Description of Analysis of Learned Controllers (Neural Nets) Currently Used in Group as of Day7 Month6 Year2019: The Possibilities for Abstractions, Descriptions and Possible Proofs of Behavior

Document UUID: bcfe849d-4394-435d-8cb9-423cf0c1e497

David Bayani [redacted]

last modified: day12 month6 year2019

1 Intended Receiptiants

David Held [redacted]
Stefan Mitsch [redacted]
André Platzer [redacted]
John Dolan [redacted]
The author only provides permission for the above two listed people to view and read this document.

2 Related Material Previously Shared by the Author

This document builds on the document with uuid d0b509df-70a7-4356-8b68-7b083d8b5208, sent to David Held and Stefan Mitsch around day31 month5 year2019. This document incorporates description of work done since that time, as well as analysis of some selected results and description of intended future direction.

The content in the document was overviewed in the PI group meeting on day7 month5 year2019. Slides from that meeting were later sent to David Held and Stefan Mitsch [1] . Immediately after that meeting, further discussion was held between the author, Jonathan Laurent [redacted], and Qin Lin [redacted], where the author more thoroughly described the process done thus far, what it provides, the intended immediate next-steps, and the slightly longer term goals, all described in this document.

To the best of my knowledge base on discussion with members of the Assured Autonomy Group (AA Group), I am the first in the group to apply the use of abstract interpretation domains (AIDs) to the learned controllers, particularly for the tasks outlined in section 3.1. Further, to the best of my knowledge, I am the first in the group to propose doing this- I refer the reader to emails over the passed year between the author, Dr. Held, and Dr. Mitsch discussing use of AIDs for exactly this. Emails between the author and Dr. Platzer prior to the existence of the AA Project show that this has been the author's intent for quite some time.

[1] Documents "presentation_day7_month6_year2019.odp" and "presentation_day7_month6_year2019.pdf" sent in email at minute18 hour12 day7 month6 year2019.

3 Overview

On my own accord, I have used abstract interpretation domains (AIDs) Cousot and Cousot (1977) and CEGAR Clarke et al. (2003) to perform various formal analysis of trained policies that Edward Ahn had provided [2] [3] , and produced numerous visualization and other materials digestible by a human to better understand the behavior and properties of the learned controllers. In short, the methods in discussion allow us to shove input-sets into a neural net and examine the output-sets, where the process of selecting sets to study can be task-dependent or task-agnostic; this allows us to perform sound, formal study of input-out relations over sets with infinitely many elements, in contrast to instance-sampling, which provides only limited intuition and no general guarantees. This method of analysis is particular amenable to the examination of our feed-forward neural nets due to their fixed, continuous arithmetic, and is not directly hindered by the transcendental activation functions present nor by the numerous multi-variate relations involved. Detailed in this document is a framework and general approach that has the very real potential of covering areas of weakness in the AA Group's other formal approaches (dL Platzer (2008), dReal Gao et al. (2013), etc.) while also providing a valuable engineering and reasoning tool to understand, explain, and refine learned controllers. It is the opinion of the author that perhaps the most interesting use of AIDs and CEGAR is not as a push-button technique to produce proofs absent human intervention, but as a tool that an expert can "collaborate" with, where the tool enlightens the human to the controller behavior, and the human provides insight to the algorithm about where is it necessary to refine abstractions in order to complete proofs. There are also interesting possibilities for improving learning by over-sampling areas of the input space during training where intermediate-controllers violate desired properties.

NOTE: at the time of writing this, the author is intentionally only suggesting use of AIDs for handling and understanding the learned controls in use. While AIDs have long been used for environment dynamics as well to varying degrees of success Clarke et al. (2003), the author is under the impression that the AA Group's existing tools (particularly dL Platzer (2008)) have shown much better handling of such components in proofs. Generally, the utility of AIDs in that arena seems minimal compared to tackling the learned controllers, in part because the AA Group claims to not yet have a good handle of the latter.

3.1 Overview of Intended Applications in the Assure Autonomy Group

There are at least three major applications for the use of abstractions of learned controllers in the endeavors of the Assured Autonomy Group:

- **Generate New Models for Down-Stream Tasks:** the abstraction of the learned controller overviewed here removes transcendental arithmetic and tangled variable relationships that occur in connected layers of a neural net, replacing them with a series of conditionals describing

[2] via Google Doc link, [redacted] shared with me on day23 month4 year2019.

[3] Later in this document, I switch from referring to "I" and "me" to "we" per typical academic stylings; the "we" that will be used refers only to me, the author.

how input sets relate to output sets. Substituting these abstractions for the original controller in the controller-plant models has the potential to greatly ease later proof and analysis. This can be done in both task-agnostic and task-informed fashions, with the latter being more efficient in terms of the number of conjuncts in the resulting abstract model.

- **Prove Properties of the Learned Controller:** As overviewed when discussing Counter Example Guided Abstraction Refinement (CEGAR), this approach leveraging AIDs provides a method of proving certain statements hold on the learned controller. This differs from the prior point in that CEGAR - given a statement to prove or find counter examples to- automatically produces a refined abstraction that has appropriate, locally-varying levels of granularity. That is, when generating a new model (as in the previous bullet), there is not a single prescribed way of joining input-sets unless multiple such input sets produce exactly the same output set; unless one provides additional criteria for increasing the granularity of the input or output abstractions, it is not clear how to vary abstraction granularity in newly generated models in task-agnostic manners. The result of having unnecessary additional divisions in our abstract model is a reduction in the transparency and tractability that the abstraction process set out to provide in the first place; CEGAR helps produce task-dependent abstractions that attempt to be maximally transparent and tractable subject to the task requirements.

- **Tool for Human Understanding: Debugging Controllers and Improve the Human Understanding of Problems**: The use of AIDs can greatly assist the engineering and experimentation process by producing human-digestible descriptions of, or insights into, a neural net's behavior. There is particular potential for benefit in our current setting where we are dealing in spaces where each axis is independently meaningful (such as velocity or position as opposed to individual pixel values in an image) and the dimension of the space is higher than directly visualizable for a human, though low enough to be mentally understood analytically [4]. These materials can be of at least the following two varieties:

  1. **Task-agnostic:** As detailed in sections 5.1.1 and 5.1.2 , we produce plots that display general properties and behaviors of the learned controllers, independent of any particular proof or global property one is trying to prove. To some extent, the abstract model formed by covering the input space (section 5.2) can also be used to help inform human understanding of the controller's behavior, though presumably such information requires more effort to digest than the plots due to their higher-dimensional nature and being presented as formulas as opposed to visuals.

  2. **Task-dependent:**A strength of CEGAR is the ability to provide counter examples to statements one proposes to be true about a controller. This allows members of our group to sanity check the learned controllers , to see that they are consistent with expected behavior. In the cases where a controller does not appear consistent with expected behavior, there are three options: (1) the controller is "flawed" and action needs to be taken to correct or address the issues resulting in the particular counter-example, (2) the human's understanding of the problem is incomplete, (3) the abstraction is too granular and the proposed counter-example is spurious. In the first case, the scientist in question might attempt to train the controller using more samples from the counter-example input region, in order to produce a better fit to desired results; it is not guaranteed that

[4] to be clear, this approach may still have utility in circumstances where these conditions do not hold

this sort of skewed-sampled training will produce the desired results while producing minimal side-effects, but in some circumstances it works well. In the second case, the counter example produced gives the human a concrete circumstance to think through which perhaps was not properly considered before. In the third case - which should be considered as a last resort in order to make sure human hubris does not prevent the corrections in (2) from occurring - the human can re-run the abstraction with finer granularity AIDs and parameters set to allow more granular examination.

3.2 Speculation on How These Techniques Could Most Effectively Be Used

It is the opinion of the author that one of the most promising use of AIDs and CEGAR is not as a push-button technique to produce proofs absent human intervention, but as a tool that an expert can "collaborate" with, where the tool enlightens the human to the controller behavior, and the human provides insight to the algorithm about where is it necessary to refine abstractions in order to complete proofs. There are also interesting possibilities - which the author intends to explore - for improving learning by over-sampling areas of the input space during training where intermediate-controllers violate desired properties.

While it is too early to say with strong conviction, it seems unlikely that, absent a human-in-the-loop to make a small number of critical adjustments, abstracting a learned model by uniformly covering the input space would produce controller descriptions that are both strong enough to prove non-trivial properties in a controller-plant model and computationally reasonable to include in proofs[5]. For example, if the input space is a four-dimensional box with side lengths L and we want to produce a description that is "granular" up to ϵ [6], then we would in general need at least $(\epsilon^{-1}L)^4$ conjunctions relating input boxes to output boxes - for ϵ sufficiently small for most proofs, this might be too burdensome in the proof process (too many branches to complete in reasonable time)[7]. A more palatable approach is to iterate use of AIDs or CEGAR with human analysis and focus the problem to select areas of the controller input-space that require more granular descriptions, preventing blind polynomial covering of the space. Both the former, task-agnostic analysis and the latter task-informed, expert-informed approach are valuable for our work - my point is that we should expect and accept that this technique is most powerful as a "partner" in design, not as a tool that either completes a proof independently or fails to be useful.

Another example of the most likely potential use is evaluating compliance with the monitors. This is distinct from the discussion in the previous paragraph in that the condition we are evaluating the controller against is fixed - in contrast to the controller-plant model situation, where the quality of our abstraction can effect what invariant is necessary to prove over iterations of the controller and plant,

[5] Then again, the alternative is double-exponential procedures such as quantifier elimination - but just because something is not most expensive does not imply it is cheap.

[6] that is, the boxes in the abstraction never have a side length greater than ϵ.

[7] Presumably our objectives and physics problems are crafted to be robust to environment noise, making it seem reasonable to assume that, given a correct controller, a sufficiently small ϵ would result in sufficiently granular abstractions to complete the proof given sufficient starting distance from safety boundaries. This is in contrast to problems where there exists critical hyperplanes over which the input-space must be divided, and any abstraction containing points on both sides of the hyperplane will always be too weak to complete the proof.

which in turn impacts whether our abstraction is sufficiently good (a potential vicious cycle)[8]. In the case of analyzing compliance to a provided monitor, CEGAR may be able to show what regions in the input space the monitor fails to comply, and an engineer can in turn use this information to refine the controller, either manually or as an automated train-then-evaluate-then-check-compliance loop. At a minimum, CEGAR has the potential to shine light on how well a monitor and a controller could be coupled together over a uniformly feasible input space.

The stake-holder here is not just the person trying to complete the proof. The people best served may well be the people making the system that down-stream we attempt to prove.

4 The Networks, AIDs and Mathmatical Soundness Proofs in Use

4.1 The Policy Network, the Input / Observation Space, and the Output/Action Space

NOTE: These networks and the nature of their pre-processing and post-processing were devised by Edward Ahn to the best of my current knowledge.

There are two networks in use currently by the group: a network trained to follow the line $y = 0$, and another designed to steer around a circle of fixed radius in a counter-clockwise direction, both in an x-y plane. The eventual intent is to use these policies, which are trained to perform well in environments that are relatively rigid, to perform in more sophisticated environments by appropriately transforming the observed state per-instance to resemble a particular case of problem for which the policies were trained; e.g., the line following policy can be made to follow arbitrary piece-wise linear paths by having a higher-level controller transform the coordinates so each segment - when the policy gets to that segment - is consider to be the line $y = 0$. A primary challenge in this work is designing safe but aggressive / efficient policies in environments where slipping and drifting can occur. Slippage and drifting are very relevant concerns for many applications of autonomous driving, whether it be off-road terrain, icy terrain, hauling large loads, or driving a high speed.

For either network, the architecture is the same. The network consists of three layers, each fully connected; there are two 32-unit hidden layers with tanh activation functions, and a 2-unit linear output layer.

For the line-following policy, the input space is five dimensional , representing the *relative* value of the vector (y, yaw, $\dot{x}$, $\dot{y}$, $y\dot{a}w$). This vector is relative to the line that is meant to be followed (which is, by default, is the line $y = 0$) - the relative nature of the observation is not critical for our purposes here, since the required transformation does not alter the possible input set to the policy, and thus does not effect our later function-image calculations. For more details on the transformation of the input space when running in the environment, see [redacted]

For the circle-following policy, the input space is four-dimensional, consisting of (Δr, θ, $\dot{\Delta r}$, $\dot{\theta}$) ,

[8]Short of integrating AIDs into KeYmeara ? or dReal, we would need to produce one fixed covering up front, and hope it is enough to satisfy the loop inter-dependencies.

where Δr is the difference between the ideal radius and the actual distance between the agent and the circle-center [9], and θ is angle-measure difference between the agent's current heading and the heading tangential to the circle at the agent's location. The policy is trained to move counter-clockwise around circle of radius one centered at the origin of the x-y plane. As before, we refer the reader to Edward's write-up for the transformation between the environment state and the state input to the network (i.e., the transformation between (x, y, yaw, $\dot{x}$, $\dot{y}$, $y\dot{a}w$) and (Δr, θ, $\dot{\Delta r}$, $\dot{\theta}$)) To avoid non-trivial work to push abstractions through the environment-state transform, our analysis in what follows simply works over the input space of the network (i.e., we look at the space over (Δr, θ, $\dot{\Delta r}$, $\dot{\theta}$) as opposed to (x, y, yaw, $\dot{x}$, $\dot{y}$, $y\dot{a}w$)). We also refer the reader to document ebc32bd9-3d55-4013-9d02-a6897cac2c09 , a communication between the author and Edward Ahn, for clarification regarding the interpretation of the variable signs; the author intends to concatenate this communication to the end of this write-up

For both networks, the output space consists of the vector (vel, yaw). The raw output from the network is post-processed to be scaled correctly and be in the acceptable range of vel $\in [0, 1.3]$ and yaw $\in [-\frac{\pi}{6}, \frac{\pi}{6}]$. The relevant scaling and clipping is implemented by **properlyTransformAction**, shown below. Notice, importantly for our future use, that **properlyTransformAction** is non-decreasing with thisAction, so the lexical ordering of the outputs is never the reverse of the lexical ordering of the inputs (i.e., if $\forall i \in [dim(\vec{x_1})].\{\vec{x_1}\}_i \leq \{\vec{x_2}\}_i$ then $\forall i \in [dim(\vec{x_1})].\{\textbf{properlyTransformAction}(\vec{x_1})\}_i \leq \{\textbf{properlyTransformAction}(\vec{x_2})\}_i$).

```
def properlyTransformAction(thisAction):
    lb = np.array([ 0.0 , -0.52359878]);
    ub = np.array([1.3, 0.52359878]);
    scaled_action = lb + (thisAction + 1.) * 0.5 * (ub - lb);
    scaled_action = np.clip(scaled_action, lb, ub);
    return scaled_action;
```

4.2 Propagating Boxes Through the Policy Network

In this section, we discuss how we conduction our abstract interpretation domains analysis. A crucial take away from this discussion is that this process allows us to produce guarantees over uncountably infinite many elements in the input space be considering finite collection of abstractions. In some of our later applications, we will see how covering the input space allows us to gain understanding of the network's behavior - these applications may seem reminiscent of plots producing by typical grid-search of the input space, feeding specific individual input vectors in and studying the resultant outputs. The difference between our approaches here and any simple sample based method is that here we can soundly make statements regarding entire infinite cardinality sets in the input and output space, where as sampling produces no guarantee unless one has additional analytic information about the stability of controller in question. The inability to ensure, in basic schemes, that samples accurately represent the behavior of the function under study results in at least: (1) labor intensive search-parameter tuning to build confidence that no obvious and critical feature was missed, (2) it not being uncommon for some features of regions of the space to be missed or inadequately represented

[9] Positive Δr means that the agent is further from the circle-center than the target radius.

due to being under sampled. In short, AIDs as used here produce formal guarantees and have many uses, where as sampling provides not formal insights in and of itself and cannot be comprehensive.

The abstract domains we consider for our AIDs are boxes. Boxes are easy to manipulate and easy to produce for bounding outputs, in contrast to more complex convex polytopes that would require solving constrains to determine boundaries and membership. This ease of manipulation comes at the cost of boxes being "less precise" than more sophisticated alternatives that have smaller volume while containing the same critical elements; larger sets suggest that certain relationships among variables may be feasible when in reality such combinations of values could not occur. As a separate note, observe that the process we will describe is related to local reachability analysis, since we compute the image of an input set under a policy π.

NOTE: the process described here leverages the fact that we are dealing with a pre-trained, fixed-weight feed-forward neural net. This would not, for instance, extend trivially to recurrent neural nets.

In order to propagate boxes through the network, we need to examine how to transform a box once is passes through a unit in the network. That is, if $u : \mathbb{R}^{I_u} \to \mathbb{R}^{O_u}$ is a unit of the network, and $\times_{i \in I_u}[a_i, b_i]$ is the input box, we need to calculate $u(\times_{i \in [I_u]}[a_i, b_i])$.

In the network in question, there are two types of activation functions: linear and tanh. Let $w \in \mathbb{R}^{I_u}$ be the weights of the unit, $b \in \mathbb{R}$ be the bias, and $x \in \mathbb{R}^{I_u}$ be the input value. We have that:

$$u_{linear}(x) =< w, x > +b$$
$$u_{tanh}(x) = tanh(< w, x > +b) = tanh(u_{linear}(x))$$

where, here, $< ., . >$ is the L2-norm. Notice that since $tanh$ is a strictly increasing function, the maximum and minimum of $u_{tanh}(x)$ occurs for the same candidate x values as the respective max or min as $u_{linear}(x)$. To find the inputs that produce the extreme values of the activation function over the input space, it thus suffices to find the values in $\times_{i \in [I_u]}[a_i, b_i]$ that maximize or minimize $< w, x >$. Trivial algebra show that:

$$\underset{x \in \times_{i \in [I_u]}[a_i, b_i]}{argmin} < x, w >=< b_i \mathbb{1}(\{w\}_i \leq 0) + a_i \mathbb{1}(\{w\}_i > 0) | i \in [I_u] >$$

$$\underset{x \in \times_{i \in [I_u]}[a_i, b_i]}{argmax} < x, w >=< a_i \mathbb{1}(\{w\}_i \leq 0) + b_i \mathbb{1}(\{w\}_i > 0) | i \in [I_u] >$$

where here, $< z_i | i >$ is sequence construction notation. With this, we can compute the images of the input space under the activation functions as follows: for $u \in \{u_{linear}, u_{tanh}\}$,

$$u(\times_{i \in I_u}[a_i, b_i]) = [u(\underset{x \in \times_{i \in [I_u]}[a_i, b_i]}{argmin} < x, w >), u(\underset{x \in \times_{i \in [I_u]}[a_i, b_i]}{argmax} < x, w >)]$$

Having established how a box should be propagated through a unit in the network, it is trivial to extend this to propagating a box through the entire network. If $u_{i,j}$ is the ith unit on the jth layer,

and $B_{i,j}$ is the input box to unit $u_{i,j}$, then one simply feeds the output box from one layer into the next as one would for a normal feed-forward network:

$$u_{i,j+1}(B_{i,j+1}) = u_{i,j+1}(\times_{h\in[I_{u_{i,j+1}}]} u_{h,j}(B_{h,j}))$$

This completes discussion of how the input box can be propagated through the network. Yet to consider, however, is how to push the box through the function **properlyTransformAction**. As noted before, however, **properlyTransformAction** respects partial ordering of the input vectors, thus,

$$\forall w \in \textbf{properlyTransformAction}(\times_{i\in[I_u]}[a_i, b_i]).$$
$$\forall h \in \{1,2\}.\{\textbf{properlyTransformAction}(< a_i | i \in [I_u] >)\}_h \leq \{w\}_h$$
$$\leq \{\textbf{properlyTransformAction}(< b_i | i \in [I_u] >)\}_h.$$

or, put another way, if

$$A = \textbf{properlyTransformAction}(< a_i | i \in [I_u] >)$$

and

$$B = \textbf{properlyTransformAction}(< b_i | i \in [I_u] >)$$

, then we have:

$$\textbf{properlyTransformAction}(\times_{i\in[I_u]}[a_i, b_i]) \subseteq [\{A\}_1, \{B\}_1] \times [\{A\}_2, \{B\}_2]$$

Even more strongly, it can be seen that the following holds:

$$\textbf{properlyTransformAction}(\times_{i\in[I_u]}[a_i, b_i]) = [\{A\}_1, \{B\}_1] \times [\{A\}_2, \{B\}_2]$$

since $< a_i | i \in [I_u] > \in \times_{i\in[I_u]}[a_i, b_i]$ and $< b_i | i \in [I_u] > \in \times_{i\in[I_u]}[a_i, b_i]$.

5 Task-Agnostic Analysis

5.1 Sampling the Input Space

In subsection section 4.2, it was established how a box may be pushed through our policy function, π. This provides stability information for a few subsets of the input space of π; we would like to garner such information over the entire input space $\mathscr{I}$ under the assumption that the set is bounded [10] . To perform such an analysis, we take strategically selected samples from $\mathscr{I}$, partitioning $\mathscr{I}$ into sets located around our samples, then use the machinery developed in section 4.2 to examine the stability in each partition.

First, we choose what our $\mathscr{I}$ is. For the line-following policy, from page 2 of the document provided by Edward Ahn, we take the following to be a reasonable superset for $\mathscr{I}$ to exist in:

[10] The boundedness assumption is for convenience; since the output of the neural net is clipped as detailed in section 4.1, we may consider axis which are unbounded (e.g., $x \in [-1, \infty]$) since the output set is necessarily bounded and thus potentially sensible.

- $y \in [-0.25, 0.25]$ m
- $yaw \in [-\frac{\pi}{3}, \frac{\pi}{3}]$ rad
- $\dot{x} \in [0, 1.3]$ m/s
- $\dot{y} \in [-0.6, 0.6]$ m/s
- $y\dot{a}w \in [-2.0, 2.0]$ rad/s

In order to generate easily understandable plots in these early investigations, we choose $\mathscr{I}$ to be

$$\mathscr{I} = \{w \in \mathbb{R}^5 | \{w\}_1 \in [-0.25, 0.25], \{w\}_2 \in [-\pi/3, \pi/3], \{w\}_3 = \dot{x}_H, \{w\}_4 = \dot{y}_H, \{w\}_5 = y\dot{a}w_H\}$$

where $\dot{x}_H$, $\dot{y}_H$, and $y\dot{a}w_H$ are hyper-parameters which we vary between different runs in order to produce different plots (see the results section for more details on how we handle these hyper-parameters). We chose to vary the "position-related" variables (y and yaw) since they are easier to interpret in terms of this experiment, in contrast to the *rate of change* of these quantities, as represented by the other three variables. Following a similar process for the circle-following policy, a reasonable superset for $\mathscr{I}$ to exist in is:

- $\Delta r \in [-0.25, 0.25]$m
- $\theta \in [-\pi, \pi]$ radians
- $\dot{\Delta r} \in [-1.5, 1.5]$ m/s
- $\dot{\theta} \in [-3.625, 3.625]$ radians/s

and we choose $\mathscr{I}$ to be:

$$\mathscr{I} = \{w \in \mathbb{R}^4 | \{w\}_1 \in [-0.25, 0.25], \{w\}_2 \in [-\pi, \pi], \{w\}_3 = \dot{\Delta r}_H, \{w\}_4 = \dot{\theta}_H\}$$

where as before, the subscript H denotes which variables are hyper-parameters [11].

From here, we draw samples from $\mathscr{I}$ by drawing from a grid produced over it. We introduce a hyper-parameter, ϵ, which controls the granularity of our analysis; in the context of controller stability analysis, for which we initially developed this[12], ϵ relates to bounds on the perturbations we are willing to consider. In what follows, we aim to refer to ϵ as the granularity, but might also refer to it as the magnitude of (acceptable) perturbation.

There is a trade-off between reducing the granularity we consider and the number of samples we need; the smaller the granularity we consider, the more samples needed and vice-versa. Being that we consider a grid over $\mathscr{I}$, and $\mathscr{I}$ is a box, we can describe how we select coordinates of our samples independently of one another. Define the following functions:

$$N(maxVal, minVal, \epsilon) \doteq \lceil \frac{maxVal - minVal}{2\epsilon} \rceil$$

[11] The super-set of $\mathscr{I}$ for circle-following is more questionable than the one selected for line-following; the former is a reasonable choice for early experimentation, but the author hopes to revise it at some point in the near future.

[12] see document d0b509df-70a7-4356-8b68-7b083d8b5208

$$c_k(maxVal, minVal, \epsilon) \doteq \frac{maxVal + minVal}{2} + 2\epsilon(\frac{N(maxVal, minVal, \epsilon)}{2} - k - \frac{1}{2})$$

$$\textbf{sampleCoordinates}(maxVal, minVal, \epsilon) \doteq \cup_{k \in [N(maxVal, minVal, \epsilon)]}\{c_k(maxVal, minVal, \epsilon)\}$$

In words, $\textbf{sampleCoordinates}(maxVal, minVal, \epsilon)$ produces a set of points, centered at the mean of maxVal and minVal, such that $[minVal, maxVal] \subseteq [c_{N(maxVal, minVal, \epsilon)} - \epsilon, c_0 + \epsilon]$ and $c_i - c_{i+1} = 2\epsilon$. We will return to the details of that in a moment. Our actual set of samples from $\mathscr{I}$ for the line-based policy is as follows:

$$\textbf{samples}(\mathscr{I}) =$$

$$\textbf{sampleCoordinates}(-0.25, 0.25, \quad 0.015 * (0.25 - (-0.25)) \quad) \times$$
$$\textbf{sampleCoordinates}(-\frac{\pi}{3}, \frac{\pi}{3}, \quad 0.015 * (\frac{\pi}{3} - (-\frac{\pi}{3})) \quad)$$

Notice that this means we consider ϵ to be roughly 1.5% of the range of the coordinates; for later reference, we will let ϵ_1 be this value $(0.015 * 2 * 0.25)$ on the first coordinate and ϵ_2 be this value on the second coordinate $(0.015 * 2 * \frac{\pi}{3})$. For the circle following policy, we consider:

$$\textbf{samples}(\mathscr{I}) =$$

$$\textbf{sampleCoordinates}(-0.25, 0.25, \quad 0.01 * (0.25 - (-0.25)) \quad) \times$$
$$\textbf{sampleCoordinates}(-\pi, \pi, \quad 0.01 * (\pi - (-\pi)) \quad)$$

Here, ϵ is roughly 1% of the range.

We digress for a moment to briefly show that our choice of samples is appropriate and does as claimed. First, notice that

$$c_i > c_{i+1}$$

and

$$c_i - \epsilon = c_{i+1} + \epsilon$$

thus

$$[c_i - \epsilon, c_i + \epsilon] \cup [c_{i+1} - \epsilon, c_{i+1} + \epsilon] = [c_{i+1} - \epsilon, c_i + \epsilon]$$

which in turn, by trivial induction, means

$$\cup_{x \in \textbf{sampleCoordinates}(maxVal, minVal, \epsilon)}[x - \epsilon, x + \epsilon] = [c_{N(maxVal, minVal, \epsilon)} - \epsilon, c_0 + \epsilon]$$

We see that this is a superset of $[minVal, maxVal]$ since basic algebra shows that $c_{N(maxVal, minVal, \epsilon)} - \epsilon \leq minVal$ and $maxVal \leq c_0 + \epsilon$. In addition to this, we see that

$$[c_i - \epsilon, c_i + \epsilon] \cap [c_{i+1} - \epsilon, c_{i+1} + \epsilon] = \{c_i - \epsilon\} = \{c_{i+1} + \epsilon\}$$

Thus, while this arrangement produces a covering of the set $[minVal, maxVal]$ when we center intervals of size 2ϵ at each sample, it also does not result in excessive overlap or excessive area outside of $[minVal, maxVal]$. This latter item - which translates in the set

$$(\cup_{x \in \textbf{sampleCoordinates}(maxVal, minVal, \epsilon)}[x - \epsilon, x + \epsilon]) \setminus [minVal, maxVal]$$

being small - is a property lacking from other slightly more natural and naive methods of sampling.

Thus far, we have discussed which samples we collect. It remains to state what we do with those samples and how they produce the information we desired. In general, to produce a human consumable, two-dimensional plot, we need a function that maps output-boxes to scalars, so we may somehow plot these scalar values across the free input parameters. We overview the use of two different such functions below: the first function we use measures the size of the output box, which provides information about the stability of the controller in different regions of the input-space. The second function we use looks at the minimum magnitude element in the output box, which translates roughly into the actual behavior of the controller for different input values.

5.1.1 Results: Stability Analysis

If $\pi : \mathbb{R}^k \to \mathbb{R}^n$ is a policy, we examined bounds on $\widetilde{\pi}(y) = \textbf{measure}(\{\pi(x) | x \in y + [-\epsilon, \epsilon]^k\})$, where y is an arbitrary given member of the input space $\mathbb{R}^k$, ϵ is a positive real bounding perturbations to the input, and **measure** is some defined method of assigning a size to a set (a measure of the set).

Ultimately, we want to bound $\widetilde{\pi}(w)$ for $w \in \mathscr{I}$, where we take the measure of a box to be the sum of the range of its coordinates [13]. By construction, and as justified in the previous paragraph, there exists $c \in \textbf{samples}(\mathscr{I})$ such that $w \in c + [-\epsilon_1, \epsilon_1] \times [-\epsilon_2, \epsilon_2]$. Thus, $w + [-\epsilon_1, \epsilon_1] \times [-\epsilon_2, \epsilon_2] \subset c + [-2\epsilon_1, 2\epsilon_1] \times [-2\epsilon_2, 2\epsilon_2]$, and in turn , this gives:

$$\widetilde{\pi}(w) \leq \textbf{measure}(\pi(c + [-2\epsilon_1, 2\epsilon_1] \times [-2\epsilon_2, 2\epsilon_2]))$$

Thus, to bound the local instability across $\mathscr{I}$, for each $c \in \textbf{sampleCoordinates}(maxVal, minVal, \epsilon)$ we bound the instability for each point in $c + [-\epsilon_1, \epsilon_1] \times [-\epsilon_2, \epsilon_2]$ by $\textbf{measure}(\pi(c + [-2\epsilon_1, 2\epsilon_1] \times [-2\epsilon_2, 2\epsilon_2]))$, recalling that $\cup_{c \in \textbf{sampleCoordinates}(maxVal, minVal, \epsilon)}(c + [-\epsilon_1, \epsilon_1] \times [-\epsilon_2, \epsilon_2]) \supseteq [minVal, maxVal]$.

To get a grasp of these values, we produce heatmaps of bounds on $\widetilde{\pi}(w)$ for each $w \in \mathscr{I}$. In order to make sense of the results more easily, we produce three heatmaps for each setting of the hyper-parameters of $\mathscr{I}$; if $[a_1, b_1] \times [a_2, b_2]$ is the box over the policy's output velocity and output steering angle after being fed a box centered around one of the samples we took, then we produce heatmaps for:

- Red: the normalized value of output velocity displacement: $\frac{b_1 - a_1}{\text{range of output velocity}} = \frac{b_1 - a_1}{0.5}$
- Blue: the normalized value of the output steering angle displacement $\frac{b_2 - a_2}{\text{range of output steering angle displacement}} = \frac{3(b_2 - a_2)}{2\pi}$
- Grey: the sum of the other two heatmaps, providing a weighted, L1 measure of displacement across the output box.

We include in this write-up several of the plots produced out of the hundreds available. We choose the hyper-parameters among the ranges listed in the bulleted list in section section 5.1. To expedite delivery of this document and to avoid purely implementation details, at the time being we spare discussion of how the hyperparameters were run in a particular order for efficient exploration.

[13] i.e, $\textbf{measure}(\times_{i \in [D]}[a_i, b_i]) = \sum_{i \in [D]}(b_i - a_i)$

- $\psi = \wedge_{i\in[B]}((x \in \times_{i\in[k]}[a_{i,j}, b_{i,j}]) \implies (y \in \times_{i\in[n]}[a'_{i,j}, b'_{i,j}]))$
- $\cap_{i\in[B]}(\text{interior}(\times_{i\in[k]}[a_{i,j}, b_{i,j}])) = \emptyset$
- $\cup_{i\in[B]}(\times_{i\in[k]}[a_{i,j}, b_{i,j}]) \supseteq \mathscr{I}$
- $y = \pi(x) \models \psi(x, y)$

Figure 1: Properties of a viable method of controller abstraction with AIDs. B is the number of conjuncts in ψ as well as the number of boxes used to cover the input space.

5.1.2 Results: Human Examination of Controller Behavior via Abstractions

Using more basic machinary than in section 5.1.1, we examine the following values the samples gathered:

$$\mathbf{beh}_i(w) = \underset{x\in\pi(w+[-\epsilon,\epsilon]^k)}{argmin} |\{x\}_i|$$

This function tells use that in the region $w + [-\epsilon, \epsilon]^k$ of the input space, the controller will produce an output with a certain minimum magnitude and the sign of that output. This is useful in the context of our robotic policies since a positive or negative steering angle tells us the direction the agent goes in, and the magnitude of that value tells us to what extreme. For the output velocity, we can discern where the robot will necessarily be moving, where is might be stopped, and where it might move quickly. In the case where $\mathbf{beh}_i(w) = 0$, we cannot determine whether the controller with perform a positive or negative action, or simply remain at zero. As we can see in the plots, this analysis is useful for determining necessarily when the controller performs in a way consistent with intuition, and the rough boundary between clearly distinct reactions.

As with the stability analysis, to get a grasp of these values, we produce heatmaps of $\mathbf{beh}_i(w)$ for each $w \in \mathscr{I}$. In order to make sense of the results more easily, we produce two heatmaps for each setting of the hyper-parameters - the red and blue heatmaps, as made for the stability analysis, expect the value displayed is for $\mathbf{beh}_i$. We do not form the grey plot since, while the "total instability" as a coupling of distinct output dimensions is sensible, such a coupling does not seem to be informative here. As with the stability results, we present several heatmaps for $\mathbf{beh}_i$ out of the hundreds available.

5.2 Controller Abstractions for Formal Use

It is possible to summarize the behavior of a learned controller as the conjunctions of conditionals over boxes that cover the input space. That is, given $\pi : \mathbb{R}^k \to \mathbb{R}^n$ and bounded $\mathscr{I}$, we can find a formula $\psi(x, y) : \mathbb{R}^k \times \mathbb{R}^n \to \{\top, \bot\}$ such that the properties in fig. 1 hold. Notice that the definition in fig. 1 allows boxes over the input space to overlap, though only on their boundaries. In fig. 2 we show part of an example derived from running on the network trained for circle-following.

The definition discussed thus far in this subsection gives little idea of how ψ would be found. Without additional criteria based on some proof to complete (i.e., in the case that we want to produce ψ in a task-agnostic fashion), we can simply cover $\mathscr{I}$ in an even-handed way. The sampling process over $\mathscr{I}$ described in section 5.1 easily extends to the full dimension of the input space; we can cover the input space by taking a sufficiently small box and translating it so that its center is located

$$\psi(x, y) =$$
$$(x \in [2.0, 4.0] \times [2.0, 4.0] \times [0.0, 0.2] \times [0.0, 0.5] \implies$$
$$y \in [0.35629961276068245, 1.3] \times [-0.15676543947217214, 0.52359878]) \wedge$$
$$(x \in [2.0, 4.0] \times [2.0, 4.0] \times [0.0, 0.2] \times [-0.5, 0.0] \implies$$
$$y \in [0.384418324439415, 1.3] \times [-0.14977990936003033, 0.52359878]) \wedge$$
$$(x \in [2.0, 4.0] \times [2.0, 4.0] \times [-0.2, 0.0] \times [0.0, 0.5] \implies$$
$$y \in [0.28179271170071807, 1.3] \times [-0.18260107011623045, 0.52359878]) \wedge$$
$$(x \in [2.0, 4.0] \times [2.0, 4.0] \times [-0.2, 0.0] \times [-0.5, 0.0] \implies$$
$$y \in [0.3142725135724193, 1.3] \times [-0.17352072493149368, 0.52359878]) \wedge \ldots$$

Figure 2: Part of an example task-agnostic abstracted controller model for use in down-stream controller-plant model proofs. The abstracted controller is a series of conditionals mapping input boxes which cover the acceptable input space to boxes containing the possible controller output.

at each of the sampled points. Unfortunately, this even-handed treatment of the inputs space can result in ψ that is highly complex in terms of its number of conjuncts given a reasonable granularity parameter, ϵ. For instance, if $\mathscr{I}$ is a k-dimensional cube of side-length L, we would need at least $(\lceil \frac{L}{\epsilon} \rceil)^k$ many cubes of side-length ϵ to cover $\mathscr{I}$; while automated proof assistants such as KeYmaera **?** may be able to handle the branching that occurs for $k \in \{4, 5\}$ with ϵ roughly 5% of L, it at a minimum would be computationally intensive and detract from human interpretation of produced proofs. Obviously, as ϵ becomes much smaller than L in this circumstance, even automated proof assistants would become overwhelmed. Trial-and-error is one method to determine which value of ϵ is enough to complete intended down-stream tasks. It is worth noting that while computational issues such as these are far from ideal, they are not a unique detractor of this approach; quantifier elimination and other corner-stones of our verification efforts are already have double-exponential runtimes.

In the case where an additional criteria is available to judge a covering of the input space, the approach discussed in section 6.3 is applicable, and helps alleviate some of the difficulties just discussed. It is worth noting that though we discussed ψ in this subsection primarily as a formal tool, the formula can also be used to inform human understanding of the controller's behavior as in section 6.2; the utility of such an approach might be more limited than in section 6.2 due to less focused filtering of the multi-dimensional data, but might also be more efficient than repeatedly running CEGAR with different objectives when an analyst is simply trying to inspect behavior for a task they are not initially familiar with.

6 Task-Informed Analysis

In section 4.2, we overviewed how to soundly push individual elements of an AID through the networks in question. In section 5.1, we discussed how to assemble multiple elements of an AID to

cover the input space and characterize the behavior of the learned controller over a specified input universe. While the task-agnostic methods outline have the potential to offer many insights, draw backs of the methods overviewed in section 5.2 are possible computation cost and size of the resulting abstraction (in terms of number of conjunctions) as the number of input blocks grows exponentially in the dimension of the input space for a fixed granularity parameter [14]. The central reason behind these difficulties is the need, given the lack of an objective provided up-front in the task-agnostic setting, to even-handedly describe the input-output relationship over the entire input space specified. Further, the materials from sections 5.1.2 and 5.1.1 were limited by human capabilities to exhaustively displaying only a lower-dimensional subspace of the input space, fixing the values of the remaining variables to define the subspace displayed. This results in it being necessary to produce a series of visuals, sampling the fixed parameters in order to gain an intuition of the higher-dimensional space - considering discrete samplings over the continuous variables limits some of what can be soundly inferred from these plots. As we will describe, in the task-dependent setting, we can intelligently attempt to alleviate these problems. The central machinery that powers this is inspired by Counter Example Guided Abstraction Refinement (CEGAR) Clarke et al. (2000).

6.1 Counter Example Guided Abstraction Refinement (CEGAR)

In the model checking world, CEGAR Clarke et al. (2000) is a well regarded technique for soundly ensuring a system meets desirable properties. In short, the approach uses intelligently adapted AIDs to attempt verification or refutation; in the case that the desirable property cannot be proven, the algorithm iteratively refines the abstraction based on where the property is in doubt, stopping when the property is either provable or has been disproven by a discovered counter example. When applied to certain families of discrete program, results returned by CEGAR are both sound and complete - this, however, comes at the cost of there not generally being a termination guarantee for CEGAR unless one is willing to allow sufficient approximations. Approximations in practice when using CEGAR tend to err on the side of safety- that if CEGAR indicates a property holds, then it is true, but the converse might not hold. This flexibility has allowed for extensions of the technique to many domains, including in hybrid system analysis Clarke et al. (2003), where the state space is necessarily uncountability infinite.

We now detail the CEGAR-like technique thus far implemented, using the abstraction domain described in section 4.2 as a base. As before, we let π be a learned controller implemented as a neural-net, $\pi : \mathscr{I} \to \mathbb{R}^{\mathscr{O}_\pi}$, where $\mathscr{I} \subseteq \mathbb{R}^{I_\pi}$ is the box $\times_{i \in [I_\pi]}[a_{\pi,i}, b_{\pi,i}]$. Let $\phi : \mathbb{R}^{I_\pi} \times \mathbb{R}^{\mathscr{O}_\pi} \to \{\top, \bot\}$ be a logical formula we would like to verify that π conforms to (i.e., $\forall w \in \mathscr{I}.\phi(w, \pi(w))$.). Since CEGAR is not generally guaranteed to terminate (and certainly would not in our CEGAR-like implementation), we introduce a function **STOP** : $\mathbb{R}^{\mathscr{O}_\pi} \to \{\top, \bot\}$ which will be used to prevent unbounded depth exploration when determining whether counter-examples are spurious or not. The outline of the basic algorithm is shown in algorithm 1. As we will discuss in a moment, some steps

[14] Arguably, this also has a direct negative impact on the human interpretability of the resulting abstraction, but, while the methods discussed in section 5.2 can help for such endeavors as a side-benefit, it was not their primary intent; it is the methods in sections 6.2 and 5.1.2 whose primary intent is to aid comprehension of what has been learned.

that typical CEGAR requires are missing, due to our slightly different aims.

The algorithm begins by forming an initial abstraction of the input space, hand designed by an expert based on their impression of what are informative separations. In our case, the initial abstraction is simply the high-dimensional "quadrants" (i.e., 2^{I_π} hyper-cubes formed with every combination of sign of the coordinates). The algorithm takes an input-abstraction, w, that has yet to be tried and forms $\pi(w)$. If $\phi(w, \pi(w))$ holds, then w is recorded as a successful area of the input space. If $\phi(w, \pi(w))$ fails to hold, there are two options. If $\mathbf{STOP}(w)$ holds, then we record w as a subset of the input space where π does not appear to satisfy ϕ. If, however, $\mathbf{STOP}(w)$ is false, we refine w and add the newly generated abstractions to the queue of inputs yet to be tried.

We have not yet attempted to include the typical CEGAR-machinary of checking whether a counter-example is spurious or not, and refining the abstractions in the case of the former. At the time being we do not do this since, in the following sections, we are not exclusively interested in whether or not a property holds over the entire input space, but where it fails if anywhere; even in the cases where we find a true counter-example, we want to continue to explore the containing box, so to gain a better idea of which subsets contain such counter-examples. Thus, while the full machinery of CEGAR may be helpful going forward to refine the space in a more informed fashion than what we do below, at the moment such mechanisms seem to buy little in addition to what is already being done.

Algorithm 1 A simplified version of the CEGAR-like analysis we conduct. Notice that, strictly speaking, it is not necessary to record both boxesContainingFailures and boxesContainingSuccesses since $\mathscr{I}$ is assumed to be given and $\cup_{z \in boxesContainingFailures}(z) = \mathscr{I} \setminus (\cup_{z \in boxesContainingSuccesses}(z))$. While not strictly necessary, it is convenient and offers non-trivial savings in computation and coding effort to simply record both as we generate them.

```
procedure BasicCegarLikeAnalysis(generateInitialAbstractions, 𝓘, STOP, φ)
    inputsToTry ← {}
    boxesContainingFailures ← {}
    boxesContainingSuccesses ← {}
    initialAbstractions ← generateInitialAbstractions(𝓘)
    inputsToTry ← initialAbstractions
    while |inputsToTry| > 0 do
        inputBox ← inputsToTry.pop()
        inputsToTry ← inputsToTry \ {inputBox}
        if φ(inputBox, π(inputBox)) then
            boxesContainingSuccesses ← boxesContainingSuccesses ∪ {inputBox}
        else
            boxesContainingFailures ← boxesContainingFailures ∪ {inputBox}
        end if
        if ¬STOP(inputBox) then
            inputsToTry ← inputsToTry ∪ refine(inputBox)
        end if
    end while
return (boxesContainingSuccesses, boxesContainingFailures)
end procedure
```

Currently, as a simple but general start, our method of refinement is simply to split the box in question along the longest "scaled" axis. Rigorously, given a box to refine, $\times_{i\in[I_\pi]}[a'_i, b'_i]$, we form two new boxes as follows:

$$h = \underset{i\in[I_\pi]}{argmax} \frac{b'_i - a'_i}{b_{\pi,i} - a_{\pi,i}}$$

$$C = \frac{b'_h - a'_h}{2}$$

$$\textbf{refine}(\times_{i\in[I_\pi]}[a'_i, b'_i]) = \{\times_{i\in[I_\pi]}[a'_i, \mathbb{1}(i \neq h)b'_i + \mathbb{1}(i = h)C],\ \times_{i\in[I_\pi]}[\mathbb{1}(i \neq h)a'_i + \mathbb{1}(i = h)C, b'_i]\}$$

The use of $b_{\pi,i} - a_{\pi,i}$ is in an attempt to control for differences in scaling and meaning among the variables comprising the input space. For instance, 20 millimeters is not commiserate with 20 radians, and our sensitivity to 3 centimeters of difference may be different given a quality that is typically on par of kilometers versus one never more than a decimeter.

A more sophisticated version of our algorithm is shown in algorithm 2. It improves on 1 by recording more granular, interior-disjoint failure boxes, which is important for concise descriptions of regions where ϕ might not hold. As noted in the caption of algorithm 1, while recording both successes and failures might not be mathematically necessary, doing so saves computational and coding expense in down-stream handling. Current work on the agenda going forward includes implementation of a basic scheme to merge boxes found in boxesContainingFailures (or boxesContainingSuccesses respectively) as a post-processing step; finding the minimal number of boxes that strictly and exclusively contain the elements in question is likely np-complete, but might be sub-modular and thus reasonably approximatable by basic methods.

6.2 Uses for Human Understanding: Sanity Checking Learned Controllers

Given a task in robotics, an engineer often has strong intuition as to the qualitative behavior of an appropriate controller in certain circumstances. CEGAR can be used to observe under what circumstances a controller definitely behaves in a manner consist with a priori expectations. Beyond confirming expectations, CEGAR can be used in a similar process of iteratively crafting and refining questions about a controller to gain insight into its behavior over the multi-dimensional input-output space. Information garnered in this way can be used to:

- **Improve Learned Controller Design and Highlight Problems:** If a given learned controller fails to agree with expected behavior, this information coupled with specific information about what cases are problematic (as provided by CEGAR) can guide redesign and improve learning.
- **Improve Human Understanding of General Problem:** If controller behavior over parts of the space contradict a priori human expectations, further exploration of the source of the disparity may help shed light on overlooked facets of the problem, improving human understanding of the task. For the robotics task we are currently considering, this may be of particular use since the input-output dimensions are higher than what can be directly visualized for humans, but low enough dimensional to by manually inspected. Further, for the robotics task we are considering, each variable in the input and output space has its own significantly

Algorithm 2 An improved version of the CEGAR-like analysis we conduct.

```
procedure DiveIntoBox(thisInputBox, ϕ, STOP):
    thisOutputBox = π(thisInputBox)
    if ϕ(thisInputBox, thisOutputBox) then
        return (⊤, {thisInputBox}, {}) //success
    else if STOP(thisInputBox) then
        return (⊥, {}, {}) //failure
    else
        boxesContainingFailures ← {}
        boxesContainingSuccesses ← {}
        refimentElements ← refine(inputBox)
        successfulChildren ← {}
        for all thisChildBox ∈ refimentElements do
            childResult ← DiveIntoBox(thisChildBox, ϕ, STOP)
            if childResult[0] then
                boxesContainingSuccesses ← boxesContainingSuccesses ∪ childResult[1]
                boxesContainingFailures ← boxesContainingFailures ∪ childResult[2]
                successfulChildren ← successfulChildren ∪ {thisChildBox}
            end if
        end for
        if |successfulChildren| > 0 then
            boxesContainingFailures ← boxesContainingFailures ∪ (refimentElements \
successfulChildren)
            return (⊤, boxesContainingSuccesses, boxesContainingFailures) //success
        else
            return (⊥, {}, {}) //failure
        end if
    end if
    raise Exception(); //Control should never reach here
    return
end procedure
procedure CegarLikeAnalysis(generateInitialAbstractions, 𝒥, STOP, ϕ)
    boxesContainingFailures ← {}
    boxesContainingSuccesses ← {}
    inputsToTry ← generateInitialAbstractions(𝒥)
    while |inputsToTry| > 0 do
        inputBox ← inputsToTry.pop()
        inputsToTry ← inputsToTry \ {inputBox}
        thisResult ← DiveIntoBox(thisInputBox, ϕ, STOP)
        if thisResult[0] then
            boxesContainingSuccesses ← boxesContainingSuccesses ∪ thisResult[1]
            boxesContainingFailures ← boxesContainingFailures ∪ thisResult[2]
        else
            boxesContainingFailures ← boxesContainingFailures ∪ {thisInputBox}
        end if
    end while
    return (boxesContainingSuccesses, boxesContainingFailures)
end procedure
```

independent and human comprehensible meaning in terms of the state. Both of these points are in contrast to, say, the several-thousand-dimensional inputs to neural networks tasks with image recognition [15] where an individual pixel should not carry much importance on its own, and the exact relationship between pixels and outputs is usually unintelligible and unintuitive.

- **Reduction of CEGAR Granularity or Explicit Search for Counter Examples:** In the event that the engineer has double-checked themselves or generally believes CEGAR was too conservative [16] , the granularity used to determine when **STOP** should be true can be reduced, potentially reducing spurious counter examples. Further, if one is interested first-and-foremost in finding a counter-example to ψ, CEGAR can be used with $\neg\psi$ to find if there are boxes over the input-space where every element of the box satisfies $\neg\psi$ - i.e., every element of the box is a counter-example to ψ.

In an attempt to sanity check the circle-following policy, we examined if the policy conformed to the following expression [17]:

$$(sign(\Delta r) = sign(\theta) = q) \implies (\text{vel} > 0 \wedge sign(\text{steering angle}) = q)$$

$$(sign(\Delta r) \in \{0, q\} \wedge sign(\theta) \in \{0, q\}) \implies (sign(\text{steering angle}) \in \{0, q\})$$

In words, the condition essentially says that is the robot is inside the circle and pointing inwards, it tries to drive outward; symmetrically, is the robot is outside the circle and facing outward, it tries to steer back towards the circle. Note that, based on how we are forming the initial abstractions, we know a priori that at least half the input space will satisfy these conditions by simply failing the hypotheses. Our analysis used the following **STOP** function, motivated for similar reasons as **refine** from section 6.1:

$$\mathbf{STOP}(\times_{i\in[k]}[a'_i, b'_i]) = (\ \ (\underset{i\in[I_\pi]}{argmax} \frac{b'_i - a'_i}{b_{\pi,i} - a_{\pi,i}}) \leq \frac{1}{32}\ \ )$$

As can be seen in the results in section 8.1, counter-examples were found to our sanity check. Examining them, we at least observed that the condition fails to hold when $sign(\dot{\Delta r}) = sign(\dot{\theta}) \neq sign(\Delta r) = sign(\theta)$. We reason that, in the case that the car is rapidly drifting toward the target circle radius and rotating toward $\theta = 0$, the car may actually need to counter-steer to avoid loosing control and avoid rotating too far - whether the car is drifting and in what way it is are represented by $\dot{\Delta r}$ and $\dot{\theta}$. Weakening our sanity-check to account for this condition we consider:

$$(sign(\Delta r) = sign(\theta) = sign(\dot{\Delta r}) = sign(\dot{\theta}) = q) \implies (\text{vel} > 0 \wedge sign(\text{steering angle}) = q)$$

$$(sign(\Delta r) \in \{0, q\} \wedge sign(\theta) \in \{0, q\} \wedge sign(\dot{\Delta r}) \in \{0, q\} \wedge sign(\dot{\theta}) \in \{0, q\}) \implies (sign(\text{steering angle}) \in \{0, q\})$$

As before, we note that at least $\frac{7}{8}^{\text{th}}$ of the input space will satisfy these conditions just by failing the hypotheses. See section 8.1 for the content of this analysis.

[15] Or outputs from image generative networks using GANs.

[16] i.e., spuriously labeled too much of the input-space as possibly failing to satisfy ψ

[17] Reminder of definition:

$$\forall x \in \mathbb{R}.((x < 0) => sign(x) = -1) \wedge ((x > 0) => sign(x) = 1) \wedge ((x = 0) => sign(x) = 0)$$

Note that while in section 8.1 we make use of further filters on boxes to extract items of interest, this approach (algorithm 2) provides the critical material on which to base further investigation: meaningful segmentations of the input space that have soundly be divided base on criteria of interest and which attempt to be both maximally granular and precise [18].

6.3 Use in Formal Proofs (Particularly for Studying Controllers with Their Monitors)

We can use CEGAR is a similar process as what was described in section 5.2 , using CEGAR to selectively refine the abstraction so that that there are more boxes at locations where we need be more precise to make the proof go through, as opposed to uniformly over the entire input space.

If CEGAR was incorporated directly into the proof process for a controller-plant model, it is not clear if we might suffer from the effects of a vicious cycle; in order to complete a proof in a controller-plant model (say in dL), we need the model abstraction to satisfy certain loop invariants, but refining the abstraction to support those invariants changes to model, in turn potentially changing the invariants we want to show. One would hope that iteratively refining with CEGAR for each new loop invariant (as opposed to starting CEGAR from scratch for each new loop invariant) would resolve this issue, assuming that controller is in fact correct, but it is not immediately clear this process would "converge-enough" to handle all necessary corner-cases.

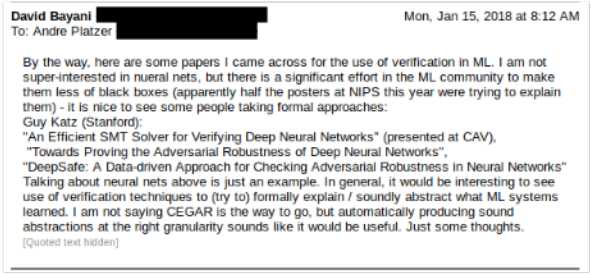
Examining when a learned controller conforms to the monitoring condition nicely side-steps some of the concerns raised about using it in proof with the controller-plant model. The monitor condition is not effected by our abstraction, and is an explicit condition over the inputs and outputs of the controller; this matches nicely with what can be done with CEGAR as proposed. Further, from a practical implementation perspective, since the monitor condition can be given up-front, we can use it in CEGAR simply as ϕ - this is in contrast to attempting a proof with the controller-plant model, which would seem to require that CEGAR be in a loop with the theorem prover in use.

7 Prior Intent and Some Prior Work

The author would like to note that it has been his interest and intent to study the use of CEGAR for verification tasks of this variety well before the existence of the AA Project. As some evidence of such, emails from day15 month1 year2018 show consideration of the subject (see fig. 3).

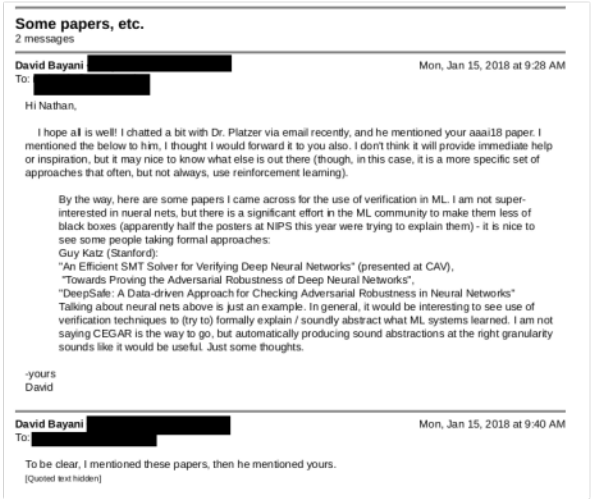
In pursuing the subject beyond what is listed in the emails (fig. 3), the author came across material from the 1990s that is relevant to this work, though never got around to reading it beyond the abstract and conclusion, but recalls saving a copy or link reference in hopes of reading later. The author hopes to update this document with a link to those pdfs when he digs those materials out of personal archives which are not immediately accessible. The author intended to send out this write-up on day7 month6 year2019 - it is not day12 month6 year2019. In order to expedite the process, the author aims to come back with this content later.

[18] I.e., we claim that there might be a counter-example in a big box only after checking that there might be counter-examples in each of the small boxes that constitute it.



David Bayani [redacted] Mon, Jan 15, 2018 at 8:12 AM
To: Andre Platzer [redacted]

By the way, here are some papers I came across for the use of verification in ML. I am not super-interested in nueral nets, but there is a significant effort in the ML community to make them less of black boxes (apparently half the posters at NIPS this year were trying to explain them) - it is nice to see some people taking formal approaches:
Guy Katz (Stanford):
"An Efficient SMT Solver for Verifying Deep Neural Networks" (presented at CAV),
"Towards Proving the Adversarial Robustness of Deep Neural Networks",
"DeepSafe: A Data-driven Approach for Checking Adversarial Robustness in Neural Networks"
Talking about neural nets above is just an example. In general, it would be interesting to see use of verification techniques to (try to) formally explain / soundly abstract what ML systems learned. I am not saying CEGAR is the way to go, but automatically producing sound abstractions at the right granularity sounds like it would be useful. Just some thoughts.
[Quoted text hidden]



Some papers, etc.
2 messages

David Bayani [redacted] Mon, Jan 15, 2018 at 9:28 AM
To: [redacted]

Hi Nathan,

I hope all is well! I chatted a bit with Dr. Platzer via email recently, and he mentioned your aaai18 paper. I mentioned the below to him, I thought I would forward it to you also. I don't think it will provide immediate help or inspiration, but it may nice to know what else is out there (though, in this case, it is a more specific set of approaches that often, but not always, use reinforcement learning).

> By the way, here are some papers I came across for the use of verification in ML. I am not super-interested in nueral nets, but there is a significant effort in the ML community to make them less of black boxes (apparently half the posters at NIPS this year were trying to explain them) - it is nice to see some people taking formal approaches:
> Guy Katz (Stanford):
> "An Efficient SMT Solver for Verifying Deep Neural Networks" (presented at CAV),
> "Towards Proving the Adversarial Robustness of Deep Neural Networks",
> "DeepSafe: A Data-driven Approach for Checking Adversarial Robustness in Neural Networks"
> Talking about neural nets above is just an example. In general, it would be interesting to see use of verification techniques to (try to) formally explain / soundly abstract what ML systems learned. I am not saying CEGAR is the way to go, but automatically producing sound abstractions at the right granularity sounds like it would be useful. Just some thoughts.

-yours
David

David Bayani [redacted] Mon, Jan 15, 2018 at 9:40 AM
To: [redacted]

To be clear, I mentioned these papers, then he mentioned yours.
[Quoted text hidden]

Figure 3: Some email content suggesting use of CEGAR for verification of the variety of robotics tasks of which the AA Group is now concerned. To the best of the author's knowledge, these emails were sent prior to the existence of the AA Project.

References

E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *International Conference on Computer Aided Verification*, pages 154–169. Springer, 2000.

E. Clarke, A. Fehnker, Z. Han, B. Krogh, O. Stursberg, and M. Theobald. Verification of hybrid systems based on counterexample-guided abstraction refinement. In H. Garavel and J. Hatcliff, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 192–207, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-36577-8.

P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '77, pages 238–252, New York, NY, USA, 1977. ACM. doi: 10.1145/512950.512973. URL `http://doi.acm.org/10.1145/512950.512973`.

S. Gao, S. Kong, and E. M. Clarke. dreal: An smt solver for nonlinear theories over the reals. In *International conference on automated deduction*, pages 208–214. Springer, 2013.

A. Platzer. *Differential Dynamic Logics: Automated Theorem Proving for Hybrid Systems*. PhD thesis, Department of Computing Science, University of Oldenburg, 2008.

8 Appendix

8.1 Result: Task-Informed Analysis, Human-Oriented Material

```
While interpreting the below, keep in mind that:
(1) Boxes include their boundaries, and that boxes where a condition is true
may border boxes where a condition is false. This means, for instance, the
```

```
maximum/minimum magnitude element where a condition is true might also might
be a maximum/minimum magnitude "element" (boundary point) among the set of
points where the condition might not hold.
(2) Elements in a box were a condition might not hold do not necessarly violate
the condition. Because box abstraction domains contain a lot of relationships
that might not actual occur in the data (i.e. would not occur if each element
was pushed through the network individually), spurious counter-examples can occur.
That means that it is not a sign of a bug in the code or process if some elements
of a box were the condition might not hold are found to obey the condition.
(3) Due to some neglect of considering corner-cases when forming the condition
to check for coupled with the initial abstraction, if a box has a corner with zero,
then likely that box failed the condition checks. This may explain some of the
behavior seen below.
(4) The initial abstraction split the input-space into boxes based on the sign
of the coordinates. Thus, the input space is at least divided into
2^{dimension of input-vector} many boxes to begin with, limiting the size of any
individual box.




V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V
Details about the following CEGAR trials:
Results when placing hypothesis only on the delta-r and theta:
boxes where condition might not have always been true
=======================================================
Total number of boxes of this variety
16792
Total volume of boxes of this variety
15.235281493061223
The input space:
[[-0.25        0.25      ]
 [-1.57079633  1.57079633]
 [-1.5         1.5       ]
 [-3.625       3.625     ]]
Percent of Input Space taken by boxes of this variety
0.4459347786698235
---

Details about the Biggest Box
The box: [[-0.25       -0.125     ]
 [-0.78539819  0.        ]
 [-1.5         0.        ]
 [ 0.          3.625     ]]
The output box from processing this input box:[[ 0.          1.3       ]
```

```
 [-0.52359878  0.52359878]]
Randomly sampled vectors from the Box
Random sample 0
Input Vector:[[-0.17593319]
 [-0.34872378]
 [-1.26439035]
 [ 0.5750324 ]]
Network Output[ 0.92247688 -0.10741966]
Random sample 1
Input Vector:[[-0.21230754]
 [-0.71493328]
 [-1.25698048]
 [ 2.76264041]]
Network Output[1.3        0.12232909]
Random sample 2
Input Vector:[[-0.1634408 ]
 [-0.57548135]
 [-1.22500189]
 [ 2.69228159]]
Network Output[1.3      0.214755]
Random sample 3
Input Vector:[[-0.20928589]
 [-0.44719676]
 [-0.93752303]
 [ 2.56821258]]
Network Output[1.3        0.23221545]
Random sample 4
Input Vector:[[-0.17621171]
 [-0.59483274]
 [-0.98399915]
 [ 1.85317617]]
Network Output[1.3        0.10616666]
Random sampling to find example input-outputs that violate desired conditions
    <input vector>:<output vector>
    [-0.24234877 -0.54886858 -0.34923189  2.07042364]:[1.3        0.08683613]
    [-0.15411424 -0.75590812 -1.3713439   2.71140114]:[1.3        0.12948042]
    [-0.1559084  -0.21293961 -1.4642298   1.94661085]:[1.3        0.25745246]
    [-0.22049175 -0.5015714  -0.25710575  1.88620746]:[1.3        0.08968075]
    [-0.20728567 -0.53267549 -0.73482618  3.03488893]:[1.3        0.21582843]
----
---

Details about Box Containing Minimum Magnitude Element
The box:[[0.         0.0625    ]
 [0.         0.39269909]
```

```
 [0.         0.75      ]
 [0.         1.8125    ]]
The output box from processing this input box:[[ 0.          1.3       ]
 [-0.52359878  0.52359878]]
The minimum magnitude element in the box:[0 0 0 0]
Network Output on the max magnitude vector:[0.69063734 0.15590869]
Randomly sampled vectors from the Box
Random sample 0
Input Vector:[[0.04144776]
 [0.31247619]
 [0.28802486]
 [1.58225243]]
Network Output[0.7466398  0.52359878]
Random sample 1
Input Vector:[[0.0475715 ]
 [0.2300228 ]
 [0.69560591]
 [0.7305592 ]]
Network Output[0.65177146 0.48638296]
Random sample 2
Input Vector:[[0.02015785]
 [0.14839234]
 [0.16466162]
 [1.11140083]]
Network Output[0.76978499 0.48569831]
Random sample 3
Input Vector:[[0.03124579]
 [0.25840651]
 [0.21423333]
 [1.0659601 ]]
Network Output[0.76719629 0.52359878]
Random sample 4
Input Vector:[[0.01898963]
 [0.13593674]
 [0.30559362]
 [0.12764385]]
Network Output[0.68413375 0.32252329]
Random sampling to find example input-outputs that violate desired conditions
    <input vector>:<output vector>
        number of attempted random samples thus far:1000
        number of attempted random samples thus far:2000
        number of attempted random samples thus far:3000
        number of attempted random samples thus far:4000
        number of attempted random samples thus far:5000
        number of attempted random samples thus far:6000
```

```
        number of attempted random samples thus far:7000
        number of attempted random samples thus far:8000
        number of attempted random samples thus far:9000
        number of attempted random samples thus far:10000
Exhausted pre-defined max number of random samples (10000) and did not find an
input-output instance violating conditions. Note that if violating instances
exist only on the boundary of the box (or exclusively on any other
lower-dimensional manifold) then the probabilty of sampling them (absent
computer-precision discretization) is zero
----
---

Details about Box Containing Maximum Magnitude Element
The box:[[-0.25       -0.21875   ]
 [-1.57079637 -1.37444675]
 [-1.5        -1.3125    ]
 [ 2.71875     3.625     ]]
The output box from processing this input box:[[ 0.          1.3       ]
 [-0.52359878  0.52359878]]
The maximum magnitude element in the box:[-0.25   -1.57079637 -1.5      3.625   ]
Network Output on the max magnitude vector[ 1.3        -0.27763788]
Randomly sampled vectors from the Box
Random sample 0
Input Vector:[[-0.22496371]
 [-1.43530494]
 [-1.33930758]
 [ 3.27534258]]
Network Output[ 1.3        -0.25340041]
Random sample 1
Input Vector:[[-0.24614962]
 [-1.50448369]
 [-1.48061604]
 [ 3.58569105]]
Network Output[ 1.3        -0.24973103]
Random sample 2
Input Vector:[[-0.24279759]
 [-1.42051783]
 [-1.39895813]
 [ 3.50893837]]
Network Output[ 1.3        -0.22015298]
Random sample 3
Input Vector:[[-0.23838088]
 [-1.43107561]
 [-1.3635306 ]
 [ 3.20773948]]
```

```
Network Output[ 1.3        -0.24115991]
Random sample 4
Input Vector:[[-0.24947652]
 [-1.4626892 ]
 [-1.39984172]
 [ 3.34943322]]
Network Output[ 1.3        -0.24261601]
Random sampling to find example input-outputs that violate desired conditions
    <input vector>:<output vector>
        number of attempted random samples thus far:1000
        number of attempted random samples thus far:2000
        number of attempted random samples thus far:3000
        number of attempted random samples thus far:4000
        number of attempted random samples thus far:5000
        number of attempted random samples thus far:6000
        number of attempted random samples thus far:7000
        number of attempted random samples thus far:8000
        number of attempted random samples thus far:9000
        number of attempted random samples thus far:10000
Exhausted pre-defined max number of random samples (10000) and did not find an
input-output instance violating conditions. Note that if violating instances
exist only on the boundary of the box (or exclusively on any other
lower-dimensional manifold) then the probabilty of sampling them (absent
computer-precision discretization) is zero
----
---

Random example boxes
Random example box number 0
[[0.234375   0.25      ]
 [0.09817477 0.19634955]
 [0.         0.1875    ]
 [1.8125     2.265625  ]]
Random example box number 1
[[-0.171875   -0.15625   ]
 [-1.07992244 -0.98174769]
 [ 0.          0.1875    ]
 [ 3.171875    3.625     ]]
Random example box number 2
[[-0.109375   -0.09375   ]
 [-1.37444675 -1.27627206]
 [ 0.375       0.5625    ]
 [ 2.71875     3.171875  ]]
Random example box number 3
[[-0.109375   -0.09375   ]
```

```
 [-1.37444675 -1.27627206]
 [ 0.09375     0.1875    ]
 [-2.0390625  -1.8125    ]]
Random example box number 4
[[ 0.1875      0.203125  ]
 [ 0.88357294  0.98174769]
 [-1.3125     -1.21875   ]
 [ 3.171875    3.3984375 ]]
---

^_^_^_
```

```
 [-3.60013647]]
Network Output[0.49315228 0.39122378]
Random sample 1
Input Vector:[[-0.08620887]
 [ 0.44159576]
 [-0.43469595]
 [-1.49008952]]
Network Output[0.77504128 0.37319521]
Random sample 2
Input Vector:[[-0.14620278]
 [ 0.39875045]
 [-0.35029321]
 [-1.41692333]]
Network Output[0.73289434 0.20010171]
Random sample 3
Input Vector:[[-0.15645758]
 [ 0.69159655]
 [-0.72438243]
 [-0.73992171]]
Network Output[0.45185477 0.33861539]
Random sample 4
Input Vector:[[-0.0455638 ]
 [ 0.70034048]
 [-0.98336659]
 [-0.83611976]]
Network Output[0.55605631 0.52359878]
---

Details about Box Containing Minimum Magnitude Element
The box:[[ 0.          0.25      ]
 [-1.57079637  0.        ]
 [ 0.          1.5       ]
 [ 0.          3.625     ]]
The output box from processing this input box:[[ 0.          1.3       ]
 [-0.52359878  0.52359878]]
The minimum magnitude element in the box:[0 0 0 0]
Network Output on the max magnitude vector:[0.69063734 0.15590869]
Randomly sampled vectors from the Box
Random sample 0
Input Vector:[[ 0.12344697]
 [-0.88891854]
 [ 0.04115726]
 [ 1.41312424]]
Network Output[ 0.5042269  -0.47087659]
Random sample 1
```

```
Input Vector:[[ 0.05264069]
 [-0.06330388]
 [ 0.14039057]
 [ 2.16539021]]
Network Output[0.53123516 0.45719858]
Random sample 2
Input Vector:[[ 0.07262579]
 [-0.39187776]
 [ 0.92726212]
 [ 1.48881124]]
Network Output[ 0.53658895 -0.08656067]
Random sample 3
Input Vector:[[ 0.10153637]
 [-0.97825216]
 [ 1.39991854]
 [ 2.02424882]]
Network Output[ 0.70688046 -0.52359878]
Random sample 4
Input Vector:[[ 0.08048548]
 [-0.4484179 ]
 [ 1.17193732]
 [ 3.04225341]]
Network Output[ 0.52581093 -0.0344961 ]
---

Details about Box Containing Maximum Magnitude Element
The box:[[-0.25       -0.234375  ]
 [-1.57079637 -1.47262156]
 [ 1.40625     1.5       ]
 [ 3.3984375   3.625     ]]
The output box from processing this input box:[[ 0.12043011  1.3       ]
 [-0.52359878 -0.52359878]]
The maximum magnitude element in the box:[-0.25     -1.57079637  1.5     3.625  ]
Network Output on the max magnitude vector[ 0.7838645  -0.52359878]
Randomly sampled vectors from the Box
Random sample 0
Input Vector:[[-0.23715144]
 [-1.49797706]
 [ 1.47478005]
 [ 3.53357006]]
Network Output[ 0.77242562 -0.52359878]
Random sample 1
Input Vector:[[-0.23759911]
 [-1.57073482]
 [ 1.42875244]
```

```
 [ 3.46618022]]
Network Output[ 0.76958498 -0.52359878]
Random sample 2
Input Vector:[[-0.24793301]
 [-1.55570428]
 [ 1.49011043]
 [ 3.50920478]]
Network Output[ 0.7722669  -0.52359878]
Random sample 3
Input Vector:[[-0.23437836]
 [-1.56499902]
 [ 1.45670642]
 [ 3.54567281]]
Network Output[ 0.77401313 -0.52359878]
Random sample 4
Input Vector:[[-0.24247958]
 [-1.49688814]
 [ 1.4595178 ]
 [ 3.6070291 ]]
Network Output[ 0.78473945 -0.52359878]
---

Random example boxes
Random example box number 0
[[ 0.234375    0.25      ]
 [ 1.47262156  1.57079637]
 [-1.40625    -1.3125    ]
 [-1.359375   -1.1328125 ]]
Random example box number 1
[[ 0.109375    0.125     ]
 [ 1.52170897  1.57079637]
 [-0.46875    -0.375     ]
 [-1.359375   -1.1328125 ]]
Random example box number 2
[[-0.0625     -0.046875  ]
 [-1.47262156 -1.37444675]
 [ 1.3125      1.40625   ]
 [-0.2265625   0.        ]]
Random example box number 3
[[-0.0625     -0.046875  ]
 [-1.27627206 -1.17809725]
 [ 0.          0.09375   ]
 [-3.171875   -2.71875   ]]
Random example box number 4
[[-0.09375    -0.078125  ]
```

```
 [-1.07992244 -0.98174769]
 [ 1.40625     1.5       ]
 [-2.71875    -2.4921875 ]]
---

^_^_^_
```

```
 [-0.85103867]
 [-1.2627313 ]]
Network Output[ 0.27200663 -0.52359878]
Random sample 2
Input Vector:[[-0.15569718]
 [-0.27388343]
 [-1.05764418]
 [-1.21561587]]
Network Output[ 0.51955637 -0.44992961]
Random sample 3
Input Vector:[[-0.20369927]
 [-0.14191382]
 [-1.10754375]
 [-1.52598061]]
Network Output[ 0.44642205 -0.41391594]
Random sample 4
Input Vector:[[-0.16542862]
 [-0.12797351]
 [-0.87446939]
 [-1.70162269]]
Network Output[ 0.49756649 -0.40068855]
Random sampling to find example input-outputs that violate desired conditions
    <input vector>:<output vector>
        number of attempted random samples thus far:1000
        number of attempted random samples thus far:2000
        number of attempted random samples thus far:3000
        number of attempted random samples thus far:4000
        number of attempted random samples thus far:5000
        number of attempted random samples thus far:6000
        number of attempted random samples thus far:7000
        number of attempted random samples thus far:8000
        number of attempted random samples thus far:9000
        number of attempted random samples thus far:10000
Exhausted pre-defined max number of random samples (10000) and did not find an
input-output instance violating conditions. Note that if violating instances
exist only on the boundary of the box (or exclusively on any other
lower-dimensional manifold) then the probabilty of sampling them (absent
computer-precision discretization) is zero
----
---

Details about Box Containing Minimum Magnitude Element
The box:[[0.         0.0625    ]
 [0.         0.39269909]
 [0.         0.75      ]
```

```
 [0.         1.8125    ]]
The output box from processing this input box:[[ 0.          1.3       ]
 [-0.52359878  0.52359878]]
The minimum magnitude element in the box:[0 0 0 0]
Network Output on the max magnitude vector:[0.69063734 0.15590869]
Randomly sampled vectors from the Box
Random sample 0
Input Vector:[[0.03081033]
 [0.14486769]
 [0.10850271]
 [1.18628697]]
Network Output[0.75012739 0.52063163]
Random sample 1
Input Vector:[[0.0306071 ]
 [0.3544189 ]
 [0.14550207]
 [0.48582586]]
Network Output[0.75339493 0.52359878]
Random sample 2
Input Vector:[[0.04125061]
 [0.04396288]
 [0.25034476]
 [0.42656215]]
Network Output[0.65807159 0.32050494]
Random sample 3
Input Vector:[[0.02861473]
 [0.34416234]
 [0.16994454]
 [0.54428905]]
Network Output[0.75856216 0.52359878]
Random sample 4
Input Vector:[[0.02648715]
 [0.25091022]
 [0.42129063]
 [1.13486349]]
Network Output[0.73247784 0.52359878]
Random sampling to find example input-outputs that violate desired conditions
    <input vector>:<output vector>
        number of attempted random samples thus far:1000
        number of attempted random samples thus far:2000
        number of attempted random samples thus far:3000
        number of attempted random samples thus far:4000
        number of attempted random samples thus far:5000
        number of attempted random samples thus far:6000
        number of attempted random samples thus far:7000
```

```
        number of attempted random samples thus far:8000
        number of attempted random samples thus far:9000
        number of attempted random samples thus far:10000
Exhausted pre-defined max number of random samples (10000) and did not find an
input-output instance violating conditions. Note that if violating instances
exist only on the boundary of the box (or exclusively on any other
lower-dimensional manifold) then the probabilty of sampling them (absent
computer-precision discretization) is zero
----
---

Details about Box Containing Maximum Magnitude Element
The box:[[-0.25       -0.21875   ]
 [-1.57079637 -1.17809725]
 [-1.5        -1.125     ]
 [-3.625      -2.71875   ]]
The output box from processing this input box:[[ 0.          1.3       ]
 [-0.52359878  0.36353551]]
The maximum magnitude element in the box:[-0.25   -1.57079637 -1.5    -3.625     ]
Network Output on the max magnitude vector[ 0.29719395 -0.52359878]
Randomly sampled vectors from the Box
Random sample 0
Input Vector:[[-0.22508777]
 [-1.21966414]
 [-1.30103316]
 [-3.28807632]]
Network Output[ 0.23279912 -0.52359878]
Random sample 1
Input Vector:[[-0.24893403]
 [-1.30772445]
 [-1.46945035]
 [-2.80106051]]
Network Output[ 0.13044161 -0.52359878]
Random sample 2
Input Vector:[[-0.24363742]
 [-1.2745214 ]
 [-1.18206541]
 [-3.24698956]]
Network Output[ 0.23307063 -0.52359878]
Random sample 3
Input Vector:[[-0.24799259]
 [-1.22984756]
 [-1.29598279]
 [-3.10421562]]
Network Output[ 0.16383254 -0.52359878]
```

```
Random sample 4
Input Vector:[[-0.2286421 ]
 [-1.34028381]
 [-1.16606158]
 [-3.12043877]]
Network Output[ 0.29359957 -0.52359878]
Random sampling to find example input-outputs that violate desired conditions
    <input vector>:<output vector>
        number of attempted random samples thus far:1000
        number of attempted random samples thus far:2000
        number of attempted random samples thus far:3000
        number of attempted random samples thus far:4000
        number of attempted random samples thus far:5000
        number of attempted random samples thus far:6000
        number of attempted random samples thus far:7000
        number of attempted random samples thus far:8000
        number of attempted random samples thus far:9000
        number of attempted random samples thus far:10000
Exhausted pre-defined max number of random samples (10000) and did not find an
input-output instance violating conditions. Note that if violating instances
exist only on the boundary of the box (or exclusively on any other
lower-dimensional manifold) then the probabilty of sampling them (absent
computer-precision discretization) is zero
----
---

Random example boxes
Random example box number 0
[[0.         0.015625  ]
 [0.         0.09817477]
 [0.375      0.46875   ]
 [3.171875   3.3984375 ]]
Random example box number 1
[[0.21875    0.234375  ]
 [0.58904862 0.68722337]
 [0.09375    0.1875    ]
 [1.359375   1.8125    ]]
Random example box number 2
[[-0.15625    -0.140625  ]
 [-1.37444675 -1.27627206]
 [-1.125      -1.03125   ]
 [-3.3984375  -3.171875  ]]
Random example box number 3
[[0.15625    0.1875    ]
 [0.58904862 0.78539819]
```

```
 [0.1875     0.375     ]
 [1.8125     2.71875   ]]
Random example box number 4
[[0.1875     0.21875   ]
 [0.58904862 0.78539819]
 [1.125      1.3125    ]
 [0.90625    1.8125    ]]
---

^_^_^_
```

```
Network Output[ 1.3        -0.13778461]
Random sample 1
Input Vector:[[-0.03421316]
 [-1.25590905]
 [-1.4308713 ]
 [ 0.30212832]]
Network Output[ 0.89040032 -0.52359878]
Random sample 2
Input Vector:[[-0.21437917]
 [-0.38824054]
 [-0.52533499]
 [ 0.73086265]]
Network Output[ 1.05364332 -0.11236838]
Random sample 3
Input Vector:[[-0.17251105]
 [-0.75144278]
 [-0.35561661]
 [ 3.28407086]]
Network Output[1.3        0.03645632]
Random sample 4
Input Vector:[[-0.10258383]
 [-0.26252556]
 [-0.26974926]
 [ 0.97929408]]
Network Output[1.20274902 0.08768249]
---

Details about Box Containing Minimum Magnitude Element
The box:[[ 0.          0.25      ]
 [ 0.          1.57079637]
 [ 0.          1.5       ]
 [-3.625       0.        ]]
The output box from processing this input box:[[ 0.          1.3       ]
 [-0.52359878  0.52359878]]
The minimum magnitude element in the box:[0 0 0 0]
Network Output on the max magnitude vector:[0.69063734 0.15590869]
Randomly sampled vectors from the Box
Random sample 0
Input Vector:[[ 0.09900171]
 [ 0.30037325]
 [ 0.36157416]
 [-1.45026939]]
Network Output[0.61900876 0.52359878]
Random sample 1
Input Vector:[[ 0.16786571]
```

```
 [ 1.03910936]
 [ 1.20251545]
 [-1.90691314]]
Network Output[0.44773573 0.52359878]
Random sample 2
Input Vector:[[ 0.10073722]
 [ 0.89397851]
 [ 0.17937899]
 [-2.87264336]]
Network Output[0.36949025 0.52359878]
Random sample 3
Input Vector:[[ 0.08525273]
 [ 0.9998535 ]
 [ 1.18679433]
 [-2.14454852]]
Network Output[0.39238909 0.52359878]
Random sample 4
Input Vector:[[ 0.11185426]
 [ 0.61133377]
 [ 0.72826031]
 [-2.39781423]]
Network Output[0.45663165 0.52359878]
---

Details about Box Containing Maximum Magnitude Element
The box:[[-0.25        0.        ]
 [-1.57079637  0.        ]
 [-1.5         0.        ]
 [ 0.          3.625     ]]
The output box from processing this input box:[[ 0.          1.3       ]
 [-0.52359878  0.52359878]]
The maximum magnitude element in the box:[-0.25  -1.57079637 -1.5   3.625 ]
Network Output on the max magnitude vector[ 1.3        -0.27763788]
Randomly sampled vectors from the Box
Random sample 0
Input Vector:[[-0.08151762]
 [-0.22595328]
 [-0.6369474 ]
 [ 1.34948148]]
Network Output[1.3        0.27047021]
Random sample 1
Input Vector:[[-0.20597505]
 [-1.26362499]
 [-0.30693095]
 [ 2.52959043]]
```

```
Network Output[ 1.3        -0.46811577]
Random sample 2
Input Vector:[[-0.11105243]
 [-0.77192256]
 [-1.31570779]
 [ 3.31636893]]
Network Output[1.3       0.1439838]
Random sample 3
Input Vector:[[-0.15533135]
 [-0.93522303]
 [-0.2541298 ]
 [ 1.38692722]]
Network Output[ 1.13683277 -0.46222329]
Random sample 4
Input Vector:[[-0.19181145]
 [-1.52614997]
 [-0.329221  ]
 [ 2.23380602]]
Network Output[ 1.00561995 -0.52359878]
---

Random example boxes
Random example box number 0
[[-0.203125   -0.1875    ]
 [-1.57079637 -1.47262156]
 [-0.46875    -0.375     ]
 [-3.171875   -2.9453125 ]]
Random example box number 1
[[-0.03125    -0.015625  ]
 [-0.78539819 -0.68722337]
 [-0.84375    -0.75      ]
 [-3.625      -3.3984375 ]]
Random example box number 2
[[0.         0.015625  ]
 [0.98174769 1.07992244]
 [0.1875     0.28125   ]
 [0.6796875  0.90625   ]]
Random example box number 3
[[0.         0.015625  ]
 [1.27627206 1.37444675]
 [1.03125    1.125     ]
 [0.2265625  0.453125  ]]
Random example box number 4
[[-0.234375   -0.21875   ]
 [-1.17809725 -1.12900984]
```

```
 [-0.28125    -0.1875    ]
 [-3.3984375  -3.171875  ]]
---

^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_^
V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V~V
Information about boxes that are true after extending the hypothesis to include
time derivatives but where not true prior to such weakening of the condition
we checked for
-------------------------------------------------------
number of such boxes:11166
Top 20 largest such boxes in descending order of size:
---
Size of box: 2.1353013161569834, percent size of input space: 0.06250000173922095, the box:
[[-0.25        0.        ]
 [ 0.          1.57079637]
 [-1.5         0.        ]
 [-3.625       0.        ]]
---
Size of box: 2.1353013161569834, percent size of input space: 0.06250000173922095, the box:
[[-0.25        0.        ]
 [ 0.          1.57079637]
 [-1.5         0.        ]
 [ 0.          3.625     ]]
---
Size of box: 2.1353013161569834, percent size of input space: 0.06250000173922095, the box:
[[-0.25        0.        ]
 [ 0.          1.57079637]
 [ 0.          1.5       ]
 [-3.625       0.        ]]
---
Size of box: 2.1353013161569834, percent size of input space: 0.06250000173922095, the box:
[[-0.25        0.        ]
 [ 0.          1.57079637]
 [ 0.          1.5       ]
 [ 0.          3.625     ]]
---
Size of box: 2.1353013161569834, percent size of input space: 0.06250000173922095, the box:
[[ 0.          0.25      ]
 [-1.57079637  0.        ]
 [-1.5         0.        ]
 [-3.625       0.        ]]
---
Size of box: 2.1353013161569834, percent size of input space: 0.06250000173922095, the box:
[[ 0.          0.25      ]
```

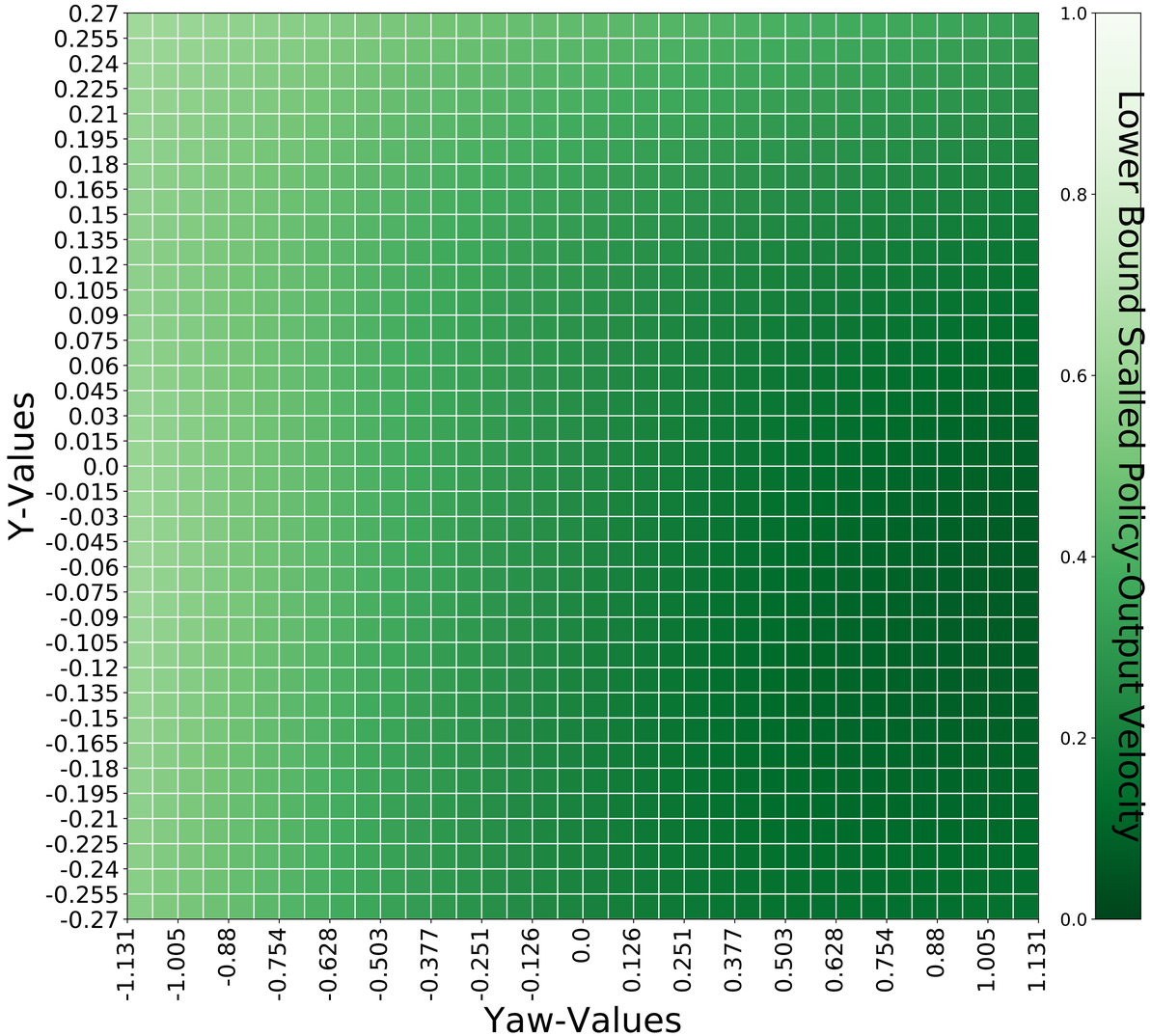
```
 [-1.57079637  0.        ]
 [-1.5         0.        ]
 [ 0.          3.625     ]]
---
Size of box: 2.1353013161569834, percent size of input space: 0.06250000173922095, the box:
[[ 0.          0.25      ]
 [-1.57079637  0.        ]
 [ 0.          1.5       ]
 [-3.625       0.        ]]
---
Size of box: 2.1353013161569834, percent size of input space: 0.06250000173922095, the box:
[[ 0.          0.25      ]
 [-1.57079637  0.        ]
 [ 0.          1.5       ]
 [ 0.          3.625     ]]
---
Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
[[-0.03125     0.        ]
 [-1.57079637 -1.37444675]
 [-1.5        -1.3125    ]
 [-3.171875   -2.71875   ]]
---
Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
[[-0.03125     0.        ]
 [-1.57079637 -1.37444675]
 [-1.5        -1.3125    ]
 [-3.625      -3.171875  ]]
---
Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
[[-0.03125     0.        ]
 [-1.57079637 -1.37444675]
 [-1.3125     -1.125     ]
 [-3.171875   -2.71875   ]]
---
Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
[[-0.03125     0.        ]
 [-1.57079637 -1.37444675]
 [-1.3125     -1.125     ]
 [-3.625      -3.171875  ]]
---
Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
[[-0.03125     0.        ]
 [-1.57079637 -1.37444675]
 [-1.125      -0.9375    ]
 [-3.171875   -2.71875   ]]
```

