Dimensional Splitting, Operator Splitting

Splitting techniques are another way to handle multiple dimensions. Unsplit MoL has some advantages, but so does operator splitting. Moreover, operator splitting is still the only viable choice for some applications (e.g. when the hydro is coupled to "stiff" equations for nuclear reactions or radiation transport).

Dimensional Splitting, Operator Splitting

In abstract notation, we can write a time-evolution problem like the equations of hydrodynamics as an equation with a *differential operator* $\mathcal{L}$, which contains all terms aside from the time derivative:

$$\frac{\partial \mathbf{U}}{\partial t} = \mathcal{L}[\mathbf{U}].$$

Formally, the solution to this equation can be written in terms of the exponential of the operator $\mathcal{L}$:

$$\mathbf{U}(t + \Delta t) = (\exp \mathcal{L})[\mathbf{U}(0)] = (1 + \Delta t \mathcal{L} + \frac{\Delta t^2}{2} \mathcal{L}_1^2 + \ldots)[\mathbf{U}(t)].$$

For the advection equation, this would look as follows:

$$\mathcal{L} = -v \frac{\partial}{\partial x}$$

And the solution would look like:

$$U(t + \delta t) = (1 + v\, \Delta t \frac{\partial}{\partial x} + \frac{v^2 \Delta t^2}{2} \frac{\partial^2}{\partial x^2} + \ldots)[U(t)].$$

Exercise: How is this related to the explicit solution of the advection equation that we discused earlier?

Dimensional Splitting

We can view our numerical solution as the result of an operator acting on the hydrodynamic state on the previous time step. If our solution is accurate to $\mathcal{O}(\Delta t^n)$ in time, then this means that our numerical solution operator agrees with

$$(\exp \mathcal{L})[\mathbf{U}(0)] = (1 + \Delta t \mathcal{L} + \frac{\Delta t^2}{2} \mathcal{L}_1^2 + \ldots)[\mathbf{U}(t)].$$

up to the term in Δt^n.

Dimensional Splitting

For the sake of simplicity, consider the hydro equations in 2D (with the $\partial/\partial z$ terms omitted) without gravitational source terms. We could then collect the terms into two differential operators $\mathcal{L}_1$ and $\mathcal{L}_2$ containing the $\partial/\partial x$ and $\partial/\partial x$ terms respectively:

$$\frac{\partial \mathbf{U}}{\partial t} = \mathcal{L}_1[\mathbf{U}] + \mathcal{L}_2[\mathbf{U}]$$

Now the solution will be:

$$e^{(\mathcal{L}_1+\mathcal{L}_2)\Delta t} = 1 + \mathcal{L}_1\Delta t + \mathcal{L}_2\Delta t + \frac{\Delta t^2}{t}\left(\mathcal{L}_1^2 + \mathcal{L}_2^2 + \mathcal{L}_1\mathcal{L}_2 + \mathcal{L}_2\mathcal{L}_1\right) + \ldots$$

Suppose we already have a solution for $\partial \mathbf{U}/\partial y = \mathcal{L}_1[\mathbf{U}]$ and $\partial \mathbf{U}/\partial x = \mathcal{L}_2[\mathbf{U}]$ (which just means that we can solve the hydro equations in 1D). Then we might try to approximate the full solution as follows:

- Do one time step Δt ignoring the $\partial/\partial y$ terms (i.e. ignoring the operator $\mathcal{L}_2$.
- Do one time step Δt ignoring the $\partial/\partial x$ terms (i.e. ignoring the operator $\mathcal{L}_1$.

This strategy is called *operator splitting* (or *dimensional splitting* in the special case of splitting operators containing derivatives with respect of the different coordinates). It can be applied to arbitrarily many operators, provided that these meet certain conditions.

