Kadath: a spectral solver for theoretical physics

Philippe Grandclément

Laboratoire de l'Univers et de ses Théories (LUTH)
CNRS / Observatoire de Paris
F-92195 Meudon, France

philippe.grandclement@obspm.fr

October 6^{th} 2015

What is KADATH ?

KADATH is a library that implements spectral methods in the context of theoretical physics.

- It is written in C++, making extensive use of object oriented programming.
- Versions are maintained via Subversion.
- Minimal website : *http://luth.obspm.fr/~luthier/grandclement/kadath.html*
- The library is described in the paper : *JCP* **220**, *3334 (2010).*
- Designed to be very modular in terms of geometry and type of equations.
- LateX-like user-interface.
- More general than its predecessor LORENE.

Describing the space

Multi-domain approach

- Space is split into several touching (not overlapping) domains.
- In each domain, the physical coordinates X are mapped to the numerical ones $X^\star$.

Why ?

- To have $\mathcal{C}^\infty$ functions only.
- To increase resolution where needed.
- To use different descriptions (functions or equations) in regions of space.

Geometries in KADATH

- 1D space.
- Cylindrical-like coordinates.
- Spherical spaces with time periodicity.
- Polar and spherical spaces.
- Bispherical geometries.
- Variable domains (surface fitting).
- Additional cases are relatively easy to include.

Describing the functions

Spectral expansion

Given a set of orthogonal functions Φ_i on an interval Λ, spectral theory gives a recipe to approximate f by

$$f \approx I_N f = \sum_{i=0}^{N} a_i \Phi_i$$

Properties

- the Φ_i are called the basis functions.
- the a_i are the coefficients.
- Multi-dimensional generalization is done by direct product of basis.

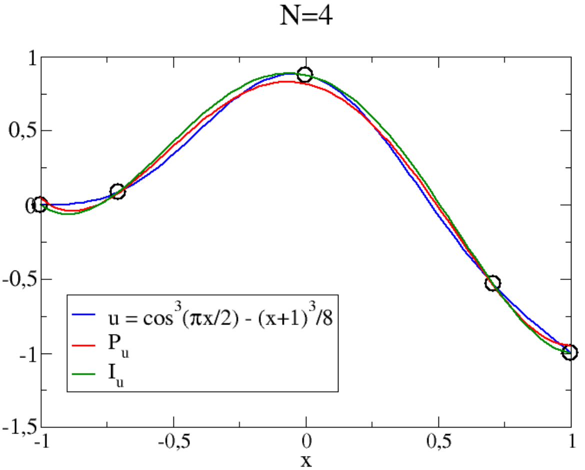
Usual basis functions

- Orthogonal polynomials : Legendre or Chebyshev.
- Trigonometrical polynomials (discrete Fourier transform).

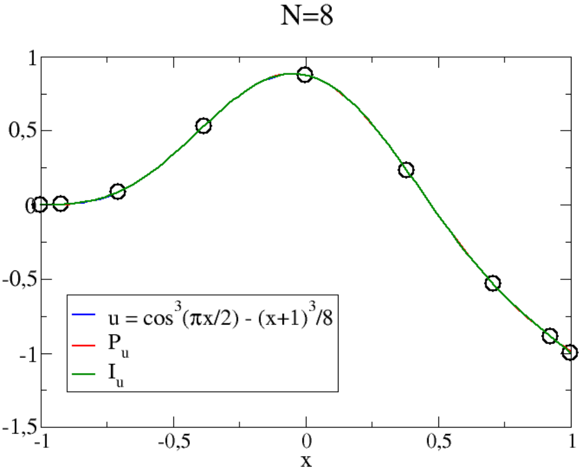
Spectral convergence

- If f is $\mathcal{C}^\infty$, then $I_N f$ converges to f faster than any power of N.
- For functions less regular (i.e. not $\mathcal{C}^\infty$) the error decrease as a power-law.

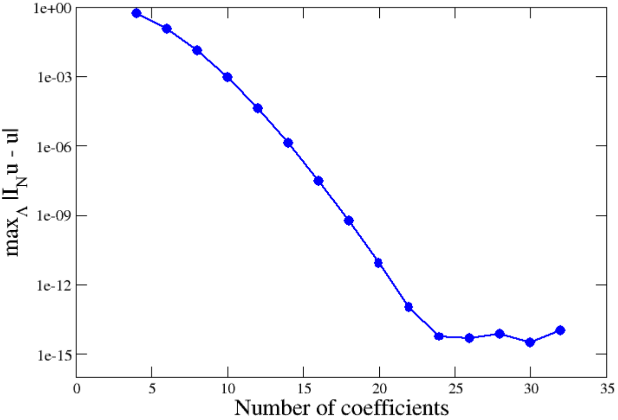
Collocation points



Collocation points



Spectral convergence



Choice of basis

Important step in setting the solver. All the terms involved in the equations must have consistent basis.

