QuasoLocalMeasures

Erik Schnetter <schnetter@cct.lsu.edu>

2010-04-02

Abstract

Calculate quasi-local measures such as masses, momenta, or angular momenta and related quantities on closed two-dimentional surfaces, including on horizons.

1 A note on evaluating 3D integrals on the horizon world tube

[NOTE: Ignore the stuff below. You can do that much easier.]

1.1 Integral transformation

The papers about dynamical horizons contain integrals over the 3D horizon world tube, expressed e.g. as

$$\int X \, d^3V \tag{1}$$

where X is some quantity that lives on the horizon. These integrals have to be transformed into a $2+1$ form so that they can be conveniently evaluated, e.g. as

$$\int X \; A \, d^2S \, dt \tag{2}$$

where d^2S is the area element on the horizon cross section contained in Σ, and dt is the coordinate time differential. The factor A should contain the extra terms due to this coordinate transformation.

Starting from the 3-volume element d^3V, let us first decompose it into the 2-volume element d^2S and a "time" coordinate on the horizon, which we call σ. Note that σ will generally be a spacelike coordinate for dynamical horizons. Let $\mathbf{Q}$ be the induced 3-metric on the horizon, and $\mathbf{q}$ be the induced 2-metric on the cross section. Then it is

$$\begin{aligned} d^3V &= \sqrt{\det Q} \; d\theta \, d\phi \, d\sigma && (3) \\ &= \frac{\sqrt{\det Q}}{\sqrt{\det q}} \; d^2S \, d\sigma && (4) \end{aligned}$$

because $d^2S = \sqrt{\det q} \, d\theta \, d\phi$.

The coordinate time differential dt and the differential $d\sigma$ will in general not be aligned because the horizon world tube will in general not have a static coordinate shape. It is

$$
\begin{aligned}
d\tau &= (\cosh\alpha)\, dt + (\sinh\alpha)\, ds && (5)\\
d\sigma &= (\cosh\alpha)\, ds + (\sinh\alpha)\, dt && (6)
\end{aligned}
$$

where s is a radial coordinate perpendicular to the horizon and also perpendicular to t, and τ is perpendicular to σ and lies in the plan spanned by t and s. τ and σ are depend on t and s via a Lorentz boost. Thus we have

$$
\begin{aligned}
\frac{d\sigma}{dt} &= (\cosh\alpha)\,\frac{ds}{dt} + (\sinh\alpha)\,\frac{dt}{dt} && (7)\\
&= \sinh\alpha \quad . && (8)
\end{aligned}
$$

Putting everything together we arrive at

$$
\int X\, \frac{\sqrt{\det Q}}{\sqrt{\det q}}\, (\sinh\alpha)\, d^2S\, dt \quad . \tag{9}
$$

1.2 The "lapse" function N_R

Starting from

$$
N_R = |\partial R| \tag{10}
$$

we find, since the radius R changes only in the σ direction,

$$
N_R^2 = g^{\sigma\sigma}\,(\partial_\sigma R)\,(\partial_\sigma R) \quad . \tag{11}
$$

If we assume $\partial_\tau R = 0$ and write $\partial_t R = \dot{R}$, and use the relations between σ and t from above, we get

$$
\begin{aligned}
\dot{R} &= \partial_t R && (12)\\
&= \frac{\partial\tau}{\partial t}\partial_\tau R + \frac{\partial\sigma}{\partial t}\partial_\sigma R && (13)\\
&= \sinh\alpha\, \partial_\sigma R && (14)
\end{aligned}
$$

[NOTE: but $\partial_t\alpha \neq 0$.] and therefore

$$
\partial_\sigma R = \frac{1}{\sinh\alpha}\,\dot{R} \quad . \tag{15}
$$

Additionally we have $g^{\sigma\sigma} = g^{ab}\sigma_a\sigma_b = g_{ab}\sigma^a\sigma^b$ where σ^a is the unit vector in the σ direction, i.e.

$$
\begin{aligned}
\tau^a &= (\cosh\alpha)\, t^a + (\sinh\alpha)\, s^a && (16)\\
\sigma^a &= (\cosh\alpha)\, s^a + (\sinh\alpha)\, t^a && (17)
\end{aligned}
$$

1.3 Special Behaviour

In order to use the IsolatedHorizon thorn on existing data (postprocessing), the following procedure is necessary.