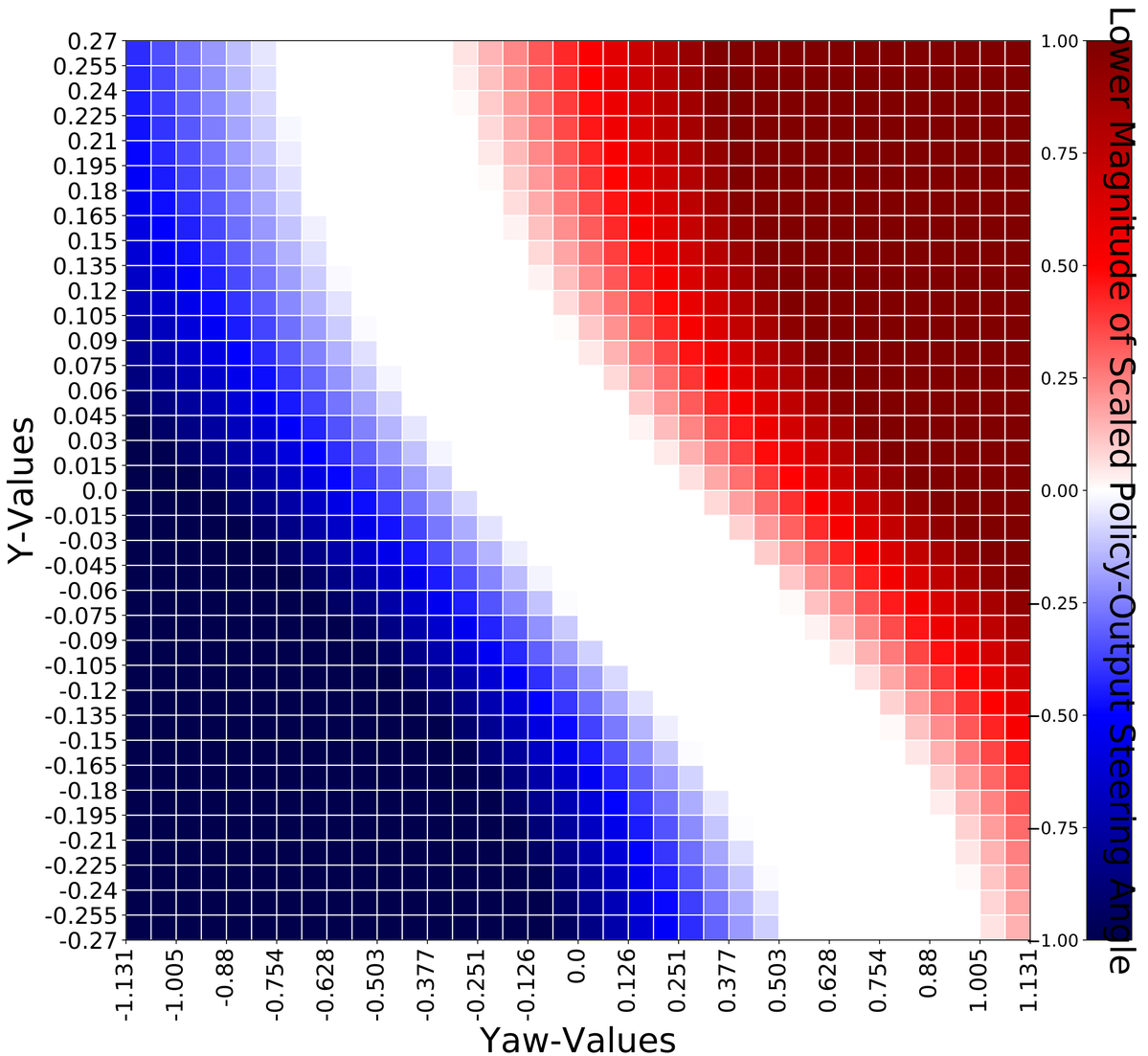
```
---
 Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
 [[-0.03125     0.        ]
  [-1.57079637 -1.37444675]
  [-1.125      -0.9375    ]
  [-3.625      -3.171875  ]]
---
 Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
 [[-0.03125     0.        ]
  [-1.57079637 -1.37444675]
  [-0.9375     -0.75      ]
  [-3.171875   -2.71875   ]]
---
 Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
 [[-0.03125     0.        ]
  [-1.57079637 -1.37444675]
  [-0.9375     -0.75      ]
  [-3.625      -3.171875  ]]
---
 Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
 [[-0.03125     0.        ]
  [-1.57079637 -1.37444675]
  [-0.75       -0.5625    ]
  [-3.171875   -2.71875   ]]
---
 Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
 [[-0.03125     0.        ]
  [-1.57079637 -1.37444675]
  [-0.75       -0.5625    ]
  [-3.625      -3.171875  ]]
---
 Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
 [[-0.03125     0.        ]
  [-1.57079637 -1.37444675]
  [-0.5625     -0.375     ]
  [-3.171875   -2.71875   ]]
---
 Size of box: 0.0005213139957049862, percent size of input space: 1.525879527713759e-05, the box:
 [[-0.03125     0.        ]
  [-1.57079637 -1.37444675]
  [-0.5625     -0.375     ]
  [-3.625      -3.171875  ]]


 ================
```

```
 Random example boxes
---
 [[-0.015625    0.        ]
  [-1.17809725 -1.07992244]
  [-1.5        -1.3125    ]
  [-3.171875   -2.71875   ]]
---
 [[-0.125      -0.109375  ]
  [-1.27627206 -1.17809725]
  [-0.9375     -0.84375   ]
  [-3.3984375  -3.171875  ]]
---
 [[0.21875    0.234375  ]
  [0.58904862 0.68722337]
  [1.125      1.21875   ]
  [3.171875   3.3984375 ]]
---
 [[0.234375   0.25      ]
  [1.22718465 1.27627206]
  [0.84375    0.9375    ]
  [3.3984375  3.625     ]]
---
 [[-0.125      -0.109375  ]
  [-1.57079637 -1.47262156]
  [-0.84375    -0.75      ]
  [-2.9453125  -2.71875   ]]
 ^-^-^-
```

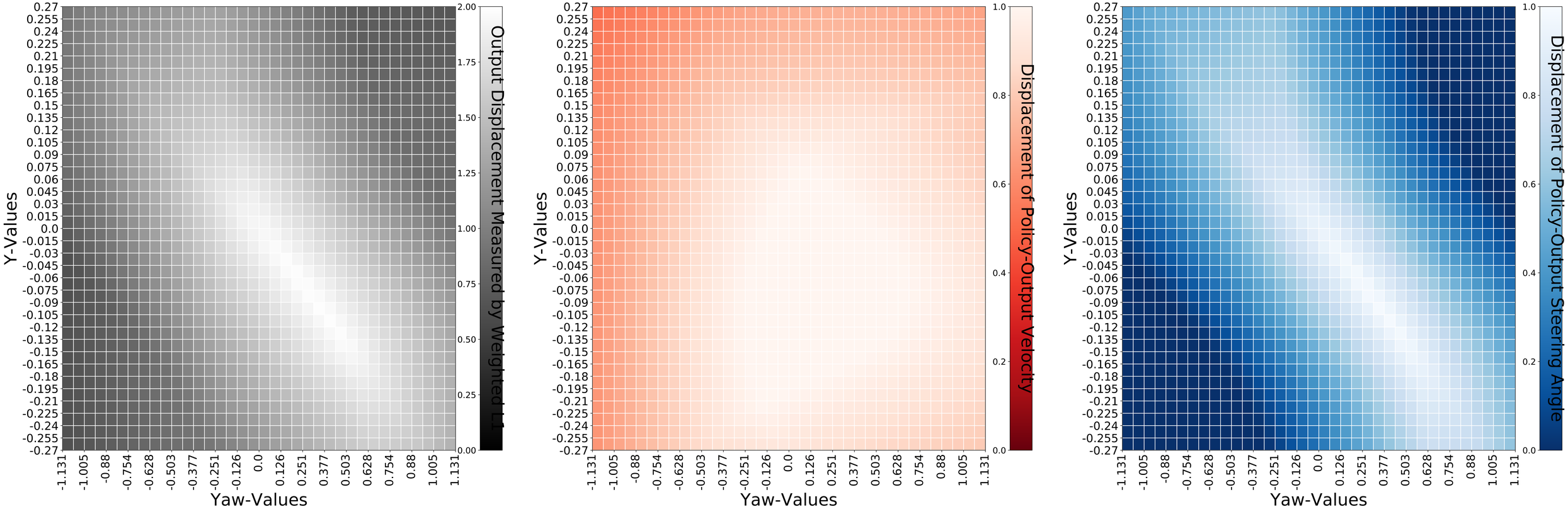
Line-Following Policy Results

Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0.046153, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 9460fd2e-e06d-4dbd-90b2-92f2893e816b
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_22_54_45_157692

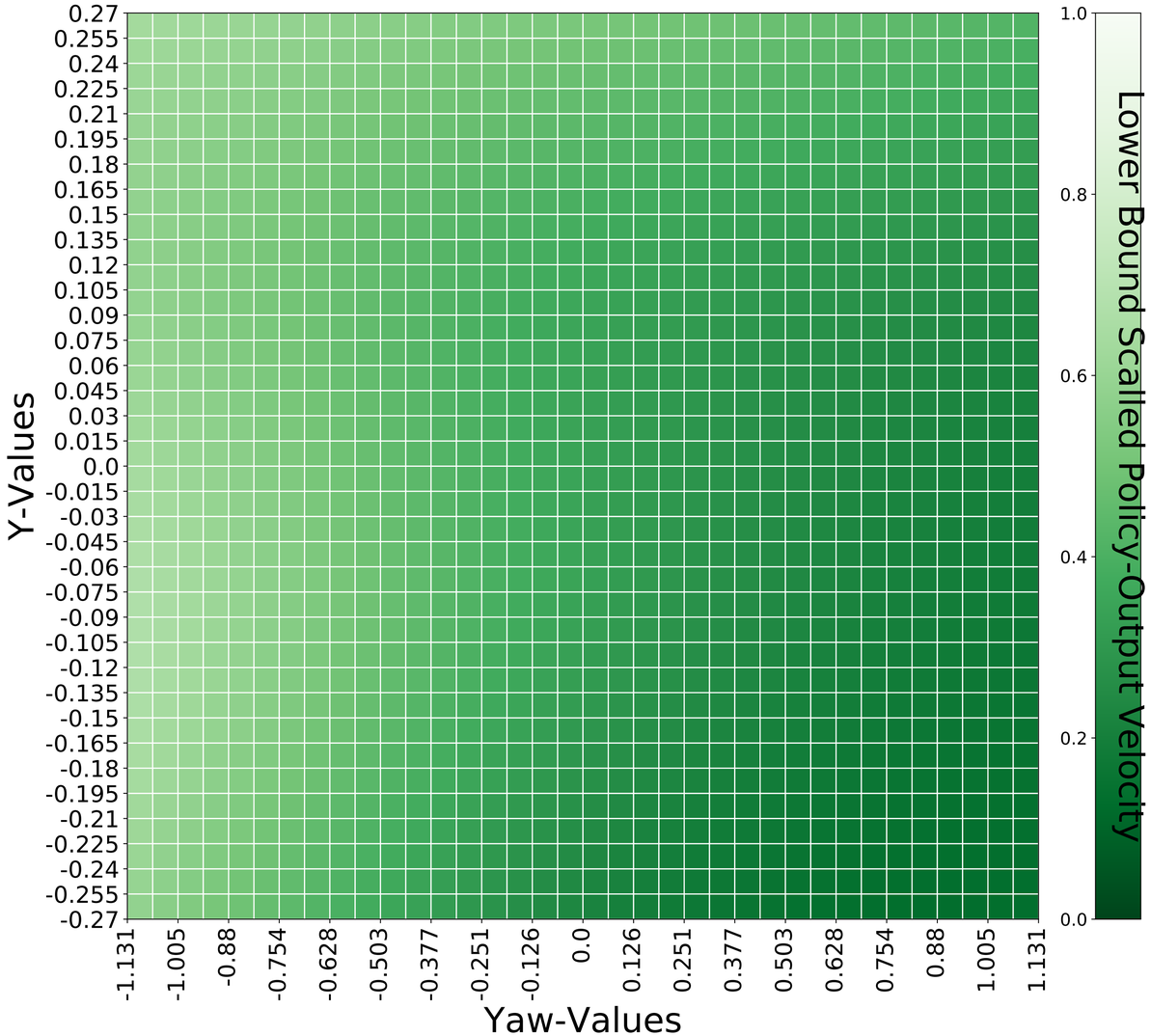


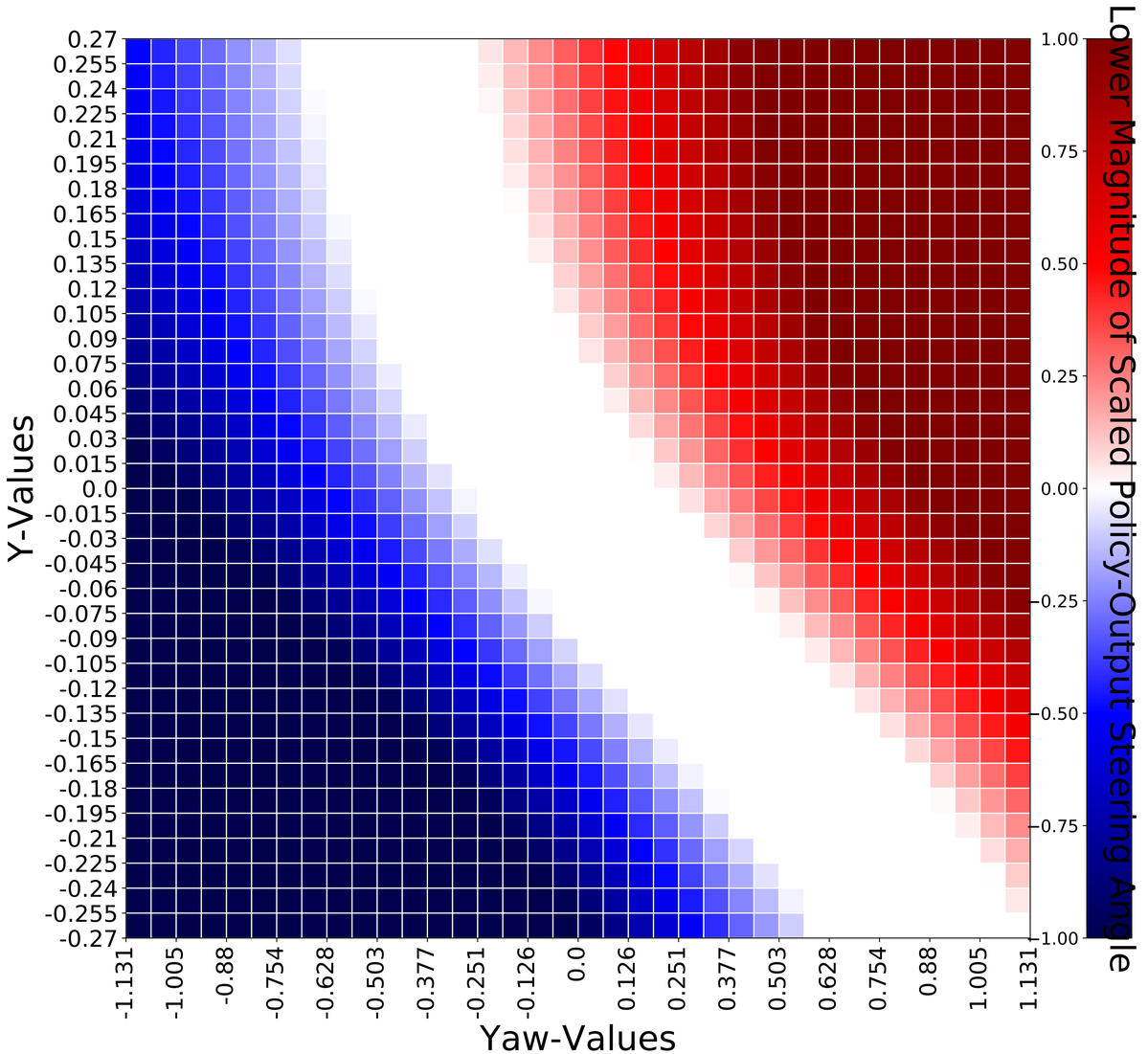


Line-Following Stability Results
xDot:0.700000, yDot:0.046153, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: f9ce36eb-fde6-4ac0-817b-d79a59048c50
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_22_53_47_400254

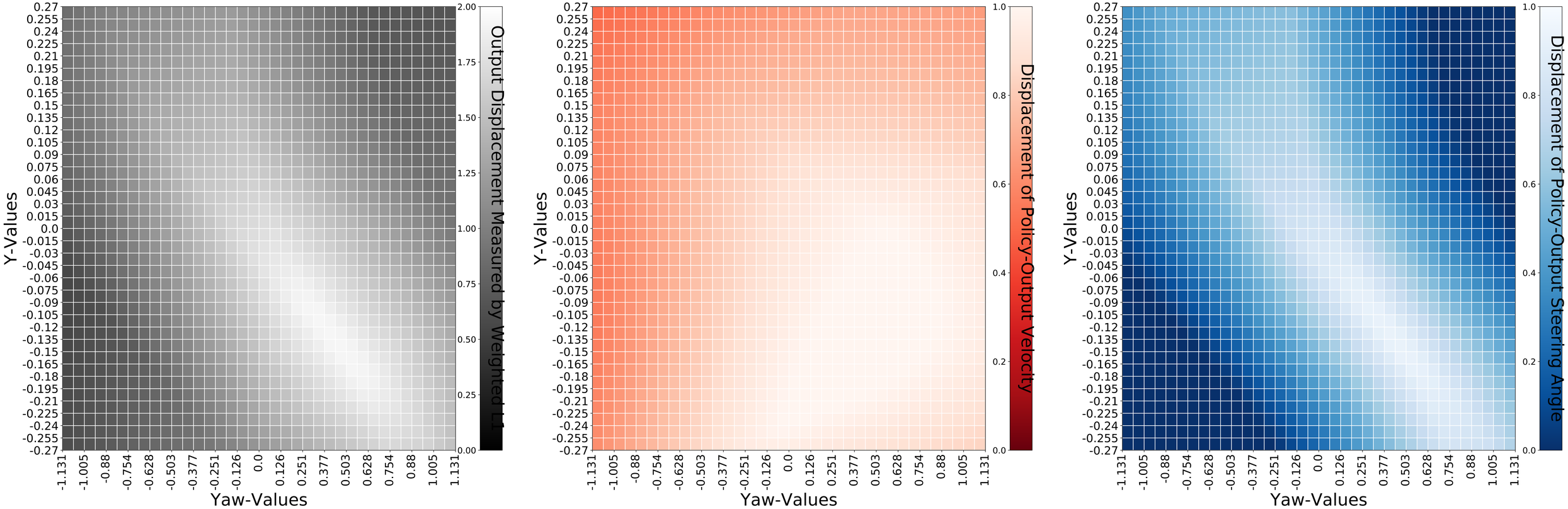


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0.046153, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 0cbc4944-ec5d-4693-87f6-035cefbc1046
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_22_56_38_153021

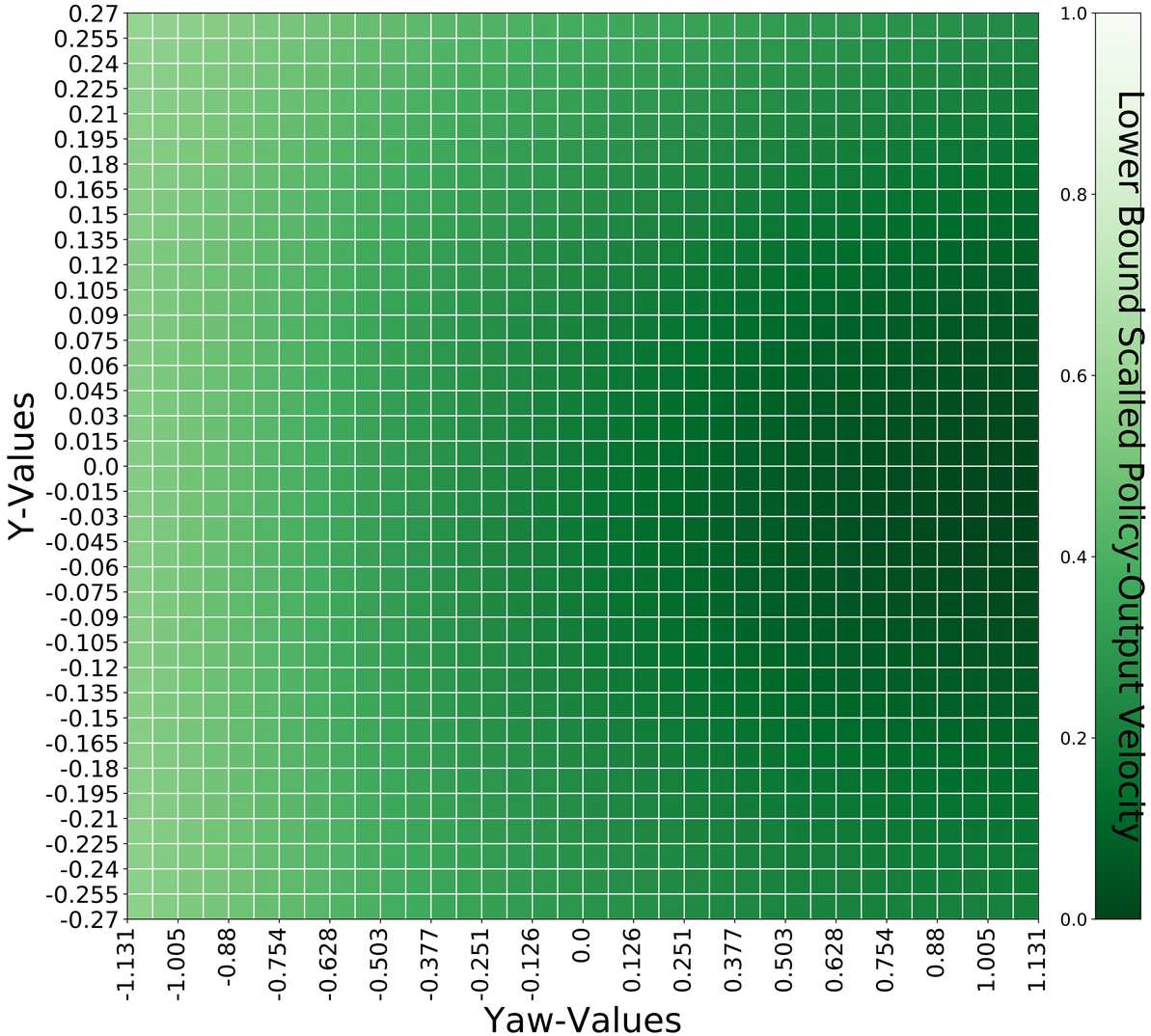


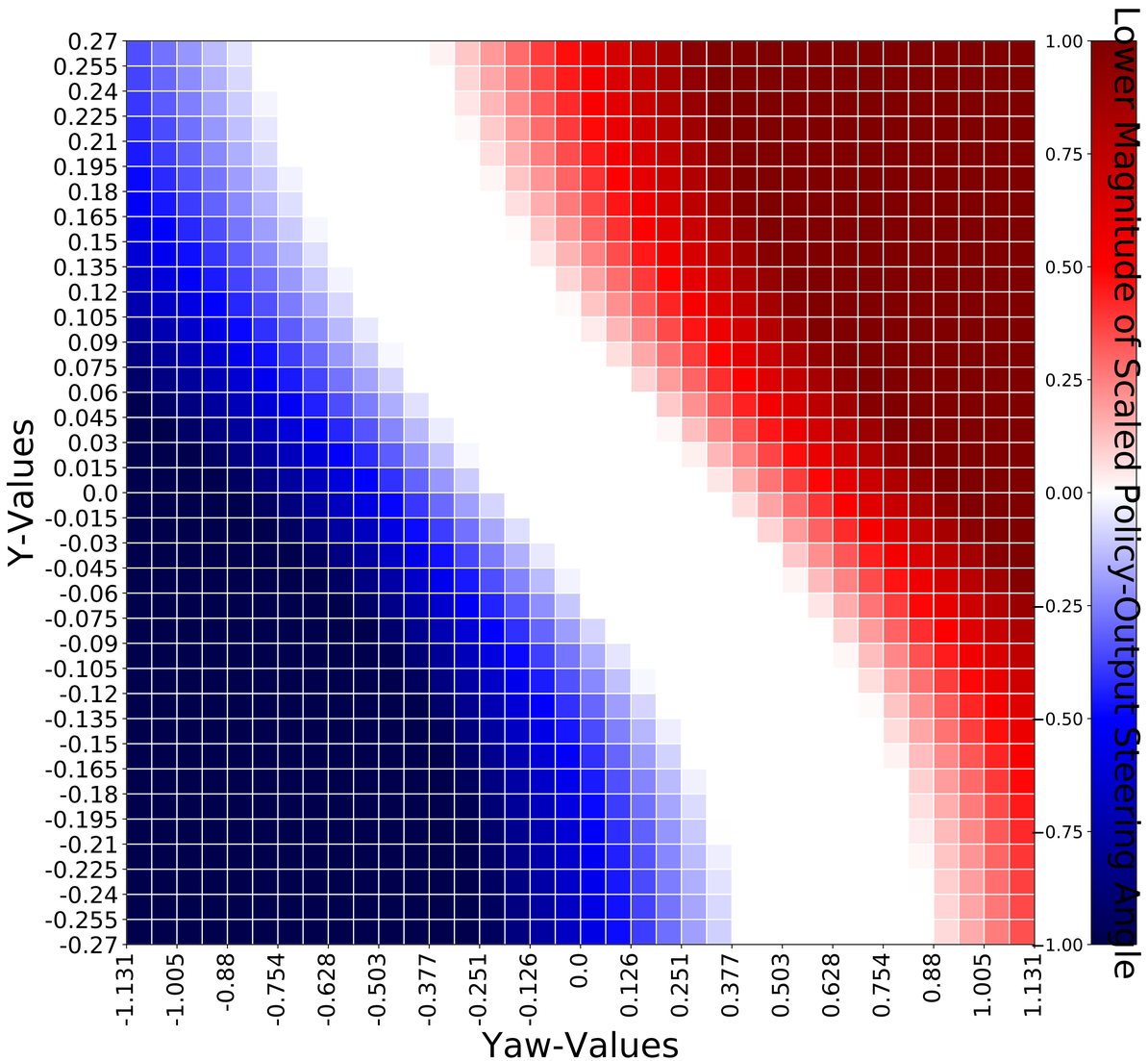


Line-Following Stability Results
xDot:0.700000, yDot:0.046153, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: b9b33d0b-6d02-4a4c-9577-e9f88a66fba8
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_22_55_37_072628

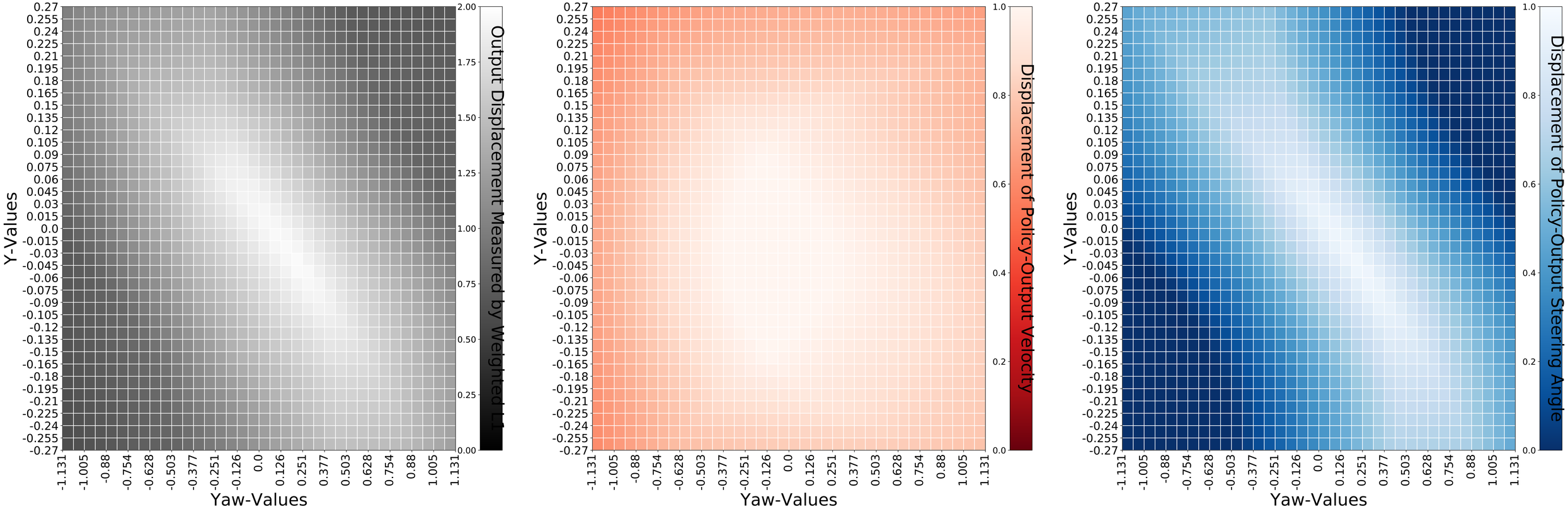


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0.046153, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 1630797e-e0ad-476c-85f2-fe3b5349b779
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_22_58_41_884142

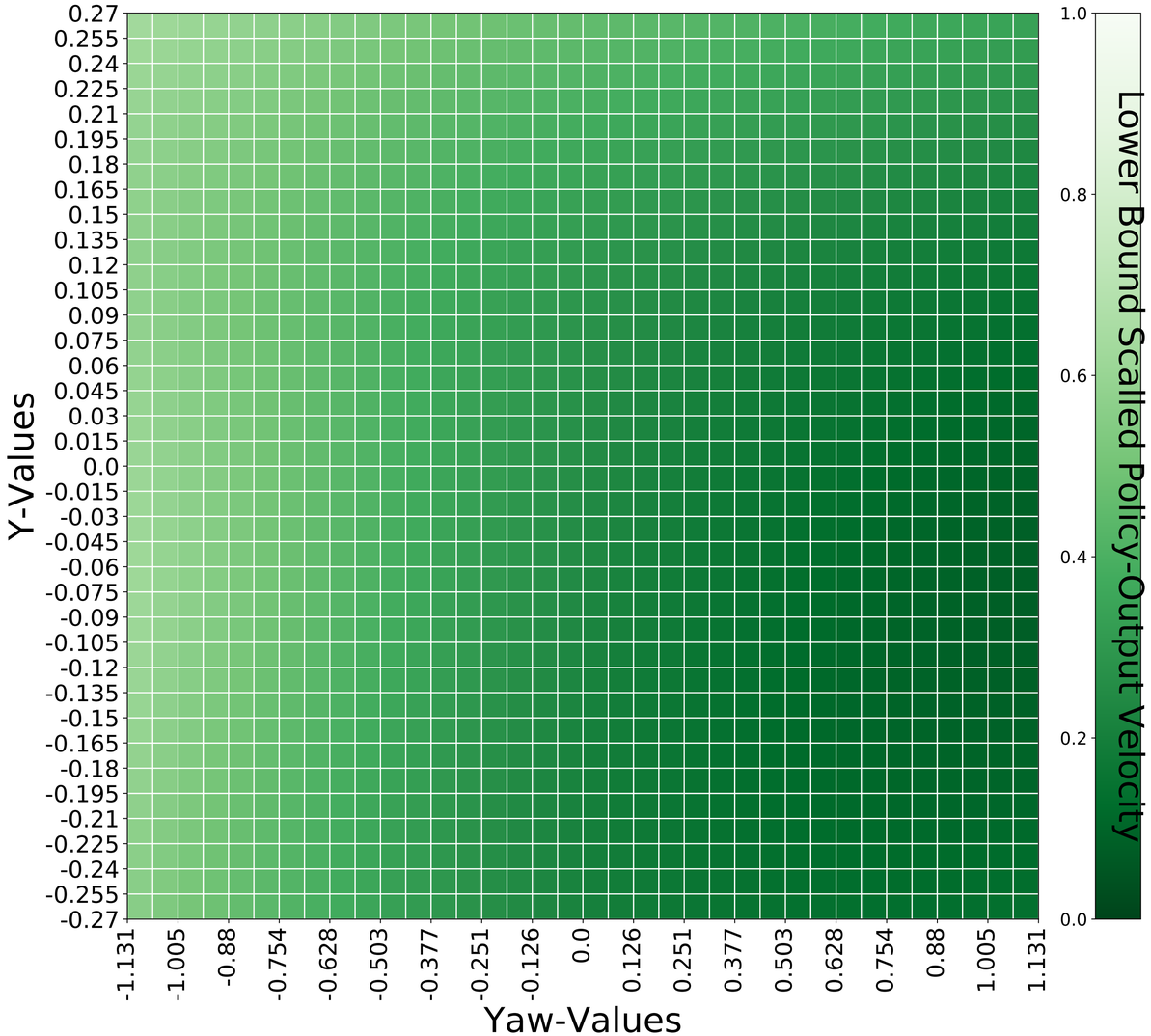


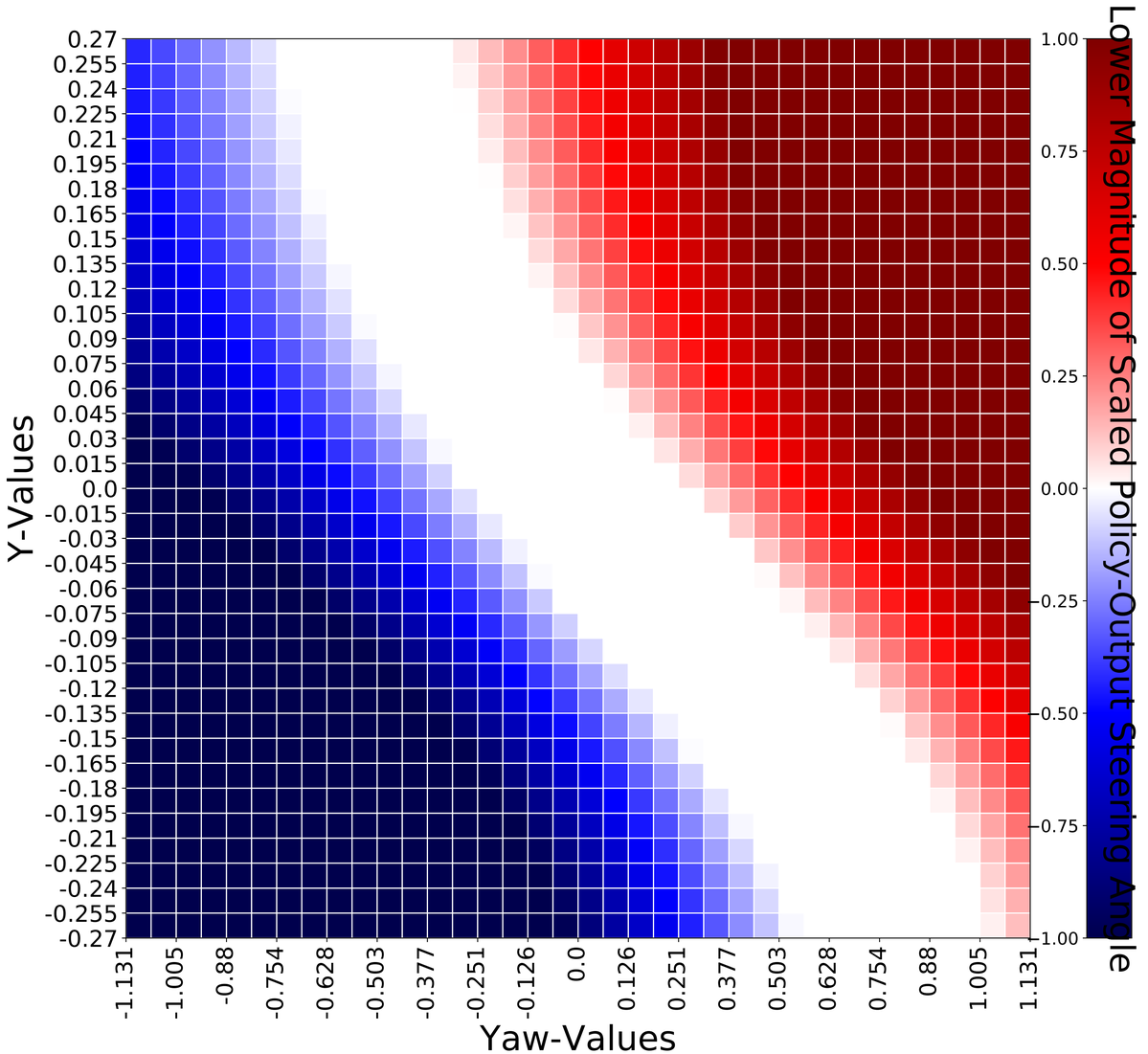


Line-Following Stability Results
xDot:0.700000, yDot:0.046153, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: be820b59-1758-4e8a-a722-29696c2d4918
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_22_57_38_639738

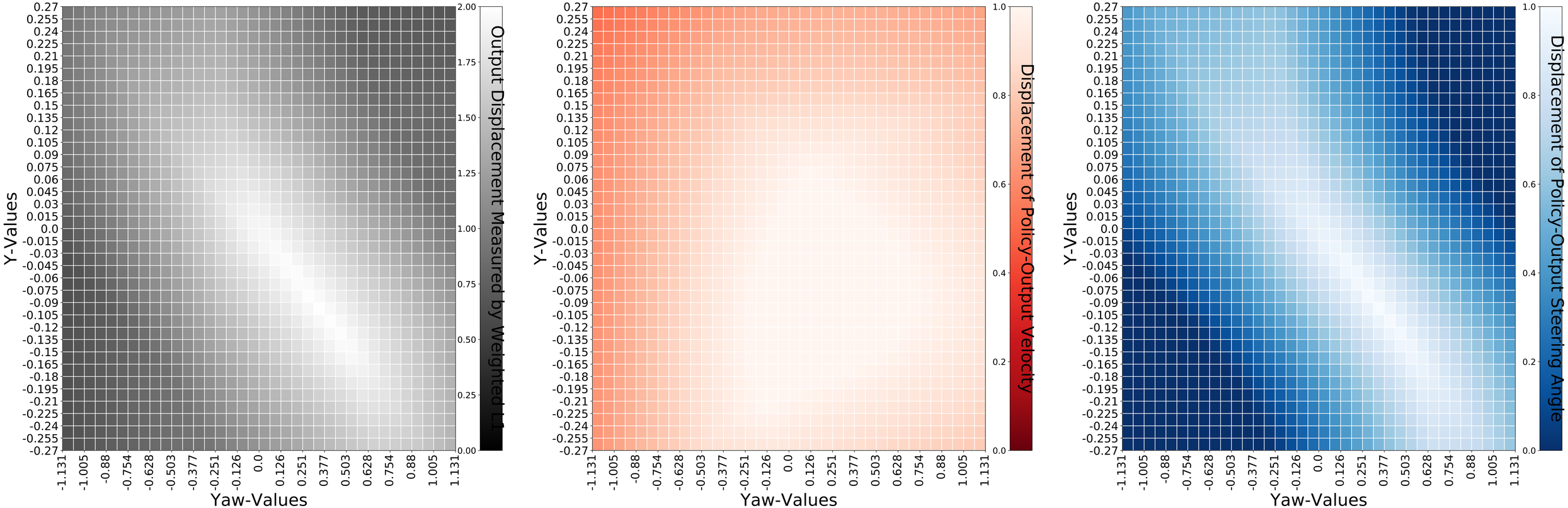


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0.046153, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 6f761544-f1a6-4903-a5a7-de2cd0b3f6b2
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_10_01_339938

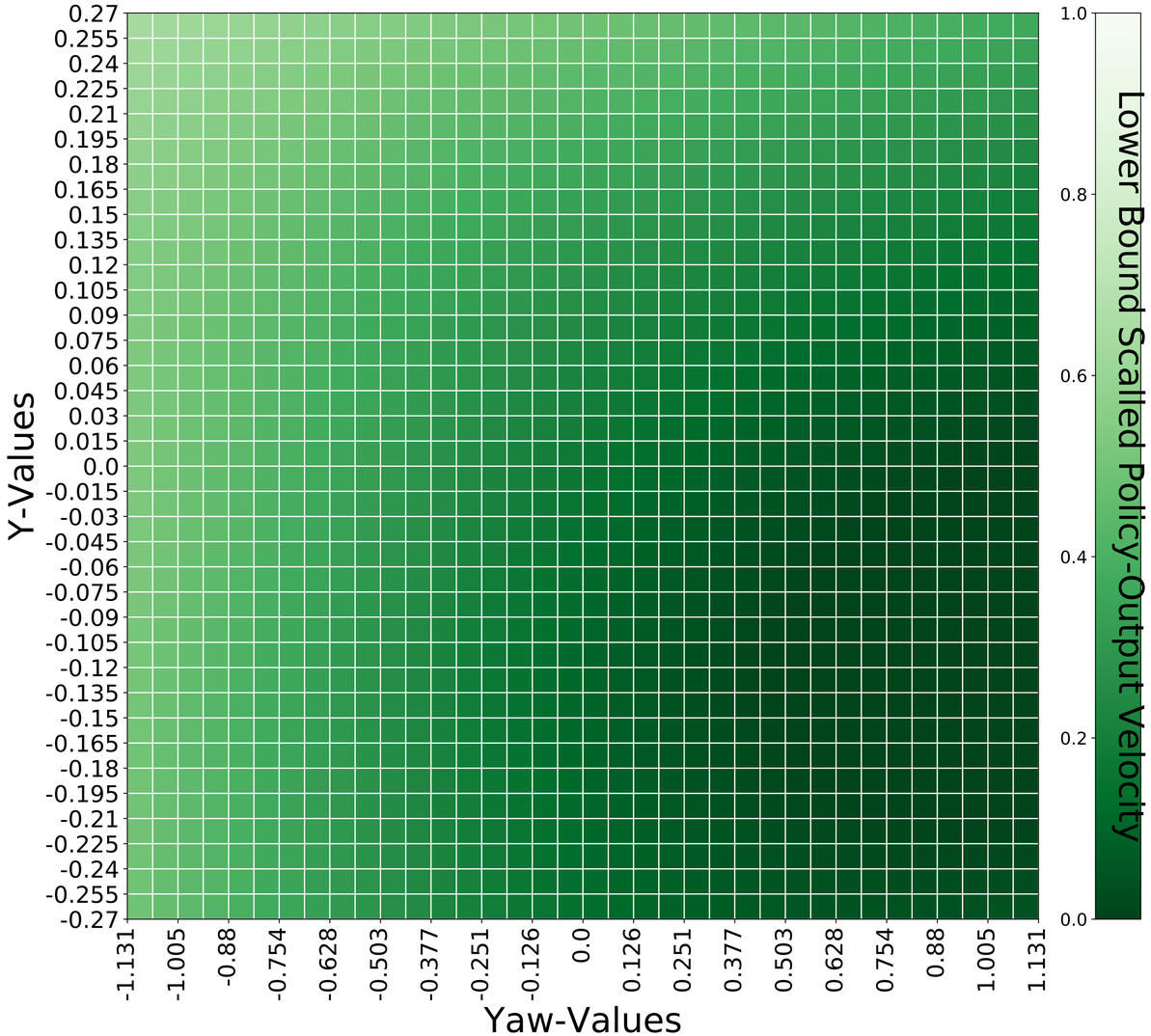


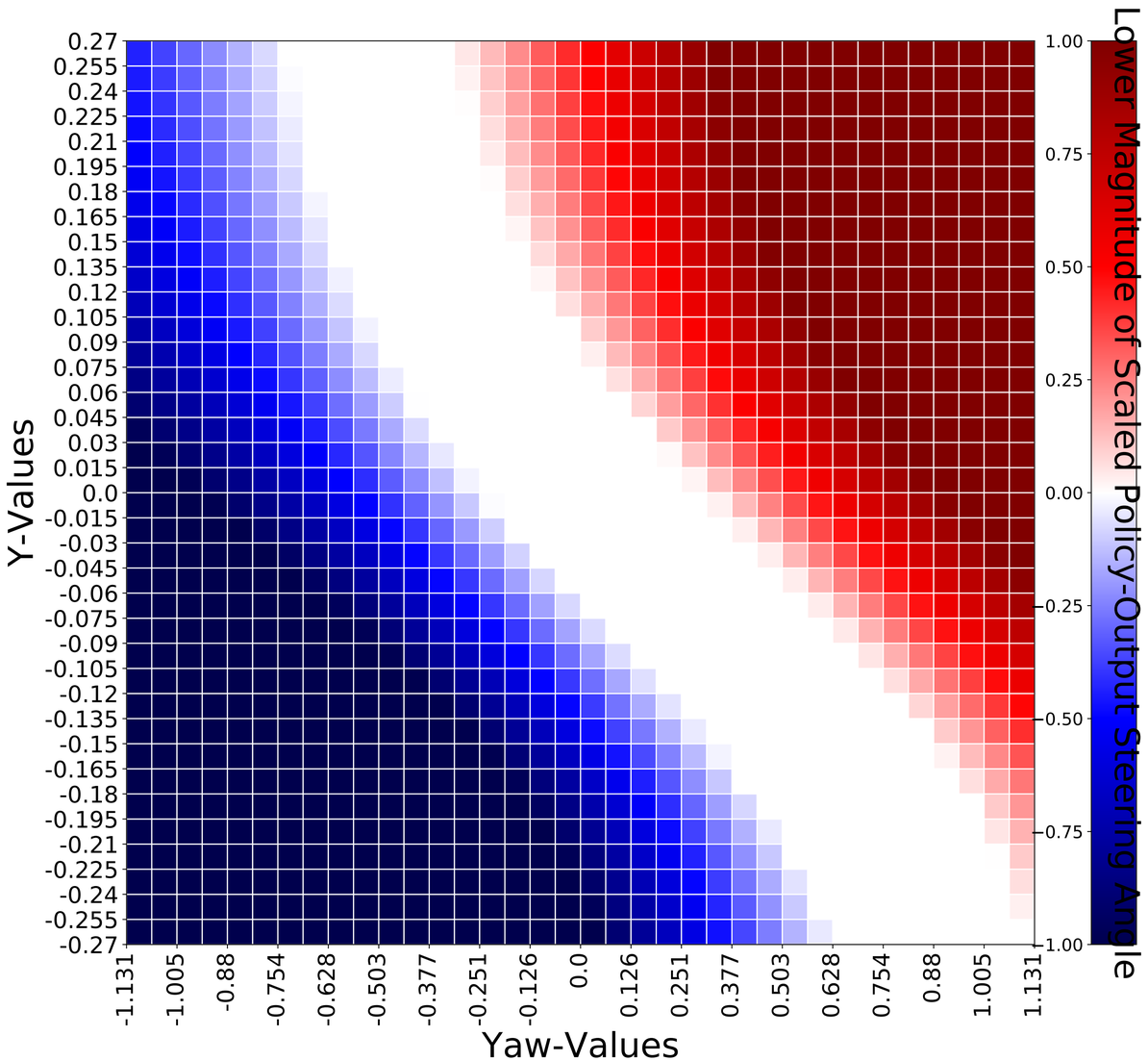


Line-Following Stability Results
xDot:0.700000, yDot:0.046153, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 358b89ed-9f4a-45ef-95b1-3622c7c0060a
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_09_09_301816

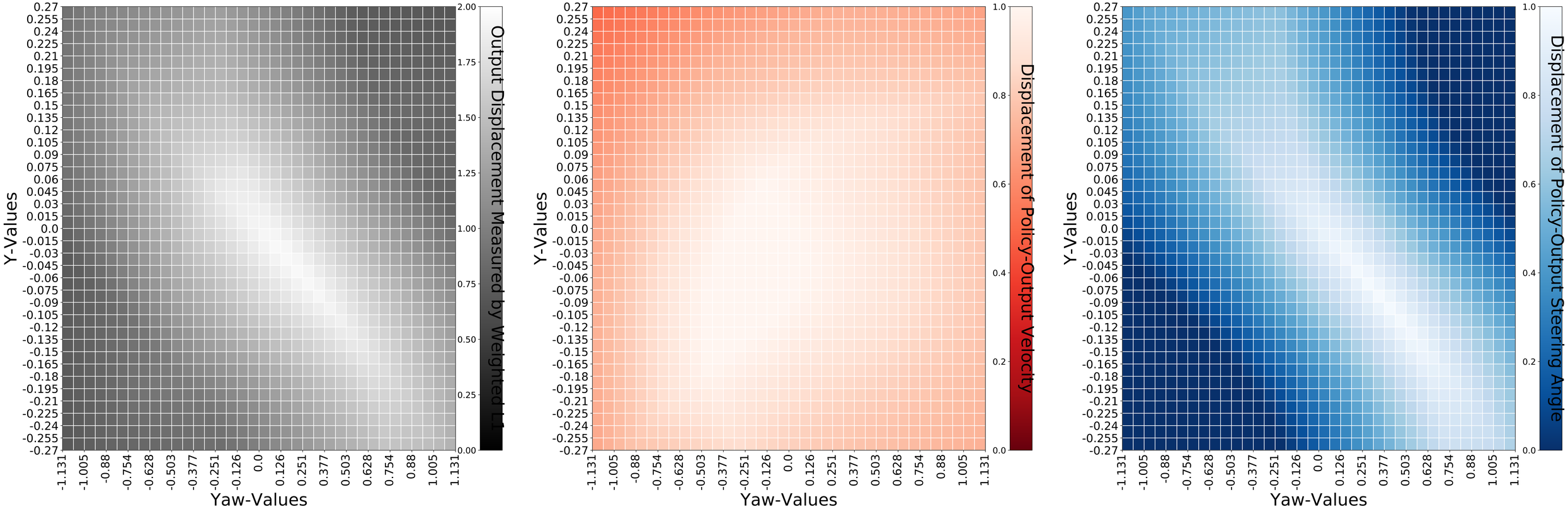


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:-0.32307, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 71040286-a08e-450b-9efd-d4baa5d995e6
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_11_55_939002

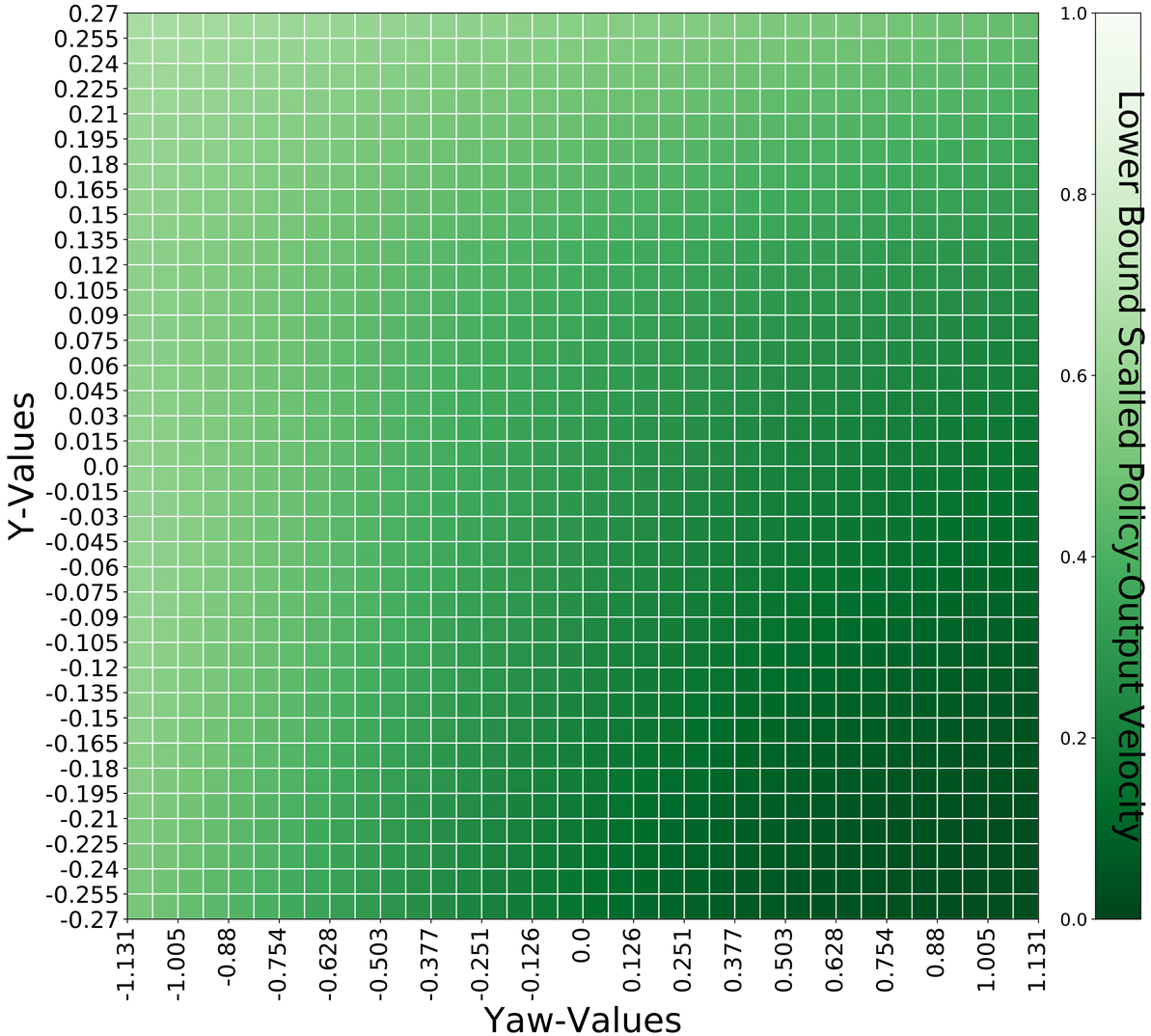


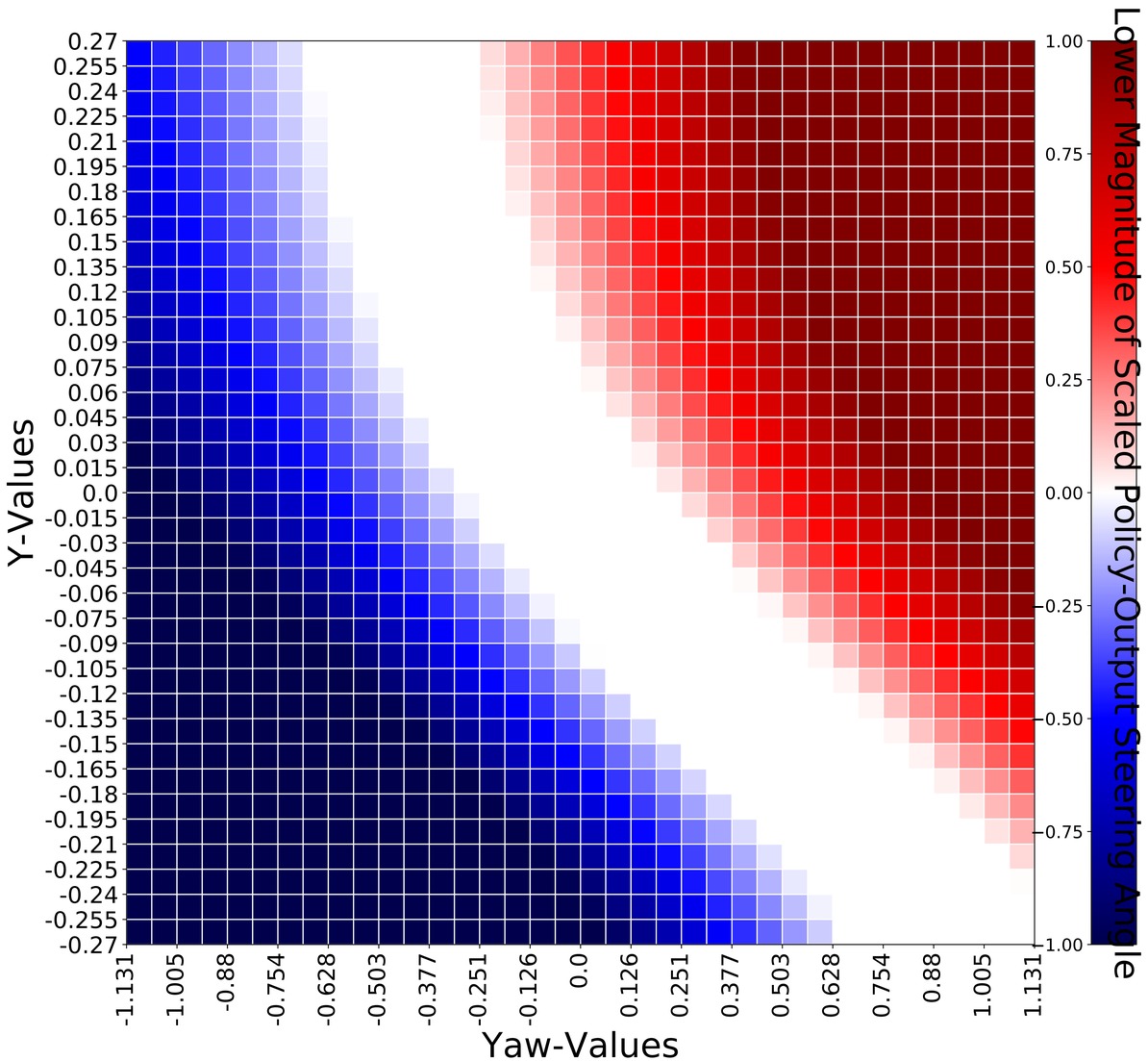


Line-Following Stability Results
xDot:0.700000, yDot:-0.32307, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: f5570486-6aa6-4c4e-ae7a-8dab053c37ed
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_11_01_713398

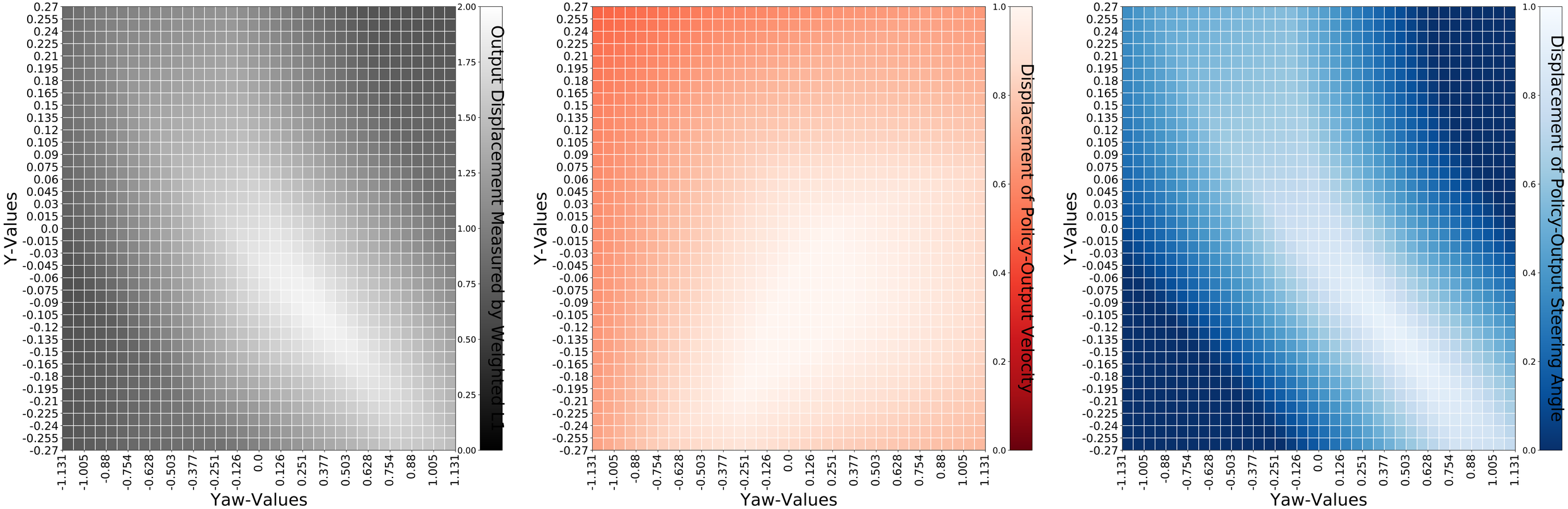


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:-0.32307, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: b0b74415-afc8-48c5-8706-153f8fe54bed
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_13_56_697800

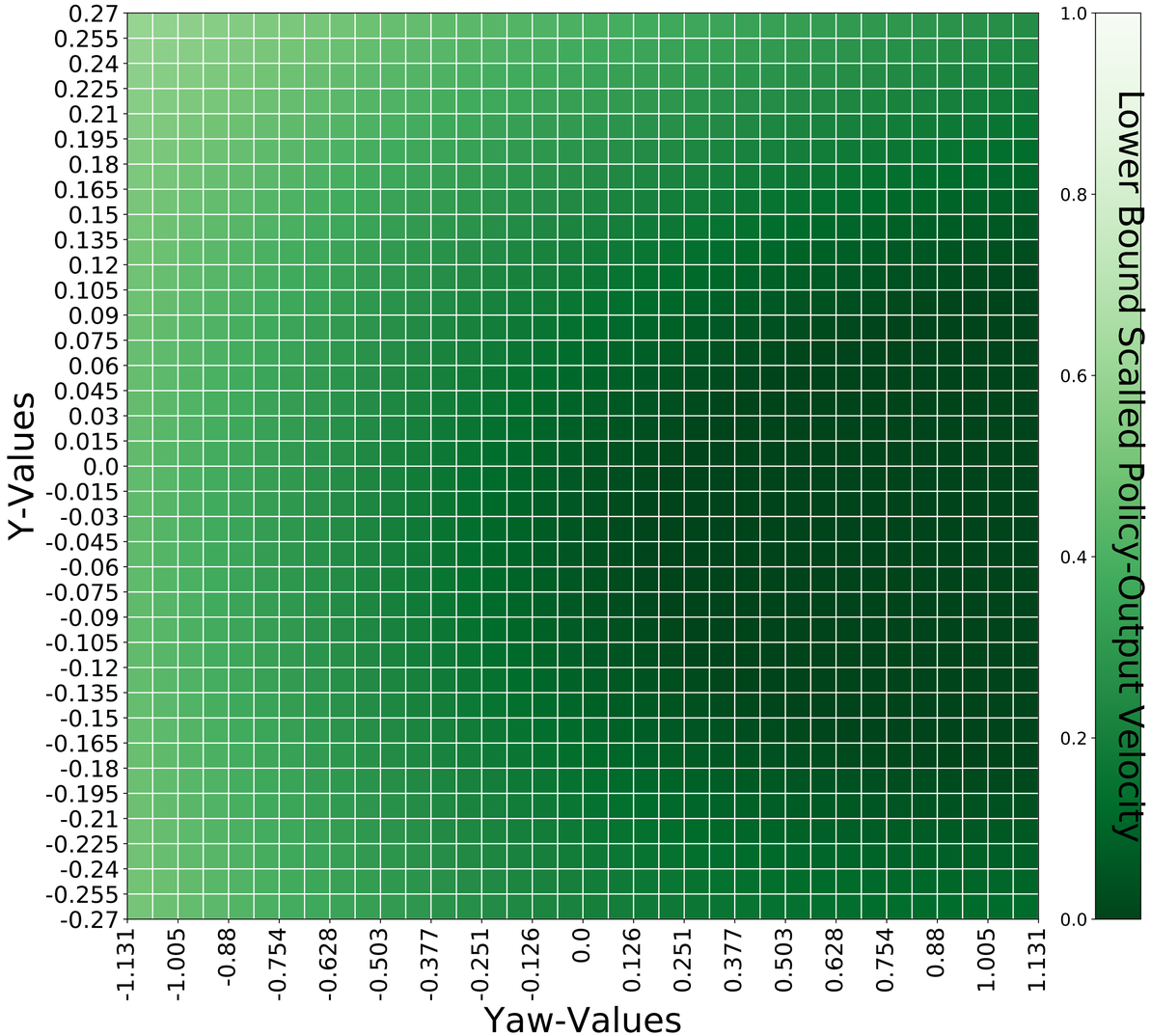


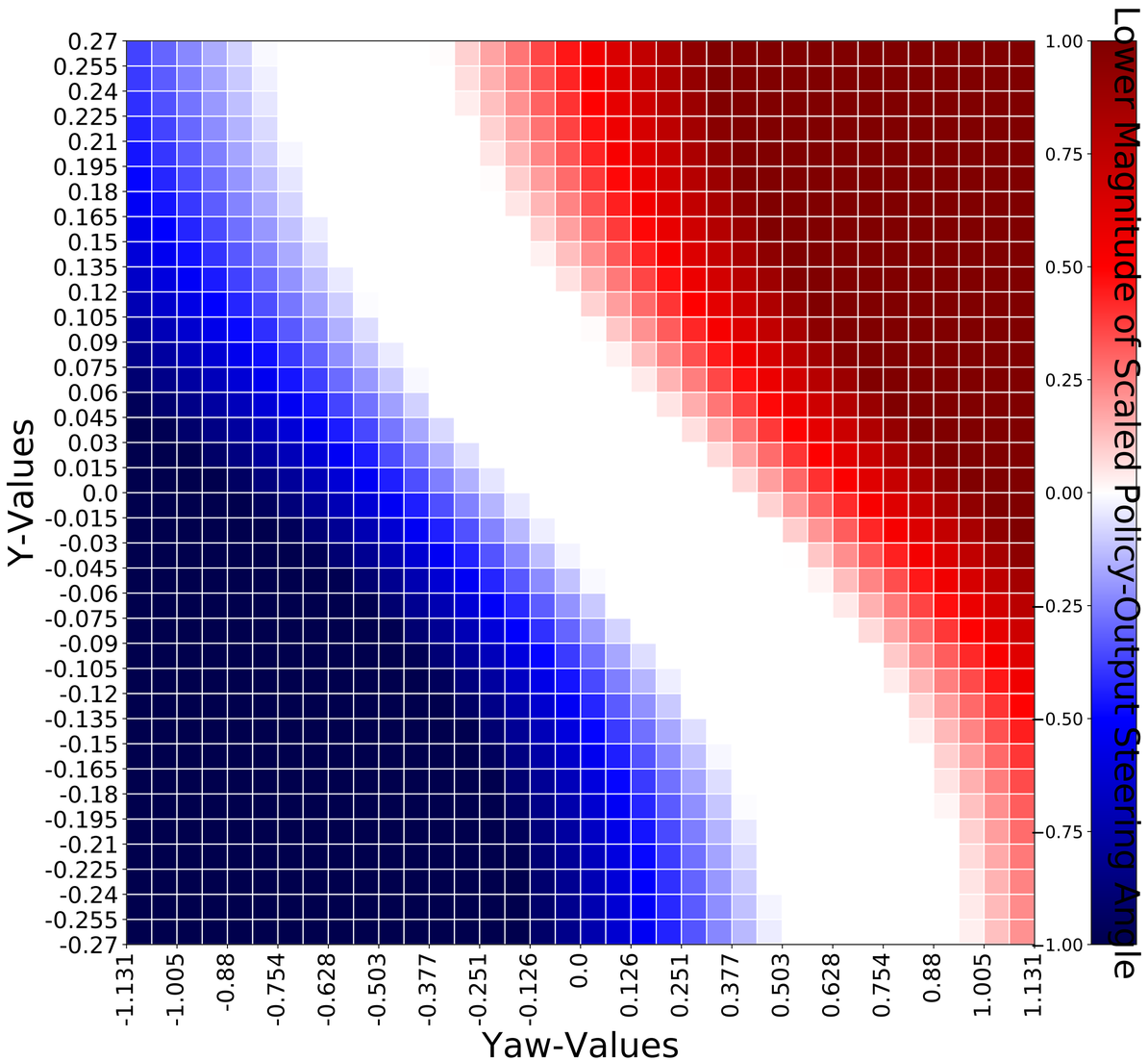


Line-Following Stability Results
xDot:0.700000, yDot:-0.32307, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: f6199108-53f1-4183-bbb3-77143c2018d6
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_12_52_899334

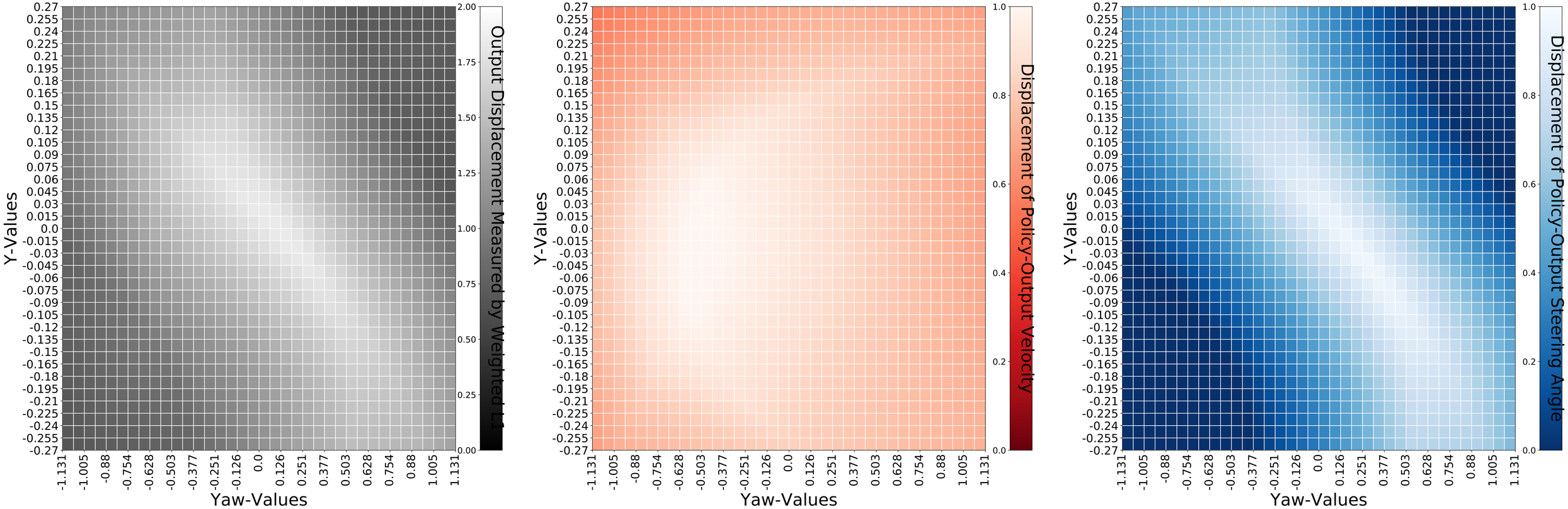


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:-0.32307, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 4c081eb7-739c-4ca6-a55e-df0852c8ca00
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_16_04_526536

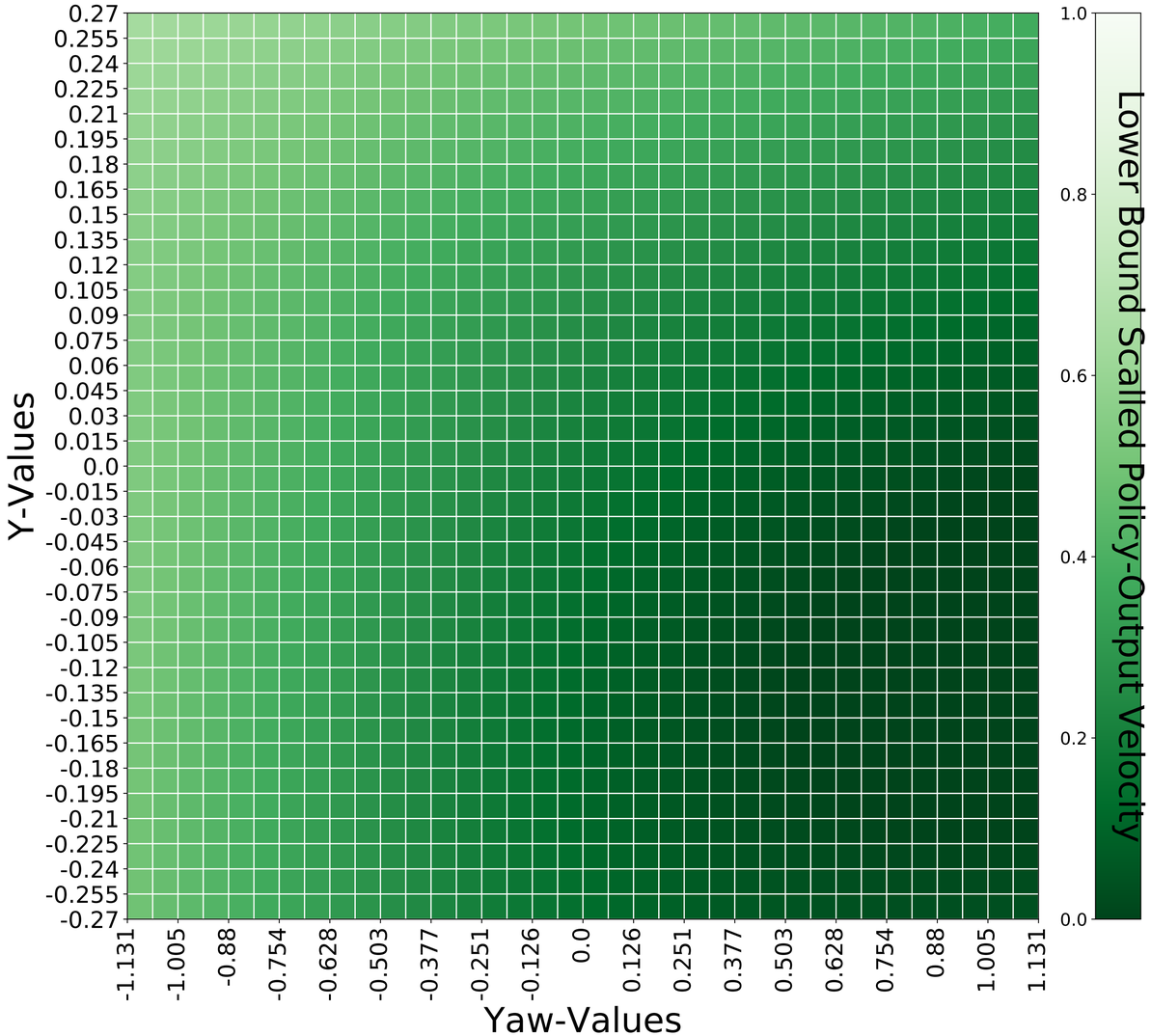


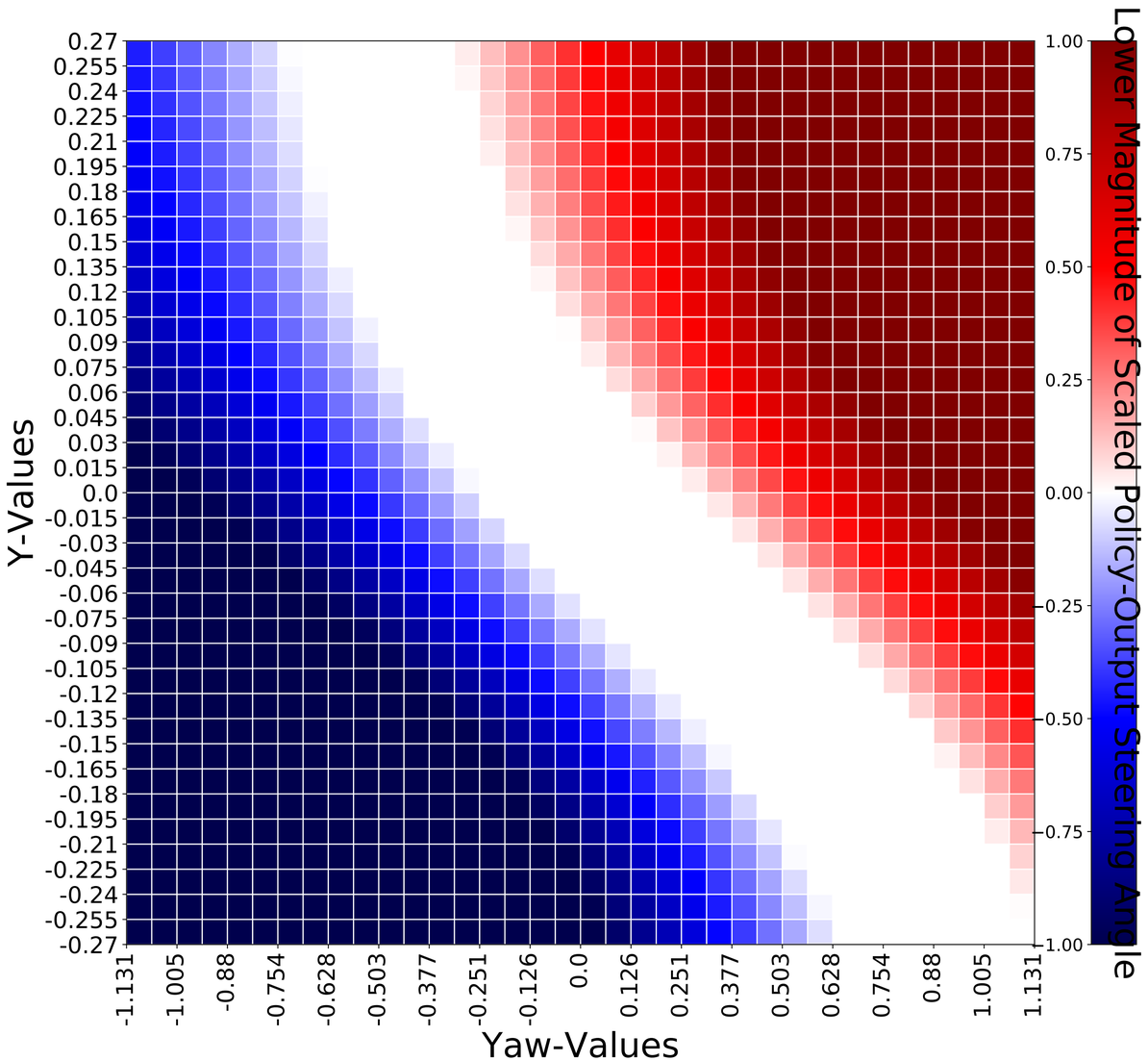


Line-Following Stability Results
xDot:0.700000, yDot:-0.32307, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: da499f61-b070-4ce0-9dcd-dee5973dc56b
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_14_59_644808

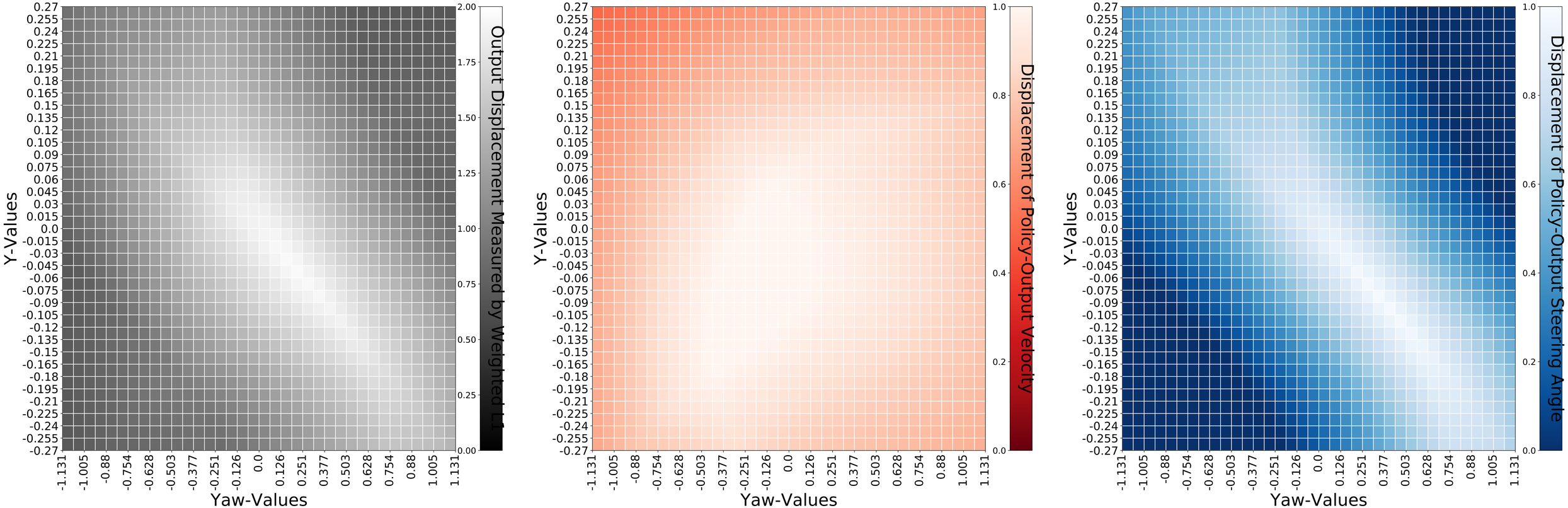


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:-0.32307, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 3d79c52b-6adf-450a-92ad-796db72569aa
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_28_45_172947

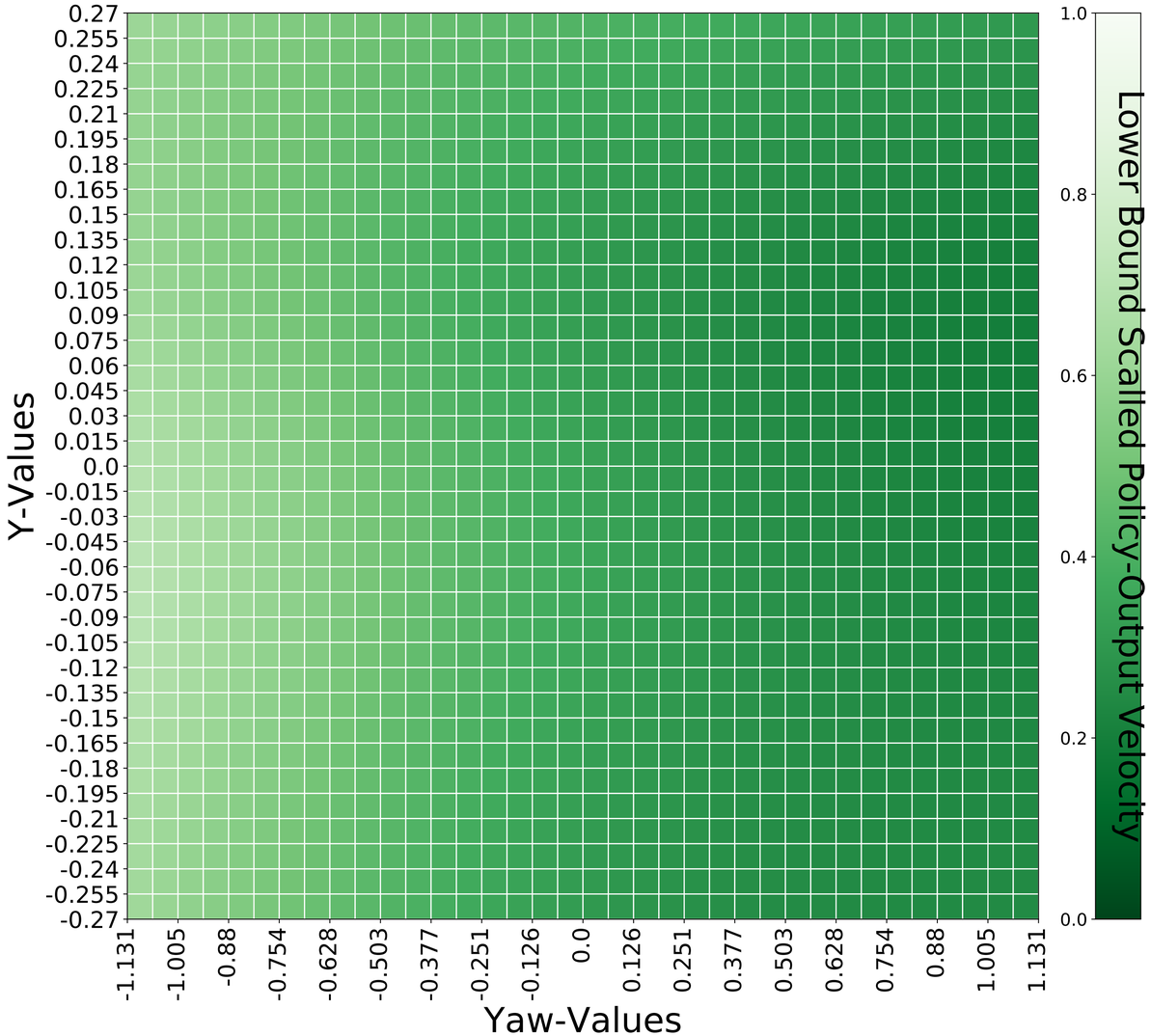


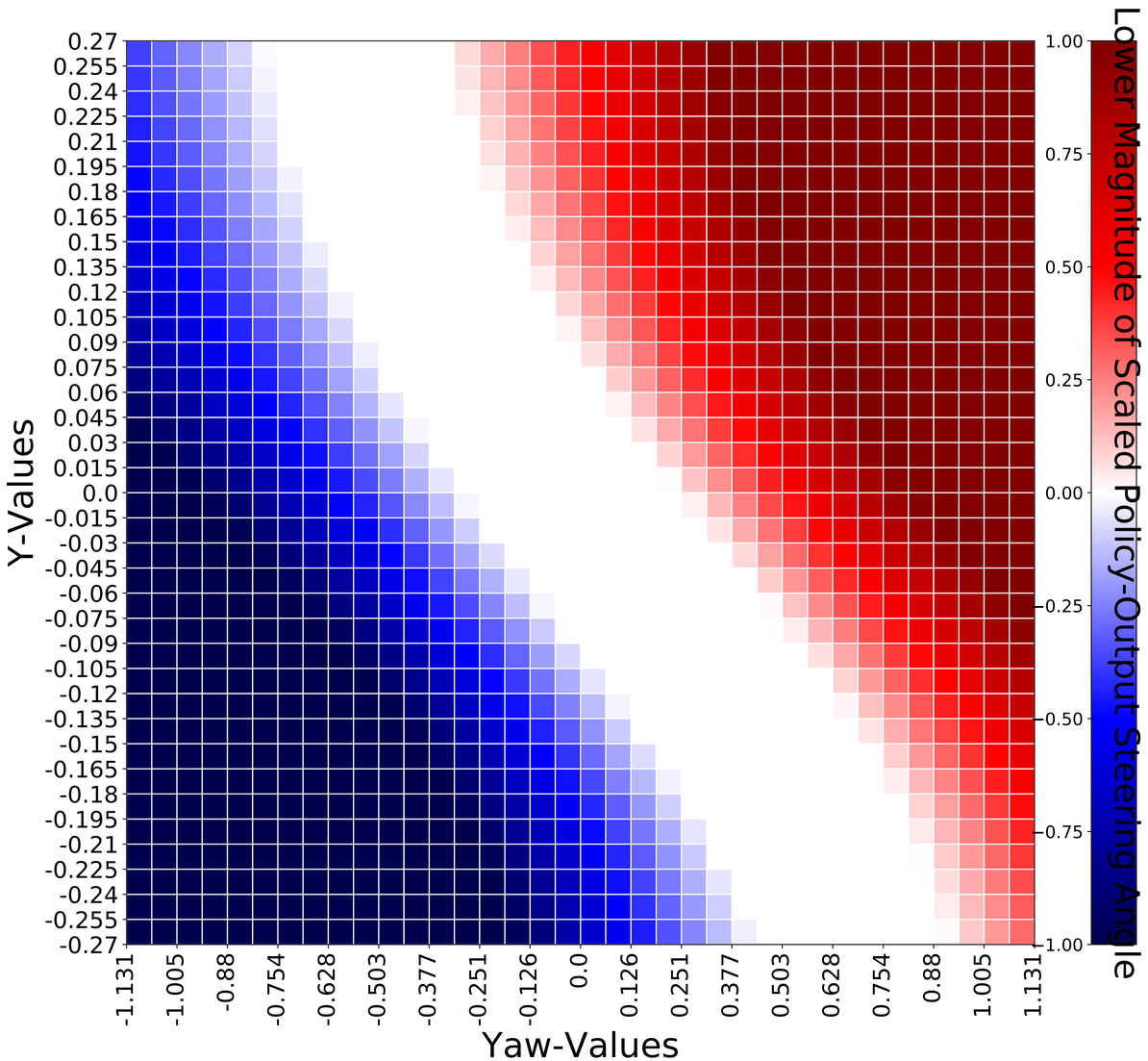


Line-Following Stability Results
xDot:0.700000, yDot:-0.32307, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 5e508f3c-28fc-47d8-83d9-8dcf6c9853a7
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_27_41_527526

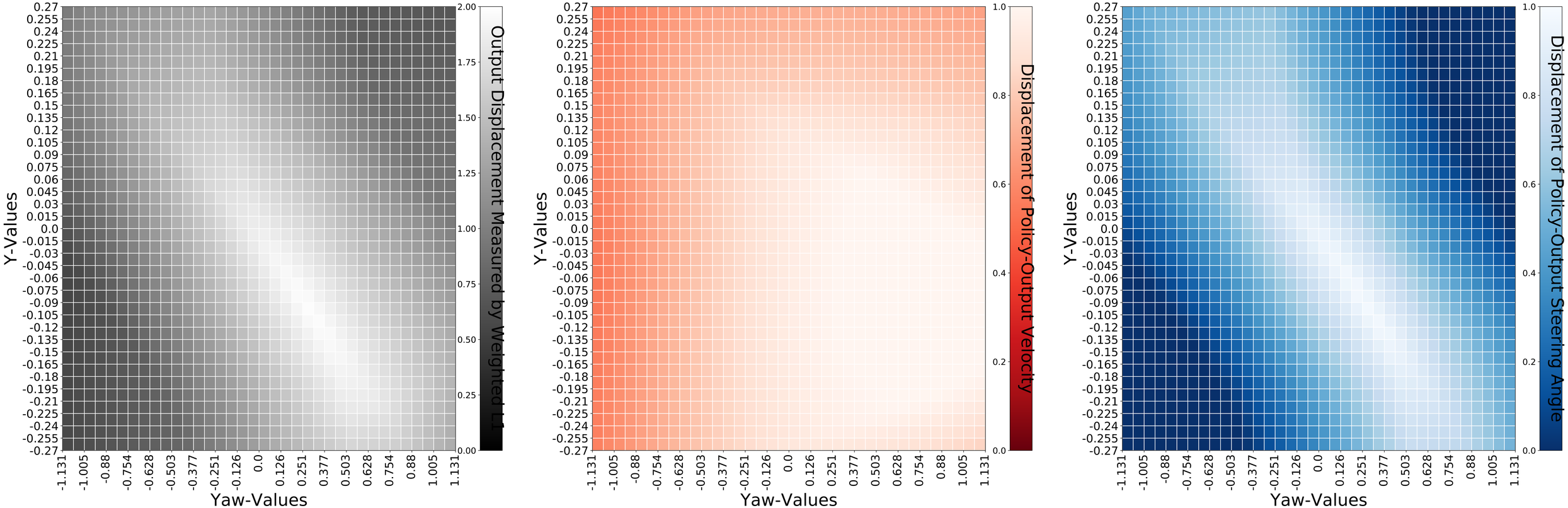


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0.415384, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 171a9646-0d31-4126-9103-7062aecdc1b6
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_30_51_471890