Guideline for scalars

- Assume that all the fields are polynomials of the Cartesian coordinates (when defined).
- Express the Cartesian coordinates in terms of the numerical ones.
- Deduce an appropriate choice of basis.

Higher order tensors

- With a Cartesian tensorial basis: given by gradient of scalars.
- For other tensorial basis: make use of the passage formulas that link to the the Cartesian one.

Dealing with field equations

Let $R = 0$ be a field equation (like $\Delta f - S = 0$). The weighted residual method provides a discretization of it by demanding that

$$(R, \xi_i) = 0 \quad \forall i \leq N$$

Properties

- $(,)$ denotes the same scalar product as the one used for the spectral expansion.
- the ξ_i are called the test functions.
- For the $\tau-$method the ξ_i are the basis functions (i.e. one works in the coefficient space).
- Some of the last residual equations must be relaxed and replaced by appropriate matching and boundary conditions to get an invertible system.
- Additional regularity conditions can be enforced by a Galerkin method.

Newton-Raphson iteration

Given a set of field equations with boundary and matching equations, `KADATH` translates it into a set of algebraic equations $\vec{F}(\vec{u}) = 0$, where $\vec{u}$ are the unknown coefficients of the fields.

The non-linear system is solved by Newton-Raphson iteration

- Initial guess $\vec{u}_0$.
- Iteration :
  - Compute $\vec{s}_i = \vec{F}(\vec{u}_i)$
  - If $\vec{s}_i$ if small enough $\Longrightarrow$ solution.
  - Otherwise, one computes the Jacobian : $\mathbf{J}_i = \frac{\partial \vec{F}}{\partial \vec{u}}(\vec{u}_i)$
  - One solves : $\mathbf{J}_i \vec{x}_i = \vec{s}_i$.
  - $\vec{u}_{i+1} = \vec{u}_i - \vec{x}_i$.

Convergence is very fast for good initial guesses.

Computation of the Jacobian

Explicit derivation of the Jacobian can be difficult for complicated sets of equations.

Automatic differentiation

- Each quantity x is supplemented by its infinitesimal variation δx.
- The dual number is defined as $\langle x, \delta x \rangle$.
- All the arithmetic is redefined on dual numbers. For instance $\langle x, \delta x \rangle \times \langle y, \delta y \rangle = \langle x \times y, x \times \delta y + \delta x \times y \rangle$.
- Consider a set of unknown $\vec{u}$, and a its variations $\delta \vec{u}$. When $\vec{F}$ is applied to $\langle \vec{u}, \delta \vec{u} \rangle$, one then gets : $\left\langle \vec{F}\left(\vec{u}\right), \delta \vec{F}\left(\vec{u}\right) \right\rangle$.
- One can show that

$$\delta \vec{F}\left(\vec{u}\right) = \mathbf{J}\left(\vec{u}\right) \times \delta \vec{u}$$

The full Jacobian is generated *column by column*, by taking all the possible values for $\delta \vec{u}$, at the price of a computation roughly twice as long.

Inversion of the Jacobian

Consider N_u unknown fields, in N_d domains, with d dimensions. If one works with N coefficients in each dimension, the Jacobian is a $m \times m$ matrix with:

$$m \approx N_d \times N_u \times N^d$$

For $N_d = 5$, $N_u = 5$, $N = 20$ and $d = 3$, one gets $m = 200 \cdot 000$, which is about 150 Go for a full matrix.

Solution

- The matrix is computed in a distributed manner.
- Easy to parallelize because of the manner the Jacobian is computed.
- The library `SCALAPACK` is used to invert the distributed matrix.

200 processors is enough for $m \approx 150 \cdot 000$.
`KADATH` has been tested on 1,024 processors (*titane* machine from the CEA).

