
      DO 50 L3=0,N-1
      DO 50 L2=0,N-1
      DO 50 L1=0,N-1
50    WRITE (1)U0(L1,L2,L3,J1)
      CLOSE (1)
      WRITE(6,*)'Completata scrittura su disco ciclo, ',I1
C
999   CONTINUE
C
1000  CONTINUE
      END
C     ******************************************************************
C     FINE PROGRAMMA PRINCIPALE
C     ******************************************************************
      SUBROUTINE AVANTI(U,A,N,M,RE,KAPPA,KAPPAE)
C     VARIABILI FORMALI IN INGRESSO
      REAL*8 U(0:N-1,0:N-1,0:N-1,1:3)
C     VARIABILI FORMALI IN USCITA
      REAL*8 A(0:N-1,0:N-1,0:N-1,1:3)
C     VARIABILI INTERNE
      REAL*8 MG(0:N-1,0:N-1,0:N-1,1:6)
      REAL*8 A1(0:N-1,0:N-1,0:N-1),A2(0:N-1,0:N-1,0:N-1)
      REAL*8 P(0:N-1,0:N-1,0:N-1,1:3)
C     COEFFICIENTE DEL MODELLO
      REAL*8 RE
C     DEFINIZIONE DEL VETTORE DEI NUMERI D'ONDA
      INTEGER KAPPA(0:N/2),KAPPAE(0:N-1)
C
C     CALCOLO DI A
C
C     1-CALCOLO DI -(U_J1*U_J2)
      DO 6 J1=1,3
      DO 6 J2=J1,3
      DO 4 L3=0,N-1
      DO 4 L2=0,N-1
      DO 4 L1=0,N-1
      A1(L1,L2,L3)=U(L1,L2,L3,J1)
4     A2(L1,L2,L3)=U(L1,L2,L3,J2)
      CALL PRODOTTO(A1,A2,A1,N,M)
      CALL ELIMINAS(A1,N)
      IF(J1-1)51,51,52
51    JS=J1+J2-1
      GOTO 53
52    JS=J1+J2
53    CONTINUE
      DO 5 L3=0,N-1
      DO 5 L2=0,N-1
      DO 5 L1=0,N-1
5     MG(L1,L2,L3,JS)=-A1(L1,L2,L3)
6     CONTINUE
C
C     2-CALCOLO DELLA DIVERGENZA DI QUESTO TENSORE (VETTORE 6 EL.)
      CALL DIVT2(MG,A,N,KAPPA)
C
C     3-I AGGIUNTA DEL GRADIENTE DELLA PRESSIONE
      CALL DIVERGENZA(A,A2,N,KAPPA)
      CALL GRAD(A2,P,N,KAPPA)
C
      DO 91 J=1,3
      DO 9 L3=0,N-1
      DO 9 L2=0,N-1
      DO 9 L1=0,N-1
```

```
      IK2=KAPPAE(L1)**2+KAPPAE(L2)**2+KAPPAE(L3)**2
      IF(IK2-0)99,99,98
98    A(L1,L2,L3,J)=A(L1,L2,L3,J)+P(L1,L2,L3,J)/IK2
99    CONTINUE
9     CONTINUE
91    CONTINUE
C
C     2.6-AGGIUNTA DI 'LAPLACIANO DI U '/'NUMERO DI REYNOLDS'
C
      CALL LAPLACEV(U,P,N,KAPPAE)
C
      DO 10 J=1,3
      DO 10 L3=0,N-1
      DO 10 L2=0,N-1
      DO 10 L1=0,N-1
10    A(L1,L2,L3,J)=A(L1,L2,L3,J)+P(L1,L2,L3,J)/RE
C
      RETURN
      END
C     ******************************************************************
C     LETTURA DATI DI PARTENZA
C     ----------------------------------------------------------------
      SUBROUTINE LEGGI(UI,N)
      REAL*8 UI(0:N-1,0:N-1,0:N-1,1:3)
      CHARACTER*20 SI
      CHARACTER*28 S
C     VIENE LETTO U0 DAL FILE SPECIFICATO DA TASTIERA
      WRITE(6,*)'Nome file dati velocita, C:\DATI\INIZIO\...'
      READ(5,*)SI
      S='C:\DATI\INIZIO\'//SI
      OPEN(1,FILE=S,FORM='UNFORMATTED',
     >ACCESS='DIRECT',RECL=8,STATUS='OLD')
      DO 50 J1=1,3
      DO 50 L3=0,N-1
      DO 50 L2=0,N-1
      DO 50 L1=0,N-1
50    READ(1)UI(L1,L2,L3,J1)
      CLOSE (1)
      RETURN
      END
C     ******************************************************************
```

Programma Les4.for - simulazione “large-eddy”

```
      PROGRAM LES4BIS
C     SIMULAZIONE TURBOLENZA OMOGENEA ED ISOTROPA
C     MA USANDO LE TRASFORMATE REALI PER RISPARMIARE TEMPO E
C     SOPRATTUTTO MEMORIA. AVANZAMENTO NEL TEMPO CON RK4.
C
C      NOTAZIONI:
C      N=NUMERO DI PUNTI IN OGNI DIREZIONE
C      M=NUMERO DI PUNTI PER IL CALCOLO DEI PRODOTTI SENZA ALYASING
C      IZ2=NUMERO DI PASSI TEMPORALI DELL'INTERA SIMULAZIONE
C      IZ=NUMERO DI PASSI TEMPORALI DOPO CUI SI SALVANO I RISULTATI SU
C         DISCO
C      IZ1=NUMERO DI CICLI DI IZ PASSI TEMPORALI
C
C     SCALARI:F(L1,L2,L3)
C        GLI INDICI INDIVIDUANO IL NUMERO D'ONDA VETTORIALE
C     VETTORI:U(L1,L2,L3,J)
C        I PRIMI TRE INDICI INDIVIDUANO IL NUMERO D'ONDA VETTORIALE, IL
C        QUARTO LA COMPONENTE DEL VETTORE
C     TENSORI:G(L1,L2,L3,J1,J2)
C        I PRIMI TRE INDICI INDIVIDUANO IL NUMERO D'ONDA VETTORIALE,
C        GLI ULTIMI DUE LE COMPONENTI DEL TENSORE.
C     TUTTE LE GRANDEZZE SONO LE TRASFORMATE DI FOURIER IN FORMA
C     HERMITIANA.
C
C     I DATI SI TROVANO NEI FILE D:\DATI\INIZIO\*_U.BIN E *_H.BIN
C     I RISULTATI VENGONO SALVATI SU DISCO NELLA DIRECTORY D:\DATI\RIS.
C     I FILE HANNO NOMI VEL**.BIN E ANG**.BIN
C     SONO TUTTI MEMORIZZATI IN FORMA BINARIA(SENZA FORMATO)
C
C     VARIABILI:
C     U(L1,L2,L3,J)=VELOCITA'
C     H(L1,L2,L3,J)=MOMENTO ANGOLARE
C
      WRITE(6,*)'Numero di punti(N=16 oppure 32)'
      READ(5,*)N
      WRITE(6,*)'Numero totale di passi temporali IZ2'
      READ(5,*)IZ2
      WRITE(6,*)'Numero di passi temporali per ciclo IZ'
      WRITE(6,*)'(dopo cui si salva su disco)'
      READ(5,*)IZ
      WRITE(6,*)'Quanti passi salvare?:1 ogni...IS'
      WRITE(6,*)'Suggerimento:IZ multiplo IS'
      READ(5,*)IS
C
      CALL PRINCIPALE(N,IZ,IZ2,IS)
      END
C     ****************************************************************
      SUBROUTINE PRINCIPALE(N,IZ,IZ2,IS)
C     VARIABILI PRINCIPALI
      REAL*8 U(0:N-1,0:N-1,0:N-1,1:3,0:IZ),H(0:N-1,0:N-1,0:N-1,1:3,0:IZ)
C     VARIABILI AUSILIARIE
      REAL*8 U0(0:N-1,0:N-1,0:N-1,1:3),H0(0:N-1,0:N-1,0:N-1,1:3)
      REAL*8 U1(0:N-1,0:N-1,0:N-1,1:3),H1(0:N-1,0:N-1,0:N-1,1:3)
      REAL*8 A1(0:N-1,0:N-1,0:N-1,1:3),B1(0:N-1,0:N-1,0:N-1,1:3)
      REAL*8 A2(0:N-1,0:N-1,0:N-1,1:3),B2(0:N-1,0:N-1,0:N-1,1:3)
      REAL*8 A3(0:N-1,0:N-1,0:N-1,1:3),B3(0:N-1,0:N-1,0:N-1,1:3)
      REAL*8 A4(0:N-1,0:N-1,0:N-1,1:3),B4(0:N-1,0:N-1,0:N-1,1:3)
      REAL*8 TA1,TA2,TA3,TA4,TB1,TB2,TB3,TB4
C     VARIABILI NOME DEI FILE
      CHARACTER*32 S
      CHARACTER S1,S2
      CHARACTER*8 NOME1
C     VARIABILI COEFFICIENTI
      REAL*8 DELTAT,RE,DD,ALPHA
```