Dimensional Splitting

x-sweep: Solve the following equation for one time step Δt:

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \frac{\partial \rho v_x}{\partial x} &= 0 \\
\frac{\partial \rho v_x}{\partial t} + \frac{\partial \rho v_x^2}{\partial x} + \frac{\partial P}{\partial x} &= 0 \\
\frac{\partial \rho v_y}{\partial t} + \frac{\partial \rho v_x v_y}{\partial x} &= 0 \\
\frac{\partial \rho e}{\partial t} + \frac{\partial (\rho e + P) v_x}{\partial x} &= 0
\end{aligned}$$

y-sweep: Solve the following equation for one time step Δt:

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \frac{\partial \rho v_y}{\partial y} &= 0 \\
\frac{\partial \rho v_x}{\partial t} + \frac{\partial \rho v_x v_y}{\partial y} &= 0 \\
\frac{\partial \rho v_y}{\partial t} + \frac{\partial \rho v_y^2}{\partial y} &= 0 \\
\frac{\partial \rho e}{\partial t} + \frac{\partial (\rho e + P) v_y}{\partial y} &= 0
\end{aligned}$$

Dimensional Splitting

This amounts to using the operator

$$
\begin{aligned}
e^{\mathcal{L}_1 \Delta t} e^{\mathcal{L}_2\ \Delta t} &= (1 + \mathcal{L}_1 \Delta t + \mathcal{L}_1^2 \frac{\Delta t^2}{t} + \ldots)(1 + \mathcal{L}_2 \Delta t + \mathcal{L}_2^2 \frac{\Delta t^2}{t} + \ldots) \\
&= 1 + \mathcal{L}_1 \Delta t + \mathcal{L}_2 \Delta t + \mathcal{L}_1^2 \frac{\Delta t^2}{t} + \mathcal{L}_2^2 \frac{\Delta t^2}{t} + +2\mathcal{L}_1 \mathcal{L}_2 \frac{\Delta t^2}{2} + \ldots\ldots)
\end{aligned}
$$

instead of $e^{(\mathcal{L}_1 + \mathcal{L}_2)\Delta t}$. If $2\mathcal{L}_1\mathcal{L}_2 = \mathcal{L}_1\mathcal{L}_2 + \mathcal{L}_2\mathcal{L}_1$, then we're fine, but *in general operators do not commute:* $\mathcal{L}_1 \neq \mathcal{L}_2$ (remember quantum mechanics)! Hence the operator-split solution agrees with the true solution only up to order Δt – it's only first order accurate.

Strang Splitting

An idea by Strang helps to recover 2nd order accuracy in time (if the individual sweeps are 2nd order accurate):

- Do an x-sweep with time step $\Delta t/2$.
- Do a y-sweep with time step Δt (or *two* y-sweeps with time step $\Delta t/2$).
- Do an x-sweep with time step $\Delta t/2$.

In operator notation, the time evolution operator becomes $e^{\mathcal{L}_1 \Delta t/2} e^{\mathcal{L}_2 \Delta t} e^{\mathcal{L}_1 \Delta t/2}$.
Exercise: Prove that the resulting scheme is 2nd order accurate in time.

Splitting – General strategy

- Do an x-sweep with time step $\Delta t/2$.
- Do a y-sweep with time step $\Delta t/2$.
- Do a z-sweep with time step $\Delta t/2$.
- Apply source terms (gravity, heating/cooling, nuclear burning), time step Δt.
- Do a z-sweep with time step $\Delta t/2$.
- Do a y-sweep with time step $\Delta t/2$.
- Do an x-sweep with time step $\Delta t/2$.

Sweeps: Practical Implementation

Dimensional splitting conveniently allows a solution of the hydro equations along individual "pencils" with constant (y, z), (x, z) and (x, y) in each sweep. This suggests the following implementation, e.g., for the x-sweep:

for all j,k **do**
  Get $\mathbf{U}_{1\ldots N_x,j,k}$ and copy to a temporary array $\tilde{\mathbf{U}}_{1\ldots N_x}$
  Reconstruct interface states using $\tilde{\mathbf{U}}_{1\ldots N_x}$
  Solve Riemann problems and compute fluxes
  Update $\tilde{\mathbf{U}}_{1\ldots N_x}$
  Copy $\tilde{\mathbf{U}}_{1\ldots N_x}$ back to $\mathbf{U}_{1\ldots N_x,j,k}$
end for

This has two advantages:

- We can recycle a 1D hydro solver and don't need to re-arrange its internal data layout (no additional array dimensions).
- The small amount of data for a 1D sweep can be stored in the low-level caches of a CPU, and can be accessed and worked on much faster by the CPU than data that needs to be accessed from the main memory.