LateX-like interface

```
        // Matter terms :
        for (int d=0 ; d<=1 ; d++) {
          syst.add_def (d, "U^i = (ome*m^i + bet^i)/N ") ;
          syst.add_def (d, "pres = kap * n^2") ;
          syst.add_def (d, "edens = mb * n + kap * n^2") ;
          syst.add_def (d, "H = log(1+2*n*kap/mb)") ;
          syst.add_def (d, "Gamsquare = 1. / (1-U_i *U^i)") ;
          syst.add_def (d, "Eeuler = Gamsquare  * (edens+pres) - pres") ;
          syst.add_def (d, "Jeuler^i = (Eeuler + pres) * U^i") ;
          syst.add_def (d, "Seuler_ij =  (Eeuler + pres) * U_i * U_j + pres* g_ij ") ;
          syst.add_def (d, "S = g^ij * Seuler_ij") ;
        }

        // Extrinsic curvature
        syst.add_def ("Dshift_i^j = D_i bet^j") ;
        syst.add_def("K_ij = 0.5 * (Dshift_ij + Dshift_ji) / N");

        // Gauge part
        syst.add_def ("V^i = g^kl * Gam_kl^i") ;
        syst.add_def ("Gauge_ij = D_i V_j + D_j V_i") ;
        syst.add_def ("Ope_ij = R_ij - 0.5*Gauge_ij") ;

        for (int d=0 ; d<=1 ; d++) {
            syst.add_def (d, "Hamilton = g^ij * Ope_ij - K_ij * K^ij - 4 * qpig * Eeuler") ;
            syst.add_def (d, "Momentum^i = D_j K^ij - 2 * qpig * Jeuler^i") ;
            syst.add_def (d, "Evol_ij = N * (Ope_ij - 2*K_ik*K_j^k) - D_i D_j N + bet^k * D_k K_ij + K_ik *
Dshift_j^k + K_jk * Dshift_i^k + N * qpig  * ((S-Eeuler)*g_ij - 2 * Seuler_ij) ") ;
        }

        for (int d=2 ; d<ndom ; d++) {
            syst.add_def (d, "Hamilton = g^ij*Ope_ij - K_ij * K^ij") ;
            syst.add_def (d, "Momentum^i = D_j K^ij") ;
            syst.add_def (d, "Evol_ij = N * (Ope_ij - 2*K_ik*K_j^k) - D_i D_j N + bet^k * D_k K_ij + K_ik *
Dshift_j^k+ K_jk * Dshift_i^k") ;
        }
```

Successful applications

- Critic solutions.
- Vortons.
- Neutron stars.
- Binary black holes.
- Breathers and quasi-breathers (see G. Fodor's talk).
- Current applications to geons (see G. Martinon's talk).
- Boson stars (published in *PRD 90, 024068 (2014)*, with C. Some and E. Gourgoulhon).

Boson star model

A boson star is described by a complex scalar field ϕ coupled to gravity. The field is invariant under a $U(1)$ symmetry :

$$\phi \longrightarrow \phi \exp(i\alpha) .$$

The Lagrangian of the matter is given by

$$\mathcal{L}_M = -\frac{1}{2}\left[g^{\mu\nu}\nabla_\mu\bar{\phi}\nabla_\nu\phi + V\left(|\phi|^2\right)\right] .$$

The induced stress-energy tensor is then

$$T_{\mu\nu} = \frac{1}{2}\left[\nabla_\mu\bar{\phi}\nabla_\nu\phi + \nabla_\nu\bar{\phi}\nabla_\mu\phi\right] - \frac{1}{2}g_{\mu\nu}\left[g^{\alpha\beta}\nabla_\alpha\bar{\phi}\nabla_\beta\phi + V\left(|\phi|^2\right)\right] .$$

In the following I will consider the simplest potential $V = |\phi|^2$.

Ansatz for the field

One seeks solutions such that

$$\phi = \phi_0 \exp\left[i\left(\omega t - k\varphi\right)\right],$$

where ϕ_0 depends only on r and θ.
k is an integer and so $k = 0$ corresponds to solutions that are spherically symmetric (I will concentrate here on the case $k \neq 0$)

Asymptotic behavior

Asymptotically, ϕ_0 obeys

$$\Delta_3 \phi_0 - \frac{k^2}{r^2 \sin^2 \theta} \phi_0 - \left(1 - \omega^2\right) \phi_0 = 0$$

It follows that the field is localized if and only if $\omega < 1$.
When $\omega \to 1$, $\phi_0 \to 0$ and its size tends to infinity.