```
C     ASSEGNAZIONE COSTANTI
      WRITE(6,*)'Passo temporale '
      READ(5,*)DELTAT
      WRITE(6,*)'Numero di Reynolds '
      READ(5,*)RE
      WRITE(6,*)'Coefficiente?(ALPHA=0.063)'
      READ(5,*)ALPHA
      ALPHA=ALPHA*8*DATAN(1.0D0)
C     DIRECTORY RISULTATI
      WRITE(6,*)'(Sotto)directory in cui si salveranno i risultati:'
      WRITE(6,*)'D:\DATI\RIS\...'
      READ(5,*)NOME1
C
      IZ1=IZ2/IZ
C
      CALL LEGGI(U0,H0,N)
      DO 2 J=1,3
      DO 2 L3=0,N-1
      DO 2 L2=0,N-1
      DO 2 L1=0,N-1
      U(L1,L2,L3,J,0)=U0(L1,L2,L3,J)
2     H(L1,L2,L3,J,0)=H0(L1,L2,L3,J)
C
C     AVANZAMENTO NEL TEMPO CON RUNGE-KUTTA 4
C     IL SOTTOPROGRAMMA AVANTI DEFINISCE LE FUNZIONI A E B
C     DU/DT=A, DH/DT=B  (D/DT=DERIVATA PARZIALE NEL TEMPO)
C
C     COEFF. AVANZAMENTO
      DD=DELTAT/6.0D0
C
      WRITE(6,*)'Terminata la lettura dei dati...'
      DO 999 I1=1,IZ1
      DO 99 I2=0,(IZ-1)
      DO 10 J1=1,3
      DO 10 L3=0,N-1
      DO 10 L2=0,N-1
      DO 10 L1=0,N-1
      U0(L1,L2,L3,J1)=U(L1,L2,L3,J1,I2)
10    H0(L1,L2,L3,J1)=H(L1,L2,L3,J1,I2)
      CALL AVANTI(U0,H0,A1,B1,N,RE,ALPHA)
      DO 15 J1=1,3
      DO 15 L3=0,N-1
      DO 15 L2=0,N-1
      DO 15 L1=0,N-1
      U1(L1,L2,L3,J1)=U0(L1,L2,L3,J1)+0.5D0*DELTAT*A1(L1,L2,L3,J1)
15    H1(L1,L2,L3,J1)=H0(L1,L2,L3,J1)+0.5D0*DELTAT*B1(L1,L2,L3,J1)
      CALL AVANTI(U1,H1,A2,B2,N,RE,ALPHA)
      DO 20 J1=1,3
      DO 20 L3=0,N-1
      DO 20 L2=0,N-1
      DO 20 L1=0,N-1
      U1(L1,L2,L3,J1)=U0(L1,L2,L3,J1)+0.5D0*DELTAT*A2(L1,L2,L3,J1)
20    H1(L1,L2,L3,J1)=H0(L1,L2,L3,J1)+0.5D0*DELTAT*B2(L1,L2,L3,J1)
      CALL AVANTI(U1,H1,A3,B3,N,RE,ALPHA)
      DO 25 J1=1,3
      DO 25 L3=0,N-1
      DO 25 L2=0,N-1
      DO 25 L1=0,N-1
      U1(L1,L2,L3,J1)=U0(L1,L2,L3,J1)+DELTAT*A3(L1,L2,L3,J1)
25    H1(L1,L2,L3,J1)=H0(L1,L2,L3,J1)+DELTAT*B3(L1,L2,L3,J1)
      CALL AVANTI(U1,H1,A4,B4,N,RE,ALPHA)
      DO 30 J1=1,3
      DO 30 L3=0,N-1
      DO 30 L2=0,N-1
      DO 30 L1=0,N-1
      TA1=A1(L1,L2,L3,J1)
```