Practical Issues: Memory Requirements

- Suppose the data we store for each cell comprises density, pressure, energy density, and the three velocity components (6 numbers per cell) in double precision.
- For a 3D simulation with 1024^3 cells, we thus need at least 6 GB main memory.
- In a hydro solver with dimensional splitting and sweeps along pencils, we roughly get away with this – temporary arrays don't take up much space.
- For an unsplit 2nd order MoL solver, we need to store the fluxes (3×5 numbers per cell, actually (6×5) per cell with 2 shared between neighbouring cells), and the intermediate results for the state variables after the first Runge-Kutta step.
- This means: 26 numbers per cell, 26GB in total (although we could save some memory if we're smart).

Practical issues: Why do we need higher-order accuracy

- Suppose we have a cubical numerical domain with N^3 cells, and suppose we want to calculate for at least one sound-crossing time-scale $t_{\rm sound}$ (which is not much).
- Since $\Delta t < \Delta x / c_s \approx t_{\rm sound}/N$, we need N time steps.
- With $\sim$50 floating point operations per grid cell per time step, the total number of operations is $\sim 50 N^4$.
- For a sustained performance of 5 GFLOPS per core we need:
  - $N = 10000$: 1157 days
  - $N = 1000$: 2.8 hours

  on one core.
- If a higher order method saves us a factor of ten in grid resolution to get the same accuracy, this clearly pays off even if it's more expensive per step by a factor of 10 or even 100.
- Remember: The total number of operations in 3D scales with N^4!
- More physics makes this more expensive, so we're talking about $\mathcal{O}(10^4)$ core-hours or more for 3D simulations.

Parallelisation

- The huge memory and computer time requirements of multi-D simulations imply that we need to distribute the data and the computational load across multiple CPUs.
- We can't cover this in this course, but we can outline the strategy.
- There are two basic paradigms: *shared-memory parallelism* and *distributed-memory parallelism*.
- In the shared-memory model, multiple processor cores work on data that resides in a shared main memory space; each of them has access to all the data all the time (in principle). Using special compiler directives (provided, e.g., by the OpenMP standard), one can tell the compiler do distribute works among different cores.
- Advantage: Relatively easy to implement (even in incremental steps, i.e. by parallelising one part of the code after the other).
- Disadvantage: Side effects if the programmer is not careful, technical limitations on the size of shared-memory units.

Domain Decomposition

- In the distributed-memory model, each process sees only its local data, if it is needed by another process it has to be sent explicitly, e.g., using the Message Passing Interface (MPI).
- In hydro simulations, the data is usually distributed by *domain decomposition*, i.e. each process gets a block of the grid to work on.
- At the beginning of each time step, the first few cells near an interface must be exchanged with the neighbouring process, where they are needed as boundary conditions in the reconstruction step.
- Global synchronisation is necessary, e.g., for determining the time step (which must be the same for each process).

Curvilinear Coordinates

Many astrophysical applications involve objects that are approximately spherical (stars). Hence, it is often advantageous to use spherical polar coordinates (better handling of boundary conditions, no artificial symmetry breaking). To formulate the equations of hydrodynamics in spherical polar coordinates, we need to use expressions for div and grad in curvilinear coordinates.
This is straightforward for the continuity equation and the energy equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

becomes:

$$\frac{\partial \rho}{\partial t} + \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \rho v_r\right) + \frac{1}{r \sin\theta}\frac{\partial}{\partial \theta}(\sin\theta\, \rho v_\theta) + \frac{1}{r \sin\theta}\frac{\partial}{\partial \varphi}(\rho v_\varphi) = 0.$$

We can then multiply by $r^2 \sin\theta$ and integrate over cells to get a manifestly conservative discretisation.

