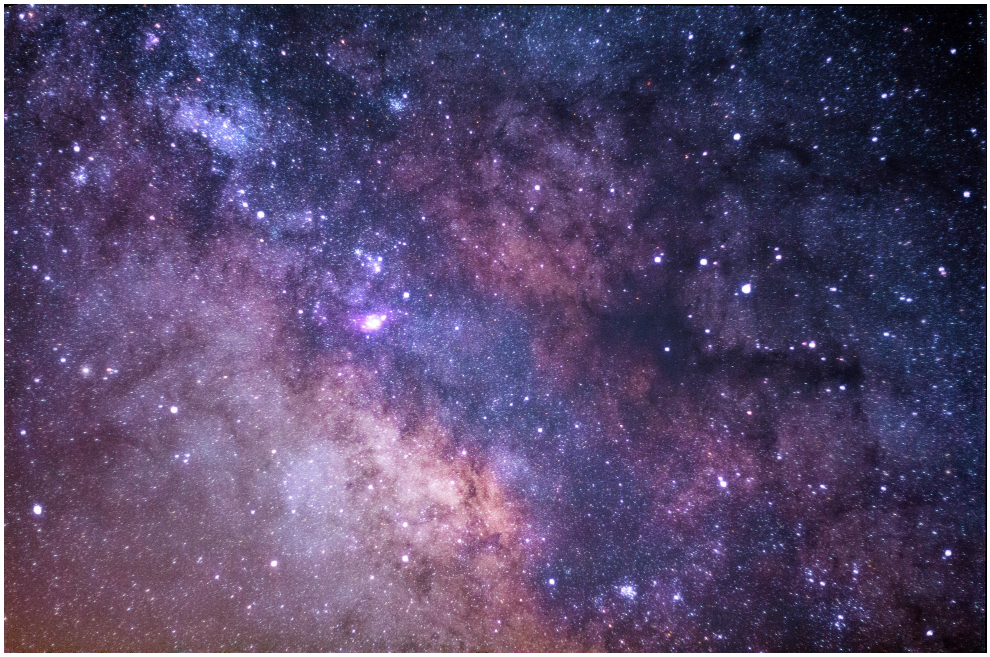
{General Relativity}

Version 0.7 ShanDong University May 3, 2019



What is more important is that you are willing to work hard;
to devote the time needed;
to ask questions when things don't make sense!

Contents

Chapter 1

xAct Application

xAct is a open-source Mathematica package, which provides the fundamental symbolic manipulation of arbitrary dimension of tensors in spacetime. And xAct syntax heritates from Mathematica, although possessed its own basis datatypes.

1.1 Quick Introduction to Mathematica

We give a quick review of Mathematica basis syntax and common operations related to using xAct.

1.1.1 Fundamental Concepts

Different from common programming language, the wolf language is functional programming language. The fundamental datatype is pattern and expressions. And every expression has a Head(We can use the function `FullForm[]`)to investigate the complicated expressions
There are also some value type inside Mathematica.In the left example, we give some simple code example about how to express DownValue and UpperValue.

```
(*-----------------------------------------------*)
    f[g[x_]]:= fg[x]
    Exp[g[x_]]:= expg[x]
    g/:g[x_] + g[y_] := gplus[x,y]
(*-----------------------------------------------*)
    g/:f[g[x_]]:Sin[g[x_]]
    g[x_] = Sin[x]
(*Think about the result of f[g[3]]*)
```

The Mathematica own evaluation mechanism behaves(for `f[g[x]]`):

- Find the user's definition of function **g**;
- Evaluate whether there is a priority existence between DownValue and UpperValue;
- Choose the DownValue definition first into the inital expression;
- Choose the UpperValue definition for the whole substitution;

1.1.2 Common Short Operators

In order to achieve an effect of expreesing Mathematica multiple-embedding expressions succintly, we often use some short notations:

- `function@parameter` which is equivalent to the action of function;
- `expression // function` which is a kind of posifix syntax compared to above;
- `expression/;expression` confine the conditions;
- `/@` The short notation for `Map[]` function;
- `/.` Represents an substitution.

1.2 Tensor-Group Representation

Since in computer-based tensor algebra system, the permutaion group and related algorithm is commonly used, we give a short description of permutation.

Definition 1.1 In a group G, a subset $X \in G$ is a generating set for G if every $g \in G$ can be written as a product of powers of elements taken from X:

$$g = x_1^{a_1} x_2^{a_2} \cdots x_r^{a_r} \tag{1.1}$$

where $x_i \in X$ and $a_i \in Z$, We also say that X generates G and write $G = \langle X \rangle$.

Definition 1.2 For β in Ω, the stabiliser of β in G is:

$$G_\beta = \{g \in G : \beta^g = \beta\}$$

Because we are considering the permutation group's elements, to achive the effect of a stabilizer, what we can do is to find a series of permutation not related to the choosed element(The inverse is obvious) Given $\beta_1, \beta_2, \ldots, \beta_i \in \Omega$,we can inductively define:

$$G_{\beta_1,\beta_2,\ldots,\beta_i} = \left(G_{\beta_1,\beta_2,\ldots,\beta_{i-1}}\right)_{\beta_i} = \left\{g \in G : \beta_j^g = \beta_j, \text{ for } j = 1, 2, \ldots, i\right\}$$

We now define the concept of a base, and its associated chain of subgroups.

Definition 1.3 A base for G is a finite sequence $B = [\beta_1, \beta_2, \ldots, \beta_k]$ of distinct points in Ω such that:

$$G_{\beta_1,\beta_2,\ldots,\beta_k} = 1$$

If we write $G^{(i)} = G_{\beta_1,\beta_2,\ldots,\beta_{i-1}}$, , then we have a chain of stabiliser:

$$G = G^{(1)} \geq G^{(2)} \geq \cdots \geq G^{(k)} \geq G^{(k+1)} = 1$$

Definition 1.4 A strong generating set for G with respect to B is a set S of group elements such that for $i = 1, 2, \ldots, k$:

$$G^{(i)} = \left\langle S \cap G^{(i)} \right\rangle$$

A strong generating set for G is a subset $S \in G$ such that S contains generators for each of the groups $G^{(i)}$ $0 \leq i \leq m$

1.3 Using xAct

Chapter 2

Different Toptics Related to NR

2.1 Energy-Momentum Tensor

Table 2.1: Energy conditions.

Name	Statement	Conditions
Weak	$T_{\alpha\beta}v^\alpha v^\beta \geq 0$	$\rho \geq 0, \quad \rho + p_i > 0$
Null	$T_{\alpha\beta}k^\alpha k^\beta \geq 0$	$\rho + p_i \geq 0$
Strong	$\left(T_{\alpha\beta} - \frac{1}{2}Tg_{\alpha\beta}\right)v^\alpha v^\beta \geq 0$	$\rho + \sum_i p_i \geq 0, \quad \rho + p_i \geq 0$

Strictly speaking, the EM tensor should be well-defined under the scope of variational principle. For an observer, what he measure is totally like in the flat spacetime. Inspired by the idea, the next step to consider the energy condition is just to do a series of coordinate transfomation.
Fisrt, we can convert the EM tensor to a more beautiful form:

$$T^{\alpha\beta} = \rho\hat{e}_0^\alpha\hat{e}_0^\beta + p_1\hat{e}_1^\alpha\hat{e}_1^\beta + p_2\hat{e}_2^\alpha\hat{e}_2^\beta + p_3\hat{e}_3^\alpha\hat{e}_3^\beta \tag{2.1}$$

$$\eta_{\mu\nu} = g_{\alpha\beta}\hat{e}_\mu^\alpha\hat{e}_\nu^\beta \tag{2.2}$$

Naturally, the components in the above equation is just the observables if in the flat spacetime.
If there is a observer with four-velocity:

$$v^\alpha = \gamma\left(\hat{e}_0^\alpha + a\hat{e}_1^\alpha + b\hat{e}_2^\alpha + c\hat{e}_3^\alpha\right)$$
$$\gamma = \left(1 - a^2 - b^2 - c^2\right)^{-1/2}$$ [1]

Tips To obtain the related energy conditions, the best method is to substitute a general normalized timelike or lightlike vector into the expression by evaluating the following results.

2.1.1 Physical Meaning

We begin with the introduction of EM field strength tensor:

$$F_{\mu\nu} = \begin{pmatrix} 0 & -E_1 & -E_2 & -E_3 \\ E_1 & 0 & B_3 & -B_2 \\ E_2 & -B_3 & 0 & B_1 \\ E_3 & B_2 & -B_1 & 0 \end{pmatrix} \tag{2.3}$$

which satisfies:

$$F_{0i} = -F_{i0} = -E_i \quad F_{ij} = \varepsilon_{ijk}B_k \tag{2.4}$$

With the help of Maxwell equations, the conservation of four-current can be:

$$\partial_\nu j^v = 0 \tag{2.5}$$

To get a geometrical meaning of current in Minkowski space(the same measure unit),what we can do is just:

- $\Delta S^0 = \left(\Delta x^1 \Delta x^2 \Delta x^3\right)$
- $\Delta S^1 = \left(\Delta x^0 \Delta x^2 \Delta x^3\right) = c(\Delta t \Delta y \Delta z)$

Thus the general current defined in the Minkowski space:

$$j^\mu = \frac{c\Delta q}{\Delta S^\mu}$$

For the matter field, a simplification is just to replace the charge density with mass density to understand the underlying physical meaning.

2.2 Congruence of Timelike Geodesic

For the congruence, we have a set of tangent vector as well as deviation vector,like the diagram below: Under some specific settings, we have formulations:

$$u^\alpha u_\alpha = -1, \quad u^\alpha_{;\beta} u^\beta = 0, \quad u^\alpha_{;\beta} \xi^\beta = \xi^\alpha_{;\beta} u^\beta, \quad u^\alpha \xi_\alpha = 0$$

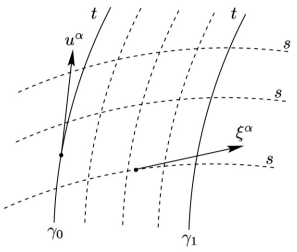


Figure 2.1: congruence

Frobenius's Theorem A congruence of curves (timelike, spacelike, null) is hypersurface orthogonal if and only if $u_{[\alpha;\beta]u_\gamma} = 0$

According to Raychaudhuri's equation and Focusing theorem, the expansion(representing the 3-D volume constructed with deviation geodesics) will always decrease or increase.

2.2.1 Null Congruences

Consider the speciality of null hypersurface, whose normal vector's norm is zero. It is important to notice that the hypersurface itself cannot give a completely basis in the spacetime.The vector came from the normal vector belongs to the hypersurface tangent space. That's to say, we still needs to find a vector to constuct the complete basis.

But the dual vector and hypersurface tangent space can "combine" into a complete basis, if we just care the coordinate components.(Actually, the dual vector and tangent vector cannot be equally weighted object)

If a null geodesic congruence is orthogonal to a hypersurface, we can show that the congruences are just lie on the hypersurface.

2.3 Integration on Hypersurface

2.4 Spacetime Property

When we talk about the homogeneous and isotropy, the specific meaning needa accurate mathematica definition:

Homogeneous We say spacetime is spatially homogeneous if there exists hyper-surfaces Σ_t, parametrised by a 'time' t that 'foliate' spacetime, such that for each t and any two points p and q in Σ_t there exists an isometry of the spatial metric that carries p into q.