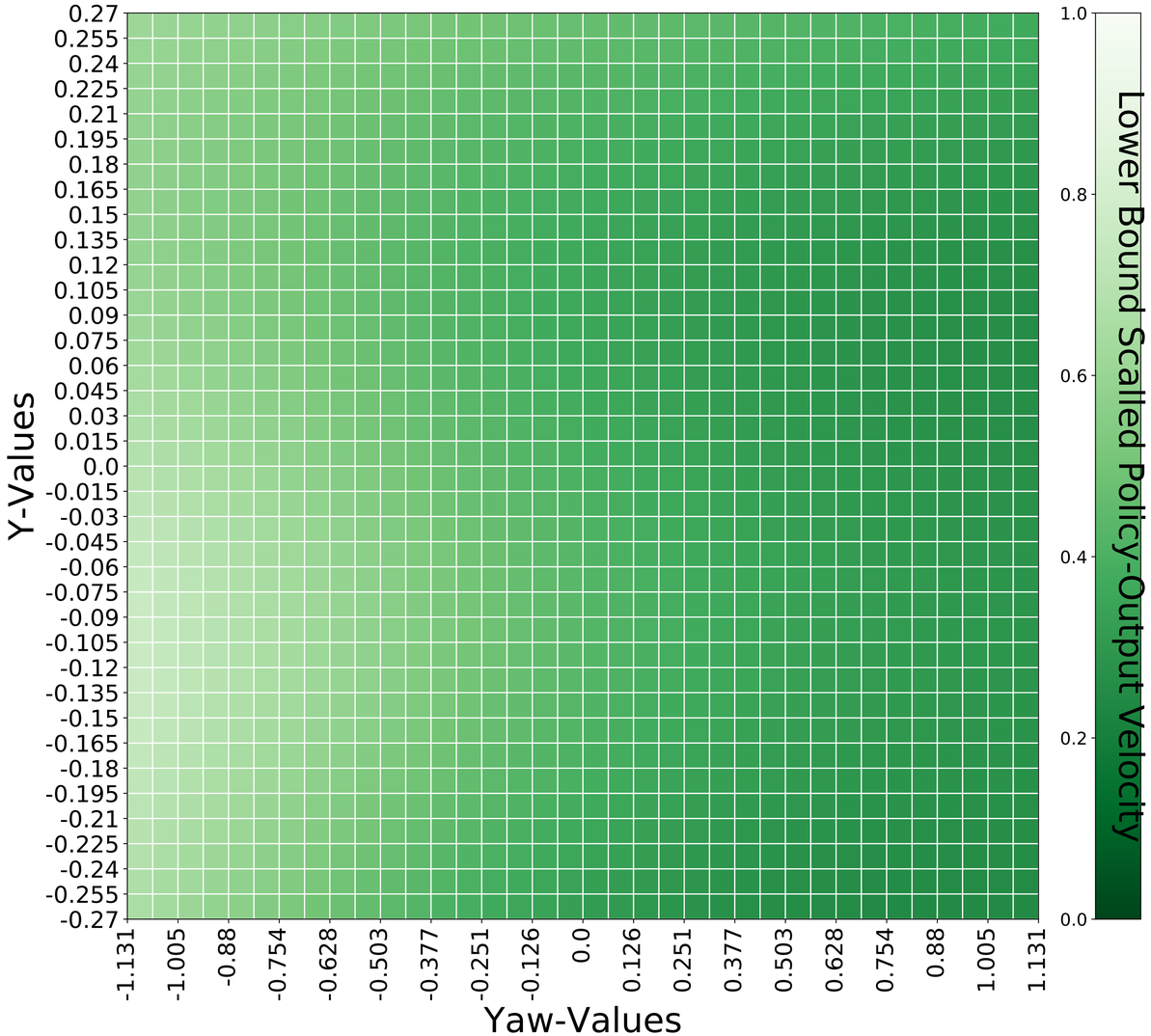


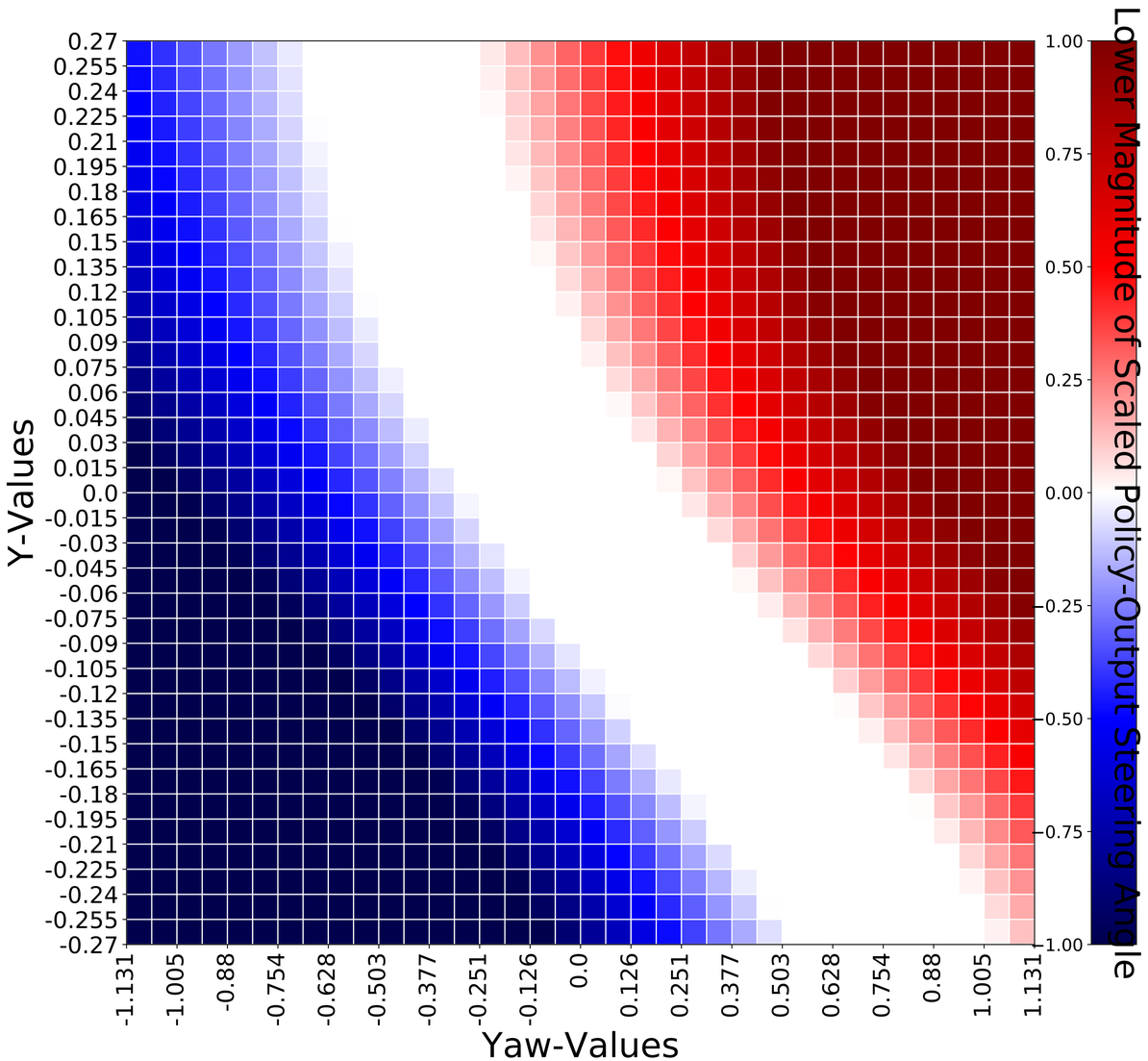


Line-Following Stability Results
xDot:0.700000, yDot:0.415384, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 16203cbb-c803-4cd7-8bc9-477190b0ff65
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_29_47_432536

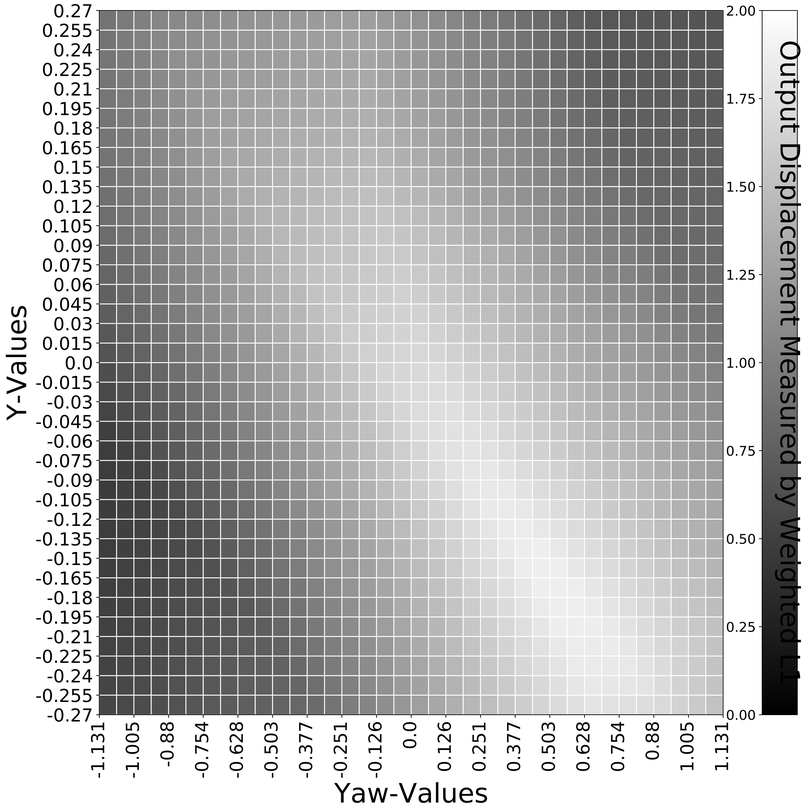


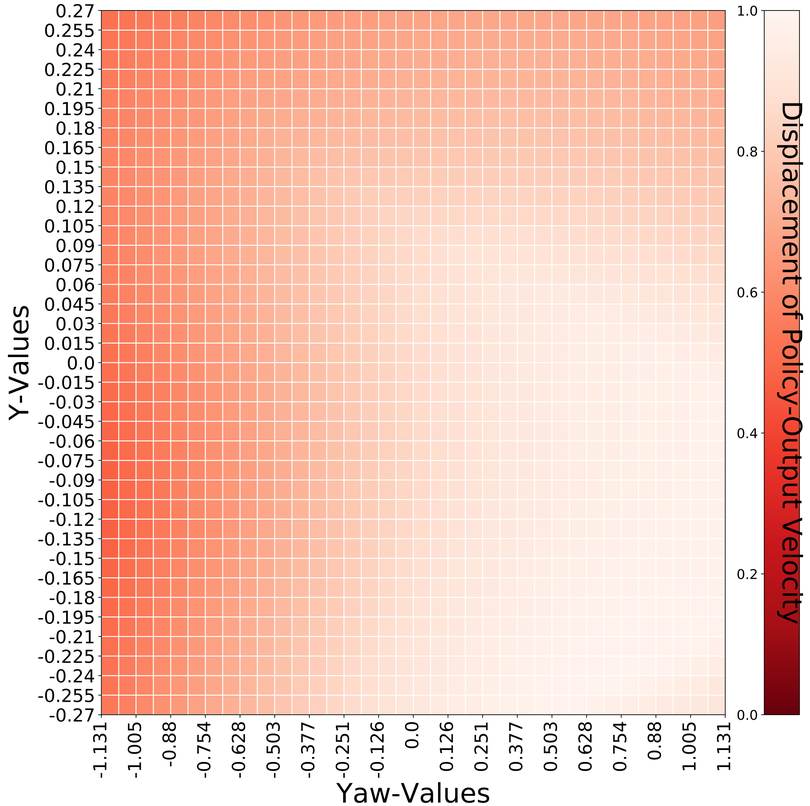
Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0.415384, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 06d35e78-c872-4e03-9fbb-d74c46f3bfe3
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_32_59_257069

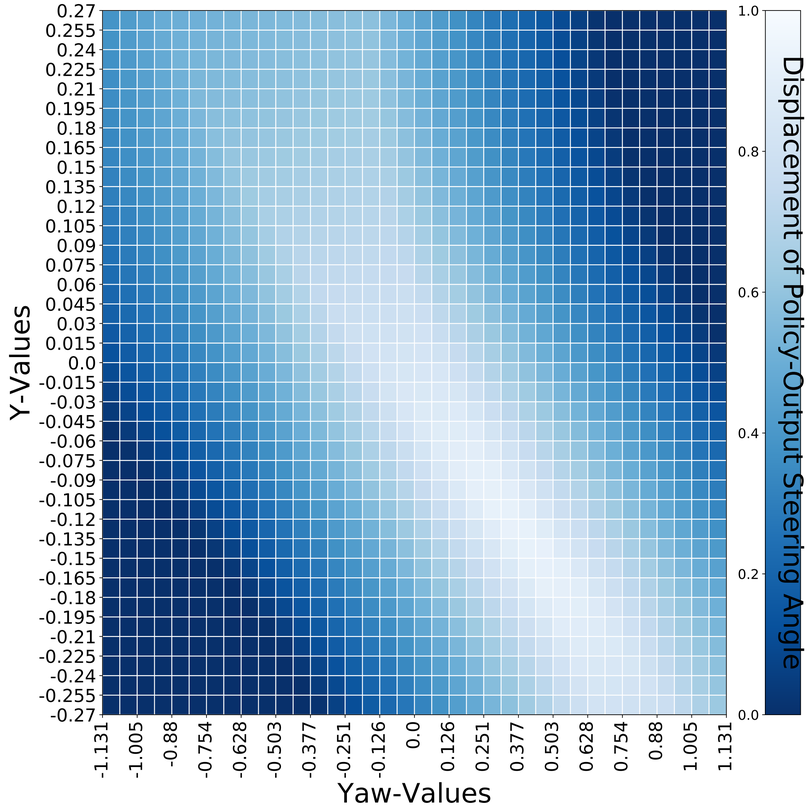




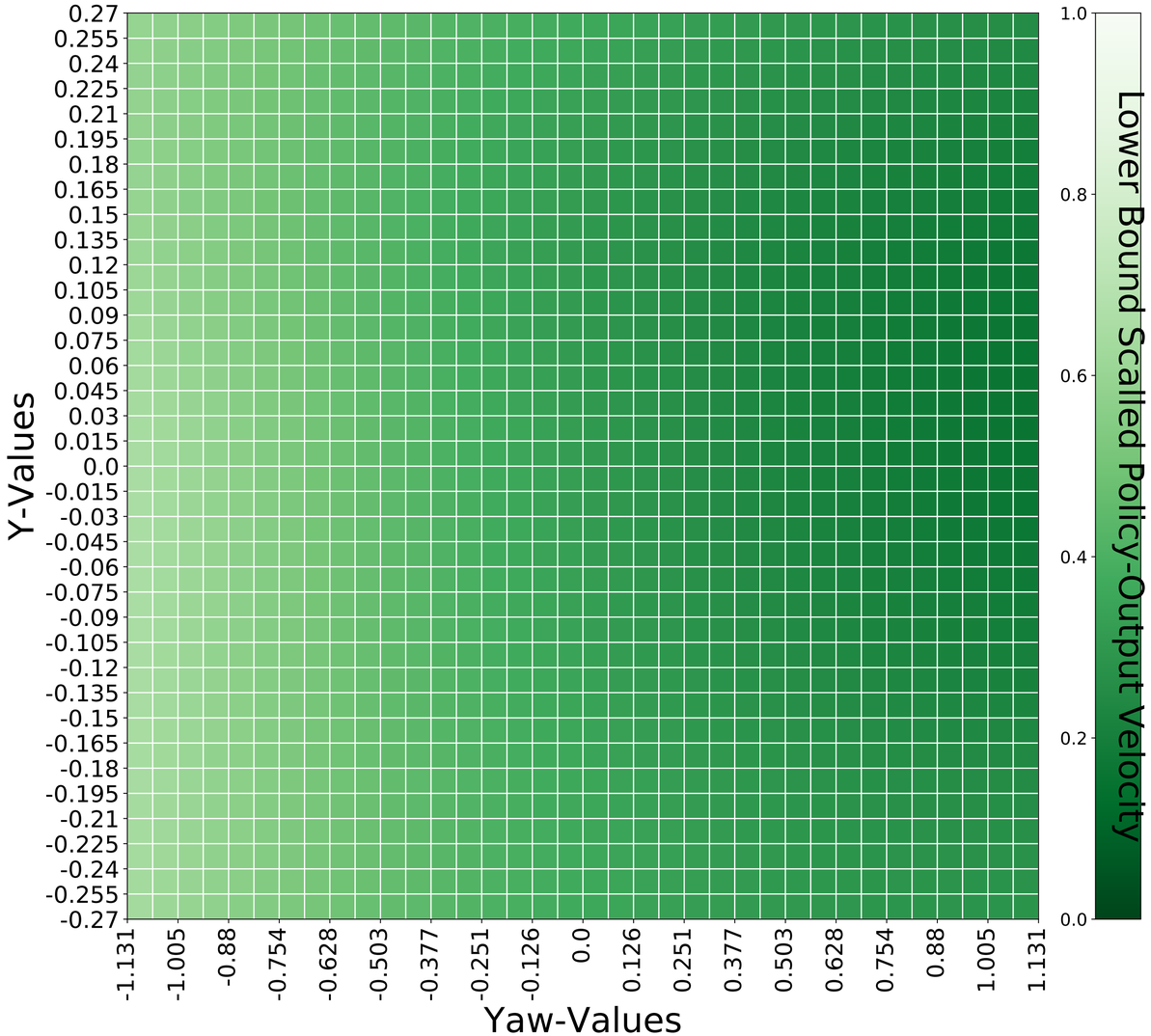
Line-Following Stability Results
xDot:0.700000, yDot:0.415384, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: ff2c08f5-4286-4725-b8e6-f9318e191249
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_31_54_636269

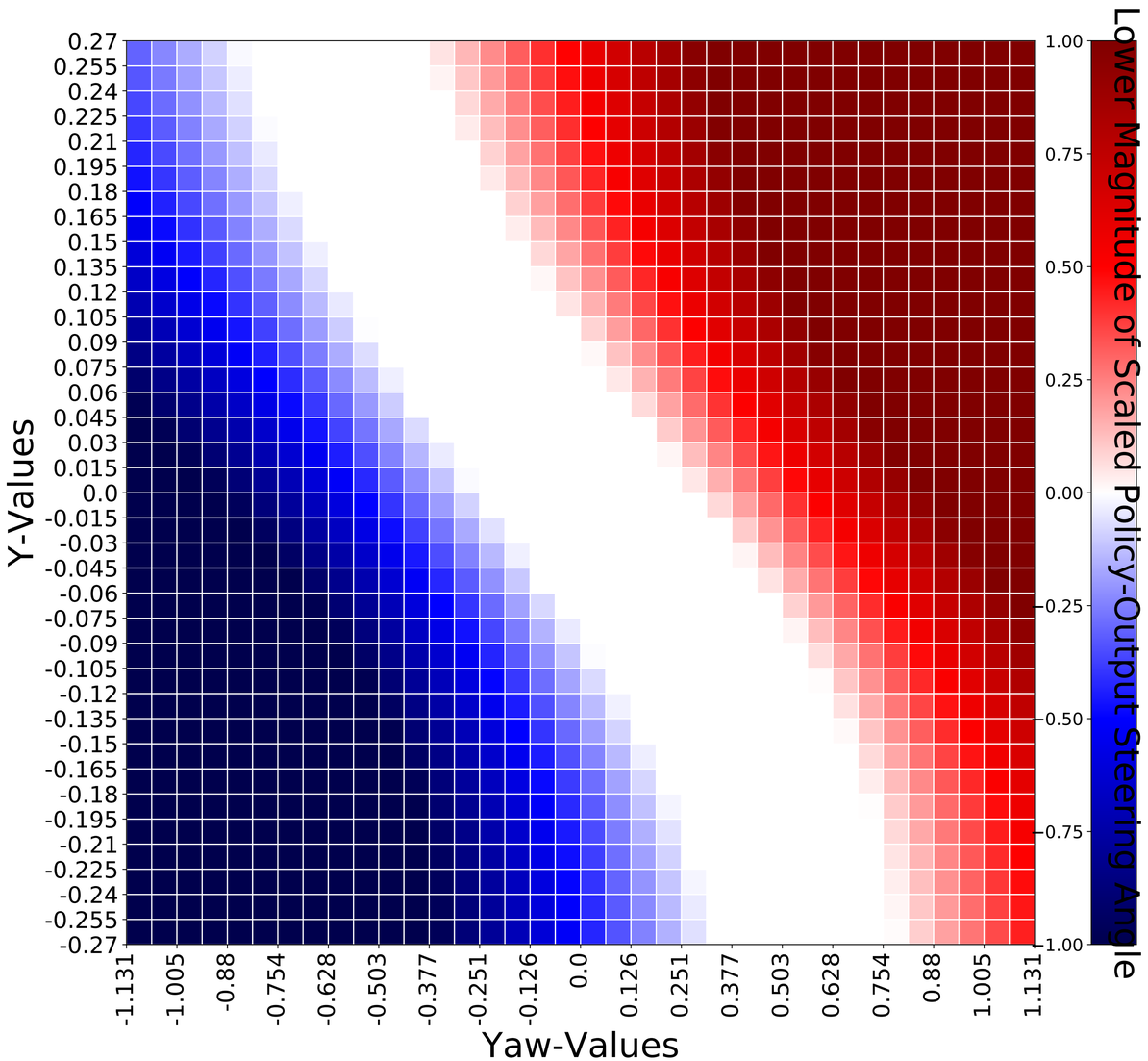




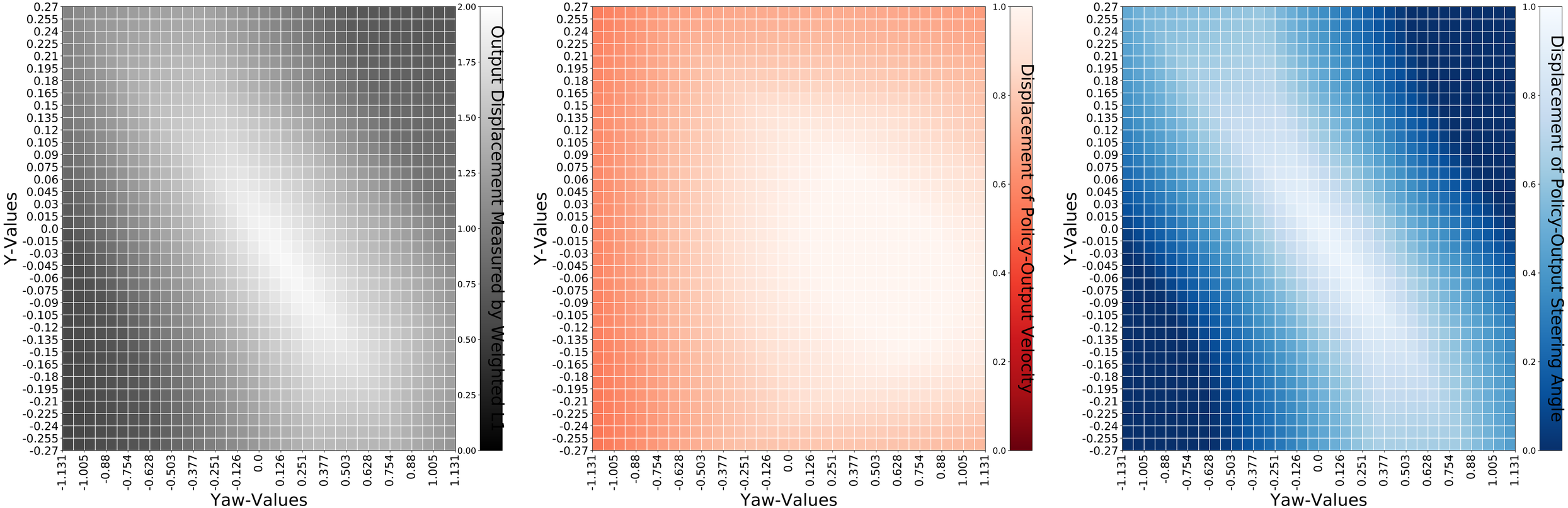


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0.415384, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: fcc3a070-8cee-4f5c-917b-bbe1f9a162de
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_35_07_196067

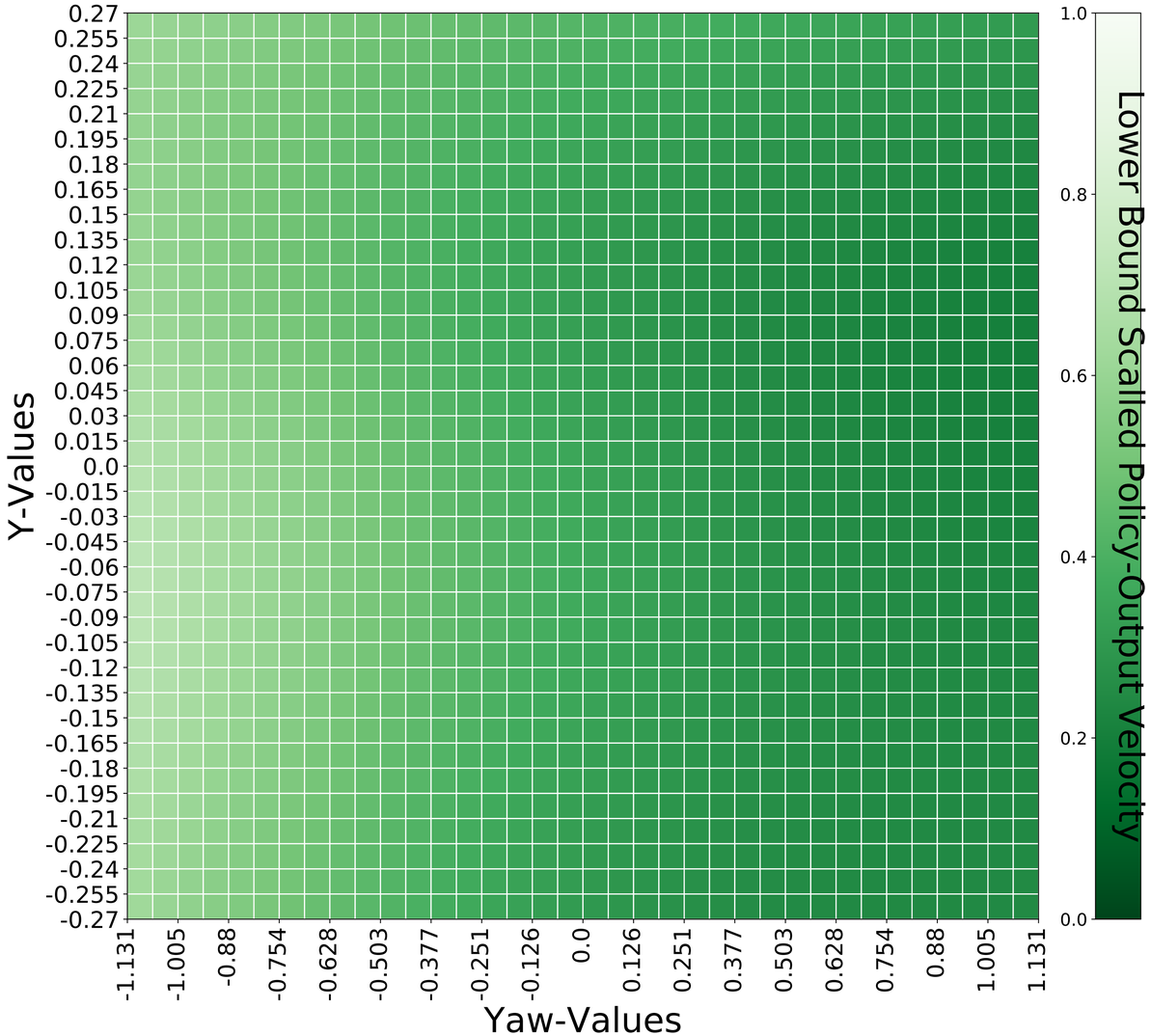


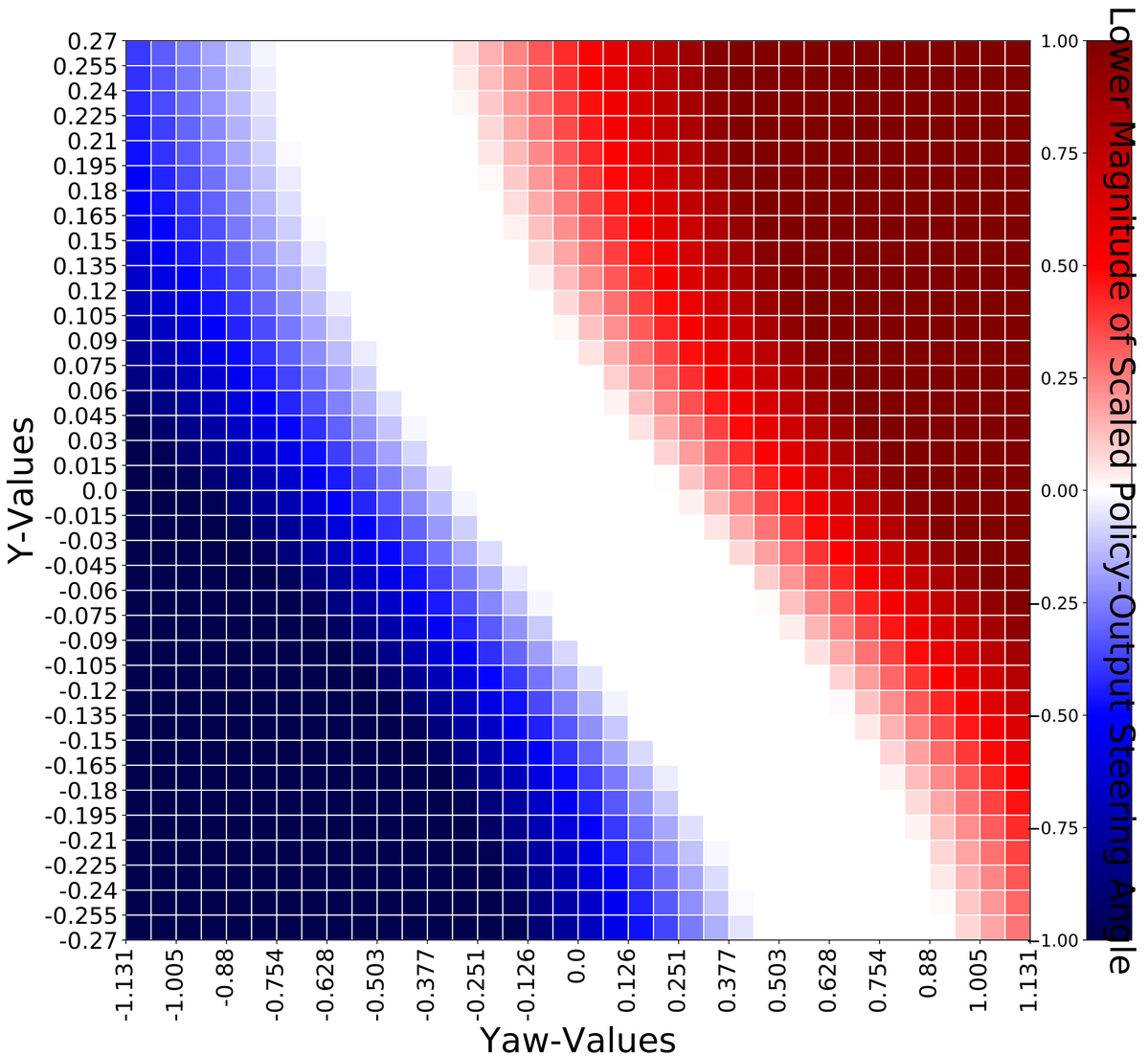


Line-Following Stability Results
xDot:0.700000, yDot:0.415384, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: d34df7cb-858d-41ff-86b1-ee3ceaad9071
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_34_02_668640

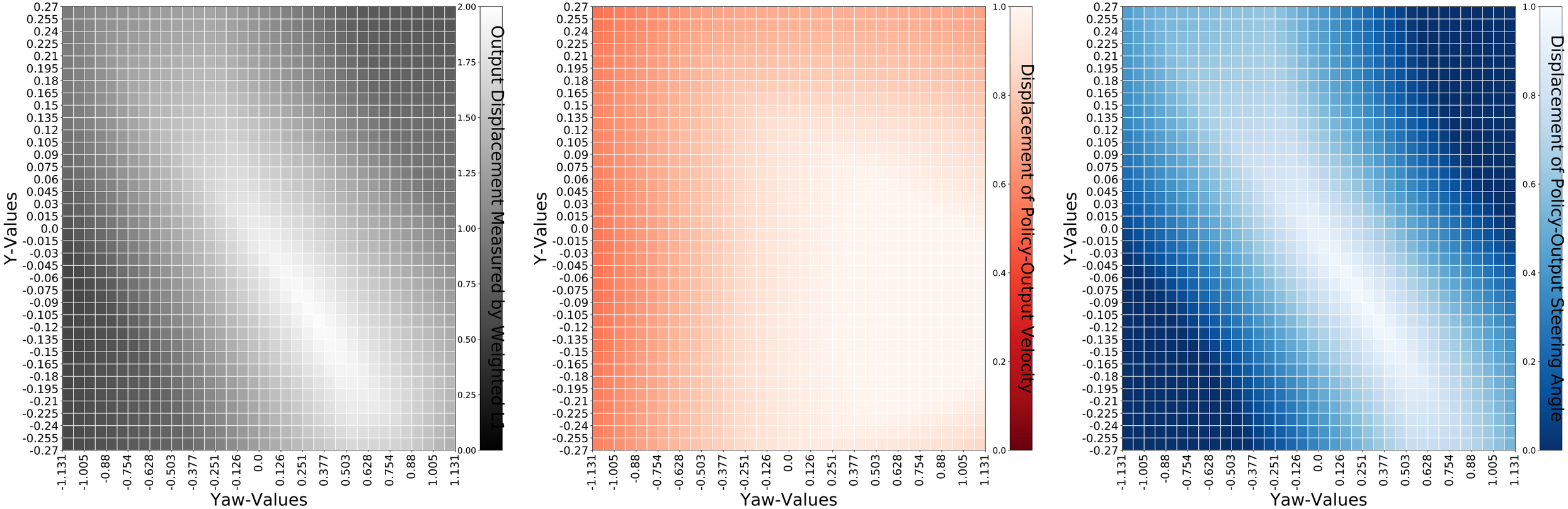


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0.415384, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 1aa0847a-9fd2-407b-81d4-273340fa3070
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_47_59_338688

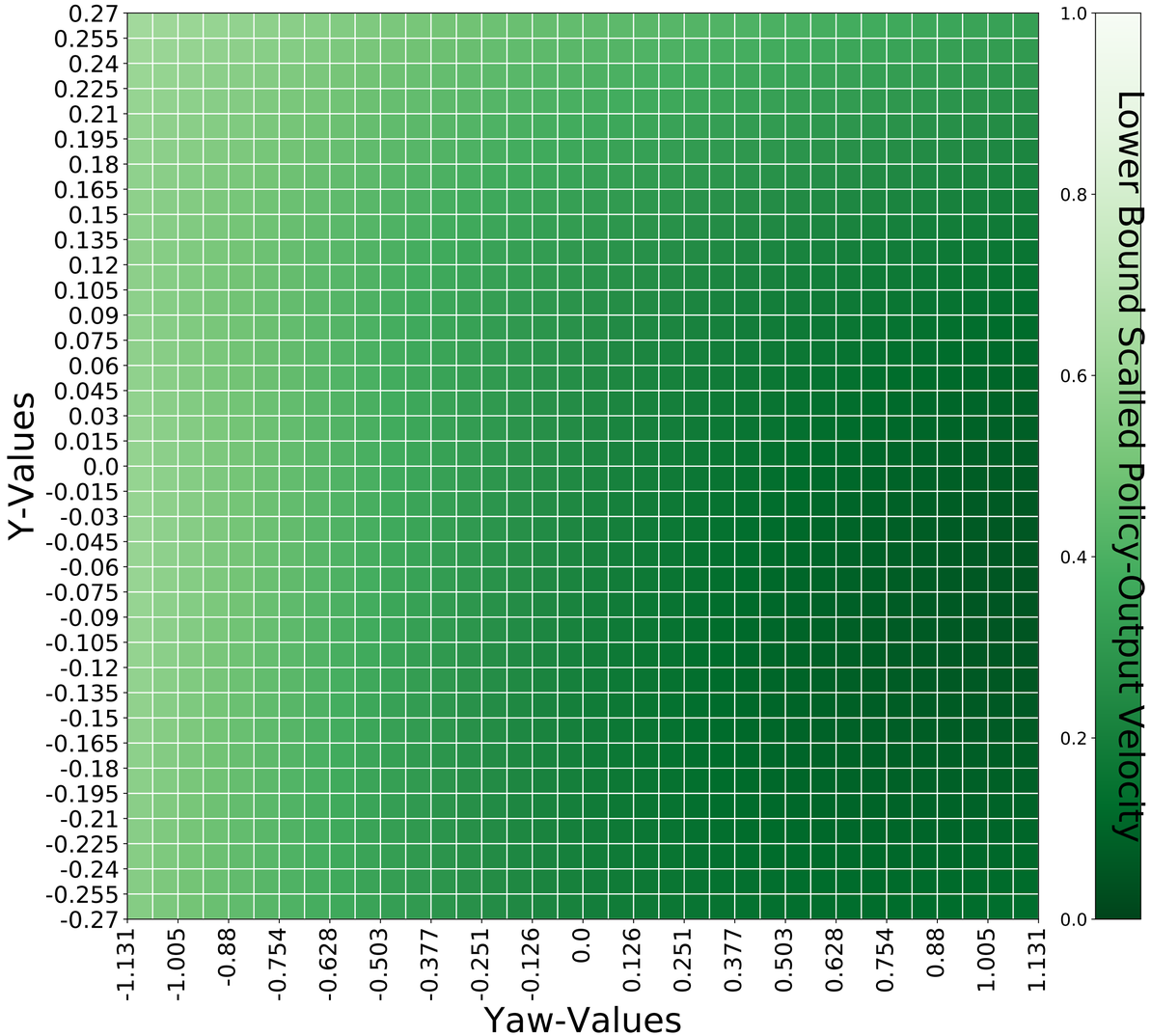


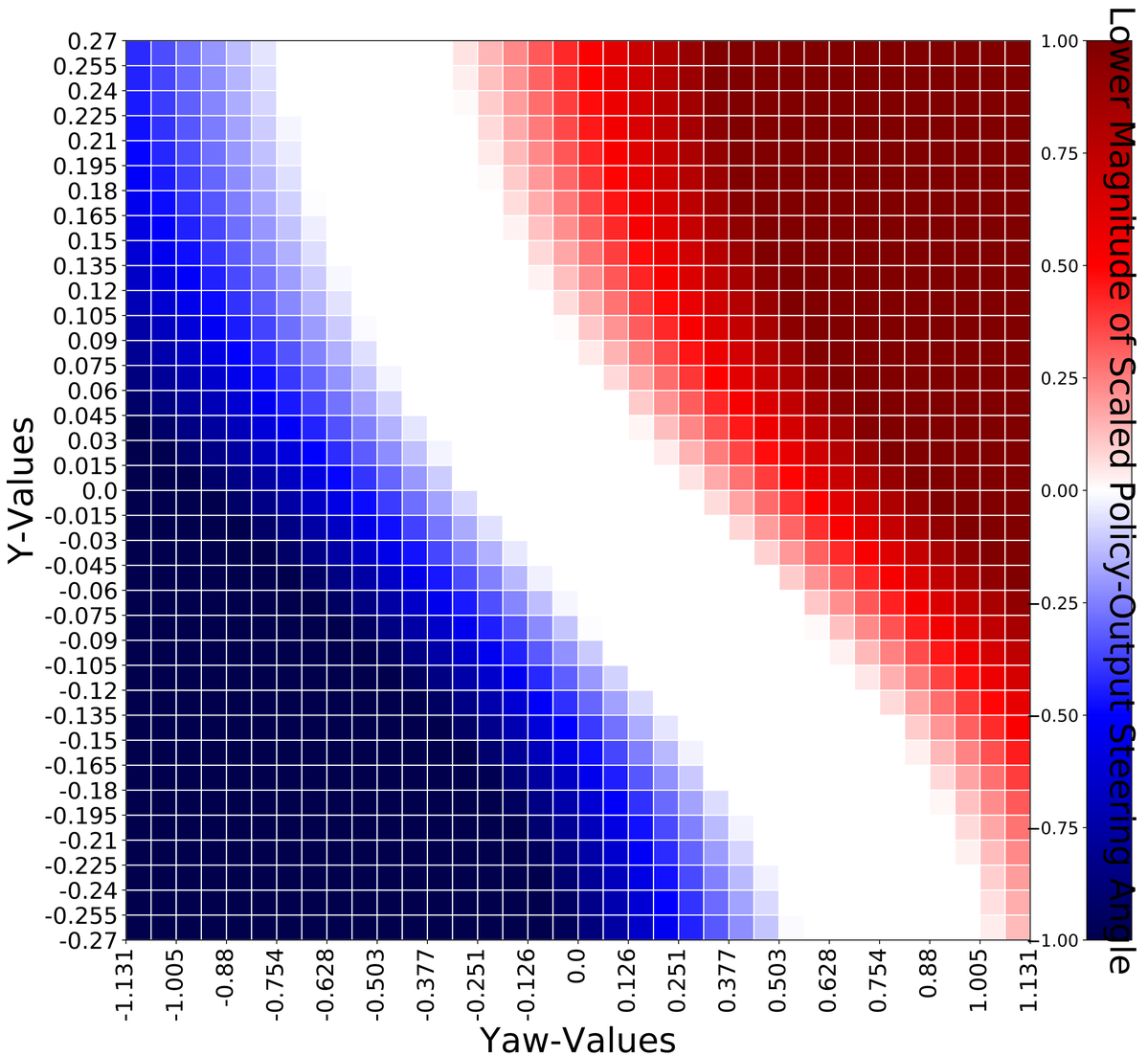


Line-Following Stability Results
xDot:0.700000, yDot:0.415384, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 58742ace-7214-4d28-8fdb-782cd9e00bdc
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_46_55_246300

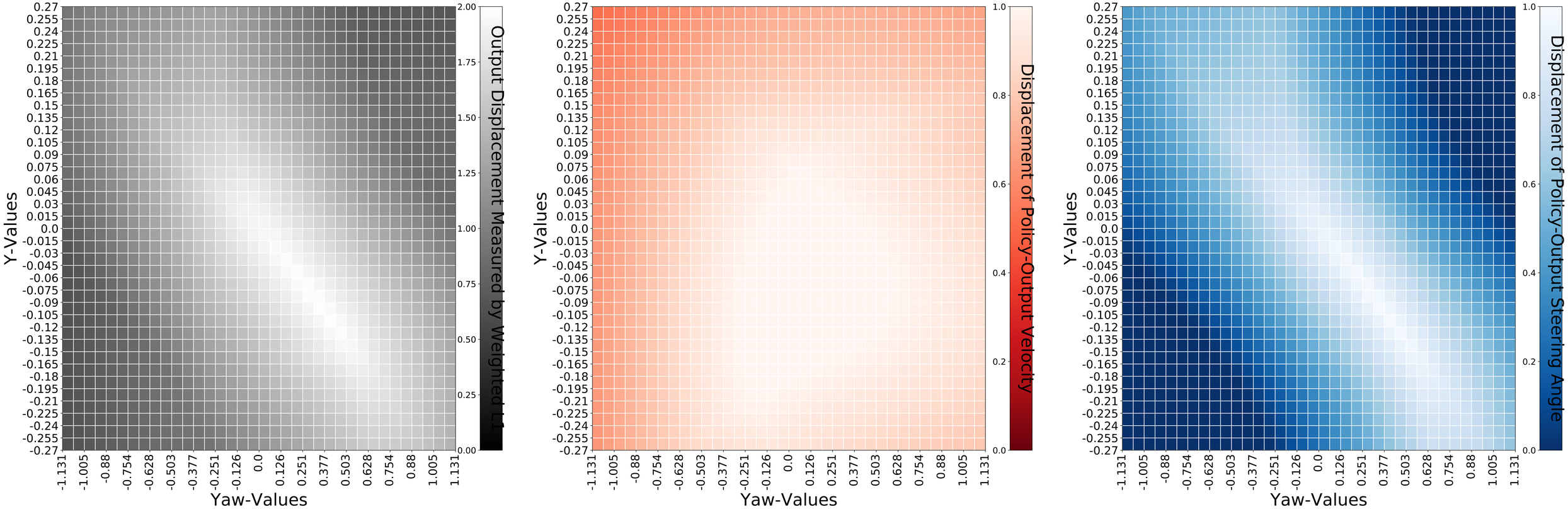


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 8e74b7de-945f-420b-bdf6-d6199dd7aa3f
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_25_20_092719

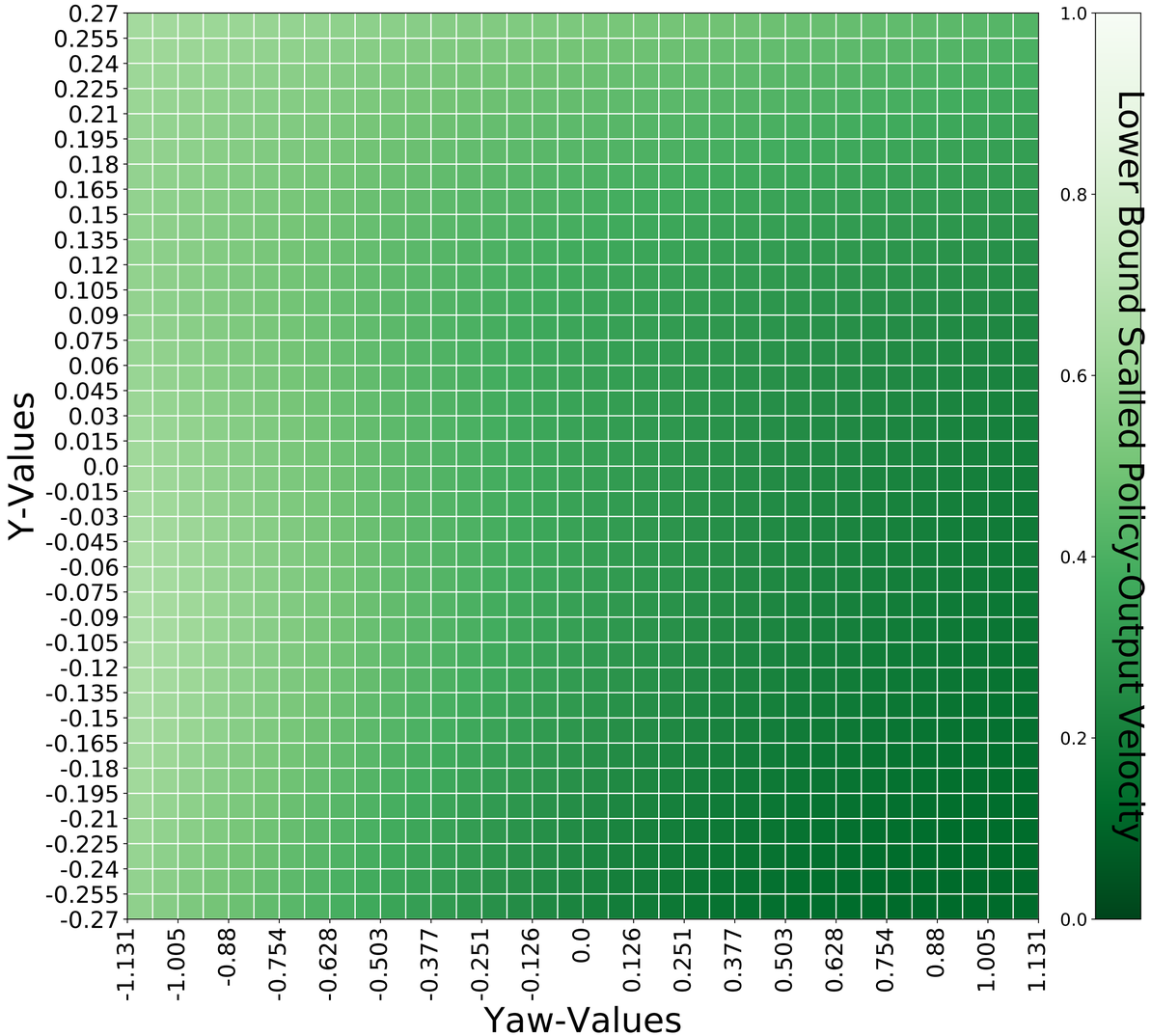


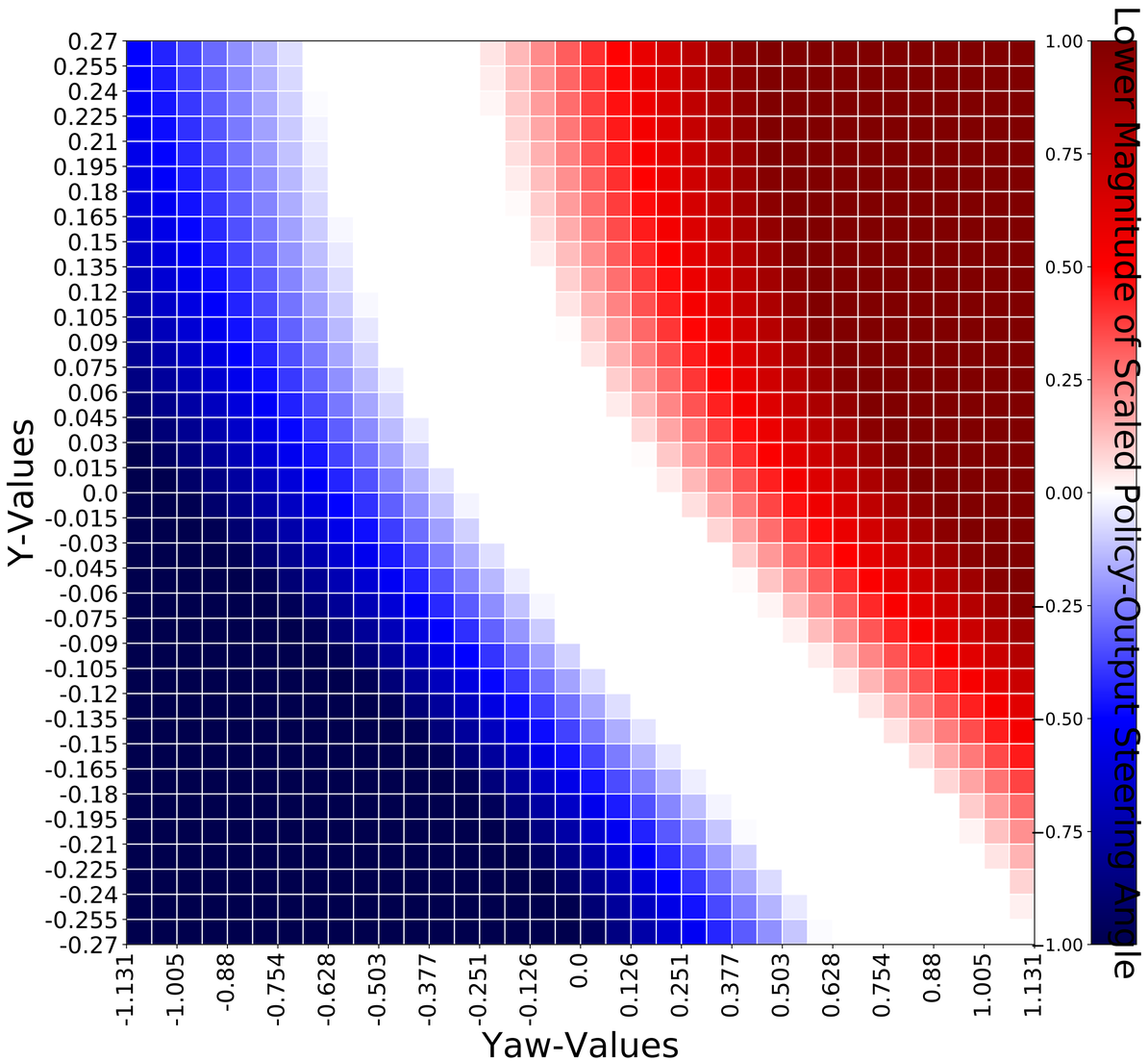


Line-Following Stability Results
xDot:0.700000, yDot:0, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: f3e176d9-6121-4967-ab18-736eba2befdd
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_24_14_582275

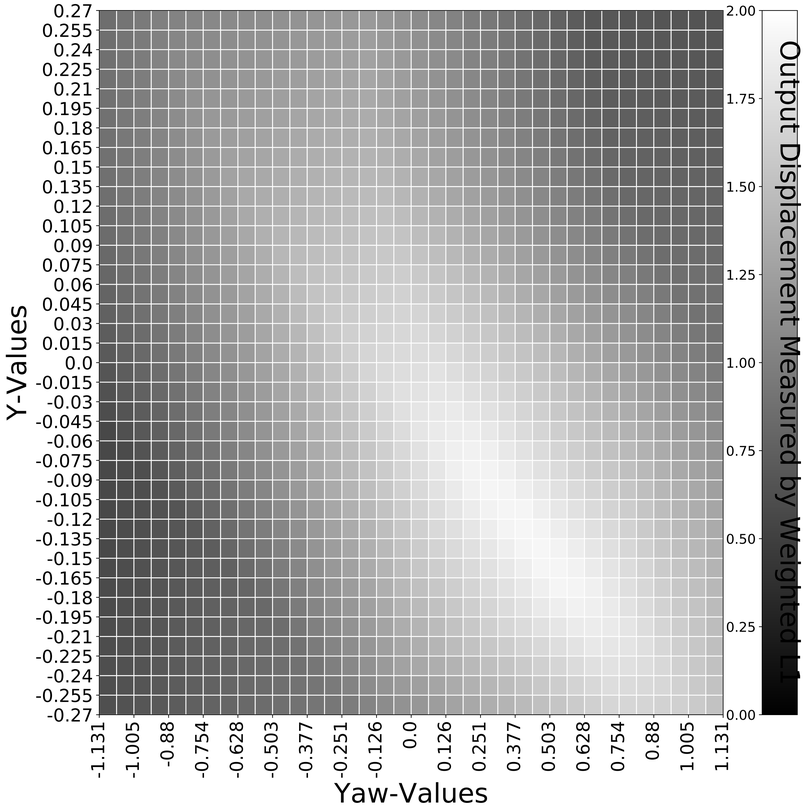


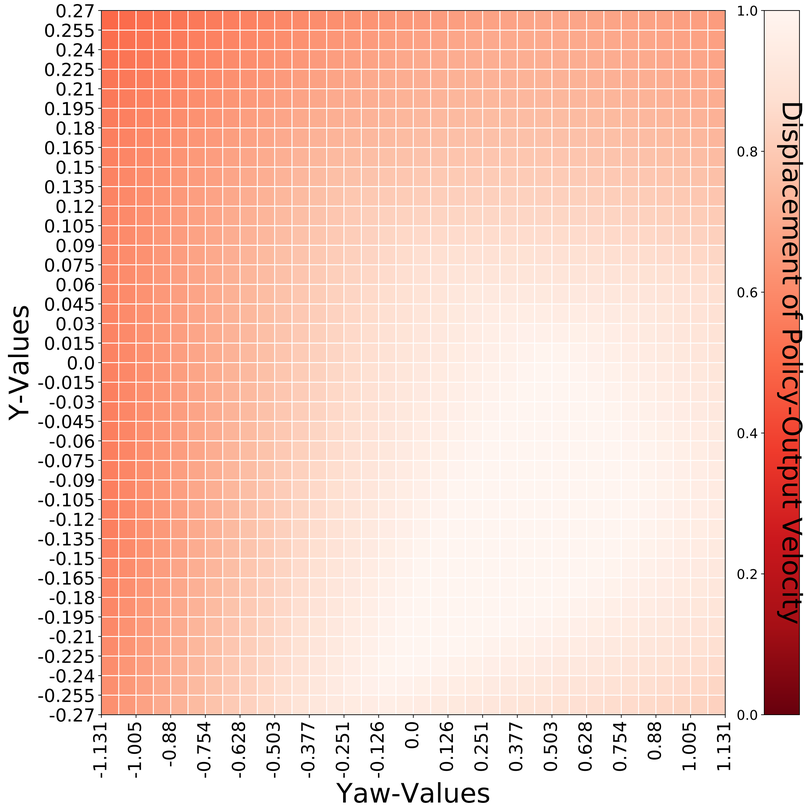
Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 99be0a15-100b-4e7c-94ea-43fe68b7e6b0
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_27_31_094874

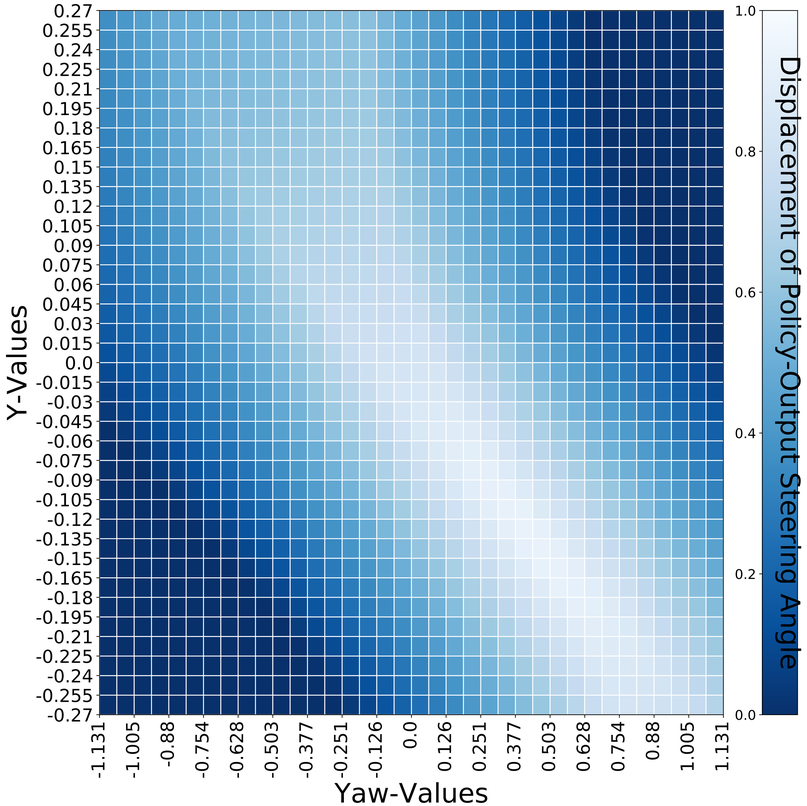




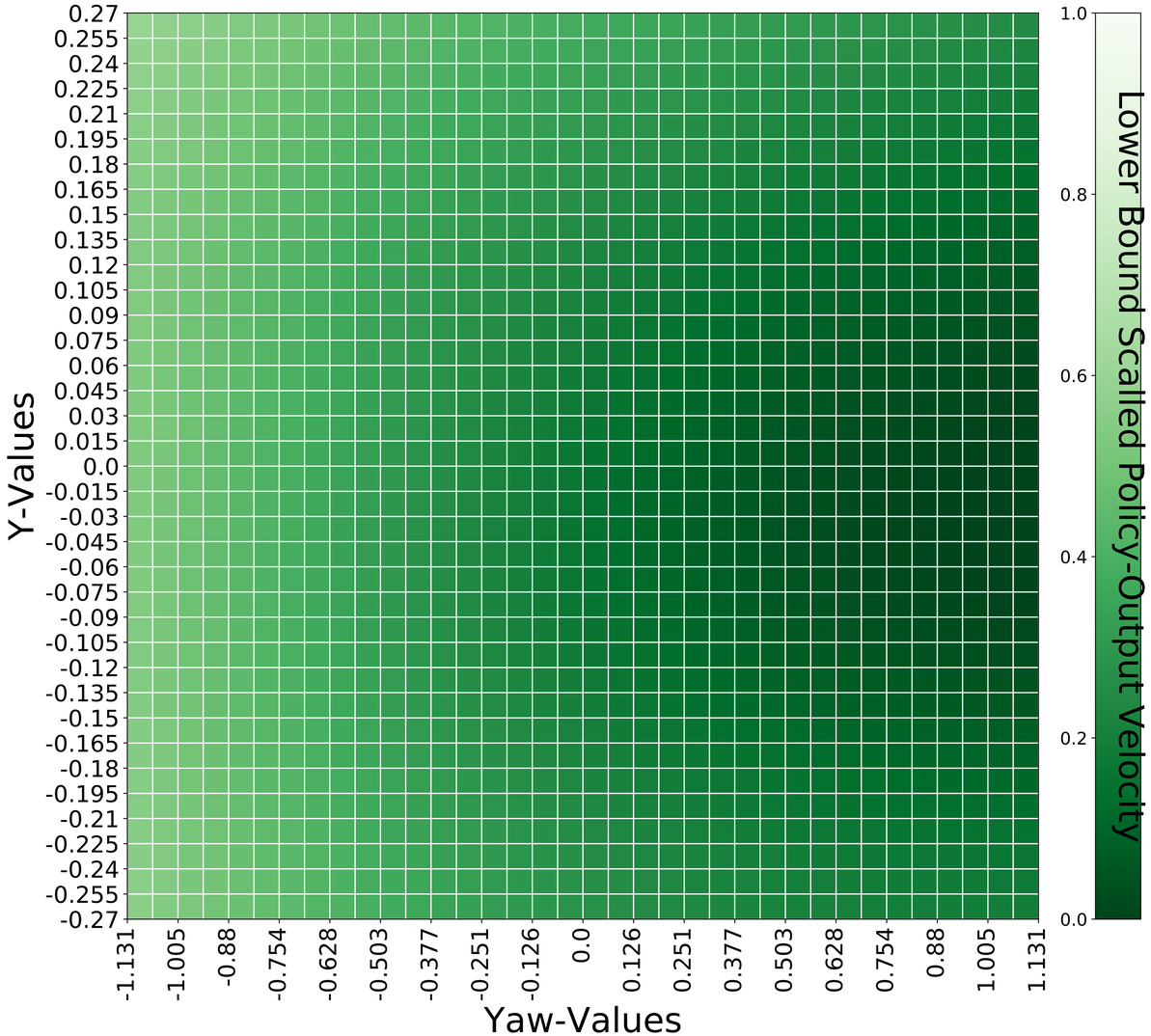
Line-Following Stability Results
xDot:0.700000, yDot:0, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: d0829dc3-7071-483a-afc9-4207be15f77b
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_26_25_043973

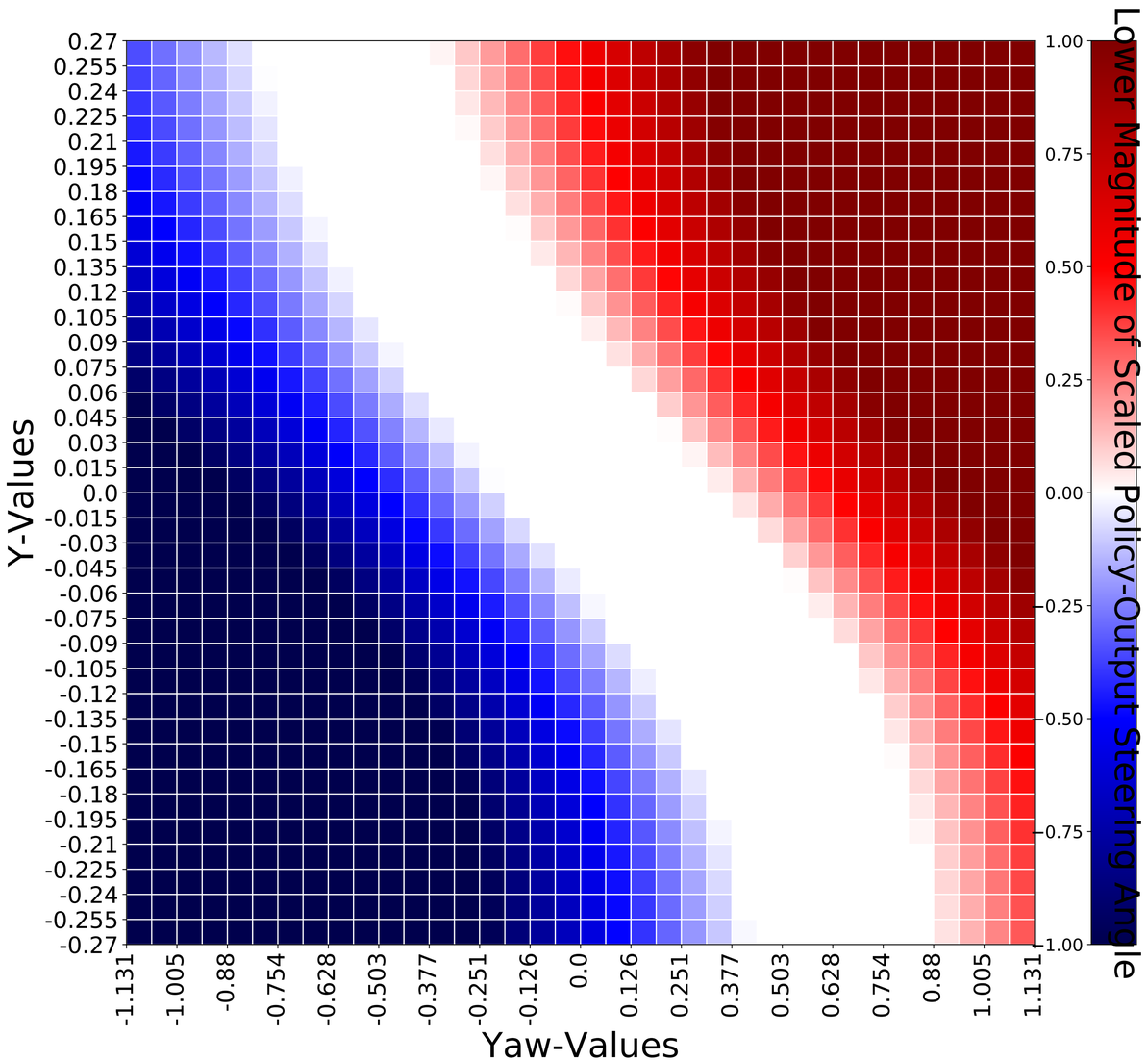




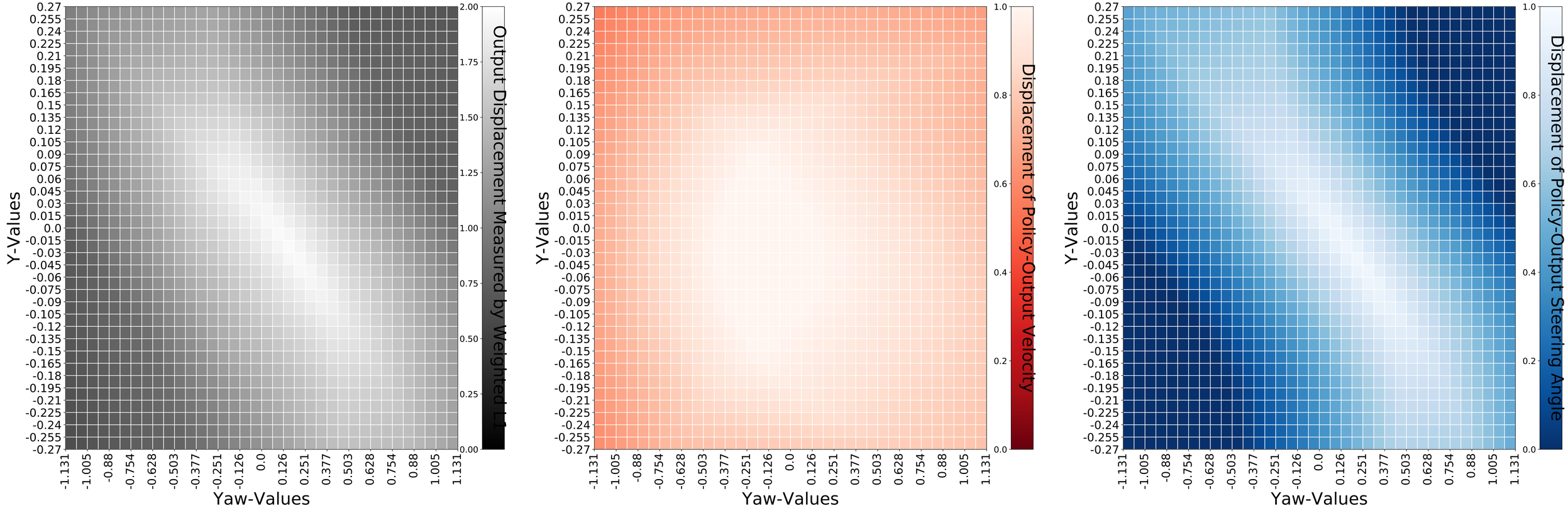


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 63c0f649-96bc-4648-a16e-e3ba513d40c7
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_29_42_338626

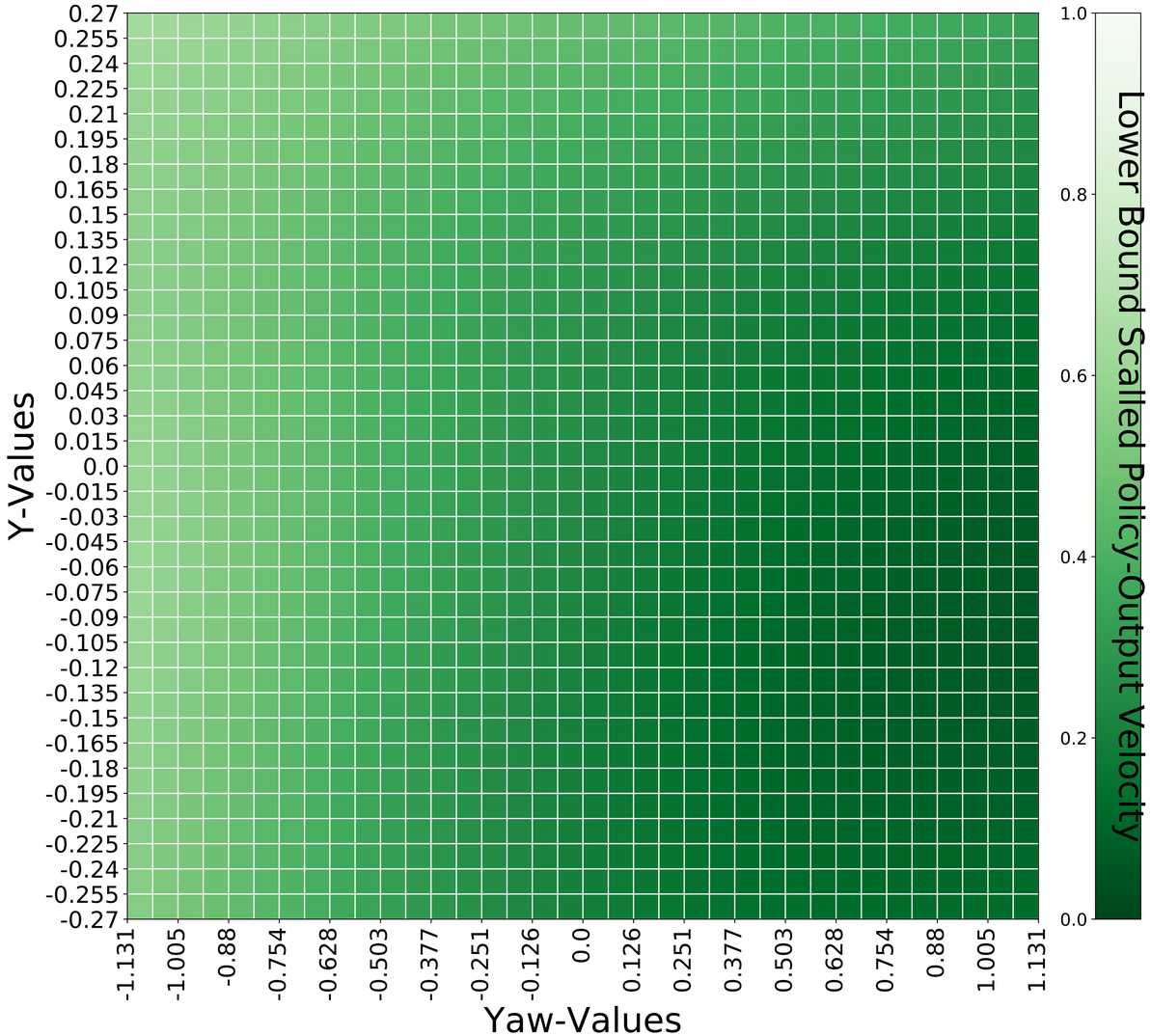


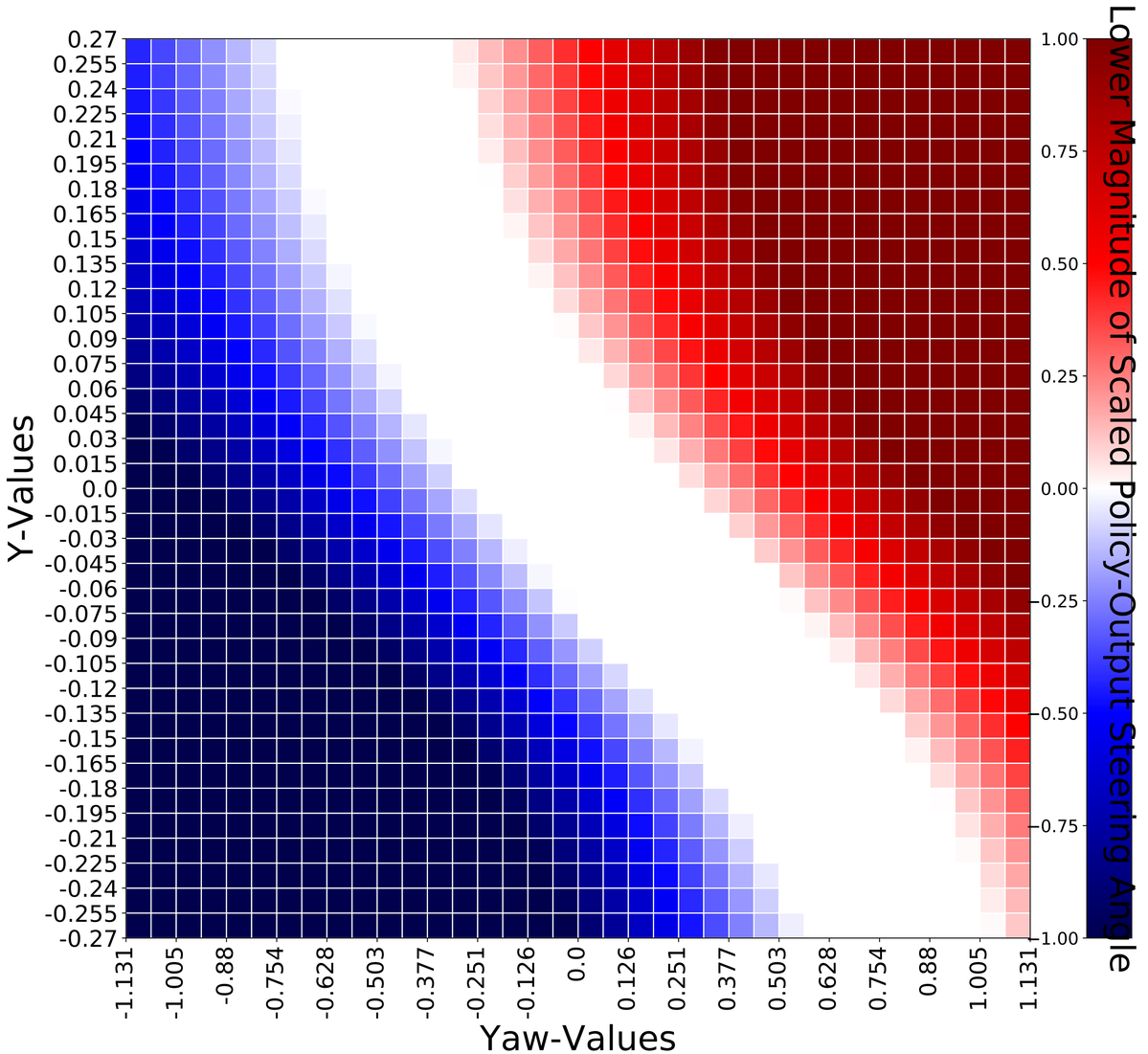


Line-Following Stability Results
xDot:0.700000, yDot:0, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 78a0b970-8f49-41c1-982a-47362396dd67
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_28_35_683976

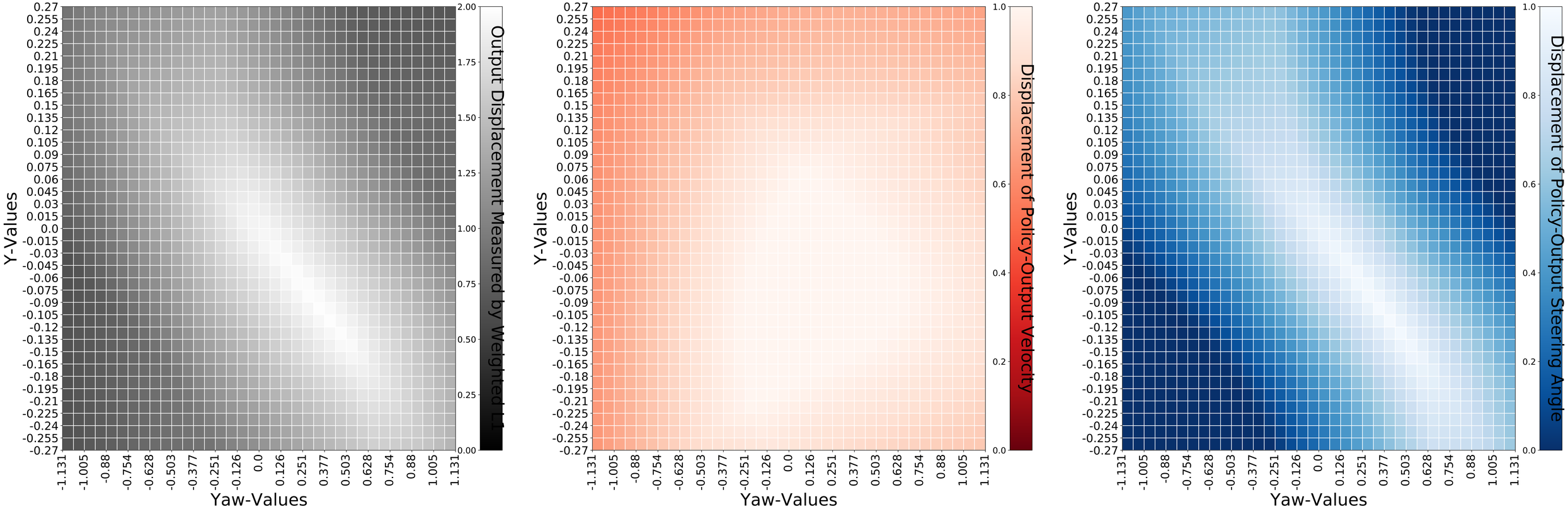


Line-Following Min. Magnitude Output Results
xDot:0.700000, yDot:0, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 740f0798-cbcd-442a-b741-9b17c52c15c1
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_42_49_096908

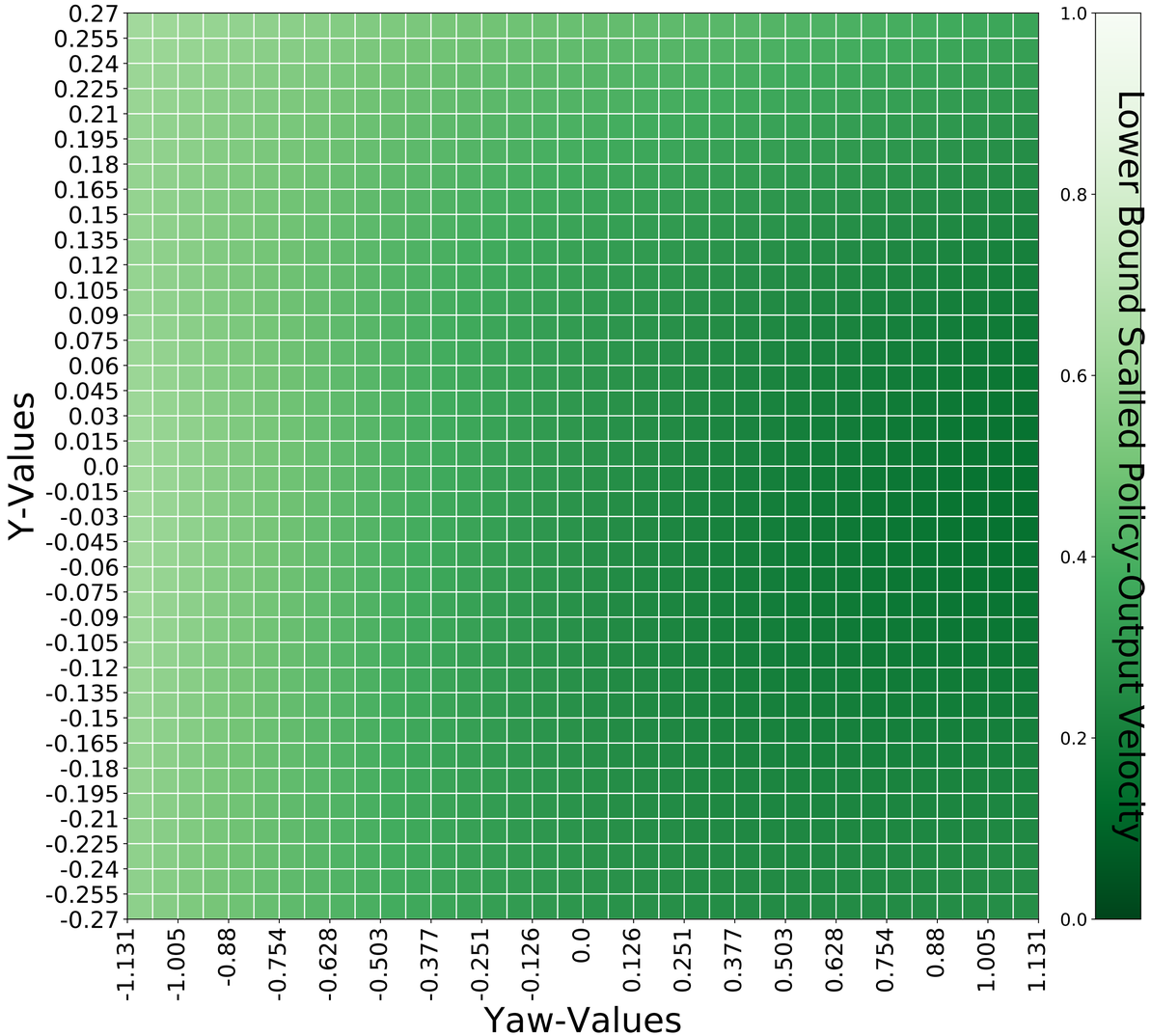


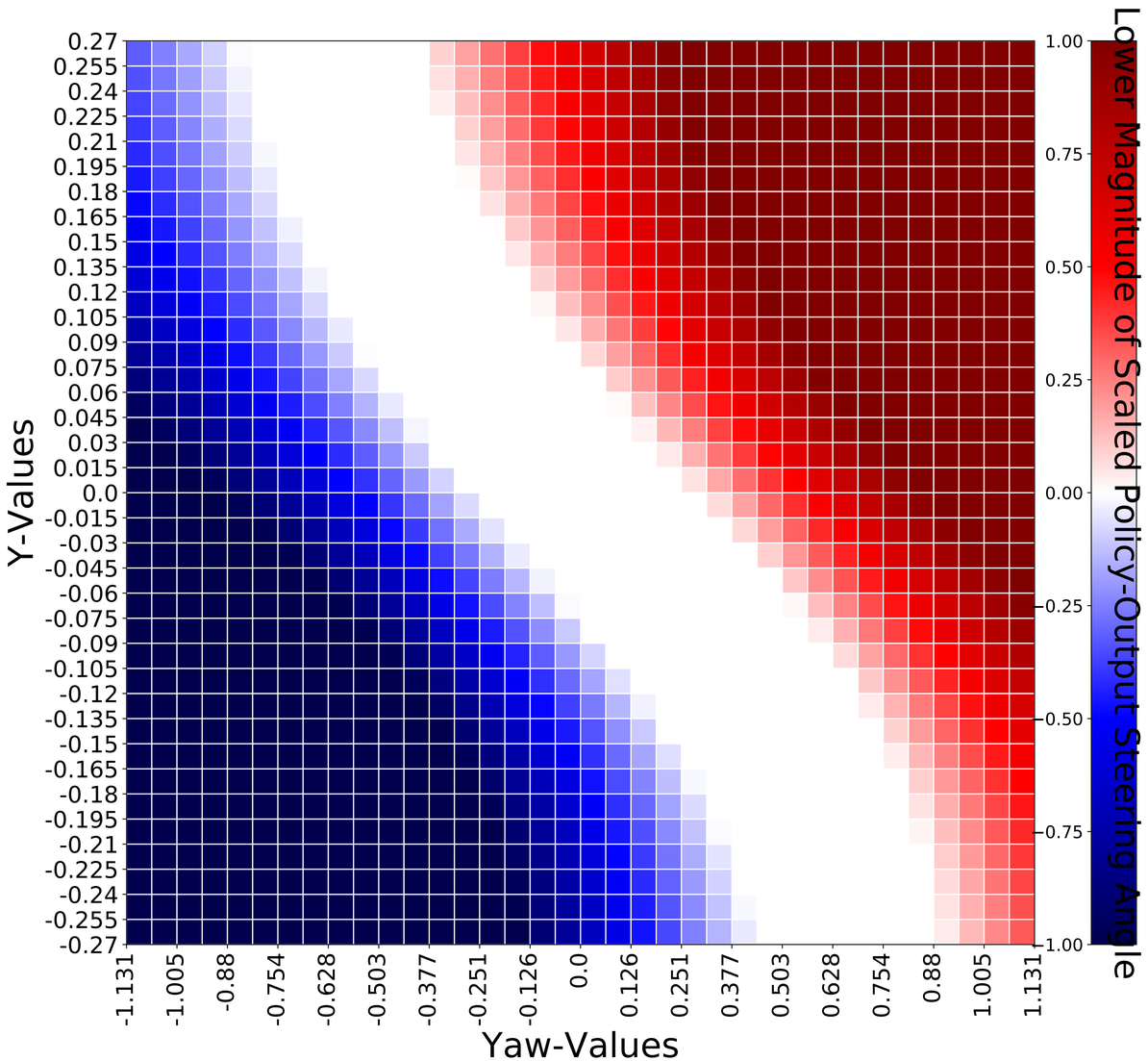


Line-Following Stability Results
xDot:0.700000, yDot:0, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 1fedf341-213c-4512-a74d-0e4bb3d4d362
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_41_42_521956

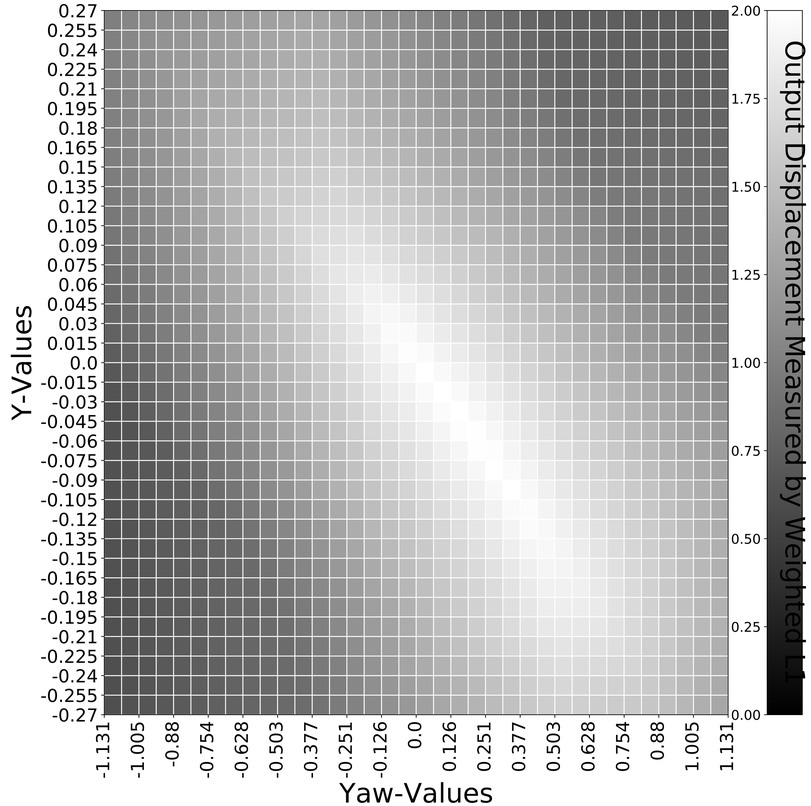


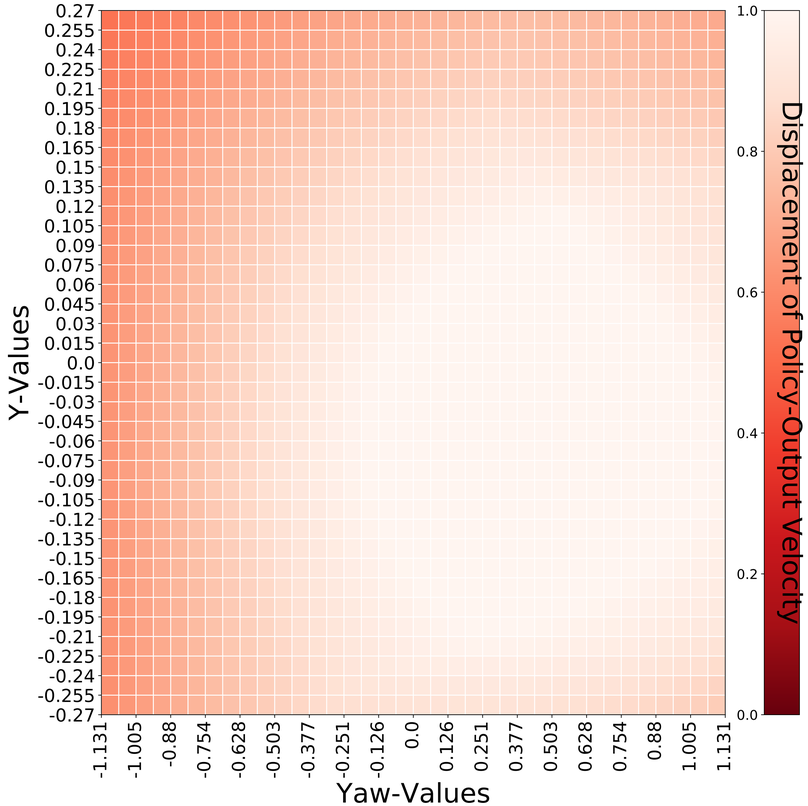
Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0.046153, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 4ca429d6-58db-44c1-8df9-229219dc1cae
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_45_00_255642

