Bootcamp Tutorial

Bootstat 2021 29th May 2021

Marten Reehorst

Polynomial Matrix Problem

- In numerics we usually discretize the space of functionals by taking derivatives around the point $z = zb = 1/2$:

$$\sum_{mn} \alpha_{mn} F^{mn}_{\Delta,L} \geq 0\,.$$

$$\text{where } F^{mn}_{\Delta,L} = \partial_z^m \partial_{\bar{z}}^n F^{\Delta_\sigma}_{\Delta,L}(z,\bar{z})\big|_{z=\bar{z}=1/2}$$

$$\text{and } F^{\Delta_\sigma}_{\Delta,L}(z,\bar{z}) = ((1-z)(1-\bar{z}))^{\Delta_\sigma} g_{\Delta,L}(z,\bar{z}) - (z\bar{z})^{\Delta_\sigma} g_{\Delta,L}(1-z,1-\bar{z}).$$

- These derivatives around $z = zb = 1/2$ can be approximated (to arbitrary precision) by a positive function times a polynomial:

$$\partial_z^m \partial_{\bar{z}}^n F_{\Delta,L}(1/2,1/2) \approx \chi_L(\Delta) P^{mn}_L(\Delta).$$

Polynomial Matrix Problem

- These derivatives around $z = zb = 1/2$ can be approximated (to arbitrary precision) by a positive function times a polynomial:

$$\partial_z^m \partial_{\bar{z}}^n F_{\Delta,L}(1/2, 1/2) \approx \chi_L(\Delta) P_L^{mn}(\Delta).$$

- Here $P_L^{mn}(\Delta)$ is a polynomial in Δ and $\chi_L(\Delta) \geq 0$ $\forall\, \Delta \geq \Delta_{\text{unitarity}}$.
- We can simply divide by the strictly positive factor $\chi_L(\Delta)$ and this will leave our search for a positive functional unchanged.
- We will keep track of $\chi_L(\Delta)$ only to improve the stability of the numerics since it can be helpful to set some appropriate scales.

Polynomial Matrix Problem

- Thus, numerical bootstrap problems can be posed as a Polynomial Matrix Problems (PMP's)

$$\begin{aligned} &\text{maximize} \quad \vec{\alpha}\cdot\vec{b} \quad \text{over} \quad \vec{\alpha}\in\mathbb{R}^N, \\ &\text{such that} \quad \vec{\alpha}\cdot\vec{M}_j(\Delta)\succeq 0 \quad \text{for all } \Delta\geq 0 \text{ and } 1\leq j\leq J \\ &\qquad\qquad \vec{\alpha}\cdot\vec{n}=1. \end{aligned}$$

- Here $\vec{M}_j(\Delta)$ are vectors of polynomials where the m-th entry corresponds to the m-th non-zero derivative of $F_{\Delta,j}$ and j labels the different families of operators that we demand positivity on. So in the simplest single correlator bootstrap j would run over the even spins: $L=0,2,4...,L_{\text{max}}$. E.g.

$$\vec{M}_0=(P_{L=0}^{(0,1)(\Delta)},...,P_{L=0}^{(n,m)(\Delta)})$$

- The functional $\vec{\alpha}$ is normalized on a vector $\vec{n}$ and the vector $\vec{b}$ can be chosen to maximize the action of the functional on for example a specific operator.

Polynomial Matrix Problem

- The identity vector is often chosen as the normalization vector since it is known to be exchanged a-priori. This automatically imposes strict positivity on one term.
- For the objective vector $\vec{b} = \vec{0}$ is often chosen if one is only interested in the feasibility of a certain spectrum assumption.
- However, by choosing $\vec{n}$ and $\vec{b}$ appropriately we can also compute bounds on OPE coefficients squared or obtain additional information (Navigator bootstrap).

OPE Bound

- First we write the crossing equations with some terms separated out

$$\lambda_{\mathbb{I}}^2 F_{\mathbb{I}} + \lambda_{\mathcal{O}_m}^2 F_{\mathcal{O}_m} + \sum_{\mathcal{O}_r} \lambda_{\mathcal{O}_r}^2 F_{\mathcal{O}_r} = 0$$

  where the sum over $\mathcal{O}_r$ is over all other operators.