Isotropy We say spacetime is spatially isotropic at a point p if for any timelike observer with tangent u^a and any two 'spatial tangent vectors' s^1 and s^2 at p, there exists a rotation that carries s^1 into s^2 but leaves p and u^a fixed.
No matter what forms of coordinate transformation, what we care about is the original tensor and the tensor located in the new place.(we can use a pointview of curves mapping)

$$ds^2 = -dt^2 + a^2(t)\left[\frac{\mathrm{d}r^2}{1-kr^2} + r^2\left(\mathrm{d}\theta^2 + \sin^2\theta \mathrm{d}\phi^2\right)\right] \tag{2.6}$$

The Robertson-Walker metric is a solution based on the assumption that the spacetime spatial hypersurface is homogeneous and isotropic, in general, we assume that the spatial hypersurface is sphere symmetric under the condition of maximally symmetric space.
Then, we can modify the coordinates by slightly changing the scale with the constant curvature(after all, the multipled coefficient doesn't matter) By combining Robertson-Walker metric with Einstein field equations, we can give a quantitative description of scale factor evolution

$$T_{\mu\nu} = (p+\rho)U_\mu U_\nu + p g_{\mu\nu} \tag{2.7}$$

The above formula is based on the rest framework, while in the isotropic and homogeneous spacetime, the rest framework will coincide with hte comoving coordinates.

2.5 Projection Tensor

In most cases of numerical relativity, we commonly focus on the 3D hypersurface evolution. The physical quantity projected onto the hypersurface thus becomes sufficient important. The general projection opertor(tensor):

$$^{2}P^a_b = \delta^a_b + n^a n_b \tag{2.8}$$

The above opertor acts on the 4D physical quantity, whose results are still defined on the 4D spacetime. To recover the hypersurface 3D version tensor, we have two methods:

The first one
We have the vector e^α_a directly acts on the tensor defined on the spacetime

The second one
After obtaining the projected results from the projection operator, we perform the projection using the "vector"

When we are concentrating on the spacelike hypersurface, the norm of $n^a n_a = -1$. We obtain two version of projection tensor, acting on the upper and lower indices:

- $h_{\alpha\beta} = g_{\alpha\beta} + n_\alpha n_\beta$
- $h_{ab} = h_{\alpha\beta}e^\alpha_a e^\beta_b = g_{\alpha\beta}e^\alpha_a e^\beta_b = h_{ab}$

To gain a unified viewpoint of the projection concepts, we infer to the intrinsic convariant derivative as follows:

$$^{3}A_{a|b} \equiv A_{\alpha;\beta}e^\alpha_a e^\beta_b \tag{2.9}$$

[2] where n^a is the normal vector of the hypersurface
[3] where A_α is defined on the hypersurface

2.5.1 Intrinsic Covariant Derivative

Once we have a tangent tensor defined on the hypersurface(spacelike or null), we could define the covariant derivative as the spacetime as follows:

$$D_\rho T^{\alpha_1\dots\alpha_p}_{\beta_1\dots\beta_q} = \gamma^{\alpha_1}_{\mu_1}\cdots\gamma^{\alpha_p}_{\mu_p}\gamma^{v_1}_{\beta_1}\cdots\gamma^{v_q}\beta_q\gamma^\sigma_\rho\nabla_\sigma T^{\mu_1\dots\mu_p}_{v_1\dots v_q} \tag{2.10}$$

On the equation (2.10), the RHS is the 4D version of the tensor on the hypersurface, the LHS is the 3D version tensor recovered to the 4D spacetime.

In order to give the self-defined explanation of the definition, we could verify two things.

- The symmetric property when acted on the scalar field function
- The metric is compatible to the covariant derivative(since the 3D version of the covariant derivative is unique when the induced metirc is given)

The rigorous version of the (2.10) is:

$$\vec{\gamma}^*_{\mathscr{M}}\boldsymbol{DT} = \overrightarrow{\gamma}^*\left[\nabla\left(\overrightarrow{\gamma}^*_{\mathscr{M}}\boldsymbol{T}\right)\right] \tag{2.11}$$

The induced metric is compatible to the definition of covariant derivative as follows:

$$\begin{aligned}(\vec{\gamma}^*\nabla\gamma)_{\alpha\beta\gamma} &= \gamma^\mu_\alpha\gamma^v_\beta\gamma^\rho_\gamma\nabla_\rho\gamma\mu v\\ &= \gamma^\mu_\alpha\gamma^v_\beta\gamma^\rho_\gamma(\underbrace{\nabla_\rho g_{\mu\nu}}_{0}+\nabla_\rho n_\mu n_v + n_\mu\nabla_\rho n_v)\\ &= \gamma^\rho_\gamma(\gamma^\mu_\alpha\underbrace{\gamma^v_\beta n_v}_{0}\nabla_\rho n_\mu + \underbrace{\gamma^\mu_\alpha n_\mu}_{0}\nabla_\rho n_v)\\ &= 0\end{aligned} \tag{2.12}$$

2.5.2 Extrinsic Curvature

Equipped with the normal vector defined on the hypersurface, we could define the extrinsic curvature:

$$\chi : \mathscr{T}_p(\Sigma) \longrightarrow \mathscr{T}_p(\Sigma) \tag{2.13}$$

$$\boldsymbol{v} \longmapsto \boldsymbol{\nabla_v n} \tag{2.14}$$

According to this definition, this tensor(explictly vector) is tangent to the hypersurface,also we could verify the symmetric property of the tensor as:

$$^{4}\boldsymbol{K}(\boldsymbol{u},\boldsymbol{v}) = -\boldsymbol{u}\cdot\nabla_{\boldsymbol{v}}\boldsymbol{n} \tag{2.15}$$

On the above equation, we could refer to the projection operator(or inner product) to give a proper forms(the linear combinations).This means that, although we could treat the extrinsic curvature as a retail store to get the input of two tangent vectors defined on the hypersurface to give the output of the scalar result.(which is a 4D viewpoint), we could also directly project the extrinsic curvature tensor into the hypersurface to achive the 3D version to act on the 3-component vectors.

$$^{5}\nabla_\beta(n^\alpha n_\alpha) == 0$$

Four Velocoty Property Under the geometric units, the norm of the four-veolocity is always 1.

Properties related to the extrinsic curvature:

$$\begin{aligned}\boldsymbol{a} &:= \boldsymbol{\nabla_n n}\\ \nabla_\beta n_\alpha &= -K_{\alpha\beta} - a_\alpha n_\beta\\ K &= -\nabla\cdot\boldsymbol{n}\\ \boldsymbol{D_u v} &= \boldsymbol{\nabla_u v} + \boldsymbol{K}(\boldsymbol{u},\boldsymbol{v})\boldsymbol{n}(4Dversion)\end{aligned} \tag{2.16}$$

[4] the minus sign comes from the dual vector to define the $(0,2)tensor$

[5] Slicing the whole spacetime, the scalar field is constant define on the spacetime

Remember Things

The extrinsic curvature tensor generally is a tensor defined on the spacetime, but tangent to the hypersurface. To get the concrete component forms, we could adapt the hypersurface-normal coordinates, explictly 3-components "4D tensor". When projected, it becomes entirely 3-components tensor.

$$\gamma^{\mu}_{\alpha}\gamma^{v}_{\beta}\nabla_{\mu}n_{v} = -K_{\beta\alpha} \tag{2.17}$$

2.5.3 Gauss-Codazzi Relations

Gauss Relations

This formluas give a relationship between intrinsic covariant derivative, 4D curvature tensor, 3D curvature tensor,etc.

$$\begin{aligned}
D_{\alpha}D_{\beta}v^{\gamma} &= \gamma^{\mu}_{\alpha}\gamma^{v}_{\beta}\gamma^{\gamma}_{\rho}(n^{\sigma}\nabla_{\mu}n_{v}\gamma^{\rho}_{\lambda}\nabla_{\sigma}v^{\lambda} + \gamma^{\sigma}_{v}\nabla_{\mu}n^{\rho}\underbrace{n_{\lambda}\nabla_{\sigma}v^{\lambda}}_{-v^{\lambda}\nabla_{\sigma}n_{\lambda}} + \gamma^{\sigma}_{v}\gamma^{\rho}_{\lambda}\nabla_{\mu}\nabla_{\sigma}v^{\lambda}) \\
&= \gamma^{\mu}_{\alpha}\gamma^{v}_{\beta}\gamma^{\gamma}_{\lambda}\nabla_{\mu}n_{v}n^{\sigma}\nabla_{\sigma}v^{\lambda} - \gamma^{\mu}_{\alpha}\gamma^{\sigma}_{\beta}\gamma^{\gamma}\rho^{\lambda}\nabla_{\mu}n^{\rho}\nabla_{\sigma}n_{\lambda} + \gamma^{\mu}_{\alpha}\gamma^{\sigma}_{\beta}\gamma^{\gamma}_{\lambda}\nabla_{\mu}\nabla_{\sigma}v^{\lambda} \\
&= -K_{\alpha\beta}\gamma^{\gamma}_{\lambda}n^{\sigma}\nabla_{\sigma}v^{\lambda} - K^{\gamma}_{\alpha}K_{\beta\lambda}v^{\lambda} + \gamma^{\mu}_{\alpha}\gamma^{\sigma}_{\beta}\gamma^{\gamma}_{\lambda}\nabla_{\mu}\nabla_{\sigma}v^{\lambda} \\
&= \left(K_{\alpha\mu}K^{\gamma}_{\beta} - K_{\beta\mu}K^{\gamma}_{\alpha}\right)v^{\mu} + \gamma^{\rho}_{\alpha}\gamma^{\sigma}_{\beta}\gamma^{\gamma}_{\lambda}\left(\nabla_{\rho}\nabla_{\sigma}v^{\lambda} - \nabla_{\sigma}\nabla_{\rho}v^{\lambda}\right) \\
(K_{\alpha\mu}K^{\gamma}\beta - K_{\beta\mu}K^{\gamma}_{\alpha})v^{\mu} + \gamma^{\rho}_{\alpha}\gamma^{\sigma}\beta\gamma^{\gamma}\lambda^{4}R^{\lambda}R^{\lambda}_{\mu\rho\sigma}v^{\mu} &= R^{\gamma}_{\mu\alpha\beta}v^{\mu}
\end{aligned} \tag{2.18}$$

[6]

And the Gauss Relation is:

$$[7]\gamma^{\mu}_{\alpha}\gamma^{v}_{\beta}\gamma^{\gamma}_{\rho}\gamma^{4}_{\lambda}R^{\rho}_{\sigma\mu\nu} = R^{\gamma}_{\lambda\alpha\beta} + \left(K^{\gamma}_{\alpha}K_{\lambda\beta} - K^{\gamma}_{\beta}K_{\alpha\lambda}\right) \tag{2.19}$$

The contracted version is:

$$\gamma^{\mu}_{\alpha}\gamma^{v}_{\beta}\beta^{4}R_{\mu\nu} + \gamma_{\alpha\mu}n^{v}\gamma^{\rho}_{\beta}n^{\sigma 4}R^{\mu}_{v\rho\sigma} = R_{\alpha\beta} + KK_{\alpha\beta} - K_{\alpha\mu}K^{\mu}_{\beta} \tag{2.20}$$

The scalar Gaussian Relations:

$${}^{4}R + 2^{4}R_{\mu\nu}n^{\mu}n^{v} = R + K^{2} - K_{ij}K^{ij} \tag{2.21}$$

Codazzi Relations

Codazzi Relations seeks derivative of the normal vector defined on the hypersurface:

$$\left(\nabla_{\alpha}\nabla_{\beta} - \nabla_{\beta}\nabla_{\alpha}\right)n^{\gamma} =^{4} R^{\gamma}_{\mu\alpha\beta}n^{\mu} \tag{2.22}$$

If we project this relation onto Σ, we get:

$$\gamma^{\mu}_{\alpha}\gamma^{v}\beta\gamma^{\gamma}\rho^{4}R^{\rho}_{\sigma\mu\nu}n^{\sigma} = \gamma^{\mu}_{\alpha}\gamma^{v}\beta\gamma^{\gamma}\rho\left(\nabla_{\mu}\nabla_{v}n^{\rho} - \nabla_{v}\nabla_{\mu}n^{\rho}\right) \tag{2.23}$$

Codazzi Relations:

$$[8]\gamma^{\gamma}_{\rho}n^{\sigma}\gamma^{\mu}_{\alpha}\gamma^{v4}_{\beta}R^{\rho}_{\sigma\mu\nu} = D_{\beta}K^{\gamma}_{\alpha} - D_{\alpha}K^{\gamma}_{\beta} \tag{2.24}$$

Contracted Codazzi Relation:

$$\gamma^{\mu}_{\alpha}n^{v4}R_{\mu\nu} = D_{\alpha}K - D_{\mu}K^{\mu}_{\alpha} \tag{2.25}$$

2.6 Foliation of Spacetime

Giving a foliation of spacetime, we could define a hypersurface denoted by a constant scalar field together with a normalized normal vector.

$$[9]n := -N\vec{\nabla}t \tag{2.26}$$

[6] all this formulas are 4D version ommiting projecting and recovering
[7] $R^{\gamma}_{\lambda\alpha\beta}$ needs to be recovered into 4D version
[8] The intrinsic covariant derivative always acts on the tangent tensor!
[9] which satisfied the condition: $\boldsymbol{n}\cdot\boldsymbol{n} = -1$

And the scalar field N hence defined is called the lapse function($N > 0$) to give a compatible defintion of spacelike foliation. Based on the explanation of above equation, we could construct a adapted coordinates based on the foliation:

$$\boldsymbol{m} := N\boldsymbol{n}$$

The hypersurface can be entirely Lie dragged into to increasing-parameter hypersurface.