- Computing time-independent quantities.

  The 3-metric and the extrinsic curvature must be available in HDF5 files.

  - Set up a parameter file for a grid structure that contains the region around the horizon. The refinement level structure and grid spacing etc. needs to be the same as in the HDF5 files, but the grids can be much smaller. You can also leave out some finer grids, i.e., reduce the number of levels. However, the coarse grid spacing must remain the same. The symmetries must also be the same.
  - Use the file reader and thorn AEIThorns/IDFileADM to read in the ADM variables from the files. The parameter file does not need to activate BSSN_MoL or any time evolution mechanism. IDFileADM acts as provider for initial data, so you don't need any other initial data either.
  - Set up your parameter file so that the AH finder runs, stores the horizon shape in SphericalSurface, and IsolatedHorizon accesses these data.

  This gives you the time-independent variables on the horizon, i.e., mostly the spin. It also allows you to look for apparent horizons if you don't know where they are.

- Computing time-independent quantities, e.g. 3-determinant of the horizon

  - Performing some time steps is necessary. Either read in lapse and shift from files, or set them arbitrarily (e.g. lapse one, shift zero).
  - Activate a time evolution thorn, i.e., BSSN_MoL, MoL, Time, etc. In order to fill the past time levels, just choose MoL::initial_data_is_crap. If you have hydrodynamics, read in the hydro variables as well.
  - Only two time steps are required. Remember that the output of IsolatedHorizon for iteration 0 and 1 are incorrect or very inaccurate, since the past time levels are not correct, and hence the time derivatives that IsolatedHorizon calculates are wrong. However, iteration 2 should be good. (One could also perform 5 iterations and cross-check.)

- If data for the extrinsic curvature is not available, but those for the 3-metric, lapse, and shift for consecutive time steps are (that is, if you have data suitable for finding event horizons), then one needs to reconstruct the extrinsic curvature first. There is a thorn AEIThorns/CalcK that helps with that. It reads the data for the 4-metric timestep after timestep, calculates the time derivative of the 3-metric through finite differencing in time, and then determines the extrinsic curvature from that, and writes it to a file. Once you have it, you can go on as above. CalcK has a small shell script that tells you what to do.

- In general, things become more interesting if a static conformal factor is involved (since more variables are present), especially if it is output only once (since it is static), which means that one has to mix variables from different time steps.

The thorns involved in this procedure have some examples. In general, this is NOT a "just do it" action; you have to know what you are doing, since you have to put the pieces together in your parameter file and make sure that everything is consistent. We may have a vision that you just call a script in a directory that contains output files and the script figures out everything else, but we're not there yet. All the ingredients are there, but you'll have to put them together in the right way. Think Lego.

2 Interpreting 2D output

2D output is given on a rectangular grid. This grid has coordinates which are regular and have a constant spacing in the θ and ϕ directions. Cactus output has only grid point indices, but does not contain the coordinates θ and ϕ themselves.

In gnuplot, one can define functions to convert indices to coordinates:

$$\theta(i) = (i - g\theta + 0.5) * \pi / n\theta \tag{18}$$

$$\phi(j) = (j - g\phi) * 2 * \pi / n\phi \tag{19}$$

where $g\theta$ and $g\phi$ is the number of ghost points in the corresponding direction, and $n\theta$ and $n\phi$ the number of interior points. Here are the same equations in gnuplot syntax:

```
theta(i) = (i - nghosts + 0.5) * pi / ntheta
phi(j) = (j - nghosts) * 2*pi / nphi
```

Usually, `nghosts=2`, `ntheta=35`, and `nphi=72`. `i` and `j` are is the integer grid point indices. Note that `ntheta` and `nphi` in the parameter file include ghost zones, while their definitions here do not include them. In general, `nphi` is even and `ntheta` is odd, because the points are staggered about the poles.

A test plot shows whether the plot is symmetric about $\pi/2$ in the θ and π in the ϕ direction. Also, plotting something axisymmetric with bitant symmetry vs. θ and vs. $\pi - \theta$, and vs. ϕ and $2\pi - \phi$, should lie exactly on top of each other.

There are also scalars `origin/delta_theta/phi` which one can use in the above equations. Then the equations read

```
theta(i) = (i + origin_theta) * delta_theta
phi(j) = (j + origin_phi) * delta_phi
```

but, of course, these four quantities are all irrational and don't look nice.

References