- We can set $\lambda_{\mathbb{I}}^2 = 1$ by normalizing the two point functions of the involved operators to 1. We then act with α and demand positivity on all $F_{\mathcal{O}_r}$ and subtract the strictly positive infinite sum. This gives the inequality

$$\lambda_{\mathcal{O}_m}^2 \alpha(F_{\mathcal{O}_m}) \leq -\alpha(F_{\mathbb{I}})$$

  So if we normalize $\alpha(F_{\mathcal{O}_m}) = 1$ we find an upper bound on $\lambda_{\mathcal{O}_m}^2$. The strictest upper bound is found by maximizing $\alpha(F_{\mathbb{I}})$.

OPE Bound

- A lower bound can be found in a similar way (exercise). In this case one has to be careful with allowing another operator in the spectrum that is continuously connected to the operator that you are bounding.

Navigator function

- In general it is much better to replace the zero-objective $\vec{0}$ by a meaningful one that tells you how far you are from the boundary of between the dis-allowed and allowed region.
- The navigator function, presented last week by Ning Su, is a way to get such a quantitative measure. It consists of maximizing the identity for a smartly chosen normalization vector. The normalization vector must be chosen such that $\vec{\alpha} \cdot \vec{b}$ is bounded for all points. (It is pointless to have a navigator function that steps from $+\infty$ to finite negative values at the boundary of the allowed region.)
- This weeks tutorial also contains an example of how to compute such a navigator function.

PMP as special case of SDP

- The search for an α_{mn} obeying the conditions above can be phrased as a semi-definite program (SDP).
- The same holds if we replace positivity of a polynomial $P(x)$ with positive semi-definiteness of a symmetric matrix of polynomials. This kind of constraints occur when you consider the crossing equation for multiple correlators.
- SDP's can be solved efficiently for example using interior-point methods.
- Bootstrap problems turn out to require very high precision. A dedicated arbitrary precision solver called `SDPB` was created and is actively being maintained.
- In practice, it is easiest and usually sufficient to think of a bootstrap questions on the level of the PMP.

PMP as special case of SDP (more details)

- Theorem due to Hilbert

$$p(x) \geq 0 \, \forall \, x \geq 0 \iff p(x) = f(x) + xg(x)$$

where $f(x)$ and $g(x)$ are sums of squares of polynomials. Let $[x]_d$ denote the vector with entries $(1, x, \ldots, x^d)$. If $f(x)$ is a sum of squares of polynomials with coefficients $\boldsymbol{c}_i = (c_{i0}, \ldots, c_{id})$, then

$$\begin{aligned} f(x) = \sum_i (\boldsymbol{c}_i^T [x]_d)^2 &= [x]_d^T \left(\sum_i \boldsymbol{c}_i \boldsymbol{c}_i^T \right) [x]_d \\ &= [x]_d^T A [x]_d, \text{ with } A \equiv \sum_i \boldsymbol{c}_i \boldsymbol{c}_i^T \succeq 0. \end{aligned}$$

PMP as special case of SDP (more details)

- So that

$$p(x) \geq 0 \, \forall \, x \geq 0 \iff$$
$$p(x) = [x]_d^T A [x]_d + x([x]_{d'}^T B [x]_{d'}),$$
$$\text{with} \qquad A, B \succeq 0,$$

SDPB

- Available cross platform through something called Docker. (For Windows and iOS use 3.2.2 and not the newest version because there is a bug in the newest one. For Ubuntu I did not encounter the bug in the newest version but perhaps using an older version is still safest.)
- The newest version of SDPB takes as input a directory containing in JSON the parameters describing the SDP.
- In order to write this input one first has to formulate the correct PMP and then convert it.
- The historic work flow is to first compute the PMP in for example Mathematica and write it to an .xml file. To do the latter there is a Mathematica notebook `SDPB.m` available with SDPB.

SDPB

For simplicity you can view all of this as a black box and all you need to write is:

- The objective $\vec{b}$, the normalization vector $\vec{n}$ and the list of vectors of polynomials $\vec{M}_j$ for which we demand

$$\alpha \cdot M_j \succeq 0 \quad \forall\, x \geq 0$$

- Since SDPB will demand positivity for all $x \geq 0$ you will have to shift x appropriately to correspond to the bound you want to impose, for example such that $x = 0$ corresponds to the unitarity bound.
- Sometimes you might have to impose positivity on a specific disconnected operator. This happens for example when you demand that there is only one relevant scalar. In this case you simply replace x with the appropriate fixed value and add it to the list $\vec{M}_j$.