Since $\vec{n}$ is a unit timelike vector normal to the hypersurface, the hypersurface Σ_t is locally the set of events that are simultaneous from the point of view of the Eulerian observer. And the local coordinates in the viewpoint of the observer is just like Eulerian Spacetime.

$$\delta\tau = \sqrt{-\boldsymbol{g}(\delta t\boldsymbol{m}, \delta t\boldsymbol{m})} = \sqrt{-\boldsymbol{g}(\boldsymbol{m}, \boldsymbol{m})}\delta t = N\delta t \tag{2.27}$$

The (2.27) can give a clear explanation of the lapse function, when selecting the foliation coordinates.

$$\begin{aligned} a_\alpha &= n^\mu \nabla_\mu n_\alpha \\ &= \frac{1}{N}\left(\nabla_\alpha N + n_\alpha n^\mu \nabla_\mu N\right) \\ &= \frac{1}{N}\gamma^\mu_\alpha \nabla_\mu N = \frac{1}{N} D_\alpha N = D_\alpha \ln N \end{aligned} \tag{2.28}$$

As a result, we could assoiate with former formlua to give:

$$\nabla_\beta n_\alpha = -K_{\alpha\beta} - D_\alpha \ln N n_\beta \tag{2.29}$$

2.6.1 The evolution of 3-Metric

On the viewpoint of the timelike observer, the induced metric evolves as:

$$\begin{aligned} \mathscr{L}_{\boldsymbol{m}}\gamma_{\alpha\beta} &= m^\mu \nabla_\mu \gamma_{\alpha\beta} + \gamma_{\mu\beta}\nabla_\alpha m^\mu + \gamma_{\alpha\mu}\nabla_\beta m^\mu \\ &= -2NK_{\alpha\beta} \\ \mathscr{L}_m \gamma_{\alpha\beta} &= \mathscr{L}_{Nn}\gamma_{\alpha\beta} \\ &= N\mathscr{L}_n \gamma_{\alpha\beta} \end{aligned} \tag{2.30}$$

The (2.30) give a important result related to the extrinsic curvature tensor:

$$\boldsymbol{K} = -\frac{1}{2}\mathscr{L}_n \boldsymbol{\gamma}$$

There are some simple rules to remember:

- The vector $\vec{n}$ is just a normalized vector from normal vecotr;
- The vector $\vec{m}$ is exactly following the coordinates of spacetime foliation along the observer.

Small Tips

The orthogonal projection operator is just a $(1,1)$ tensor object,although we always focus the component when making use of it.

Evolution of the Orthogonal Projector

As a result, we can show that the Lie derivative of the orthogonal projection operator(tensor) is just:

$$\mathscr{L}_m \vec{\gamma} = 0 \tag{2.31}$$

Actually, from the the equation (2.31), we can deduce an important result:

$$^{10}\boldsymbol{T} \text{ tangent to } \Sigma_t \Longrightarrow \mathscr{L}_{\boldsymbol{m}}\boldsymbol{T} \text{ tangent to } \Sigma_t$$

[10] Think about the Lie dragged effect of a tangent tensor

2.6.2 the 3+1 Decomposition of the Riemann Tensor

We project the 4D Riemann tensor only twice onto Σ_t and once along $\boldsymbol{n}$:

$$\begin{aligned}
\gamma_{\alpha\mu}n^\sigma\gamma^v\beta^4R^\mu_{\rho v\sigma}n^\rho &= \gamma_{\alpha\mu}n^\sigma\gamma^v\beta\left(\nabla_v\nabla_\sigma n^\mu-\nabla_\sigma\nabla_v n^\mu\right)\\
&= -K_{\alpha\sigma}K^\sigma\beta+\frac{1}{N}D_\beta D_\alpha N+\gamma^\mu_\alpha\gamma^v\beta n^\sigma\nabla_\sigma K_{\mu\nu}\\
\mathscr{L}_{\boldsymbol{m}}K_{\alpha\beta} &= m^\mu\nabla_\mu K_{\alpha\beta}+K_{\mu\beta}\nabla_\alpha m^\mu+K_{\alpha\mu}\nabla_\beta m^\mu\\
\mathscr{L}_m K_{\alpha\beta} &= N\gamma^\mu_\alpha\gamma^v_\beta n^\sigma\nabla_\sigma K_{\mu\nu}-2NK_{\alpha\mu}K^\mu_\beta\\
\Longrightarrow \gamma^\mu_\alpha\gamma^v_\beta{}^4R_{\mu\nu} &= -\frac{1}{N}\mathscr{L}_m K_{\alpha\beta}-\frac{1}{N}D_\alpha D_\beta N+R_{\alpha\beta}+KK_{\alpha\beta}-2K_{\alpha\mu}K^\mu_\beta
\end{aligned}\tag{2.32}$$

Directly contracting based on the above final equation:

$$\begin{aligned}
\gamma^{\mu v}{}^4R_{\mu\nu} &= -\frac{1}{N}\gamma^{ij}\mathscr{L}_m K_{ij}-\frac{1}{N}D_iD^iN+R+K^2-2K_{ij}K^{ij}\\
\text{with }\mathscr{L}_m\gamma^{ij} &= 2NK^{ij}\\
-\gamma^{ij}\mathscr{L}_m K_{ij} &= -\mathscr{L}_m K+2NK_{ij}K^{ij}\\
{}^4R+{}^4R_{\mu\nu}n^\mu n^v &= R+K^2-\frac{1}{N}\mathscr{L}_m K-\frac{1}{N}D_iD^iN
\end{aligned}\tag{2.33}$$

2.7 Summary

Some useful formulas to be used in the future:

$$\begin{aligned}
\mathscr{L}_m\gamma^{ij} &= 2NK^{ij}\\
\mathscr{L}_m K_{\alpha\beta} &= N\gamma^\mu_\alpha\gamma^v_\beta n^\sigma\nabla_\sigma K_{\mu\nu}-2NK_{\alpha\mu}K^\mu_\beta\\
\boldsymbol{K} &= -\frac{1}{2}\mathscr{L}_{\boldsymbol{n}}\boldsymbol{\gamma}
\end{aligned}\tag{2.34}$$

2.8 Supplementing Proof

2.8.1 What is the $h^{\alpha\beta}$

Prior to discussing ADM formlism, we introduce the concept of projecting operator and induced matric, in the special case of spacelike hypersurface, we have:

$$h_{\alpha\beta}=g_{\alpha\beta}+n_\alpha n_\beta$$

There is a question, how can we give the definition of $h^{\alpha\beta}$ according to the above equation.
The principle is:

- $h_{\alpha\beta}$ is still 4D version tensor, its upper-index correspondent is just raised by the inverse metric;
- $h_{\alpha\beta}$ is just a measure of the norm of vector in spacetime, although in the hypersurface;

Thus, the final result is just:[11]

$$h^{\mu\gamma}=g_{\alpha\beta}+n_\alpha n_\beta(g^{\alpha\mu}g^{\beta\gamma})\neq h^{-1}_{\alpha\beta}\tag{2.35}$$

2.8.2 Deduction — A

In Possion's book, he gives such a equation:

$$\left(\varepsilon n^\alpha n^\beta+h^{\alpha\beta}\right)n_{\alpha;\beta}=h^{\alpha\beta}n_{\alpha;\beta}$$

[11] Not the INVERSE of $h_{\alpha\beta}$

The question is why:

$$n^\alpha n^\beta n_{\alpha;\beta} = 0 \tag{2.36}$$

The proof as follows:

$$n^\alpha n^\beta \nabla_\beta n_\alpha = n^\alpha n^\beta \nabla_\beta g_{\alpha\mu} n^\mu = n_\mu n^\beta \nabla_\beta n^\mu$$

Then, we have:

$$n^\alpha n^\beta n_{\alpha;\beta} = \frac{1}{2}(n^\alpha n^\beta n_{\alpha;\beta} + n_\alpha n^\beta \nabla_\beta n^\alpha) = \frac{1}{2}(n^\beta \nabla_\beta(-1)) = 0 \tag{2.37}$$

2.8.3 Deduction — B

$$h^{\alpha\beta}\delta(g_{\gamma\beta})_{,\alpha} = 0$$

The above equation only comes into existence on the hypersurface, here we give a relative portable proof:

$$\begin{aligned} {}_{12}h^{\alpha\beta} &= h^{ab} e^\alpha_a e^\beta_b \\ 0 &= \delta(g_{\gamma\beta})_{,\alpha} e^\alpha_a \end{aligned} \tag{2.38}$$

[12] We only need to swith between the adpative coordinates and the original coordinates when we come into the hypersurface

Chapter 3

Towards BSSN Formulas

3.1 Fully Projection of the Einstein Equation

The Einstein Field equation (without considering the cosmological cosntant) is:

$$ {}^4\boldsymbol{R} - \frac{1}{2}{}^4 R\boldsymbol{g} = 8\pi \boldsymbol{T} \tag{3.1} $$

Since we need to evolve the Einstein equation (3.1) in the $3+1$ formlism, we needs to project it onto spacelike hypersurface and normal vector:

Hypersurface

[1]
$$ \begin{aligned} \vec{\gamma}^* \boldsymbol{R} &= 8\pi \left(\vec{\boldsymbol{v}}^* \boldsymbol{T} - \frac{1}{2} T \vec{\boldsymbol{\gamma}}^* \boldsymbol{g} \right) \\ \mathscr{L}_{\boldsymbol{m}} K_{\alpha\beta} &= -D_\alpha D_\beta N + N \left\{ R_{\alpha\beta} + K K_{\alpha\beta} - 2K_{\alpha\mu} K^\mu \beta + 4\pi \left[(S-E)\gamma_{\alpha\beta} - 2S_{\alpha\beta} \right] \right\} \end{aligned} \tag{3.2} $$

Entirely Hypersurface Version $\Longrightarrow$ Do a component projection using E_a^α.

$$ {}^2\boldsymbol{T} = \boldsymbol{S} + \underline{\boldsymbol{n}} \otimes \boldsymbol{p} + \boldsymbol{p} \otimes \underline{\boldsymbol{n}} + E\boldsymbol{n} \otimes \underline{\boldsymbol{n}} $$

Normal vector

$$ \begin{aligned} {}^4\boldsymbol{R}(\boldsymbol{n}, \boldsymbol{n}) + \frac{1}{2}{}^4 R &= 8\pi \boldsymbol{T}(\boldsymbol{n}, \boldsymbol{n}) \\ R + K^2 - K_{ij} K^{ij} &= 16\pi E \end{aligned} \tag{3.3} $$

[3]

In the above equation,the EM tensor discussed in the Chapter-2 gives an illustration about the components meaning.