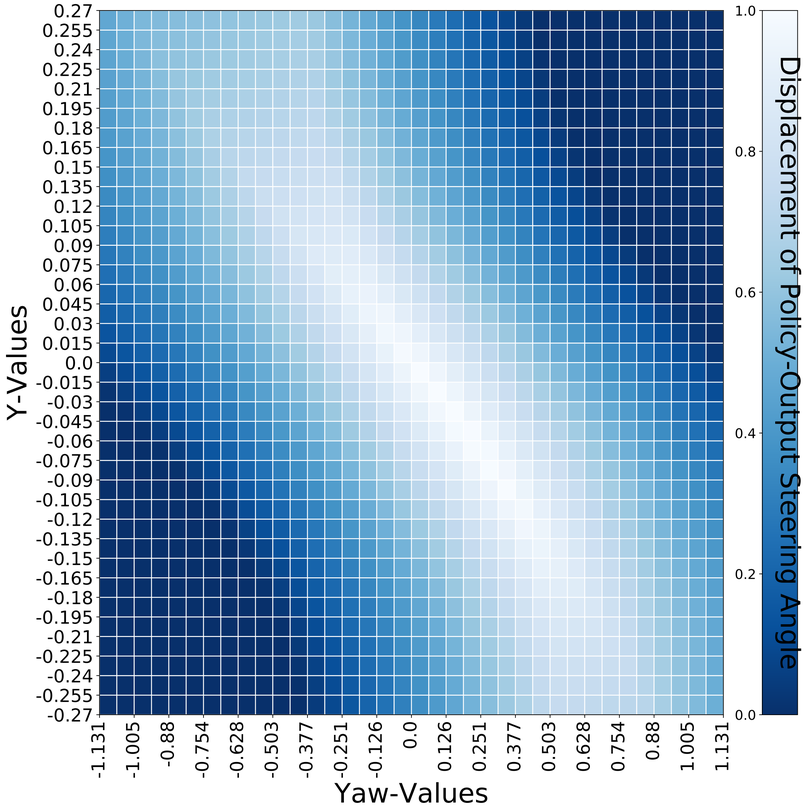




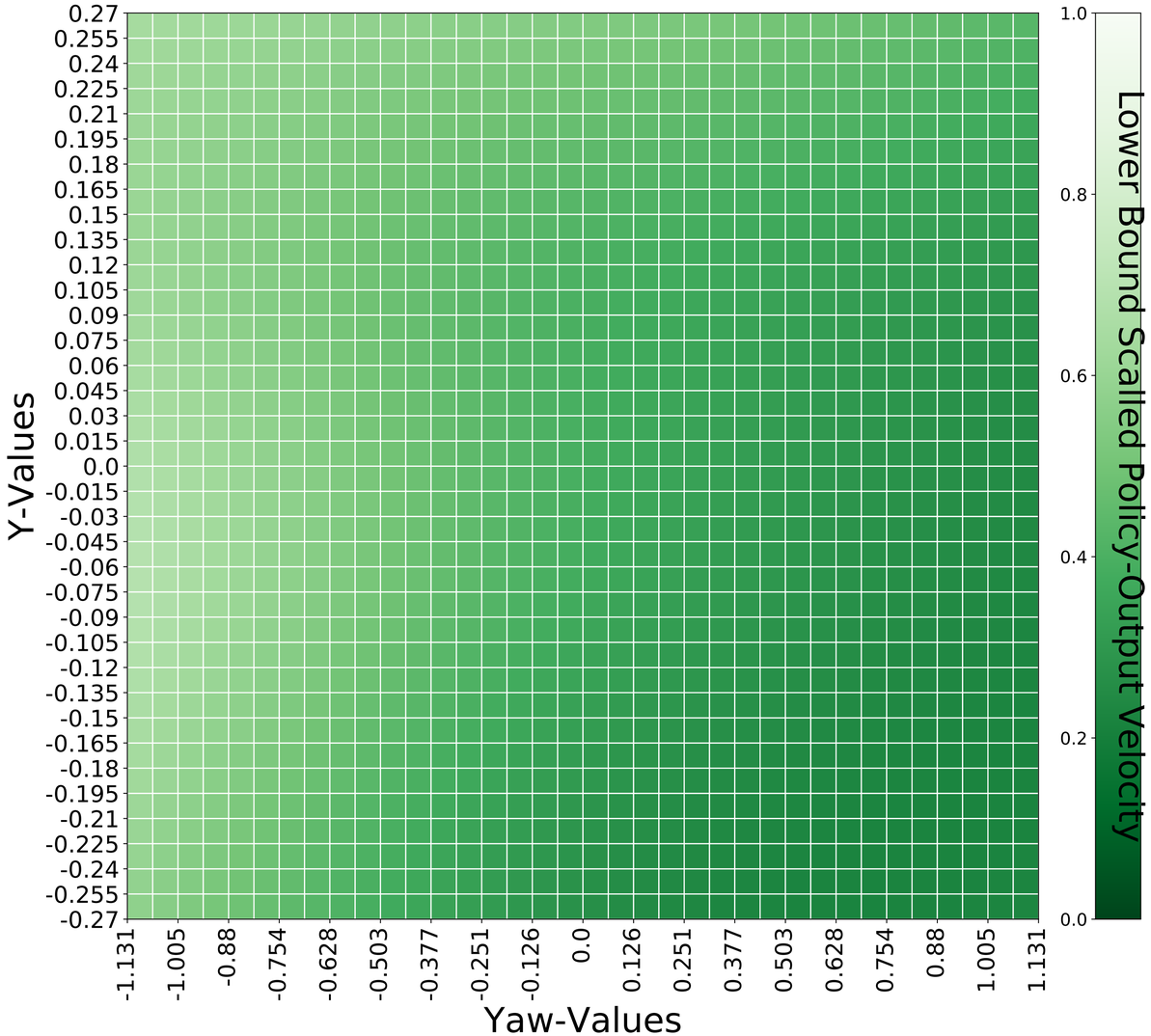
Line-Following Stability Results
xDot:0.300000, yDot:0.046153, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 58d764ef-3773-4f0d-a31c-5c50ab02a068
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_43_54_094278

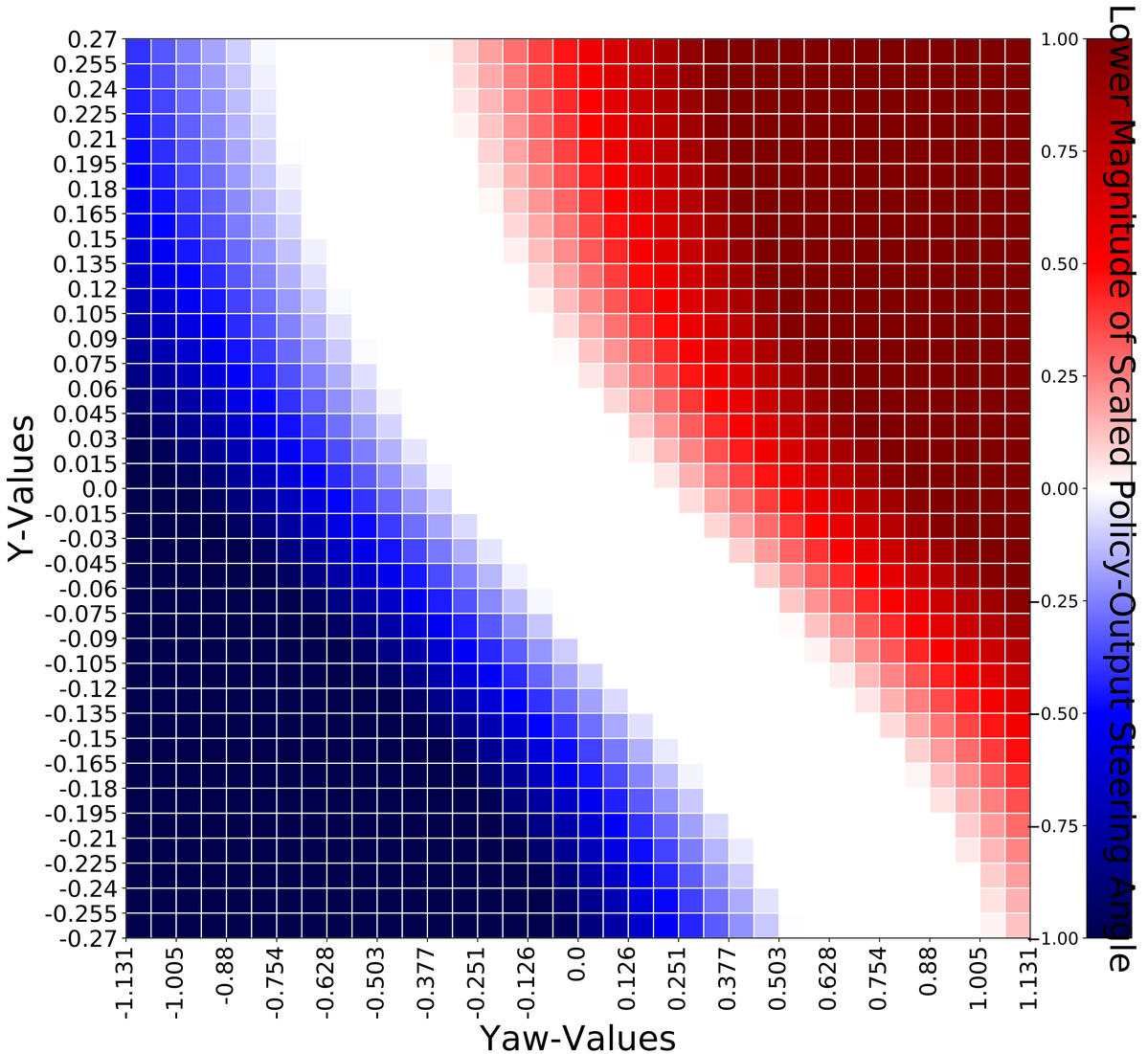




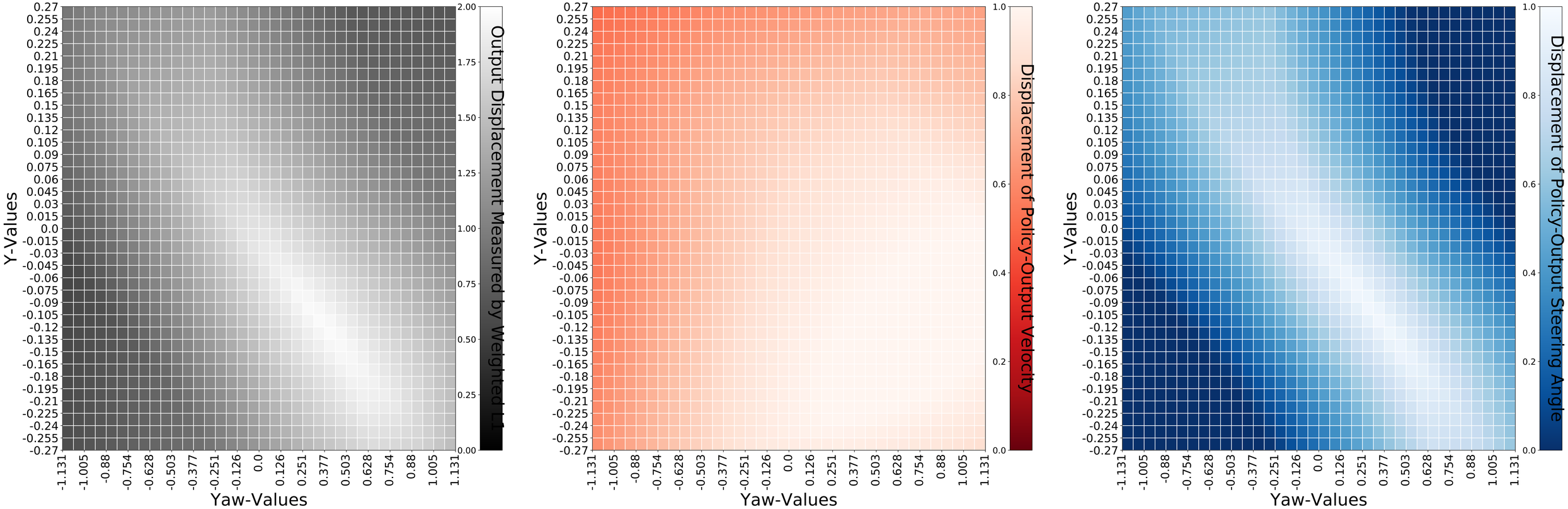


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0.046153, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 93910317-d32e-4a92-9023-1a3fb7d5d0f0
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_47_11_398078

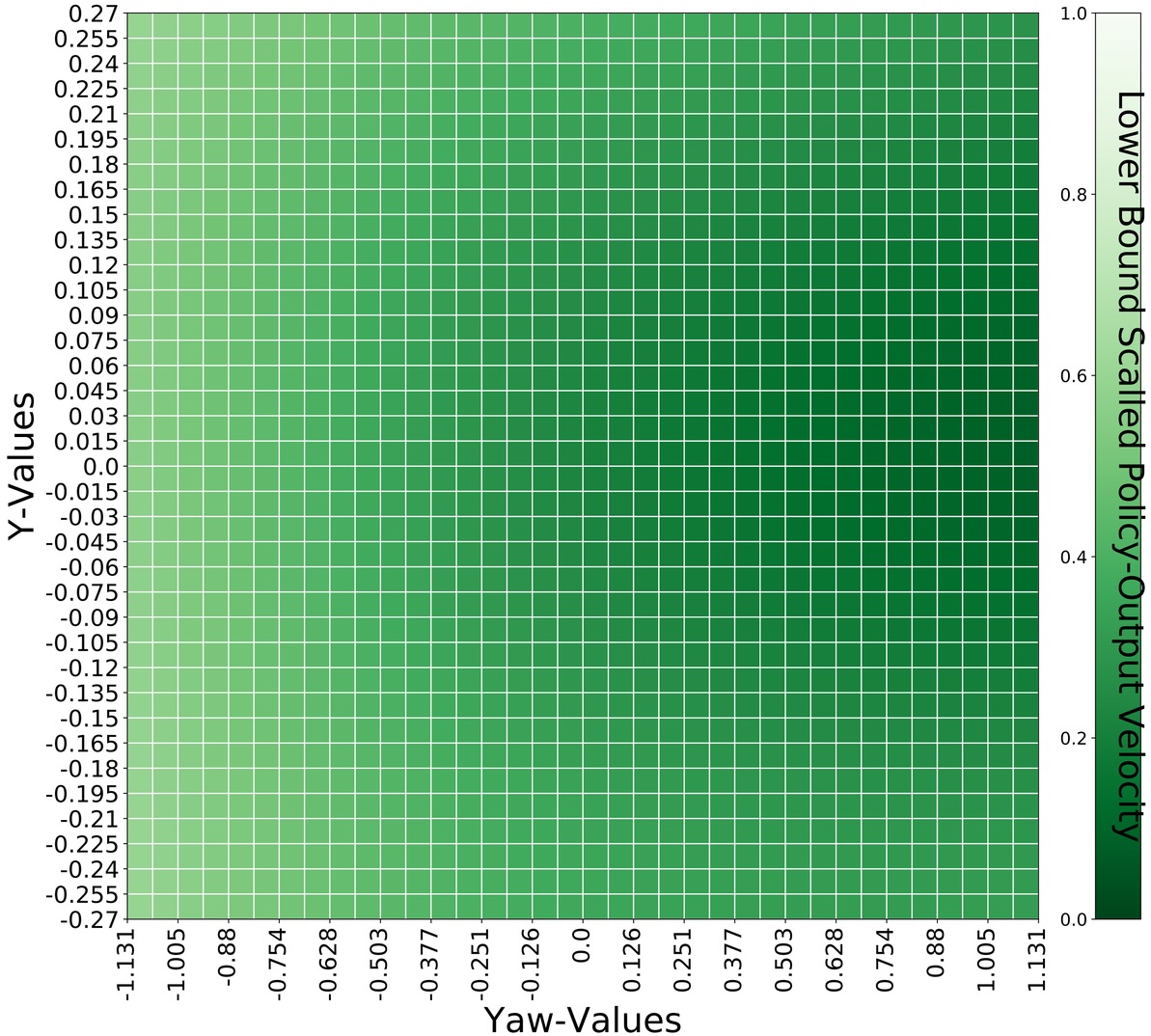


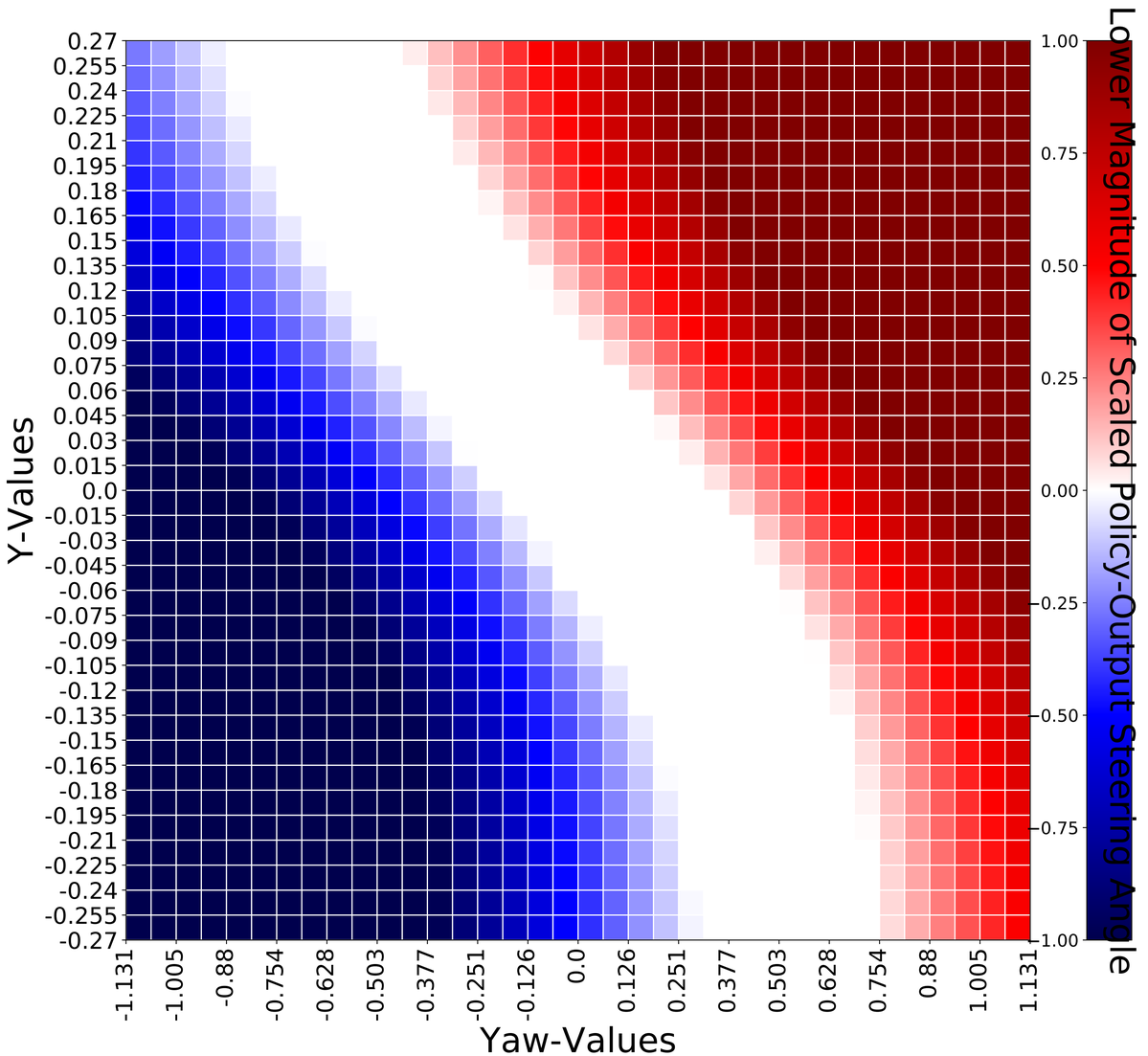


Line-Following Stability Results
xDot:0.300000, yDot:0.046153, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 73aa5ac1-5c5a-4a21-8c86-73edfdf9849f
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_46_05_155526

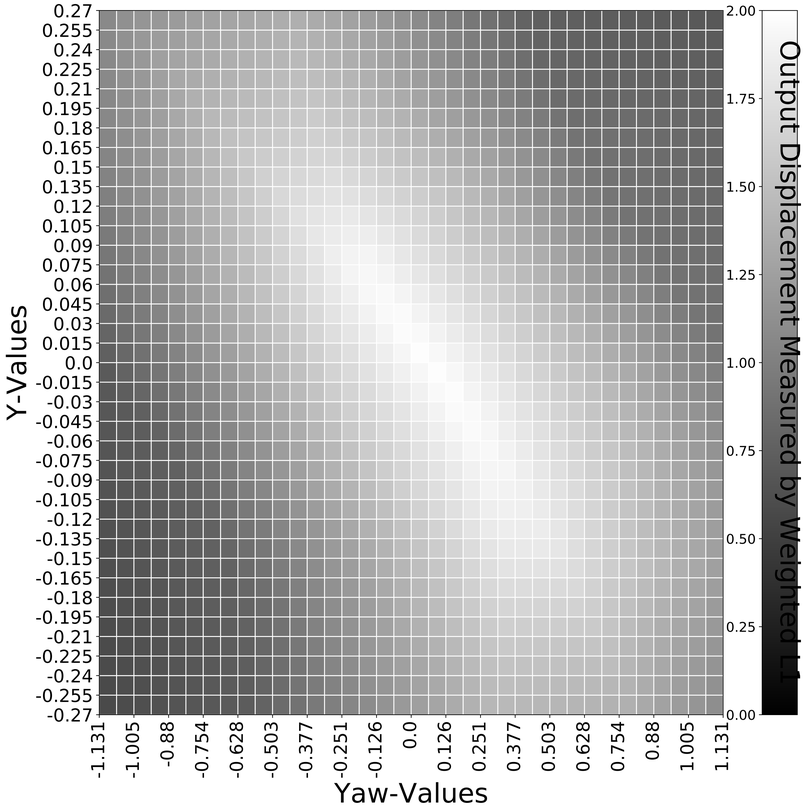


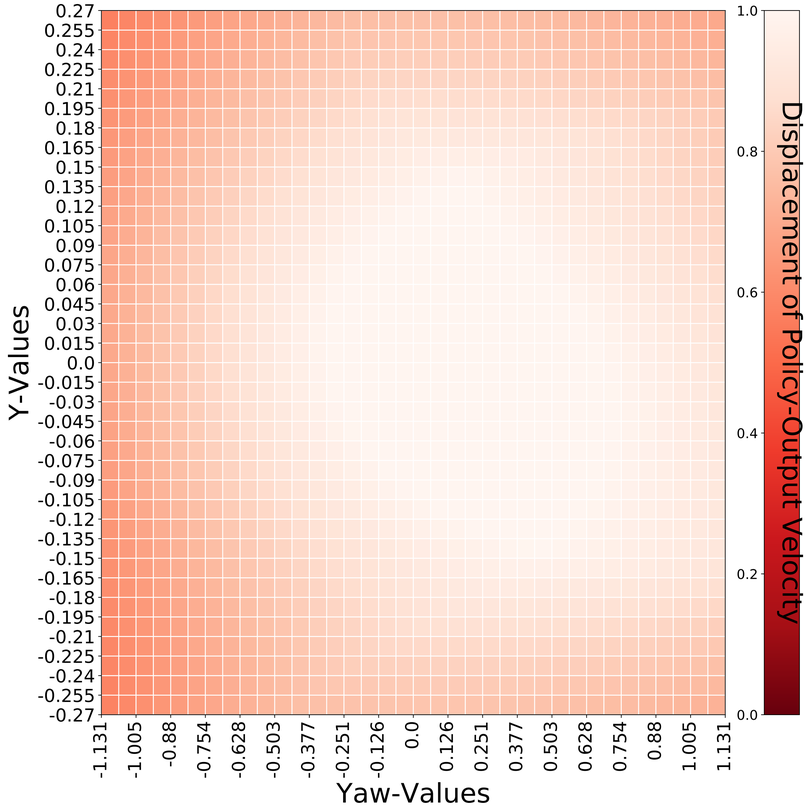
Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0.046153, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: e053ba15-4ec6-4b81-bdd9-0756e7b49dc7
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_49_20_153742

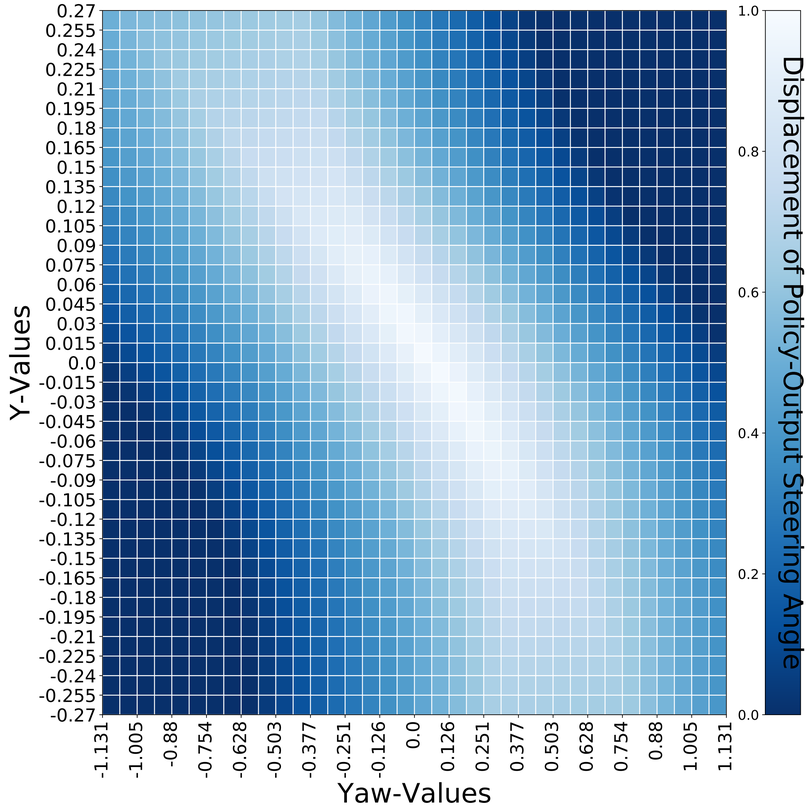




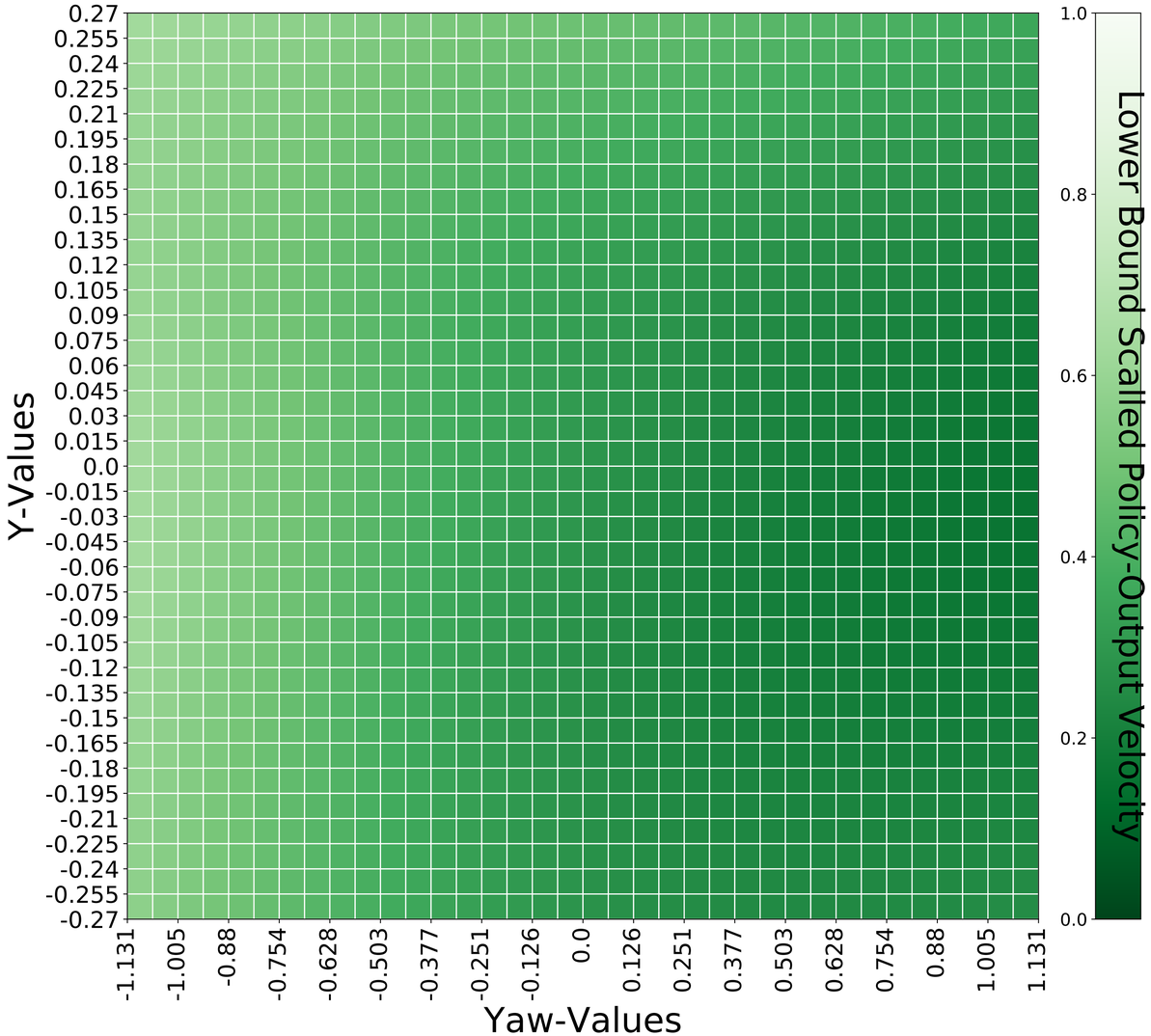
Line-Following Stability Results
xDot:0.300000, yDot:0.046153, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: dd907bb9-3ff9-4890-a700-df6009b96b75
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_48_14_272232

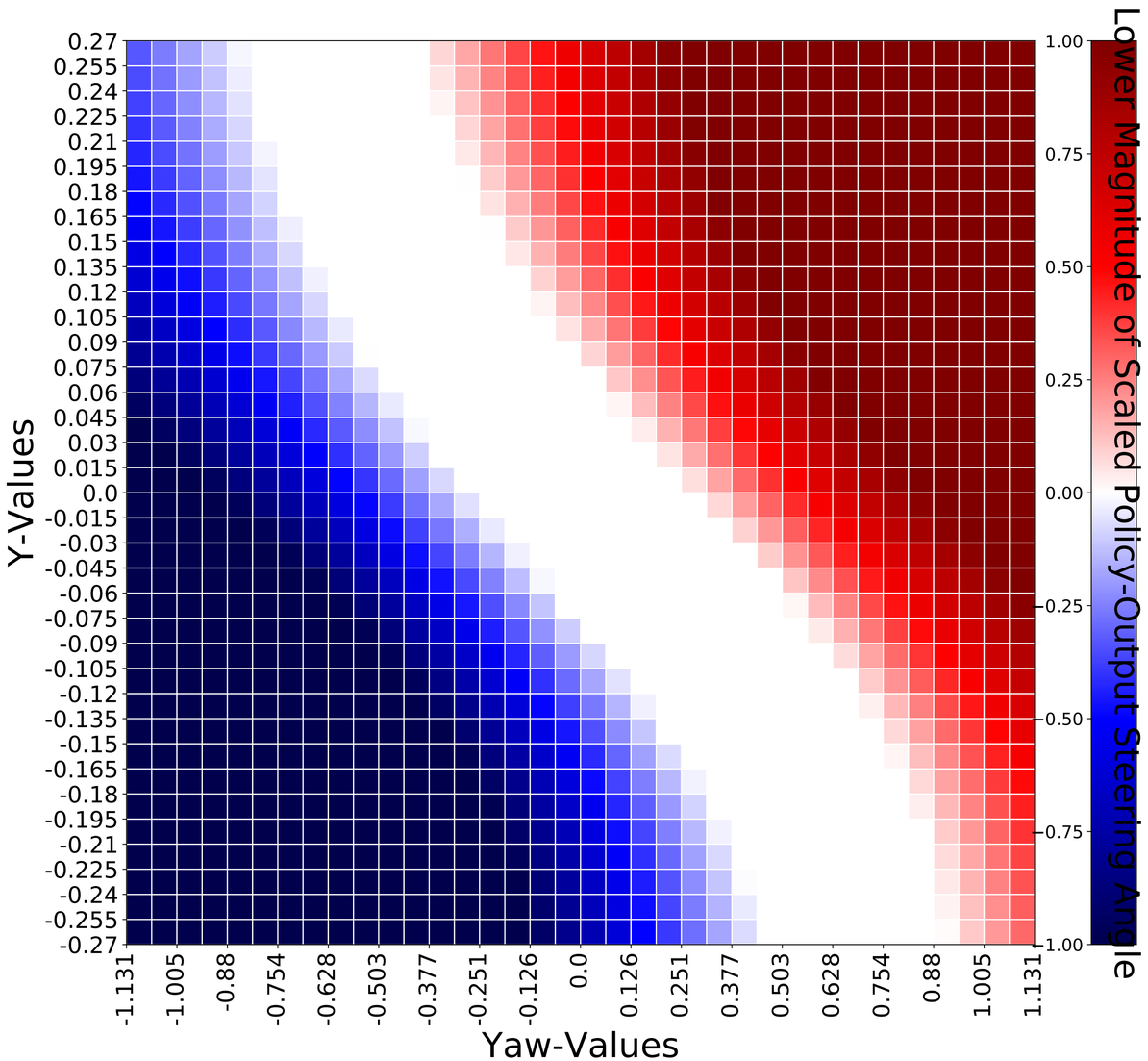




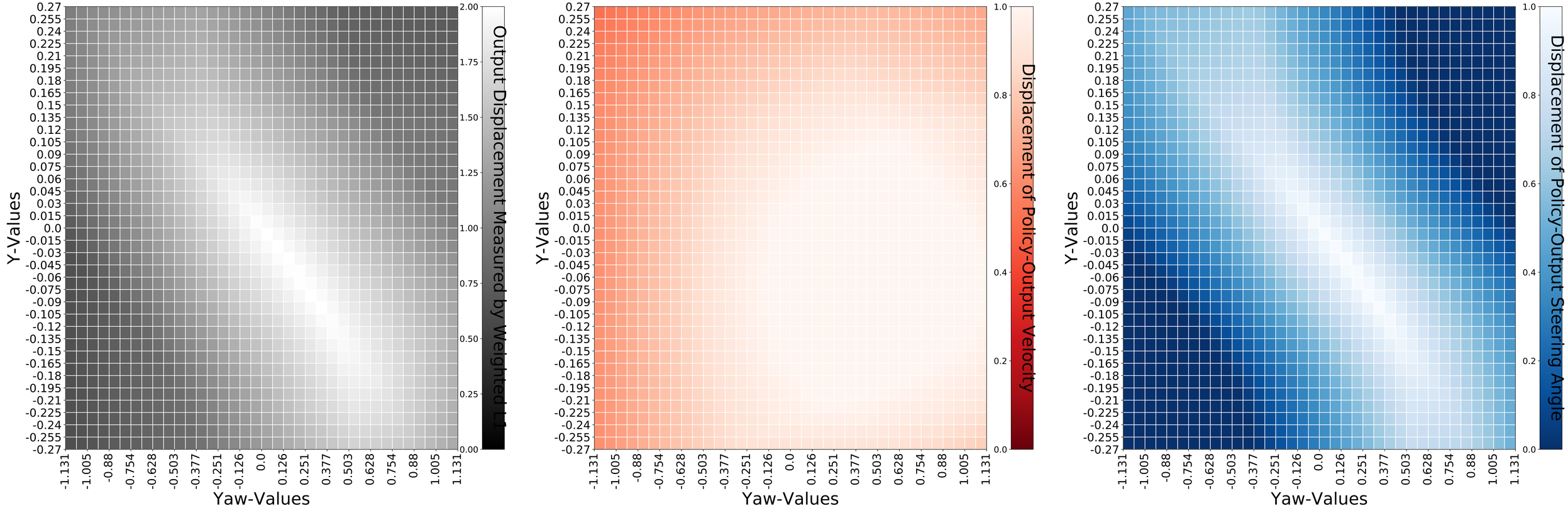


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0.046153, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 1197507a-8d7d-4a7a-9a68-633632a42d59
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_02_24_710555

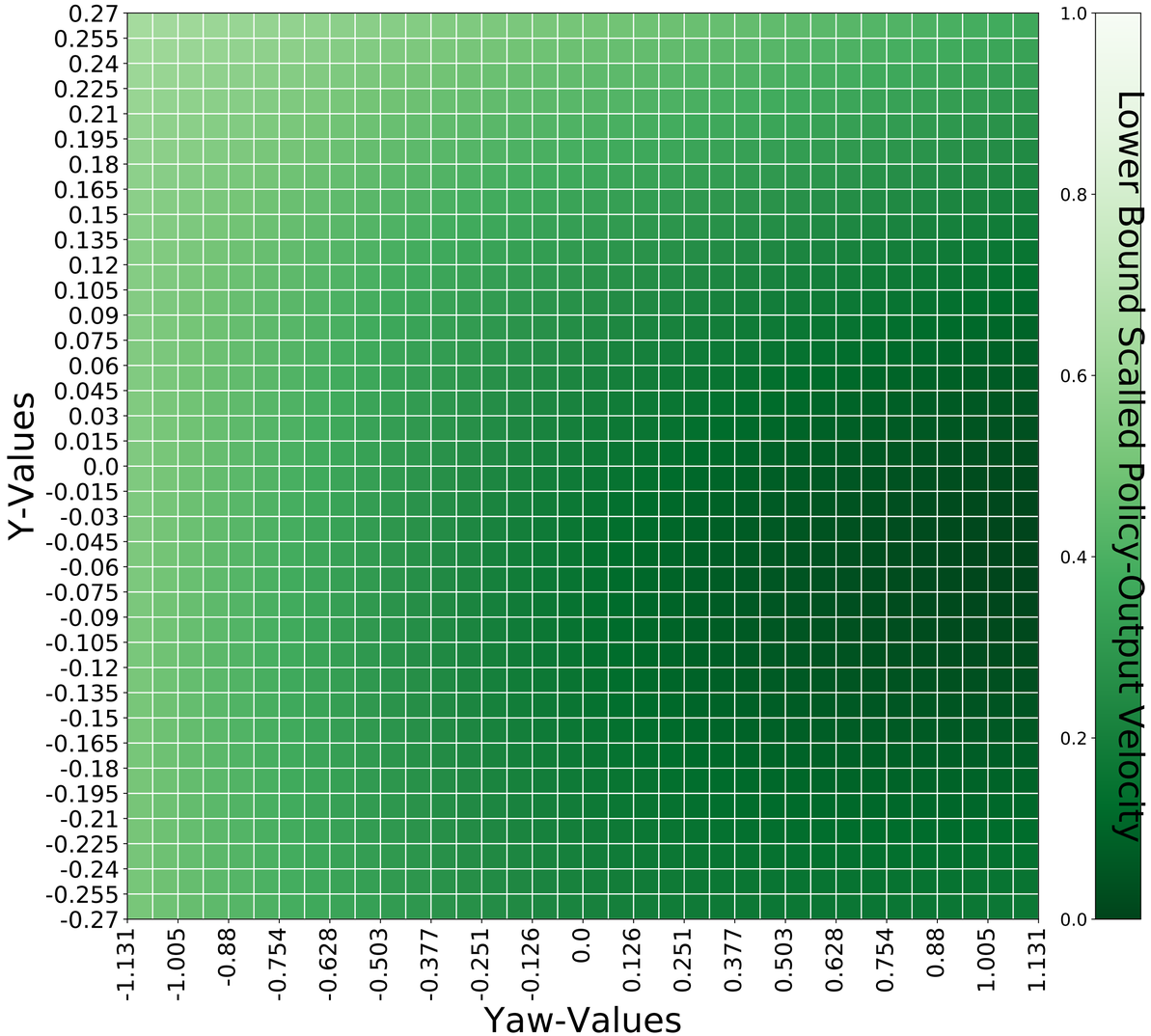


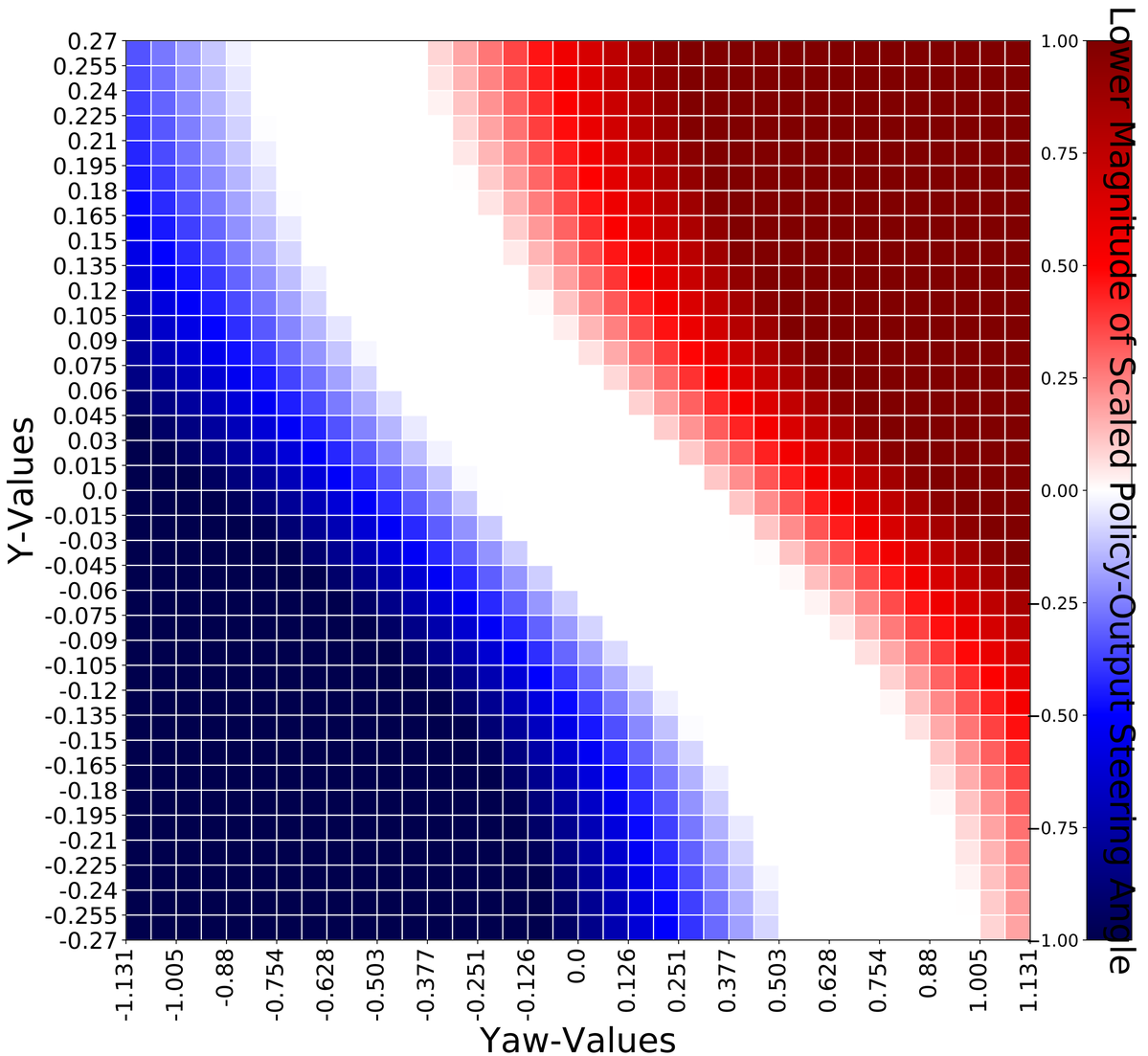


Line-Following Stability Results
xDot:0.300000, yDot:0.046153, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: e92923d1-6814-4ca6-a0df-cdcc29e89983
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_01_19_994644

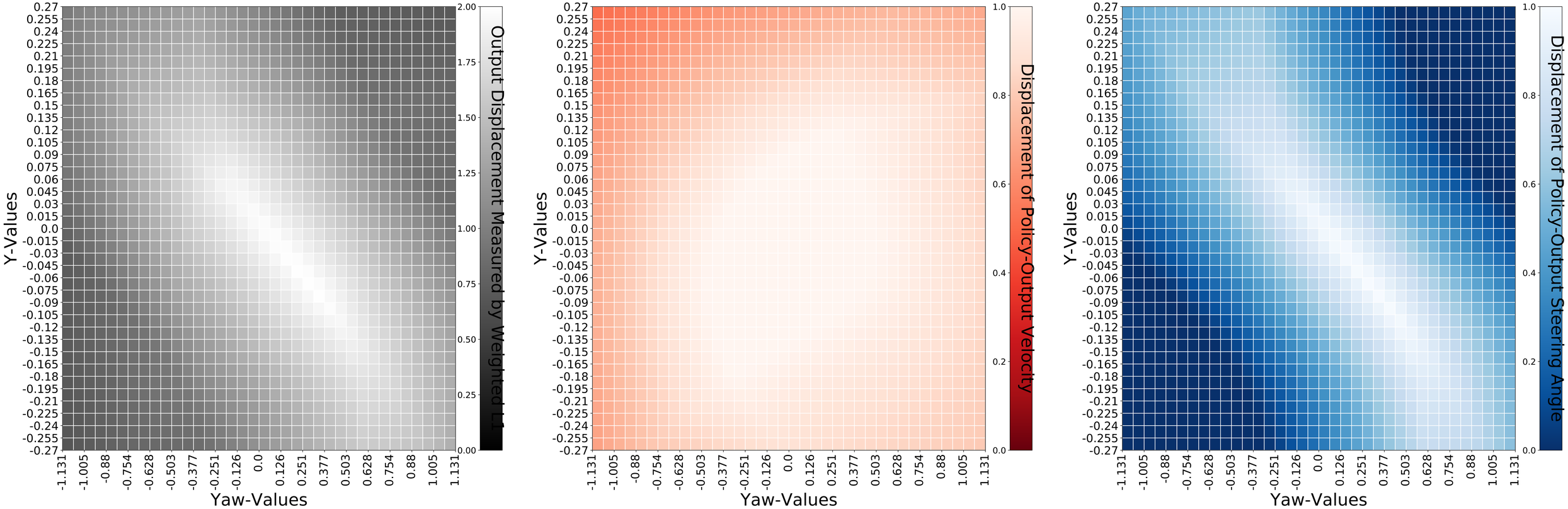


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:-0.32307, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: c75200c6-7dc9-4fb6-9e3a-8e5a2ea2fbfa
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_04_34_462741

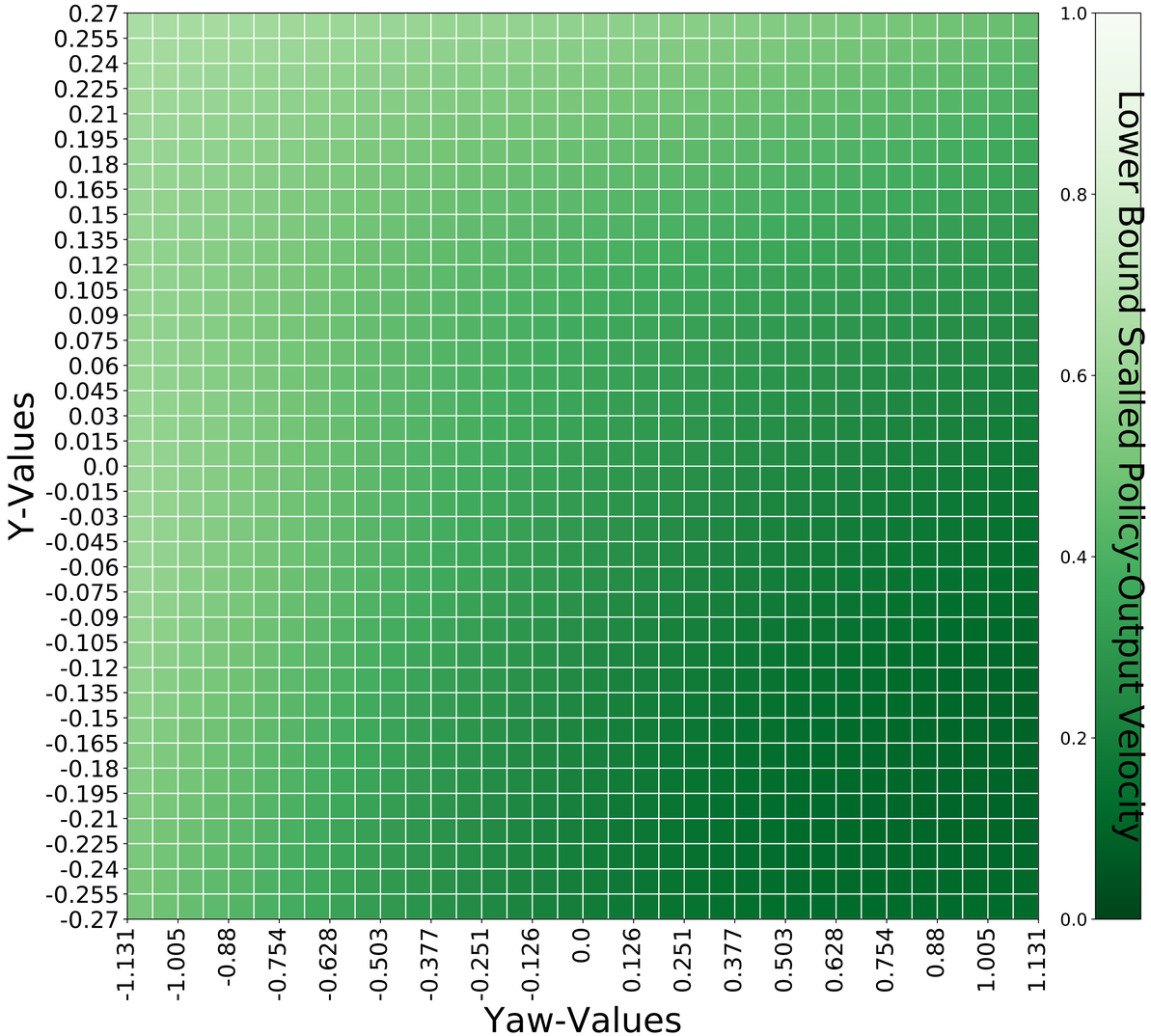


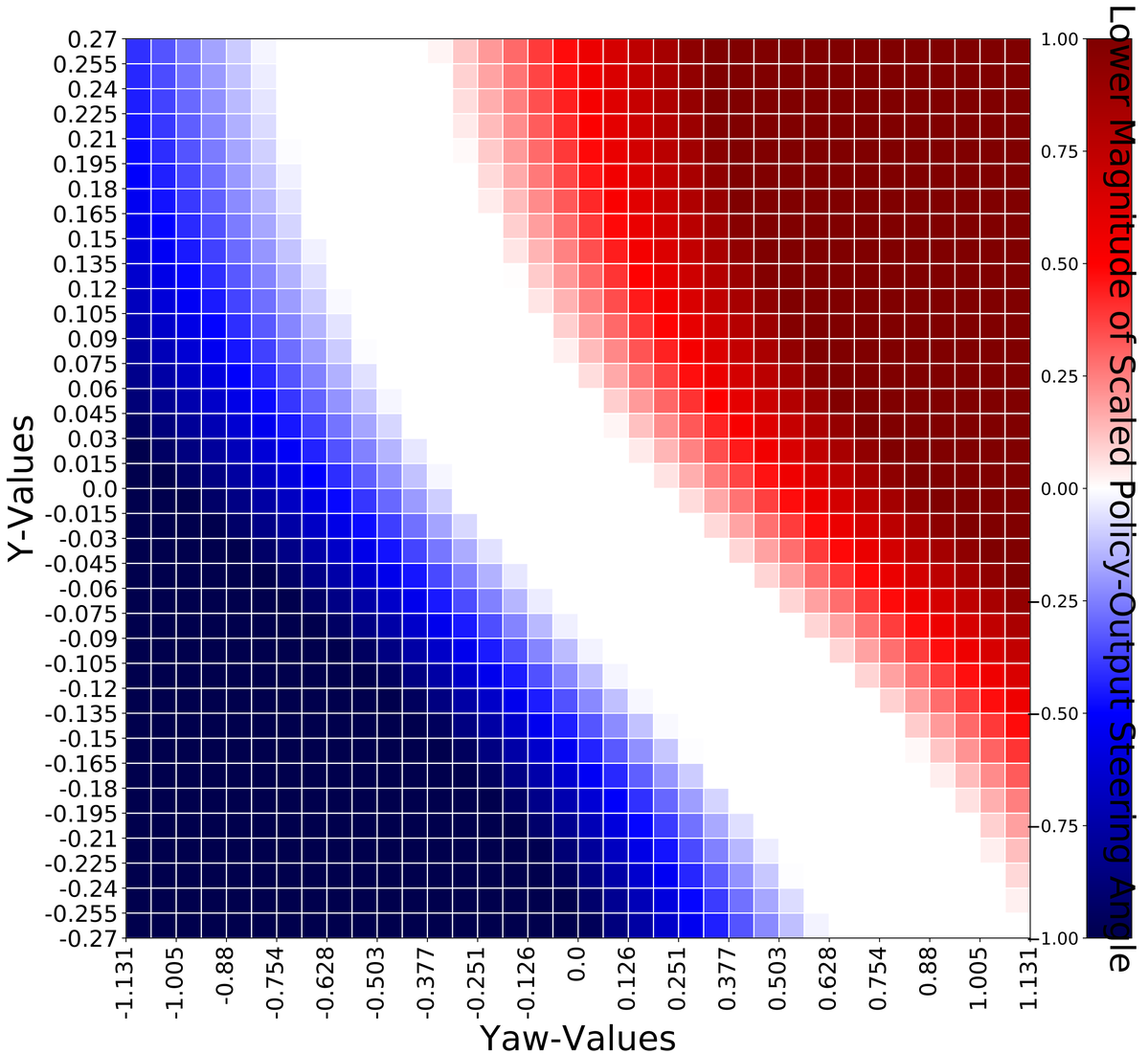


Line-Following Stability Results
xDot:0.300000, yDot:-0.32307, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: c204d01f-91c2-4778-9fb1-307fd5afd0cc
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_03_28_226949

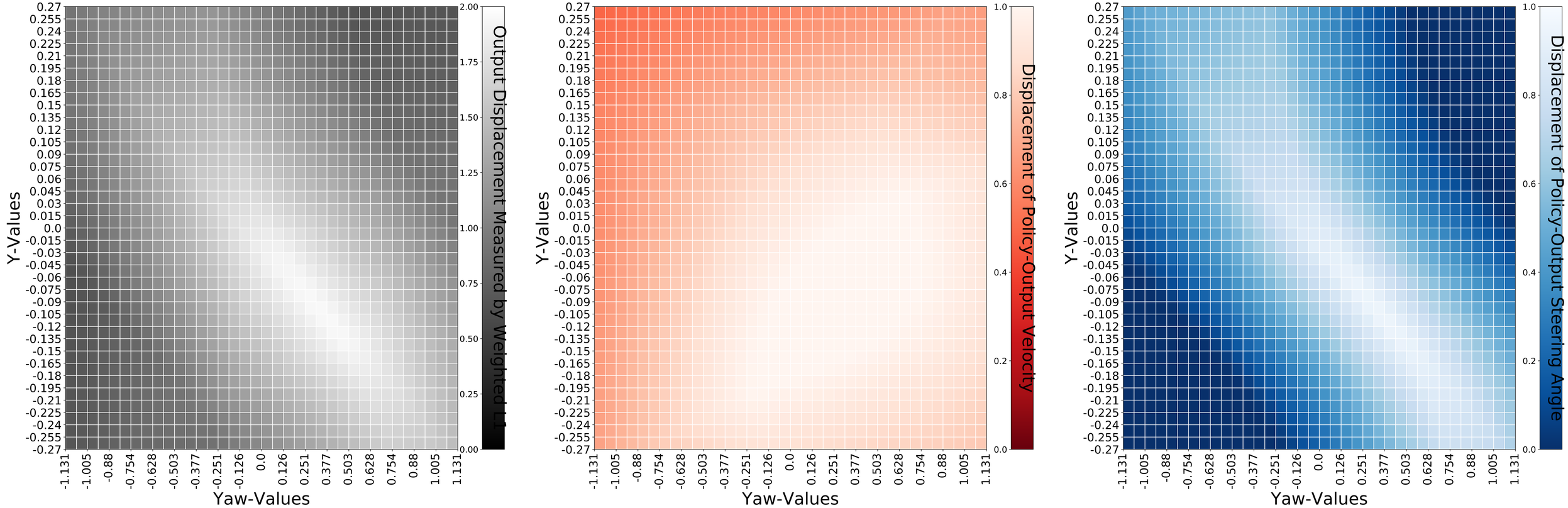


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:-0.32307, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 18e9280e-575a-41e4-ad9d-7db42004742c
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_06_45_814212

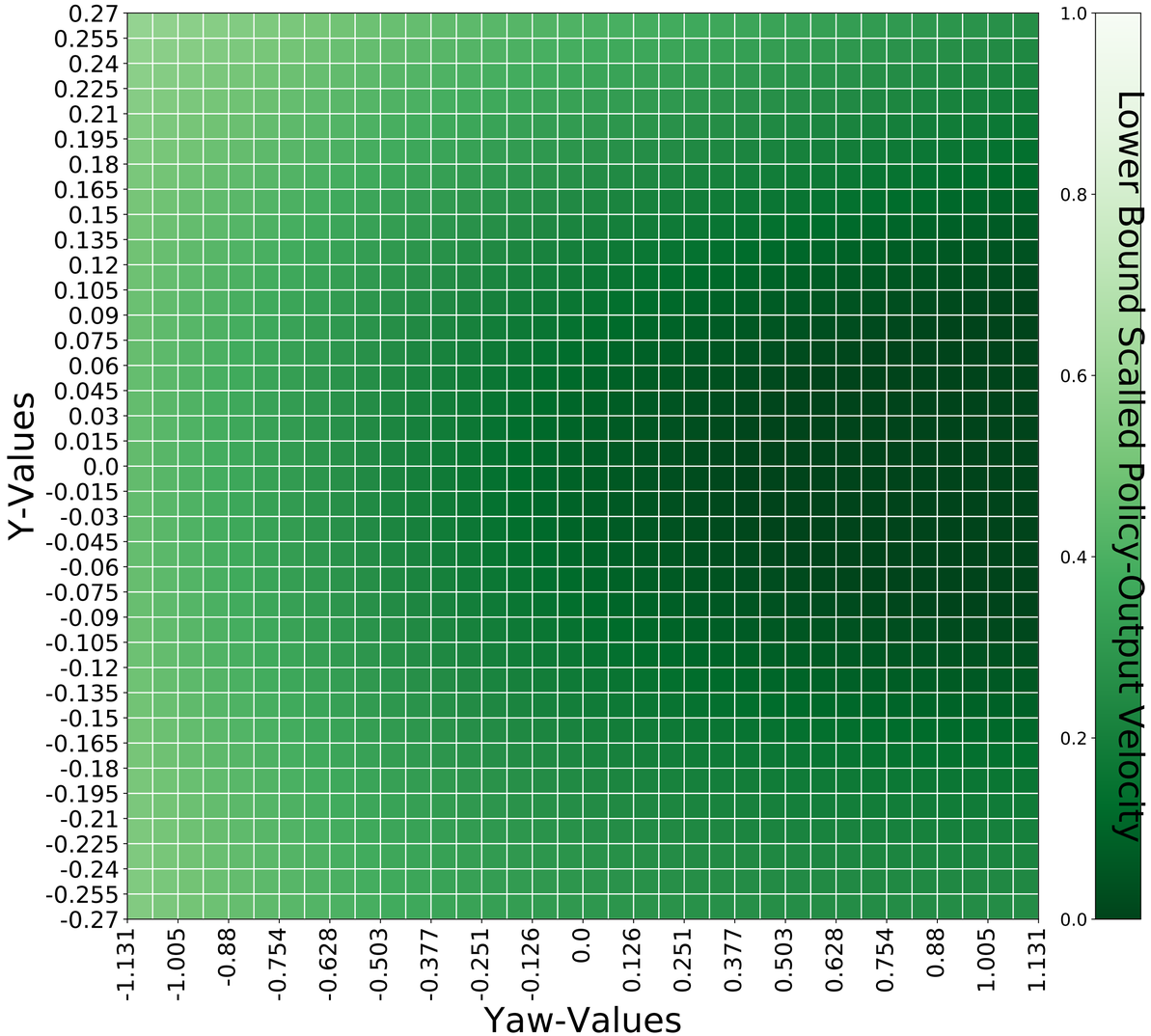


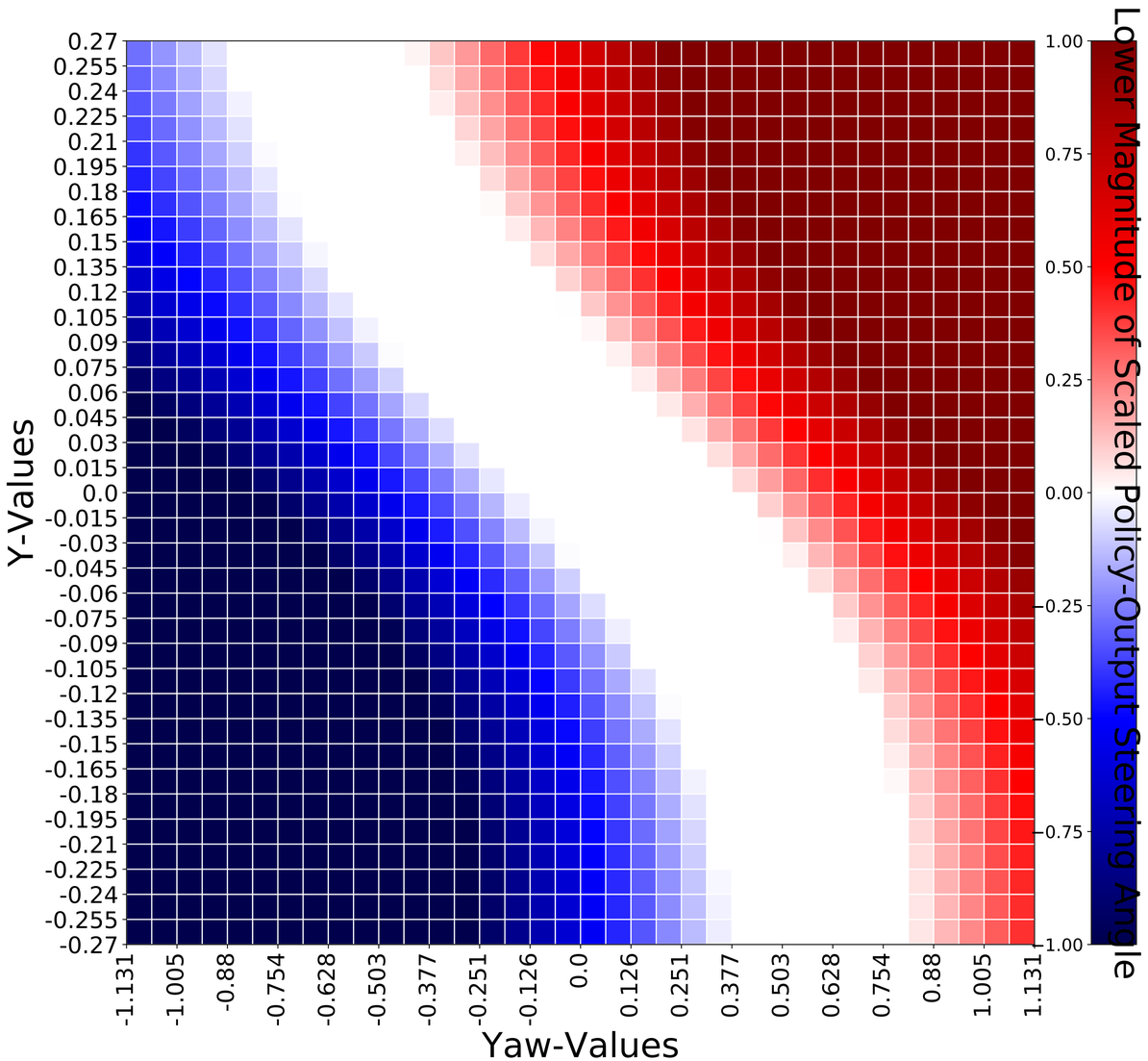


Line-Following Stability Results
xDot:0.300000, yDot:-0.32307, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 53a3810a-0b29-4f67-9f89-3db175f5f6dc
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_05_39_406576

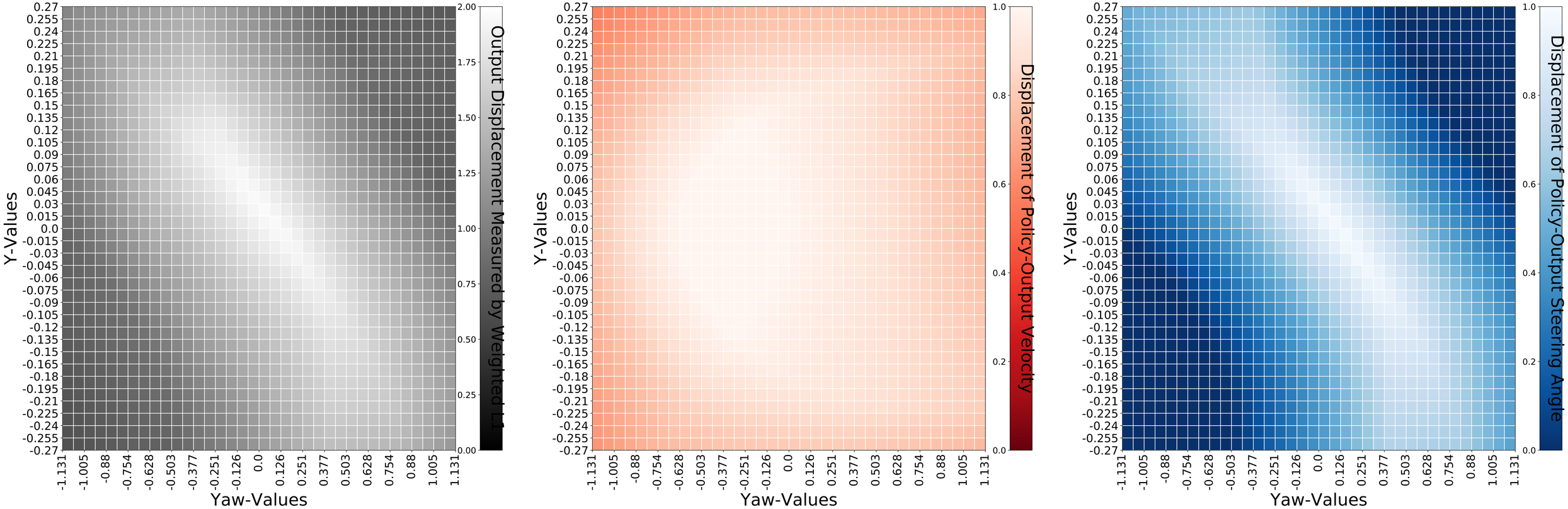


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:-0.32307, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 4a135b82-571a-45f1-9197-7c0b7eb51175
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_08_56_511875

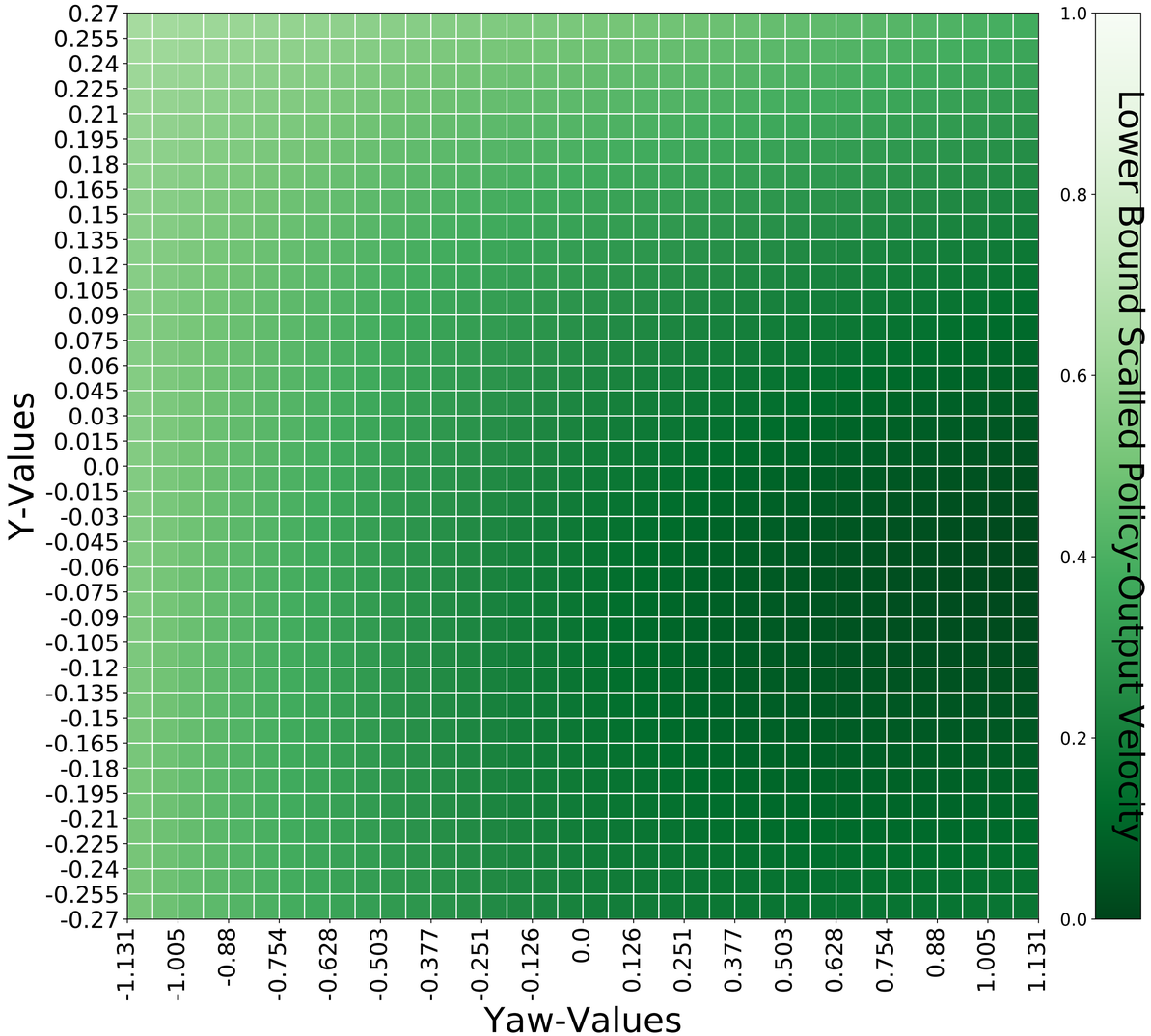


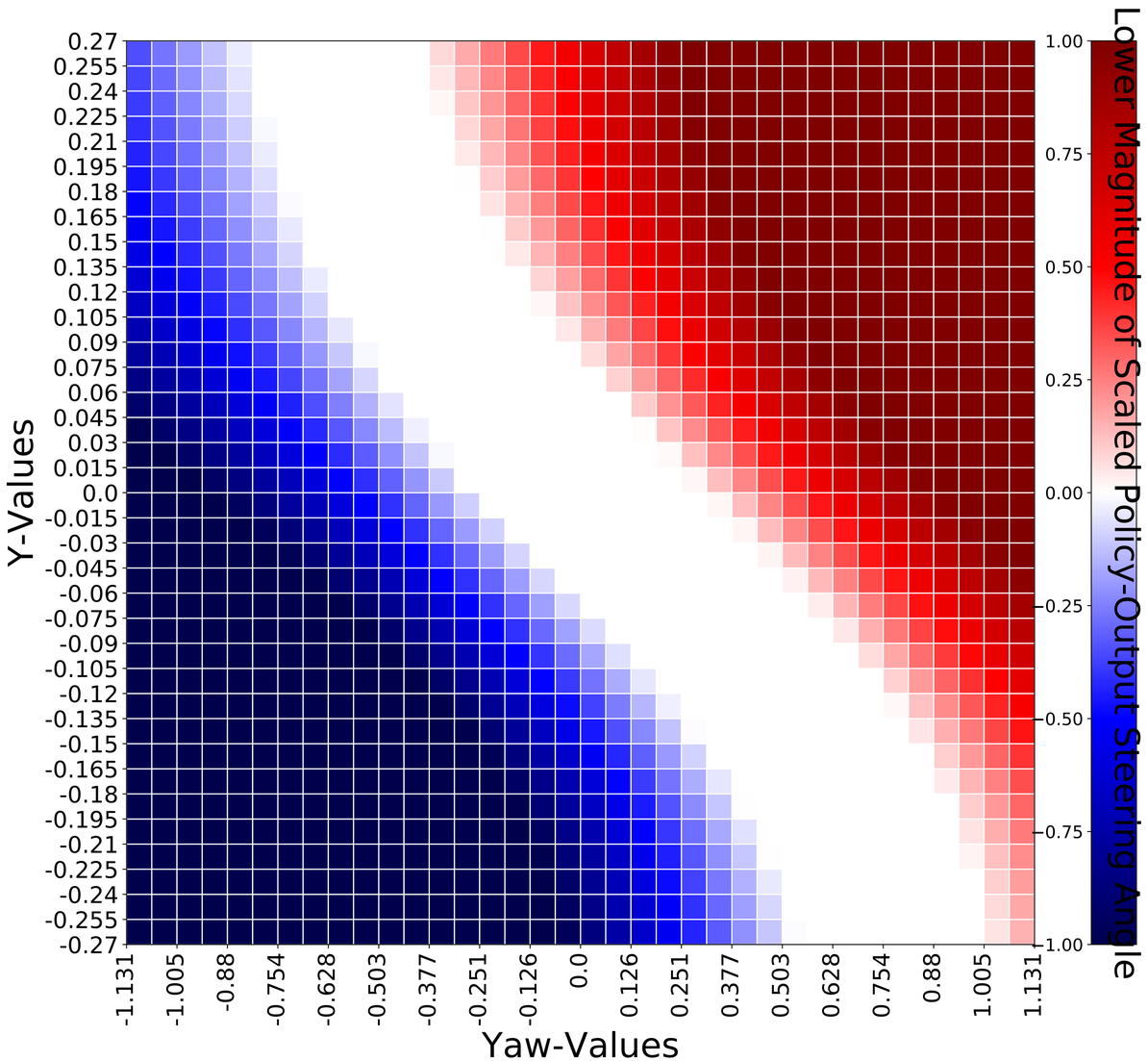


Line-Following Stability Results
xDot:0.300000, yDot:-0.32307, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 86665bfa-d1a1-4e3e-b486-c04e5bf24073
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_07_50_443175

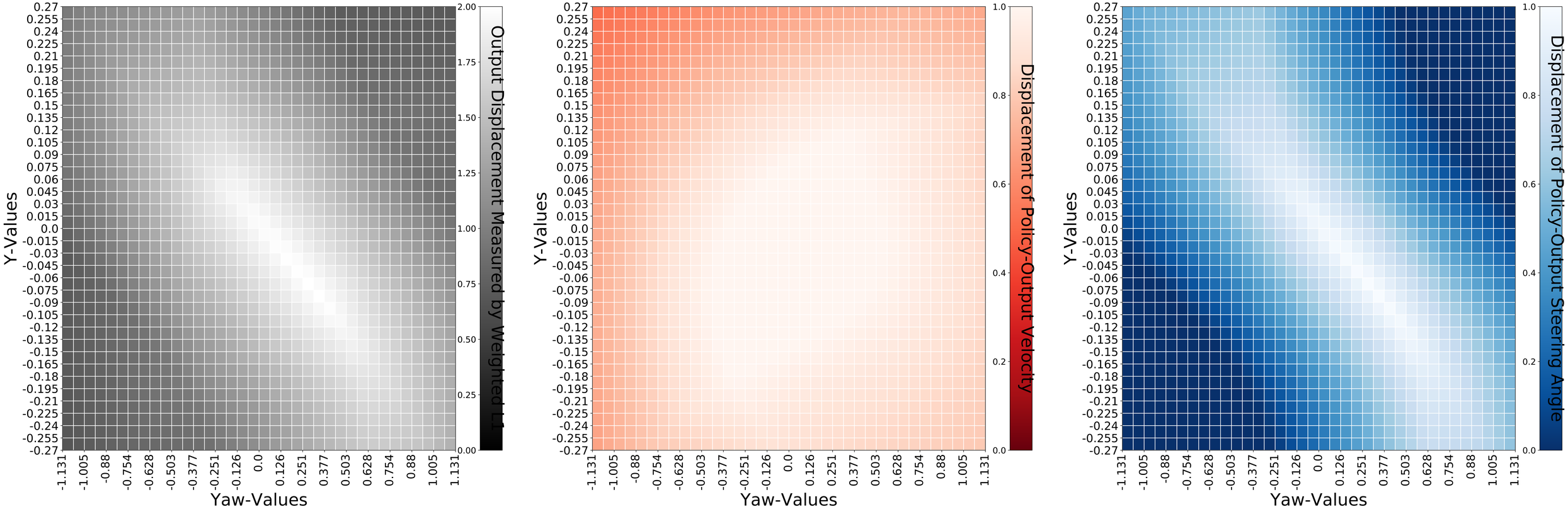


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:-0.32307, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 759b77c4-7f89-4b02-a733-b673a688b483
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_22_03_507103

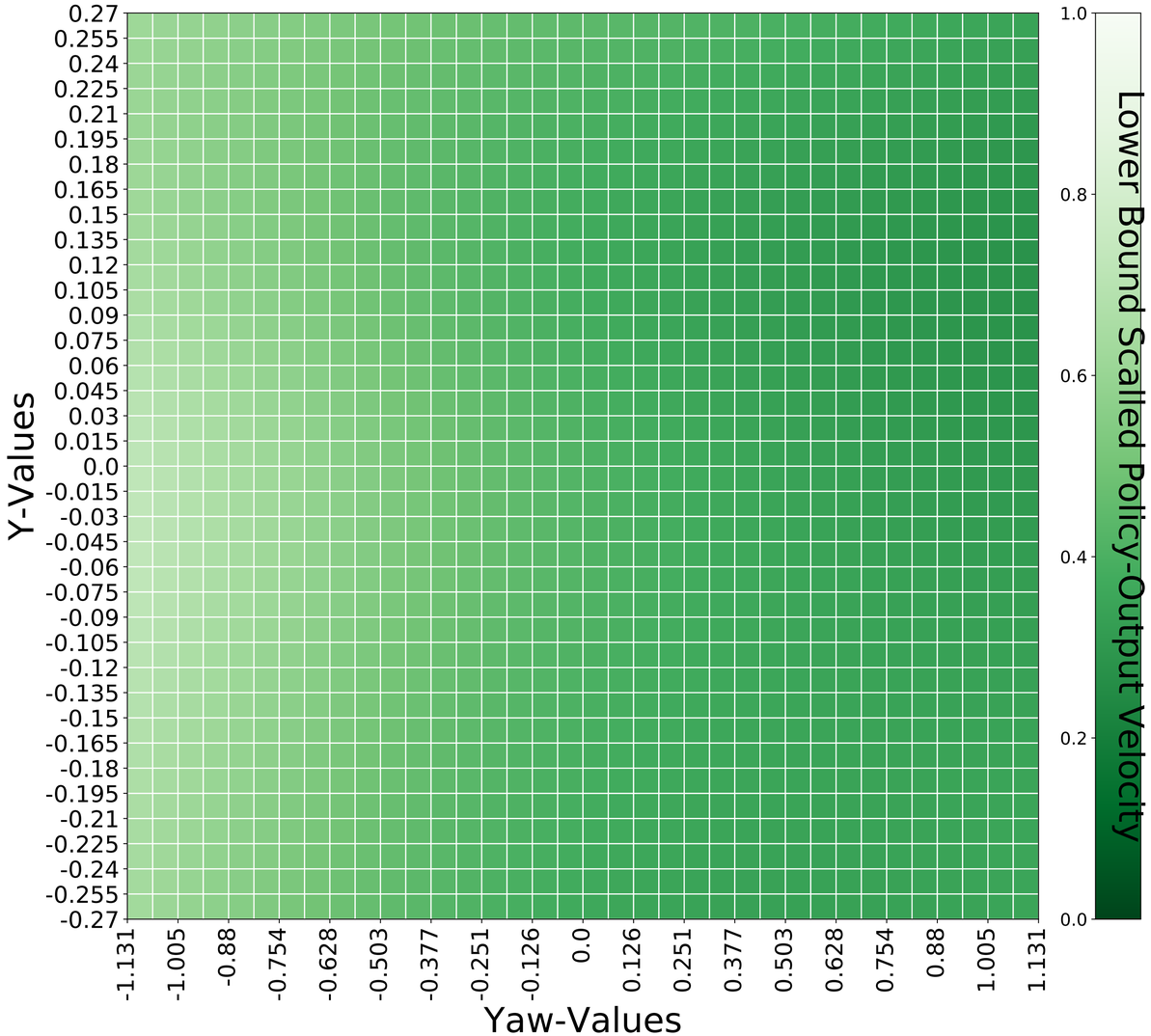


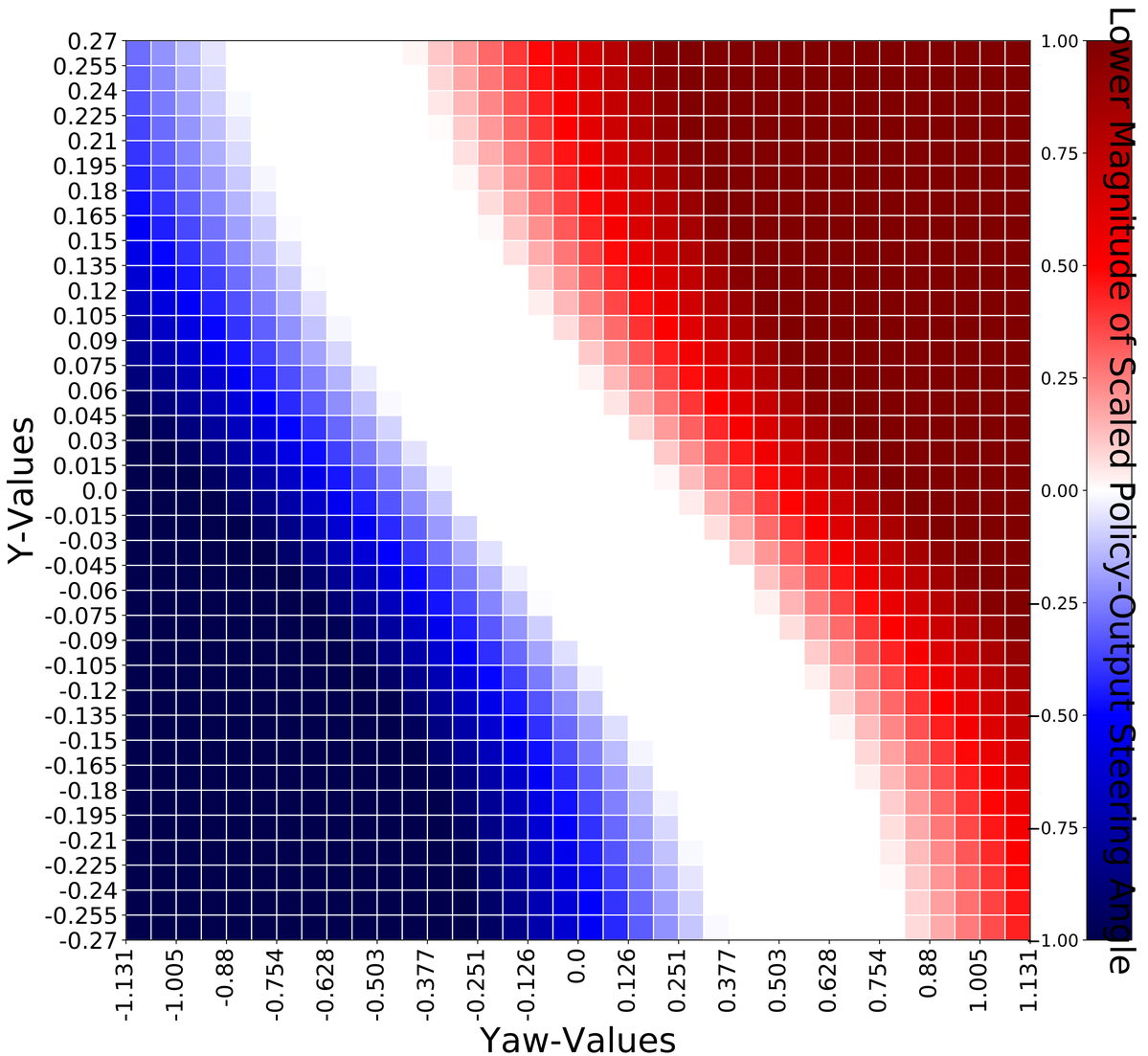


Line-Following Stability Results
xDot:0.300000, yDot:-0.32307, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 8ad3ebbf-f1c2-458a-a597-65f1472523fb
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_20_57_579317

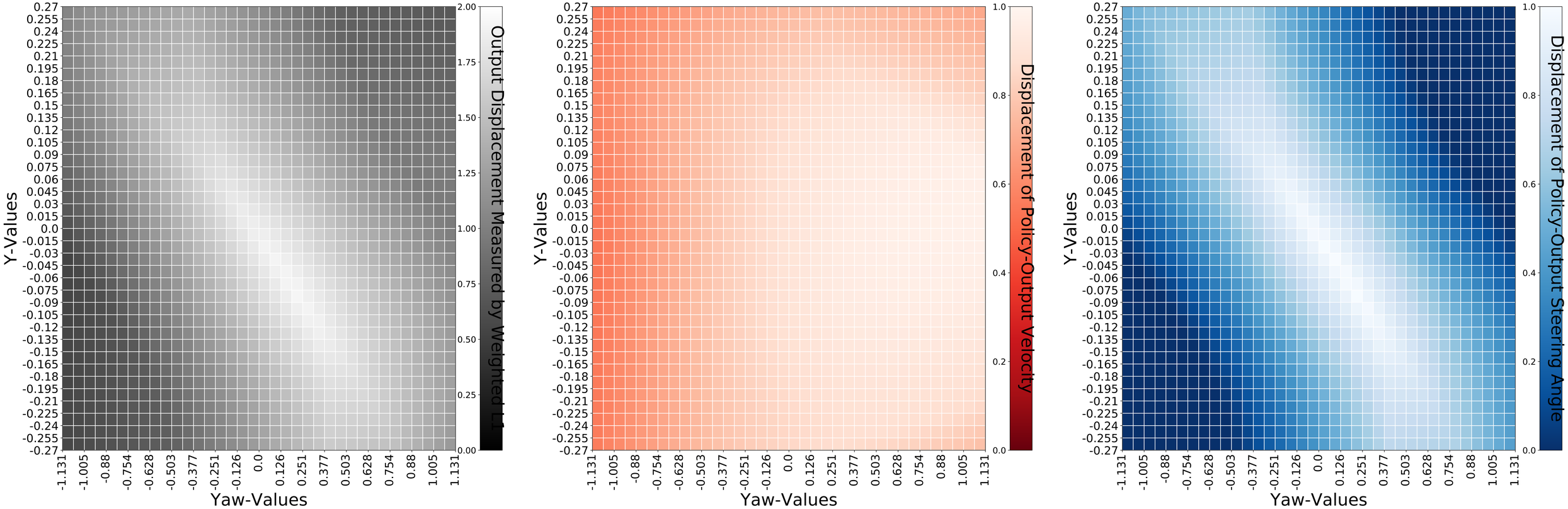


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0.415384, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 33a90b04-fd9b-49dd-8bec-3b187beb370d
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_24_14_742987

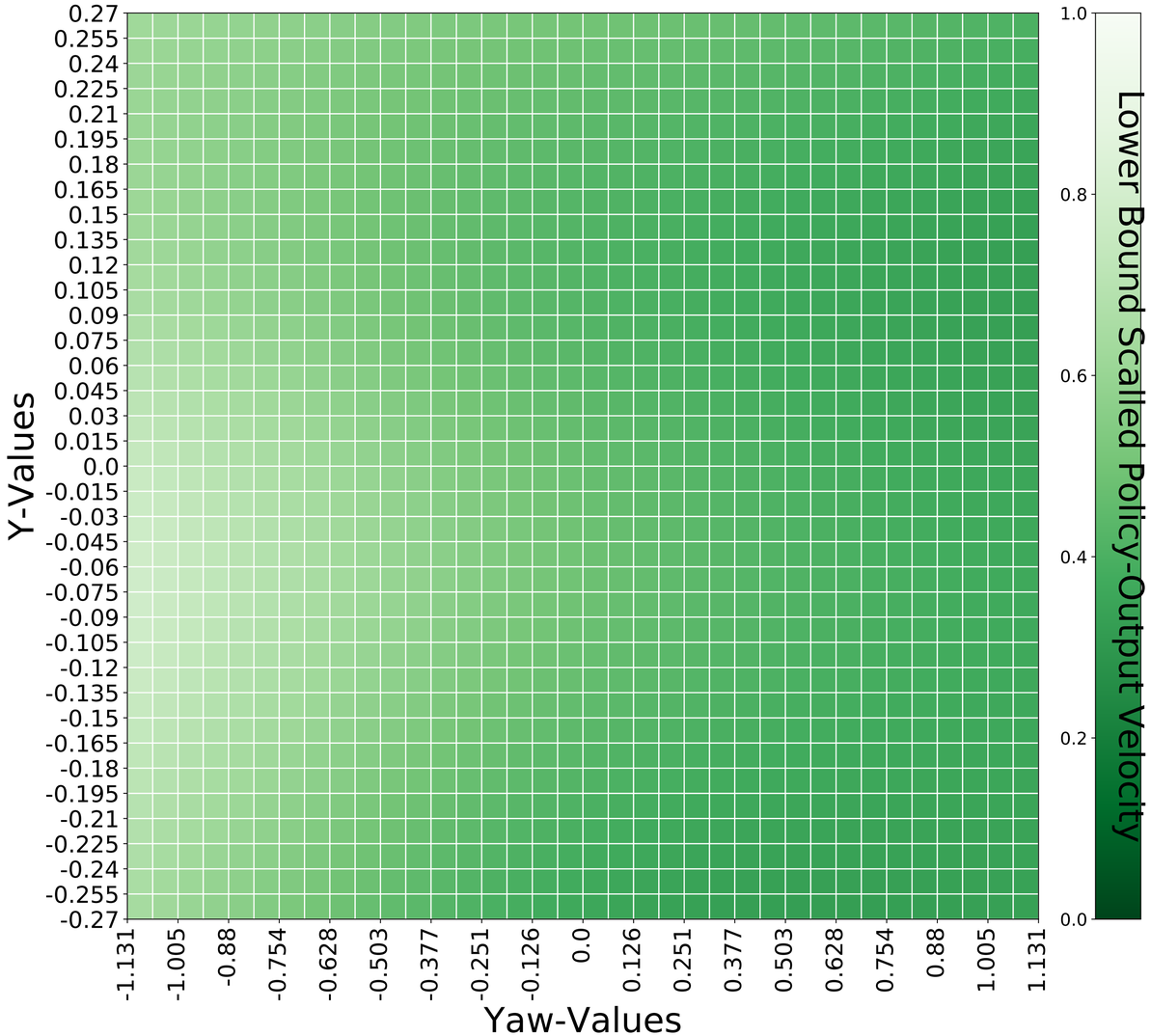


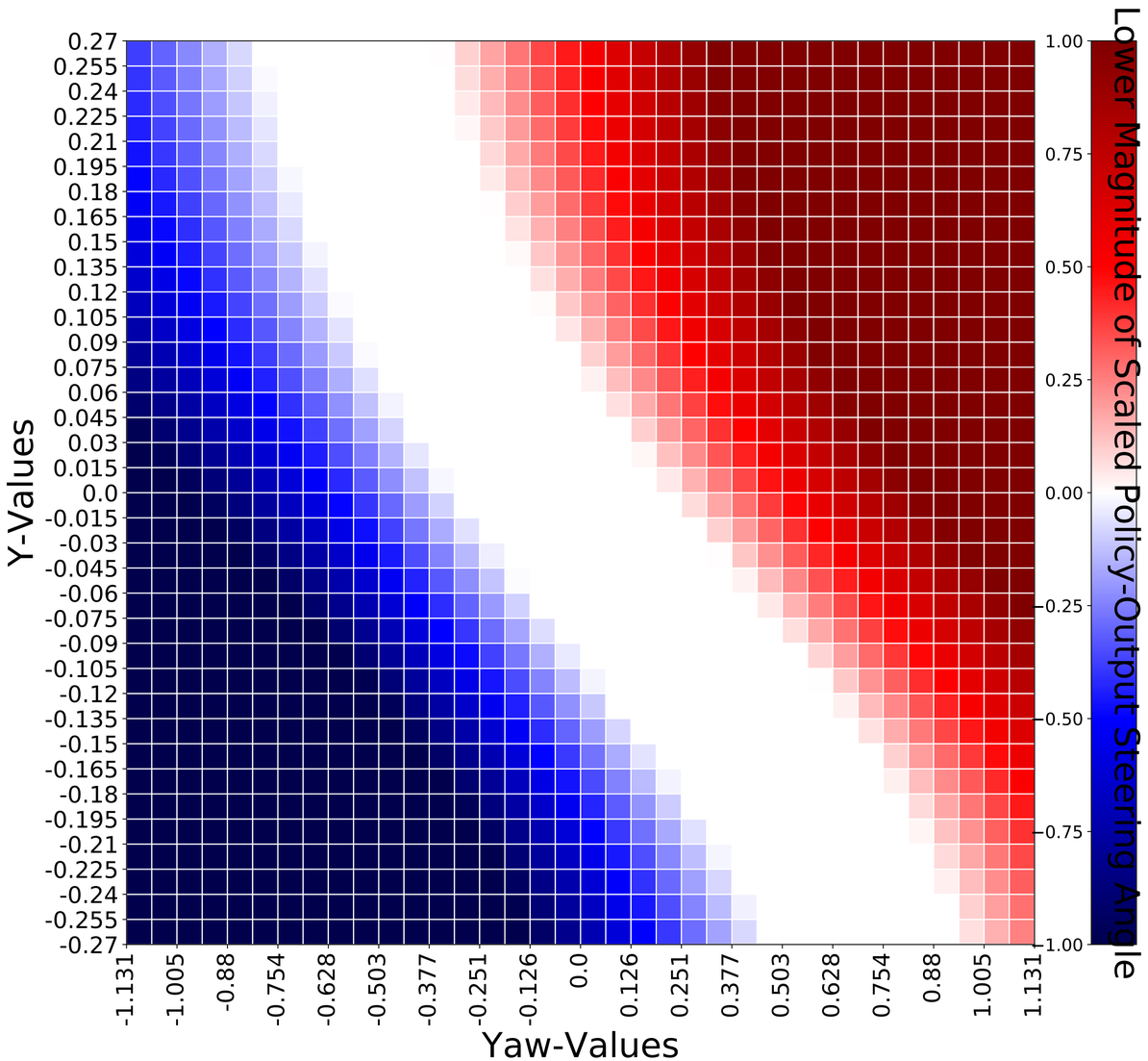


Line-Following Stability Results
xDot:0.300000, yDot:0.415384, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 0f91abef-30ba-4bfe-9513-6277f6af6161
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_23_08_164343