Curvilinear Coordinates

The energy equation

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot [(\rho e + P)\mathbf{v}] = v \cdot \nabla \Phi$$

reads

$$\frac{\partial \rho e}{\partial t} + \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2(e+P)v_r\right) + \frac{1}{r\sin\theta}\frac{\partial}{\partial \theta}(\sin\theta\,(\rho e + P)v_\theta) + \frac{1}{r\sin\theta}\frac{\partial}{\partial \varphi}[(\rho e + P)$$
$$= \rho v_r \frac{\partial \Phi}{\partial r} + \rho v_\theta \frac{1}{r}\frac{\partial \Phi}{\partial \theta} + \rho v_\varphi \frac{1}{r\sin\theta}\frac{\partial \Phi}{\partial \varphi},$$

in spherical polar coordinates.
The case of the momentum equation

$$\frac{\partial \rho \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\rho\mathbf{v} + \rho\mathbf{v}(\nabla\cdot\mathbf{v}) + \nabla P = -\rho\nabla\Phi,$$

is somewhat more complicated.

To evaluate this, we need to use

$$\nabla \cdot \mathbf{v} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 v_r\right) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\, v_\theta) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\varphi}(v_\varphi) = 0,$$

(similar to what we did before), and

$$\mathbf{v}\cdot\nabla = v_r\frac{\partial}{\partial r} + v_\theta\frac{1}{r}\frac{\partial}{\partial\theta} + v_\varphi\frac{1}{r\sin\theta}\frac{\partial}{\partial\varphi}.$$

Since the coordinate derivatives are to be applied to *vectors* here,

$$(v_r\frac{\partial}{\partial r} + v_\theta\frac{1}{r}\frac{\partial}{\partial\theta} + v_\varphi\frac{1}{r\sin\theta}\frac{\partial}{\partial\varphi})(v_r\mathbf{n}_r + v_\theta\mathbf{n}_\theta + v_\varphi\mathbf{n}_\varphi),$$

we also need derivatives of the unit vectors, e.g.,

$$\begin{aligned}\frac{\partial\mathbf{n}_r}{\partial r} &= 0\\ \frac{\partial\mathbf{n}_r}{\partial\theta} &= \mathbf{n}_\theta\\ \frac{\partial\mathbf{n}_r}{\partial\varphi} &= \sin\theta\mathbf{n}_\varphi.\end{aligned}$$

Curvilnear coordinates

After a lot of algebra, we obtain the following form for the momentum equation:

$$\frac{\partial \rho v_r}{\partial t} + \frac{1}{r^2}\frac{\partial r^2 \rho v_r^2}{\partial r} + \frac{1}{r\sin\theta}\frac{\partial r\sin\theta \rho v_r v_\theta}{\partial \theta} + \frac{1}{r\sin\theta}\frac{\partial \rho v_r v_\varphi}{\partial \varphi} + \frac{\partial P}{\partial r} = \rho\frac{v_\theta^2 + v_\varphi^2}{r} + \rho\frac{\partial \Phi}{\partial r}$$

$$\frac{\partial \rho v_\theta}{\partial t} + \frac{1}{r^2}\frac{\partial r^2 \rho v_r v_\theta}{\partial r} + \frac{1}{r\sin\theta}\frac{\partial r\sin\theta \rho v_\theta^2}{\partial \theta} + \frac{1}{r\sin\theta}\frac{\partial \rho v_\theta v_\varphi}{\partial \varphi} + \frac{1}{r}\frac{\partial P}{\partial \theta} = \rho\frac{v_\varphi^2\cot\theta - v_r v_\theta}{r} + \frac{\rho}{r}\frac{\partial \Phi}{\partial \theta}$$

$$\frac{\partial \rho v_\varphi}{\partial t} + \frac{1}{r^2}\frac{\partial r^2 \rho v_r v_\varphi}{\partial r} + \frac{1}{r\sin\theta}\frac{\partial r\sin\theta \rho v_\theta v_\varphi}{\partial \theta} + \frac{1}{r\sin\theta}\frac{\partial \rho v_\varphi^2}{\partial \varphi} + \frac{1}{r\sin\theta}\frac{\partial P}{\partial \varphi} = -\rho\frac{v_r v_\varphi + v_\theta v_\varphi}{r} + \frac{\rho}{r\sin\theta}\frac{\partial \Phi}{\partial \varphi}$$

Note: The momentum equation is no longer strictly flux-conservative. We get source terms due to fictitious forces that cannot be eliminated if we use a spherical polar coordinate basis for our vectors. Moreover, the fictitious forces formally have singularities on the axis/at the origin.