```
      TA2=A2(L1,L2,L3,J1)
      TA3=A3(L1,L2,L3,J1)
      TA4=A4(L1,L2,L3,J1)
      TB1=B1(L1,L2,L3,J1)
      TB2=B2(L1,L2,L3,J1)
      TB3=B3(L1,L2,L3,J1)
      TB4=B4(L1,L2,L3,J1)
      U(L1,L2,L3,J1,I2+1)=U(L1,L2,L3,J1,I2)+DD*(TA1+2*TA2+2*TA3+TA4)
30    H(L1,L2,L3,J1,I2+1)=H(L1,L2,L3,J1,I2)+DD*(TB1+2*TB2+2*TB3+TB4)
      WRITE(6,*)'Terminato passo temporale ',I2,'/',I1
99    CONTINUE
C     ADESSO SALVIAMO SU DISCO GLI Z PASSI TEMPORALI CALCOLATI,
C     E RIINIZIALIZZIAMO I VETTORI U ED H
      WRITE(6,*)'SCRITTURA RISULTATI SU DISCO',I1
      J1=MOD(I1,10)
      J2=MOD(I1/10,10)
      S1=CHAR(J2+ICHAR('0'))
      S2=CHAR(J1+ICHAR('0'))
      S='D:\DATI\RIS\'//NOME1//'\VEL'//S1//S2//'.BIN'
      OPEN(1,FILE=S,FORM='UNFORMATTED',ACCESS='DIRECT',
     >RECL=8,STATUS='NEW')
      IIZ=IZ/IS
      DO 50 J2=1,IIZ
      JJ2=IS*J2
DO 50 J1=1,3
      DO 50 L3=0,N-1
      DO 50 L2=0,N-1
      DO 50 L1=0,N-1
50    WRITE (1)U(L1,L2,L3,J1,JJ2)
      CLOSE (1)
      WRITE(6,*)'Completata scrittura su disco delle velocita...'
      S='D:\DATI\RIS\'//NOME1//'\ANG'//S1//S2//'.BIN'
      OPEN (2,FILE=S,FORM='UNFORMATTED',
     >ACCESS='DIRECT',RECL=8,STATUS='NEW')
      DO 60 J2=1,IIZ
      JJ2=J2*IS
      DO 60 J1=1,3
      DO 60 L3=0,N-1
      DO 60 L2=0,N-1
      DO 60 L1=0,N-1
60    WRITE (2)H(L1,L2,L3,J1,JJ2)
      CLOSE (2)
      WRITE(6,*)'Completata scrittura su disco dei momenti...'
      DO 70 J1=1,3
      DO 70 L3=0,N-1
      DO 70 L2=0,N-1
      DO 70 L1=0,N-1
      U(L1,L2,L3,J1,0)=U(L1,L2,L3,J1,IZ)
70    H(L1,L2,L3,J1,0)=H(L1,L2,L3,J1,IZ)
      WRITE(6,*)'Scrittura completata, variabili riinizializzate.'
C
999   CONTINUE
C
1000  CONTINUE
      END
C     ****************************************************************
C     FINE PROGRAMMA PRINCIPALE
C     ****************************************************************
      SUBROUTINE AVANTI(U,H,A,B,N,RE,ALPHA)
C     VARIABILI FORMALI IN INGRESSO
      REAL*8 U(0:N-1,0:N-1,0:N-1,1:3),H(0:N-1,0:N-1,0:N-1,1:3)
C     VARIABILI FORMALI IN USCITA
      REAL*8 A(0:N-1,0:N-1,0:N-1,1:3),B(0:N-1,0:N-1,0:N-1,1:3)
C     VARIABILI INTERNE
      REAL*8 G(0:N-1,0:N-1,0:N-1,1:3,1:3),A1(0:N-1,0:N-1,0:N-1)
      REAL*8 A2(0:N-1,0:N-1,0:N-1),MM(0:N-1,0:N-1,0:N-1)
```

```
      REAL*8 F(0:N-1,0:N-1,0:N-1,1:3),P(0:N-1,0:N-1,0:N-1,1:3)
      REAL*8 GC(0:N-1,0:N-1,0:N-1,1:3,1:3),DH(0:N-1,0:N-1,0:N-1),TB
      REAL*8 G1(0:N-1,0:N-1,0:N-1,3,3)
C     COEFFICIENTE DEL MODELLO
      REAL*8 COEFF,ALPHA,EPSILON,RE
C     DEFINIZIONE DEL VETTORE DEI NUMERI D'ONDA
      INTEGER KAPPA(0:N/2),KAPPAE(0:N-1)
      DO 111 I=0,N/2
      KAPPA(I)=I
111   KAPPAE(I)=I
      DO 112 I=1,N/2-1
112   KAPPAE(N-I)=I
C     DEFINIZIONE COEFFICIENTE DEL MODELLO
C     ALPHA=0.03D0
      EPSILON=DBLE(1)/N
      COEFF=ALPHA*EPSILON**2
C
      M=3*N/2
C     1-CALCOLO DI MM=MODULO DI H (NORMA EUCLIDEA IN R^3)
      CALL MMODULO(H,MM,N,M)
      CALL ELIMINAS(MM,N)
C
C     2-CALCOLO DI A
C
C     2.1-CALCOLIAMO 'MODULO DI H'*'PARTE SIMMETRICA DEL GRADIENTE DI
C     VELOCITA''.SI VALUTA SOLO LA PARTE SUPERIORE DELLA MATRICE.
      DO 3 J1=1,3
      DO 3 J2=J1,3
      DO 1 L3=0,N-1
      DO 1 L2=0,N-1
      DO 1 L1=0,N-1
1     A1(L1,L2,L3)=G(L1,L2,L3,J1,J2)+G(L1,L2,L3,J2,J1)
      CALL PRODOTTO(MM,A1,A1,N,M)
      CALL ELIMINAS(A1,N)
      DO 2 L3=0,N-1
      DO 2 L2=0,N-1
      DO 2 L1=0,N-1
2     G(L1,L2,L3,J1,J2)=A1(L1,L2,L3)
3     CONTINUE
C
C     2.2-AGGIUNGIAMO -(U_J1*U_J2)
      DO 6 J1=1,3
      DO 6 J2=J1,3
      DO 4 L3=0,N-1
      DO 4 L2=0,N-1
      DO 4 L1=0,N-1
      A1(L1,L2,L3)=U(L1,L2,L3,J1)
4     A2(L1,L2,L3)=U(L1,L2,L3,J2)
      CALL PRODOTTO(A1,A2,A1,N,M)
      CALL ELIMINAS(A1,N)
      DO 5 L3=0,N-1
      DO 5 L2=0,N-1
      DO 5 L1=0,N-1
5     G(L1,L2,L3,J1,J2)=COEFF*G(L1,L2,L3,J1,J2)-A1(L1,L2,L3)
6     CONTINUE
C
C     2.3-'RADDOPPIAMO' LA MATRICE PER SCRIVERLA TUTTA
      DO 8 J1=2,3
      DO 8 J2=1,(J1-1)
      DO 7 L3=0,N-1
      DO 7 L2=0,N-1
      DO 7 L1=0,N-1
7     G(L1,L2,L3,J1,J2)=G(L1,L2,L3,J2,J1)
8     CONTINUE
C
C     2.4-CALCOLIAMO LA DIVERGENZA DI QUESTO TENSORE
```