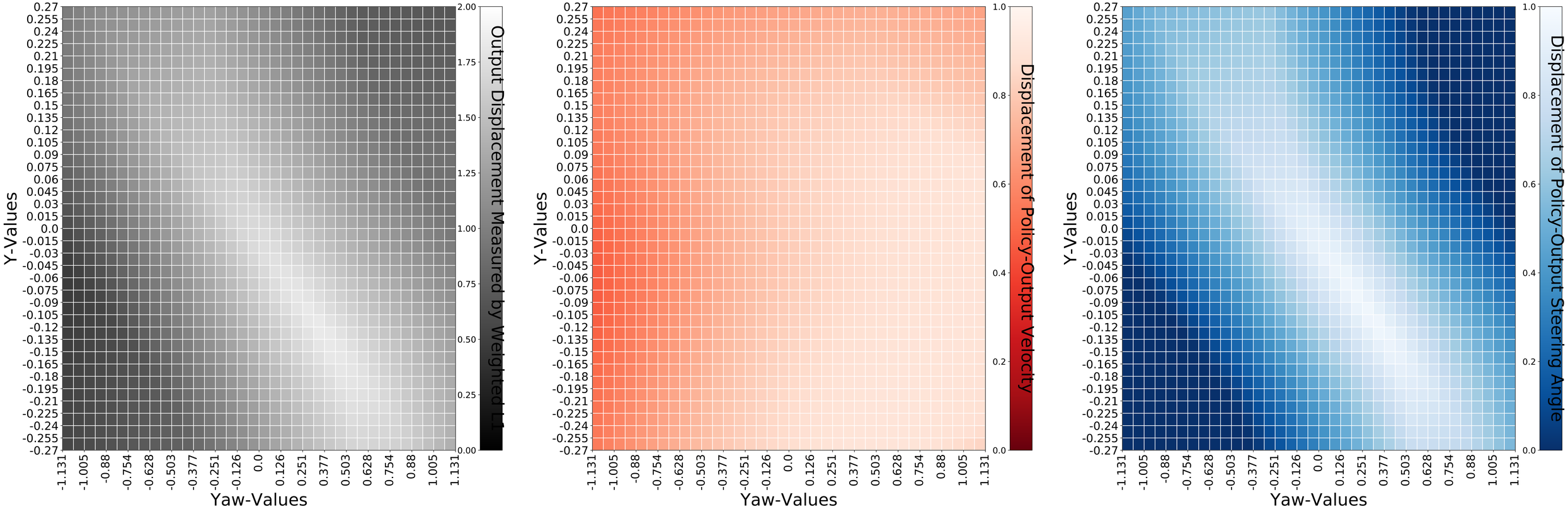


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0.415384, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 6aa1fb98-bcb6-4aee-b7bb-a9c922db88d5
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_26_26_093533

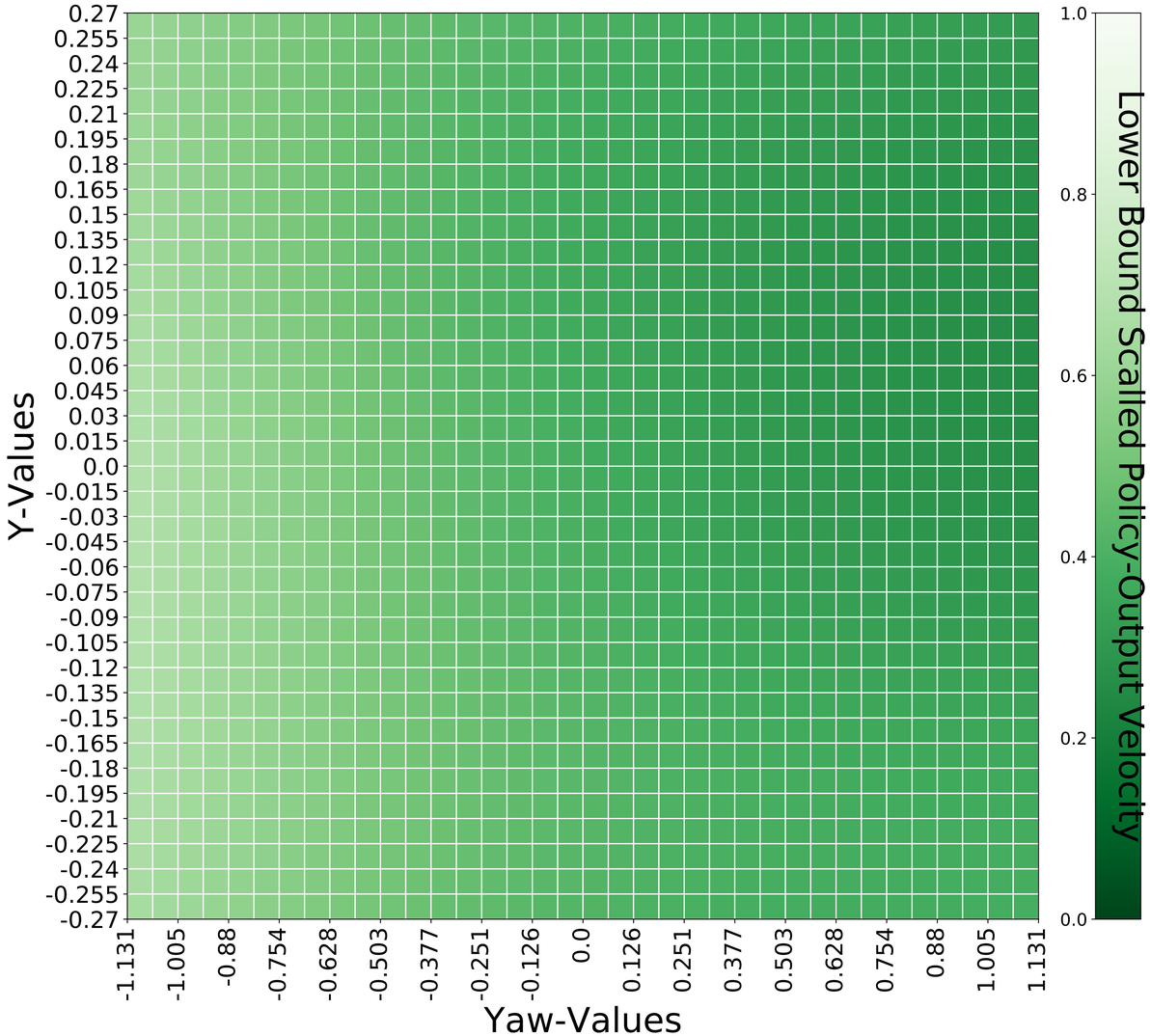


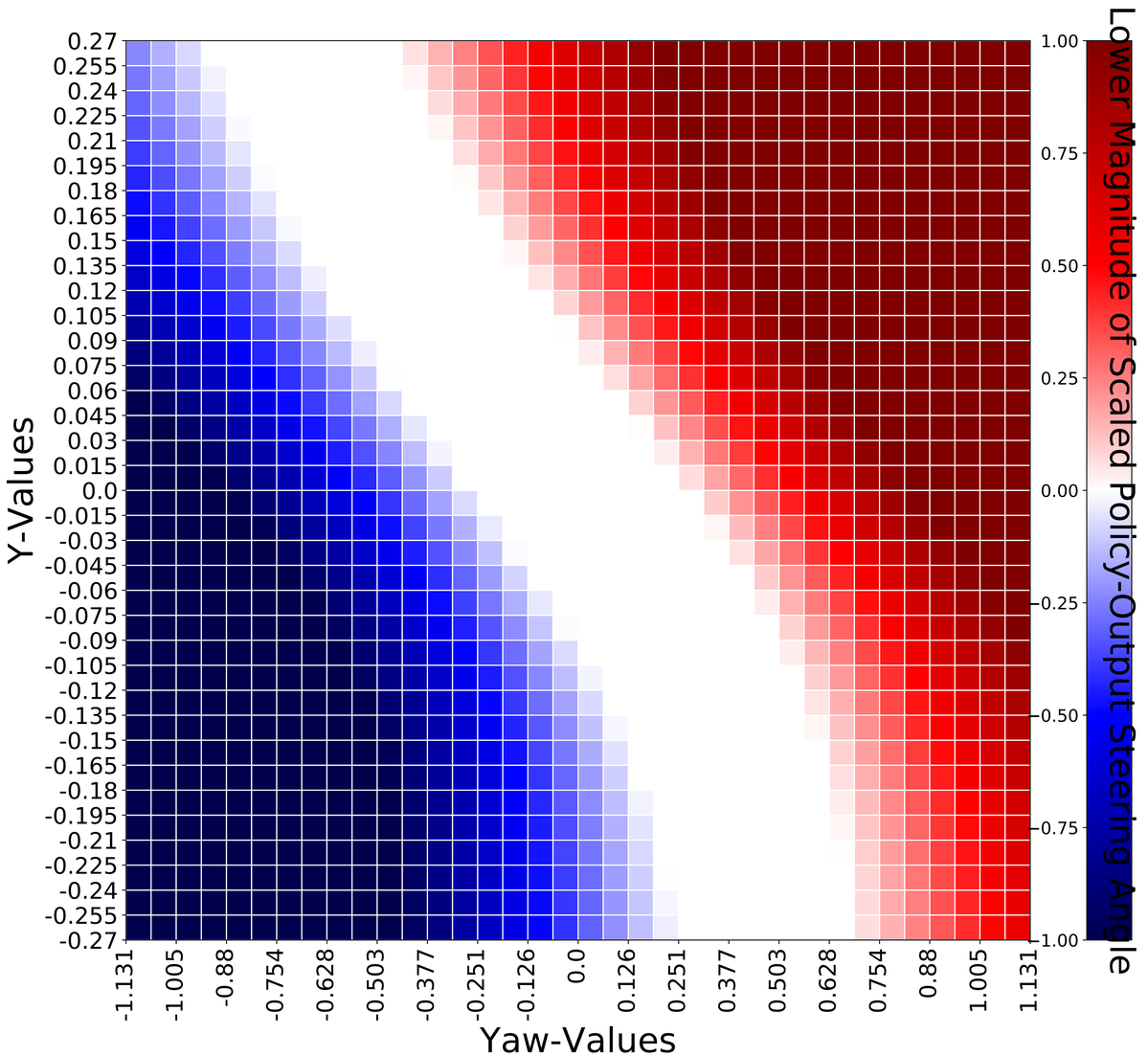


Line-Following Stability Results
xDot:0.300000, yDot:0.415384, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 6350d65f-3b9c-485f-bff6-c392568c6236
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_25_19_755153

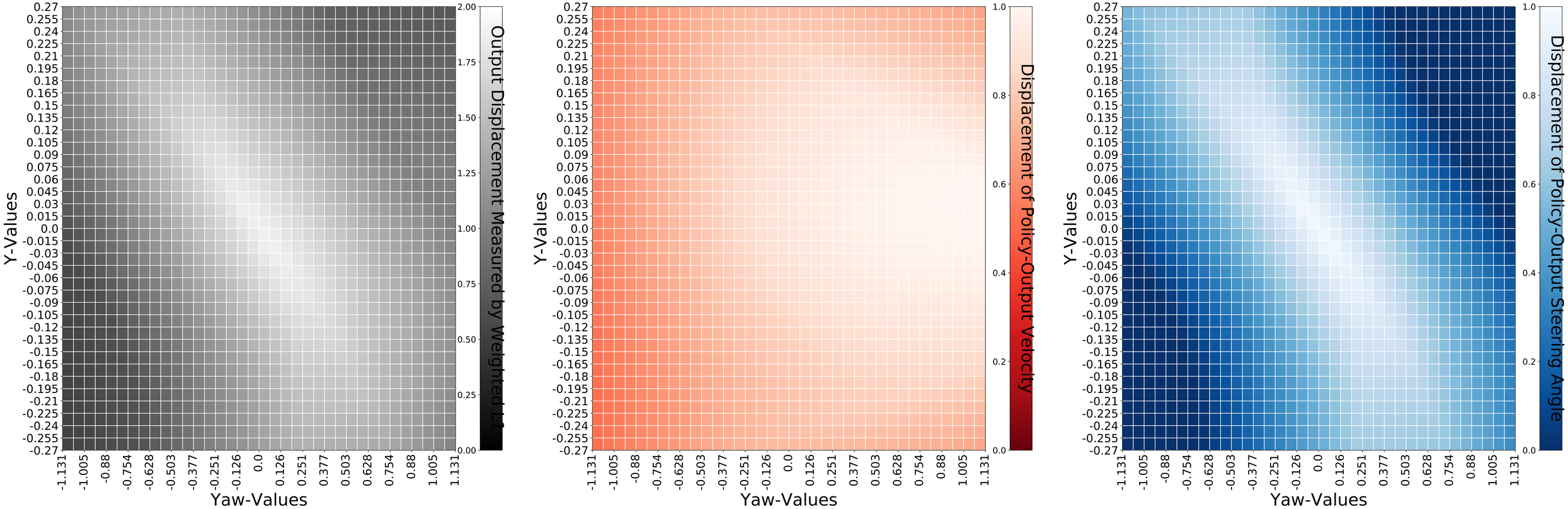


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0.415384, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 59802a94-29a4-43fa-8b17-06ad06f7de1d
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_28_37_153752

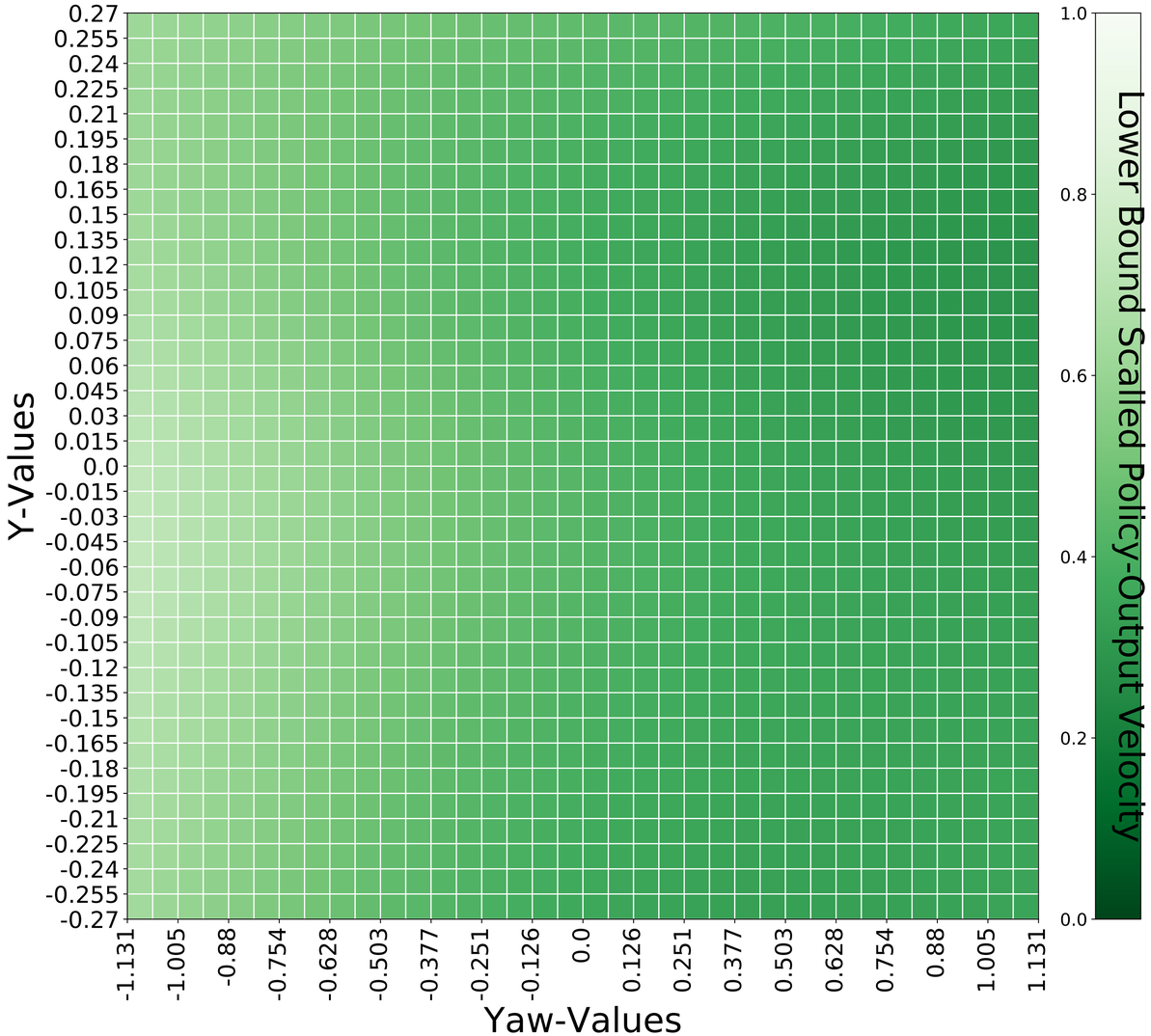


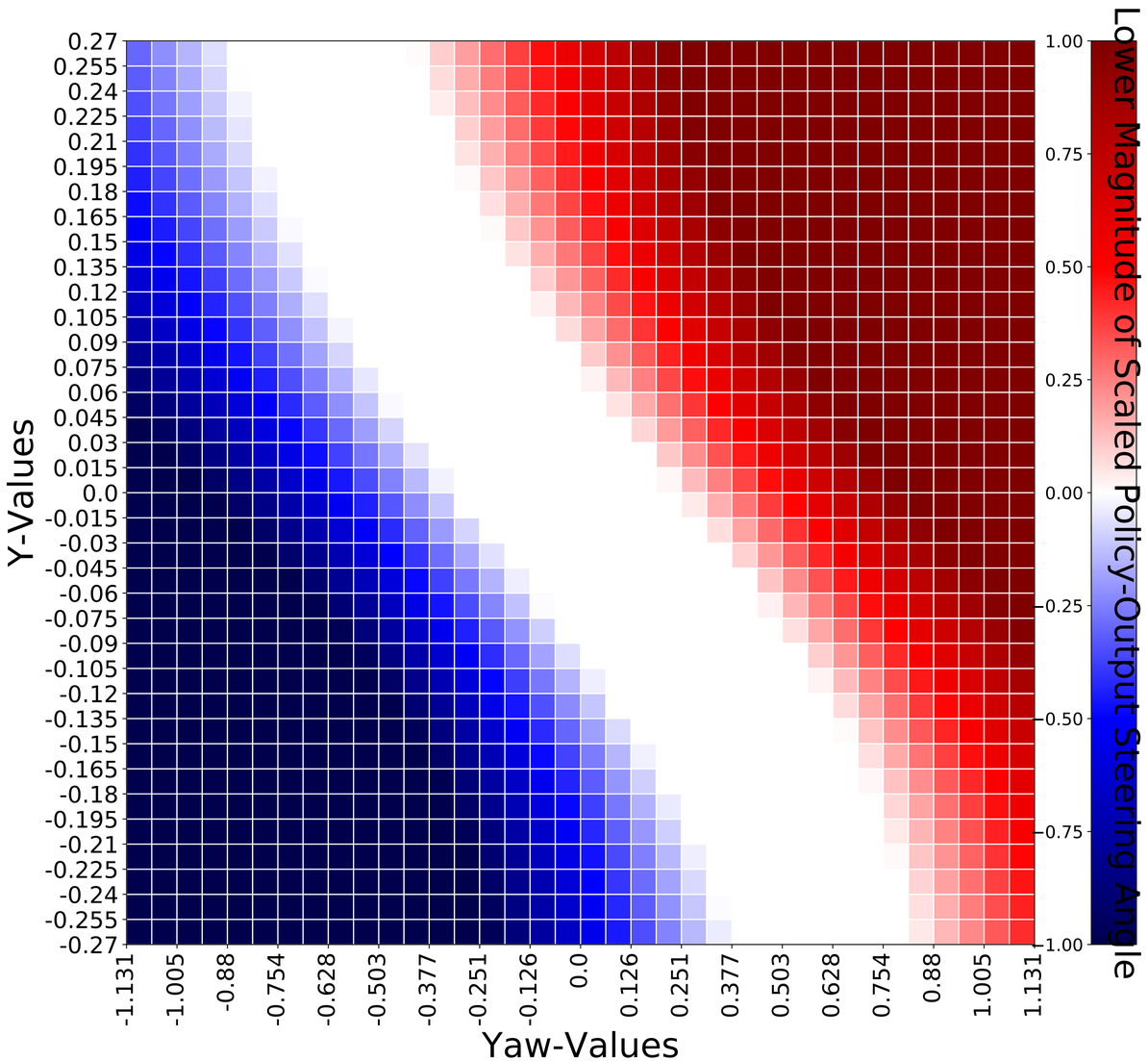


Line-Following Stability Results
xDot:0.300000, yDot:0.415384, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 52cc99c1-2ddf-4463-8dd1-aaabefdfb1b6
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_27_31_028191

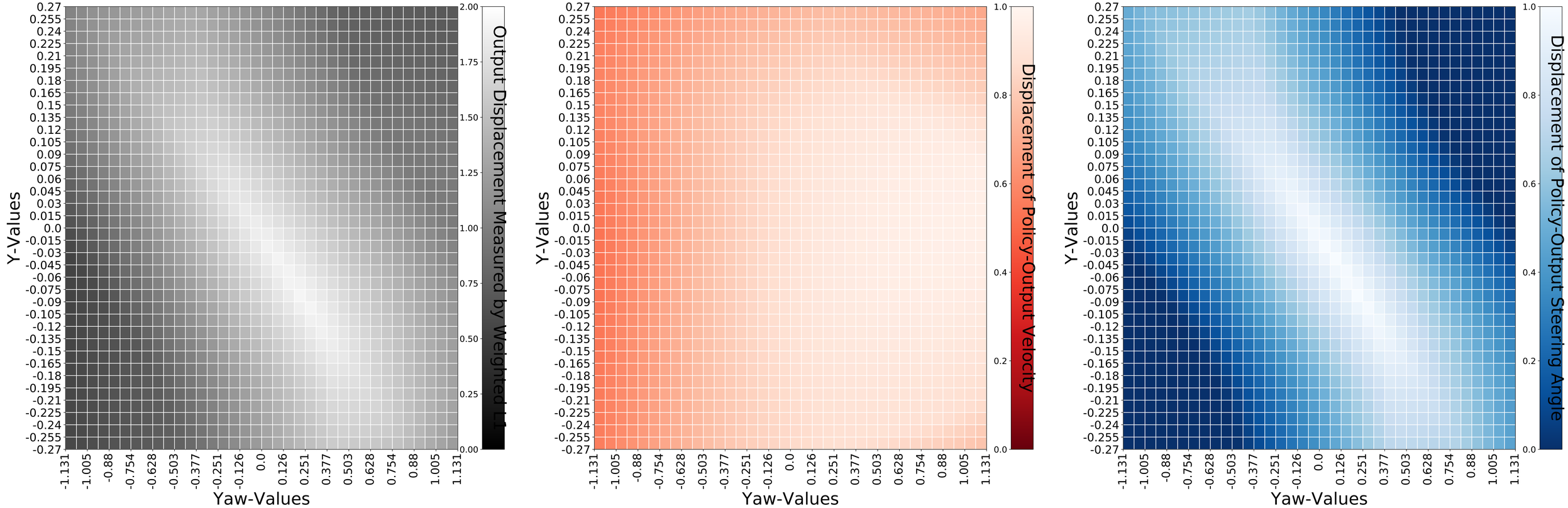


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0.415384, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 63bbe972-5472-41d4-a4f3-8efedf15d6c2
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_41_31_345664

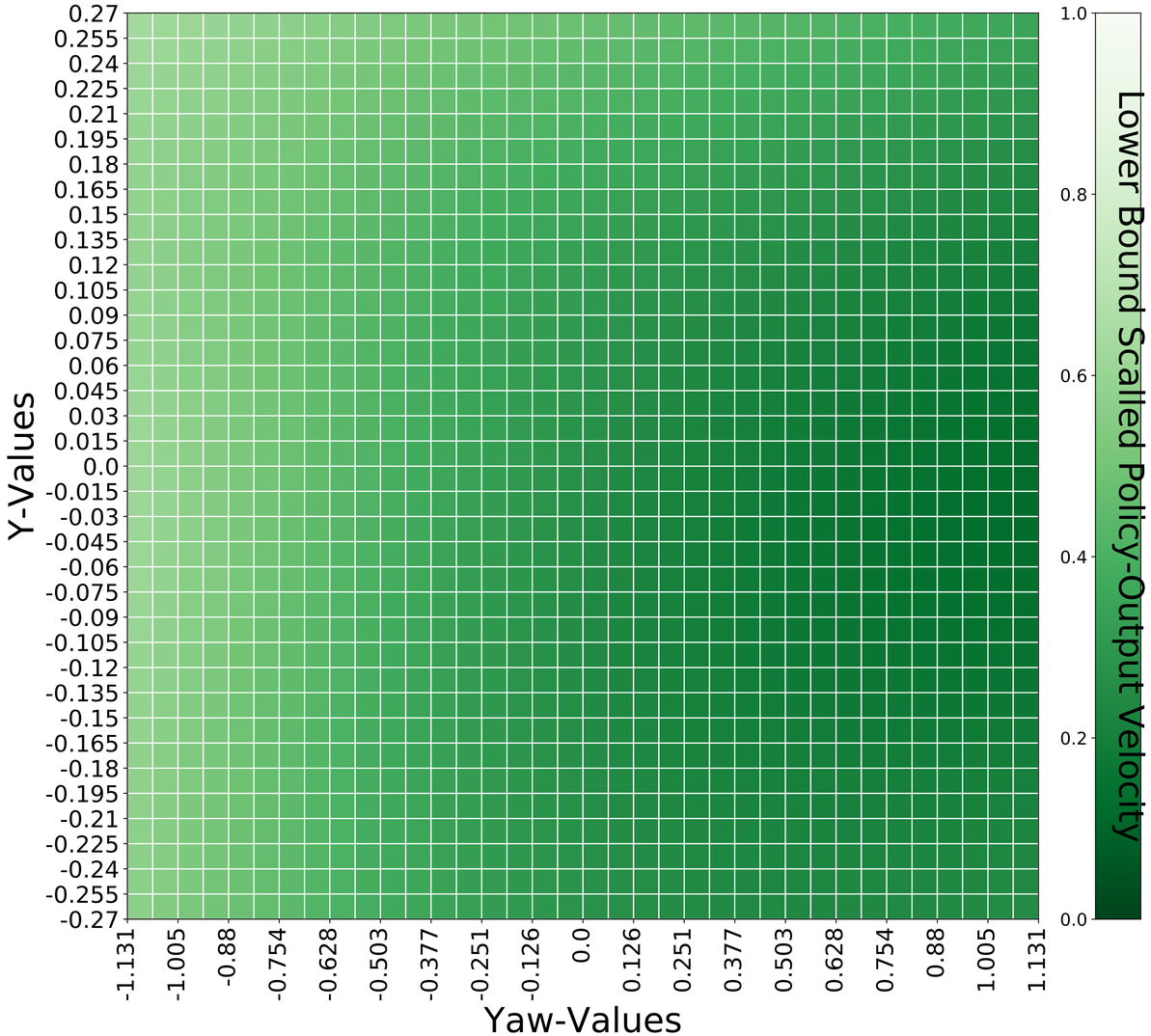


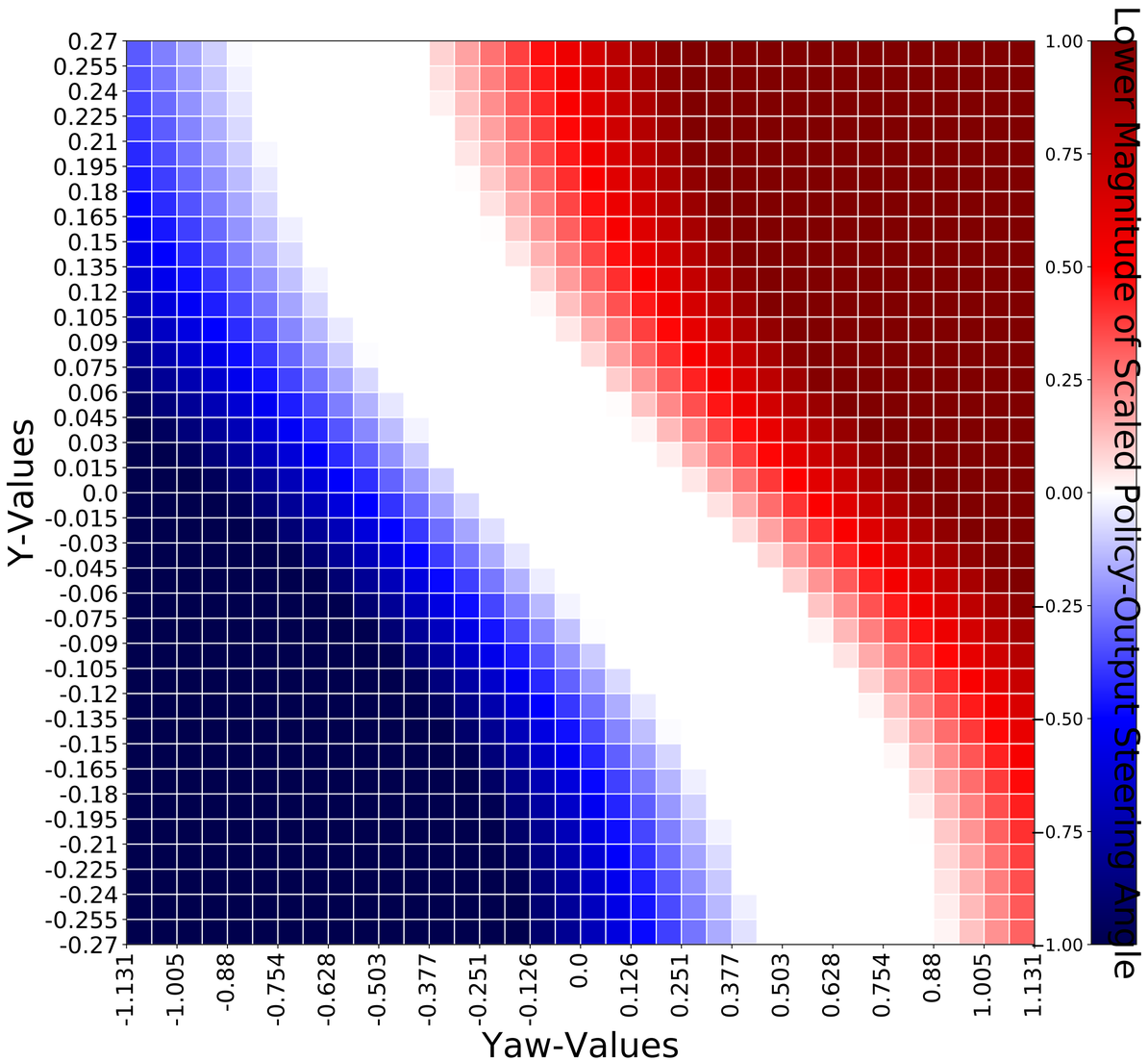


Line-Following Stability Results
xDot:0.300000, yDot:0.415384, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: d44c118e-caf4-4aa6-af10-20915aef985b
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_40_26_096783

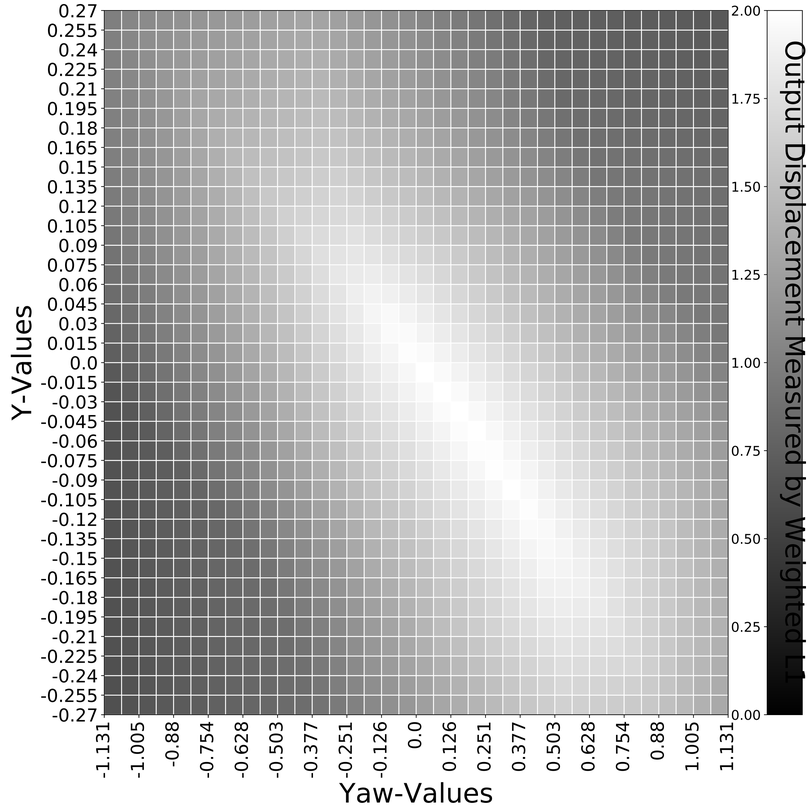


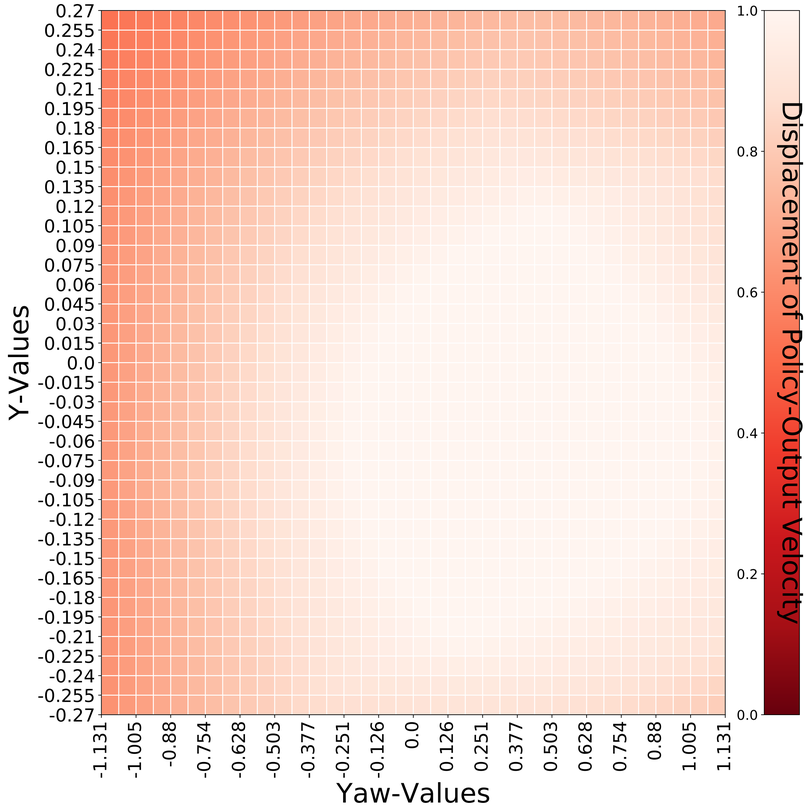
Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: d440a892-be27-448c-8f2b-6538e115d22f
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_12_18_600359

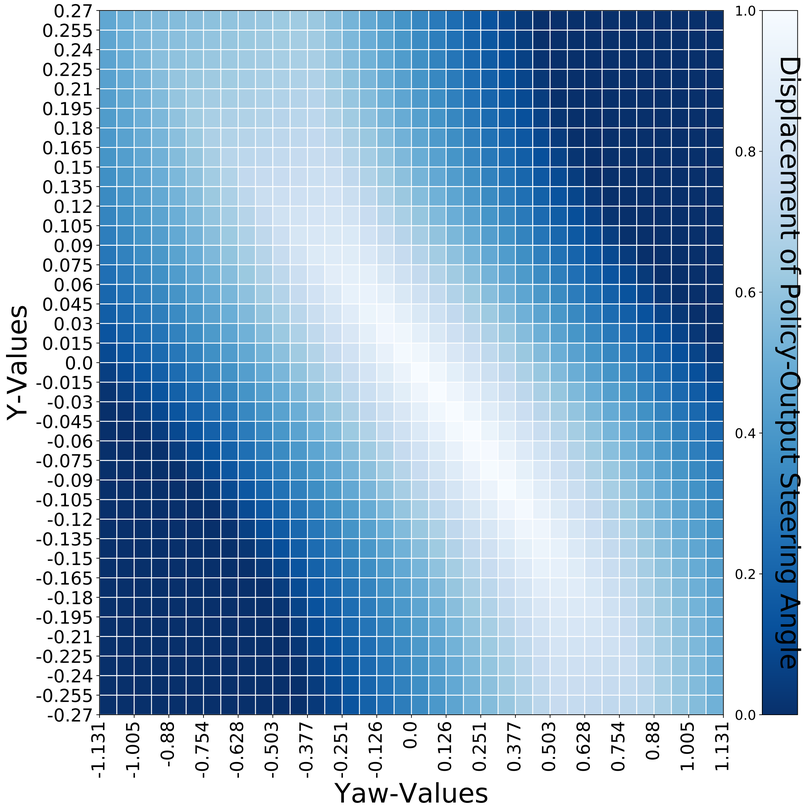




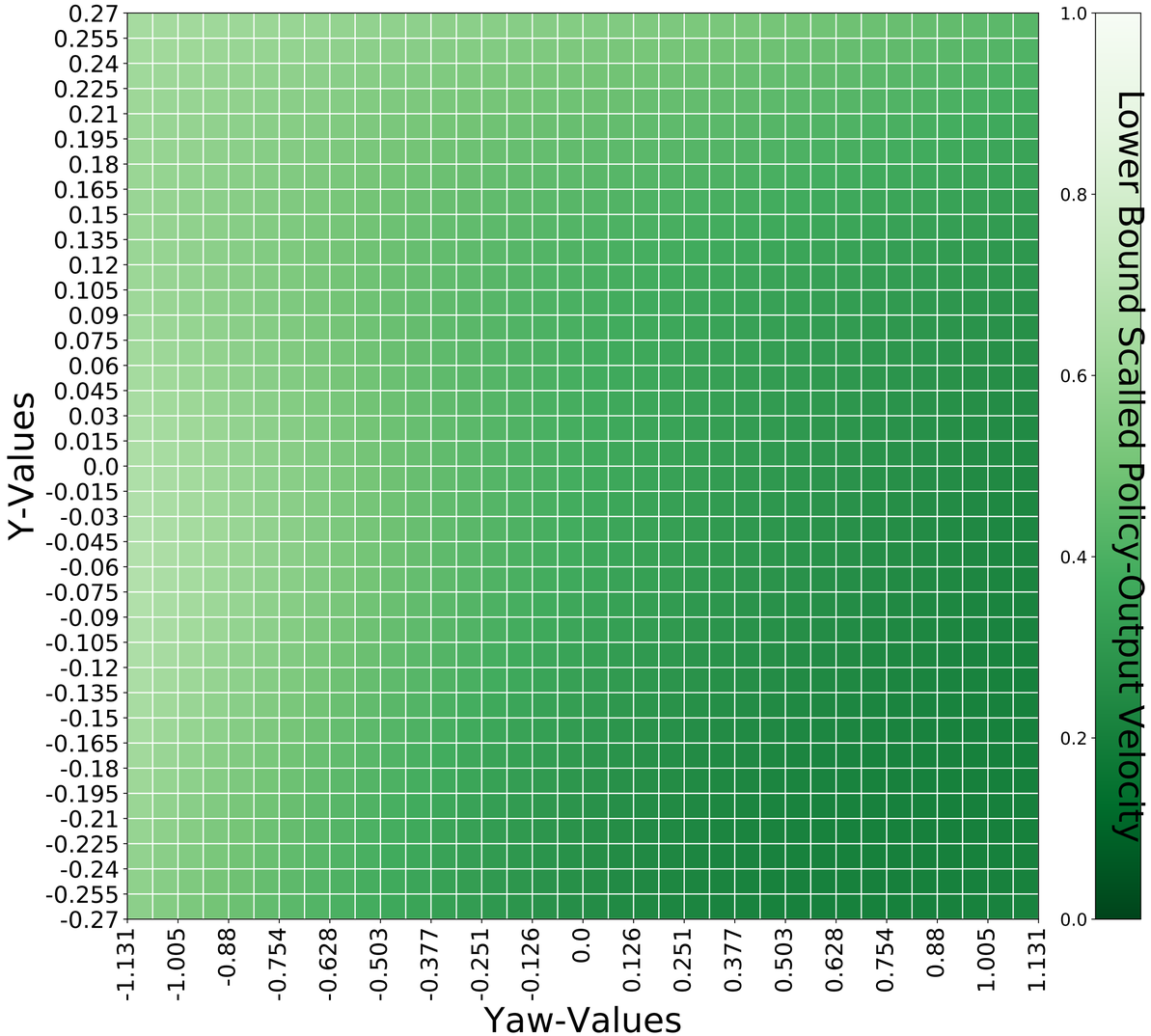
Line-Following Stability Results
xDot:0.300000, yDot:0, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: c1b35a6d-1a12-485a-a4c8-ae26a91c2e86
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_11_14_889998

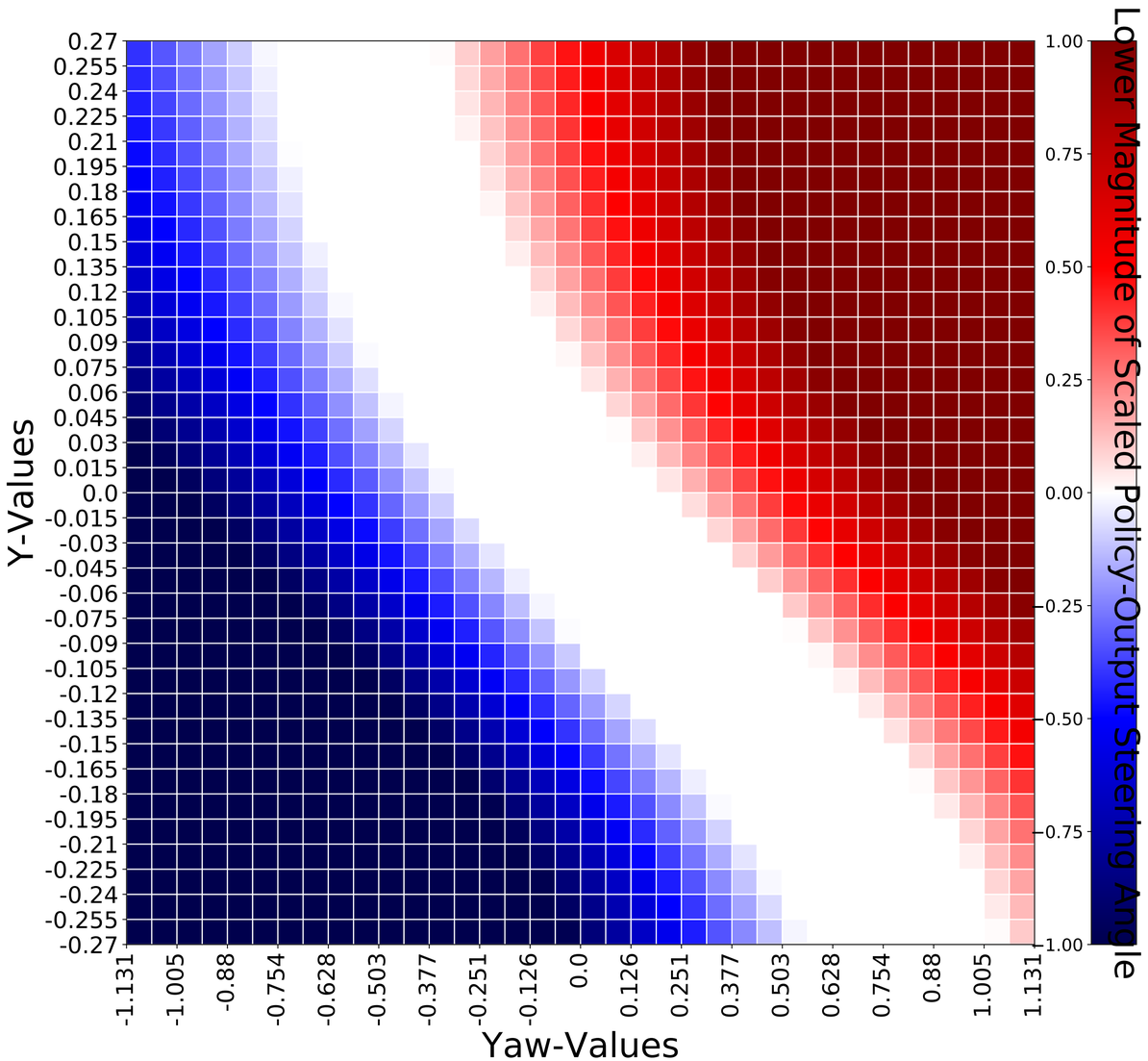




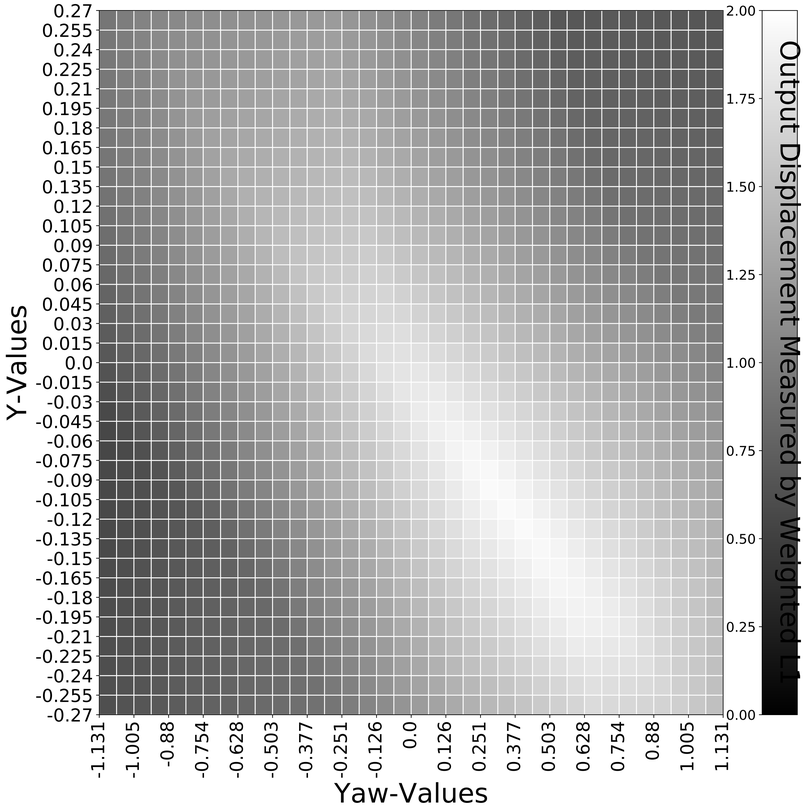


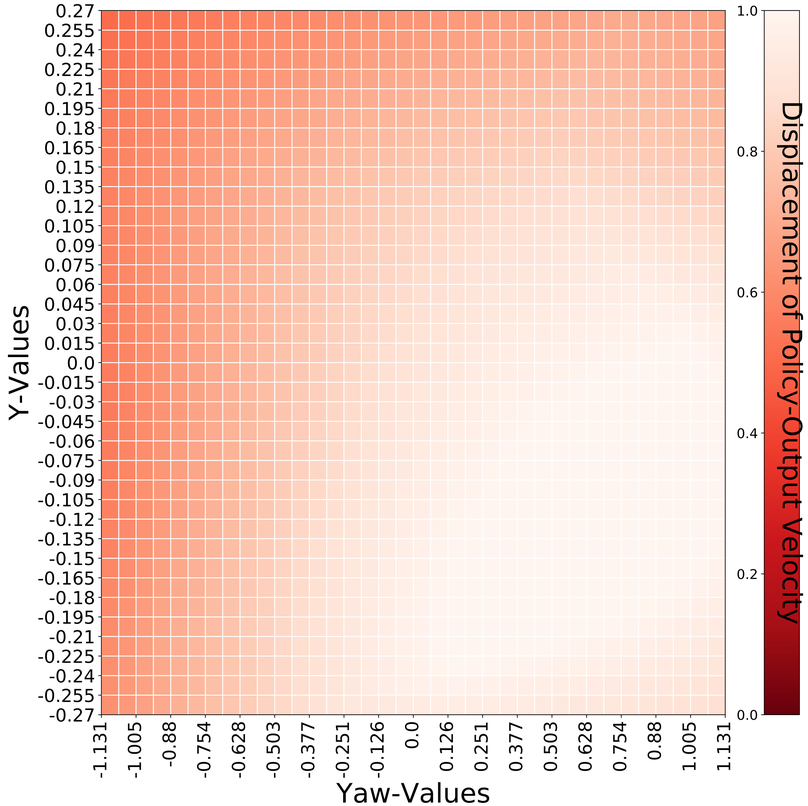
Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: d1e0b992-3e02-4918-860a-4f535367d22a
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_14_06_850193

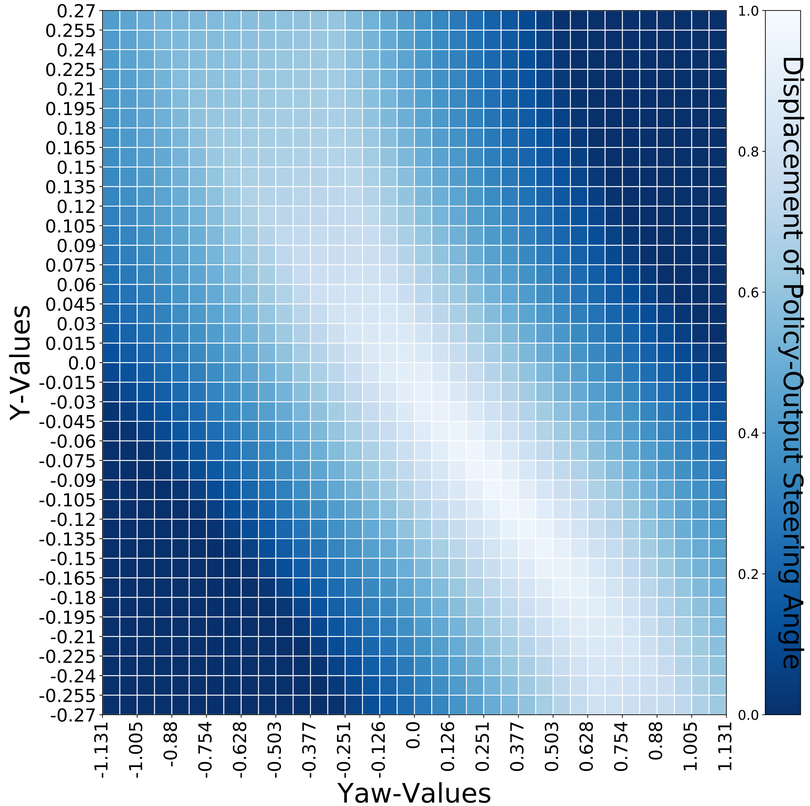




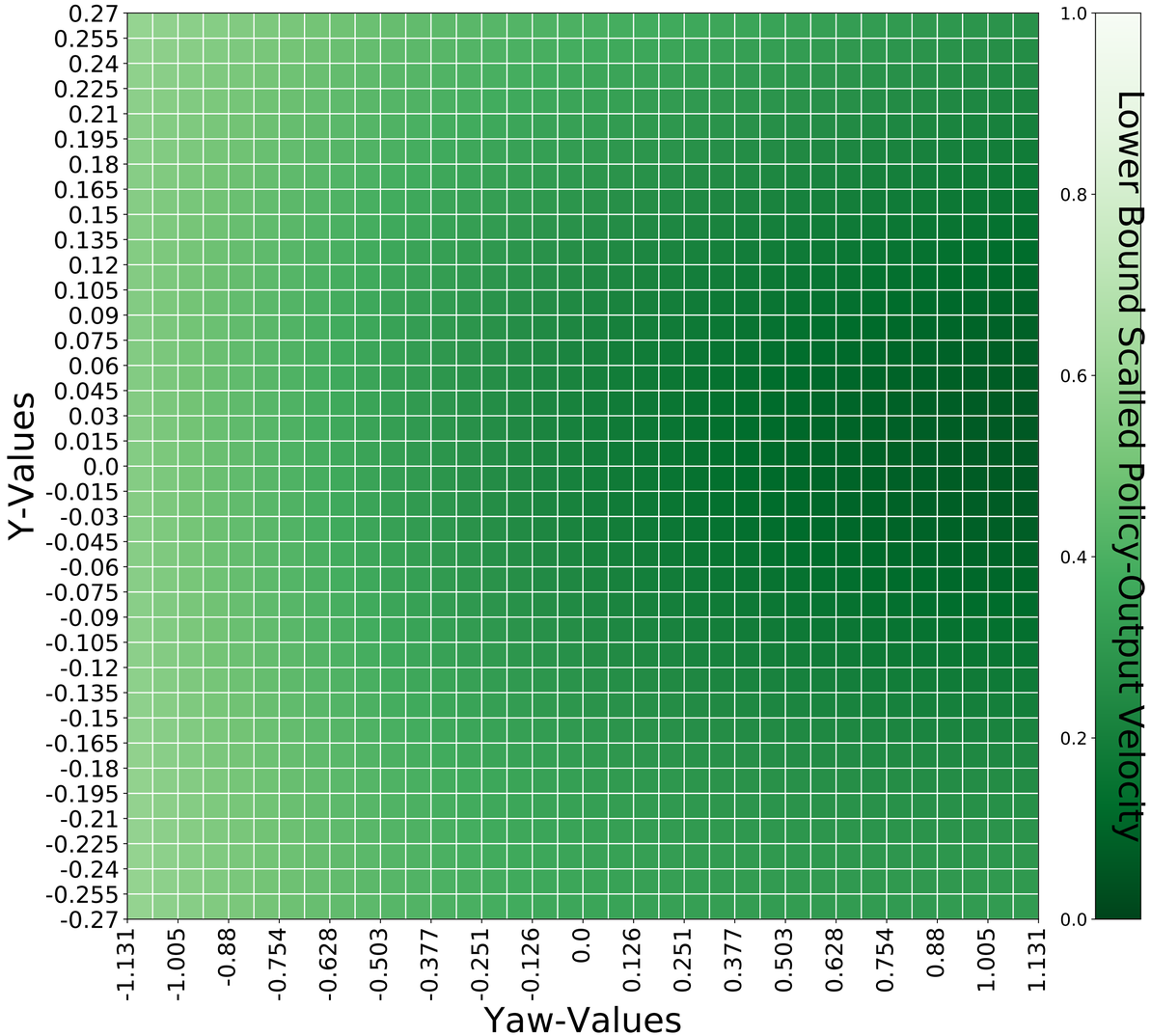
Line-Following Stability Results
xDot:0.300000, yDot:0, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 73ee85a8-a3cf-400c-bb7c-570139e6387a
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_13_11_068820

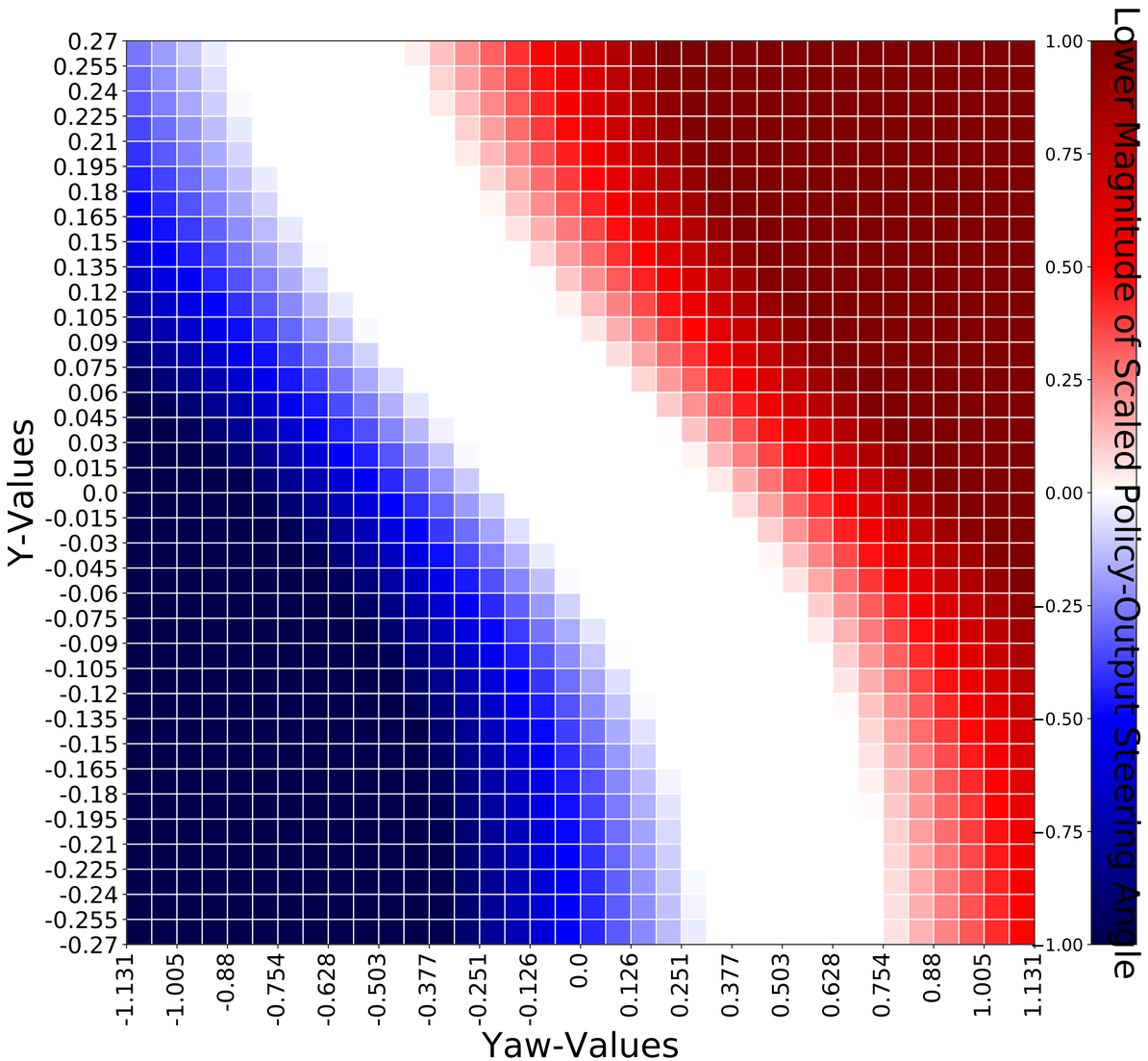




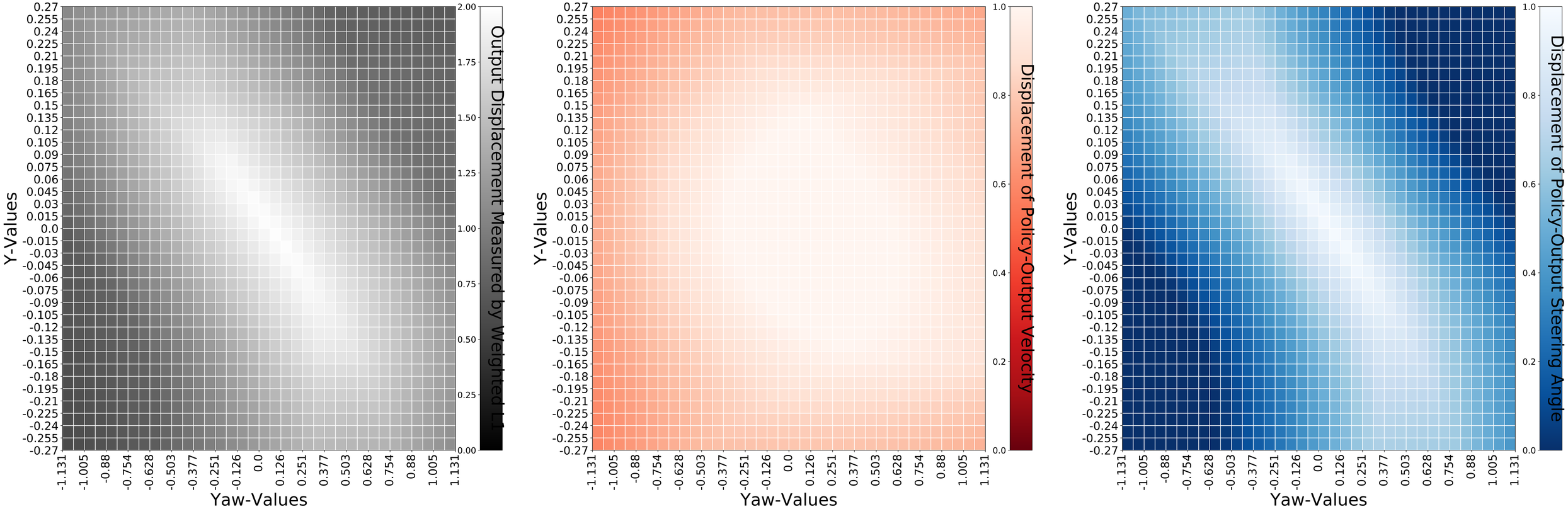


Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 9152f0d6-7fa4-465c-bb1f-e221049d1e2b
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_16_00_453754

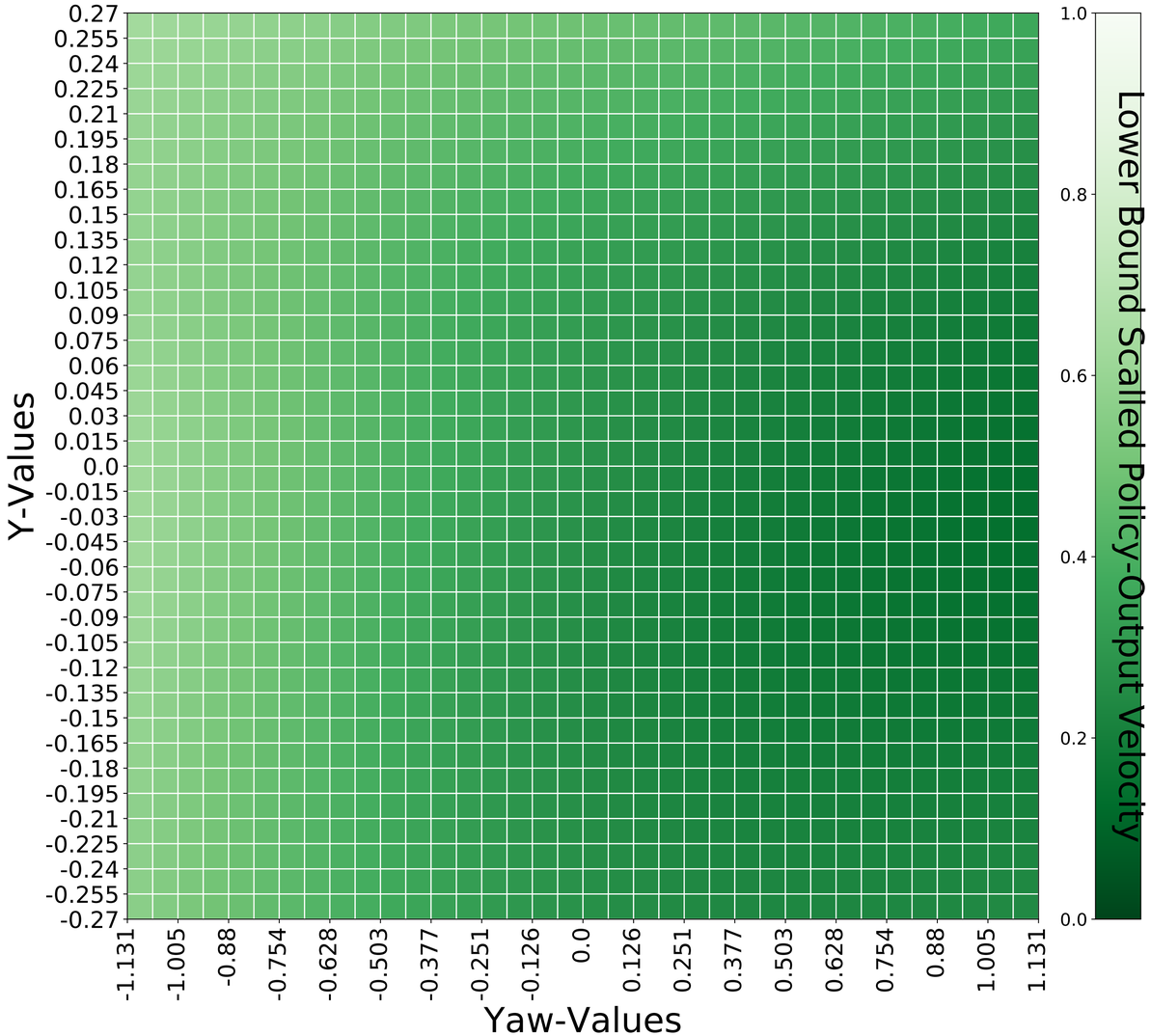


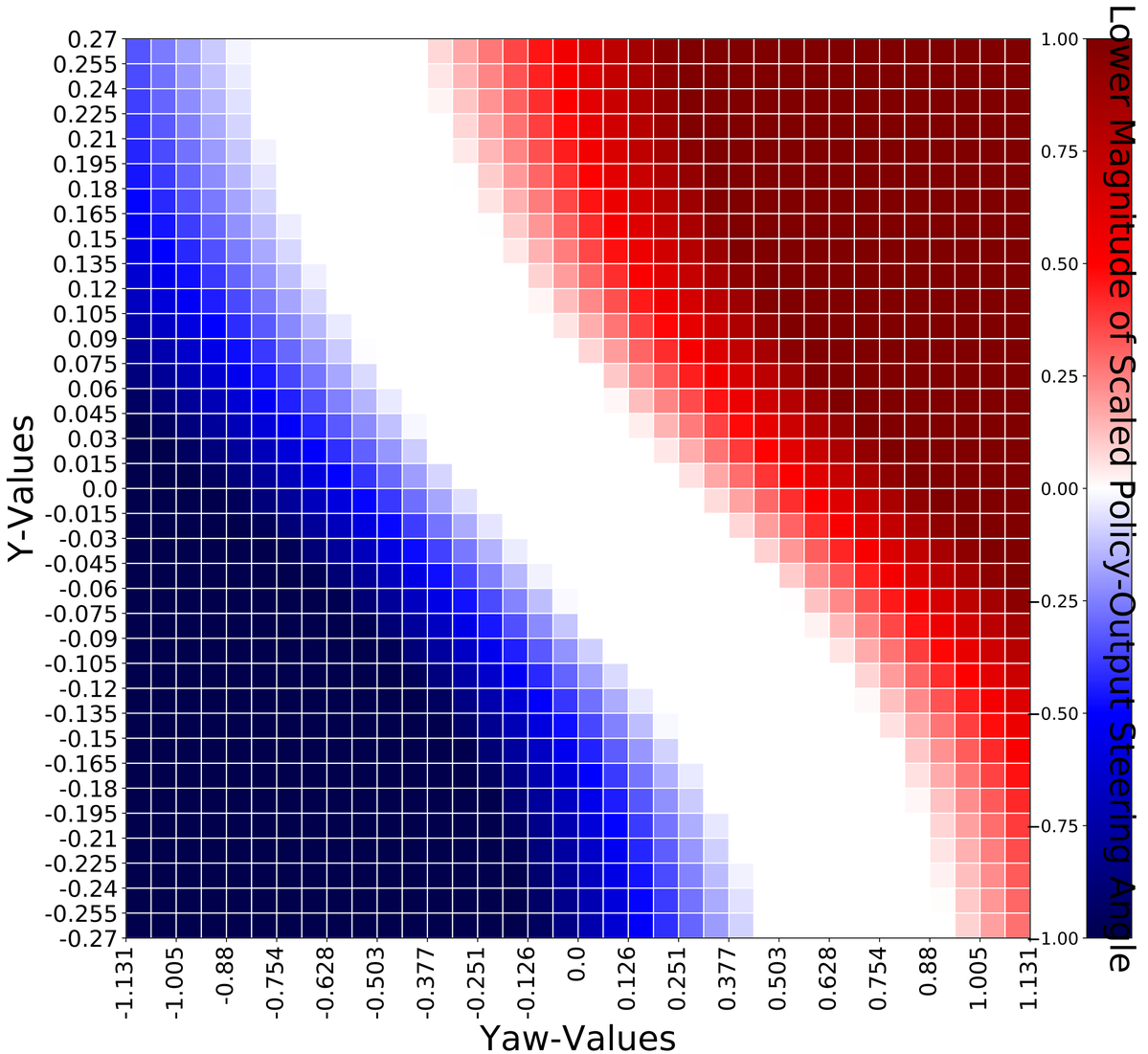


Line-Following Stability Results
xDot:0.300000, yDot:0, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: aec181e6-9749-4152-a431-33e7deaa0487
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_15_05_769030

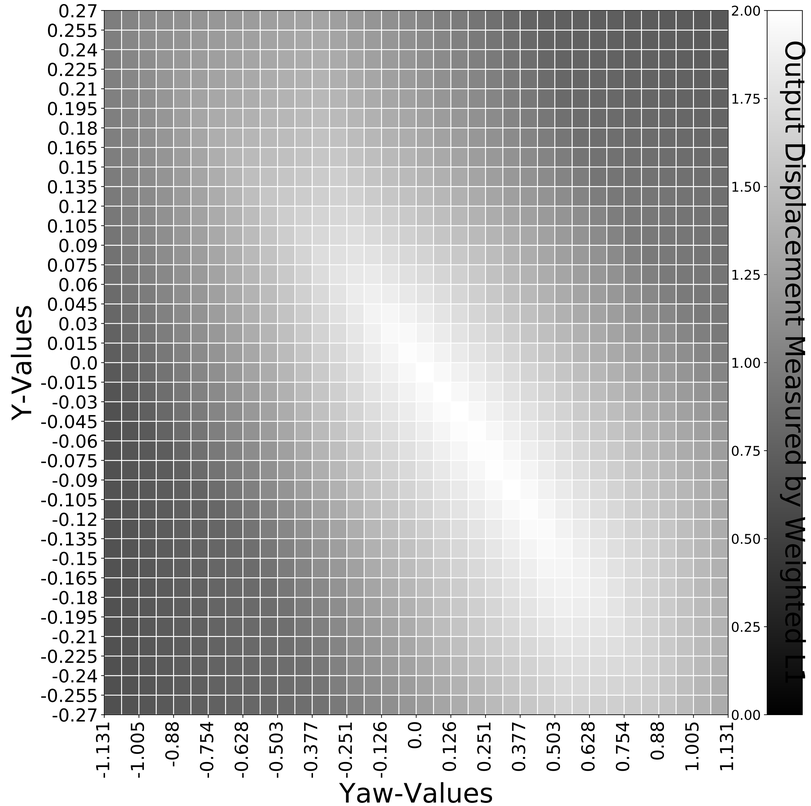


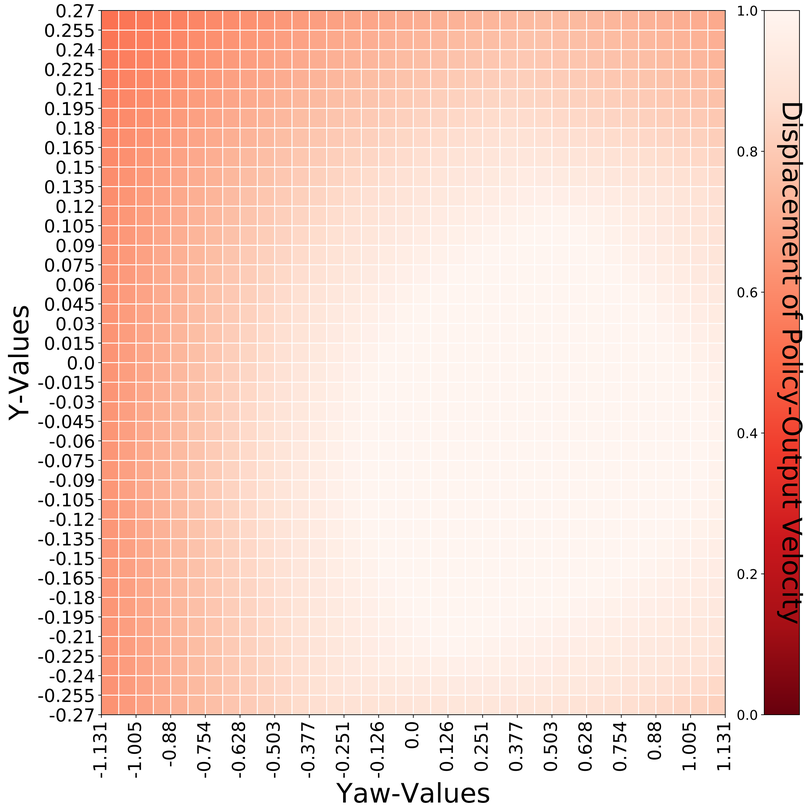
Line-Following Min. Magnitude Output Results
xDot:0.300000, yDot:0, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 8930df90-3045-4aa8-9e75-6dabec1e5684
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_27_35_734229

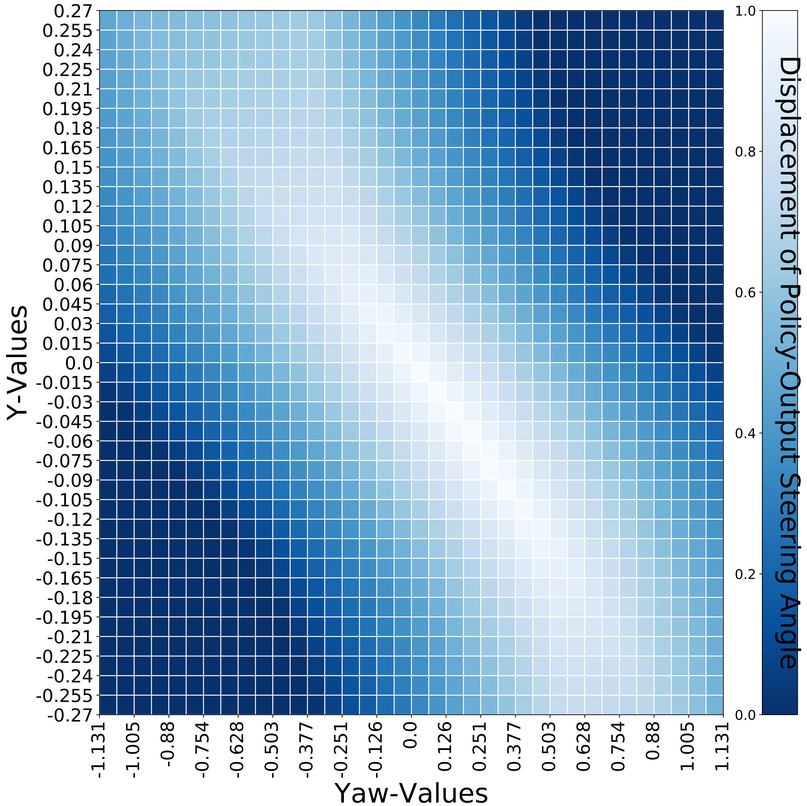




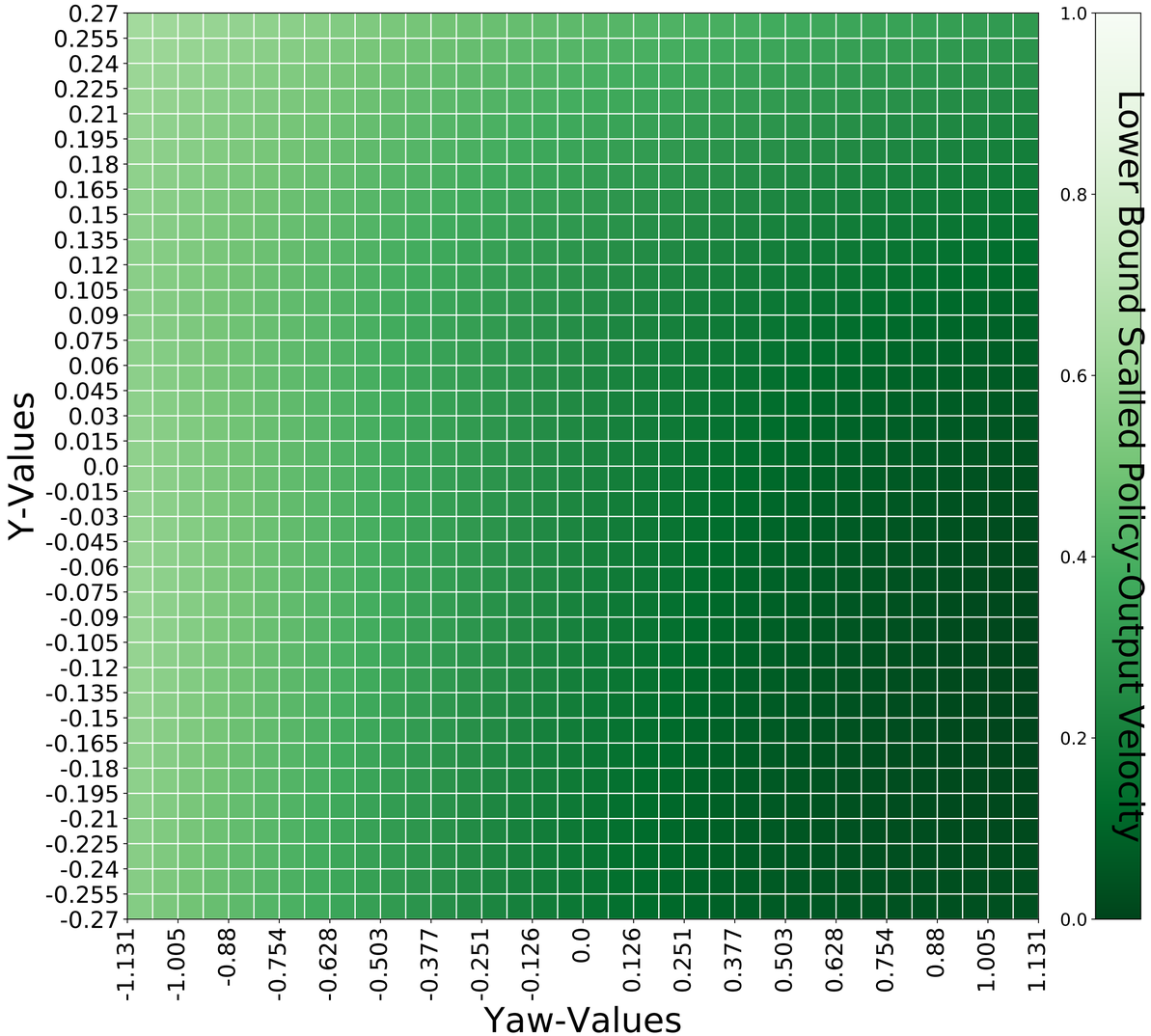
Line-Following Stability Results
xDot:0.300000, yDot:0, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 4702cab0-e9e2-4ea3-b8fb-93f6e3389835
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_26_30_228249

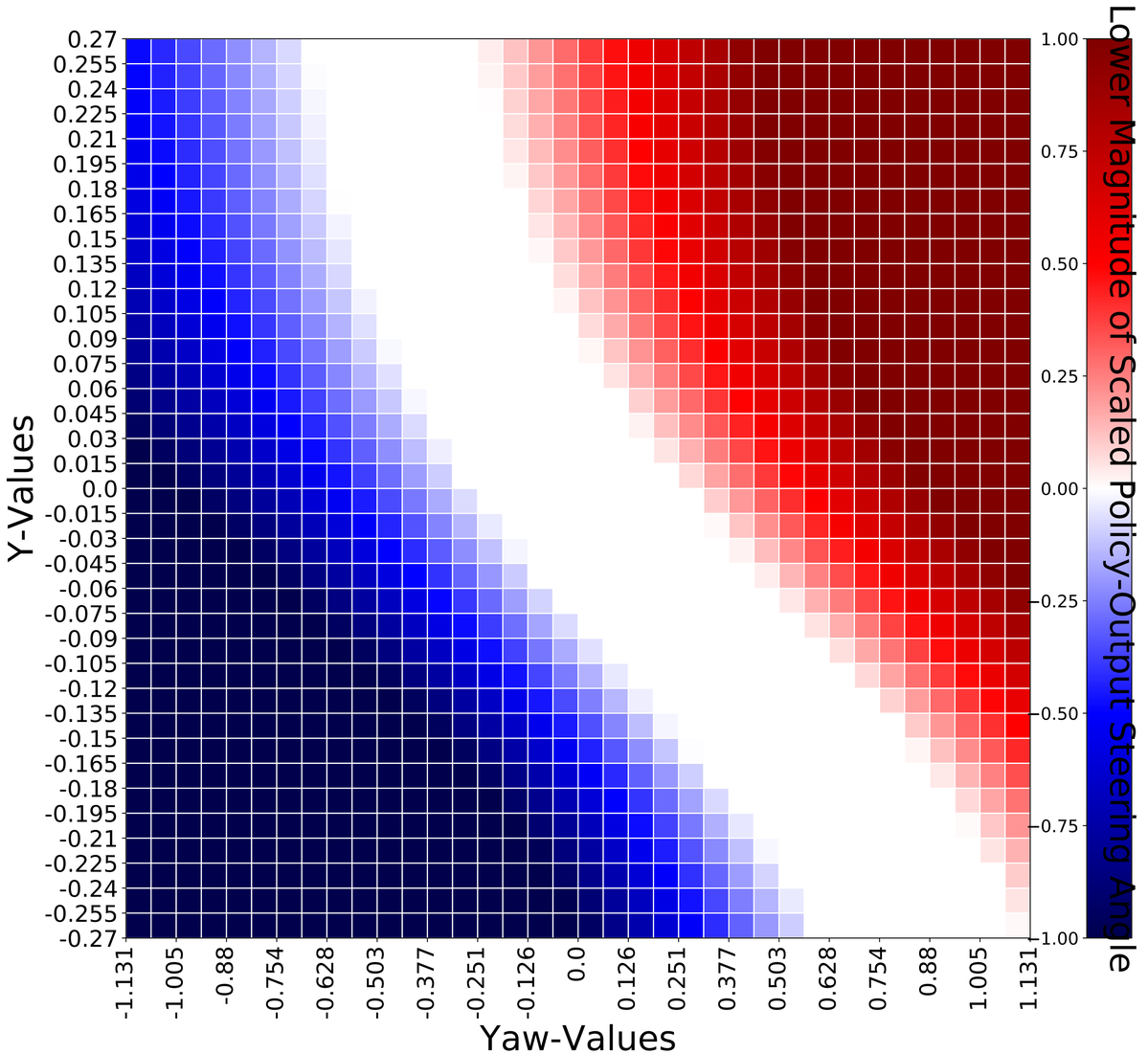




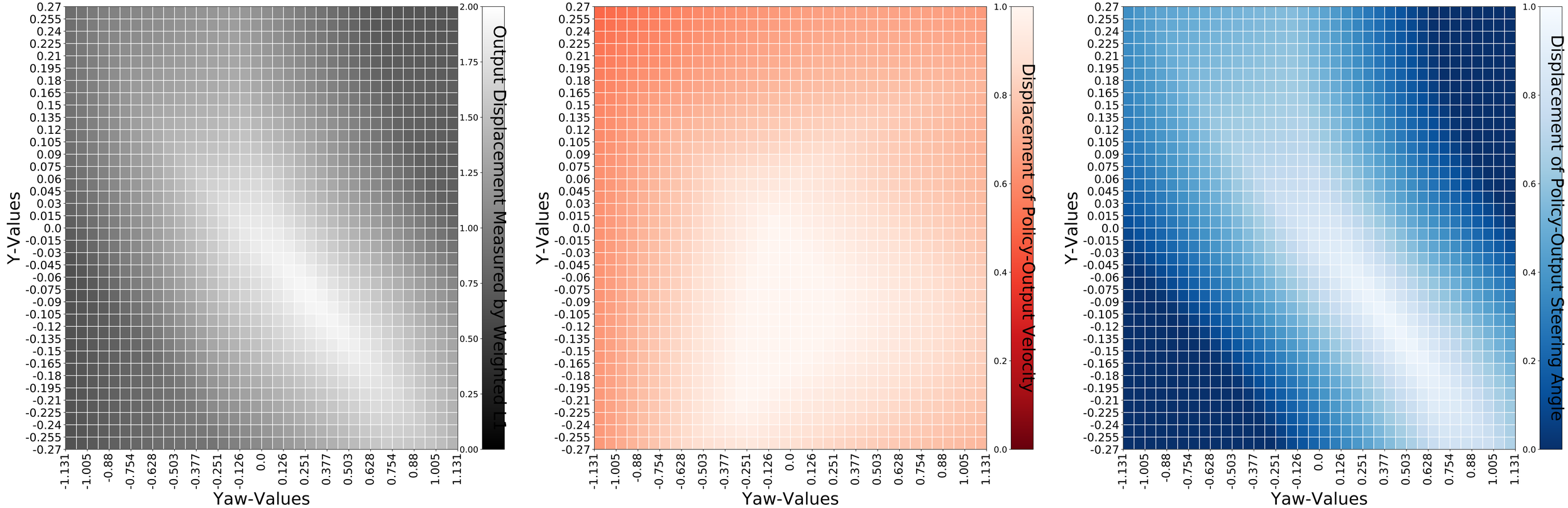


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0.046153, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 3440ca8f-d6a1-4ddc-bc63-96ecdf55f6d7
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_29_38_124451