Geometric Source Terms

In practice, this is not a huge problem if we adhere to two rules of thumb:

- Do not operator-split fictitious forces and the corresponding flux terms.
- For discretisation, choose a form of the equations that keeps the geoemtric source terms small.
- Look at limiting cases (e.g. hydrostatic equilibrium).

For example, one could discretize $\partial P/\partial r$ as $(P_{i+1/2} - P_{i-1/2})/(r_{i+1/2} - r_{i-1/2})$, or use a pseudo-conservative form

$$\frac{1}{r^2}\frac{\partial P r^2}{\partial r} - \frac{2P}{r},$$

and discretise it as

$$\frac{1}{\Delta V_i}\left[P_{i+1/2}\Delta A^2_{i+1/2} - P_{i-1/2}\Delta A^2_{i-1/2}\right] - \frac{2P_i}{r_i}.$$

If we have pressure equilibrium ($P = \mathrm{const.}$ everyhwhere), the first finite-difference representation vanishes, and no numerical perturbations are generated. It is therefore a better choice.

Gravity

The gravitational source term looks fairly simple, but can frequently cause problems. Naively one could think that one can just use cell-centred values of the density and velocity to evalute the source terms in the momentum equation and energy equation and operator-split these terms. But there are several pitfalls:

- Near hydrostatic equilibrium, operator-splitting can perturb the velocity field very much during the individual steps because pressure and gravity forces are both large and nearly cancel. One can use an unsplit approach to fix this.
- For piecewise-linear reconstruction, it is very difficult to ensure that pressure and gravity balance each other well, unless one includes gravity effects in the Riemann solver.
- Computing the source term in the energy equation based on cell-centred values of ρ and $\mathbf{v}$ can lead to a strong violation of total energy conservation. One can avoid this by partially absorbing the source term in the fluxes:

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot [(\rho e + P + \rho\Phi)\mathbf{v}] = \rho \frac{\partial \Phi}{\partial t}.$$

Self-Gravity

For many applications (neutron star oscillations, supernovae, star formation, etc.), we need to include the self-gravity of the gas by solving the Poisson equation

$$\Delta\Phi = 4\pi G\rho$$

for the gravitational potential.
Different from the hydro equations, this is an elliptic equation: The domain of dependence is infinite, the density at a given point influences the field everywhere else. Formally, the solution is still simple (using Green's functions):

$$\Phi(\mathbf{r}) = -\int \frac{G\rho}{|\mathbf{r}-\mathbf{r}'|}\,\mathrm{d}^3\mathbf{r}'.$$

For discrete particles, we would just compute a sum,

$$\Phi(\mathbf{r}_i) = -\sum_{j\neq i} \frac{Gm_j}{|\mathbf{r}_i-\mathbf{r}_j|},$$

and might do something similar on a grid (summing over all cells, indexed by j):

$$\Phi(\mathbf{r}_i) = -\sum_{j\neq i} \frac{G\rho_j\Delta V_j}{|\mathbf{r}_i-\mathbf{r}_j|},$$

Self-Gravity

This "direct summation" is problematic, however, if we have many particles, or a large grid. For N particles, we would need $\mathcal{O}(10N^2)$ operations, and $\mathcal{O}(10N^6)$ for a 3D grid of M^3 zones. For practical purposes, this would be exorbitant, see the following examples:

Simulation	Zones/Particles	Operations per time step
3D supernova	$550 \times 128 \times 256$	10^{15}
Star formation (SPH)	5×10^6	3×10^{14}
Millennium run (N-body, cosmology)	10^9	10^{21}

Since this is merely the operation count per time-step, direct summation is out of the question for large simulations.