```
      CALL DIVT(G,F,N,KAPPA)
C
C     2.5-I SEGUENTI PASSI SERVONO PER AGGIUNGERE IL GRADIENTE DELLA
C         PRESSIONE
      CALL DIVERGENZA(F,A1,N,KAPPA)
      CALL GRAD(A1,P,N,KAPPA)
C
      DO 91 J=1,3
      A(0,0,0,J)=F(0,0,0,J)
      DO 9 L3=0,N-1
      DO 9 L2=0,N-1
      DO 9 L1=0,N-1
      IK2=KAPPAE(L1)**2+KAPPAE(L2)**2+KAPPAE(L3)**2
      IF(IK2-0)99,99,98
98    A(L1,L2,L3,J)=F(L1,L2,L3,J)+P(L1,L2,L3,J)/IK2
99    CONTINUE
9     CONTINUE
91    CONTINUE
C
C     2.6-AGGIUNGIAMO 'LAPLACIANO DI U '/'NUMERO DI REYNOLDS'
C
      CALL LAPLACEV(U,F,N,KAPPAE)
C
      DO 10 J=1,3
      DO 10 L3=0,N-1
      DO 10 L2=0,N-1
      DO 10 L1=0,N-1
10    A(L1,L2,L3,J)=A(L1,L2,L3,J)+F(L1,L2,L3,J)/RE
C
C     3-CALCOLO DI B
C
C     3.0.1 GRADIENTE DI H
      CALL GRADT(H,G,N,KAPPA)
      CALL DIVERGENZA(H,DH,N,KAPPA)
C     3.0.1 TOLGO LA TRACCIA AL GRADIENTE DI H.
      DO 201 J=1,3
      DO 201 L3=0,N-1
      DO 201 L2=0,N-1
      DO 201 L1=0,N-1
201   G(L1,L2,L3,J,J)=G(L1,L2,L3,J,J)-DH(L1,L2,L3)/3.0D0
C
C     3.1-CALCOLIAMO 'MODULO DI H'*'PARTE SIMMETRICA DEL DEL GRADIENTE DI H'
C     (CALCOLIAMO SOLTANTO LA PARTE SUPERIORE DELLA MATRICE)
      DO 13 J1=1,3
      DO 13 J2=J1,3
      DO 11 L3=0,N-1
      DO 11 L2=0,N-1
      DO 11 L1=0,N-1
11    A1(L1,L2,L3)=G(L1,L2,L3,J1,J2)+G(L1,L2,L3,J2,J1)
      CALL PRODOTTO(MM,A1,A1,N,M)
      CALL ELIMINAS(A1,N)
      DO 12 L3=0,N-1
      DO 12 L2=0,N-1
      DO 12 L1=0,N-1
12    G1(L1,L2,L3,J1,J2)=A1(L1,L2,L3)
13    CONTINUE
C
C     3.1BIS-'RADDOPPIAMO' LA MATRICE SIMMETRICA
      DO 210 J1=2,3
      DO 210 J2=1,(J1-1)
      DO 119 L3=0,N-1
      DO 119 L2=0,N-1
      DO 119 L1=0,N-1
119   G1(L1,L2,L3,J1,J2)=G1(L1,L2,L3,J2,J1)
210   CONTINUE
C
```

```
C     3.2-AGGIUNGIAMO -(H_I*U_J)+(H_J*U_I)
C     3.2.1-CALCOLO DI U_J1*H_J2=GC_J1,J2
      DO 18 J1=1,3
      DO 18 J2=1,3
      IF(J1.NE.J2)THEN
      DO 14 L3=0,N-1
      DO 14 L2=0,N-1
      DO 14 L1=0,N-1
      A1(L1,L2,L3)=U(L1,L2,L3,J1)
14    A2(L1,L2,L3)=H(L1,L2,L3,J2)
      CALL PRODOTTO(A1,A2,A1,N,M)
      CALL ELIMINAS(A1,N)
      DO 15 L3=0,N-1
      DO 15 L2=0,N-1
      DO 15 L1=0,N-1
15    GC(L1,L2,L3,J1,J2)=A1(L1,L2,L3)
      END IF
18    CONTINUE
C
C     3.2.1-SOMMIAMO I DUE CONTRIBUTI
      DO 118 J1=1,3
      DO 118 J2=1,3
      IF(J1.NE.J2)THEN
      DO 128 L3=0,N-1
      DO 128 L2=0,N-1
      DO 128 L1=0,N-1
      TB=GC(L1,L2,L3,J1,J2)-GC(L1,L2,L3,J2,J1)
128   G(L1,L2,L3,J1,J2)=COEFF*G1(L1,L2,L3,J1,J2)+TB
      ELSE
      DO 129 L3=0,N-1
      DO 129 L2=0,N-1
      DO 129 L1=0,N-1
129   G(L1,L2,L3,J1,J2)=COEFF*G1(L1,L2,L3,J1,J2)
      END IF
118   CONTINUE
C
C     3.4-PRENDIAMONE LA DIVERGENZA
      CALL DIVT(G,B,N,KAPPA)
C
C     3.5-AGGIUNGIAMO 'LAPLACIANO DI H'/'NUMERO DI REYNOLDS'
      CALL LAPLACEV(H,F,N,KAPPAE)
      DO 21 J=1,3
      DO 21 L3=0,N-1
      DO 21 L2=0,N-1
      DO 21 L1=0,N-1
21    B(L1,L2,L3,J)=B(L1,L2,L3,J)+F(L1,L2,L3,J)/RE
C
C     3.6 CORREZIONE DIVERGENZA DI H
      DO 33 J=1,3
      DO 31 L3=0,N-1
      DO 31 L2=0,N-1
      DO 31 L1=0,N-1
31    A1(L1,L2,L3)=U(L1,L2,L3,J)
      CALL PRODOTTO(A1,DH,A1,N,M)
      CALL ELIMINAS(A1,N)
      DO 32 L3=0,N-1
      DO 32 L2=0,N-1
      DO 32 L1=0,N-1
32    B(L1,L2,L3,J)=B(L1,L2,L3,J)-A1(L1,L2,L3)
33    CONTINUE
C
      RETURN
      END
C     ****************************************************************
C     LETTURA DATI DI PARTENZA
C     ----------------------------------------------------------------
```

```
      SUBROUTINE LEGGI(UI,HI,N)
      REAL*8 UI(0:N-1,0:N-1,0:N-1,1:3),HI(0:N-1,0:N-1,0:N-1,1:3)
      CHARACTER*20 SI
      CHARACTER*28 S
C     LETTURA U0 ED H0 DA FILE
      WRITE(6,*)'Nome file dati velocita, D:\DATI\INIZIO\...'
      READ(5,*)SI
      S='D:\DATI\INIZIO\'//SI
      OPEN(1,FILE=S,FORM='UNFORMATTED',
     >ACCESS='DIRECT',RECL=8,STATUS='OLD')
      DO 50 J1=1,3
      DO 50 L3=0,N-1
      DO 50 L2=0,N-1
      DO 50 L1=0,N-1
50    READ(1)UI(L1,L2,L3,J1)
      CLOSE (1)
      WRITE(6,*)'Nome file dati momento, D:\DATI\INIZIO\...'
      READ(5,*)SI
      S='D:\DATI\INIZIO\'//SI
      OPEN(2,FILE=S,FORM='UNFORMATTED',
     >ACCESS='DIRECT',RECL=8,STATUS='OLD')
      DO 60 J1=1,3
      DO 60 L3=0,N-1
      DO 60 L2=0,N-1
      DO 60 L1=0,N-1
60    READ(2)HI(L1,L2,L3,J1)
      CLOSE (2)
      RETURN
      END
C     ****************************************************************
```

Sottoprogrammi

Si riportano i sottoprogrammi descritti nel paragrafo 3 ed usati in entrambi i codici. Per i sottoprogrammi NAG si vedano gli appositi manuali.