3.2 Coordinates Adapted to the Foliation

Once we have a proper foliation, we can make use of it to create an well-defined coordinates.The idea goes as follow:

- On the hypersurface, we create a coordinates $\left(x^1, x^2, x^3\right)$;
- The hypersurface function t as another component(integral line) to Lie drag the hypersurface coordinates;

We shall call $\left(x^i\right) = \left(x^1, x^2, x^3\right)$ the spatial coordinates,and the coordinate basis vectors are:

$$ \begin{aligned} \partial_t &:= \frac{\partial}{\partial t} \\ \boldsymbol{\partial}_i &:= \frac{\partial}{\partial x^i}, \quad i \in \{1, 2, 3\} \end{aligned} \tag{3.4} $$

Notice The coordinate "time" is possibly not timelike, but the normal vector is certainly timelike forever.

[1] Combined with the Gauss Relations,and all these tensors are recover to 4D version

[2] Image what the observer would watch!

A vivid illustration as:

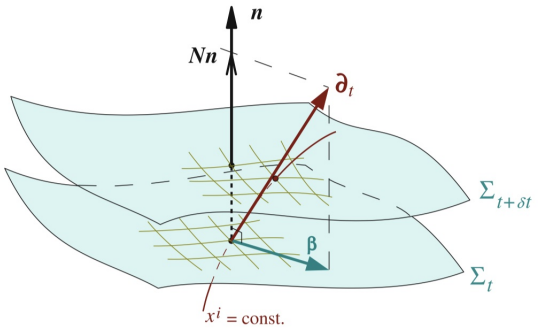


Figure 3.1: Adapted Coordinates For Foliation

Baed on the selected coordinates, we can decompose the time component coordinates:

$$^{4}\boldsymbol{\partial}_t =: \boldsymbol{m} + \boldsymbol{\beta}$$

Thus, a natural choice for the $\boldsymbol{\beta}$ is:

$$\boldsymbol{\beta} =: \beta^i \boldsymbol{\partial}_i \quad \text{and} \quad \underline{\boldsymbol{\beta}} =: \beta_i \mathbf{d}x^i$$

The metric in this coordinates:

$$g_{\alpha\beta} = \begin{pmatrix} g_{00} & g_{0j} \\ g_{i0} & g_{ij} \end{pmatrix} = \begin{pmatrix} -N^2 + \beta_k\beta^k & \beta_j \\ \beta_i & \gamma_{ij} \end{pmatrix} \tag{3.5}$$

The inverse metric in this coordinates:

$$g^{\alpha\beta} = \begin{pmatrix} g^{00} & g^{0j} \\ g^{i0} & g^{ij} \end{pmatrix} = \begin{pmatrix} -\frac{1}{N^2} & \frac{\beta^j}{N^2} \\ \frac{\beta^i}{N^2} & \gamma^{ij} - \frac{\beta^i\beta^j}{N^2} \end{pmatrix} \tag{3.6}$$

Or the metric in the components form:

$$g_{\mu\nu}\mathrm{d}x^\mu \mathrm{d}x^\nu = -N^2\mathrm{d}t^2 + \gamma_{ij}\left(\mathrm{d}x^i + \beta^i\mathrm{d}t\right)\left(\mathrm{d}x^j + \beta^j\mathrm{d}t\right) \tag{3.7}$$

Another useful formula in numerical simulations:

$$\sqrt{-g} = N\sqrt{\gamma} \tag{3.8}$$

3.3 Einstein Equation in 3+1 Formlism

There are a lot of benefits to adapt this special kinds of coordinates, one of them is that we can express various derivative in just the 3-components form regarless the extra requirements.

$$\begin{gathered} \mathscr{L}_m T^{i\cdots}_{j\cdots} = \left(\frac{\partial}{\partial t} - \mathscr{L}_\beta\right) T^{i\cdots}_{j\cdots} \\ \text{where the tensor } T^{i\cdots}_{j\cdots} \text{ is tangent to the hypersurface} \\ \Rightarrow \mathscr{L}_{\boldsymbol{m}} K_{ij} = \left(\frac{\partial}{\partial t} - \mathscr{L}_\beta\right) K_{ij} \\ \text{Important Equation: } \left(\frac{\partial}{\partial t} - \mathscr{L}_\beta\right)\gamma_{ij} = -2NK_{ij} \end{gathered} \tag{3.9}$$

[4] We can verify the vector $\boldsymbol{\beta}$ is tangent to the hypersurface

Comined the equation (3.9) with former constraint condition exerted by the EM tensor, we could get a complete description of the spacetime metric evolution:

$$\begin{aligned}\left(\frac{\partial}{\partial t}-\mathscr{L}_\beta\right)\gamma_{ij} &= -2NK_{ij}\\ \left(\frac{\partial}{\partial t}-\mathscr{L}_\beta\right)K_{ij} &= -D_iD_jN+N\left\{R_{ij}+KK_{ij}-2K_{ik}K^k_j+4\pi\left[(S-E)\gamma_{ij}-2S_{ij}\right]\right\}\\ R+K^2-K_{ij}K^{ij} &= 16\pi E\\ D_jK^j_i-D_iK &= 8\pi p_i\end{aligned} \tag{3.10}$$

Notes

Acted objects in the above equation (3.10) can be exactly expressed in the 3-componets forms due to the adaption of the special coordinates. As a results, the evolution equation is indeed a 3+1 equation with the hypersurface along the time coordinates.
We shall concentrate on the elements in the equation (3.10) $\mathscr{L}_\beta K_{ij}$, whose usual behaviour needs to be reprensented in the 4D version coordinates, than projecting into the hypersurface(although the results must be tangent to the hypersurface), but in this special coordinates, we can entirely ignore the time-side to investigate the (3×3) behaviours!

3.4 BSSNOK Formlism

The BSSN (Baumgarte, Shapiro, Shibata and Nakamura) formulation gives an alternative to represent the ADM equations following a conformal metric transformation, which proves to be particularly robust in the numerical evolution of a large variety of spacetimes, both with and without matter.This equations are also called BSSNOK equations in memory of the intial creaters.
We begin with a conformal transformation:

$$\tilde{\gamma}_{ij} := \Psi^{-4}\gamma_{ij}$$

In the BSSNOK formulation, we choose the conformal factor in such a way that the conformal metric γ_{ij} has unit determinant.

$$\psi^4=\gamma^{1/3} \quad \Rightarrow \quad \psi=\gamma^{1/12} \tag{3.11}$$

As a result, we calulate the evolution of the 3D metric:

$$\partial_t\gamma=\gamma\left(-2\alpha K+2D_i\beta^i\right)=-2\gamma\left(\alpha K-\partial_i\beta^i\right)+\beta^i\partial_i\gamma \tag{3.12}$$

In the above equation (3.12), we shall make use of the result:

$$\begin{aligned}{}^{5}\frac{1}{2}g^{\mu\nu}\frac{\partial g_{\mu v}}{\partial x^\alpha} &= \frac{1}{2}\frac{\partial}{\partial x^\alpha}\ln|g|\\ \partial_t\gamma_{ij} &= -2\alpha K_{ij}+D_i\beta_j+D_j\beta_i\end{aligned} \tag{3.13}$$

3.4.1 Proof

Here, we give a detailed proof of above results (3.12):

$$\begin{aligned}\gamma^{ij}\partial_t\gamma_{ij}=\frac{\partial}{\partial t}ln(\gamma)&=\frac{1}{\gamma}\frac{\partial}{\partial t}\gamma\\ \Rightarrow\partial_t\gamma&=\gamma\left(-2\alpha K+2D_i\beta^i\right)\\ D_u\beta^v&=\frac{\partial}{\partial x^u}\beta^v+\Gamma^v_{um}\beta^m\\ \Rightarrow\Gamma^\mu_{\alpha\mu}&=\frac{1}{2}g^{\mu v}\frac{\partial g_{\mu v}}{\partial x^\alpha}\end{aligned} \tag{3.14}$$

Which reduces to $\beta^i\partial_i\gamma$ In the last term

[5] Very useful equation in GR derivations

Attention

The conformal metric $\tilde{\gamma}_{ij}$ is not a tensor, neither is the determinant of metric γ_{ij}. But they are one kind of tensor density, recalling the definition:

$$\boldsymbol{\tau} = \gamma^{n/2}\boldsymbol{T}$$

At least, we should keep in mind that it makes no sense to do various manipulation on some objects(which are not tensors!) So we need to make some modifications to our original defintion and test it based on the most of textbook materials.

3.4.2 Introduction of Background Metric

we introduce an extra structure on the hypersurfaces Σ_t namely a background metric f to fit the (3.4.1), and we ask the signature to be $(+,+,+)$ and Lie dragged along the time coordinate lines, which means:

$$\frac{\partial f_{ij}}{\partial t} = 0$$

Attenstion The extra metric is just another structure of the spacetime, certainly it will be helpful to clarify a lot of concepts.

Based on the extra structure, we recall the defintion of the conformal metric at the beginning (3.4), we can convert it into a real tensor by express Ψ as:

$$\boldsymbol{\Psi} := \left(\frac{\gamma}{f}\right)^{1/12} \quad \gamma := \det\left(\gamma_{ij}\right), \quad f := \det\left(f_{ij}\right) \tag{3.15}$$

By this construction, the determinant of conformal metric becomes:

$$\det\left(\tilde{\gamma}_{ij}\right) = f$$

Attention In many textbook, it is very common to see people manipulate the conformal metric directly(some obscure). But things become more clear, if we take the viewpoint that we always adapt the rules that the determinant of the extra metric is 1 forever.

