An overview of the Kadath library

Philippe Grandclément

API ondes gravitationnelles, Meudon, 17 novembre 2023

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

philippe.grandclement@obspm.fr

KADATH library

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 git.
- Website : *www.kadath.obspm.fr*
- 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.

A test problem

Find the conformal factor Ψ of the Schwarzschild black hole in QI coordinates.

System of equations

- Bulk : $\Delta\Psi = 0$.
- Inner BC : $\Psi_{,r} + \frac{1}{2a}\Psi = 0$
- Outer BC : $\Psi = 1$

a is the radius of the black hole and the solution is

$$\Psi\left(r\right) = 1 + \frac{a}{r}.$$

Concept in 1D

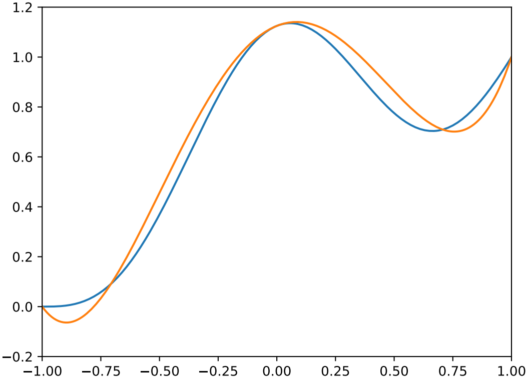
Given a set of orthogonal functions Φ_i on an interval Λ, spectral theory gives a recipe to approximate f by

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

Properties

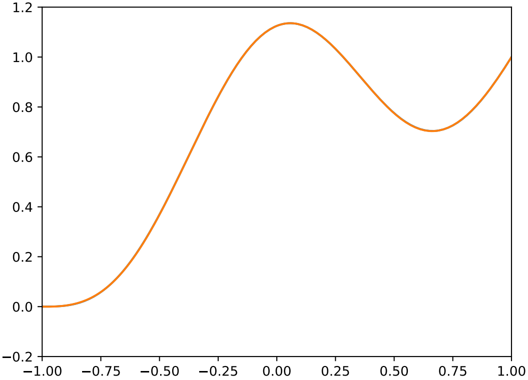
- the Φ_i are called the basis functions.
- the a_i are the coefficients : it is the quantity stored on the computer.
- Multi-dimensional generalization is done by direct product of basis.
- The computation of the a_i comes from the Gauss quadratures.

Example of interpolant for $N = 4$



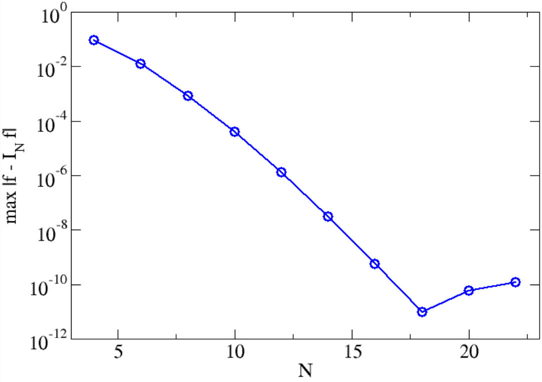
blue curve $f(x) = \cos^3(\pi x/2) + (x+1)^3/8$; *orange* : $I_4 f$.

Example of interpolant for $N = 8$



blue curve $f(x) = \cos^3(\pi x/2) + (x+1)^3/8$; *orange* : $I_8 f$.

Spectral convergence

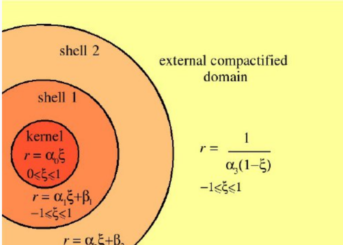


Multi-domain setting

Numerical coordinates

- Space is divided into several numerical domains.
- In each domain there is a link between the physical coordinates X and the numerical ones $X^\star$.
- Spectral expansion is performed with respect to $X^\star$.
- Non-periodic coordinates are expanded wrt to polynomials.
- Periodic coordinates (i.e. angles) are described by trigonometrical functions.

Example spherical space

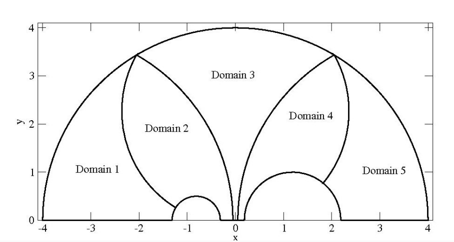


Setting the space in KADATH

```
// 3D :
int dim = 3 ;

// Number of points in each dimension
Dim_array res (dim) ;
res.set(0) = 13 ; res.set(1) = 5 ; res.set(2) = 4 ;

// Center of the coordinates
Point center (dim) ;
for (int i=1 ; i<=dim ; i++)
    center.set(i) = 0 ;

// Number of domains and boundaries :
int ndom = 4 ;
Array<double> bounds (ndom-1) ;
// Radius of the BH
double aa = 1.323 ;
bounds.set(0) = aa ; bounds.set(1) = 1.7557*aa ; bounds.set(2) = 2.9861*aa ;

// Chebyshev or Legendre :
int type_coloc = CHEB_TYPE ;

// Spherical space :
Space_spheric space(type_coloc, center, res, bounds) ;
```

Other spaces available

- Cylindrical space.
- Bispherical space.
- Spaces with periodic time coordinates.
- Spaces with adaptable domains.
- Spaces with various symmetries.
- Additional ones relatively easy to include.