```
C     ****************************************************************
C     SOTTOPROGRAMMI PER IL CALCOLO DEGLI OPERATORI DIFFERENZIALI:
C     DERIVATA: CALCOLA LA DERIVATA PARZIALE
C     DIVERGENZA:DIVERGENZA DI U  VETTORE
C     DIVT: DIVERGENZA DI UN TENSORE
C     GRAD: GRADIENTE DI UNO SCALARE
C     GRADT: GRADIENTE DI UN VETTORE
C     ----------------------------------------------------------------
      SUBROUTINE DERIVATA(F,J,DF,N,KAPPA)
C     CALCOLA LA DERIVATA PARZIALE DI F RISPETTO AD X_J
      REAL*8 F(0:N-1,0:N-1,0:N-1),DF(0:N-1,0:N-1,0:N-1)
      INTEGER KAPPA(0:N/2)
C     ESECUZIONE DERIVATA
C     1-SELEZIONE VARIABILE RISPETTO A CUI DERIVARE
C     2- MOLTIPLICAZIONE PEI NUMERI D'ONDA OPPORTUNI
C     ( SI ESEGUE UNO DEI TRE BLOCCHI ETICHETTATI 991-992-993)
      IF(J-2)1,2,3
1     DO 991 K3=0,N-1
      DO 991 K2=0,N-1
      DF(0,K2,K3)=0.D0
      DO 91 K1=1,(N/2-1)
      DF(K1,K2,K3)=-KAPPA(K1)*F(N-K1,K2,K3)
91    DF(N-K1,K2,K3)=KAPPA(K1)*F(K1,K2,K3)
991   DF(N/2,K2,K3)=0.D0
      GOTO 999
C
2     DO 992 K3=0,N-1
      DO 992 K1=0,N-1
      DF(K1,0,K3)=0.D0
      DO 92 K2=1,(N/2-1)
      DF(K1,K2,K3)=-KAPPA(K2)*F(K1,N-K2,K3)
92    DF(K1,N-K2,K3)=KAPPA(K2)*F(K1,K2,K3)
992   DF(K1,N/2,K3)=0.D0
      GOTO 999
C
3     DO 993 K2=0,N-1
      DO 993 K1=0,N-1
      DF(K1,K2,0)=0.D0
      DO 93 K3=1,(N/2-1)
      DF(K1,K2,K3)=-KAPPA(K3)*F(K1,K2,N-K3)
93    DF(K1,K2,N-K3)=KAPPA(K3)*F(K1,K2,K3)
993   DF(K1,K2,N/2)=0.D0
999   CONTINUE
C
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE DIVERGENZA(U,DIVU,N,KAPPA)
      REAL*8 U(0:N-1,0:N-1,0:N-1,1:3),DIVU(0:N-1,0:N-1,0:N-1)
      REAL*8 F(0:N-1,0:N-1,0:N-1),DF(0:N-1,0:N-1,0:N-1)
C
      INTEGER KAPPA(0:N/2)
C
      DO 19 L3=0,N-1
      DO 19 L2=0,N-1
      DO 19 L1=0,N-1
19    DIVU(L1,L2,L3)=0.D0
C
      DO 2 J=1,3
      DO 3 J3=0,N-1
      DO 3 J2=0,N-1
```

```
      DO 3 J1=0,N-1
3     F(J1,J2,J3)=U(J1,J2,J3,J)
      CALL DERIVATA(F,J,DF,N,KAPPA)
      DO 4 J3=0,N-1
      DO 4 J2=0,N-1
      DO 4 J1=0,N-1
4     DIVU(J1,J2,J3)=DIVU(J1,J2,J3)+DF(J1,J2,J3)
2     CONTINUE
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE DIVT2(T,DT,N,K)
C     CALCOLA LA DIVERGENZA DI UN TENSORE SIMMETRICO T
      REAL*8 T(0:N-1,0:N-1,0:N-1,1:6)
      REAL*8 DT(0:N-1,0:N-1,0:N-1,1:3)
      INTEGER K(0:N/2)
      REAL*8 A(0:N-1,0:N-1,0:N-1,1:3),B(0:N-1,0:N-1,0:N-1)
C
      DO 1 JS=1,3
      IF(JS-2)101,102,103
101   JA=1
      JB=2
      JC=3
      GOTO 109
102   JA=2
      JB=4
      JC=5
      GOTO 109
103   JA=3
      JB=5
      JC=6
109   CONTINUE
      DO 2 L3=0,N-1
      DO 2 L2=0,N-1
      DO 2 L1=0,N-1
      B(L1,L2,L3)=0.0D0
      A(L1,L2,L3,1)=T(L1,L2,L3,JA)
      A(L1,L2,L3,2)=T(L1,L2,L3,JB)
2     A(L1,L2,L3,3)=T(L1,L2,L3,JC)
C
      CALL DIVERGENZA(A,B,N,K)
C
      DO 3 L3=0,N-1
      DO 3 L2=0,N-1
      DO 3 L1=0,N-1
      DT(L1,L2,L3,JS)=B(L1,L2,L3)
3     CONTINUE
1     CONTINUE
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE GRAD(P,GP,N,K)
      REAL*8 P(0:N-1,0:N-1,0:N-1),GP(0:N-1,0:N-1,0:N-1,1:3)
      INTEGER K(0:N/2)
      REAL*8 FD(0:N-1,0:N-1,0:N-1)
C
      DO 1 J=1,3
      CALL DERIVATA(P,J,FD,N,K)
      DO 2 L3=0,N-1
      DO 2 L2=0,N-1
      DO 2 L1=0,N-1
2     GP(L1,L2,L3,J)=FD(L1,L2,L3)
1     CONTINUE
      RETURN
      END
C     ----------------------------------------------------------------
```