3.4.3 Conformal Connection

Once we have a proper defintion of real conformal metric, we can construct the corresponding convariant derivative, Ricc tensor, etc. Here, we give a series formulas to illustrate the potential links with the original metric:

The Covariant Derivative $\tilde{\boldsymbol{D}}\tilde{\gamma} = 0$

Christoffel Symbols $\tilde{\Gamma}^k_{ij} = \frac{1}{2}\tilde{\gamma}^{kl}\left(\frac{\partial\tilde{\gamma}_{lj}}{\partial x^i} + \frac{\partial\tilde{\gamma}_{il}}{\partial x^j} - \frac{\partial\tilde{\gamma}_{ij}}{\partial x^l}\right)$

Difference of Covariant Derivative

$$D_k T^{i_1\ldots i_p}_{j_1\ldots j_q} = \tilde{D}_k T^{i_1\ldots i_p}_{j_1\ldots j_q} + \sum_{r=1}^{p} C^{i_r}_{kl} T^{i_1\ldots l\ldots i_p}_{j_1\ldots j_q} \tag{3.16}$$

where $C^k_{ij} := \Gamma^k_{ij} - \tilde{\Gamma}^k_{ij}$ is a real tensor, and its components are:

$$C^k_{ij} = \frac{1}{2}\gamma^{kl}\left(\tilde{D}_i\gamma_{lj} + \tilde{D}_j\gamma_{il} - \tilde{D}_l\gamma_{ij}\right) \tag{3.17}$$

Replacing the γ_{ij}, γ^{ij} with $\tilde{\gamma}_{ij}$, $\tilde{\gamma}^{ij}$ and Ψ, the more conscise expression:

$$C^k_{ij} = 2\left(\delta^k_i\tilde{D}_j\ln\Psi + \delta^k_j\tilde{D}_i\ln\Psi - \tilde{D}^k\ln\Psi\tilde{\gamma}_{ij}\right) \tag{3.18}$$

Ricci Tensor

$$\tilde{D}_j\tilde{D}_i v^j - \tilde{D}_i\tilde{D}_j v^j = \tilde{R}_{ij}v^j R_{ij} = \tilde{R}_{ij} + \tilde{D}_k C^k_{ij} \quad -\tilde{D}_i C^k_{kj} + C^k_{ij}C^l_{lk} - C^k_{il}C^l_{kj} \tag{3.19}$$

Following the same principle:

$$R_{ij} = \tilde{R}_{ij} - 2\tilde{D}_i\tilde{D}_j \ln\Psi - 2\tilde{D}_k\tilde{D}^k \ln\Psi\tilde{\gamma}_{ij} + 4\tilde{D}_i \ln\Psi\tilde{D}_j \ln\Psi - 4\tilde{D}_k \ln\Psi\tilde{D}^k \ln\Psi\tilde{D}_{ij} \tag{3.20}$$

Scalar Curvature

$$^6R = \Psi^{-4}\tilde{R} - 8\Psi^{-5}\tilde{D}_i\tilde{D}^i\Psi \tag{3.21}$$

Than we decompose the extrinsic curvature into traceless part:

$$\boldsymbol{A} := \boldsymbol{K} - \frac{1}{3}K\boldsymbol{\gamma}$$

Conformal Decomposition of the Traceless Part

Attention The conformal part of A_{ij} is just a definition, not a derivative results from the intial conformal metric.

$$A^{ij} = \Psi^\alpha \tilde{A}^{ij} \tag{3.22}$$

3.4.4 Natural Scaling

There are two natural scaling which we shall see later, following the time-evoltion and momentum constraint:

"Time-Evolution" Scaling

We begin with the time evolution of $\boldsymbol{\gamma}$ in terms of $\boldsymbol{K}$:

$$\mathscr{L}_m\left(\Psi^4\tilde{\gamma}_{ij}\right) = -2NA_{ij} - \frac{2}{3}NK\gamma_{ij} \tag{3.23}$$

Finally, we could get:

$$\left(\frac{\partial}{\partial t} - \mathscr{L}_\beta\right)\tilde{\gamma}_{ij} = -2N\tilde{A}_{ij} - \frac{2}{3}\tilde{D}_k\beta^k\tilde{\gamma}_{ij} \tag{3.24}$$

Note In the textbook of "Numerical Relativity: Solving Einstein Equations On the Computer", the author express the above equation as:

$$\left(\frac{\partial}{\partial t} - \mathscr{L}_\beta\right)\tilde{\gamma}_{ij} = -2N\tilde{A}_{ij} - \frac{2}{3}\partial_k\beta^k\tilde{\gamma}_{ij}$$

But we can verify, this equation is just a special case of the determinant of the background metric($det(f_{ij} = 1)$), using the tracted formulas:

$$\nabla\cdot\boldsymbol{v} = \frac{1}{\sqrt{|g|}}\frac{\partial}{\partial x^\mu}\left(\sqrt{|g|}v^\mu\right)$$

In the equation of (3.24), it suggests to introduce the quantity:

$$\tilde{A}_{ij} := \Psi^{-4}A_{ij}$$

to write:

$$\left(\frac{\partial}{\partial t} - \mathscr{L}_\beta\right)\tilde{\gamma}_{ij} = -2N\tilde{A}_{ij} - \frac{2}{3}\tilde{D}_k\beta^k\tilde{\gamma}_{ij} \tag{3.25}$$

We then define the corresponding up-index traceless part of $\boldsymbol{K}_{ij}$ as $\tilde{A}^{ij} := \tilde{\gamma}^{ik}\tilde{\gamma}^{jl}\tilde{A}_{kl}$

$$\left(\frac{\partial}{\partial t} - \mathscr{L}_\beta\right)\tilde{\gamma}^{ij} = 2N\tilde{A}^{ij} + \frac{2}{3}\tilde{D}_k\beta^k\tilde{\gamma}^{ij} \tag{3.26}$$

[6] The defintion: $\tilde{R} := \tilde{\gamma}^{ij}\tilde{R}_{ij}$

Momentum-Constraints Scaling

For the original momentum constrain equation following a conformal transformation:

$$\tilde{D}_j \hat{A}^{ij} - \frac{2}{3}\Psi^6 \tilde{D}^i K = 8\pi \Psi^{10} p^i \tag{3.27}$$

Where in the equation (3.27), we define the relationship:

$$\hat{A}^{ij} := \Psi^{10} A^{ij}$$

3.5 Conformal Form of the 3+1 Einstein Equations

3.5.1 Dynamical Part of Einstein Equations

We will use three equations to construct the entire conformal forms:

$$\begin{aligned}
&\textbf{One } \left(\frac{\partial}{\partial t} - \mathscr{L}_\beta\right)\gamma_{ij} = -2NK_{ij} \\
&\textbf{Two } \left(\frac{\partial}{\partial t} - \mathscr{L}_\beta\right)K = -D_i D^i N + N\left[4\pi(E+S) + K_{ij}K^{ij}\right] \\
&\textbf{As Well as } \mathscr{L}_{\boldsymbol{m}} A_{ij} = -\boldsymbol{D}_i D_j N + N\left[\boldsymbol{R}_{ij} + \frac{1}{3}KA_{ij} - 2A_{ik}A^k_j - 8\pi\left(S_{ij} - \frac{1}{3}S\gamma_{ij}\right)\right] + \frac{1}{3}\left(D_k D^k N - NR\right)\gamma_{ij}
\end{aligned} \tag{3.28}$$

Just to mention a few, there maybe two types of equal-class expression of the "BSSN" Formulas:

3.5.2 Two Kinds of Represention

Numerical Relativity: Solving Einstein Equations On the Computer

Common in most of Numerical Relativity literature:[7]

$$\begin{aligned}
\partial_t \phi &= -\frac{1}{6}\alpha K + \beta^i \partial_i \phi + \frac{1}{6}\partial_i \beta^i \\
\partial_t \overline{\gamma}_{ij} &= -2\alpha \tilde{A}_{ij} + \beta^k \partial_k \overline{\gamma}_{ij} + \overline{\gamma}_{ik}\partial_j \beta^k + \overline{\gamma}_{kj}\partial_i \beta^k - \frac{2}{3}\overline{\gamma}_{ij}\partial_k \beta^k \\
\partial_t K &= -\gamma^{ij} D_j D_i \alpha + \alpha\left(\tilde{A}_{ij}\tilde{A}^{ij} + \frac{1}{3}K^2\right) + 4\pi\alpha(\rho + S) + \beta^i \partial_i K \\
\partial_t \tilde{A}_{ij} &= e^{-4\phi}\left(-\left(D_i D_j \alpha\right)^{TF} + \alpha\left(R^{TF}_{ij} - 8\pi S^{TF}_{ij}\right)\right) + \alpha\left(K\tilde{A}_{ij} - 2\tilde{A}_{il}\tilde{A}^l_j\right) \\
&\quad + \beta^k \partial_k \tilde{A}_{ij} + \tilde{A}_{ik}\partial_j \beta^k + \tilde{A}_{kj}\partial_i \beta^k - \frac{2}{3}\tilde{A}_{ij}\partial_k \beta^k
\end{aligned} \tag{3.29}$$

Another kinds of expression, although seemingly very different from the above, we can prove that they are totally equvilent.

[7] All the conformal part of A_{ij} base on the choice of $\alpha = -4$

Bases of Numerical Relativity

This book gives a very detailed derivations:

$$\begin{aligned}
\left(\frac{\partial}{\partial t}-\mathscr{L}_\beta\right)\Psi &= \frac{\Psi}{6}\left(\tilde{D}_i\beta^i - NK\right)\\
\left(\frac{\partial}{\partial t}-\mathscr{L}_\beta\right)\tilde{\gamma}_{ij} &= -2N\tilde{A}_{ij}-\frac{2}{3}\tilde{D}_k\beta^k\tilde{\gamma}_{ij}\\
\left(\frac{\partial}{\partial t}-\mathscr{L}_\beta\right)K &= -\Psi^{-4}\left(\tilde{D}_i\tilde{D}^iN+2\tilde{D}_i\ln\Psi\tilde{D}^iN\right)+N\left[4\pi(E+S)+\tilde{A}_{ij}\tilde{A}^{ij}+\frac{K^2}{3}\right]\\
\left(\frac{\partial}{\partial t}-\mathscr{L}_\beta\right)\tilde{A}_{ij} &= -\frac{2}{3}\tilde{D}_k\beta^k\tilde{A}_{ij}+N\left[K\tilde{A}_{ij}-2\tilde{\gamma}^{kl}\tilde{A}_{ik}\tilde{A}_{jl}-8\pi\left(\Psi^{-4}S_{ij}-\frac{S}{3}\tilde{\gamma}_{ij}\right)\right]\\
&\quad+\Psi^{-4}\{-\tilde{D}_i\tilde{D}_jN+2\tilde{D}_i\ln\psi\tilde{D}_jN+2\tilde{D}_j\ln\Psi\tilde{D}_iN+\frac{1}{3}\left(\tilde{D}_k\tilde{D}^kN-4\tilde{D}_k\ln\Psi\tilde{D}^kN\right)\tilde{\gamma}_{ij}\\
&\quad+N[\tilde{R}_{ij}-\frac{1}{3}\tilde{R}\tilde{\gamma}_{ij}-2\tilde{D}_i\tilde{D}_j\ln\Psi+4\tilde{D}_i\ln\Psi\tilde{D}_j\ln\Psi+\frac{2}{3}\left(\tilde{D}_k\tilde{D}^k\ln\Psi-2\tilde{D}_k\ln\Psi\tilde{D}^k\ln\Psi\right)\tilde{\gamma}_{ij}]\}
\end{aligned}\tag{3.30}$$

Here, we give a detailed proof about the equivalence between these two versions of equations.
Some useful formulas are:

The Evolution of Trace-free Part of K_{ij}

$$\mathscr{L}_{\boldsymbol{m}}A_{ij}=-D_iD_jN+N\left[R_{ij}+\frac{1}{3}KA_{ij}-2A_{ik}A^k_j-8\pi\left(S_{ij}-\frac{1}{3}S\gamma_{ij}\right)\right]+\frac{1}{3}\left(D_kD^kN-NR\right)\gamma_{ij}$$

Contracted Derivative

$$D_iD^iN=\Psi^{-4}\left(\tilde{D}_i\tilde{D}^iN+2\tilde{D}_i\ln\Psi\tilde{D}^iN\right)$$

Inner Product of K_{ij}

$$K_{ij}K^{ij}=\left(A_{ij}+\frac{K}{3}\gamma_{ij}\right)\left(A^{ij}+\frac{K}{3}\gamma^{ij}\right)=A_{ij}A^{ij}+\frac{K^2}{3}=\tilde{A}_{ij}\dot{A}^{ij}+\frac{K^2}{3}$$

Double Derivative of Lapse Function

$$D_iD^iN=\tilde{D}_i\tilde{D}_jN-2\left(\tilde{D}_i\ln\Psi\tilde{D}_jN+\tilde{D}_j\ln\Psi\tilde{D}_iN-\tilde{D}^k\ln\Psi\tilde{D}_kN\tilde{\gamma}_{ij}\right)$$

Key Points There are some points needs to be solved:

- How to evaluate the result $\mathscr{L}_\beta\Psi$;
- How to manipulate the complicated combinations of the last terms in the (3.30);
- How to links various conformal parts and non-comformal parts like (R_{ij},$\tilde{R}_{ij}$) and (R,$\tilde{R}$);

If you shall be careful and patient, you can at last prove the equivalence between the equation (3.29) and (3.30)!