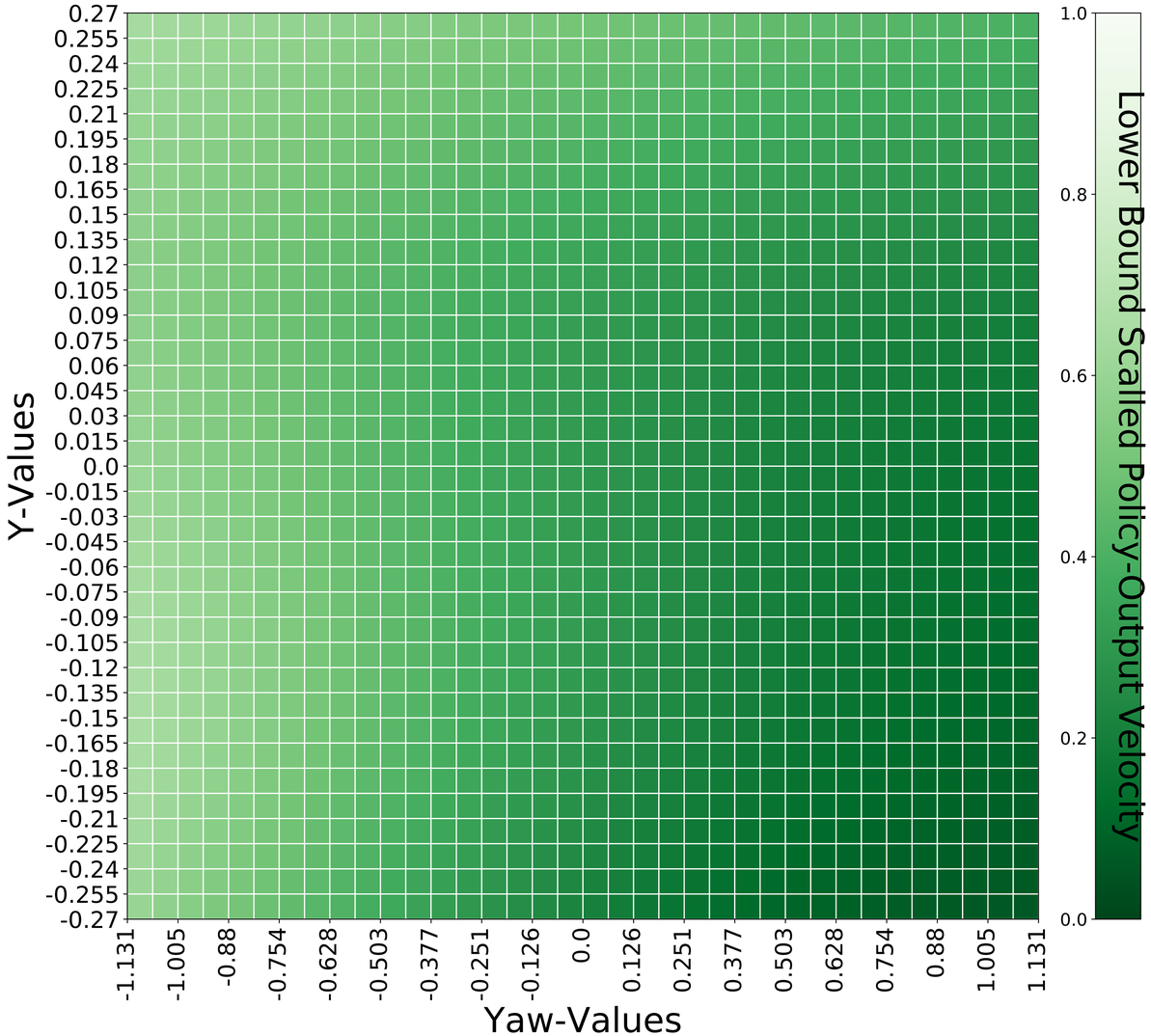


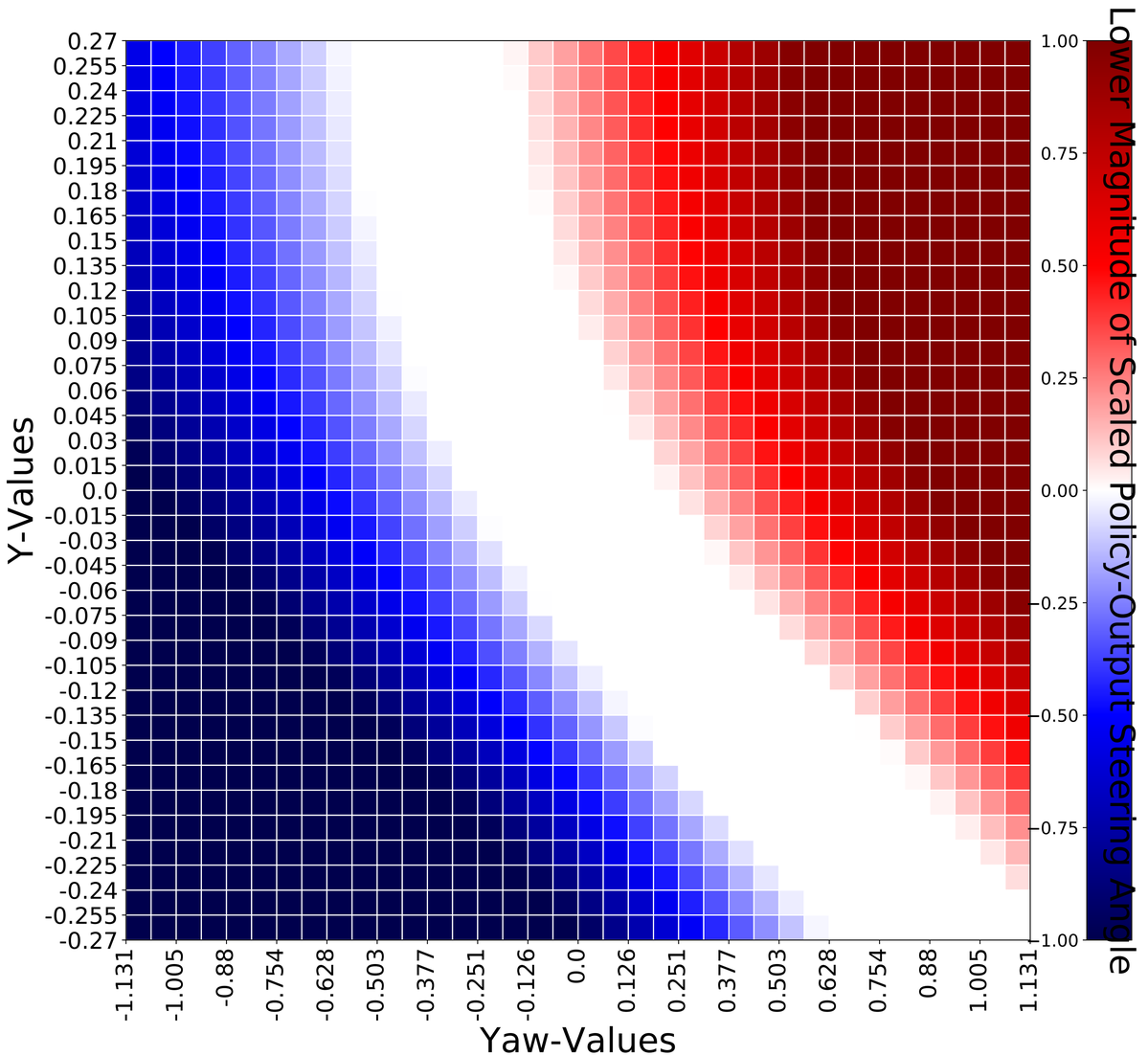


Line-Following Stability Results
xDot:1.1, yDot:0.046153, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: be7afb71-1487-4b0b-a07c-7ac7056fa602
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_28_39_118814

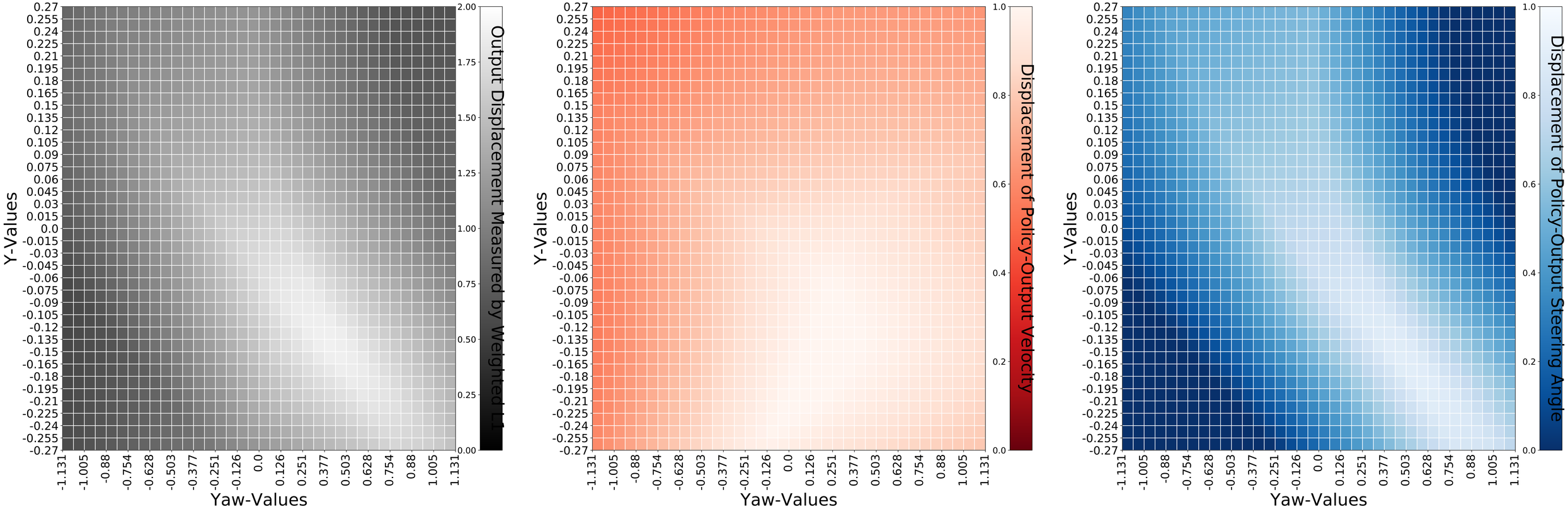


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0.046153, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 57d68697-c71a-4c04-91b2-fae38d6ad1f3
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_31_30_853298

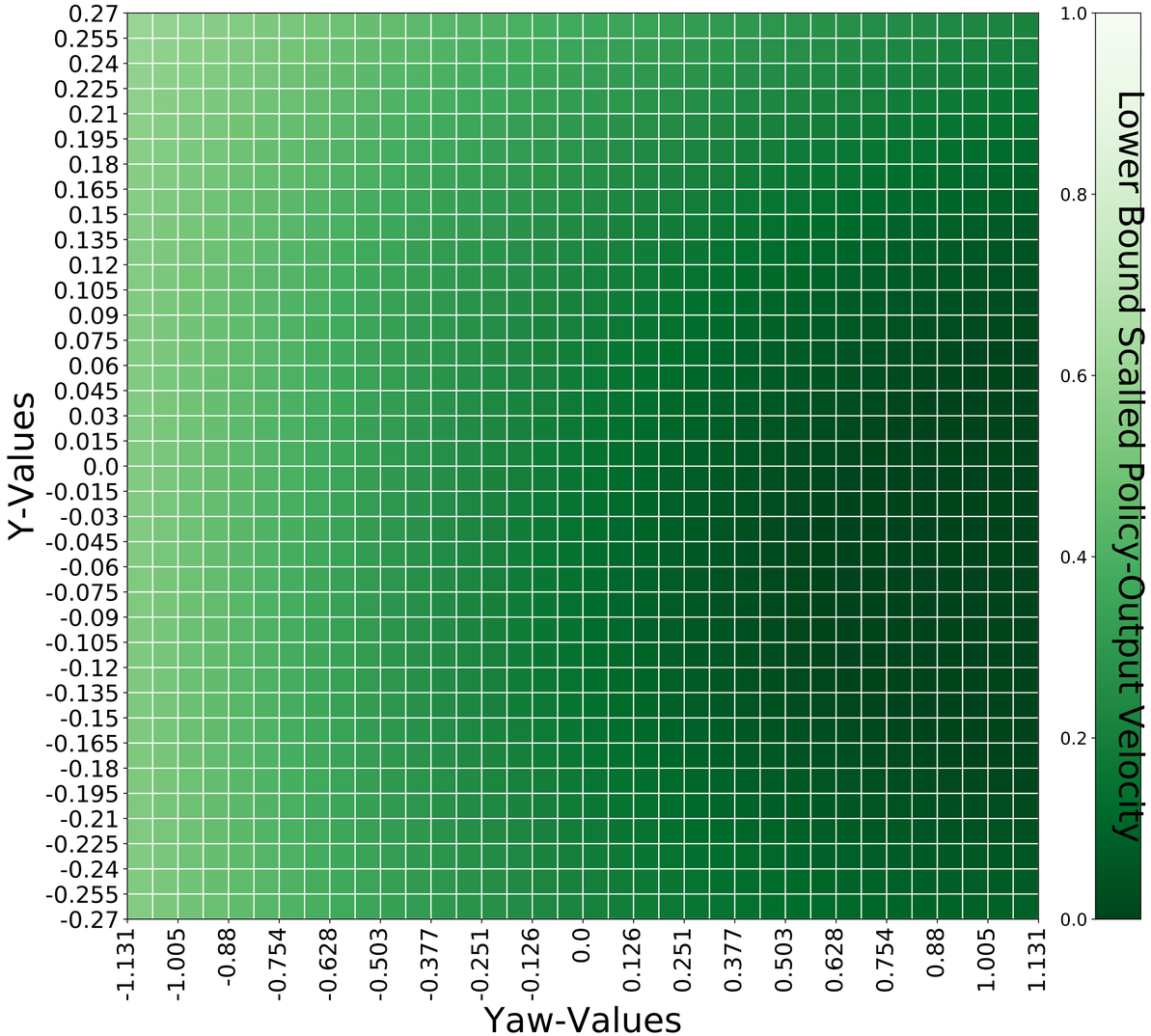


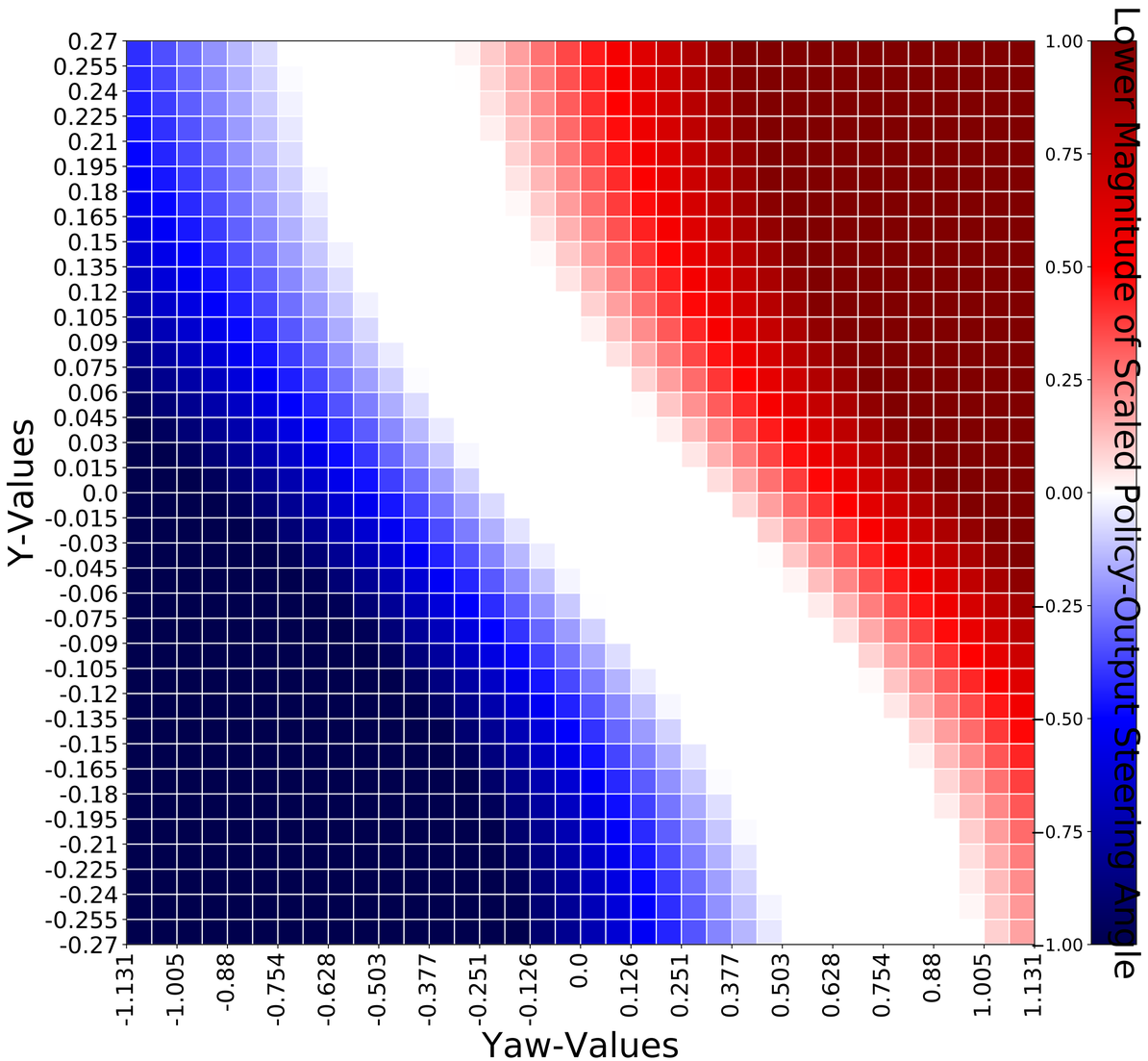


Line-Following Stability Results
xDot:1.1, yDot:0.046153, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 50722ab0-f1fc-4cd9-830f-5ee618be2bab
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_30_31_185852

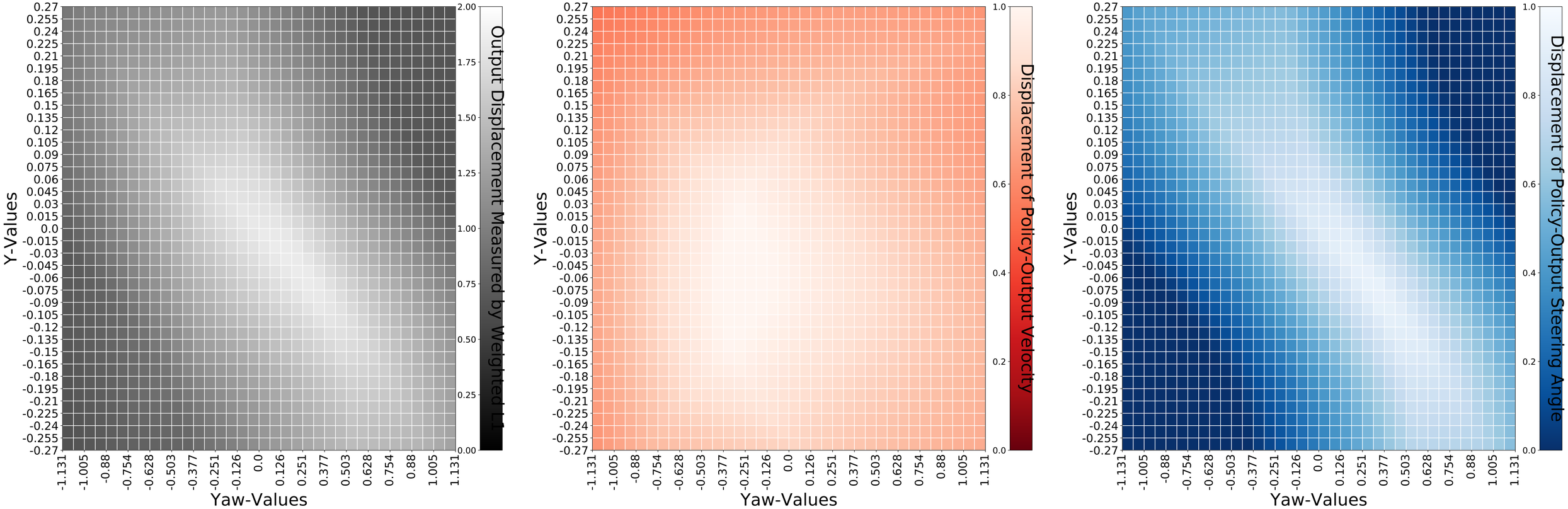


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0.046153, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 8d6b636d-2ebc-411d-97c7-3a50b3855fe1
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_33_33_494784

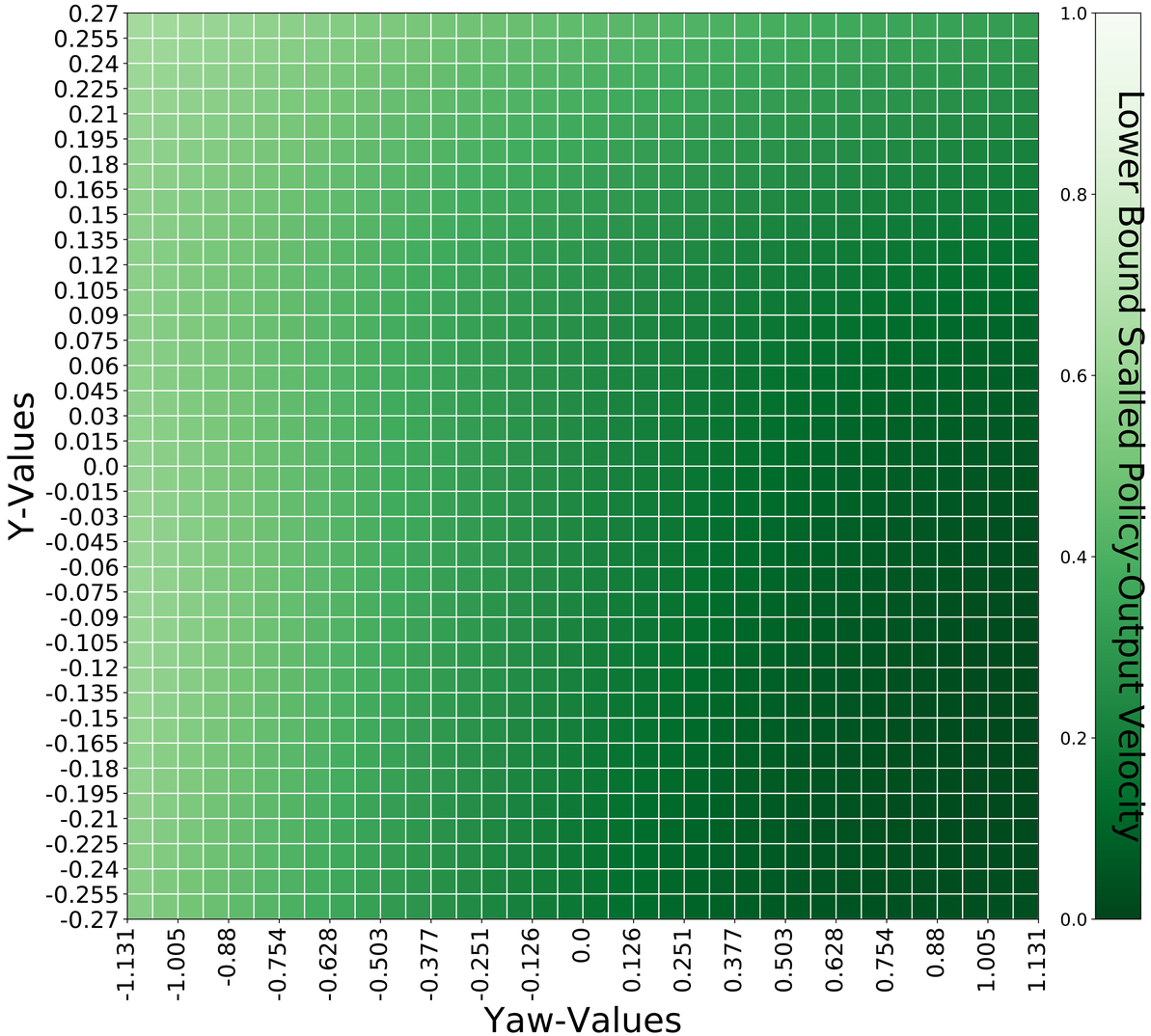


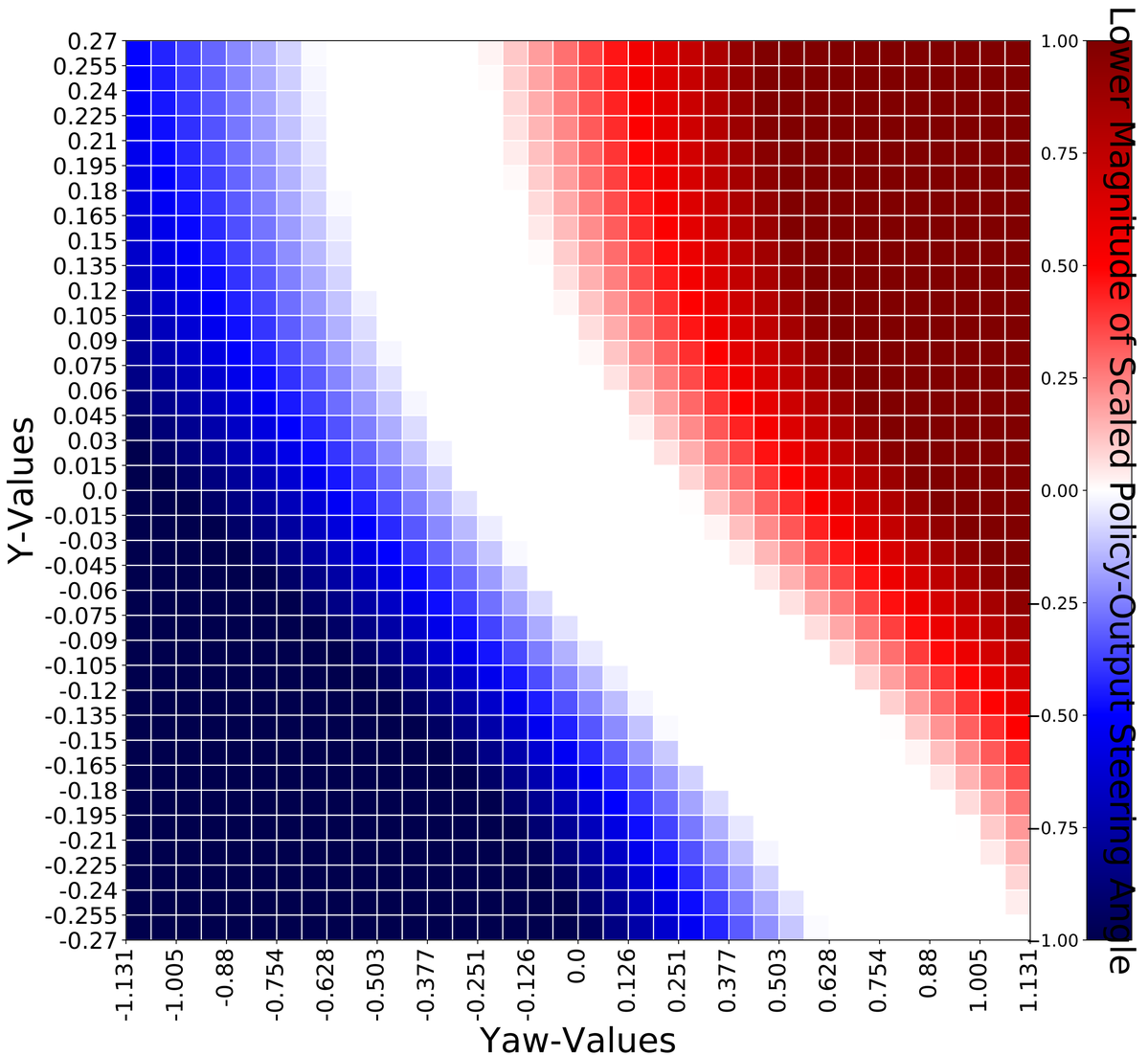


Line-Following Stability Results
xDot:1.1, yDot:0.046153, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 598474e0-3bd8-40a0-9e94-409094ac4a75
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_32_31_642645

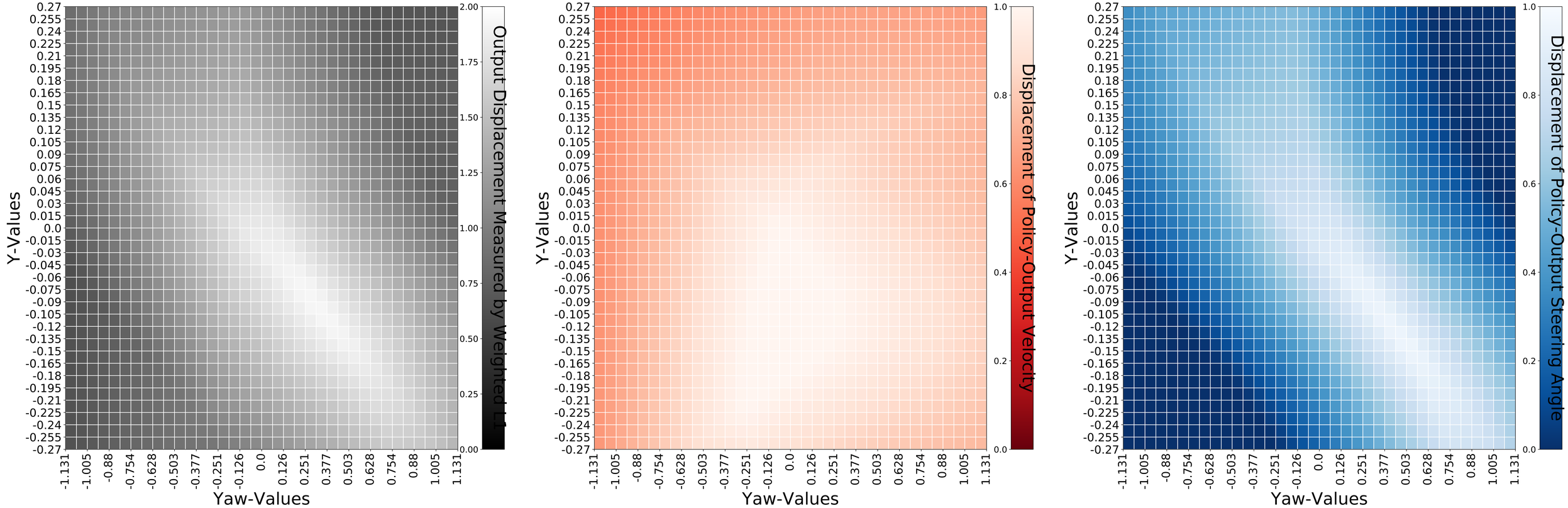


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0.046153, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 77f19871-bbe5-4212-a390-f0f0edb35023
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_45_22_127941

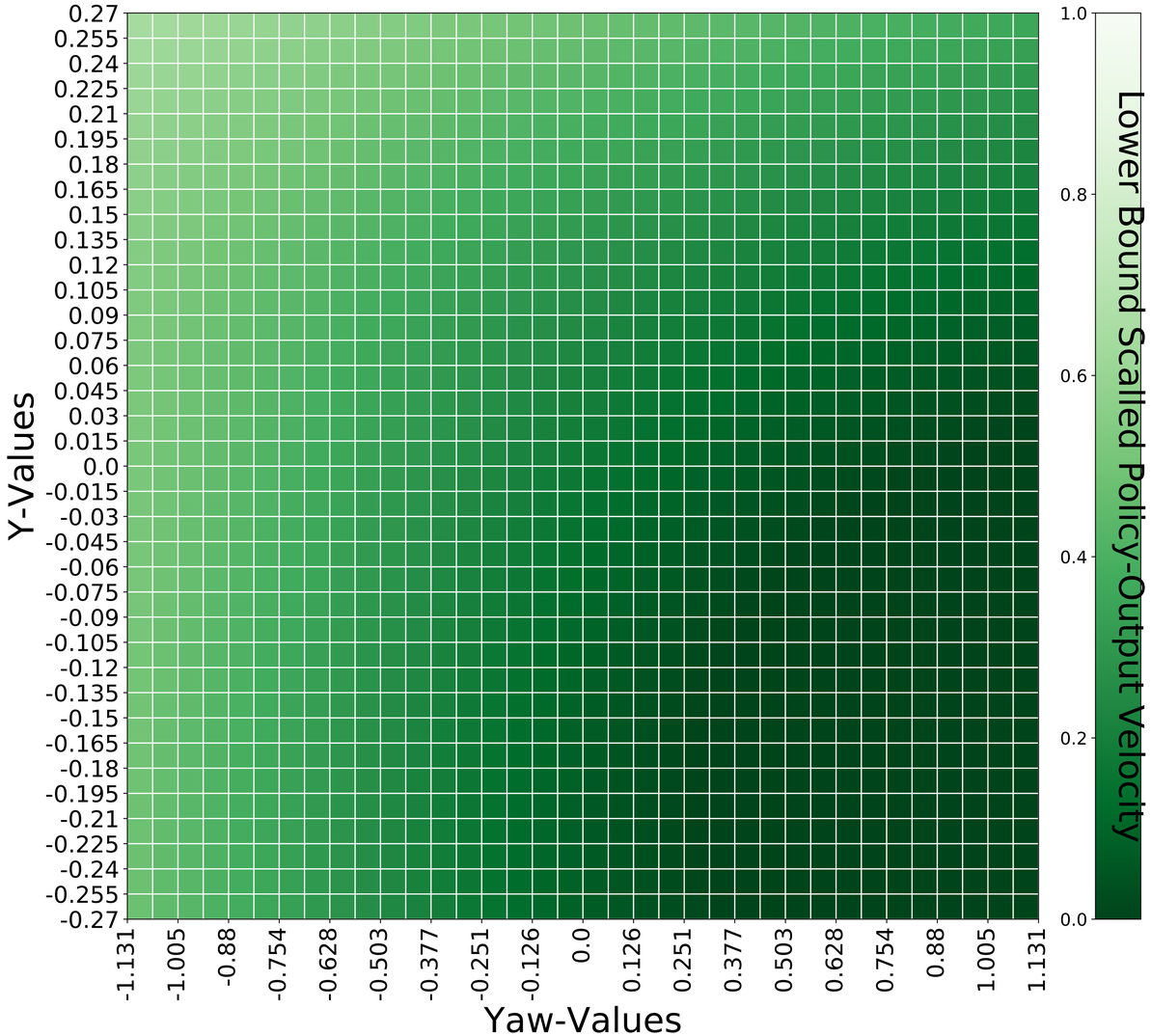


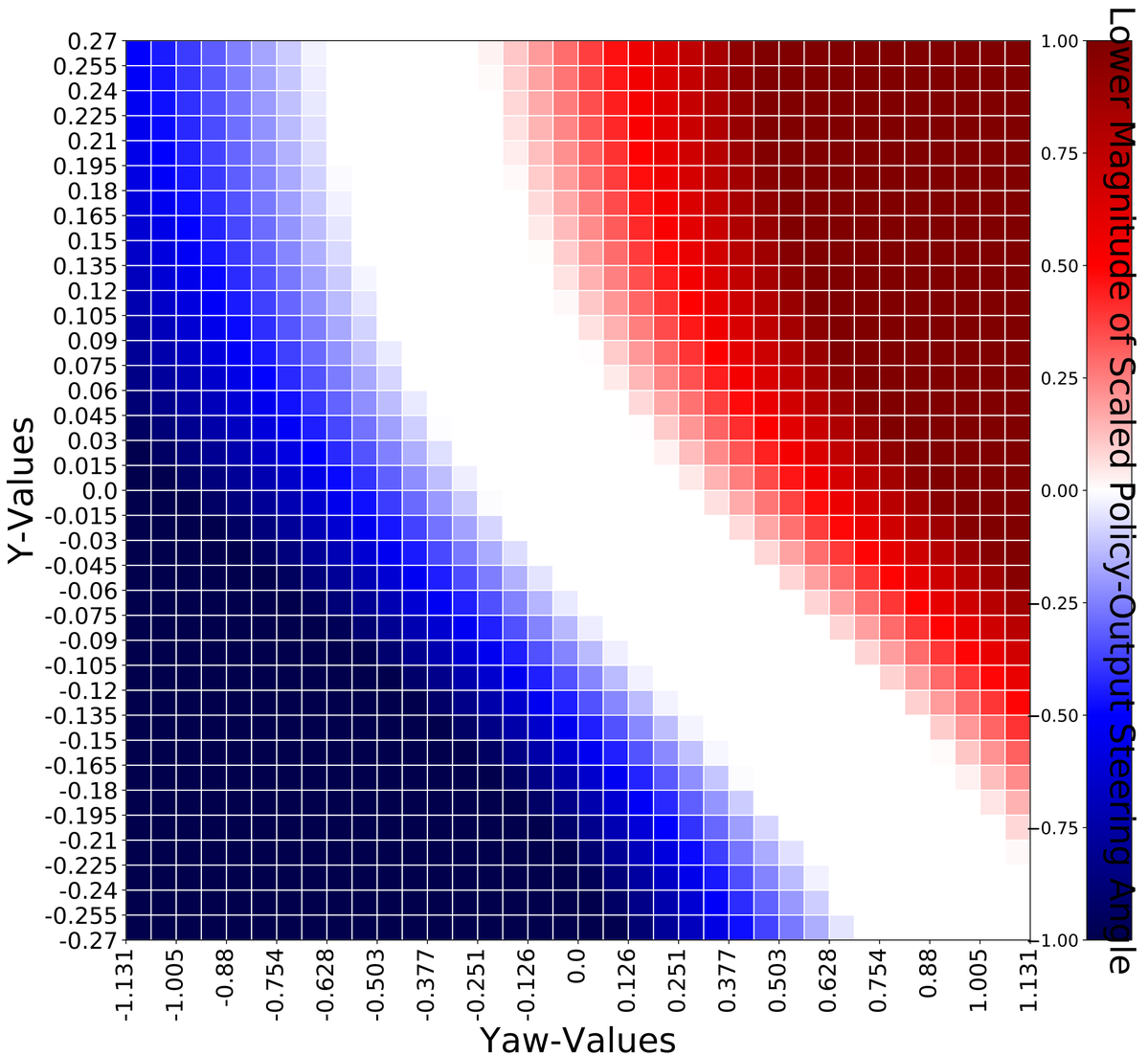


Line-Following Stability Results
xDot:1.1, yDot:0.046153, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: afba8712-8b7e-4592-b16b-eee7d70cdf2c
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_44_18_791031

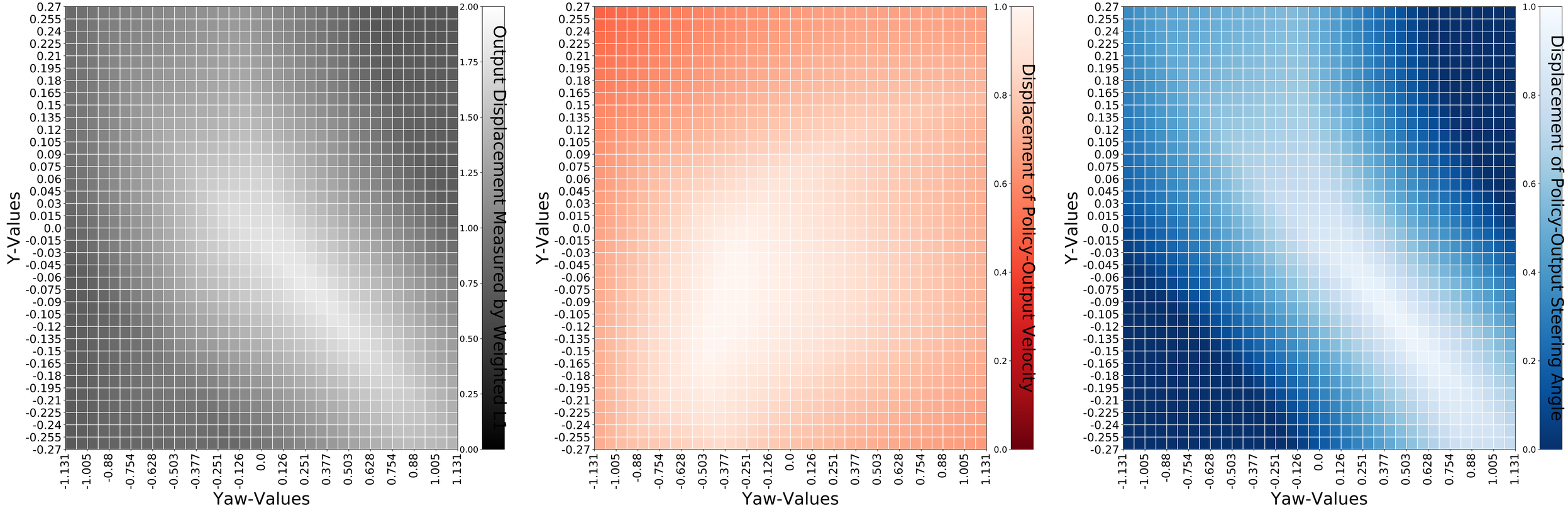


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:-0.32307, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 78a4e4b5-f68b-4217-82fc-eea7793b19f3
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_47_13_648045

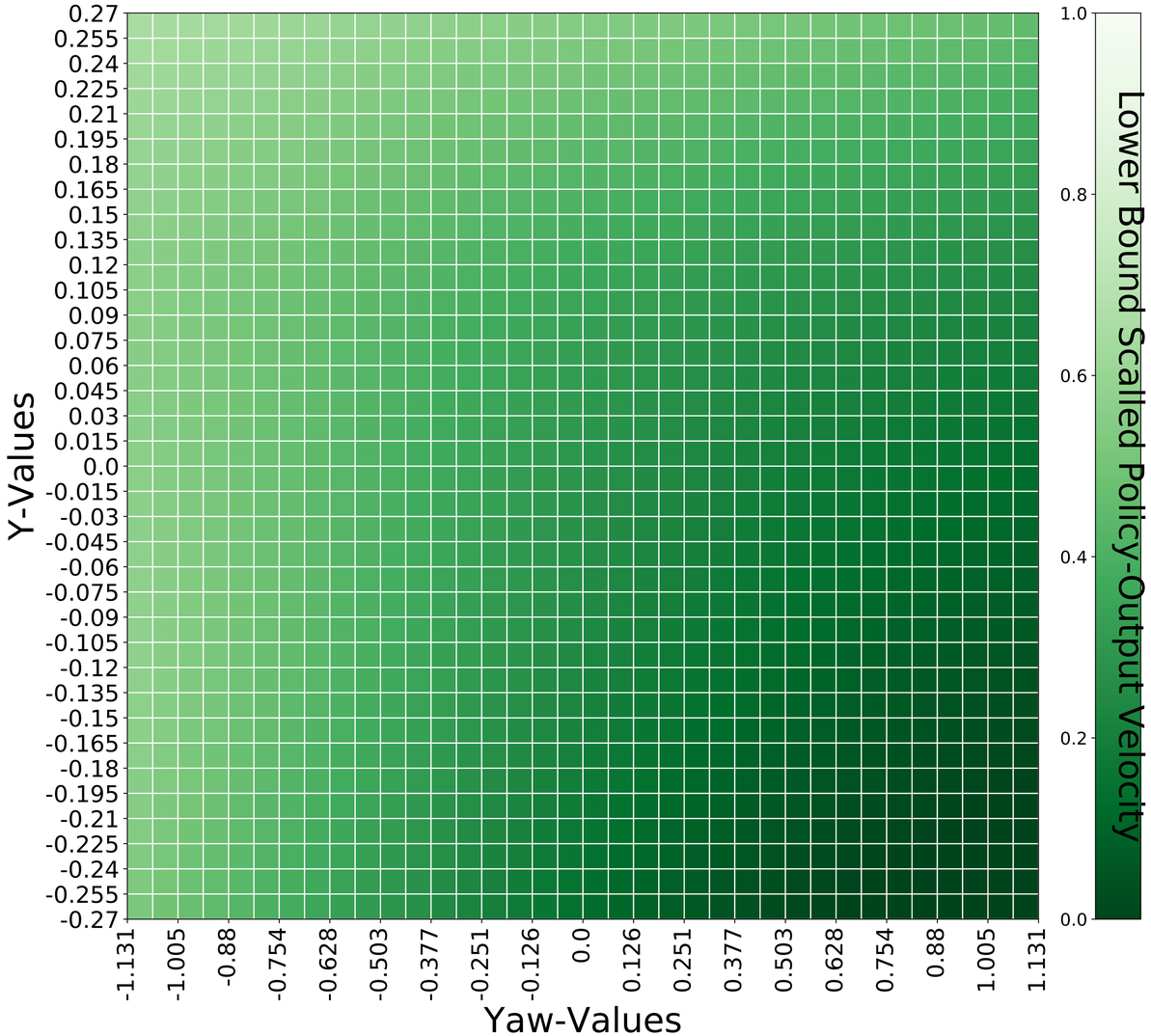


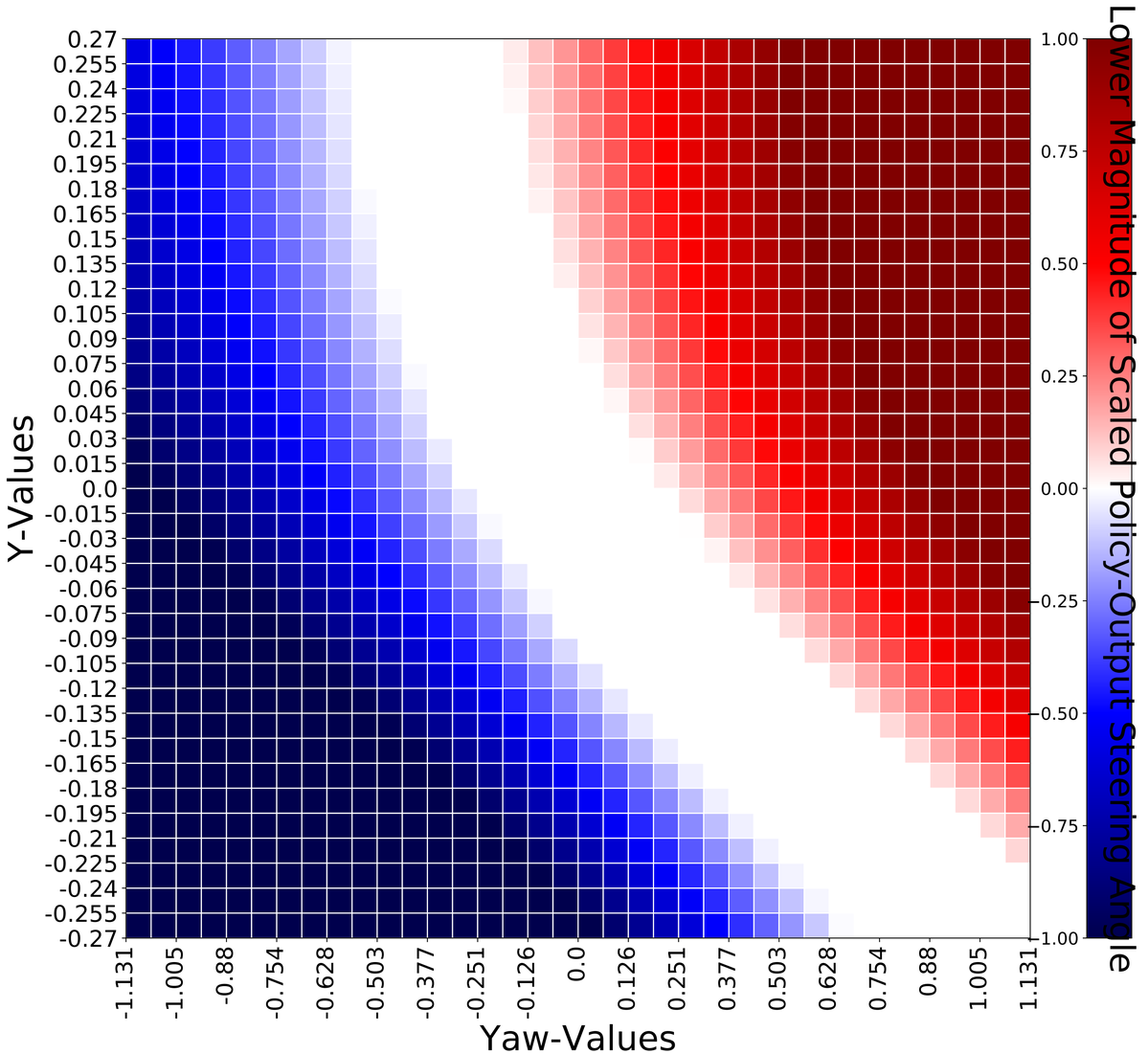


Line-Following Stability Results
xDot:1.1, yDot:-0.32307, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: cc5e2605-dc25-4f91-a7f0-8449e888255a
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_46_20_698355

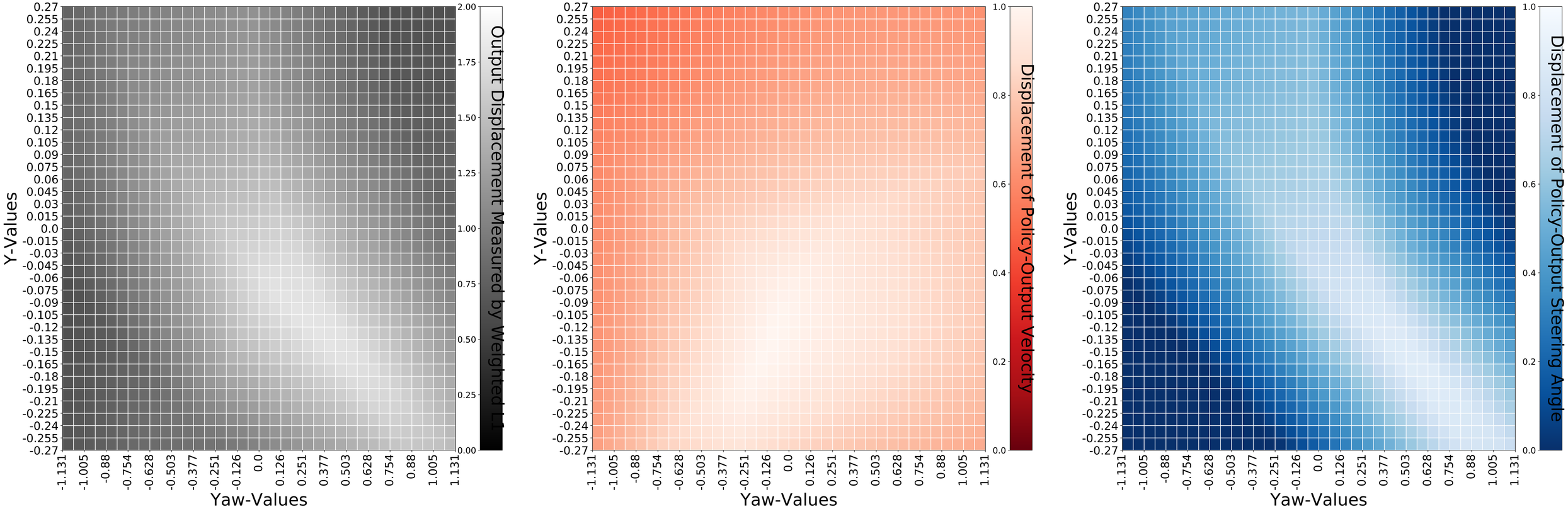


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:-0.32307, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: bc6ec879-18e8-4689-a8be-e2735f9cfd43
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_49_04_585465

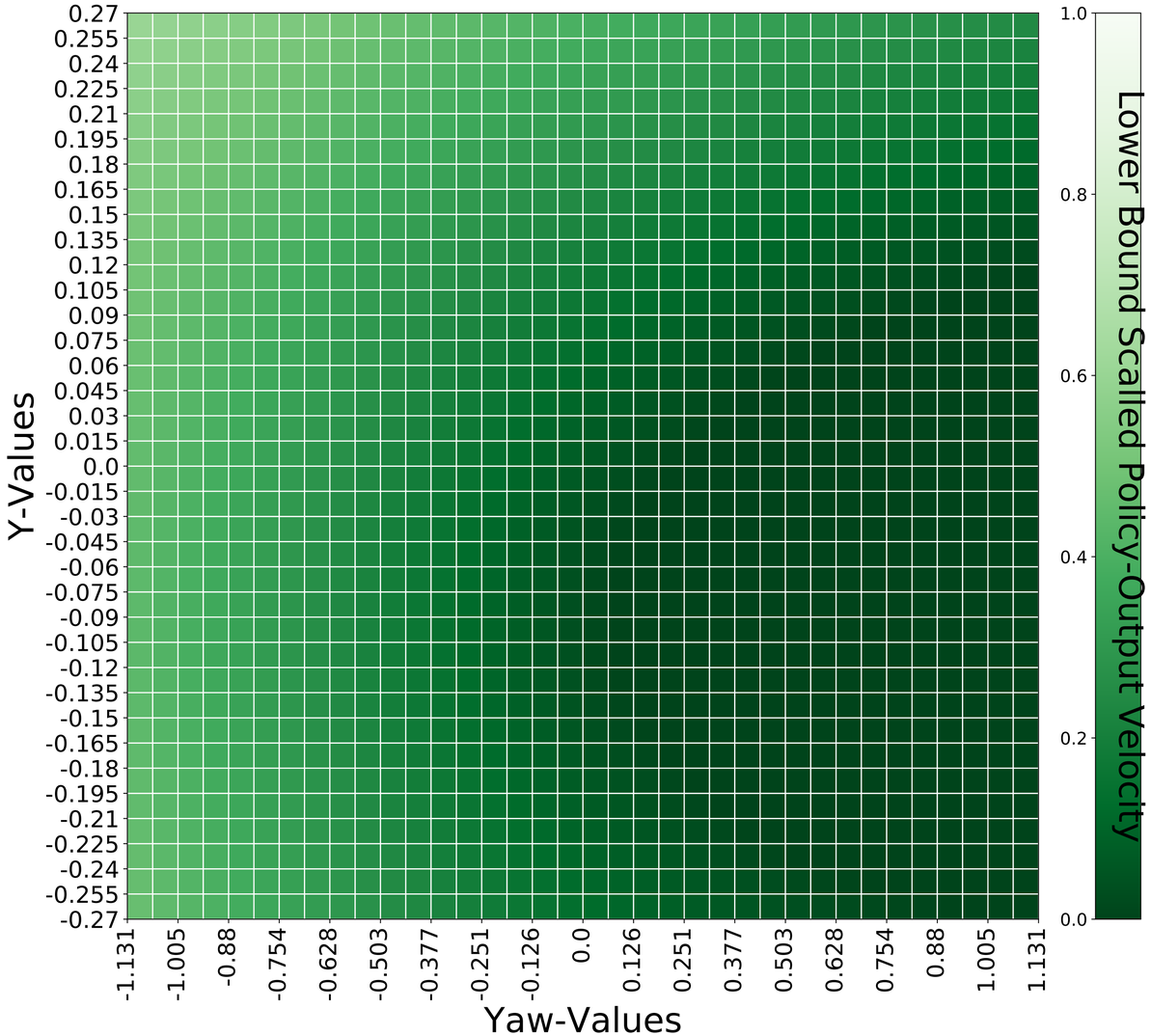


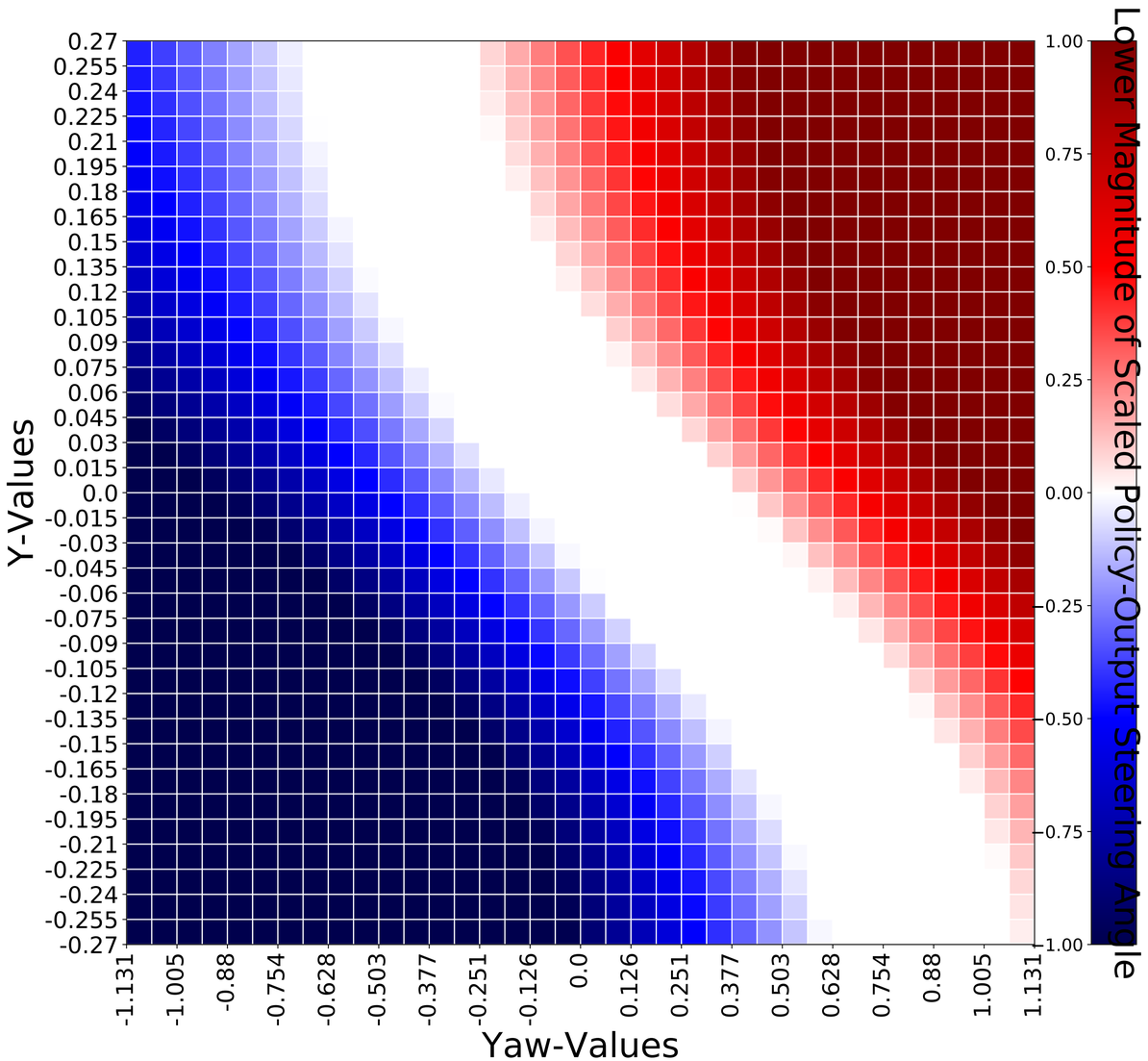


Line-Following Stability Results
xDot:1.1, yDot:-0.32307, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: dc806fe7-2c63-42d4-bfca-24322814862d
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_48_06_590332

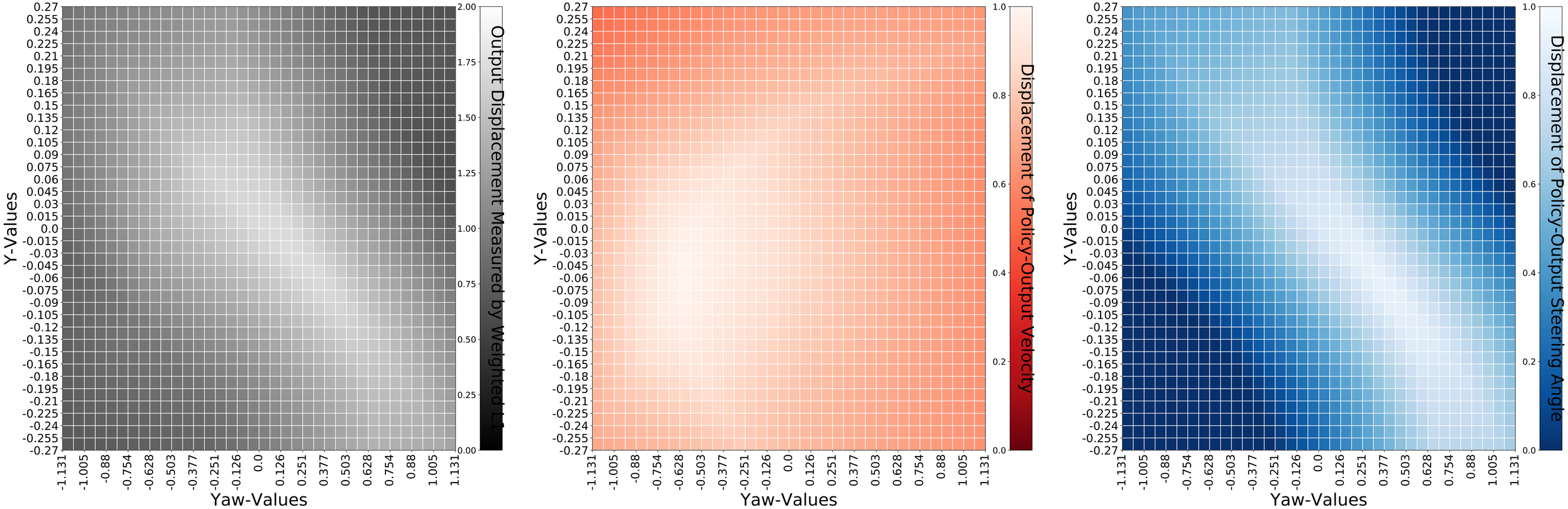


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:-0.32307, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: a0e1898a-55b1-4733-9756-d3028a804599
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_50_59_591831

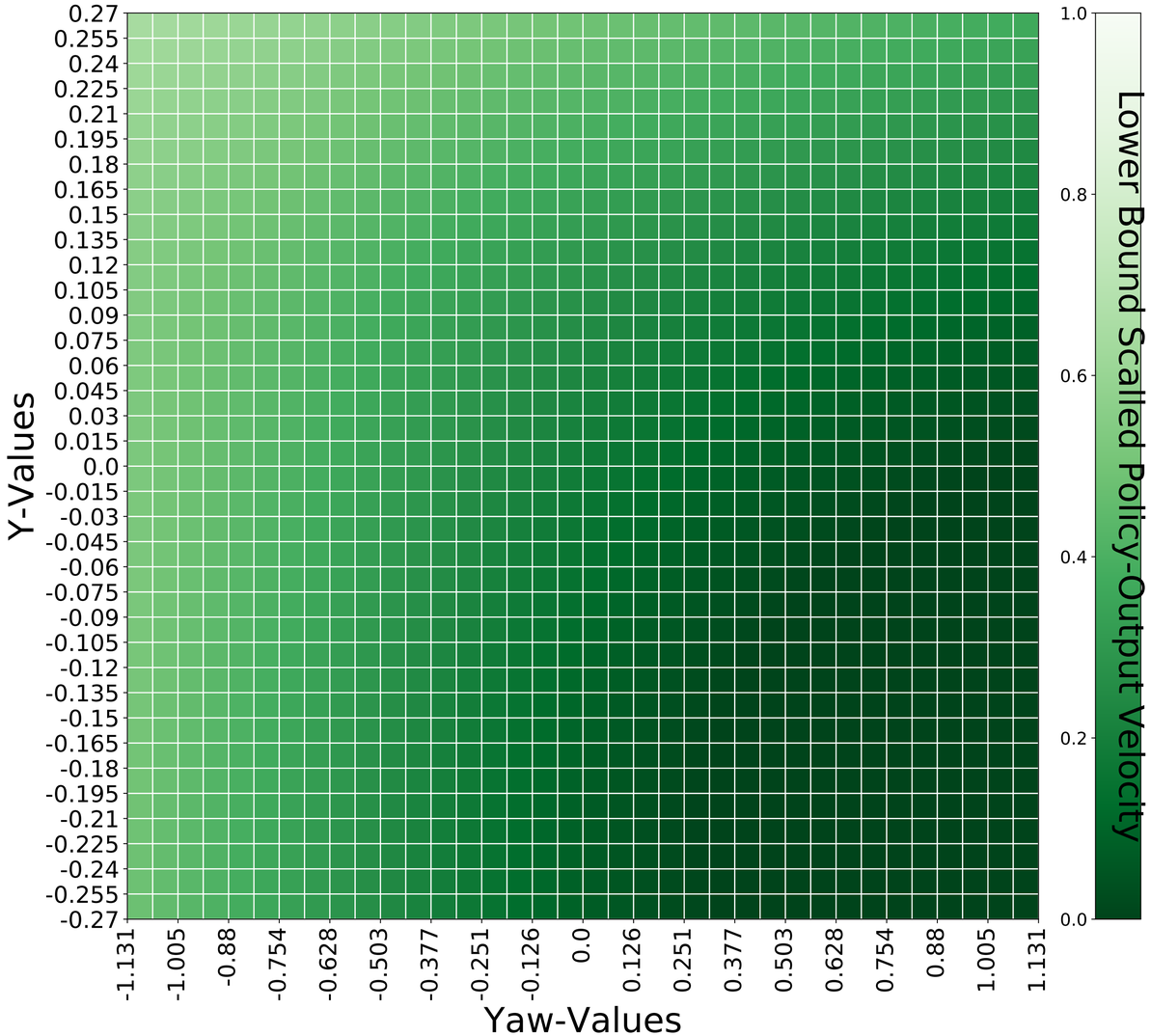


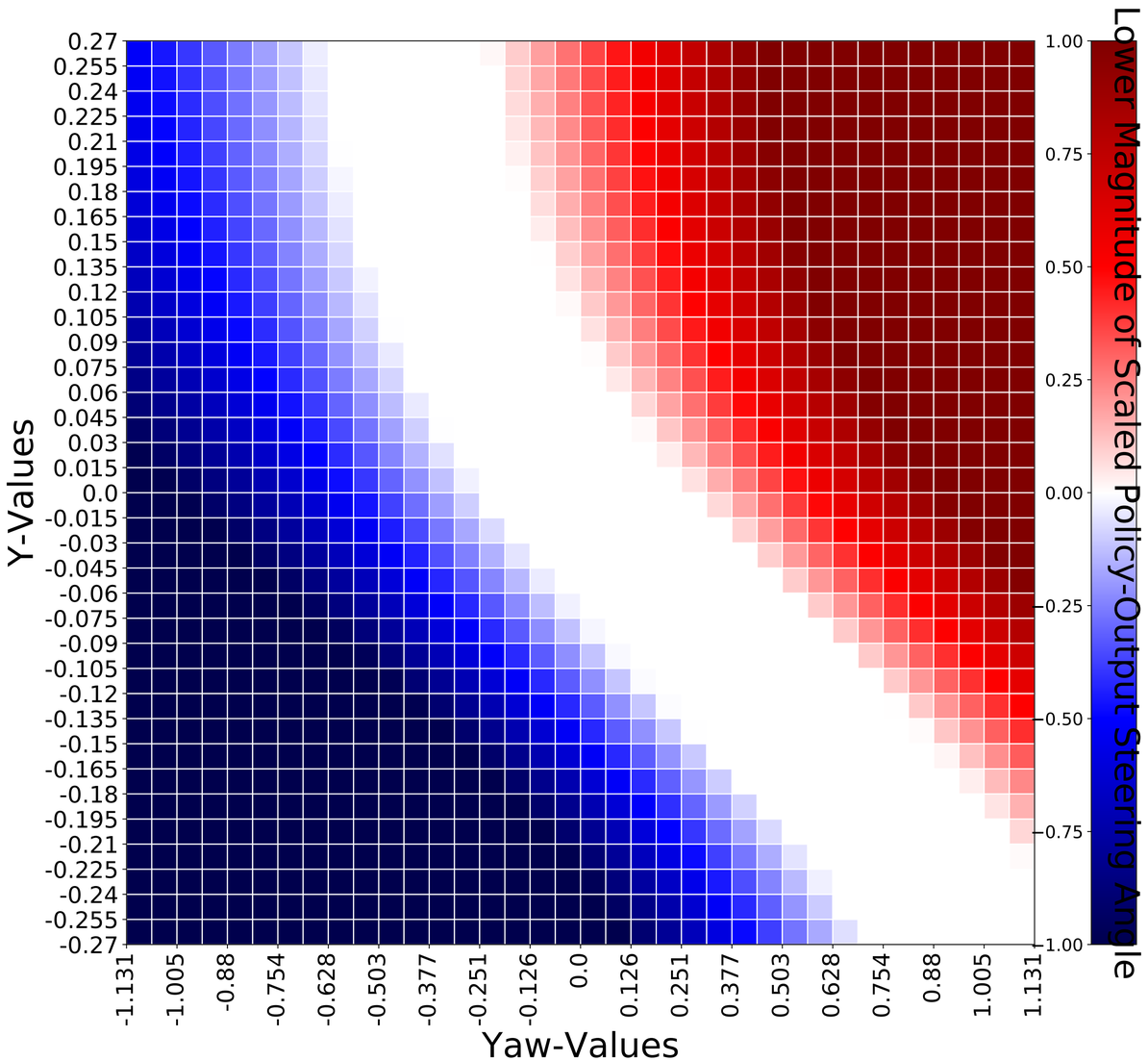


Line-Following Stability Results
xDot:1.1, yDot:-0.32307, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 484c6ad6-cb01-46c2-99a0-bf302ce40c26
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_04_50_02_776589

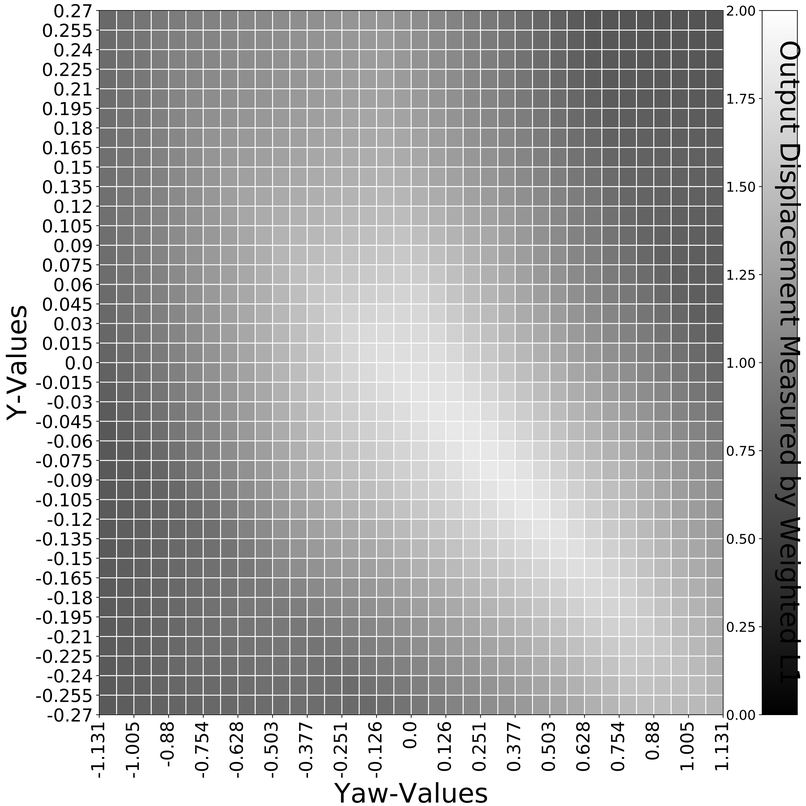


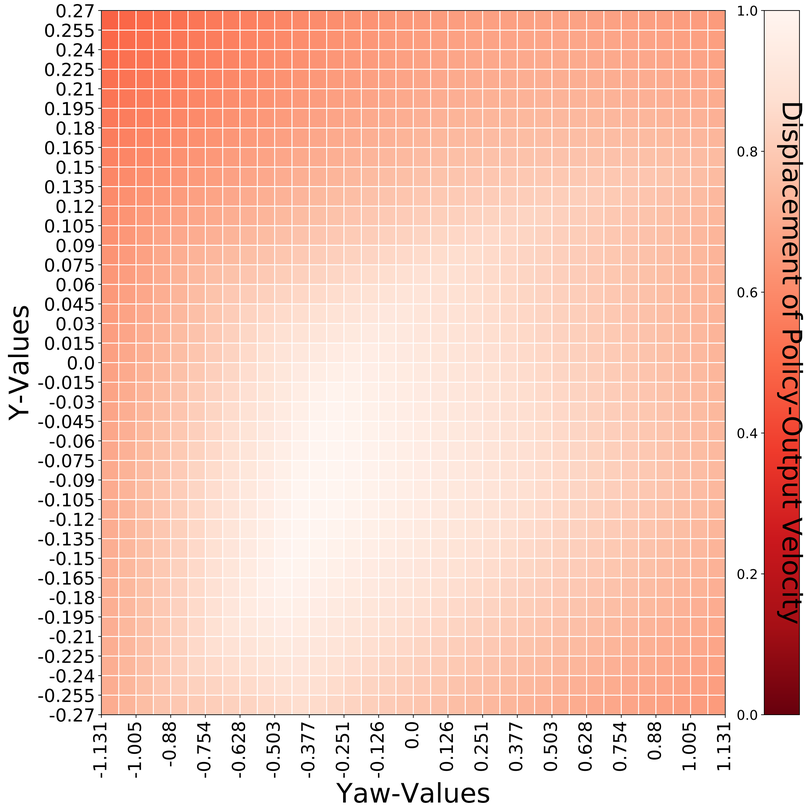
Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:-0.32307, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 64d50715-32da-4158-88f2-2e73e2d4aecd
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_05_02_59_501657

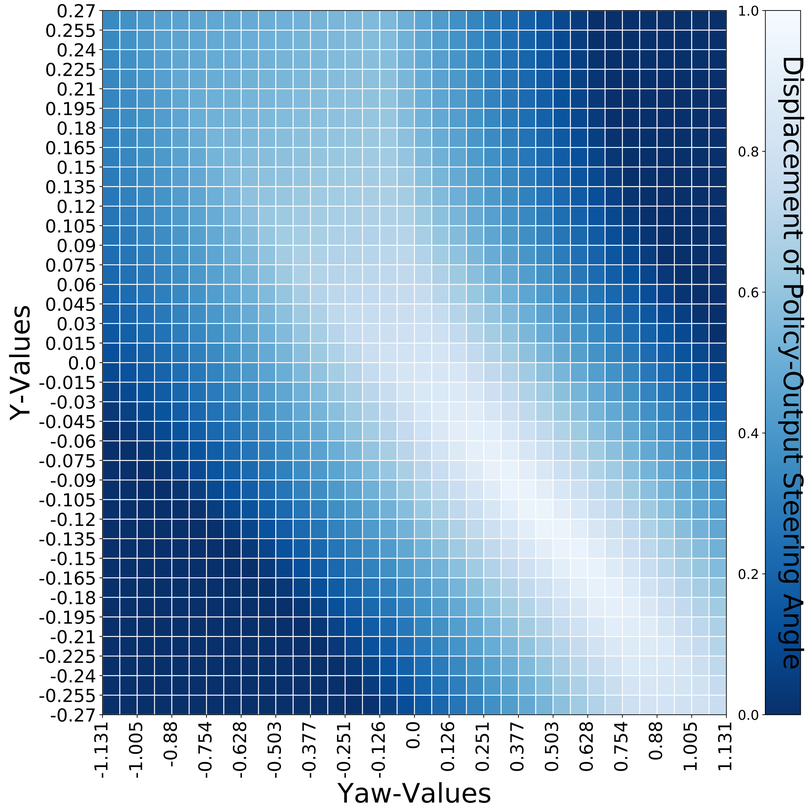




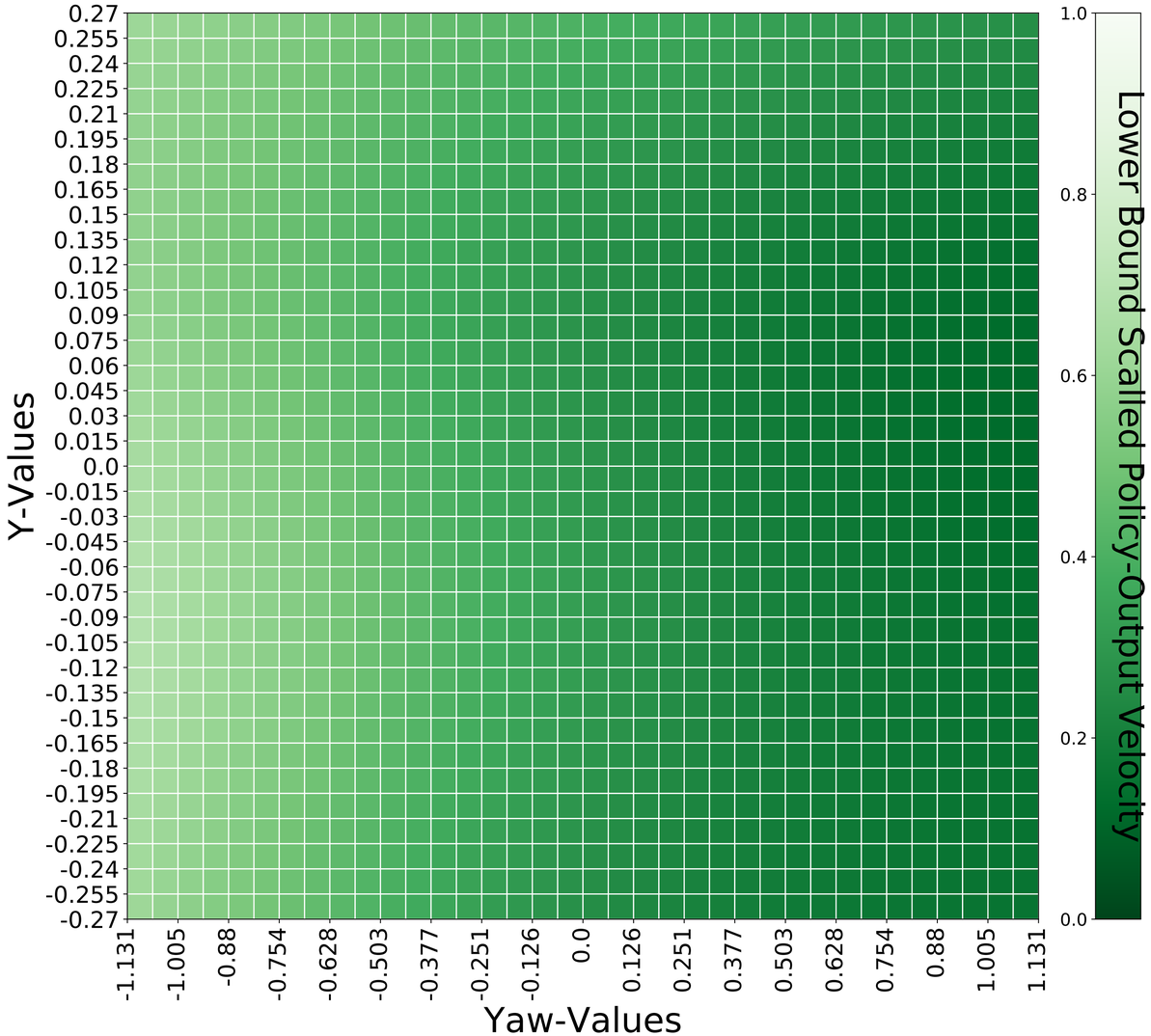
Line-Following Stability Results
xDot:1.1, yDot:-0.32307, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: aeeb9b37-8143-4f0e-b7dd-9df4594ed391
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_05_01_56_978509

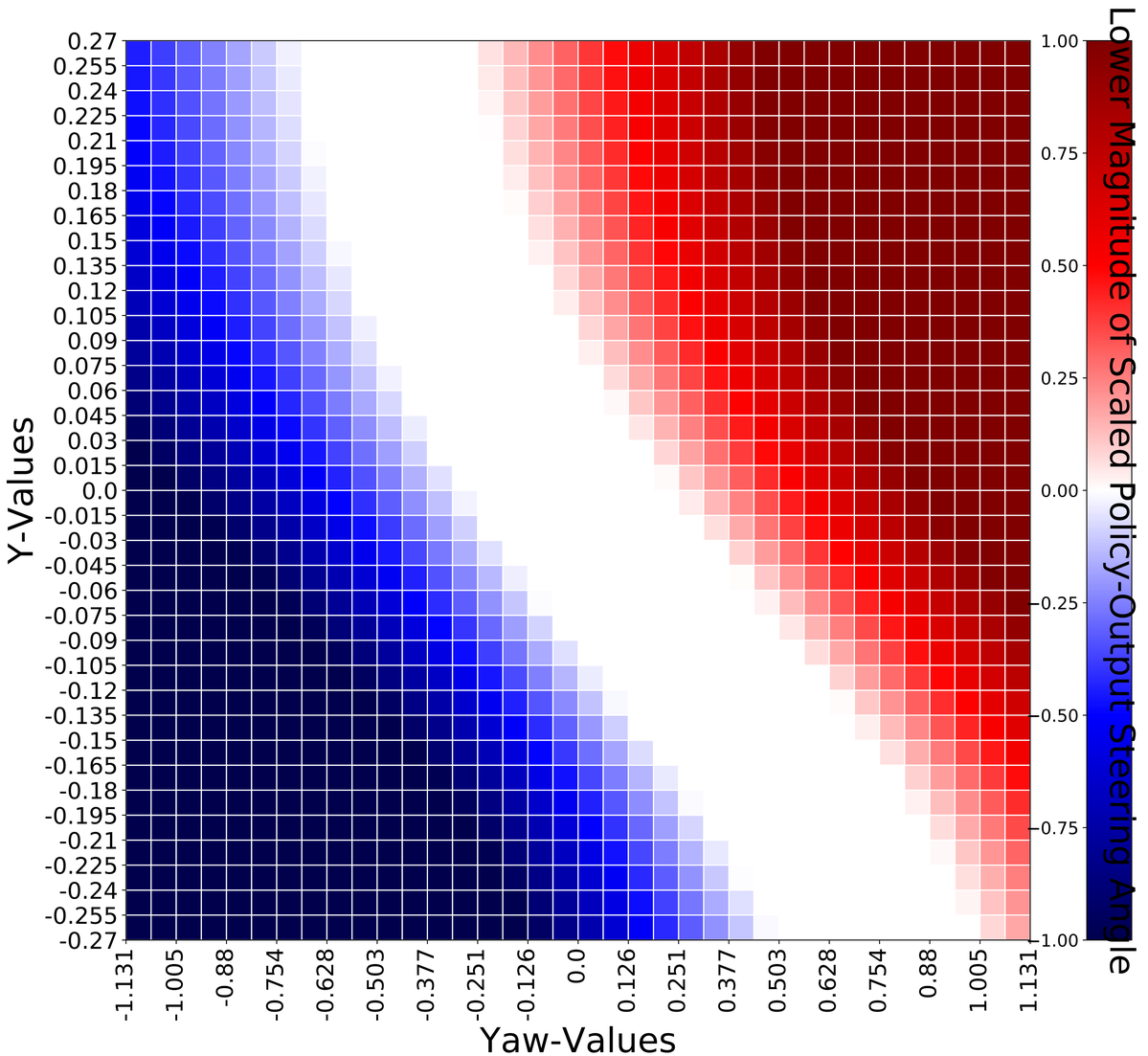




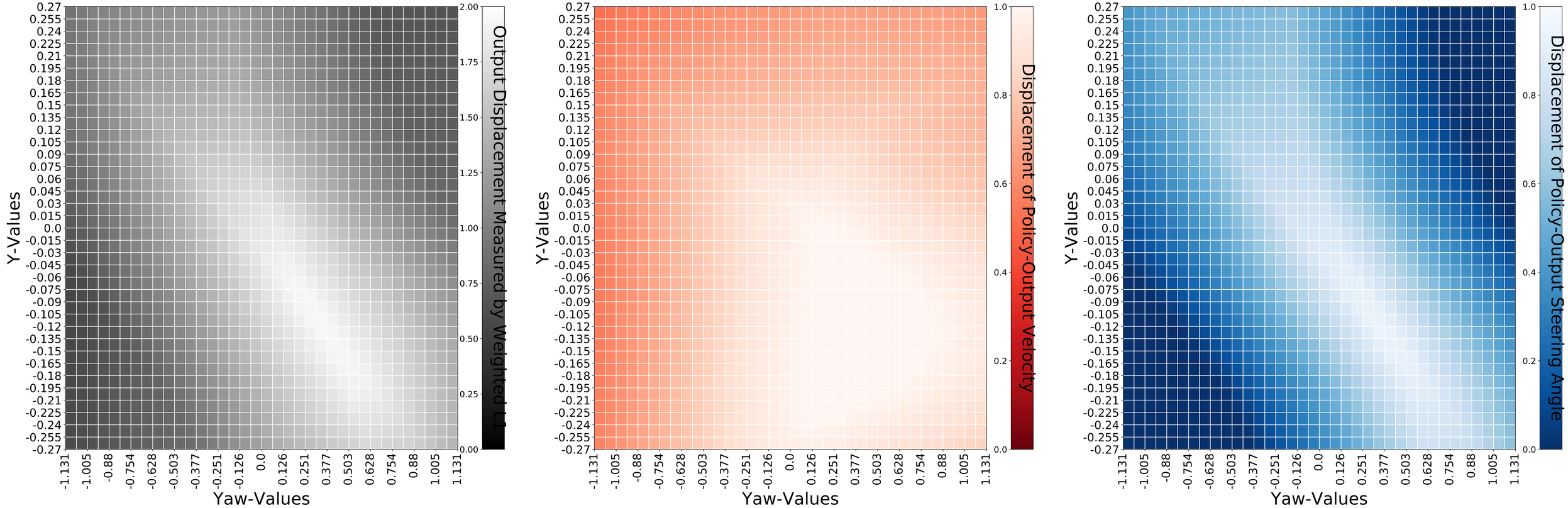


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0.415384, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: e83e90d2-64c5-413e-a473-dec38392e143
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_05_04_54_289755

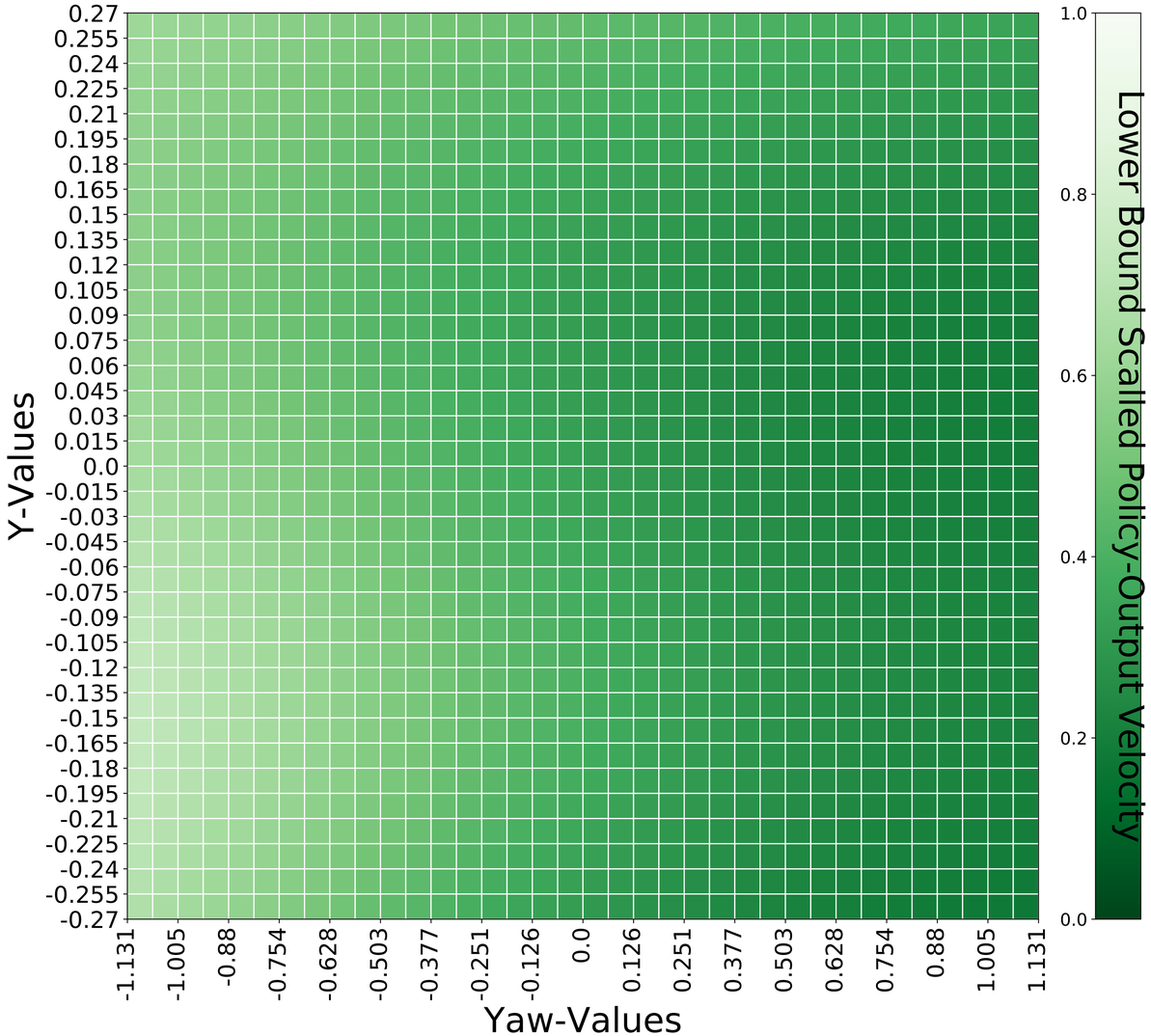


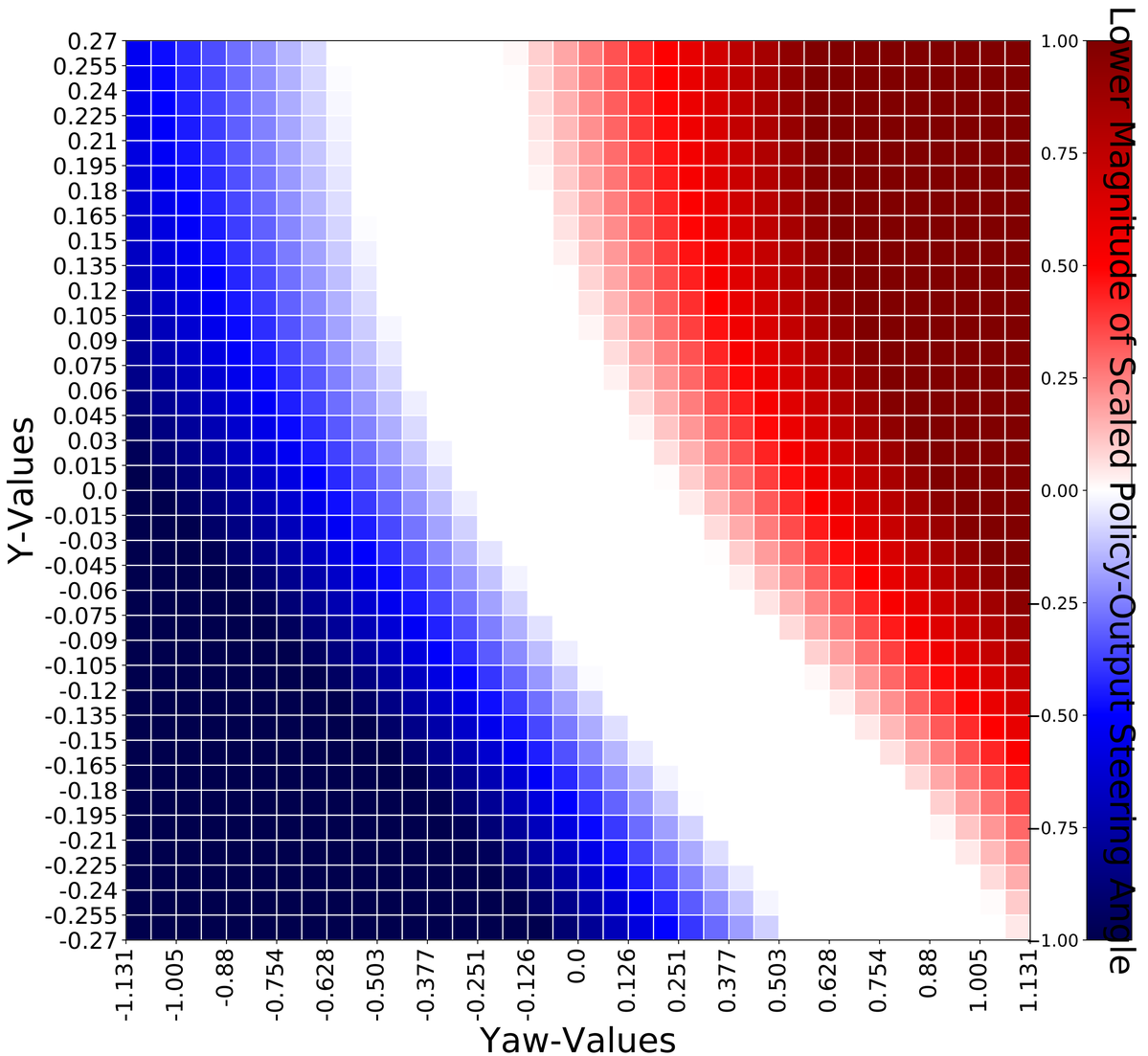


Line-Following Stability Results
xDot:1.1, yDot:0.415384, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 8b8f55ce-8288-4fb5-a29e-9eff6a484d4e
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_05_03_59_008236