Setting up the PMP

Constructing $\vec{M}_j$ involves a few steps.

- First one has to (efficiently) compute the appropriate derivatives of the conformal blocks appearing in the crossing function F. One standard tool to do this is `scalar_blocks`. [1]
- Next you have to put these derivatives together correctly to form the vector $\vec{M}_j$.
- Finally, you can give this to the `SDPB` through some interface as found for example in `SDPB.m`.

[1] There is an issue with the rational approximation made in `scalar_blocks`. It does not make the optimal minimal choice for the separation into the denominator $\chi(x)$ and the numerator $P(x)$. Using it still gives the correct results but is numerically not optimal for convergence properties. An alternative code to compute the conformal block approximations is included in SimpleBoot by Ning Su. This is a great tool for setting up PMP's but is not the simplest to get started.

Exercise set 2

- Unfortunately part of getting started with numerical bootstrap is just a matter of managing to install the appropriate tools. So the first exercise will be to actually manage to install `SDPB.m` (`scalar_blocks` will come with it).
- The provided notebook shows a minimal workflow for writing and solving the PMP corresponding to the question of the maximal allowed gap Δ_ϵ in the single correlator bootstrap.
- You can then use this to find the Ising kink in $d = 3$
- You can also try whether there are any assumptions on the spectrum that you can make that will allow you to isolate the Ising model, i.e. give you a closed island.

What's next

- Once you are able to run and understand the provided notebook you are also ready to find your own bounds.
- Bounds on different sectors, $L = 2, 4, \ldots$, different assumptions on isolated operators...
- Other spacetime dimensions (`scalar_blocks` works in any dimension).
- Multi-correlator bounds, e.g. $\langle\sigma\sigma\sigma\sigma\rangle$, $\langle\sigma\sigma\epsilon\epsilon\rangle$, and $\langle\epsilon\epsilon\epsilon\epsilon\rangle$.
- Global symmetries $O(N)$, $SU(N)$,...
- The main principle is finding which polynomials you want to demand positivity (or semi-positive definiteness) on and for which values of x.

Navigator (more details)

- We relax the crossing equations such that they always have a solution for some value of λ

$$F_{0,0}(u,v)+\lambda M(u,v)+\sum_{(\Delta,\ell)\in S(\Delta_*)} p_{\Delta,\ell}F_{\Delta,\ell}(u,v)=0, \quad p_{\Delta,\ell}\geq 0$$

- This can be done for example by taking $M(u,v)$ to be the Generalized Free Field (GFF) solution.
- In practice you only have to add the GFF-operators that are not already allowed by your spectrum assumptions $S(\Delta_*)$.
- When maximizing the bound on Δ_ϵ in the single correlator we have been studying the only GFF operator that is excluded is σ^2 with dimension $2\Delta_\sigma$.

Navigator (more details)

- By taking $M_{\mathrm{GFF}}(u,v) = F_{2\Delta_\sigma,0}(u,v)$ the crossing equation always has a solution.
- We can minimize the contribution from the GFF solutions to crossing by minimizing λ.
- For points disallowed by crossing a positive λ is required to satisfy the 'relaxed' crossing constraint.
- At the boundary of the allowed and disallowed region a solution without the λ contribution, i.e. $\lambda = 0$ has to exist.
- By choosing the normalization vector n and the objective vector b appropriately we can minimize λ using SDPB.

Navigator (more details)

- By choosing the normalization vector n and the objective vector b appropriately we can minimize λ using SDPB.

$$\max \alpha(F_{0,0}) \text{ over all linear functionals } \alpha \text{ such that}$$
$$\alpha(M) = -1$$
$$\alpha(F_{\Delta,\ell}) \geq 0 \text{ for all } (\Delta,\ell) \in S(\Delta_*)$$

Aside: Additional tools

Many other useful and interesting tools have been developed over the past years. Some examples of interesting ones are:

- Autoboot (by Mocho Go), this can automatically give you the crossing equations appropriately decomposed by irrep for many groups and representations.
- SimpleBoot (by Ning Su). This is an interface that can set up any PMP's arising in the conformal bootstrap given the crossing equations in the format outputted by Autoboot.
- For computations involving spinning external operators the set up is much more complicated. Some of this has recently been simplified for $d = 3$ by the availability of `blocks_3d` to automatically compute spinning blocks.