3.5.3 Conformal Constraint Equations

Hamiltonian Constraint

Using former results we got, we can express the Hamiltonian Constriant Equations as:

$$\tilde{D}_i\tilde{D}^i\Psi-\frac{1}{8}\tilde{R}\Psi+\left(\frac{1}{8}\tilde{A}_{ij}\tilde{A}^{ij}-\frac{1}{12}K^2+2\pi E\right)\Psi^5=0\tag{3.31}$$

In order to obtain the initial **Lichnerowicz equation**, we exert the transformation ($\hat{A}^{ij}=\Psi^6\tilde{A}^{ij}$) and ($\hat{A}_{ij}=\Psi^6\tilde{A}_{ij}$)
Then we obtain:

$$\tilde{D}_i\tilde{D}^i\Psi-\frac{1}{8}\tilde{R}\Psi+\frac{1}{8}\hat{A}_{ij}\hat{A}^{ij}\Psi^{-7}+\left(2\pi E-\frac{1}{12}K^2\right)\Psi^5=0\tag{3.32}$$

Momentum Constraint

In the section of MOMENTUM-CONSTRAINT SCALING, we actually obtain the equation in terms of $\hat{A}_{ij}$, to recover the $\tilde{A}_{ij}$ version:

$$\tilde{D}_j \tilde{A}^{ij} + 6\tilde{A}^{ij}\tilde{D}_j \ln \Psi - \frac{2}{3}\tilde{D}^i K = 8\pi \Psi^4 p^i \tag{3.33}$$

Derivative of Tensor Density

At the beginning, it might be intriging to consider the Lie derivative & the covariant derivative of tensor density(which doesn't transform like a tensor as usual).As a result, we shall think about how this definition can be achieved.[8]
To be specific, the initial meaning of a Lie derivative is just a object under a coordinate transformation rules, then evaluate its difference. We follow the same rules to calculate it. Some people has proven that the Lie derivative of a tensor density is still the same rank and weight tensor density!
As a simple example, we consider the case:

$$\mathscr{L}_m \ln \det(\tilde{\gamma}_{ij}) = \frac{1}{\det(\tilde{\gamma}_{ij})}\mathscr{L}_m \ln \det(\tilde{\gamma}_{ij})$$

It is surprising that the above equation changes into a weight-zeo expression(which is exactly a scalar in spacetime!)
For another one, the covariant derivative of tensor density, we need to be careful about the defintion of the derivative(the former and latter still being the tensor density)

$$^{9}\mathfrak{l}_{;i} = \mathfrak{l}_{,i} - w\Gamma_i \mathfrak{l}$$

In the end, the general governing rules:

$$\begin{aligned} \mathfrak{C}^{ij\cdots\cdots}_{kl\ldots;m} &= \mathfrak{C}^{ij\cdots}_{kl\ldots,m} + \Gamma^i_{nm}\mathfrak{C}^{nj\cdots}_{kl\ldots} + \Gamma^j_{nm}\mathfrak{C}^{in\cdots\cdots}_{kl\ldots} + \cdots \\ &\quad - \Gamma^n_{km}\mathfrak{C}^{ij\cdots}_{nl\ldots} - \Gamma^n_{lm}\mathfrak{C}^{ij\cdots}_{kn\ldots} - \cdots - w\Gamma_m \mathfrak{C}^{ij\cdots}_{kl\ldots} \end{aligned} \tag{3.34}$$

3.5.4 Summary

If we are goint to solve the original highly non-linear PDE Einstein Equation, the total indepent variables are

$$^{10}10 - 4 + (4)$$

After the conformal transformation, what we get is the freedom:

$$^{11}1 + 5 + 1 + 5 + (4) - 6$$

As a result, it will be obvious that the total indepent variables are keeping the same. But this transformation achieve the goal that we transform the intial ADM equation(Weak-Hyperbolic Equation) into a more stable PDE. The numerical simulation of the spacetime or seeking the numerical solution of the Einstein Equation become more robust in some sense!

[8] Notice that in the BSSN equations, we have defined lots of Lie derivative of tensor density
[9] We add extra terms to satisfy the requirement
[10] Dsepends on whether or not we abandon the coordinate freedom
[11] The Lapse and Shift function are bundled with the choice of spacetime coordinates

Chapter 4

Mass and Angular Momentum in GR

Just to mention a few important viewpoints about the difference about the Komar physical related quantities and ADM quantities:

- The Komar quantities comes from the integral conserved current related to killing vector;
- The ADM quantities comes from the invarience of Hamiltonian of the spacetime under certain coordinate transformation in the asymptotically infinity(If we focus on the final form of the action, which is just the infinity position integral)

When we aware of the above difference, we could understand why there will be difference between the same physical quantity, although both of them seems very physical.

Chapter 5

Black Holes

5.1 Spherical-Symmetry Metric

We first give two formal definition of "stationary" and "static":

- **Stationary** $\partial_t g_{\alpha\beta} = 0$ if there timelike Killing vector.
- **Static** $\partial_t g_{\alpha\beta} = 0$ and $g_{tk} = 0$ orthognal conditions

The spherical-symmetry metric's form behaves:

$$ds^2 = -A(r)dt^2 + B(r)dr^2 + D(r)r^2\left(d\theta^2 + \sin^2\theta d\phi^2\right) \tag{5.1}$$

Here, we list some of the properties related to Riemman tensor and Christoff symbols:

$$\begin{aligned}
\Gamma^r_{rr} = \frac{B'}{2B} \quad \Gamma^r_{tt} &= \frac{A'}{2B} \\
\Gamma^r_{\theta\theta} = -\frac{r}{B} \quad \Gamma^r_{\phi\phi} &= -\frac{r\sin^2\theta}{B} \\
\Gamma^\theta_{\theta r} = \Gamma^\phi_{\phi r} = \frac{1}{r} \quad \Gamma^t_{tr} &= \frac{A'}{2A} \\
\Gamma^\theta_{\phi\phi} = -\sin\theta\cos\theta \quad \Gamma^\phi_{\phi\theta} &= \cot\theta \\
R_{tt} = \frac{A''}{2B} - \frac{A'}{4B}\left(\frac{A'}{A} + \frac{B'}{B}\right) + \frac{A'}{rB} \\
R_{rr} = -\frac{A''}{2A} + \frac{A'}{4A}\left(\frac{A'}{A} + \frac{B'}{B}\right) + \frac{B'}{rB} \\
R_{\theta\theta} = 1 - \frac{1}{B} - \frac{r}{2B}\left(\frac{A'}{A} - \frac{B'}{B}\right)
\end{aligned} \tag{5.2}$$

5.2 Schwartzchild Metric

5.2.1 Birkhoff's theorem

This theorem tells us that the spherical-symmetrical object(mass distribution might be time-dependent), the external solution to the spacetime is:

$$ds^2 = -\left(1 - \frac{2M}{r}\right)dt^2 + \left(1 - \frac{2M}{r}\right)^{-1} dr^2 + r^2 d\Omega^2 \tag{5.3}$$

And the coordinate singularity at the point $(r = 2M)$ may not be the spacetime singularity, we need to investigate the spacetime nested properties(like curvature).
So changing into another proper coordinate system can be a good idea.

5.2.2 Null Geodesic

An important consequence of Schwartzchild metric form is that the null worldline in this kind of spacetime is exactly an null geodesic
Here, we give some useful results:

$$\Gamma^0_{00} = 0\Gamma^0_{01} = \Gamma^0_{10} = -\frac{a}{2r^2\left(1+\frac{a}{r}\right)}$$
$$\Gamma^1_{00} = -\frac{1}{2}\left(1+\frac{a}{r}\right)\frac{a}{r^2}\Gamma^1_{11} = \frac{a}{2r^2\left(1+\frac{a}{r}\right)}$$
$$\Gamma^1_{01} = \Gamma^1_{10} = 0\Gamma^0_{11} = 0\Gamma^2_{12} = \Gamma^2_{21} = \frac{1}{r}$$
$$\Gamma^1_{22} = -r\left(1+\frac{a}{r}\right)\Gamma^3_{13} = \Gamma^3_{31} = \frac{1}{r}$$
$$\Gamma^1_{33} = -\left(1+\frac{a}{r}\right)r\sin^2\theta\Gamma^3_{23} = \Gamma^3_{32} = \frac{\cos\theta}{\sin\theta}\Gamma^2_{33} = -\sin\theta\cos\theta$$

and the parameterization of geodesic equation is:

$$-1 = -\left(1+\frac{a}{r}\right)\left(\frac{dt}{d\tau}\right)^2 + \frac{1}{c^2\left(1+\frac{a}{r}\right)}\left(\frac{dr}{d\tau}\right)^2 + \frac{r^2}{c^2}\left(\frac{d\theta}{d\tau}\right)^2 + \frac{r^2}{c^2}\sin^2\theta\left(\frac{d\varphi}{d\tau}\right)^2 \tag{5.4}$$

We give the conclusion that:

- $U = const$ is a null geodesic
- $V = const$ is a null geodesic

Notice We shall not be confused with what is the worldline in spacetime, in the original coordinates (t, r, θ, ϕ), the worldline is just a form of parameterization.

5.2.3 Coordinate Transformation

In order to find a proper coordinate, we investigate the null curve along the radial direction. Recalling infinitesimal distance:

$$[1]ds^2 = 0 = -\left(1-\frac{2M}{r}\right)\mathrm{d}t^2 + \left(1-\frac{2M}{r}\right)^{-1}\mathrm{d}r^2$$

then, we integrate the partial equation to get the explicit expression:

$$r^* = \int\frac{dr}{1-2M/r} = r + 2M\ln\left|\frac{r}{2M}-1\right| \tag{5.5}$$

Notice In the above equation (5.5), we use the notation "||" to give a covaraint description of the spacetime(backward & forward of 2M).

We can find that the coordinate singularity ($r = 2M$) can be pushed to infinity, if we switch to this kind of coordinates:

$$u = t - r^*, \quad v = t + r^*$$

which reduces to $v - u = -\infty$ (infinity points on the diagram of UV coordinates).
In order to drag the infinite points to finite distance on the diagram, we execute such a transformation further:

$V = e^{v/4M}$

$U = \pm e^{-u/4M}$ the symbol depends on whether r is larger than 2M or not.

[1] dual basis vector is exactly the infinitesimal element of curve

Then, we could obtain:

$$ds^2 = -\frac{32M^3}{r}e^{-r/2M}dUdV + r^2d\Omega^2 \tag{5.6}$$

And the relationship with the original coordinates r behaves as:

$$e^{r/2M}\left(\frac{r}{2M} - 1\right) = -UV$$

which can conclude as:

$$U < 0, r > 2M$$
$$U > 0, r < 2M$$

[2] V is always large than zero.