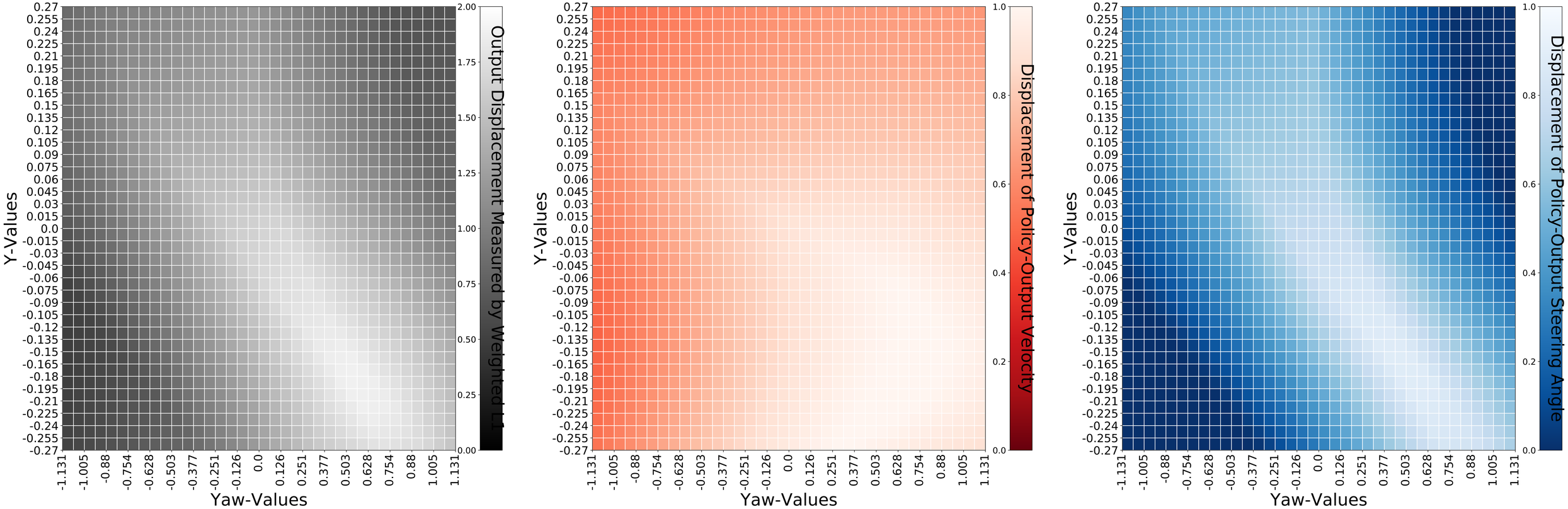


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0.415384, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 0cdc11c6-fb9e-402b-9ba2-4ad5c389e064
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_05_06_47_759805

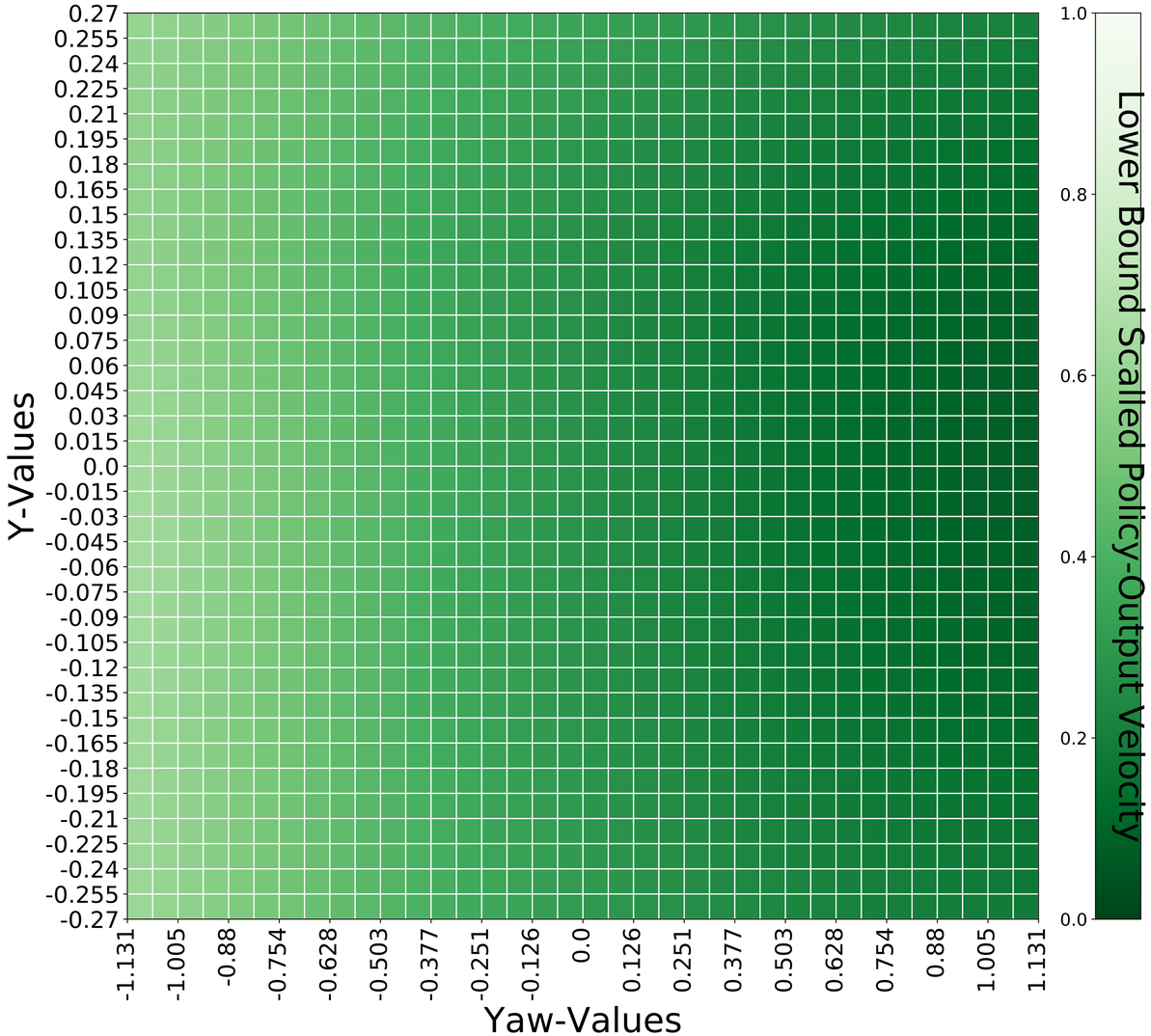


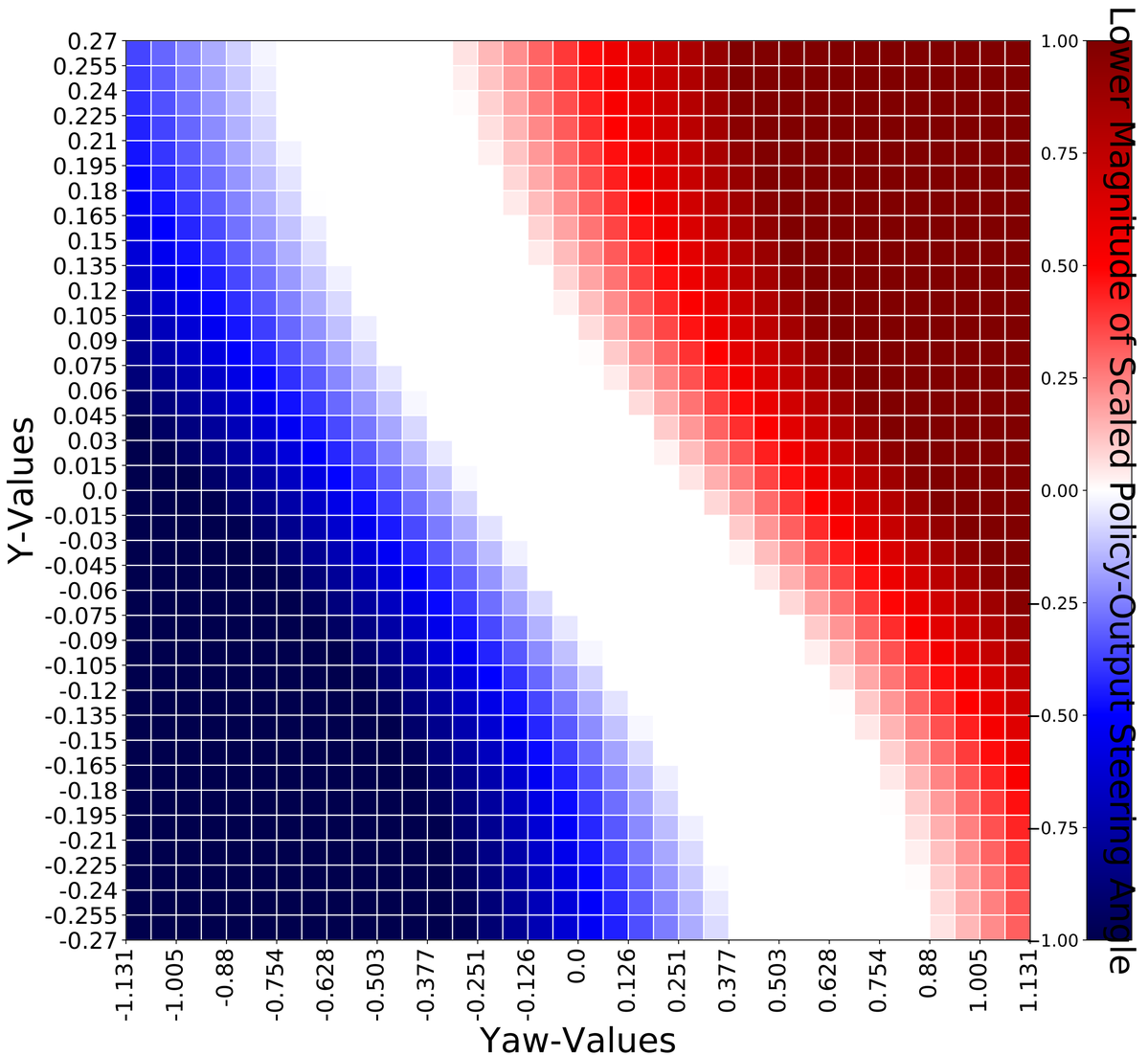


Line-Following Stability Results
xDot:1.1, yDot:0.415384, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: ac41dd82-0a9b-46a1-87da-15d7373b0fcb
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_05_05_49_062191

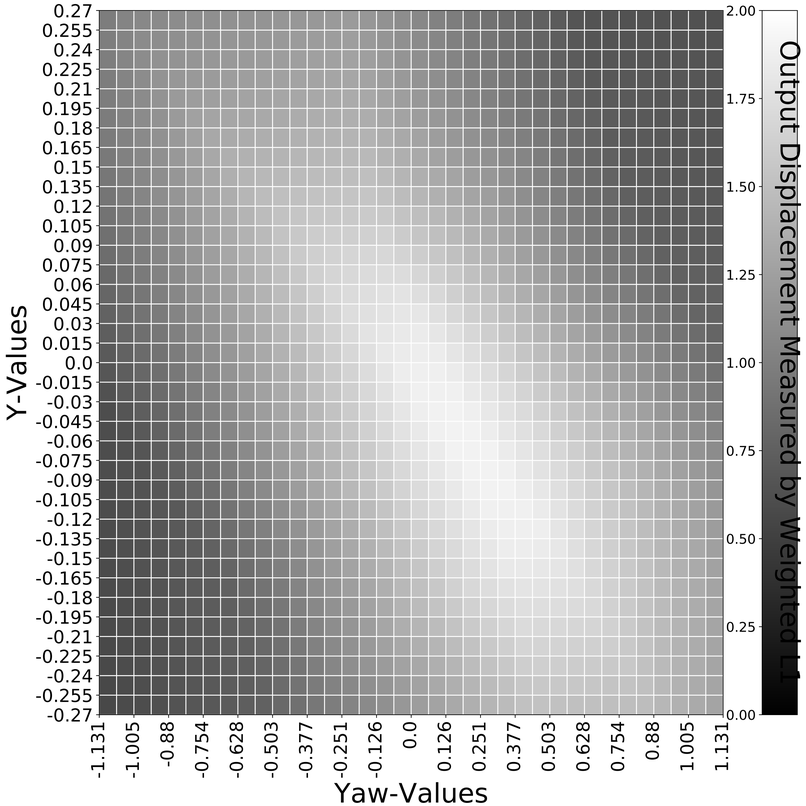


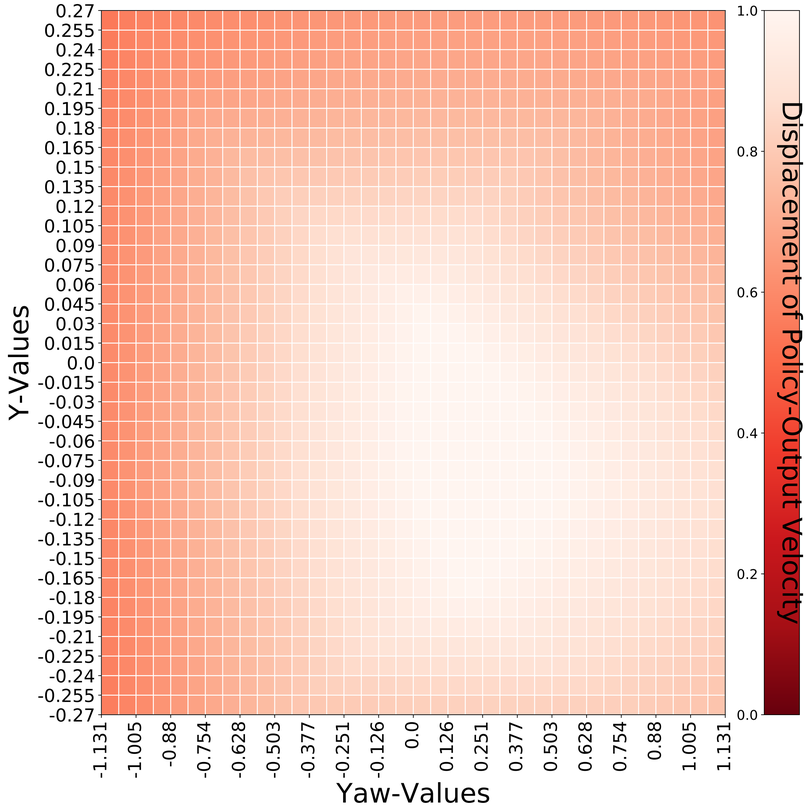
Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0.415384, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: cb143748-040a-474f-b542-86a09af97475
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_05_08_42_834374

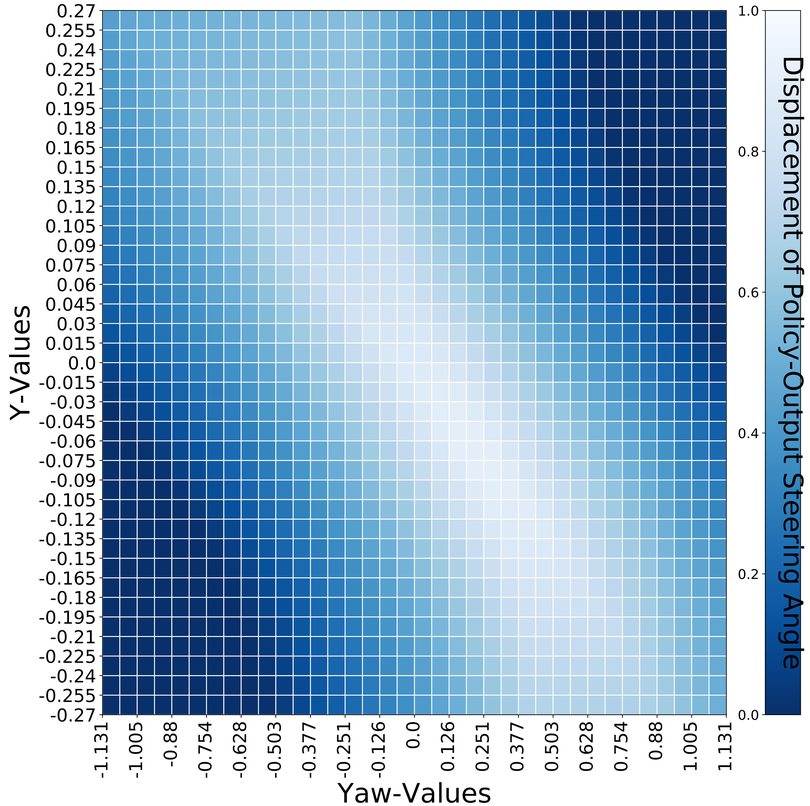




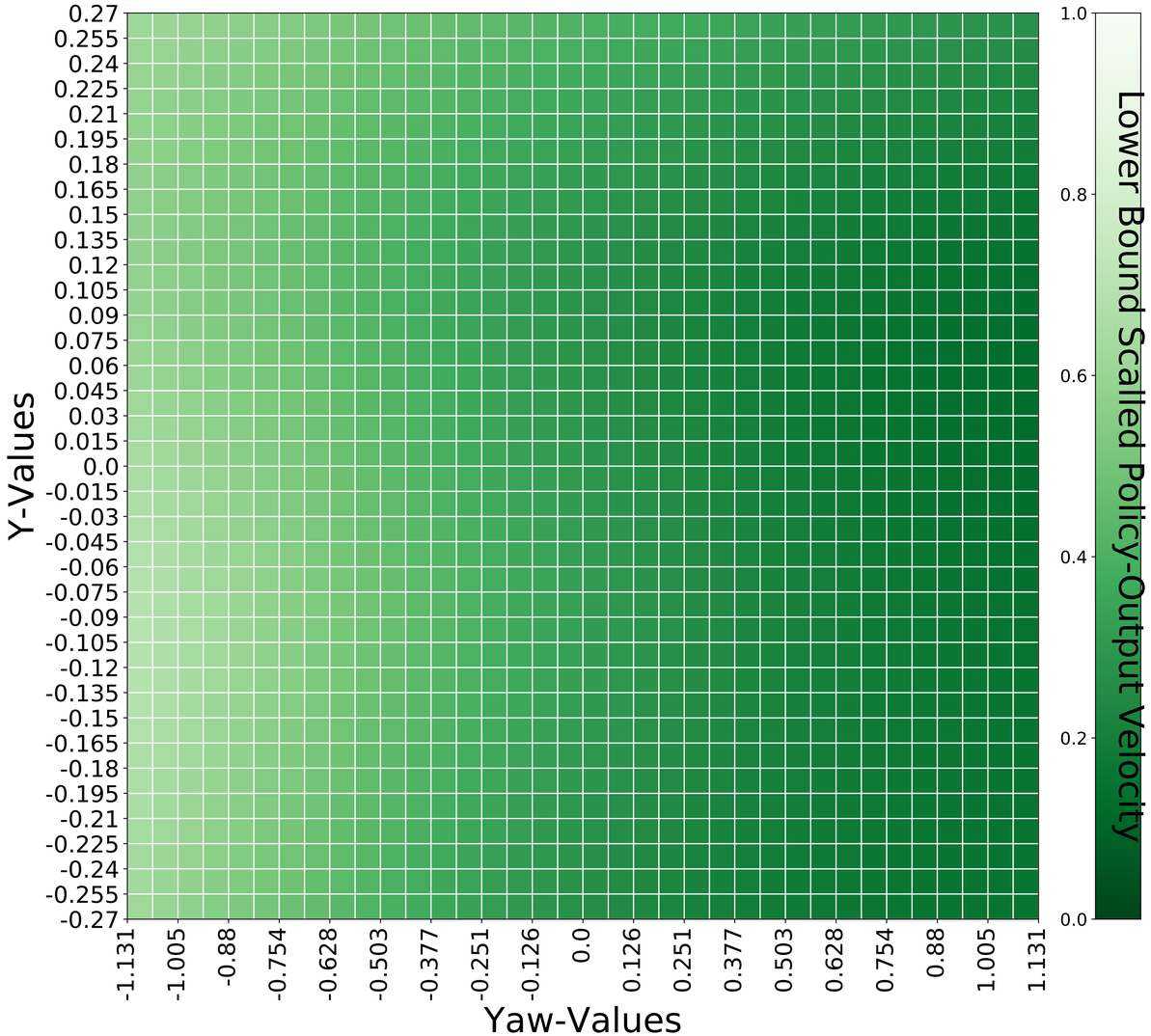
Line-Following Stability Results
xDot:1.1, yDot:0.415384, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: f44f38ed-2a46-4a2f-b8d8-cfcf2146e5e9
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_05_07_43_626777

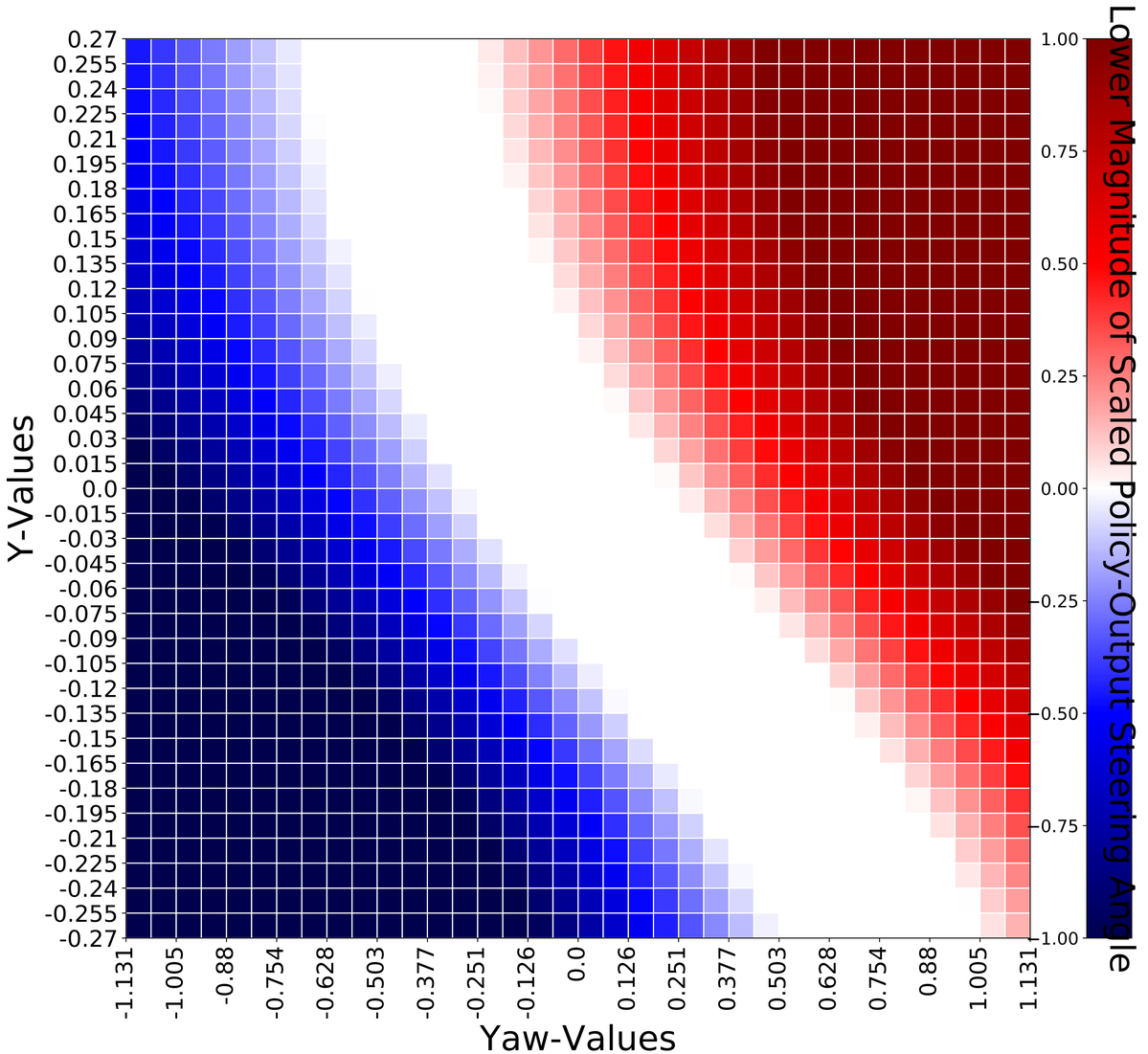




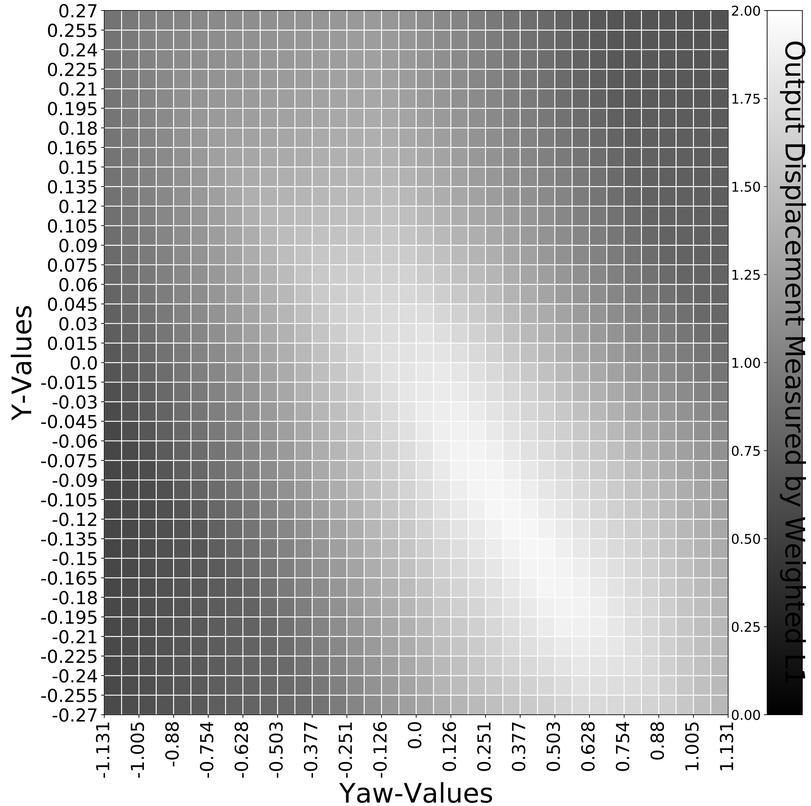


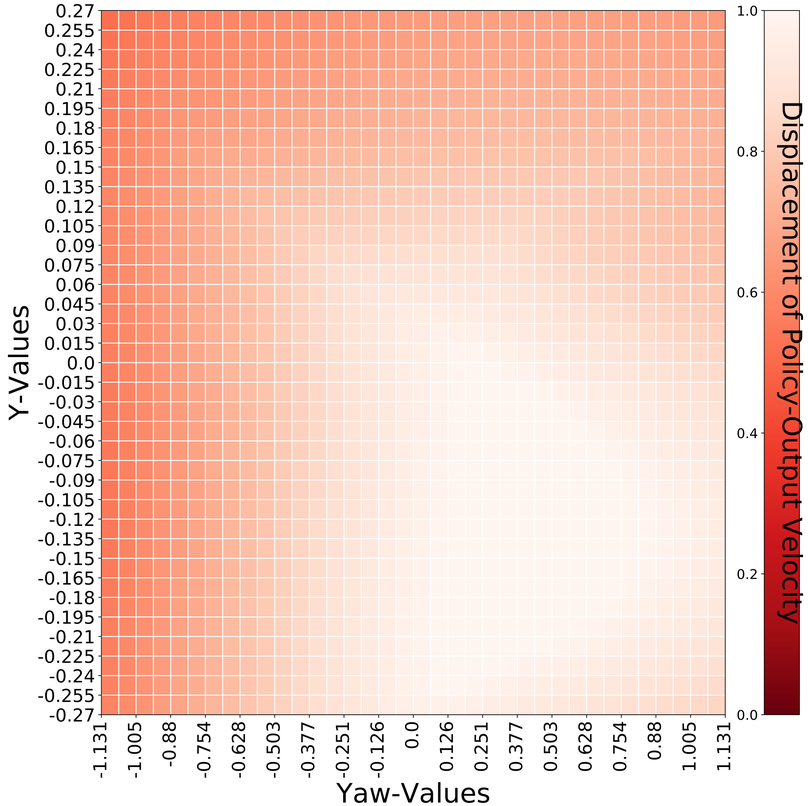
Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0.415384, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 41f26bfb-a475-4ac9-be57-e1b35517f051
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_05_20_14_437093

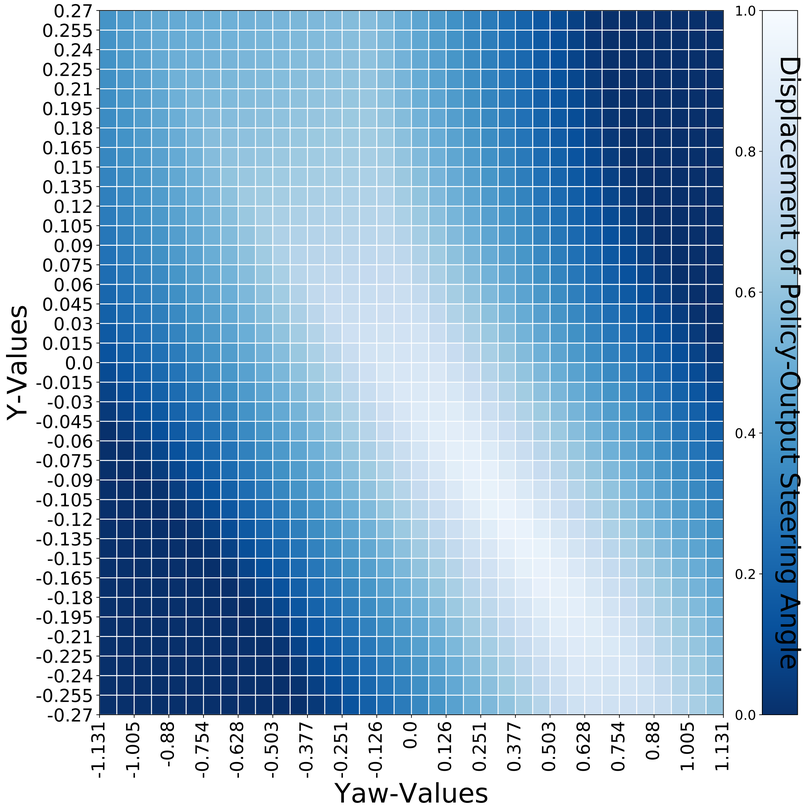




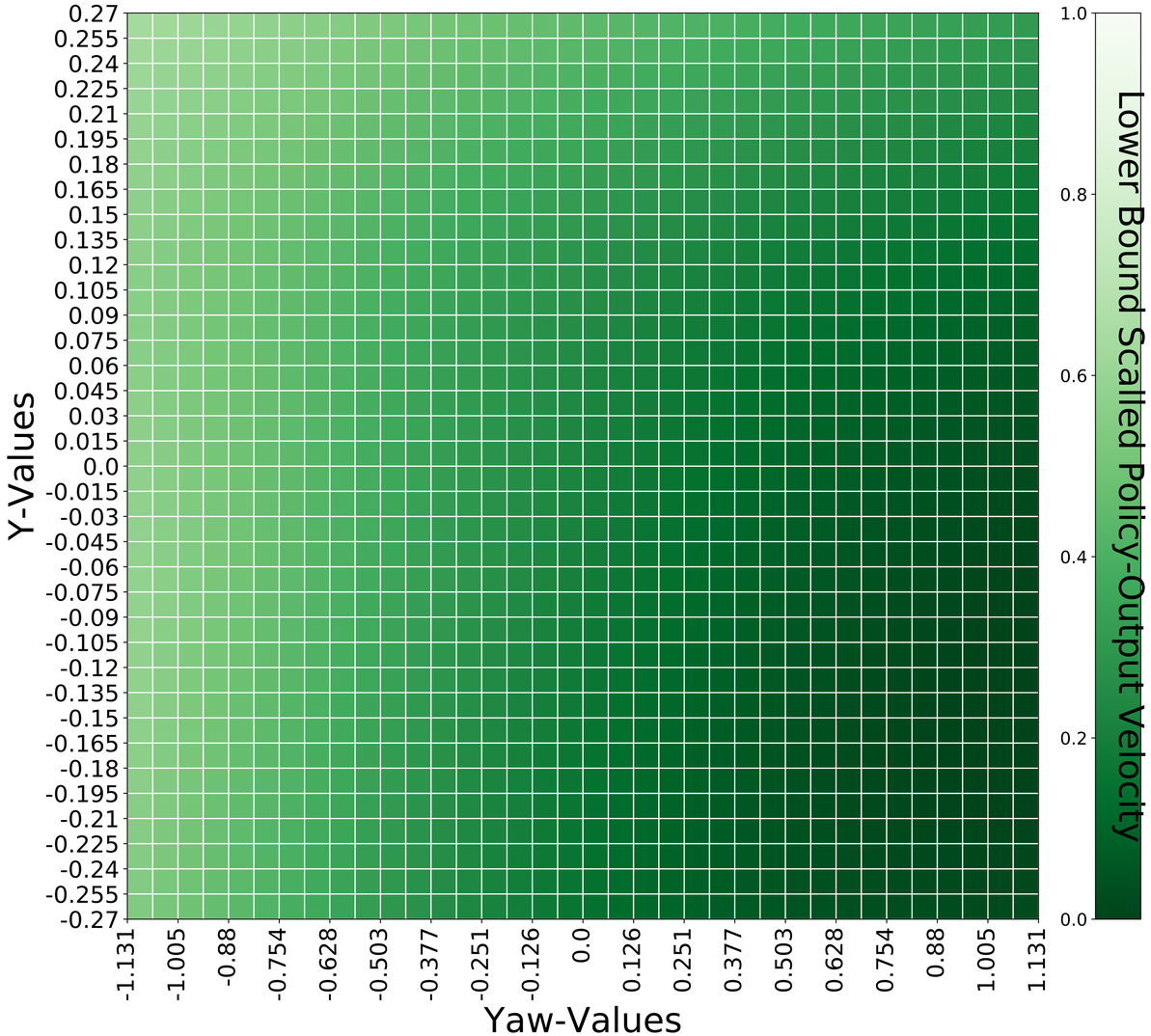
Line-Following Stability Results
xDot:1.1, yDot:0.415384, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 68fa4e79-b9f6-4dc3-bc71-723f7a9167e6
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_05_19_20_016046

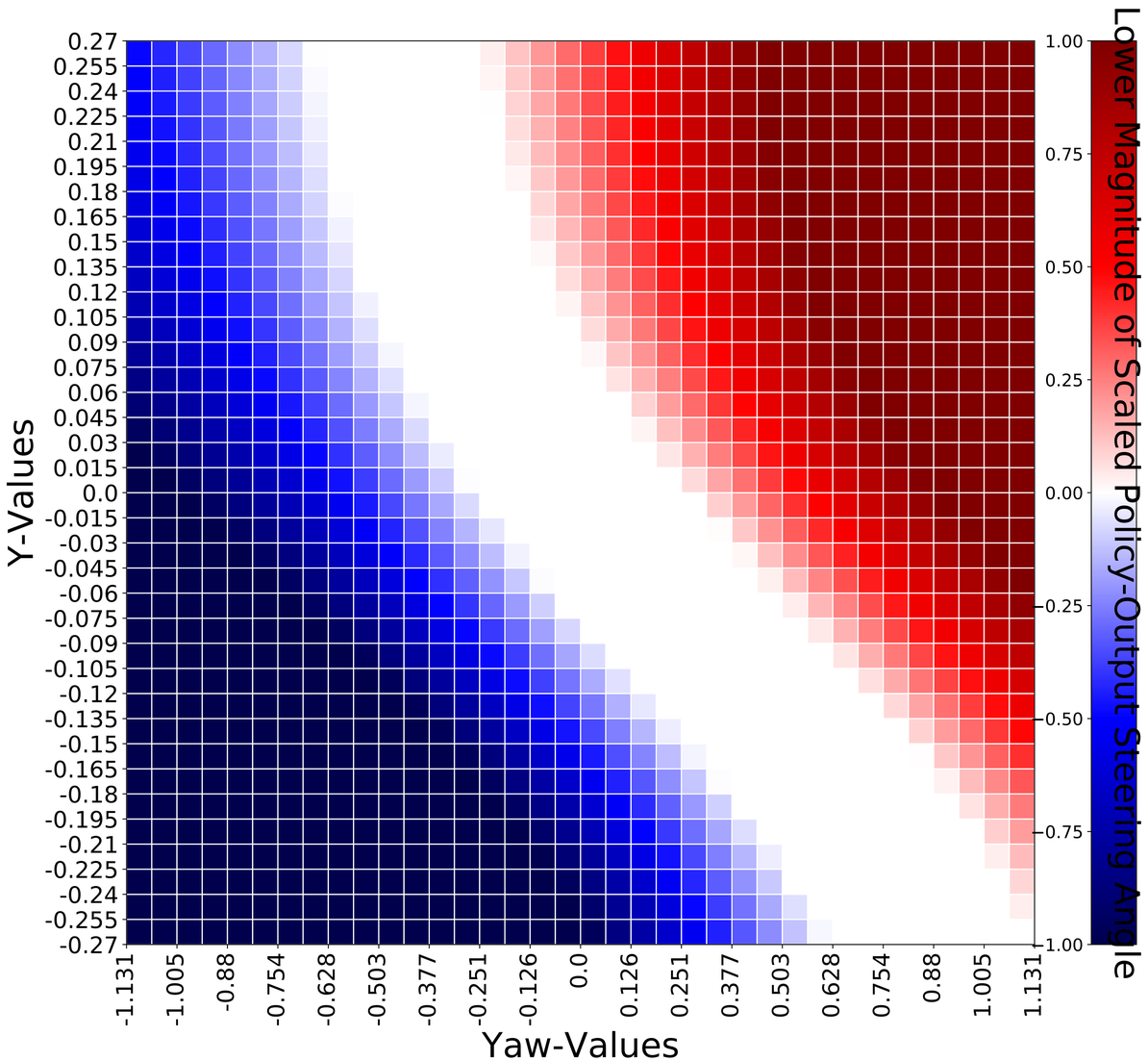




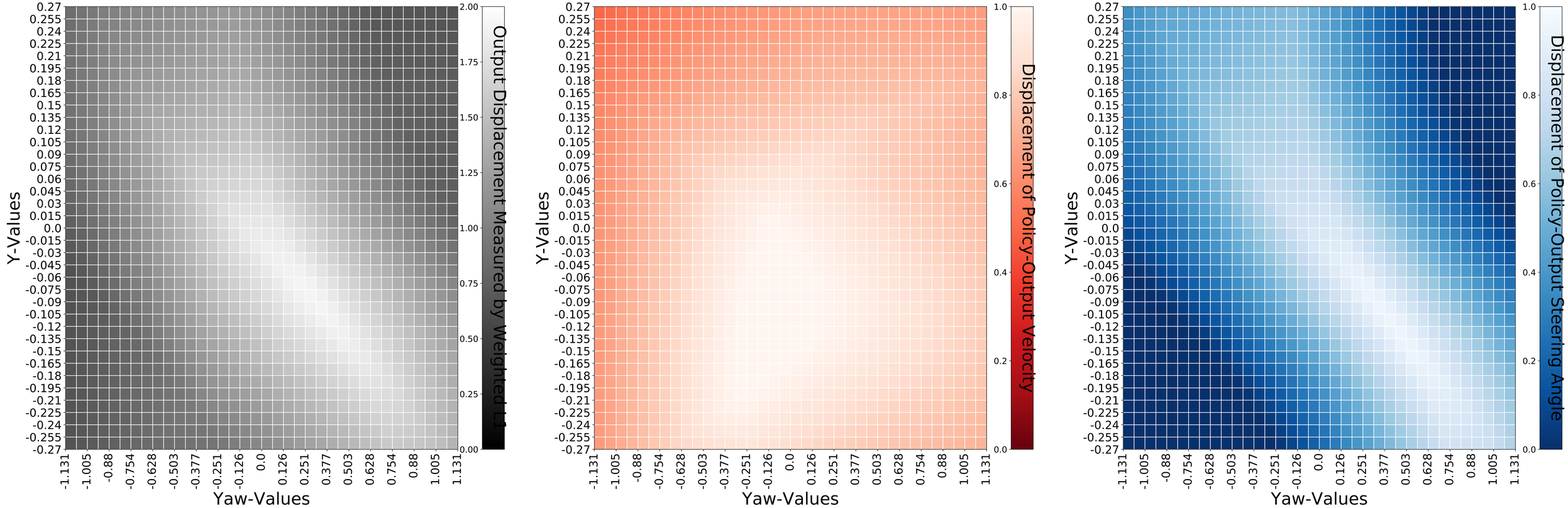


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: ea4b1364-7aa1-4b82-aa3e-c01ce88bdc9b
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_50_32_798533

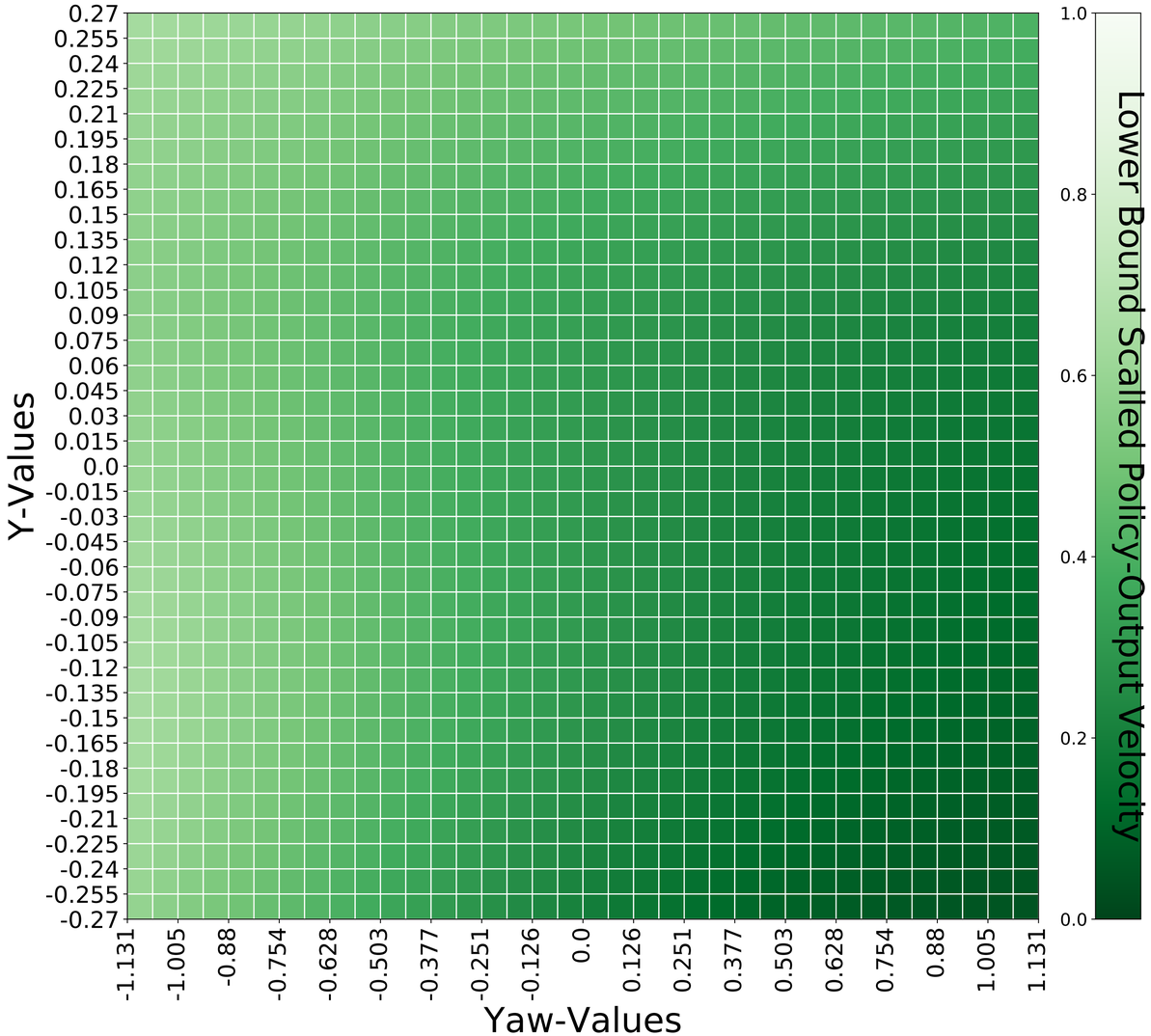


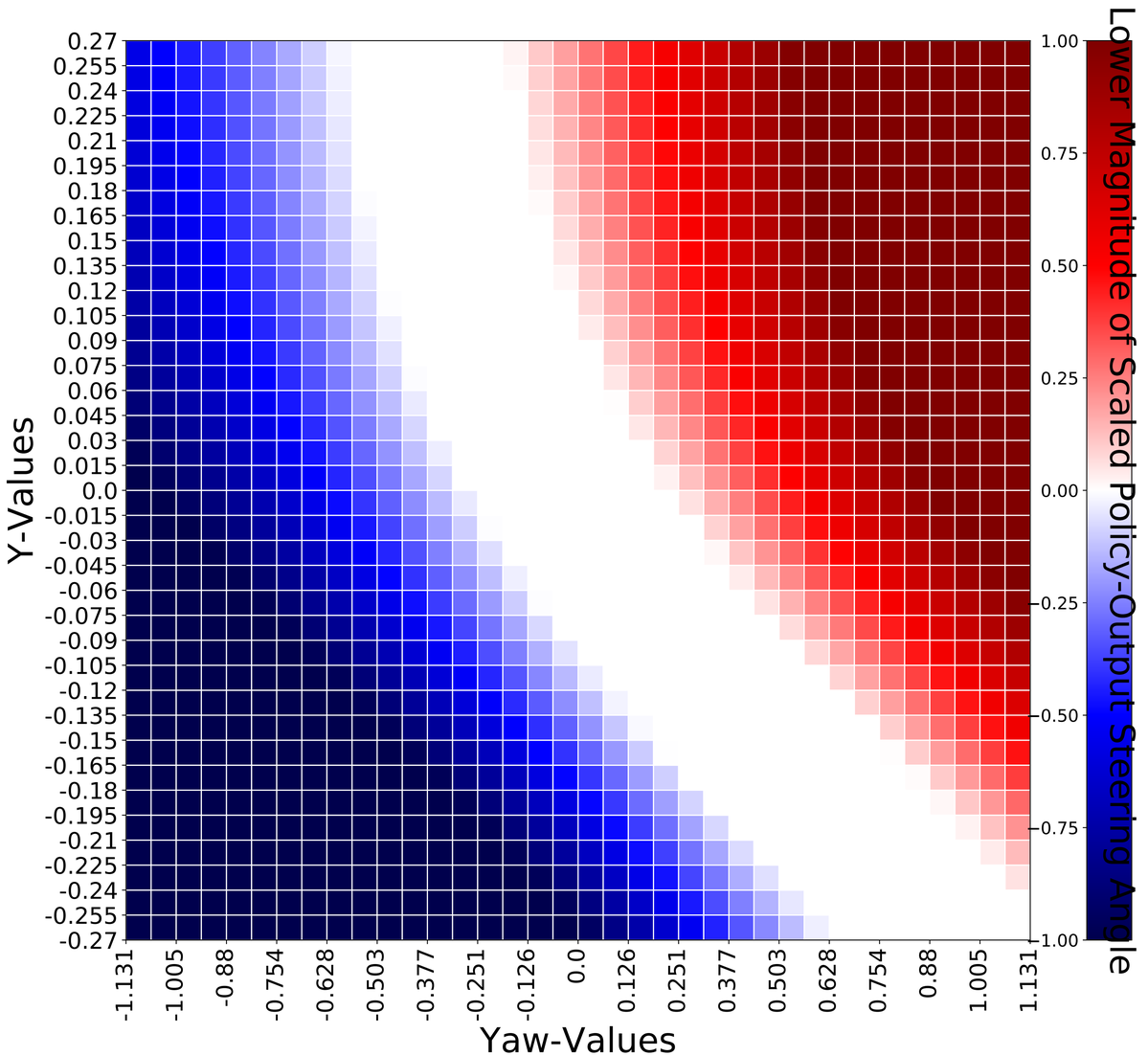


Line-Following Stability Results
xDot:1.1, yDot:0, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: d304b586-4caf-4169-baba-6d5d3ebee851
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_49_36_170244

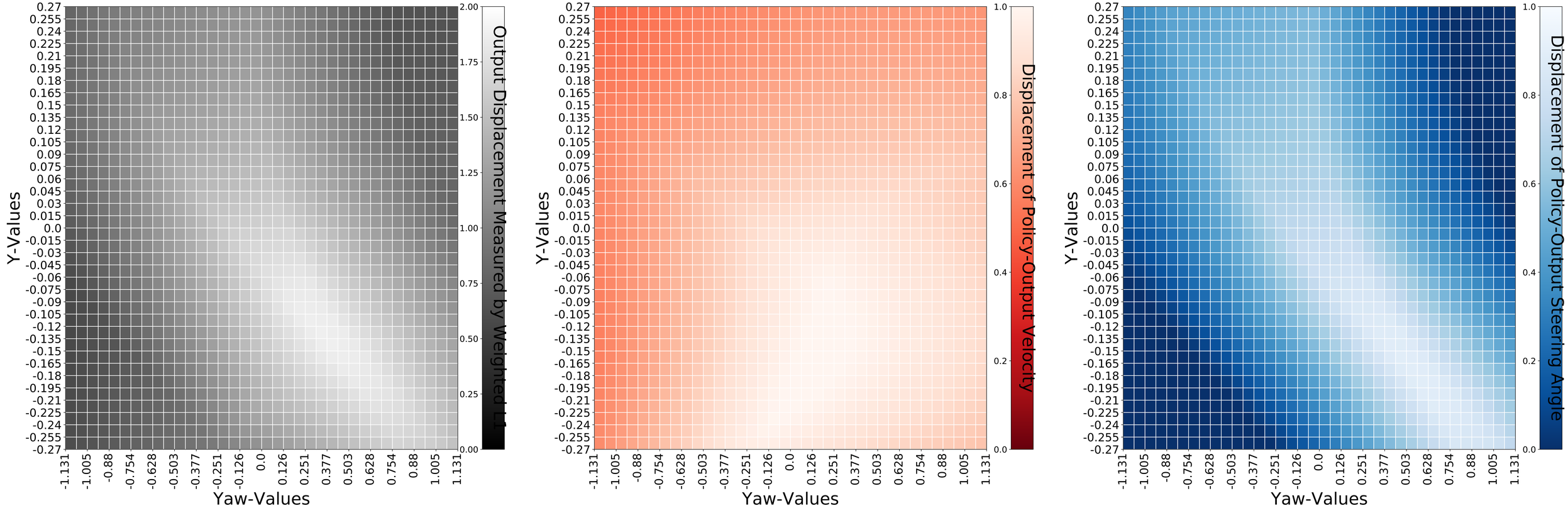


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 655157cb-4e61-473d-9c27-868cca599bdf
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_52_31_197875

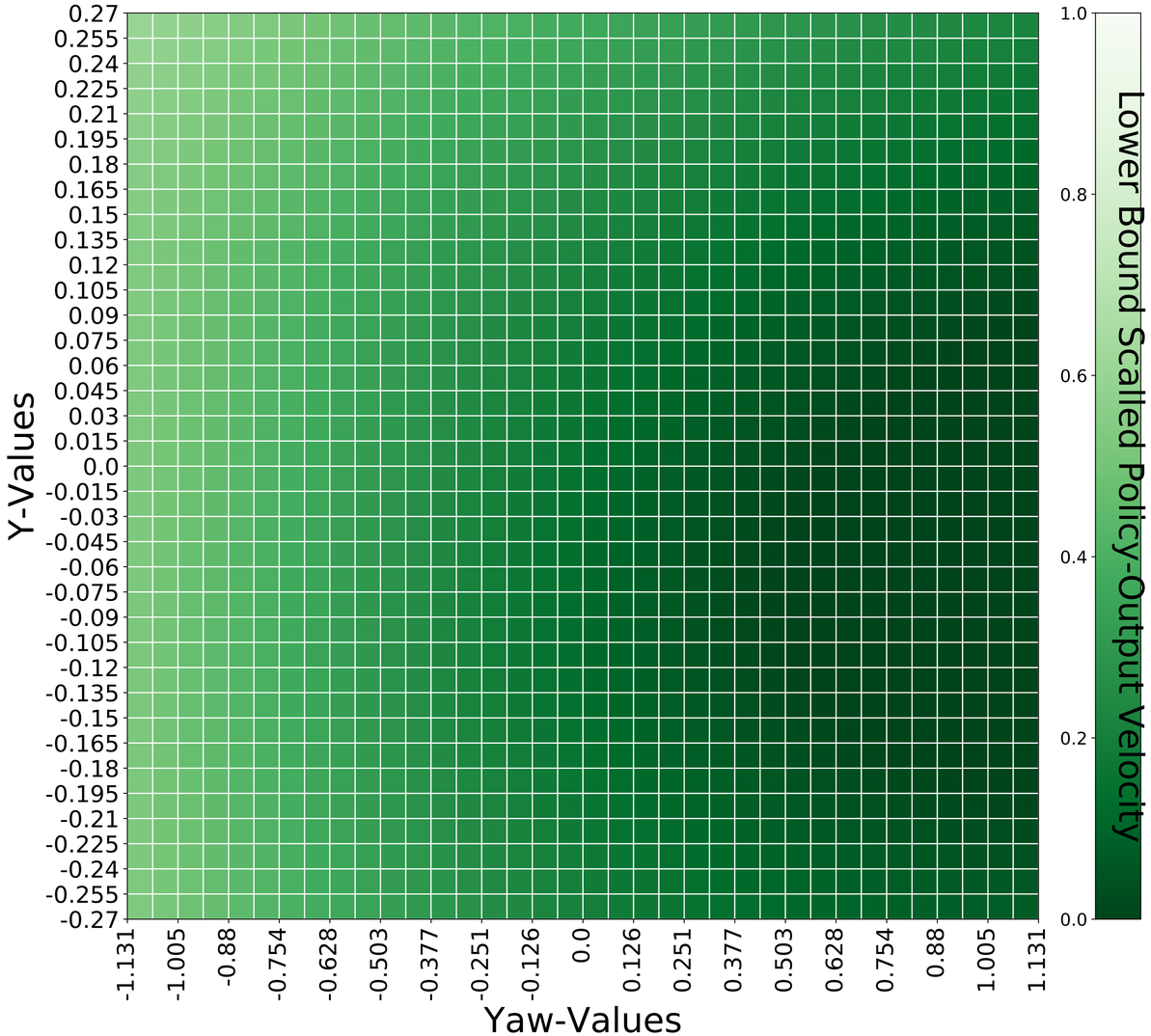


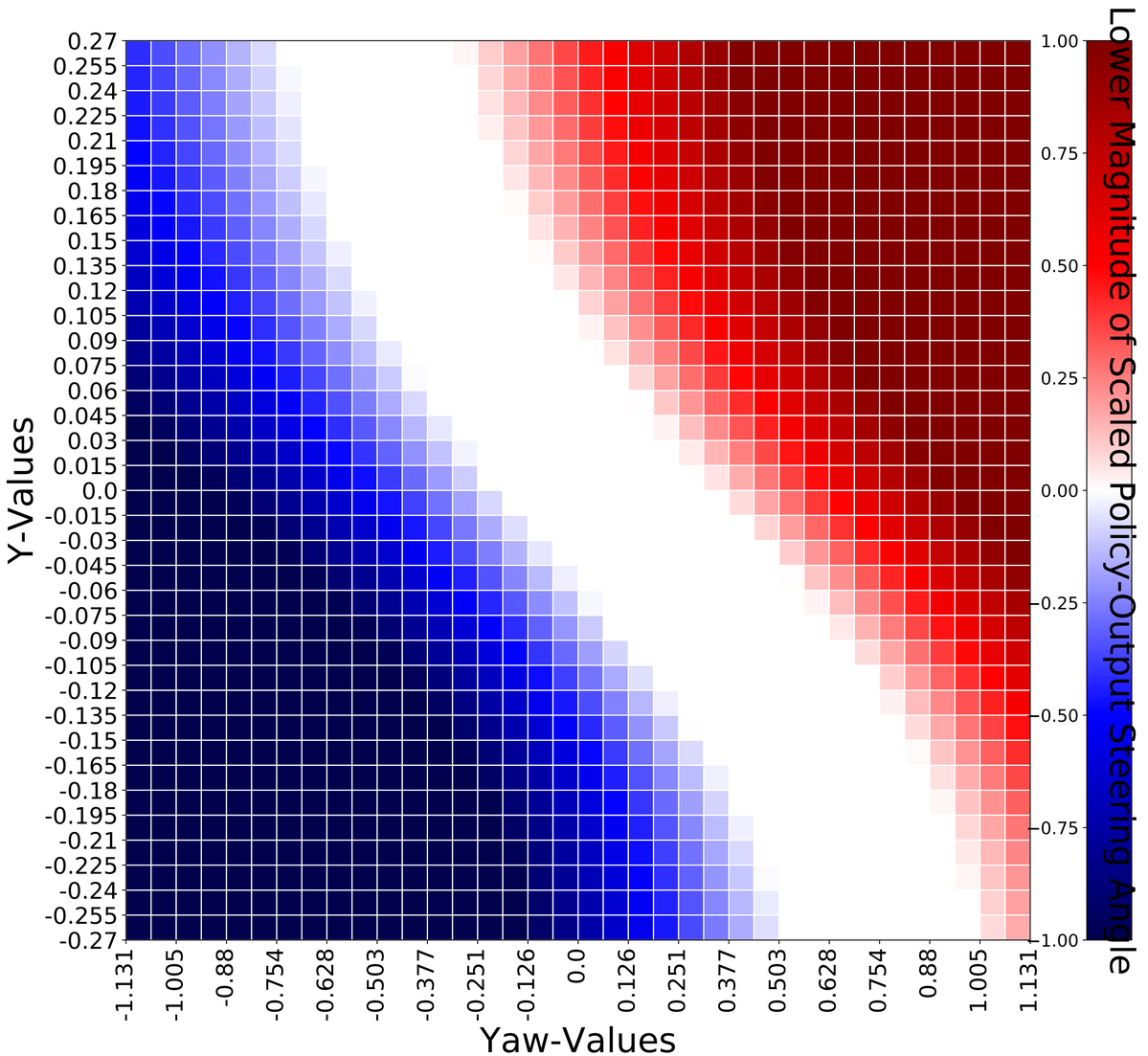


Line-Following Stability Results
xDot:1.1, yDot:0, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 7169b96b-1742-41ee-bf0c-8b52547c04d1
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_51_26_505538

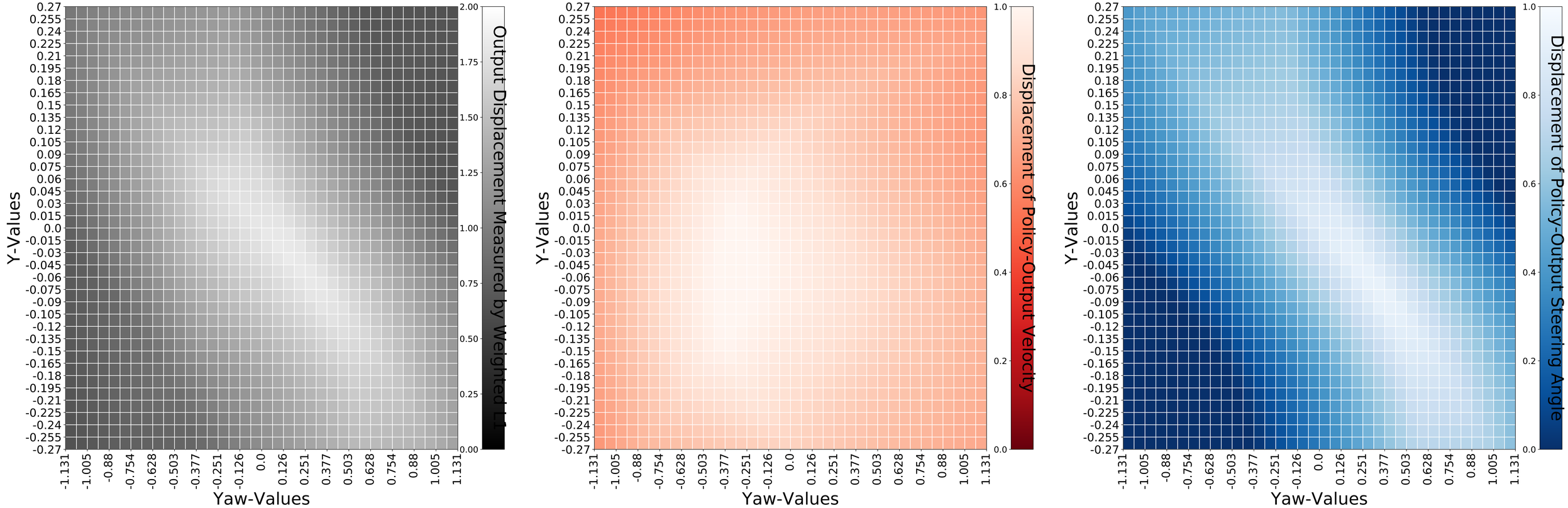


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 77d89fbd-034f-44c8-ba91-c816c34eb875
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_54_35_962506

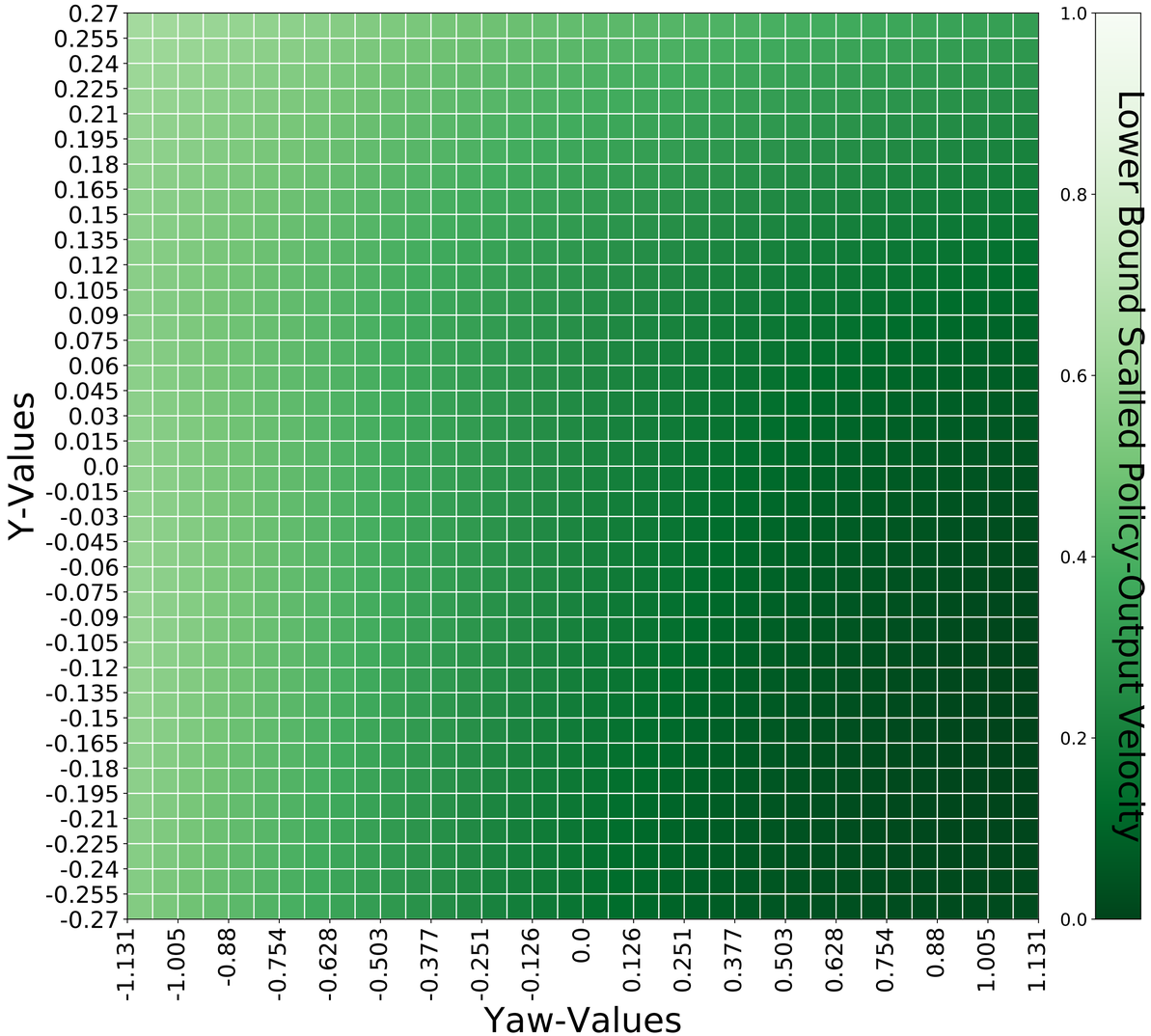


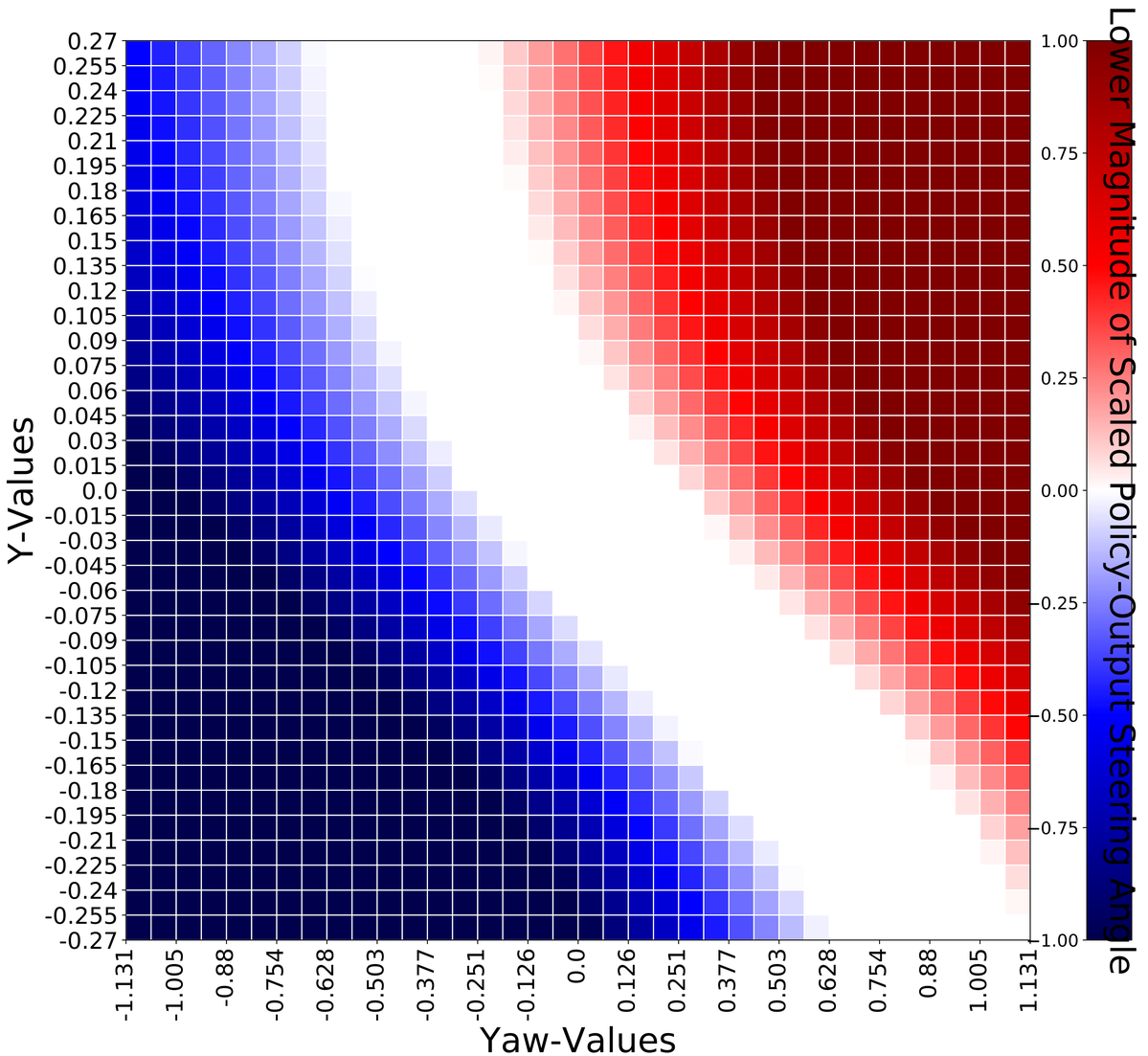


Line-Following Stability Results
xDot:1.1, yDot:0, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: e27051c6-79d5-4f6c-8138-65239cd201ea
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_53_33_829206

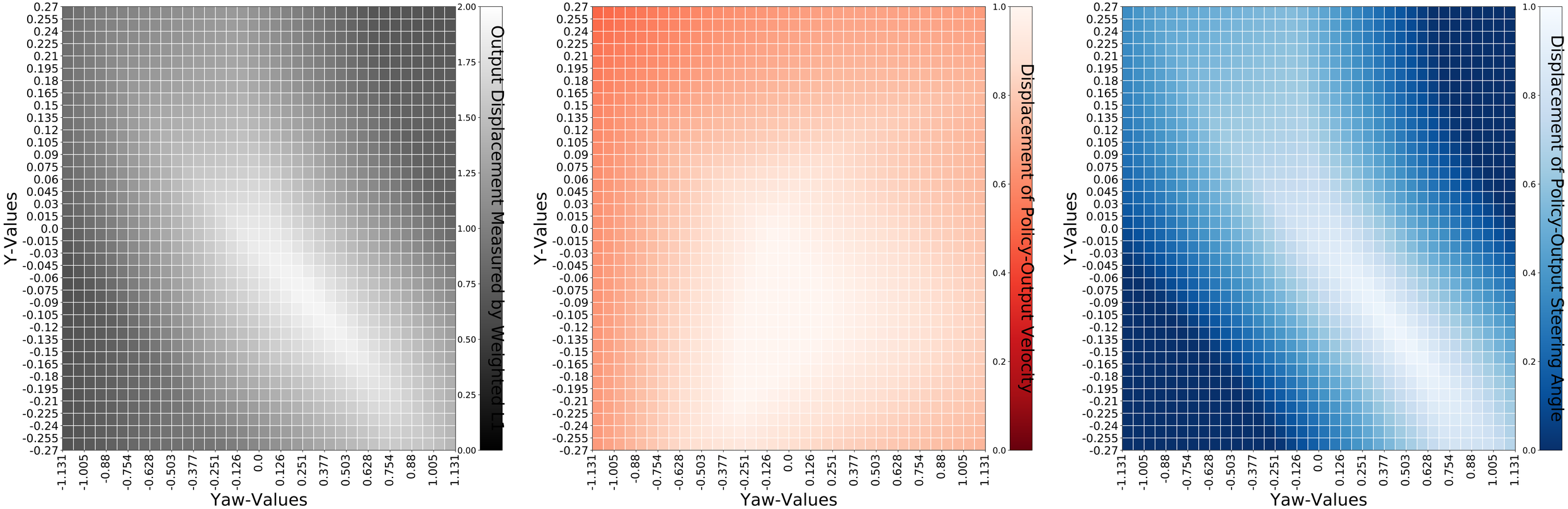


Line-Following Min. Magnitude Output Results
xDot:1.1, yDot:0, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 623ecb05-ce86-4ad7-b046-68f91e7f3977
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_07_06_00_833404

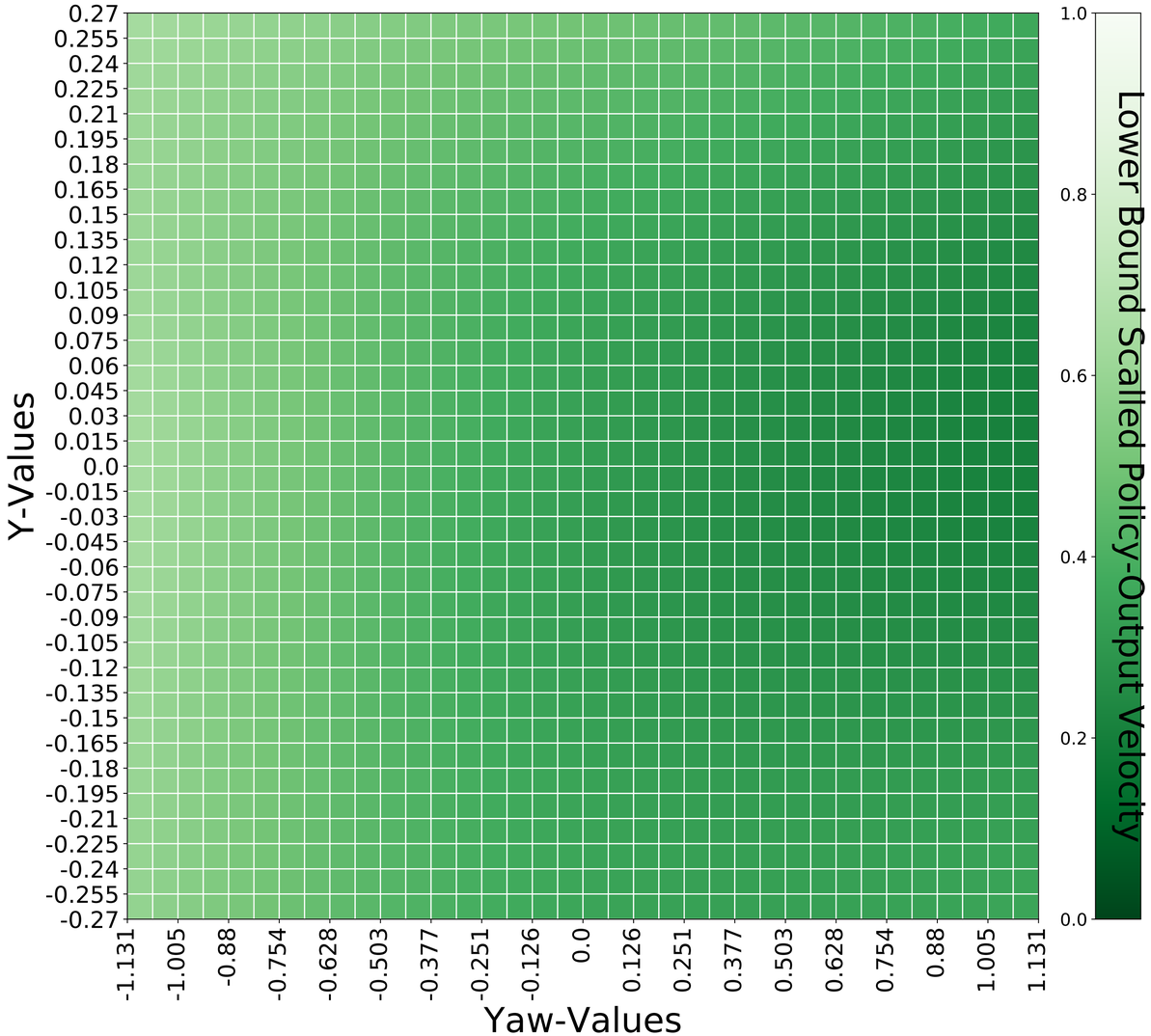


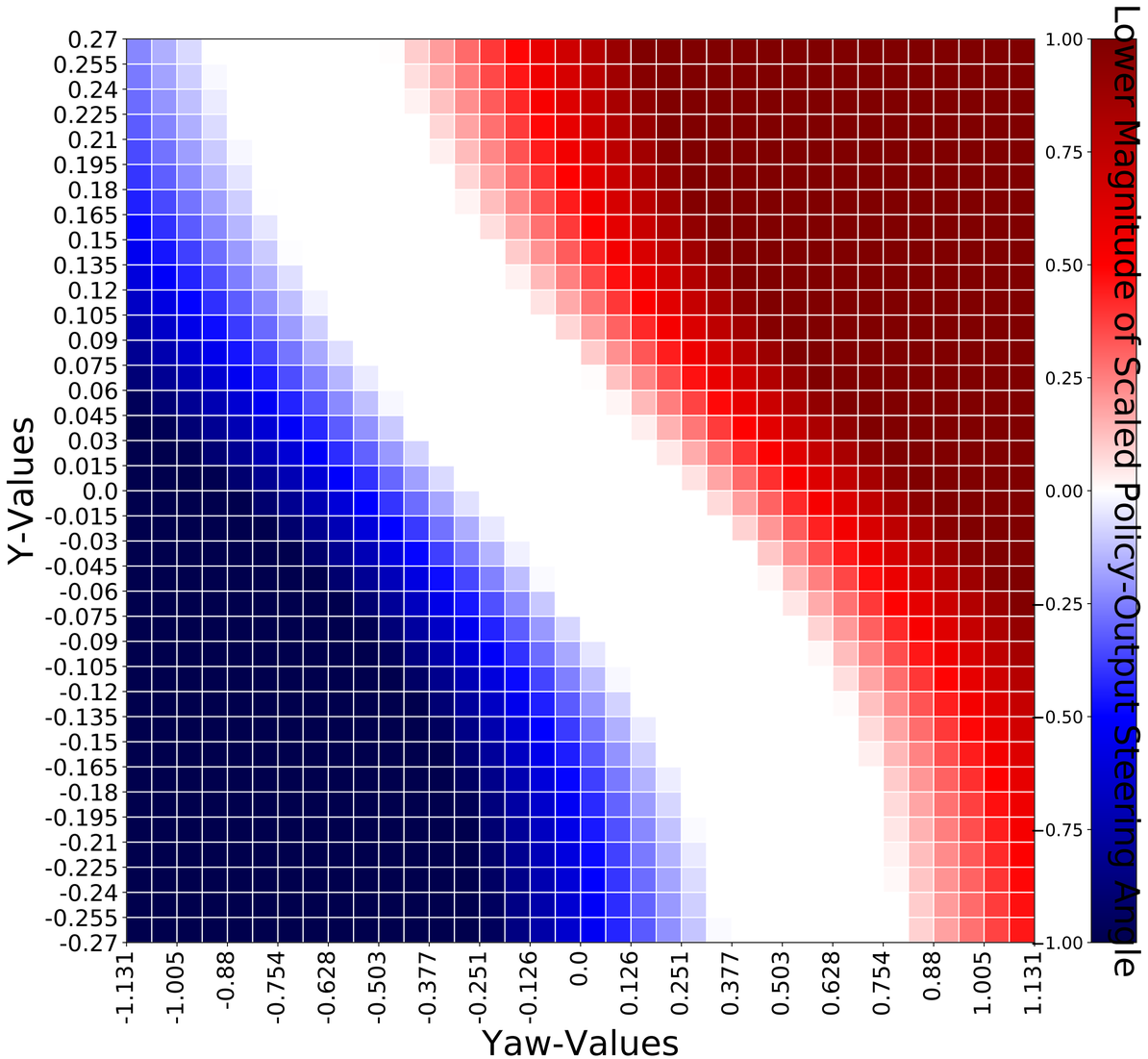


Line-Following Stability Results
xDot:1.1, yDot:0, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: e618da73-9b99-4a50-a50e-b495d25f5fc0
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_07_05_00_891222

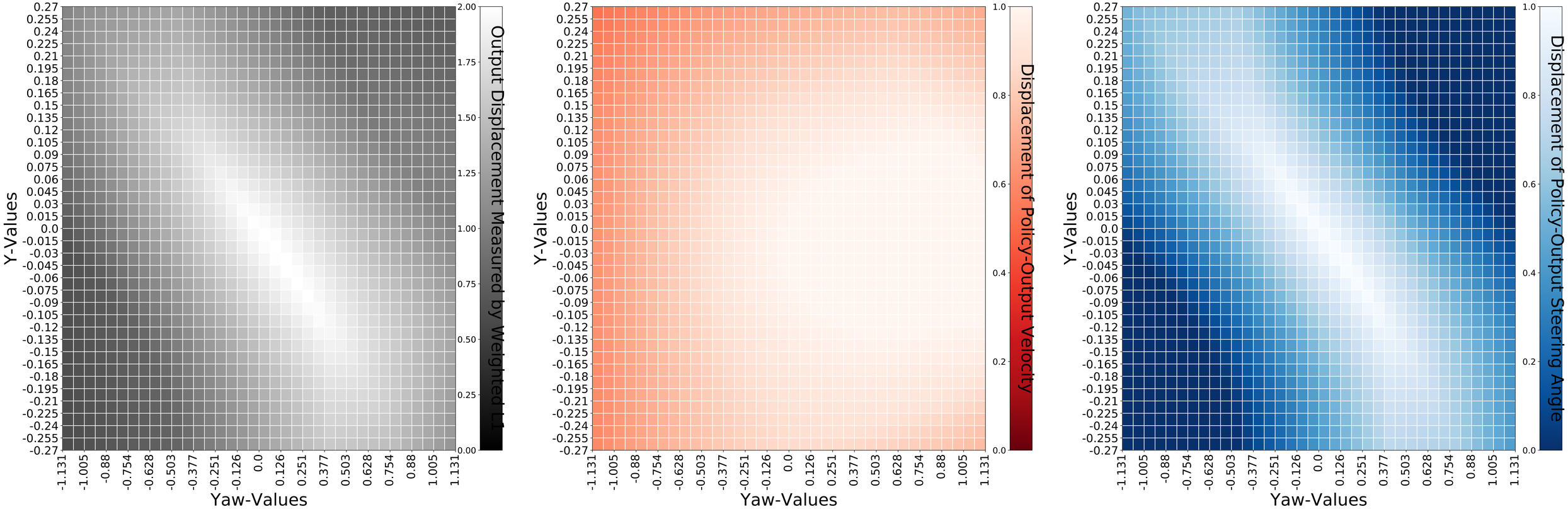


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0.046153, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 2f5999fe-3d89-4e3b-80a6-8143883fc839
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_18_58_33_780714

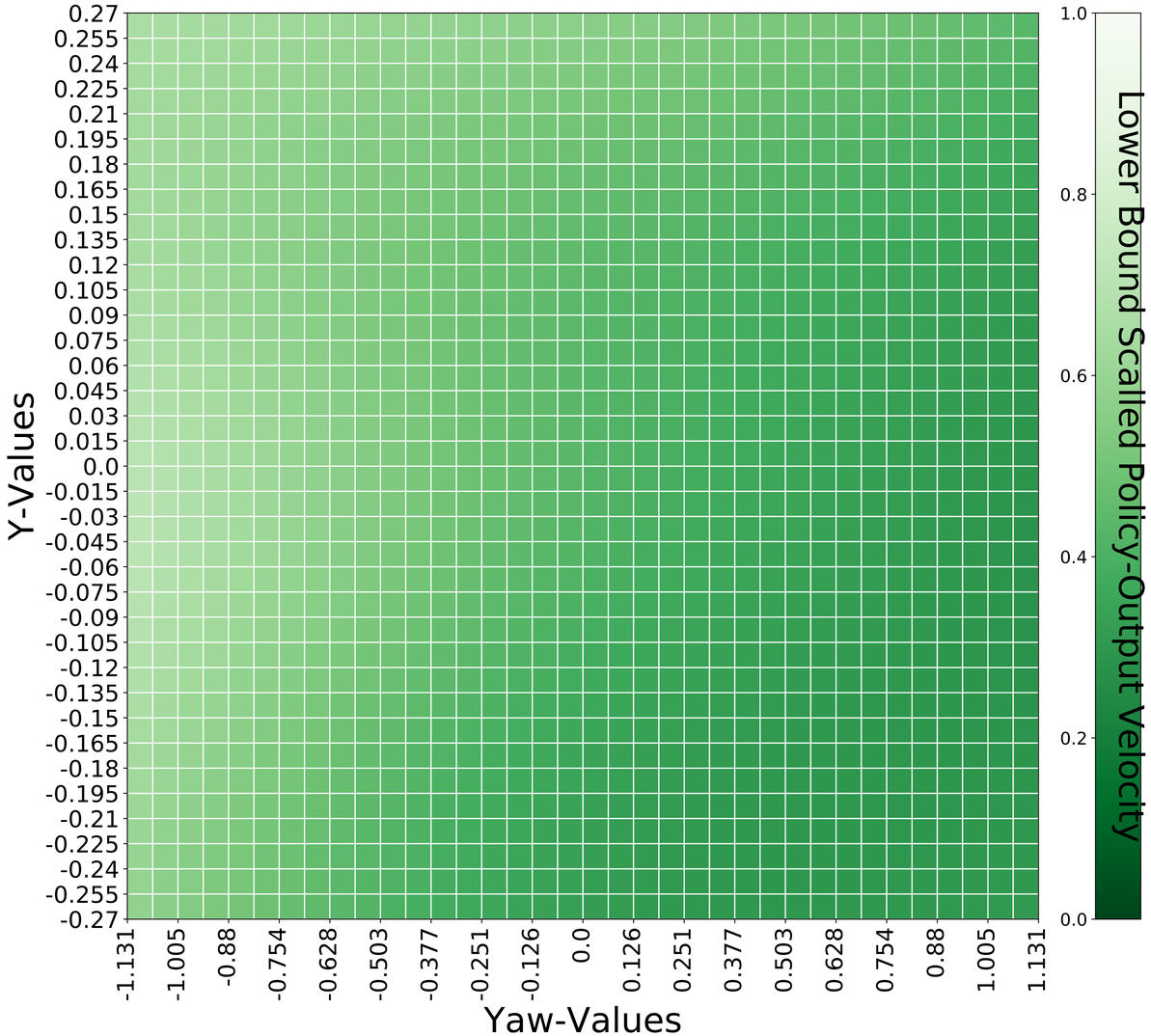


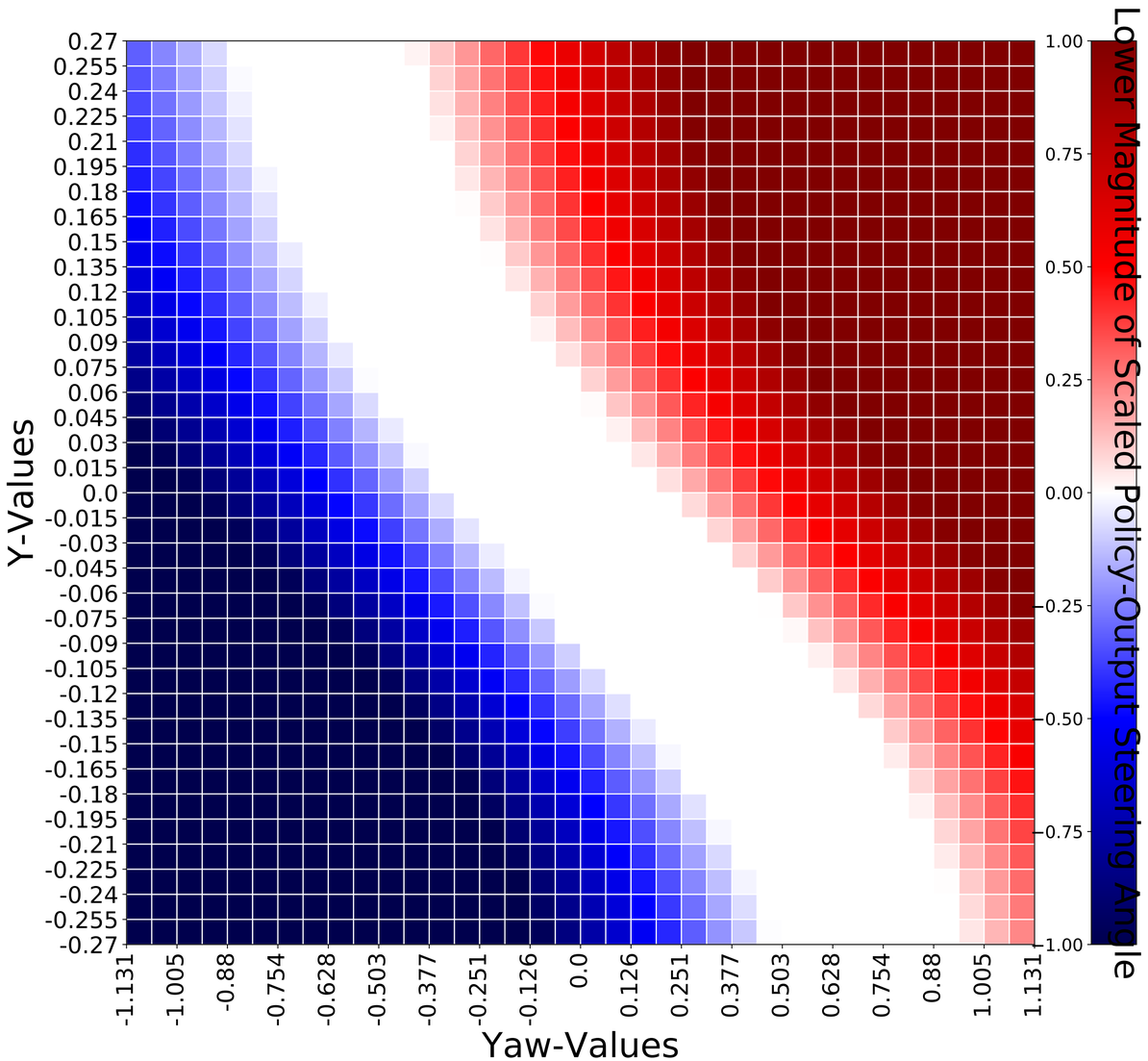


Line-Following Stability Results
xDot:0.0, yDot:0.046153, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: b2a478be-6515-4789-8147-f4a7e7477be4
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_18_57_35_824612

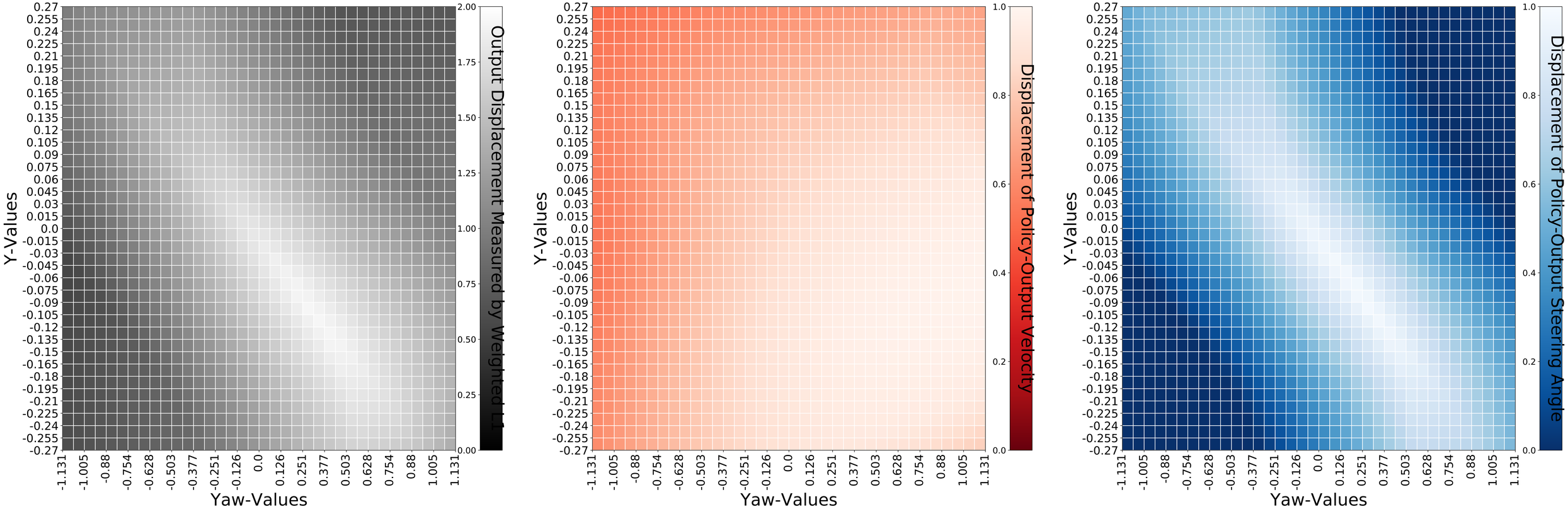


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0.046153, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 21048c07-1c73-4f66-8a47-a9e70dc86944
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_00_35_344783