```
      SUBROUTINE GRADT(U,GU,N,K)
      REAL*8 U(0:N-1,0:N-1,0:N-1,1:3),GU(0:N-1,0:N-1,0:N-1,1:3,1:3)
      INTEGER K(0:N/2)
      REAL*8 F(0:N-1,0:N-1,0:N-1),DF(0:N-1,0:N-1,0:N-1)
C
      DO 1 J1=1,3
      DO 2 L3=0,N-1
      DO 2 L2=0,N-1
      DO 2 L1=0,N-1
2     F(L1,L2,L3)=U(L1,L2,L3,J1)
      DO 1 J2=1,3
      CALL DERIVATA(F,J2,DF,N,K)
      DO 3 L3=0,N-1
      DO 3 L2=0,N-1
      DO 3 L1=0,N-1
3     GU(L1,L2,L3,J1,J2)=DF(L1,L2,L3)
1     CONTINUE
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE LAPLACE(F,LF,N,KAPPA)
      REAL*8 F(0:N-1,0:N-1,0:N-1),LF(0:N-1,0:N-1,0:N-1)
      INTEGER KAPPA(0:N-1)
C     CALCOLA IL LAPLACIANO DI UNO SCALARE F.
C     VIENE CALCOLATO DIRETTAMENTE SENZA FAR USO DEI SOTTOPROGRAMMI
C     DIVERGENZA E GRADIENTE
      DO 1 K3=0,N-1
      DO 1 K2=0,N-1
      DO 1 K1=0,N-1
      IK=KAPPA(K1)**2+KAPPA(K2)**2+KAPPA(K3)**2
1     LF(K1,K2,K3)=-F(K1,K2,K3)*IK
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE LAPLACEV(U,LU,N,KAPPA)
      REAL*8 U(0:N-1,0:N-1,0:N-1,1:3),LU(0:N-1,0:N-1,0:N-1,1:3)
      REAL*8 A(0:N-1,0:N-1,0:N-1)
      INTEGER KAPPA(0:N-1)
C     CALCOLA IL LAPLACIANO DI UN VETTORE U USANDO IL SOTTOPROGRAMMA
C     LAPLACE.
      DO 3 J=1,3
      DO 1 L3=0,N-1
      DO 1 L2=0,N-1
      DO 1 L1=0,N-1
1     A(L1,L2,L3)=U(L1,L2,L3,J)
      CALL LAPLACE(A,A,N,KAPPA)
      DO 2 L3=0,N-1
      DO 2 L2=0,N-1
      DO 2 L1=0,N-1
2     LU(L1,L2,L3,J)=A(L1,L2,L3)
3     CONTINUE
      RETURN
      END
C     ****************************************************************
C     SOTTOPROGRAMMI PER IL CALCOLO DEL PRODOTTO SENZA ALYASING
C     PRODOTTO: EFFETTUA MATERIALMENTE IL PRODOTTO
C     TRONCA-ESPANDI-SEL: UTILIZZATI IN PRODOTTO
C     ----------------------------------------------------------------
      SUBROUTINE PRODOTTO(U,V,W,N,M)
C     SI CALCOLA W=U*V, USANDO UN ALGORITMO CHE ELIMINA L'ALYASING
C     (L'ALGORITMO DELL'ESPANSIONE DA N A 3N/2)
      REAL*8 U(0:N-1,0:N-1,0:N-1),V(0:N-1,0:N-1,0:N-1)
      REAL*8 W(0:N-1,0:N-1,0:N-1)
      REAL*8 UE(0:M-1,0:M-1,0:M-1),VE(0:M-1,0:M-1,0:M-1)
C
C     0-AZZERAMENTO DELLE VARIABILI(NON SI SA MAI)
```

```
      DO 1 L3=0,M-1
      DO 1 L2=0,M-1
      DO 1 L1=0,M-1
      UE(L1,L2,L3)=0.0D0
1     VE(L1,L2,L3)=0.0D0
C     1-ESPANSIONE AD M PUNTI
      CALL ESPANDI(U,UE,N,M)
      CALL ESPANDI(V,VE,N,M)
C     2-ANTITRASFORMAZIONE
      CALL ATRASF3S(UE,M)
      CALL ATRASF3S(VE,M)
C     3-ESECUZIONE PRODOTTO
      DO 2 L3=0,M-1
      DO 2 L2=0,M-1
      DO 2 L1=0,M-1
2     UE(L1,L2,L3)=UE(L1,L2,L3)*VE(L1,L2,L3)
C     4-TRASFORMAZIONE
      CALL TRASF3S(UE,M)
      CALL TRONCA(UE,W,N,M)
      RETURN
      END
C     ---------------------------------------------------------------
      SUBROUTINE TRONCA(FE,FT,N,M)
C     FE VIENW TRONCATO AD FT
C     NOTA: N<M
      REAL*8 FE(0:M-1,0:M-1,0:M-1),FT(0:N-1,0:N-1,0:N-1)
      DO 2 L3=0,N-1
      CALL SEL(L3,LL3,N,M)
      DO 2 L2=0,N-1
      CALL SEL(L2,LL2,N,M)
      DO 2 L1=0,N-1
      CALL SEL(L1,LL1,N,M)
      FT(L1,L2,L3)=FE(LL1,LL2,LL3)
2     CONTINUE
      RETURN
      END
C     ---------------------------------------------------------------
      SUBROUTINE ESPANDI(FT,FE,N,M)
      REAL*8 FT(0:N-1,0:N-1,0:N-1),FE(0:M-1,0:M-1,0:M-1)
C     FT VIENE SPANSO AD FE
C     NOTA:N<M
      DO 2 L3=0,N-1
      CALL SEL(L3,LL3,N,M)
      DO 2 L2=0,N-1
      CALL SEL(L2,LL2,N,M)
      DO 2 L1=0,N-1
      CALL SEL(L1,LL1,N,M)
      FE(LL1,LL2,LL3)=FT(L1,L2,L3)
2     CONTINUE
      RETURN
      END
C     ---------------------------------------------------------------
      SUBROUTINE SEL(J,JJ,N,M)
C     SELEZIONA GLI INDICI PER LA TRONCATURA E L'ESPANSIONE
      IF(J-N/2)1,2,2
1     JJ=J
      GOTO 3
2     JJ=M-N+J
3     CONTINUE
      RETURN
      END
C     ***************************************************************
C     SOTTOPROGRAMMA PER IL CALCOLO DEL MODULO
C     ---------------------------------------------------------------
      SUBROUTINE MMODULO(F,FM,N,M)
C     SI CALCOLA IL MODULO FM DI F, EVITANDO L'ALYASING, IN MODO ANALOGO
```