Notice We have a one-to-one relationship between the coordinates (UV) and (rt), and the upper region of (UV) corresponds to real Schwartzchild spacetime($r > 0$ regions). And the conformal diagram behaves likes belows:

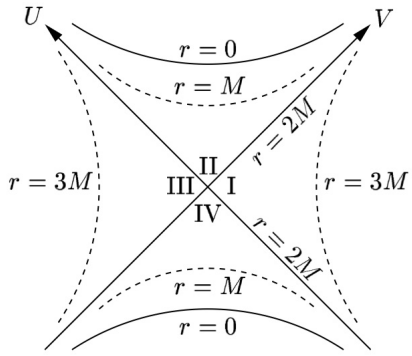


Figure 5.1: Diagram for UV coordinates

We shall be aware of the fact that the null rays in the (UV) diagram corresponds to the (UV constant) straight lines, and the relationship is:

- $v = const$ corresponds to inward radius picture
- $u = const$ corresponds to outward radius picture

[2] Obeying the standard rule

5.2.4 Penrose Diagram

we do such a coordinate transformation further:

$$\begin{aligned} \tilde{U} &= \arctan U \\ \tilde{V} &= \arctan V \end{aligned} \tag{5.7}$$

The Penrose diagram of Schwartzchild metric behaves like below:

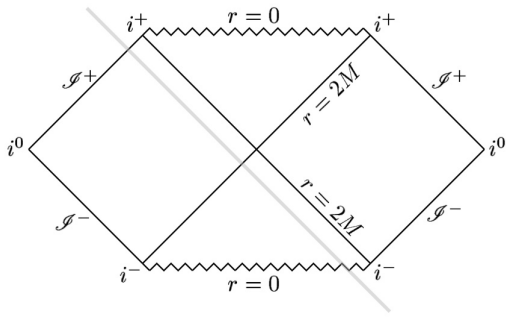


Figure 5.2: Penrose Diagram

5.2.5 Eddington-Finkelstein Coordinates

Using the one of coordinate transformation following the equation (5.2.3),we could recover to the Eddington-Finkelstein form metric:

$$\begin{aligned} ds^2 &= -(1-2m/r)dv^2 + 2dvdr + r^2 d\Omega^2 \quad \text{ingoing null geodesic} \\ ds^2 &= -(1-2m/r)du^2 - 2dudr + r^2 d\Omega^2 \quad \text{outgoing null geodesic} \end{aligned} \tag{5.8}$$

Notice The ingoing & outgoing definition seems not obvious since the parameter r behaves differently in different regions, but if we only focus on the coordinate transfomation we could understand that ingoing and outgoing ref to the diagram of tr^*.
To be more specific:

$$\begin{aligned} (dr^*/dt = -1) \quad &\text{for ingoing null geodesic} \\ (dr^*/dt = 1) \quad &\text{for outgoing null geodesic} \end{aligned}$$

5.3 Horizons

5.3.1 Event Horizon

The first question is how to define an event horizon. A more physical definition goes as follow:

Definition 1 An event horizon is a hypersurface separating those spacetime points that are connected to infinity by a timelike path from those that are not.

A more mathematical definition goes as follow:

Definition 2 The event horizon is just a null hypersface

We should be aware of the fact that the generator for a null hypersurface is actually a null geodesic if we choose proper affine parameter. And the event horizon is more understandable than apparent horizon.

5.3.2 Apparent Horizon

To give a slightly rigorous definition of apparent horizon, we give the specific procedure for finding apparent horizon and trapped surface. The procedure:

- Choose a well-defined spacelike hypersurface in the spacetime
- Find the outgoing and ingoing rays in the spacetime(Maybe need to change into another coordinates)
- Find the closed 2-surface on the hypersurface which is orthognal to the null geodesic, satisfying the conditon that the expansion of null geodesic keeps negative on the surface
- The boundary of the 2-surface is just the apparent horizon
- If we extend the slicing of the spacelike hypersface, and we can find a family of apparent horizon, the combination of them forms trapping horizon in the spacetime.

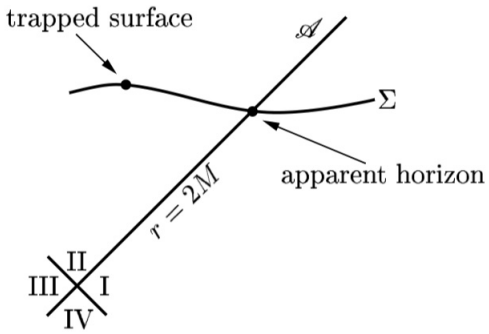


Figure 5.3: Apparent Horizon

If we analyze the above procedure of find apparent horizon, we can notice that inside the spacelike horizon, the induced metric has the form $(1, 1, 1)$, since the null geodesic always be orthogonal to the genering hypersurface, the trapped surface is actually an intersection of the two hypersurface, and we can make sure that the 2-surface's metric must be $(1, 1)$.

5.4 Reissner-Nordstrøm Metric

Reissner-Nordstrøm metric describes the circumstance of gravitional field coupled with Maxwell Equations, besides the static charge distribution satisfies the spherical symmetry conditions.

$$ds^2 = -\left(1 - \frac{2m}{r} + \frac{q^2}{r^2}\right) dt^2 + \left(1 - \frac{2m}{r} + \frac{q^2}{r^2}\right)^{-1} dr^2 + r^2 d\Omega^2 \tag{5.9}$$

Tips In order to derive the Reissner-Nordstrøm Metric, the simple method is to combine the EM tensor into the spherical-symmetry metric to give the Riemman tensor's component, then to integrate or compare different coefficients.

5.5 Kerr Metric

The Kerr Metric describing the axissymetric spacetime:

$$ds^2 = -\left(1 - \frac{2Mr}{\Sigma}\right) dt^2 - \frac{4aMr}{\Sigma} \sin^2\theta dt d\phi + \frac{\left(r^2 + a^2\right)^2 - a^2 \Delta \sin^2\theta}{\Sigma} \sin^2\theta d\phi^2 + \frac{\Sigma}{\Delta} dr^2 + \Sigma d\theta^2 \tag{5.10}$$

where,the two symbols represents:

$$\Delta = r^2 - 2Mr + a^2 \quad \text{and} \quad \Sigma = r^2 + a^2 \cos^2\theta$$

Chapter 6

Numerical Algorithm

6.1 Finite Difference

To solve the basic problem as:

$$u'(t) = -au(t) \tag{6.1}$$

There are a lot of different finite difference methods to represent the problem:

- Forward Differnece & Backward Difference
- Crank-Nicolson Difference

According to Taylo expansion:

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \frac{1}{6}f'''(a)(x-a)^3 + \cdots$$

We can prove that, comparing to Forward Difference and Backward Difference, the Crank-Nicolson Diffence is two-order accurate:

$$\begin{aligned} \frac{u^{n+1}-u^n}{t_{n+1}-t_n} &= -au^{n+\frac{1}{2}} = -a \times (u^n + u^{n+\frac{1}{2}}) \\ \frac{u^{n+1}-u^n}{t_{n+1}-t_n} &= u'(t_{n+\frac{1}{2}}) + O(\Delta t^2) \end{aligned} \tag{6.2}$$

Or commonly using the notation representation:

$$[D_t u]^n = \frac{u^{n+\frac{1}{2}} - u^{n-\frac{1}{2}}}{\Delta t} \approx \frac{d}{dt}u\left(t_n\right) \tag{6.3}$$

6.2 Methods Of Lines

MoLs is a method to make use of existing ODE numerical solutions to solve the PDE problems efficiently.We give a short introduction to MoLs in terms of the simplest example:

$$u_t + vu_x = 0$$

The basic idea of the MOL is to replace the spatial (boundary value) derivatives in the PDE with algebraic approximations. The above equation can be converted into:

$$\frac{du_i}{dt} = -v\frac{u_i - u_{i-1}}{\Delta x}, 1 \le i \le M$$

Then, we use the integration method to evolve the time-dependent numerical solutions:

$$\begin{aligned} \Rightarrow \frac{u_i^{n+1}-u_i^n}{\Delta t} &= -v\frac{u_i^n-u_{i-1}^n}{\Delta x} \\ \text{Or solving explictly:} & \\ u_i^{n+1} = u_i^n - (v\Delta t/\Delta x)\left(u_i^n - u_{i-1}^n\right), i &= 1,2,\dots M \end{aligned} \tag{6.4}$$

And the critical coefficient $(v\Delta t/\Delta x)$ is important to determine the stability of PDE, which is called the Courant-Friedricks-Lewy or CFL number, must remain below a critical value.
For a higher rank PDE, we need to rewrite the RHS, in order to give the higher derivative definitions:

$$\begin{aligned} \frac{du_i}{dt} &= -v\frac{u_{i+1}-u_{i-1}}{2\Delta x}, 1 \le i \le M \\ u_i^{n+1} &= u_i^n - \frac{v\Delta t}{2\Delta x}\left(u_{i+1}^n - u_{i-1}^n\right), i = 1,2,\dots M \end{aligned} \tag{6.5}$$

6.3 Runge-Kutta Method

This method makes use of existing ODE numerical methods to slove the PDE problems efficiently.Here we use an example to illustrate this method clearly:
Suppose we need to evolve such a diffuse equations:

$$y_x = F(x,y)$$

From the Euler forward method, we estimate the slope between neighboring points to be the value at the point of the current point(Which is initial origin of the errors). In order to be more accurate, we seek into the four-order Runge-Kutta methods(Which adapts several points inside the interval to estimate the slopes more accurately.)
Combined with Taylor series expansion, we give the formulas about the methods.

$$\begin{aligned} k_1 &= f\left(x_j, y_j\right) \\ k_2 &= f\left(x_j + h/2, y_j + hk_1/2\right) \\ k_3 &= f\left(x_j + h/2, y_j + hk_2/2\right) \\ k_4 &= f\left(x_j + h, y_j + hk_3\right) \\ y_{j+1} &= y_j + h\left(k_1 + 2k_2 + 2k_3 + k_4\right)/6 \end{aligned} \tag{6.6}$$

6.4 Programming

6.4.1 A Short Introduction to Python Package Management

The Python Package(`Package_Name`) catalog usually show as:

- `__init__.py`
- `file1.py`
- `submodule1`
  - `__init__.py`
  - `file2.py`

Usually, the file (`__init__.py`) is needed to indicate that the existence of `Package_Name` to the Python compiler. Inside the file `Package_Name`, people usually control the practical behaviour of importing, such as which modules needs to be imported and when the function defined inside a module needs to be used.
Here we give some in-use syntax for package mangement:

```
import Package_Name
# import all of the contents of the Package_Name and use the Package_Name as the begining
from Package_Name import *
# if the field name __all__ is defined(often as list), the contents of __all__ imported
from Package_Name.submodule1 import file2 # deep into the submodule
from Package_Name import function_name # can also import a defined function into the current
'''
When we import certain package or subpackge, Python Compiler would often execute the whole __init__.py
'''
```

A Sympy `__init__.py` content:[1]

```
# Top __init__.py to import printing subpackge
    from .printing import *
#-----------------------------------------------------------#
# import several functions inside the printing subpackge
    from .ccode import ccode, print_ccode
    __all__ += ['ccode', 'print_ccode']

    from .cxxcode import cxxcode
    __all__ += ['cxxcode']
```

6.4.2 Scientific Computing

Some simple Python script for plotting and calulating:[2]

```
conda info --envs(-e) #Get a list of all my environments, active environment shown with *
conda create -n env-name [python-version] package-spec # create a environment for install package
conda activate env-name # implement inside some enviroment
conda deactivate env-name # quit some enviroment
```

Numpy & Matplotlib fundamental usage:[3]

6.4.3 Matplotlib Introduction

According to the official tutorial, the best way to import the matplotlib package is obtained by:[4]

```
import numpy as np # import numpy package
import matplotlib.pyplot as plt # import matplotlib package
```

There are some concepts such as:

Axes This is what you think of as 'a plot', it is the region of the image with the data space. A given figure can contain many Axes, but a given Axes object can only be in one Figure. The Axes contains two (or three in the case of 3D) Axis objects.

Figure The whole figure. The figure keeps track of all the child Axes, a smattering of 'special' artists (titles, figure legends, etc), and the canvas,A figure can have any number of Axes.

[1] when we use `from sympy import cxxcode`, the file `__init__.py` would be throughly executed then check the content of the list `__all__`
[2] Highly recommend to use the LaTeX Package minted
[3] Highly recommend to go through the official tutorial about matplotlib & numpy
[4] Since the namespace problem of pylab

Some simple function interface:

- `subplot(nrows, ncols, index, **kwargs) subplot(pos, **kwargs)`

  the subplot will take the index position on a grid with nrows rows and ncols columns. index starts at 1 in the upper left corner and increases to the right

- `subplots(nrows=1, ncols=1,squeeze=True)`

  The returned object ax can be either a single Axes object or an array of Axes objects if more than one subplot was created.