3+1 decomposition

We use the 3+1 decomposition in quasi-isotropic coordinates :

$$\mathrm{d}s^2 = -N^2\mathrm{d}t^2 + A^2\left(\mathrm{d}r^2 + r^2\mathrm{d}\theta^2\right) + B^2 r^2 \sin^2\theta \left(\mathrm{d}\varphi - N^\varphi \mathrm{d}t\right)^2 .$$

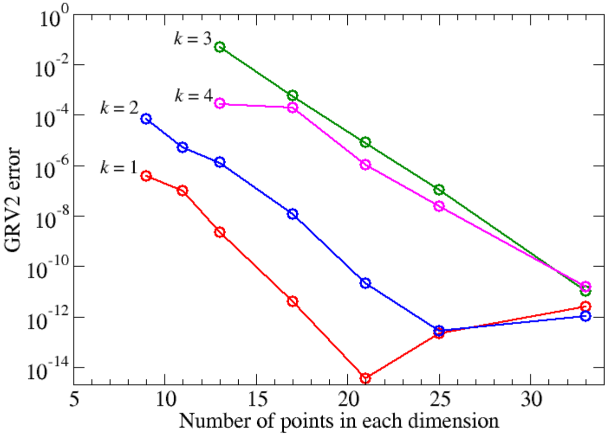
N, A, B and N^φ depend only on r and θ.
Metric fields must obey Einstein's equations and the complex field Klein-Gordon one.

Numerical setting

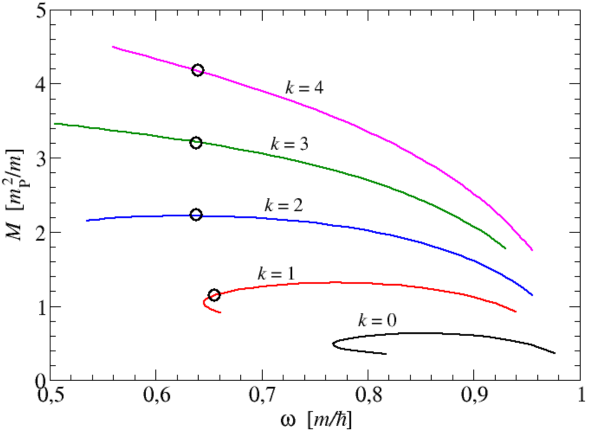
Equations are solved using the *Polar* domains of Kadath.

- The unknowns are combinations of the metric fields N, A, B and N^{φ} plus the matter term ϕ_0.
- The equations are the 3+1 ones + Klein-Gordon.
- For each k one needs a good initial guess.
- Sequences are computed by varying ω.

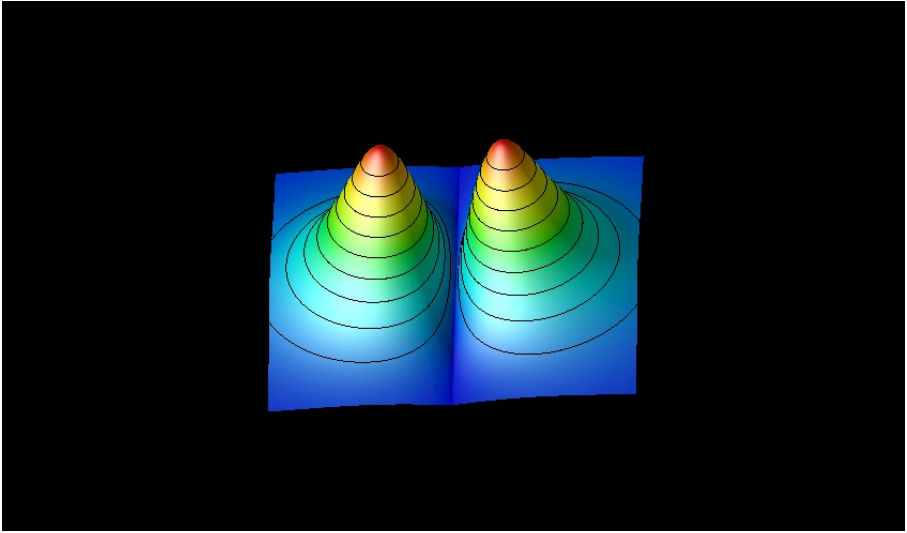
Measure of precision: virial error



ADM mass



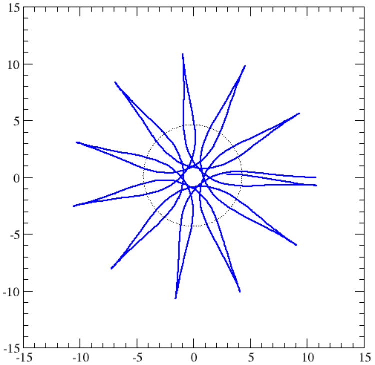
Field : toroidal configuration



Orbits

Geodesics around boson stars can be numerical integrated using the Gyoto code (*http://gyoto.obspm.fr/*).

Due to the absence of event horizon, particles can pass very close to the center: new type of orbits.



Conclusions

- Kadath design is satisfactory.
- Applications begin to be numerous.
- Users are still (very) few.
- Lack of tutorials, documentations.
- Come talk to me...