```
C     A QUANTO SI EFFETTUA NEL SOTTOPROGRAMMA PRODOTTO
      REAL*8 F(0:N-1,0:N-1,0:N-1,1:3),FM(0:N-1,0:N-1,0:N-1)
C     VARIABILI AUSILIARIE
      REAL*8 A(0:N-1,0:N-1,0:N-1),AE(0:M-1,0:M-1,0:M-1)
      REAL*8 B(0:M-1,0:M-1,0:M-1,1:3)
C     1-SI ESPANDE E SI ANTITRASFORMA OGNI COMPONENTE DI F
C     METTENDO IL RISULATATO IN B(CHE E' F ESPANSO)
      DO 2 J=1,3
      DO 1 L3=0,N-1
      DO 1 L2=0,N-1
      DO 1 L1=0,N-1
1     A(L1,L2,L3)=F(L1,L2,L3,J)
      DO 10 L3=0,M-1
      DO 10 L2=0,M-1
      DO 10 L1=0,M-1
10    AE(L1,L2,L3)=0.D0
      CALL ESPANDI(A,AE,N,M)
      CALL ATRASF3S(AE,M)
      DO 3 L3=0,M-1
      DO 3 L2=0,M-1
      DO 3 L1=0,M-1
3     B(L1,L2,L3,J)=AE(L1,L2,L3)
2     CONTINUE
C     2-ADESSO SI QUADRANO E SI SOMMANO LE COMPONENTI DI B IN OGNI PUNTO
      DO 5 L3=0,M-1
      DO 5 L2=0,M-1
      DO 5 L1=0,M-1
      AE(L1,L2,L3)=0.D0
      DO 4 J=1,3
4     AE(L1,L2,L3)=AE(L1,L2,L3)+B(L1,L2,L3,J)**2
C     PRENDO LA RADICE
5     AE(L1,L2,L3)=DSQRT(AE(L1,L2,L3))
C     RITRASFORMO E TRONCO
      CALL TRASF3S(AE,M)
      CALL TRONCA(AE,FM,N,M)
      RETURN
      END
C     ****************************************************************
C     SOTTOPROGRAMMI PER IL CALCOLO DELLA TRASFORMATA E DELLA
C     ANTITRASFORMATA.
C     TRASF1X: CALCOLA LA TRASFORMATA IN UNA DIMENSIONE DI UNA SEQUENZA
C     REALI.
C     TRASF3X: CALCOLA LA TRASFORMATA IN 3D DI NUMERI REALI.
C     *I RISULTATI SONO POSTI IN FORMA HERMITIANA*
C     ATRASF1: ANTITRASFORMATA 1D DI DATI POSTI IN FORMA HERMITIANA
C     ATRASF3: ANTITRAASFORMATA 3D DI DATI IN FORMA HERMITIANA
C     *I RISULTATI SONO NUMERI REALI*
C     ----------------------------------------------------------------
      SUBROUTINE TRASF1(F,N)
      REAL*8 F(0:N-1),WORK(N)
C     CALCOLO DELLA TRASFORMATA UNIDIMENSIONALE DI F.
C     I RISULTATI VENGONO MESSI NELLA VARIABILE F.
C     SI USA C06FAF DELLA NAG.
      IFAIL=0
      CALL C06FAF(F,N,WORK,IFAIL)
CC     CORREZIONE
C      DO 1 J=0,N-1
C1     F(J)=F(J)/(N**0.5D0)
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE TRASF3X(U,UT,N)
      REAL*8 U(0:N-1,0:N-1,0:N-1),UT(0:N-1,0:N-1,0:N-1)
      REAL*8 F(0:N-1),CORR
C     SI CALCOLA LA TRASFORMATA UT IN 3D DI UNO SCALARE U SU
C     N PUNTI, USANDO TRASF1X PER LA TRASFORMATA UNIDIMENSIONALE
```

```
C     1-COPIA DEI DATI
      DO 1 J3=0,N-1
      DO 1 J2=0,N-1
      DO 1 J1=0,N-1
1     UT(J1,J2,J3)=U(J1,J2,J3)
C     2-ESECUZIONE TRASFORMATE:
C     BLOCCO 1-TRASFORMATE NELLA DIREZIONE 1
      DO 10 J3=0,N-1
      DO 10 J2=0,N-1
      DO 11 J1=0,N-1
11    F(J1)=UT(J1,J2,J3)
      CALL TRASF1(F,N)
      DO 12 K1=0,N-1
12    UT(K1,J2,J3)=F(K1)
10    CONTINUE
C     BLOCCO 2-TRASFORMATE NELLA DIREZIONE 2
      DO 20 J3=0,N-1
      DO 20 K1=0,N-1
      DO 21 J2=0,N-1
21    F(J2)=UT(K1,J2,J3)
      CALL TRASF1(F,N)
      DO 22 K2=0,N-1
22    UT(K1,K2,J3)=F(K2)
20    CONTINUE
C     BLOCCO 3-TRASFORMATE NELLA DIREZIONE 3
      DO 30 K2=0,N-1
      DO 30 K1=0,N-1
      DO 31 J3=0,N-1
31    F(J3)=UT(K1,K2,J3)
      CALL TRASF1(F,N)
      DO 32 K3=0,N-1
32    UT(K1,K2,K3)=F(K3)
30    CONTINUE
C     CORREZIONE COEFFICIENTE
      CORR=1.0D0/(N**1.5D0)
      DO 90 K3=0,N-1
      DO 90 K2=0,N-1
      DO 90 K1=0,N-1
90    UT(K1,K2,K3)=UT(K1,K2,K3)*CORR
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE TRASF3S(UT,N)
      REAL*8 UT(0:N-1,0:N-1,0:N-1)
      REAL*8 F(0:N-1),CORR
C     IN USCITA LA TRASFORMATA UT IN 3D DI UNO SCALARE UT SU
C     N PUNTI, USANDO TRASF1 PER LA TRASFORMATA UNIDIMENSIONALE
C     VERSIONE OTTIMIZZATA
C     2-ESECUZIONE TRASFORMATE:
C     BLOCCO 1-TRASFORMATE NELLA DIREZIONE 1
      DO 10 J3=0,N-1
      DO 10 J2=0,N-1
      DO 11 J1=0,N-1
11    F(J1)=UT(J1,J2,J3)
      CALL TRASF1(F,N)
      DO 12 K1=0,N-1
12    UT(K1,J2,J3)=F(K1)
10    CONTINUE
C     BLOCCO 2-TRASFORMATE NELLA DIREZIONE 2
      DO 20 J3=0,N-1
      DO 20 K1=0,N-1
      DO 21 J2=0,N-1
21    F(J2)=UT(K1,J2,J3)
      CALL TRASF1(F,N)
DO 22 K2=0,N-1
22    UT(K1,K2,J3)=F(K2)
```

```
20    CONTINUE
C     BLOCCO 3-TRASFORMATE NELLA DIREZIONE 3
      DO 30 K2=0,N-1
      DO 30 K1=0,N-1
      DO 31 J3=0,N-1
31    F(J3)=UT(K1,K2,J3)
      CALL TRASF1(F,N)
      DO 32 K3=0,N-1
32    UT(K1,K2,K3)=F(K3)
30    CONTINUE
C     CORREZIONE COEFFICIENTE
      CORR=1.0D0/(N**1.5D0)
      DO 90 K3=0,N-1
      DO 90 K2=0,N-1
      DO 90 K1=0,N-1
90    UT(K1,K2,K3)=UT(K1,K2,K3)*CORR
      RETURN
      END
C     ---------------------------------------------------------------
C     ---------------------------------------------------------------
      SUBROUTINE ATRASF1(F,N)
      REAL*8 F(0:N-1),WORK(N)
C     CALCOLO DELLA ANTITRASFORMATA 1D DI UNO SCALARE F,
C     USANDO C06GBF(CONIUGATO HERMITIANO) E
C     C06FBF (PER LA TRASFORMATA VERA E PROPRIA).
      IFAIL=0
      CALL C06GBF(F,N,IFAIL)
      CALL C06FBF(F,N,WORK,IFAIL)
C     CORREZIONE
C      DO 1 J=0,N-1
C1     F(J)=F(J)*(N**0.5D0)
      RETURN
      END
C     ---------------------------------------------------------------
      SUBROUTINE ATRASF3X(UT,U,N)
C     ESEGUE ANTITRASFORMATA REALE IN 3D
      REAL*8 UT(0:N-1,0:N-1,0:N-1)
      REAL*8 U(0:N-1,0:N-1,0:N-1)
      REAL*8 F(0:N-1),CORR
C     1-COPIA DI UT IN U
      DO 1 J3=0,N-1
      DO 1 J2=0,N-1
      DO 1 J1=0,N-1
1     U(J1,J2,J3)=UT(J1,J2,J3)
C     2-ESECUZIONE ANTITRASFORMATE
C     BLOCCO 1-ANTITRASFORMATE DIREZIONE 1
      DO 10 J3=0,N-1
      DO 10 J2=0,N-1
      DO 11 J1=0,N-1
11    F(J1)=U(J1,J2,J3)
      CALL ATRASF1(F,N)
      DO 12 K1=0,N-1
12    U(K1,J2,J3)=F(K1)
10    CONTINUE
C     BLOCCO 2-ANTITRASFORMATE DIREZIONE 2
      DO 20 J3=0,N-1
      DO 20 K1=0,N-1
      DO 21 J2=0,N-1
21    F(J2)=U(K1,J2,J3)
      CALL ATRASF1(F,N)
      DO 22 K2=0,N-1
22    U(K1,K2,J3)=F(K2)
20    CONTINUE
C     BLOCCO 3-ANTITRASFORMATE DIREZIONE 3
      DO 30 K2=0,N-1
      DO 30 K1=0,N-1
```