`KADATH` management of the spectral basis

- For every computation, `KADATH` tries to assert the basis of the result.
- Straightforward for things like the product, inverse, sum etc...
- For other computations (like $\exp$, $\cos$, $\sqrt{}$) the base cannot be directly obtained and is lost.
- **Important rule** set the base by hand if and only if it is required.
- Be careful when enforcing the standard base. For instance $\rho = \sqrt{x^2 + y^2}$ is not expanded onto the standard base.
- Most of the errors in using `KADATH` come from inappropriate setting of the basis.

Setting the fields in KADATH

```
// Initial guess for the conformal factor :
Scalar conf (space) ;
conf = 1. ;
conf.std_base() ;

// The analytic solution (except in the nucleus)
Scalar sol (space) ;
sol.set_domain(0) = 0. ;
for (int d=1 ; d<ndom ; d++)
        sol.set_domain(d) = 1 + aa / space.get_domain(d)->get_radius() ;
sol.std_base() ;
```

Weighted residual method

Consider a field equation $R = 0$ (ex. $\Delta f - S = 0$). The discretization demands that

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

Properties

- $(,)$ is the same scalar product as the one used for the spectral approximation.
- the ξ_i are called the test functions.
- For the τ-method, the ξ_i are the basis functions.
- Amounts to cancel the coefficients of R.
- Some equations are relaxed and must be replaced by appropriate boundary and matching conditions.

The discrete system

Original system

- Unknowns : tensorial fields.
- Equations : partial derivative equations.

Discretized system

- Unknowns : coefficients $\vec{u}$.
- Equations : algebraic system $\vec{F}\left(\vec{u}\right) = 0$.

Properties

- For a linear system $\vec{F}\left(\vec{u}\right) = 0 \Longleftrightarrow A^i_j u^j = S^i$
- In general $\vec{F}\left(\vec{u}\right)$ is even not known analytically.
- $\vec{u}$ is sought numerically.

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.

Numerical resources

Consider N_u unknown fields, in N_d domains, with d dimensions. If the resolution is N in each dimension, the Jacobian is an $m \times m$ 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 reaches $m = 200\,000$

Solution

- The matrix is distributed on several processors.
- Easy because the Jacobian is computed column by column.
- The library `SCALAPACK` is used to invert the distributed matrix.

- $d = 1$ problems : sequential.
- $d = 2$ problems : 100 processors (mesocenters).
- $d = 3$ problems : 1000 processors (national supercomputers).

Solving the system with KADATH

```
// Solve the equation in space outside the nucleus
System_of_eqs syst (space, 1, ndom-1) ;
// Only one unknown
syst.add_var ("P", conf) ;
// One user defined constant
syst.add_cst ("a", aa) ;

// Inner BC
syst.eq_eq_bc (1, INNER_BC, "dn(P)+0.5/a*P=0") ;

for (int d=1 ; d<ndom ; d++) {
	// Bulk equation (2nd order)
	syst.add_eq_inside (d, "Lap(P)=0") ;
	if (d!=ndom-1) {
		// Matching of the solution
		syst.add_eq_matching (d, OUTER_BC, "P") ;
		// Matching of the radial derivative
		syst.add_eq_matching (d, OUTER_BC, "dn(P)") ;
}
// Outer BC
syst.add_eq_bc (ndom-1, OUTER_BC, "P=1") ;

// Newton-Raphson
double conv ;
bool endloop = false ;
int ite = 1 ;
while (!endloop) {
	endloop = syst.do_newton(1e-8, conv) ;
	cout << "Newton iteration " << ite << " " << conv << endl ;
	ite++ ;
}
```

Advanced topics : definitions

- When an expression of the unknowns appears often.
- That expression can be made into a definition.
- Simplifies the writing of the equations.
- Makes the code faster as the definitions are computed only when needed.

```
// Extrinsic curvature tensor
syst.add_def ("K_ij=(D_i B_j+D_j B_i)2/N") ;

// Can be used in other expressions
// Hamiltonian constraint
syst.add_def ("H=R-K_ij*K^ij") ;
// Momentum constraints
syst.add_def ("M^i=D_j K^ij") ;
```

Advanced topics : metrics

- Special type of second order tensor.
- Enables the index manipulation.
- Enables the use of covariant derivative.
- Enables the use of Riemann and Ricci tensors

```
// Definition of a metric (from a second order tensor)
// Here met is an unknown also (use Metric_const otherwise)
Metric_general met (gmet) ;

// Associates the metric to the system
met.set_system (syst, "g") ;

// Now you can compute things like
syst.add_def ("derN=D_i_N") ;
// The Ricci is known
syst.add_def ("Ricci_ij=R_ij") ;
```

Advanced topics : global unknowns

- Some unknowns are numbers, not fields.
- Associated with integral equations.

```
double omega = 0. ;

// Omega is an unknown
syst.add_var ("ome", omega) ;

// Equality of the ADM and Komar masses forces the right value of omega
// Can be expressed as
space.add_eq_int_inf (syst, "integ(dn(N)+2*dn(P))=0") ;
```

Last words

- Additional specialized features (adapted domains).
- Many successful applications (boson stars, hairy black holes, initial data for general relativity).
- Additional functionalities are included regularly.
- The number of users increases, at last...

Try it...

Kadath website (https ://kadath.obspm.fr) has some tutorials... Have fun...