Self-Gravity – Spherical Harmonics

When using spherical polar coordinates, one can resort to a multipole expansion of the gravitational potential. Recall that the Laplacian in spherical polar coordinates is given by

$$\Delta\Phi = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\Phi}{\partial r}\right) + \frac{1}{r^2}\left[\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial\Phi}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2\Phi}{\partial\varphi^2}\right].$$

The eigenfunctions of the angular part of the Laplacian,

$$\Delta_\Omega \frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\varphi^2},$$

are the spherical harmonics $Y_\ell^m(\theta,\varphi)$ (remember the solution of the Schrödinger equation for the hydrogen atom):

$$\Delta_\Omega Y_\ell^m(\theta,\varphi) = -\ell(\ell+1)Y_\ell^m(\theta,\varphi)$$

The lowest-order spherical harmonics are (different normalisations and phase conventions are possible):

$$Y_0^0(\theta,\varphi) = \frac{1}{2}\sqrt{\frac{1}{\pi}}, \quad Y_1^{-1}(\theta,\varphi) = \frac{1}{2}\sqrt{\frac{3}{2\pi}}e^{-i\varphi}\sin\theta,$$

$$Y_0^0(\theta,\varphi) = -\frac{1}{2}\sqrt{\frac{3}{\pi}}\cos\theta, \quad Y_0^1(\theta,\varphi) = -\frac{1}{2}\sqrt{\frac{3}{2\pi}}e^{i\varphi}\sin\theta$$

Self-Gravity – Spherical Harmonics

Now we can decompose the solution into tensor products of the spherical harmonics and functions $f_{\ell,m}$ that depend only on r:

$$\Phi = \sum_{\ell,m} f_{\ell,m}(r) Y_\ell^m(\theta,\varphi)$$

$$\begin{aligned}\Delta\Phi &= \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\Phi}{\partial r}\right) + \frac{1}{r^2}\Delta_\Omega\Phi = \sum_{\ell,m}\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f_{\ell,m}}{\partial r}\right)Y_\ell^m + \frac{1}{r^2}f_{\ell,m}\Delta_\Omega Y_\ell^m\right] \\ &= \sum_{\ell,m}\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f_{\ell,m}}{\partial r}\right)Y_\ell^m - \frac{\ell(\ell+1)}{r^2}f_{\ell,m}Y_\ell^m\right].\end{aligned}$$

Thus, Poisson's equation becomes:

$$\sum_{\ell,m}\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f_{\ell,m}}{\partial r}\right)Y_\ell^m - \frac{\ell(\ell+1)}{r^2}f_{\ell,m}Y_\ell^m\right] = 4\pi G\rho$$

Now we multiply both sides with $Y^\star_{\ell,m}(\theta,\varphi)$ (the star denotes complex conjugation) and integrate over θ and φ. Because of the orthogonality relation $\int Y^\star_{\ell,m} Y_{\ell',m'}\,\mathrm{d}\Omega = \delta_{\ell\ell'}\delta_{mm'}$ this decouples the different ℓ and m.

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f_{\ell,m}}{\partial r}\right) - \frac{\ell(\ell+1)}{r^2}f_{\ell,m} = \underbrace{4\pi G\int Y^{\star}_{\ell,m}\rho(r,\theta\varphi)\mathrm{d}\Omega}_{\bar{\rho}_{\ell,m}}(r)$$

The solution to this second-order equation can be boiled down to two integrals:

$$f_{\ell,m}(r) = -\frac{1}{2\ell+1}\left[\frac{1}{r^{\ell+1}}\int_0^r \bar{\rho}_{\ell,m}(r')r'^{\ell+2}\,\mathrm{d}r' + r^{\ell}\int_r^{\infty}\bar{\rho}_{\ell,m}(r')r'^{\ell+2}\,\mathrm{d}r'\right].$$

Since the integrals can be computed recursively (from their values in the neighbouring zones), $\mathcal{O}(2N_r)$ operations are required for each ℓ and m. The total operation count is $\mathcal{O}(N_r N_\theta N_\varphi N_{\mathrm{multipoles}}) \sim \mathcal{O}(N^3 N_{\mathrm{multipoles}})$. Since the gravitational field is smooth and the higher multipoles contribute little, one can get away with a small number of multipoles ($20\ldots50$) or even try the monopole approximation (only take $\ell = m = 0$ into account). Compared to direct summation we save a factor of $N^3/N_{\mathrm{multipoles}}$ in operations. In 3D, this is several orders of magnitude.