```
      DO 31 J3=0,N-1
31    F(J3)=U(K1,K2,J3)
      CALL ATRASF1(F,N)
      DO 32 K3=0,N-1
32    U(K1,K2,K3)=F(K3)
30    CONTINUE
C     CORREZIONE
      CORR=N**1.5D0
      DO 90 K3=0,N-1
      DO 90 K2=0,N-1
      DO 90 K1=0,N-1
90    U(K1,K2,K3)=U(K1,K2,K3)*CORR
      RETURN
      END
C     ---------------------------------------------------------------
      SUBROUTINE ATRASF3S(U,N)
C     ESEGUE ANTITRASFORMATA REALE IN 3D
      REAL*8 U(0:N-1,0:N-1,0:N-1)
      REAL*8 F(0:N-1),CORR
C     VERSIONE OTTIMIZZATA PER L'USO ESCLUSIVO IN COMBINAZIONE CON I
C     SOTTOPROGRAMMI PRODOTTO O MODULO
C     1-DEFINIZIONE INDICI
      NRID=2*N/3
C     2-ESECUZIONE ANTITRASFORMATE
C     BLOCCO 1-ANTITRASFORMATE DIREZIONE 1
      DO 10 J3=0,NRID-1
      CALL SEL(J3,JJ3,NRID,N)
      DO 10 J2=0,NRID-1
      CALL SEL(J2,JJ2,NRID,N)
      DO 11 J1=0,N-1
11    F(J1)=U(J1,JJ2,JJ3)
      CALL ATRASF1(F,N)
      DO 12 K1=0,N-1
12    U(K1,JJ2,JJ3)=F(K1)
10    CONTINUE
C     BLOCCO 2-ANTITRASFORMATE DIREZIONE 2
      DO 20 J3=0,NRID-1
      CALL SEL(J3,JJ3,NRID,N)
      DO 20 K1=0,N-1
      DO 21 J2=0,N-1
21    F(J2)=U(K1,J2,JJ3)
      CALL ATRASF1(F,N)
      DO 22 K2=0,N-1
22    U(K1,K2,JJ3)=F(K2)
20    CONTINUE
C     BLOCCO 3-ANTITRASFORMATE DIREZIONE 3
      DO 30 K2=0,N-1
      DO 30 K1=0,N-1
      DO 31 J3=0,N-1
31    F(J3)=U(K1,K2,J3)
      CALL ATRASF1(F,N)
      DO 32 K3=0,N-1
32    U(K1,K2,K3)=F(K3)
30    CONTINUE
C     CORREZIONE
      CORR=N**1.5D0
      DO 90 K3=0,N-1
      DO 90 K2=0,N-1
      DO 90 K1=0,N-1
90    U(K1,K2,K3)=U(K1,K2,K3)*CORR
      RETURN
      END
C     ****************************************************************
C     ALTRI SOTTOPROGRAMMI
C     ---------------------------------------------------------------
      SUBROUTINE ELIMINAS(U,N)
```

```
C     ELIMINA LA COMPONENTE "SPURIA"
      REAL*8 U(0:N-1,0:N-1,0:N-1)
C
      DO 1 L3=0,N-1
      DO 1 L2=0,N-1
1     U(N/2,L2,L3)=0.0D0
C
      DO 2 L3=0,N-1
      DO 2 L1=0,N-1
2     U(L1,N/2,L3)=0.0D0
C
      DO 3 L2=0,N-1
      DO 3 L1=0,N-1
3     U(L1,L2,N/2)=0.0D0
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE ELIMINAV(U,N)
C     ELIMINA LA COMPONENTE "SPURIA"
      REAL*8 U(0:N-1,0:N-1,0:N-1,1:3)
C
      DO 99 J=1,3
      DO 1 L3=0,N-1
      DO 1 L2=0,N-1
1     U(N/2,L2,L3,J)=0.0D0
C
      DO 2 L3=0,N-1
      DO 2 L1=0,N-1
2     U(L1,N/2,L3,J)=0.0D0
C
      DO 3 L2=0,N-1
      DO 3 L1=0,N-1
3     U(L1,L2,N/2,J)=0.0D0
99    CONTINUE
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE ELIMINAD(U,N,KAPPA,KAPPAE)
C     ELIMINA LA DIVERGENZA DA UN VETTORE
      REAL*8 U(0:N-1,0:N-1,0:N-1,1:3),DU(0:N-1,0:N-1,0:N-1)
      REAL*8 PHI(0:N-1,0:N-1,0:N-1),GPHI(0:N-1,0:N-1,0:N-1,1:3)
      INTEGER KAPPA(N/2),KAPPAE(0:N-1)
C
      CALL DIVERGENZA(U,DU,N,KAPPA)
      DO 91 J=1,3
      PHI(0,0,0)=0.0D0
      DO 9 L3=0,N-1
      DO 9 L2=0,N-1
      DO 9 L1=0,N-1
      IK2=KAPPAE(L1)**2+KAPPAE(L2)**2+KAPPAE(L3)**2
      IF(IK2-0)99,99,98
98    PHI(L1,L2,L3)=DU(L1,L2,L3)/IK2
99    CONTINUE
9     CONTINUE
91    CONTINUE
C
      CALL GRAD(PHI,GPHI,N,KAPPA)
      DO 1 J=1,3
      DO 1 L3=0,N-1
      DO 1 L2=0,N-1
      DO 1 L1=0,N-1
1     U(L1,L2,L3,J)=U(L1,L2,L3,J)+GPHI(L1,L2,L3,J)
C     CONTROLLO
      CALL DIVERGENZA(U,DU,N,KAPPA)
      RETURN
      END
```