Simple Example about Matplotlib:[5]

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0.0, 2, 0.01)
y1 = np.sin(2 * np.pi * x)
y2 = 1.2 * np.sin(4 * np.pi * x)
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, sharex=True) # can also use the array object syntax

ax1.fill_between(x, 0, y1)
ax1.set_ylabel('between y1 and 0')

ax2.fill_between(x, y1, 1)
ax2.set_ylabel('between y1 and 1')

ax3.fill_between(x, y1, y2)
ax3.set_ylabel('between y1 and y2')
ax3.set_xlabel('x')
plt.show()
```

An Introduction to Backend

Due to different usage of matplotlib in practice, matplotlib can target different outputs, and each of these capabilities is called a backend; the ”frontend” is the user facing code, i.e., the plotting code, whereas the ”backend” does all the hard work behind-the-scenes to make the figure. There are two types of backends: user interface backends (for use in pygtk, wxpython, tkinter, qt4, or macosx; also referred to as ”interactive backends”) and hardcopy backends to make image files (PNG, SVG, PDF, PS; also referred to as ”non-interactive backends”).

6.4.4 Saving the computing memory

There are some tricks to help save large memory cost during the process of computing.

- Save all the intermediate & final results into a file to save, rather than use array object to store all the intermediate results.
- Python can deal with the string type numbers such as ”2345.896” directly into float type using the syntax `float(data)`[6]

A good example:

```
    import matplotlib.pyplot as plt
    import numpy as np

```

[5] Learn to write module codes in the style of Python
[6] Python can also detect the scientific notation representation

```
    def solver_memsave(I, a, T, dt, theta, filename='sol.dat'):
    """
    Solve u'=-a*u, u(0)=I, for t in (0,T] with steps of dt. Minimum use of memory. The solution
    is stored in a file(with name filename) for later plotting.
    """
    dt = float(dt)          # avoid integer division
    Nt = int(round(T/dt))   # no of intervals
    outfile = open(filename, 'w')
    # u: time level n+1, u_1: time level n
    t = 0
    u_1 = I
    outfile.write('%.16E  %.16E\n' % (t, u_1))
    for n in range(1, Nt+1):
        u = (1 - (1-theta)*a*dt)/(1 + theta*dt*a)*u_1
        u_1 = u
        t += dt
        outfile.write('%.16E  %.16E\n' % (t, u))
    outfile.close()
    return u, t

    def read_file(filename='sol.dat'):
        infile = open(filename, 'r')
        u = [];  t = []
        for line in infile:
            words = line.split() # effectively deal with two columns database
            if len(words) != 2:
                print 'Found more than two numbers on a line!', words
                sys.exit(1)  # abort good coding habits!
            t.append(float(words[0]))
            u.append(float(words[1]))
        return np.array(t), np.array(u)

    def read_file_numpy(filename='sol.dat'):
        data = np.loadtxt(filename)
        t = data[:,0]
        u = data[:,1]
        return t, u
```

Managing the System Memory

Contrary to the Python's memory manangement, the dynamic memory allocation in C++ is more efficient.

```
    // This is a simple example to show how the array's memory manangement is dealt with inside C++
    int rows = 5, cols = 3;
    double** A; // A is a pointer pointing to the continuous address of pointers
    A = new double* [rows]; // allocate the pointer's memory
    for (int i = 0; i < rows; i++)
    {
        A[i] = new double [cols]; // allocate the pointer's object memory
    }
    for (int i = 0; i < rows; i++) // delete the object' memory
    {
        delete[] A[i]; // easily forgotten one!
    }
    delete[] A;
```

There are some points needs to be focused on:

- Be sure that every pointer has a explicit reference object (`assert (p_x != NULL)`)
- recycle every used pointer to keep from the memory overflow in large scale computing.

The above example can be reorgnize as a module function as follows.

```
double** AllocateMatrixMemory(int numRows, int numCols)
// allocate the memory as a module function
{
    double** matrix = new double* [numRows];
    for (int i = 0; i < numRows; i++)
    {
        matrix[i] = new double [numCols];
    }
    return matrix;
}
void FreeMatrixMemory(int numRows, double** matrix)
// recycle the memory as module function
{
    for(int i = 0; i < numRows; i++)
    {
        delete[] matrix[i];
    }
    delete[] matrix;
}
```

Here, we supplement some C++ operator overloading example for a later review:

```
std::ostream& operator<< (std::ostream& output, const ComplexNumber& z);
// overload the opertor << for printing the class object;
ComplexNumber ComplexNumber::operator- (const ComplexNumber& z) const;
//overload the operator for the difference of the class object;
/* The operator overloading needs to satisfy the original definition, for class member function,
it is worthy to concentrate on the main owner of the operator.
*/
```

Some practical coding standard to follow:

- Variables are declared close to where they are used, rather than at the beginning of a function;
- Locally declared variable names have underscores, such as (`total_sum`);
- Where types are pointers or references the " * "or " & " character is written adjacent to the native type, with no space between,such as (`int* p_number`);
- Pointer names begin with "p", with an exception of dynamic memory allocation cases;
- Function names are in camel-case, which also applies in the class-member methods;
- Names of arguments to functions (and class methods) are in also camel-case, but they begin in lower-case;
- Class data which have access controls are also in camel-case with "m" (for "my") to denote "`private`" or "`protected`" and "mp" for public;
- Class names are also in camel-case;

6.4.5 An Introduction to Numpy

Numpy as a Python package which provide various and efficient array disposal.The existence of Numpy make it possible for Python to get an approximate speed like C++.
There are some fundamental Numpy concepts:

- The basis object in Numpy is array `np.array{}`
- The array object is generally adapted in scientific computing and data visualization.

```
import numpy as np
c = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]]) # This is an array object with (2,2,3)
b = np.array([[[1], [2]], [[3], [4]]]) # This is an array object with (2,2,1)
# from the most outer bracket to the most inner bracket to compute the dimensions
```

Some commonly used array-creating methods:

```
b = np.arange(1, 9, 2) # beginning point to open ending point with the step
c = np.linspace(0, 1, 6)   # beginning point to open ending point with data numbers
# quick create the target-dimension array object
a = np.ones((3, 3))
b = np.zeros((2, 2))
# everytime we create the array object, we can always denote the desired data type as possible
c = np.array([1, 2, 3], dtype=float)
# the array object itself is a mutable object
```

Small Tips For Seeking Help

```
%psearch # magic methods: match possible function object name inside the IPython Package.
numpy.lookfor() # can look for numpy inside API documentation for proper infomation.
```

Chapter 7

Parallel Algorithm

Chapter 8

Code Generation Tools

In this chapter, we shall discuss how to transform tensor-form algebra equations into hignly accurate programming codes using the language of Python.
Besides,there are two relative mature tools can be referenced to generate codes in numerical relativity community:

- **Kranc :** An component of Einstein Toolkit, which is tightly linked with Mathematica to generate highly parallel C and C++ codes.
- **SENR/NRPy+ :** A totally Python tools to generate C codes in suitable coordinates.

Why and How Code generation refers to the act of converting a symbolic expression into equivalent code in some language, typically for numeric evaluation. This allows one to use programming tools to symbolically model a problem and generate fast numerical code for specific platforms that executes that model. This is a powerful tool that is useful to scientists in many domains. Code generation allows users to speed up existing code, to deal only with the high level mathematics of a problem, avoids mathematical errors and typos, makes it possible to deal with expressions that would otherwise be too large to write by hand, and opens possibilities to perform automatic mathematical optimizations of expressions.

8.1 Short Introduction to Sympy

[1]

8.1.1 Important Concepts In Sympy Library

Symbolic Computation

Symbolic computation deals with the computation of mathematical objects symbolically. This means that the mathematical objects are represented exactly, not approximately, and mathematical expressions with unevaluated variables are left in symbolic form.
Like Mathematica, symbolic computation can give us an accurate results based on the forms of input.

A Python Library

Sympy doesn't expand the language of Python, which means that everything you master fits in Sympy, which reflects:

- Every variable for later use must have a definition.
- Original Python operator and expression syntax are the same in the framework of Sympy.
- Object in Sympy is Expression(like Mathematica) is immutable, which means the function changing the expression indeed return a new expression object

[1] A detailed introduction, see the official documentation

```
from sympy import * # a better choice might be     import sympy as codes
x,y,z = symbols('x y z') # symbols('x,y,z') is also ok
# Expression Substitution
expr = x + 1
expr.subs(x, 2)  # The substitution must be defined object or Python nested object
expr = cos(2*x)
expr.evalf(subs={x: 2.4}) # the subs flag, which takes a dictionary of Symbol: point pairs.
```

Simple Manipulation When we needs to evaluate some expression on the grid points, substitution of thousands points is unrealistic. The easiest way to convert a SymPy expression to an expression that can be numerically evaluated is to use the `lambdify`[2]

```
import numpy as np
a = numpy.arange(10)
expr = sin(x)
f = lambdify(x, expr, "numpy")
f(a)
```

In the above example, what the function `lambdify` does is converting the SymPy names to the names of the given numerical library. It is also possible to reuse our defined functions just like `f = lambdify(x, expr, {"sin":mysin})`[3]

[2] Its behavior is similar to Python nested `lamda` function.
[3] function `mysin` is defined, the input parameter is a dictionary.

Chapter 9

Python Programming

Chapter 10

C++ Programming

This chapter collects some useful knowledge when coming across different but first-looking difficult concepts.

10.1 Using Template in C++

As an important part of generic programming, the wide use of template greatly expand the influence of C++ itself.The compiler will generate type-specific code after the process of compiling.

10.1.1 Template Class

The fundamental usage of template class as:

```
// declaration of template class
template <typename T> class ClassA;
// definition of template class
template <typename T> class ClassA
{
        T member;
};
```

Practical Usage of Template Class

```
template <typename T>
class vector
{
public:
        void push_back(T const&);
        void clear();

private:
        T* elements;
};

// Instance of template class
vector<int> intArray;
vector<float> floatArray;

// Using the instance of template class
intArray.push_back(5);
floatArray.push_back(3.0f);
```

Notice 1 The parameter inside the template shall be type(fundamental datatype, struct, class type), which can also include integer type. Besides, the keywords `typename` shall be used after c++11 standard, the keyword `class` is more common in the past.

Notice 2 When we compile multiple files which might contain template declarations and definitions, we must be careful about how to include these files in the `main.cpp` files properly.Just to consider the whole things that the compiler needs to do when we just input the command `g++ -O -o exe .cpp .cpp`. The compiler needs to preprocess, compile, link and finally organize them together to a executable file. Thus the template declarations and definitions must be put together to be picked by the compiler during the period of linkage.

10.1.2 Some STL Concepts

`vector` object(obtainer) as one part of STL, it gives us a common interface to deal with data more efficiently. As the name **STL** shows the `vector` obtainer is still a template class when considering its achievement.
Here, we remind some concepts about operator:

- `a::b` is only used if b is a member of the class (or namespace) a. That is, in this case a will always be the name of a class (or namespace);
- `a.b` is only used if b is a member of the object (or reference to an object) a. So for a.b, a will always be an actual object (or a reference to an object) of a class;
- `a->b` if a is a pointer to the class instance and b is a member of class;

And the special meaning of the synatax `std::vector<int>::iterator it = myvector.begin();` implies that iterator is a typedef object inside the class. Like the rule of `namespace`, the class'specific type iterator can only be used inside the class, so we use the scope resolution operator `::`