One can also expand the Green's function into spherical (E. Müller& M. Steinmetz 1995, Computer Physics Communications, 89,45) to arrive at the same result.

Self-Gravity – Cartesian Coordinates

In Cartesian coordinates, we can exploit the properties of the Fourier transfrom:

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{(2\pi)^3} \int e^{i\mathbf{k}\mathbf{x}} \hat{f}(\mathbf{k}) \, \mathrm{d}^3 k \\ \hat{f}(\mathbf{k}) &= \int e^{-i\mathbf{k}\mathbf{x}} \hat{f}(\mathbf{x}) \, \mathrm{d}^3 x. \end{aligned}$$

Derivatives in real space correspond to multiplications with ik_i in Fourier space, and convolutions in real space correspond to products in Fourier space:

$$\begin{aligned} \frac{\partial f}{\partial x} &\leftrightarrow ik_x \hat{f} \\ \nabla f &\leftrightarrow i\mathbf{k}\hat{f} \\ \Delta f &\leftrightarrow -\mathbf{k}^2 \hat{f} \\ \int f(\mathbf{x}-\mathbf{x}') g(\mathbf{x}') \, \mathrm{d}^3 x' &\leftrightarrow \hat{f}(\mathbf{k}) \hat{g}(\mathbf{k}) \end{aligned}$$

Therefore, Poisson's equation reads

$$-k^2\hat{\Phi}(\mathbf{k}) = 4\pi G\hat{\rho}(\mathbf{k})$$

in Fourier space. Once we have the Fourier transform $\hat{\rho}$ of ρ, we only need $\mathcal{O}(N^3)$ operations instead of $\mathcal{O}(N^6)$ – we simply divide $4\pi G\hat{\rho}$ by k^2 to get the solution.
However, if we compute the discrete Fourier transform (DFT) by brute force, then this again takes $\mathcal{O}(N^3 \times N^3) = \mathcal{O}(N^6)$ operations, so we haven't saved anything yet. Fortunately, there is a divide-and-conquer alogrithm, the fast Fourier transform (FFT), to carry out the transform (and its inverse) in $\mathcal{O}(\mathcal{N}^\ni \log \mathcal{N}^\ni)$ operations. Thus, one almost save a factor of N^3 compared to direct summation. Implementations of the FFT are available as free software (e.g. the FFT library `http://www.fftw.org/`).

Self-Gravity – Cartesian Coordinates with Non-Periodic Boundary Conditions

Prima facie the FFT-based solution only works nicely for periodic boundary conditions in all directions. However, if we want the correct solution for an unbounded domain ($\Phi \to 0$ for $r \to \infty$, ...), we can still transform the integral resolution of the solution with the Green's function into a product in Fourier space because it is just a convolution:

$$\begin{aligned} \Phi(\mathbf{r}) &= -\int \frac{G\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\,\mathrm{d}^3\mathbf{r}' \\ \hat{\Phi}(\mathbf{k}) &= -\frac{G}{2\pi}\hat{\rho}(\mathbf{k})g(\mathbf{k}), \end{aligned}$$

where $g(\mathbf{k})$ is the Fourier transform of $1/|\mathbf{r}|$. Of course, if we use periodic boundary conditions, then we effectively convolve the Green's function with a density distribution that is periodically mirrored outside a simulation box, and this is not what we want. However, doubling the computational domain in every direction and padding the other octants with zeros removes this problem, and we recover the solution for boundaries at infinity.

Other Methods for Solving Poisson's Equation

A number of other methods exist to solve Poisson's equation (exactly up to discretisation error, or approximately):

- Fast direct sparse matrix solvers for the discretised Poisson equation
- Conjugate Gradient Method
- Multigrid
- Tree methods (Barnes-Hut, fast multipole method) for SPH/N-body simulations (in principle also suited for grid-based codes)