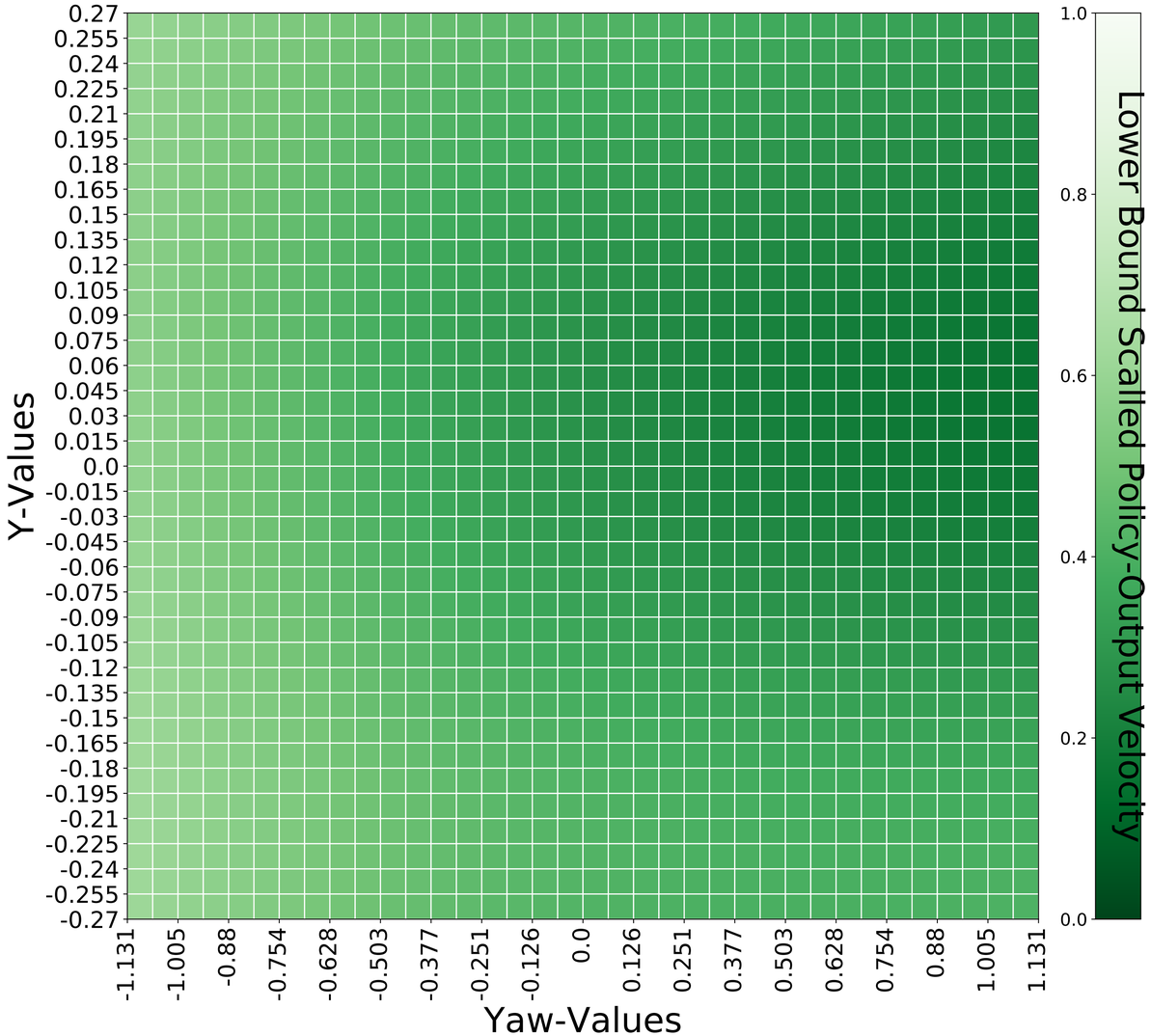


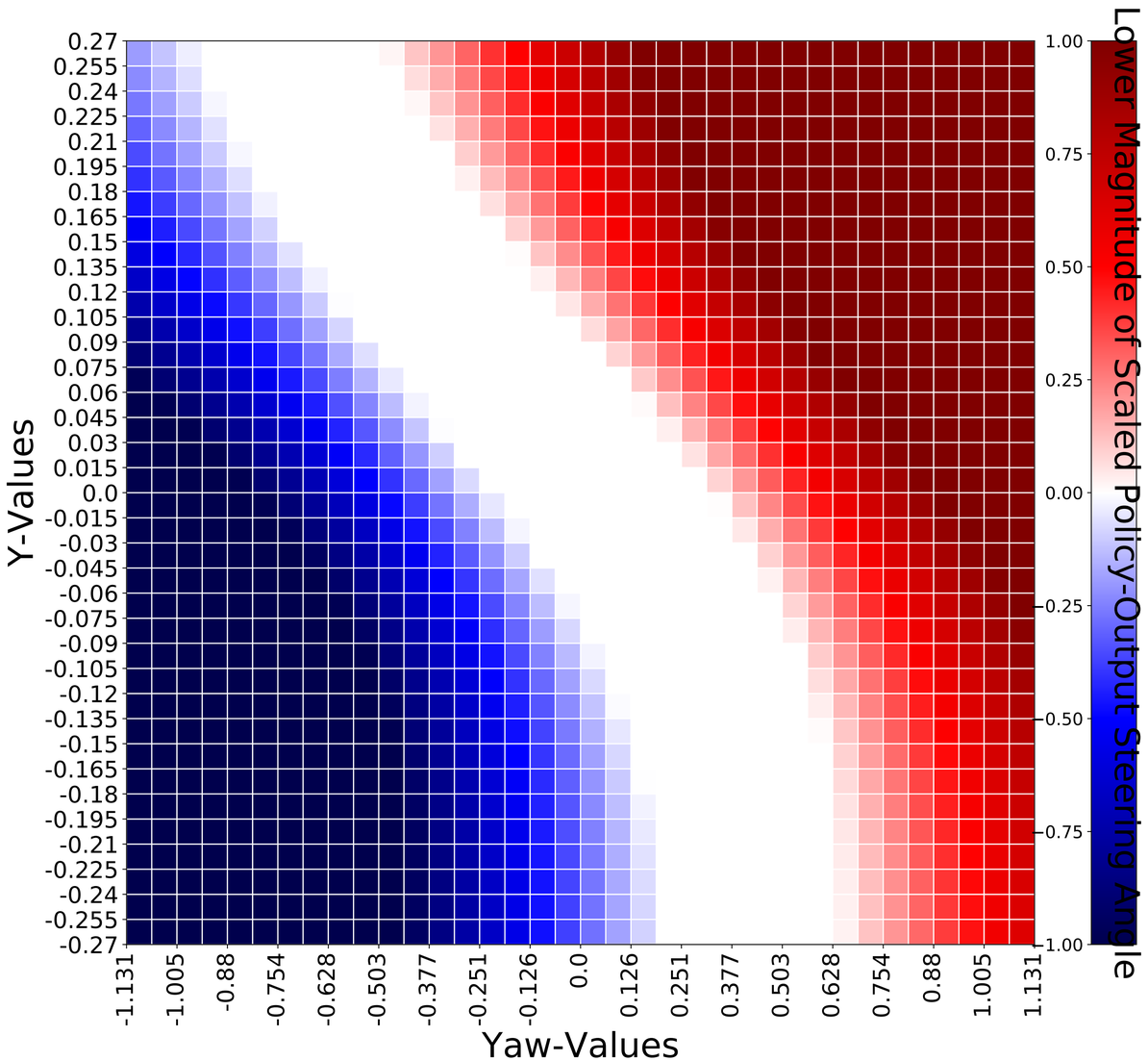


Line-Following Stability Results
xDot:0.0, yDot:0.046153, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 5497e08b-c553-4277-9e9f-3ded129b2b9d
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_18_59_38_060112

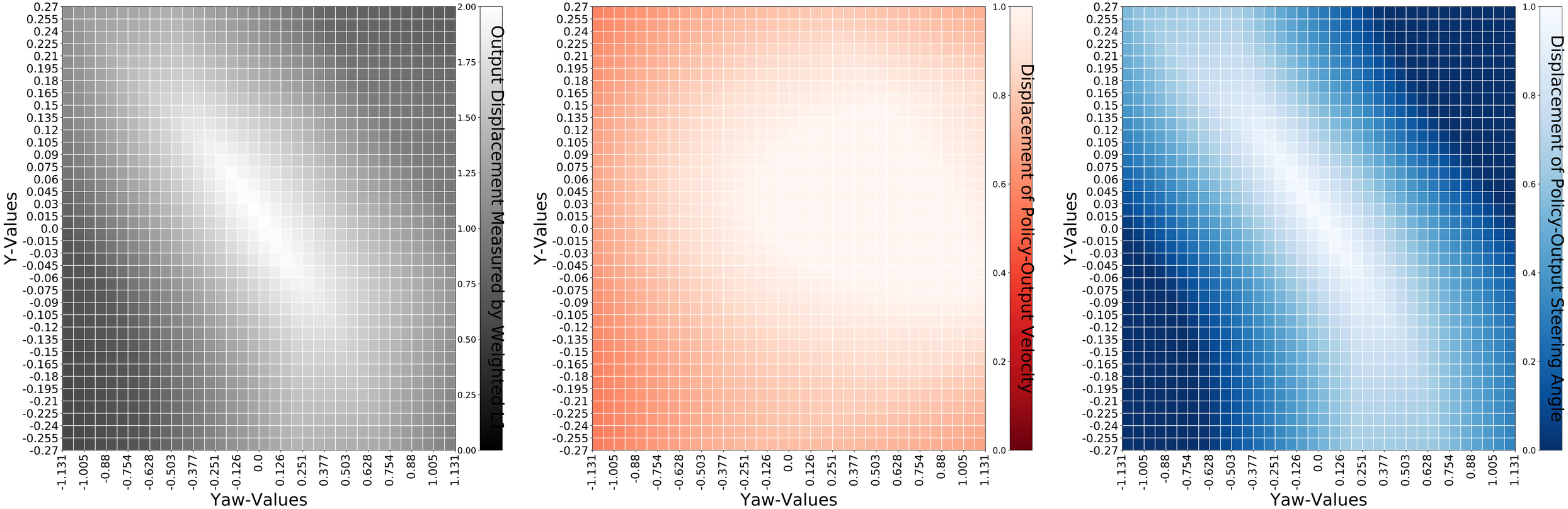


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0.046153, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 0a5274e6-c7bb-4a4d-a516-2043307f1310
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_02_35_851922

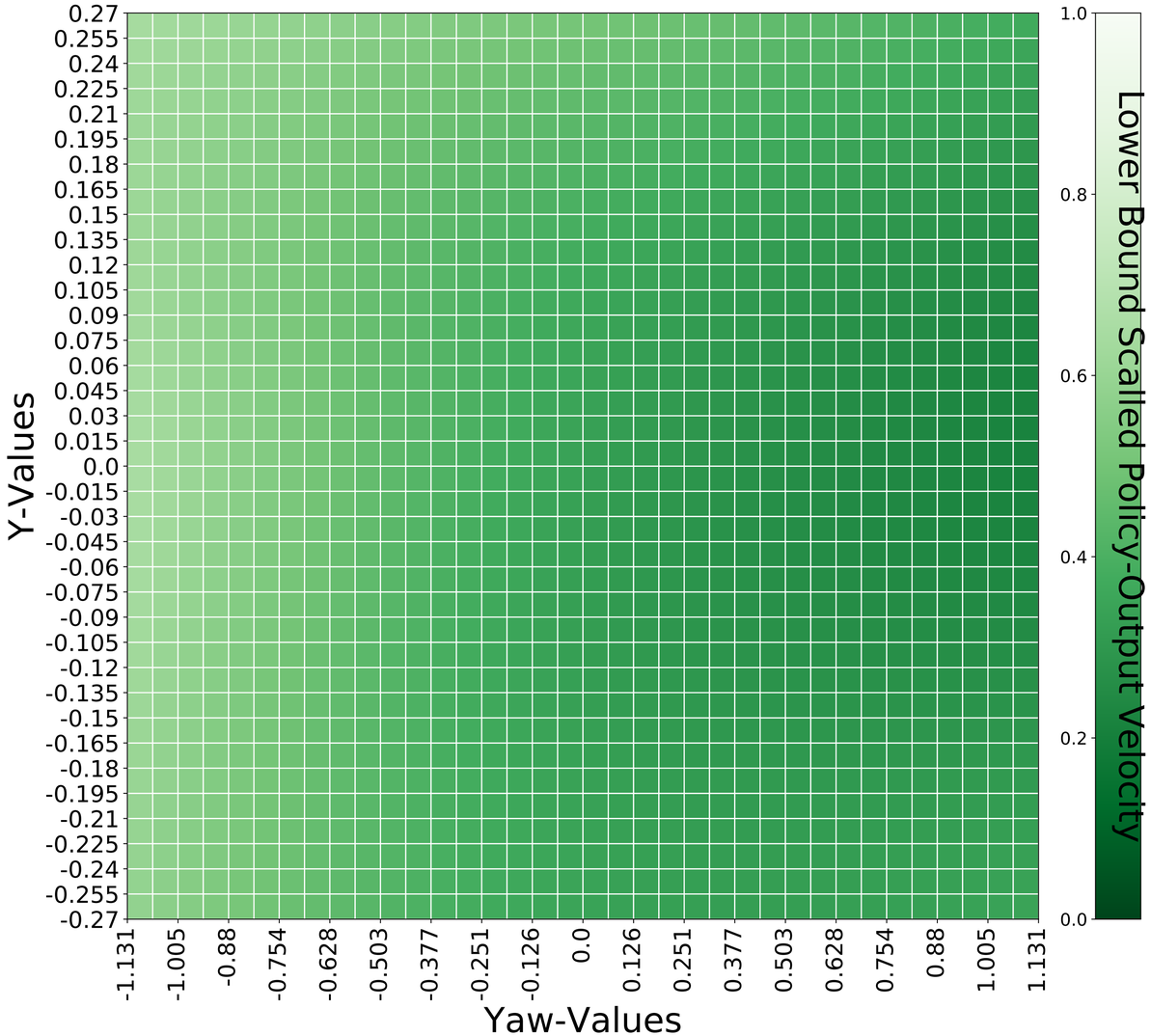


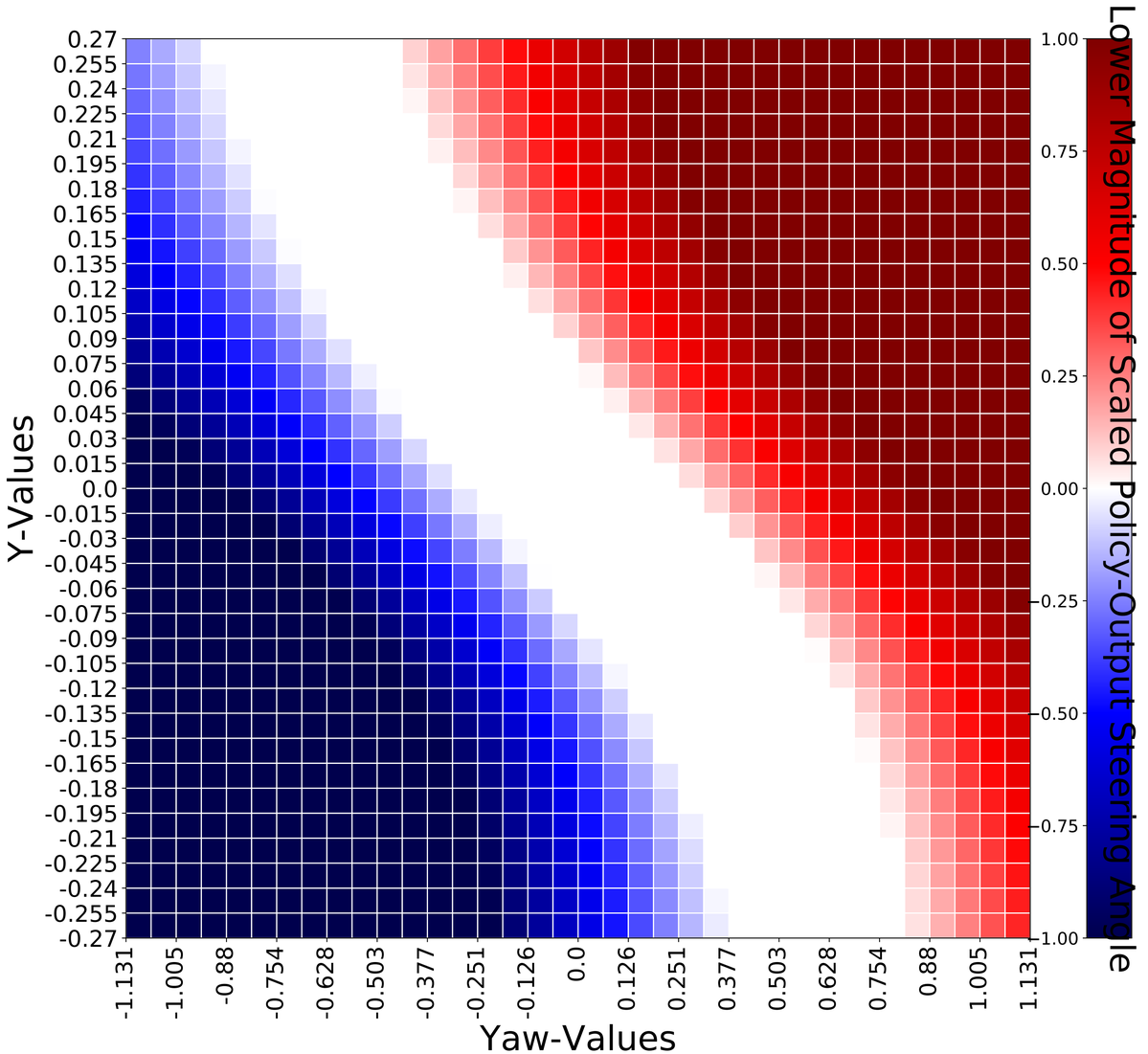


Line-Following Stability Results
xDot:0.0, yDot:0.046153, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: d998b249-9432-4899-95bc-092cf2a33a0b
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_01_30_475275

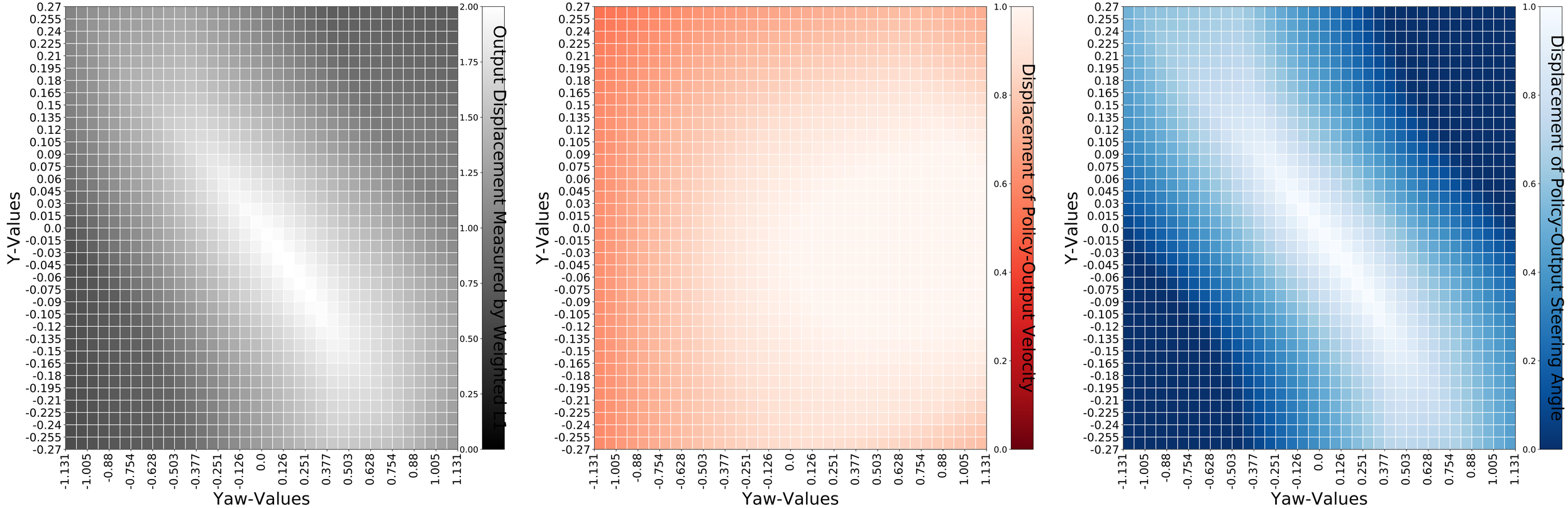


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0.046153, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: c50a3141-b11f-43e9-b950-1aceaa77d89c
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_14_33_285292

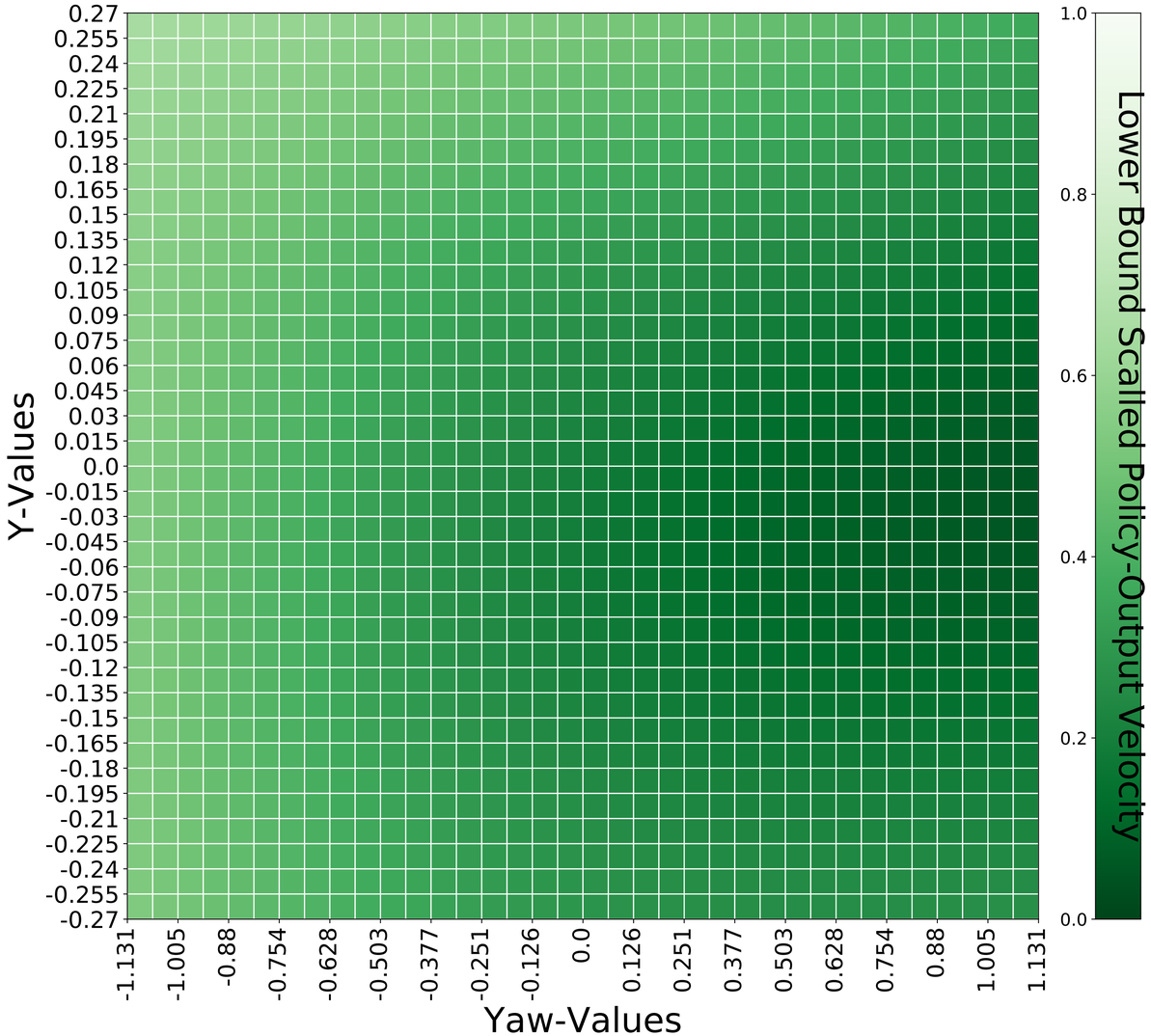


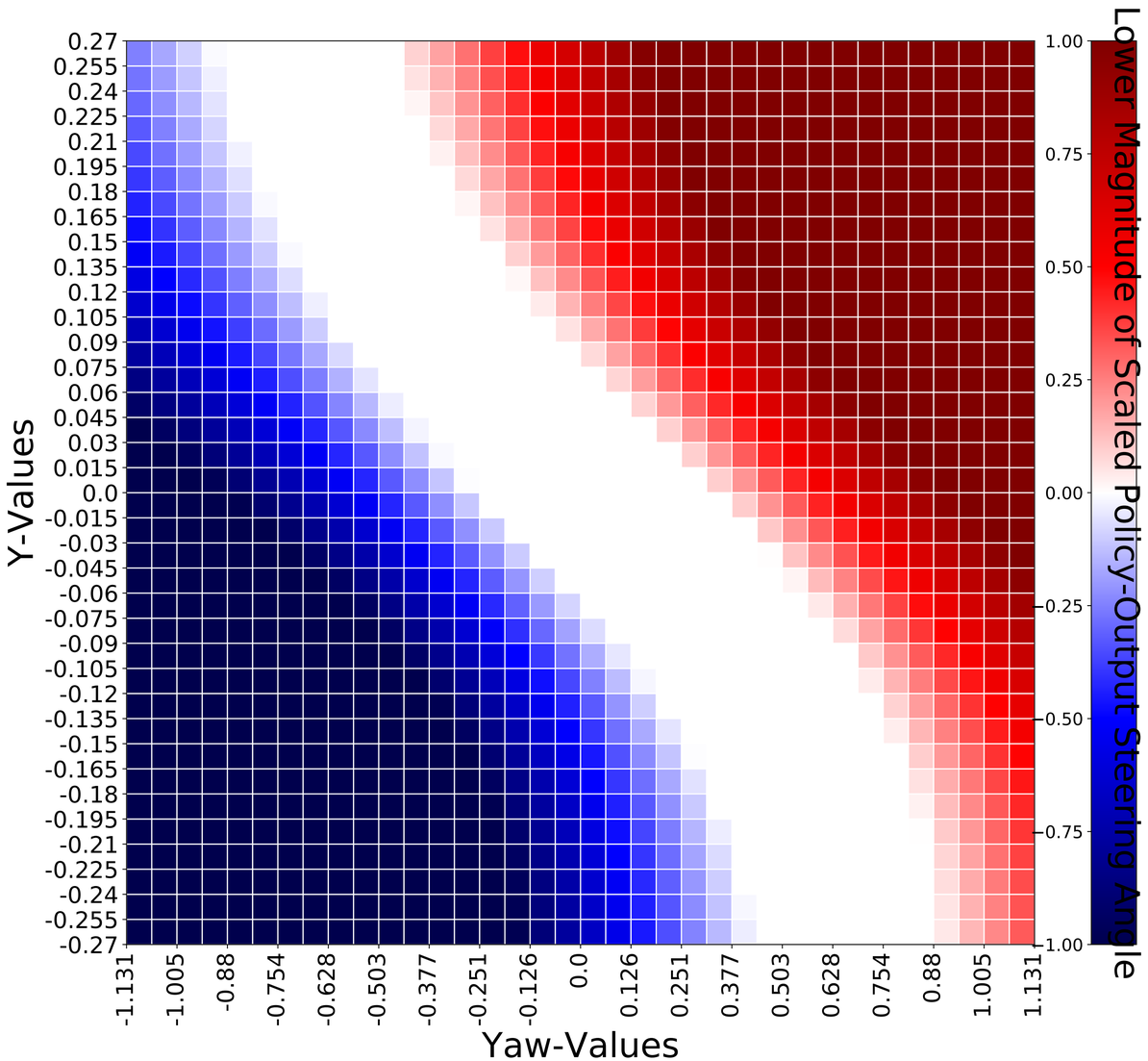


Line-Following Stability Results
xDot:0.0, yDot:0.046153, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 8eaa22a0-c348-456c-a9ad-c900b4ed99ee
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_13_29_190129

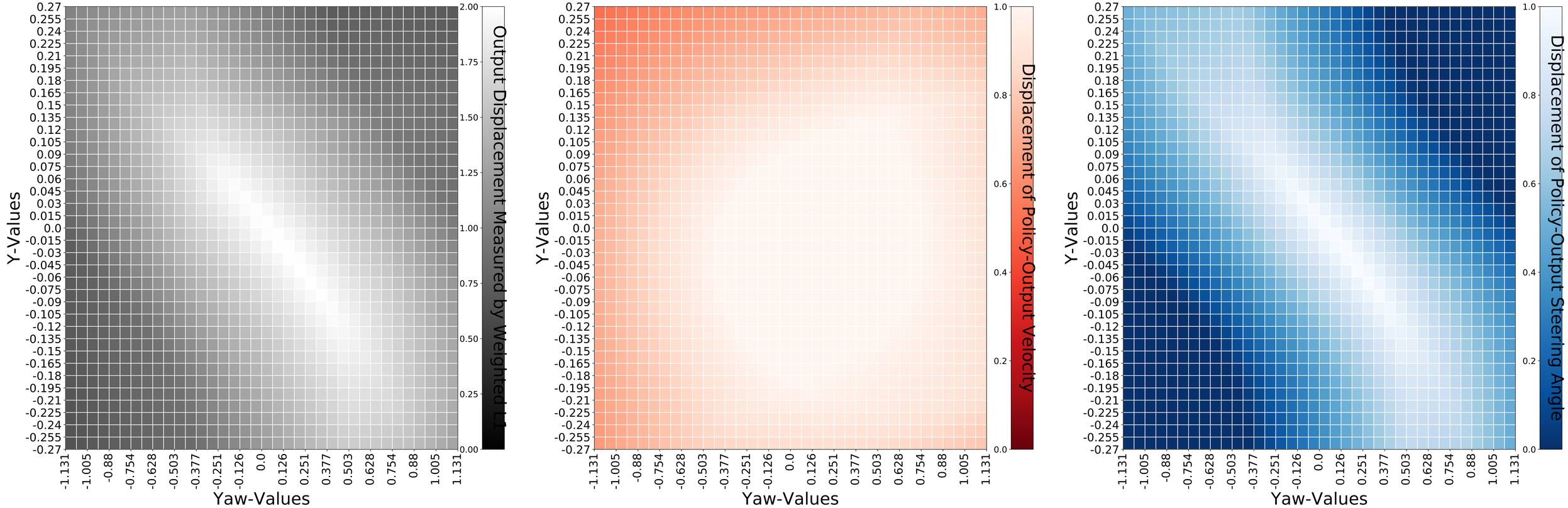


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:-0.32307, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 64c7b57e-1325-4a57-aa57-64ff9effa6b3
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_16_38_915414

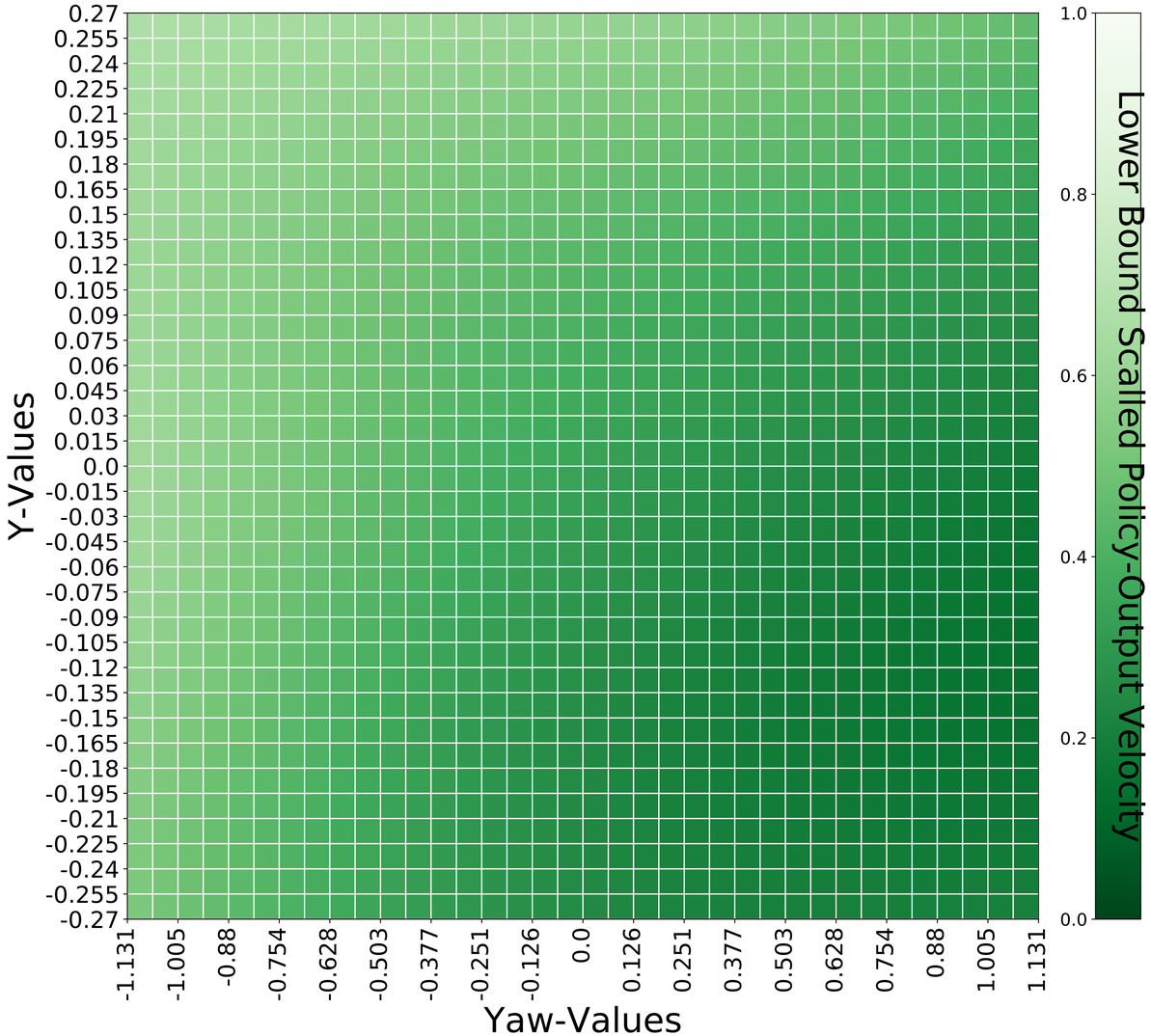


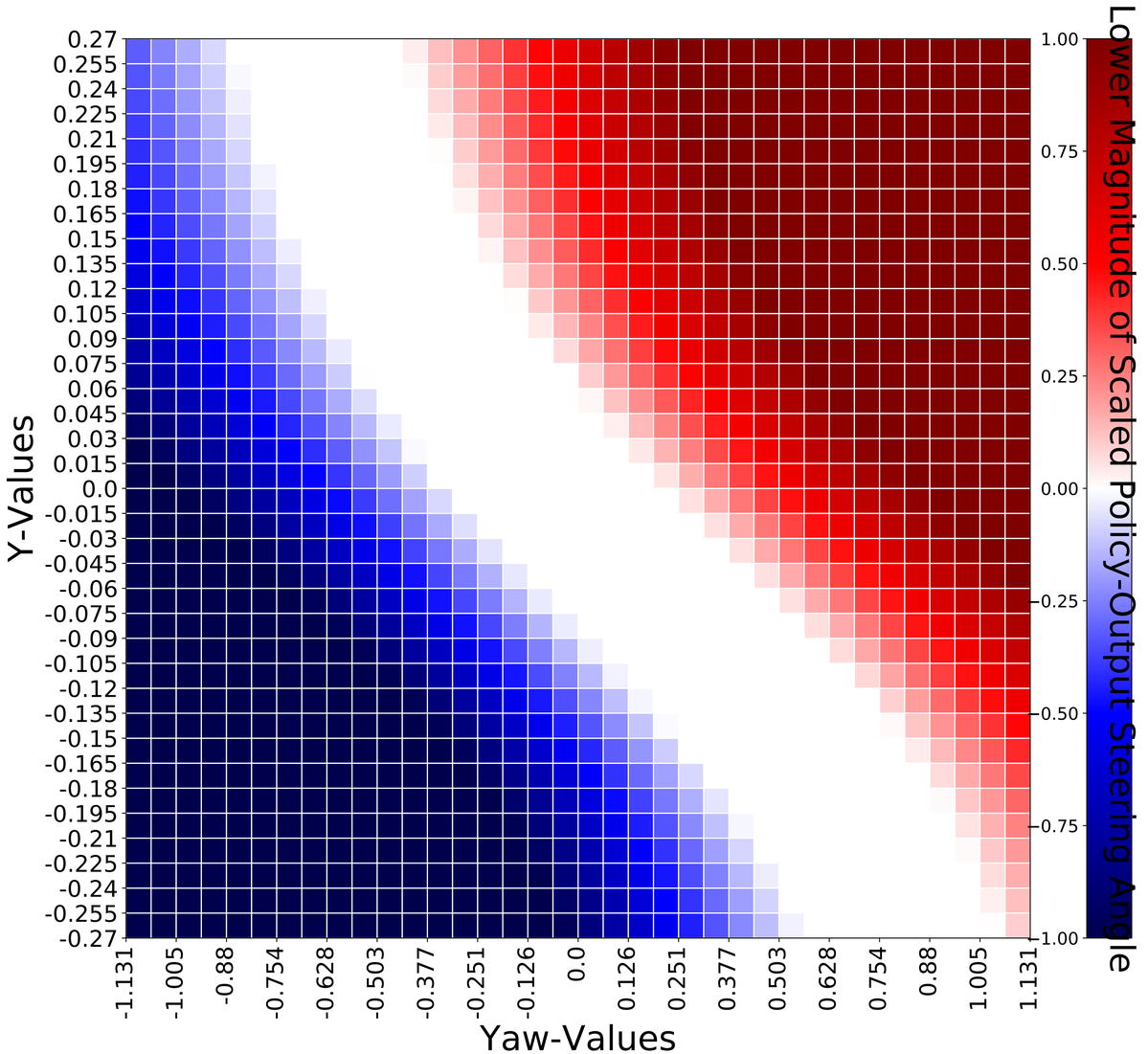


Line-Following Stability Results
xDot:0.0, yDot:-0.32307, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 5e41a9a8-a329-4563-b66f-21f6b1e28110
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_15_37_419116

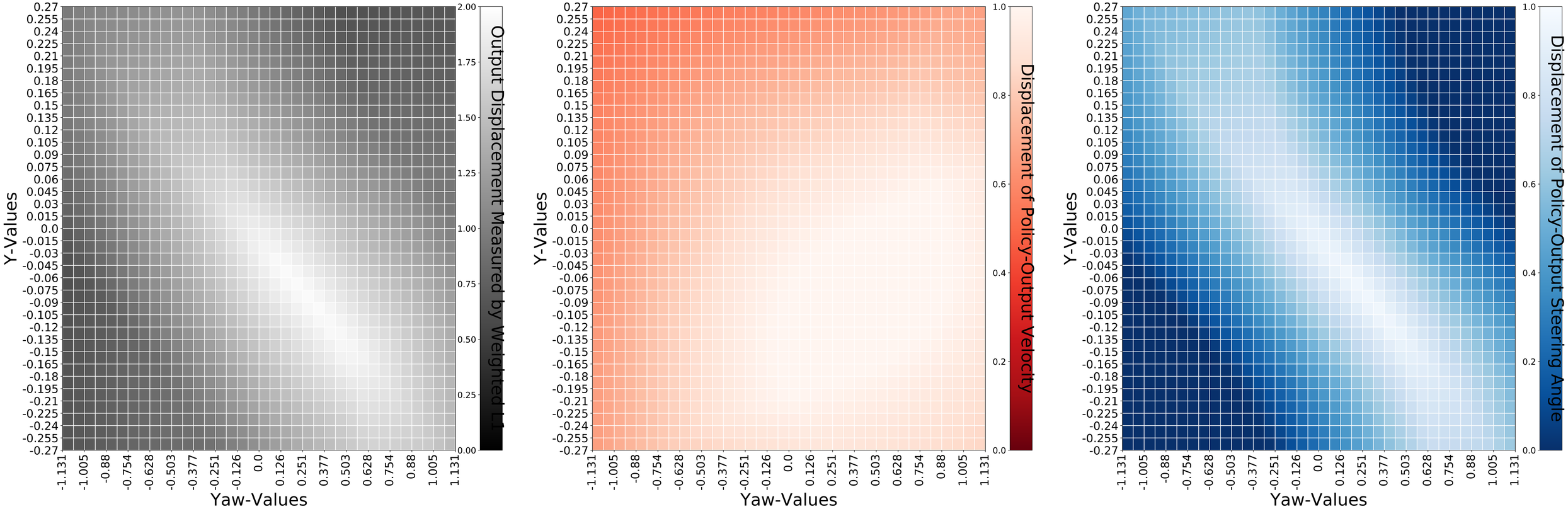


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:-0.32307, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 27c5db1c-e90f-42dc-bd47-af39f1b8847d
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_18_39_274701

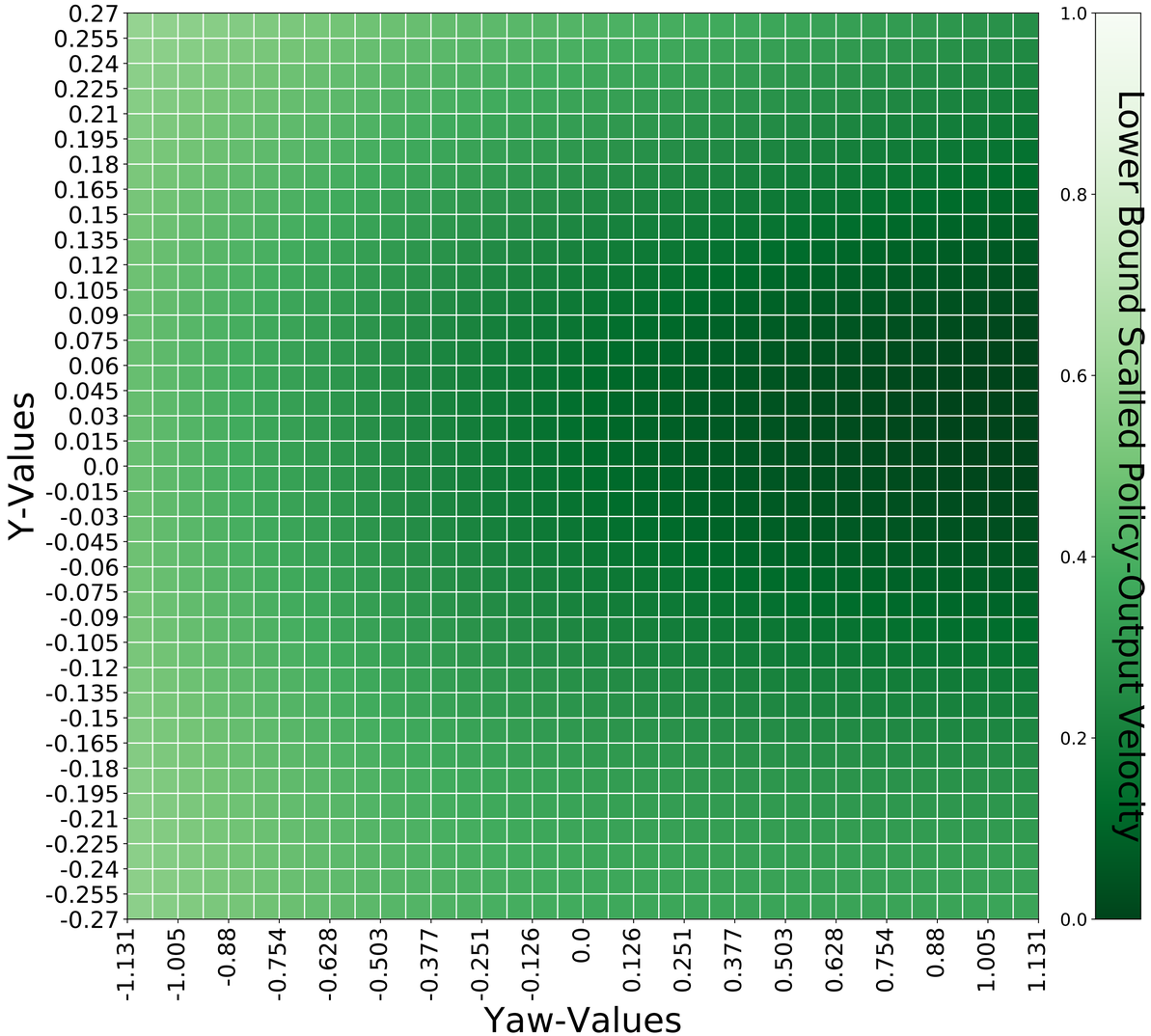


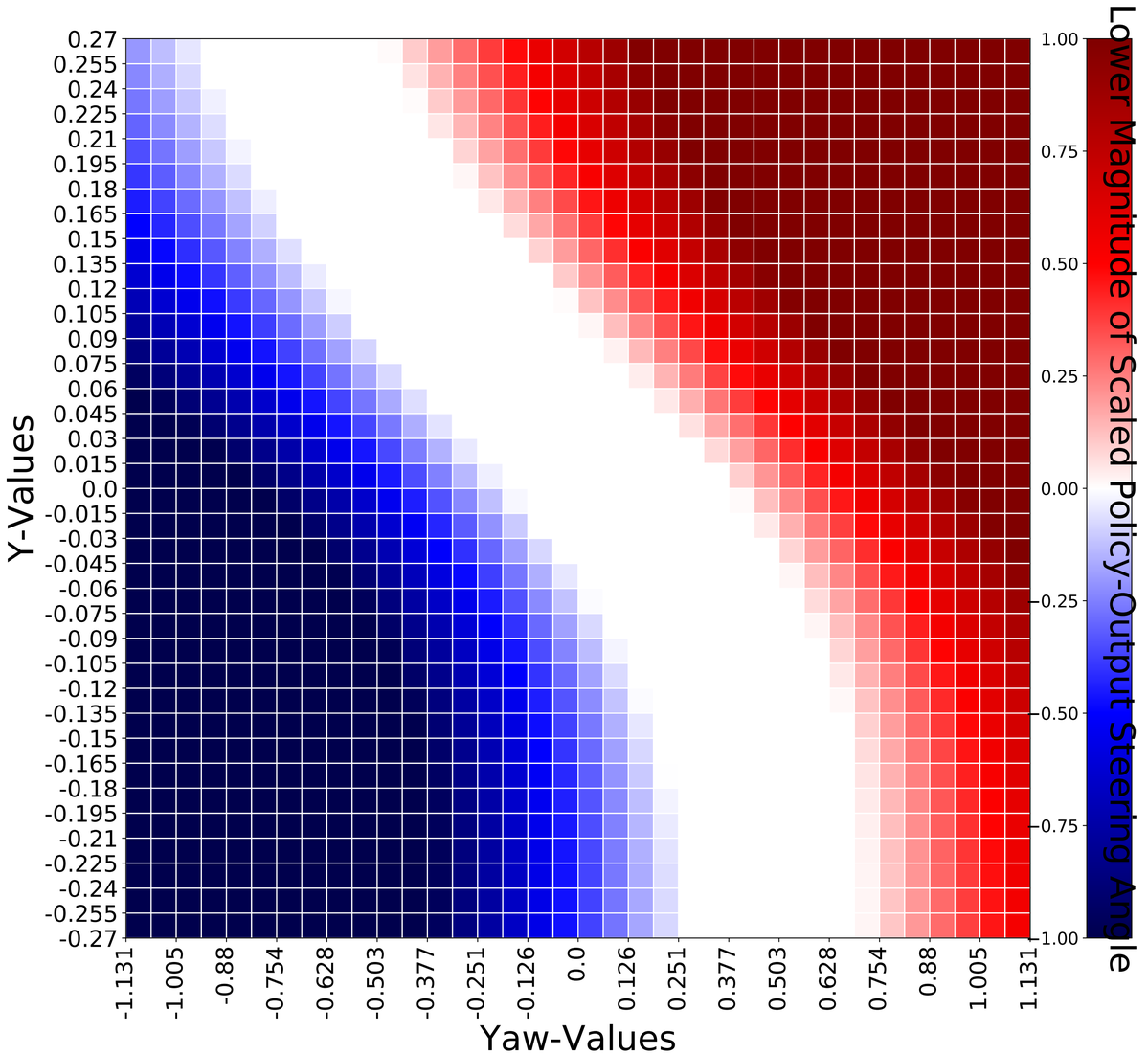


Line-Following Stability Results
xDot:0.0, yDot:-0.32307, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 894e1137-7f01-4912-b153-e63784a51f2d
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_17_35_601946

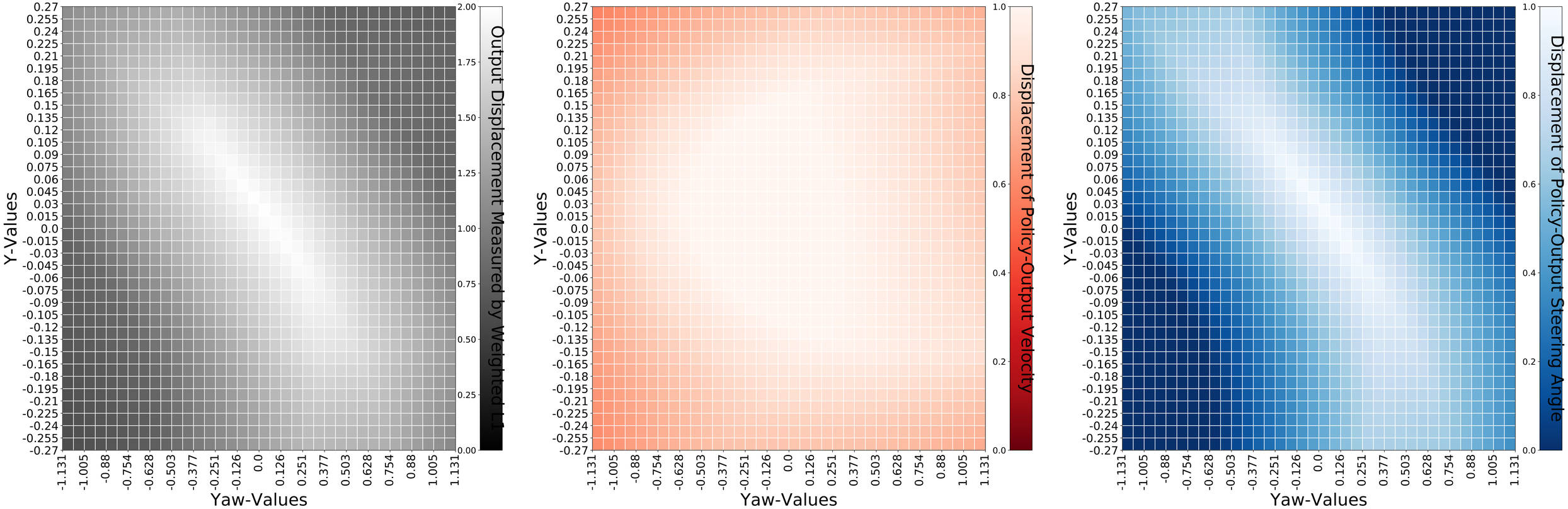


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:-0.32307, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: fd4a745e-a767-4e85-8ea9-e12985e9777e
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_20_37_693234

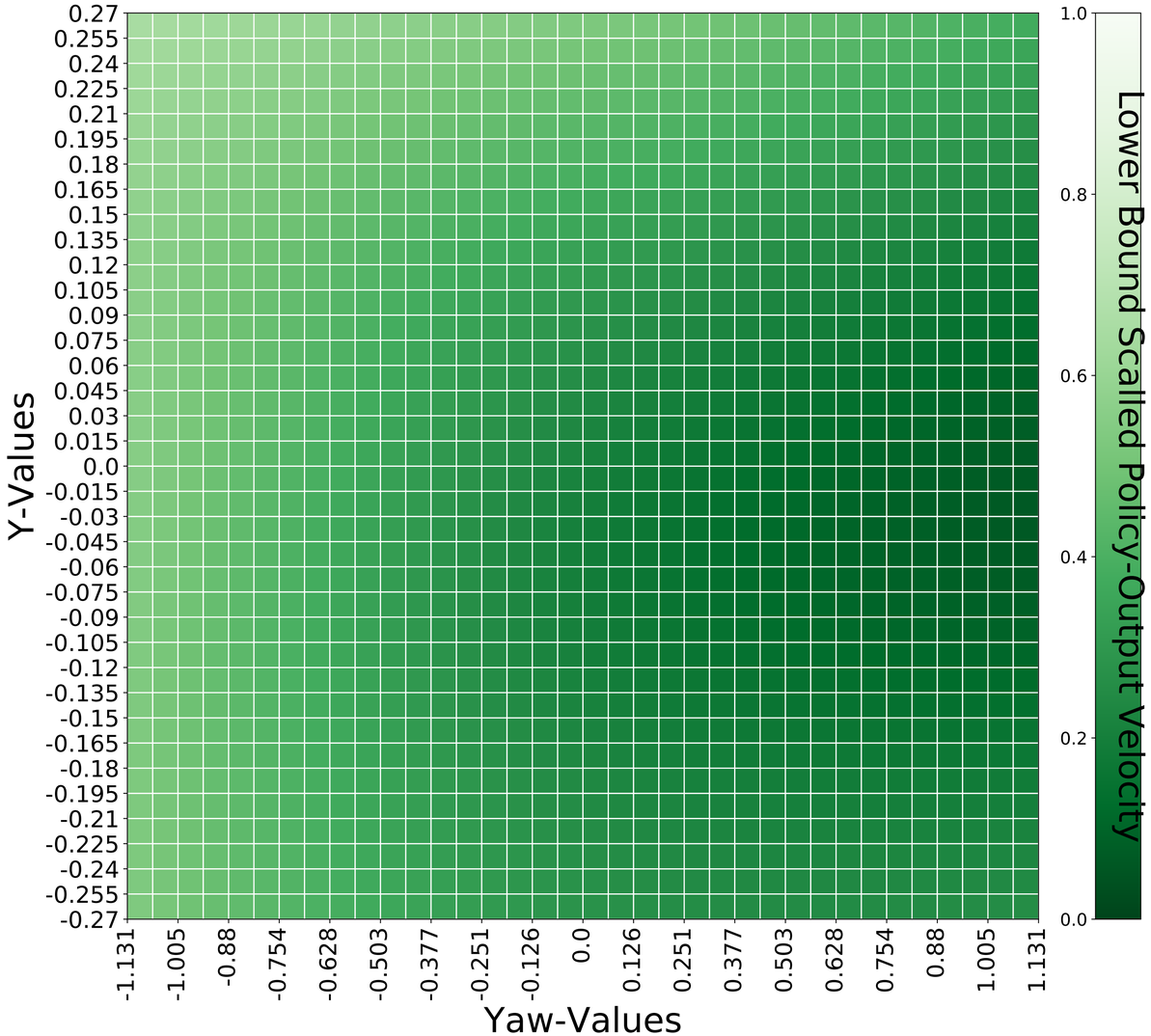


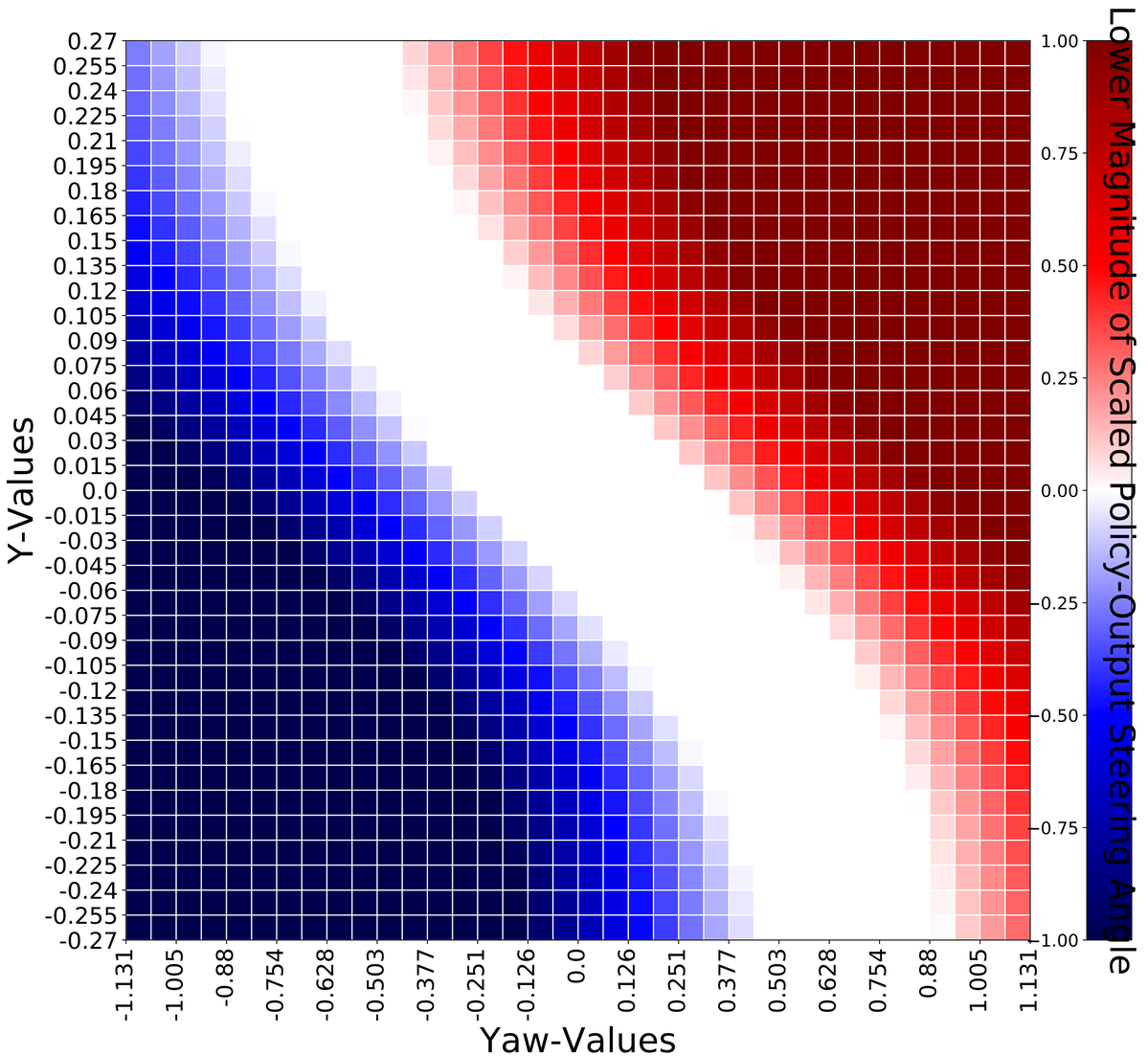


Line-Following Stability Results
xDot:0.0, yDot:-0.32307, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 943452a4-5b8e-43ce-8ca3-76981a246851
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_19_38_948712

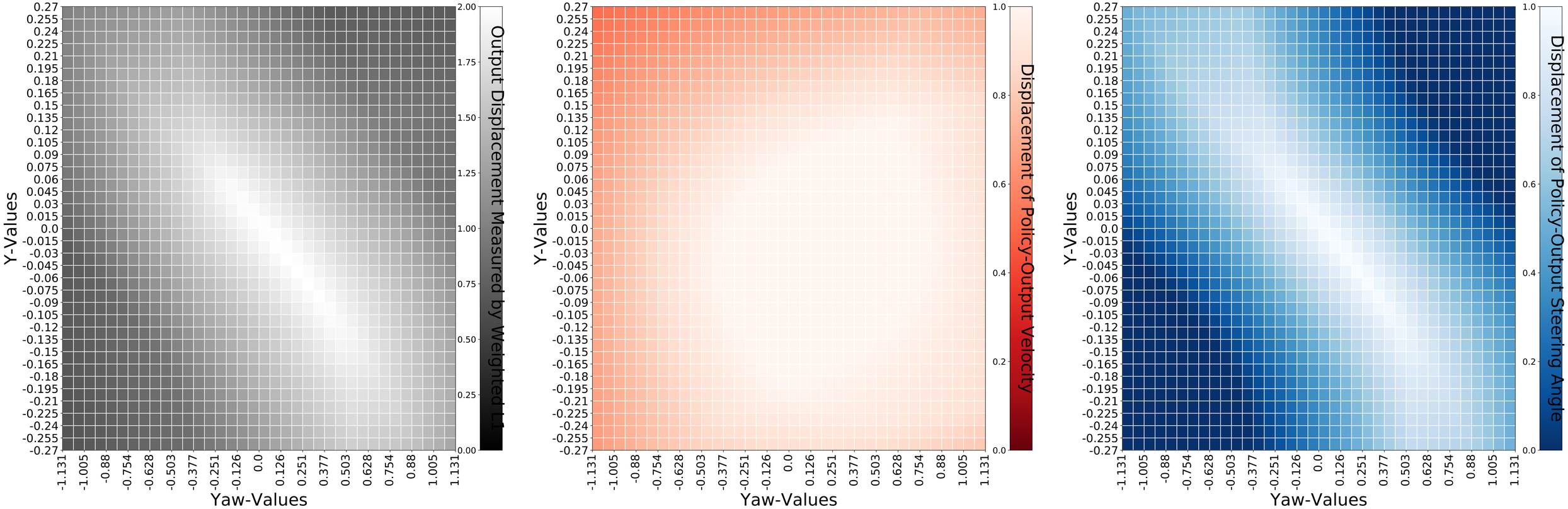


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:-0.32307, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 1641b406-98bd-4cda-aa1c-3558bd8a5686
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_32_35_040717

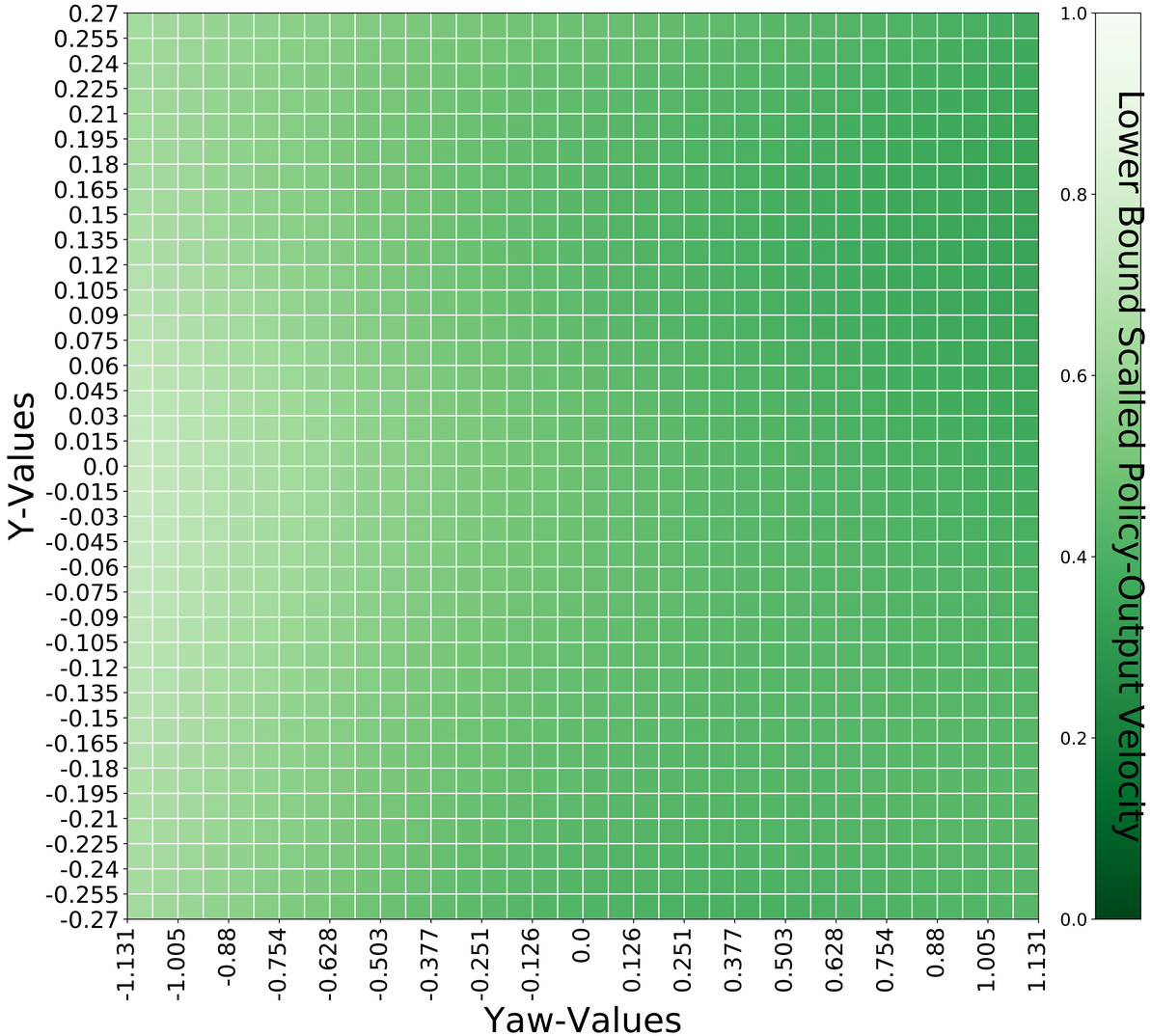


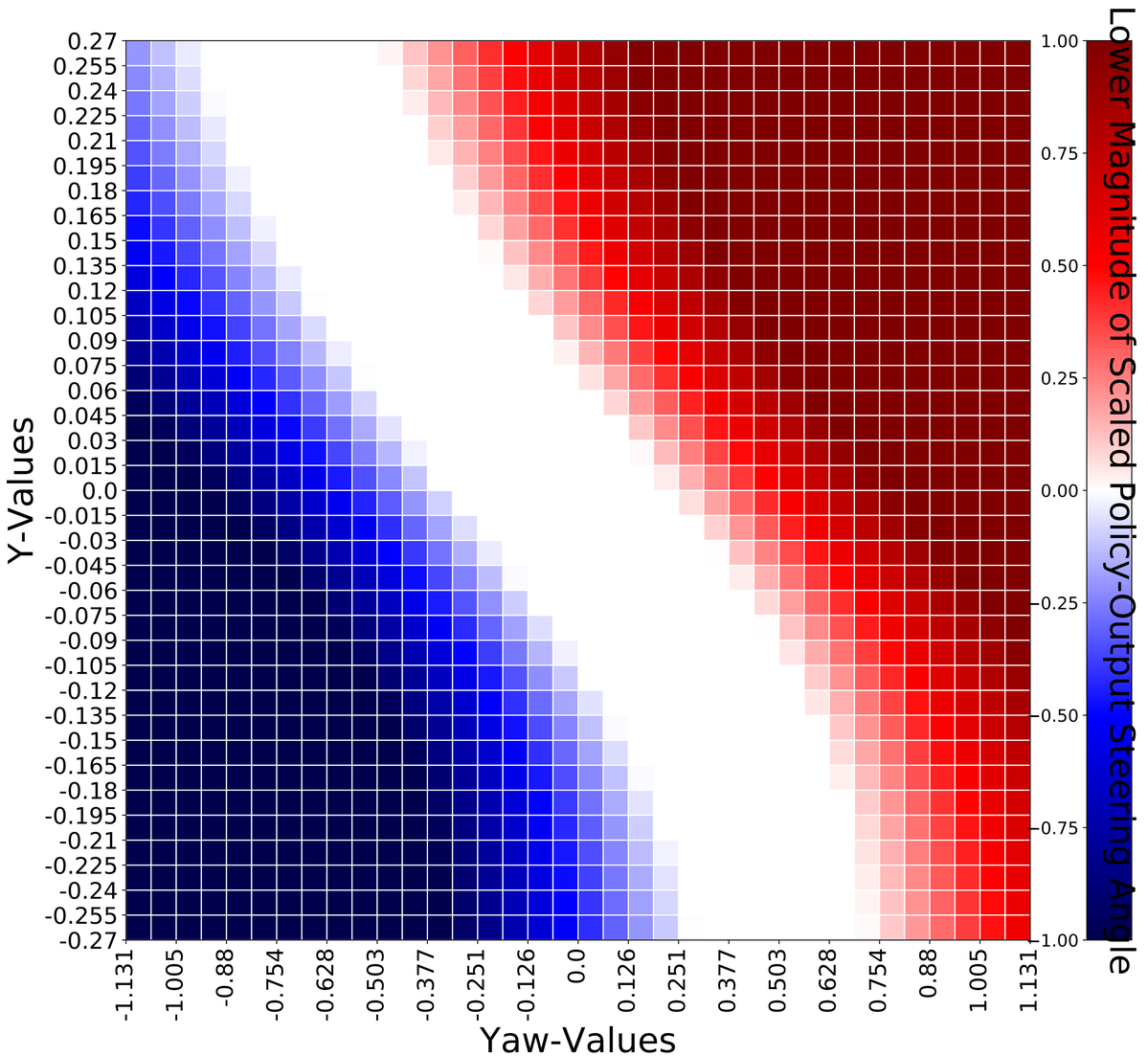


Line-Following Stability Results
xDot:0.0, yDot:-0.32307, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 19b1a662-c576-41ec-87db-3b9d2be084d3
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_31_32_326729

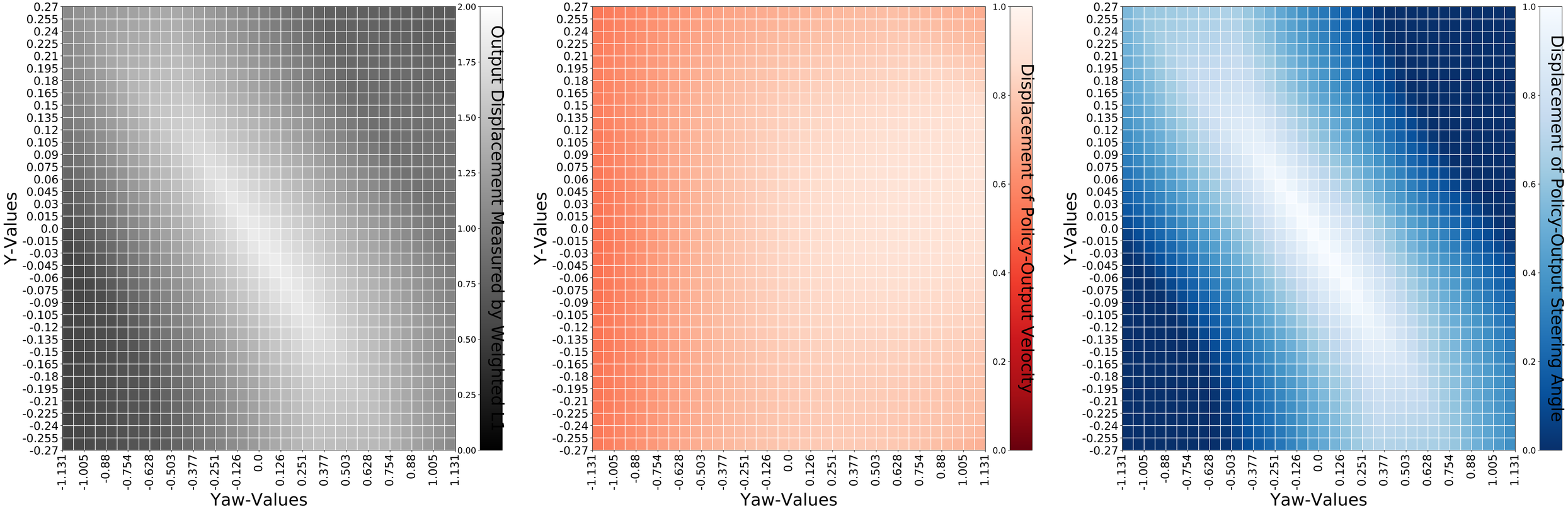


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0.415384, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 8ae429ff-0fb5-4340-ad5c-cb8e8b9487ac
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_34_36_977341

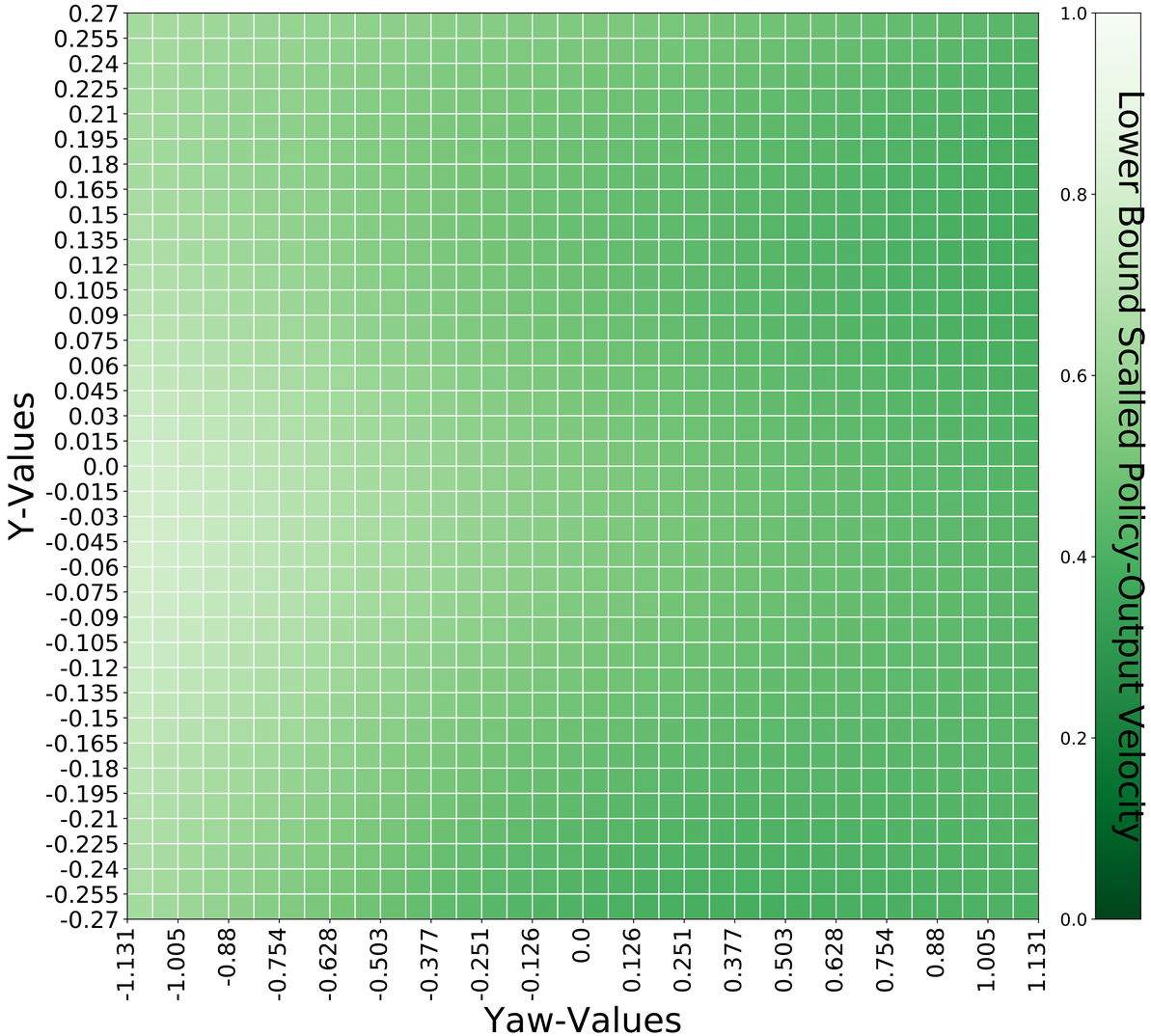


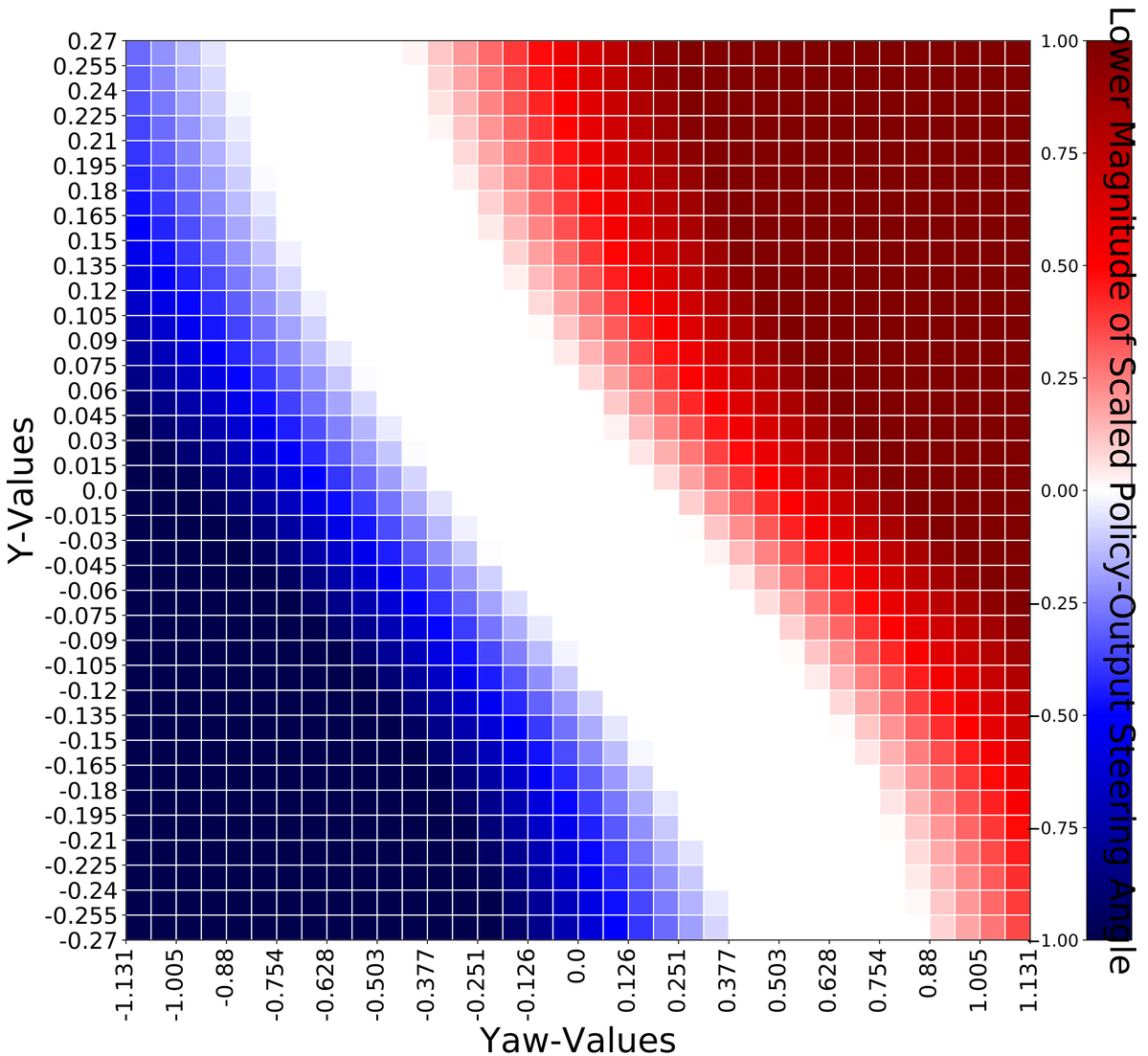


Line-Following Stability Results
xDot:0.0, yDot:0.415384, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 69a58351-9f9c-48f5-9842-9784cd6b343c
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_33_32_037073

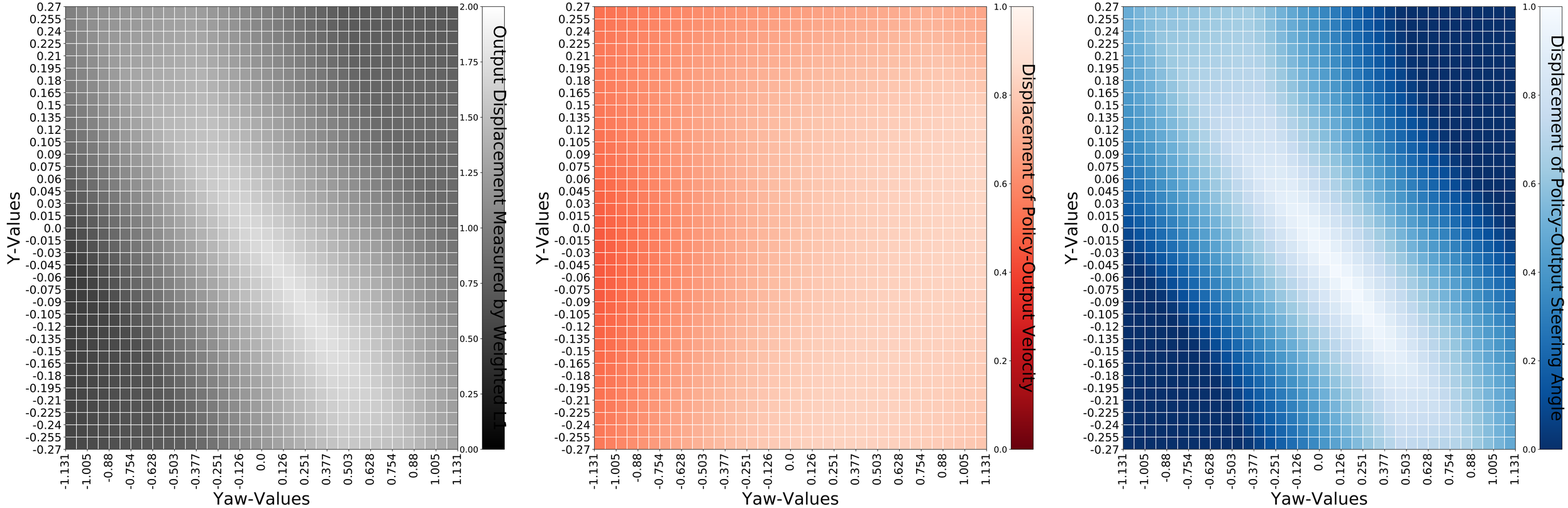


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0.415384, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: da05c1c0-9444-4fd0-a118-fd02bef3c614
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_36_39_855200

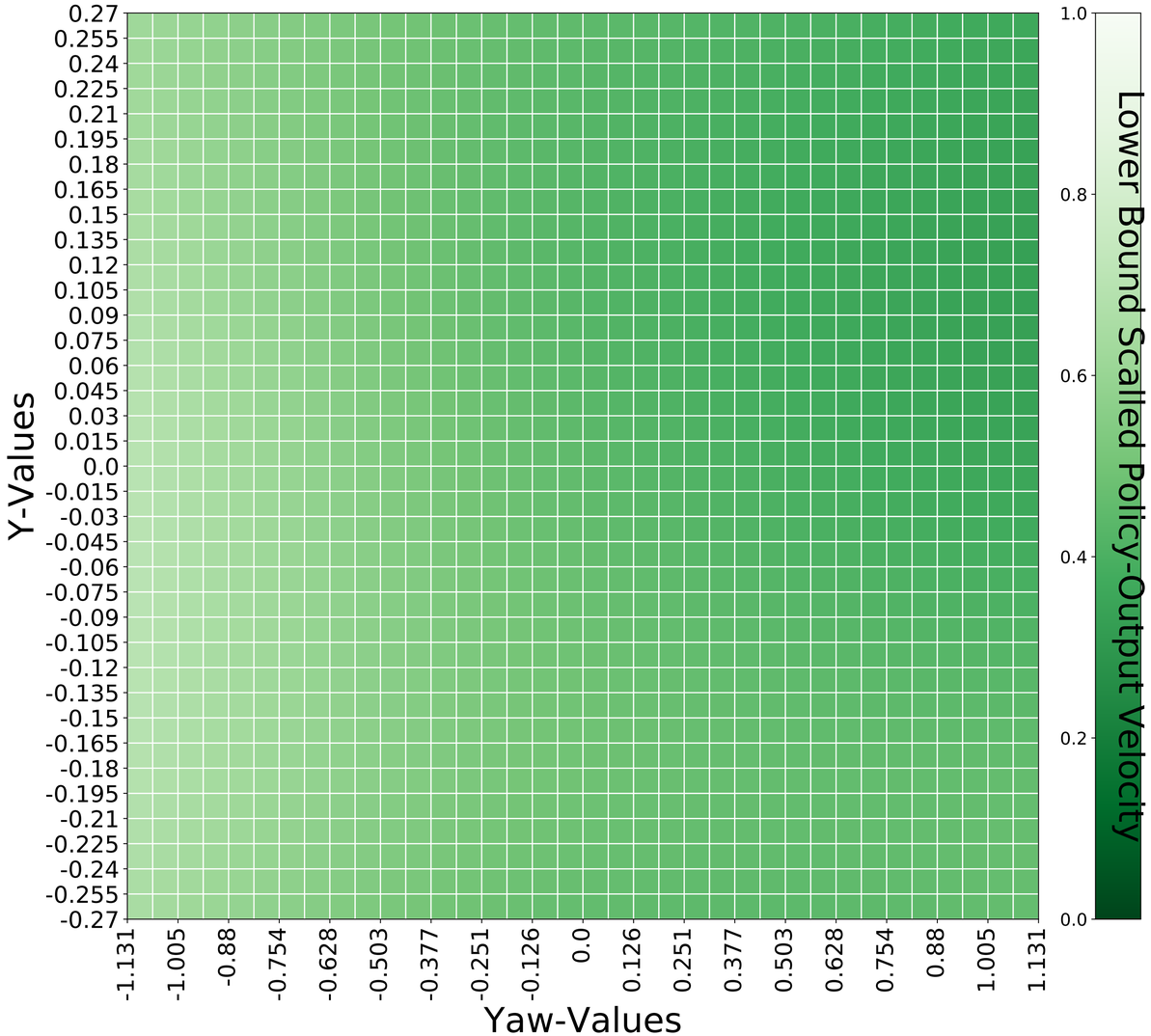


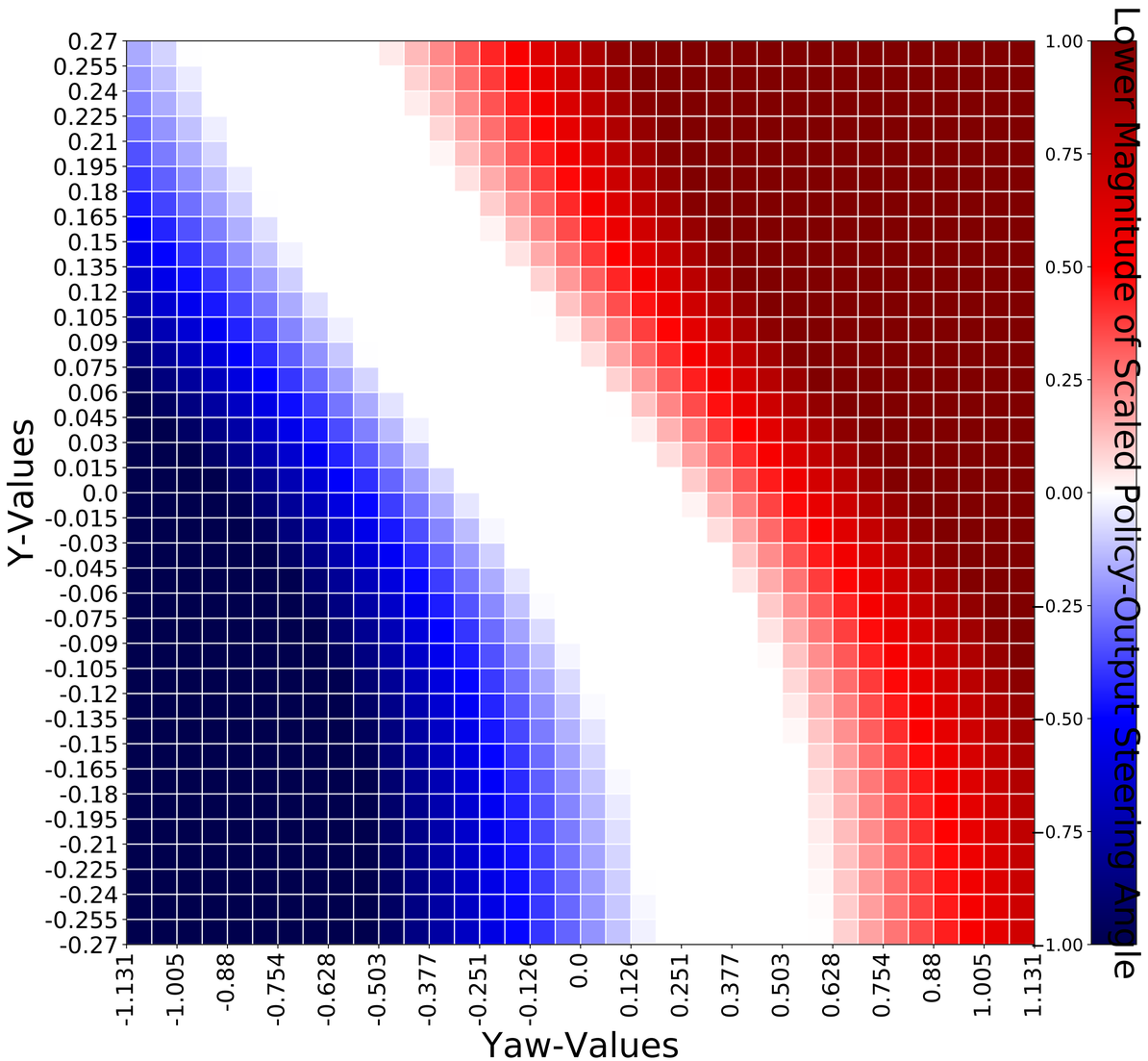


Line-Following Stability Results
xDot:0.0, yDot:0.415384, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: c1bd0ce2-b6b0-4d32-a4f5-e22a9c122de5
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_35_39_352312

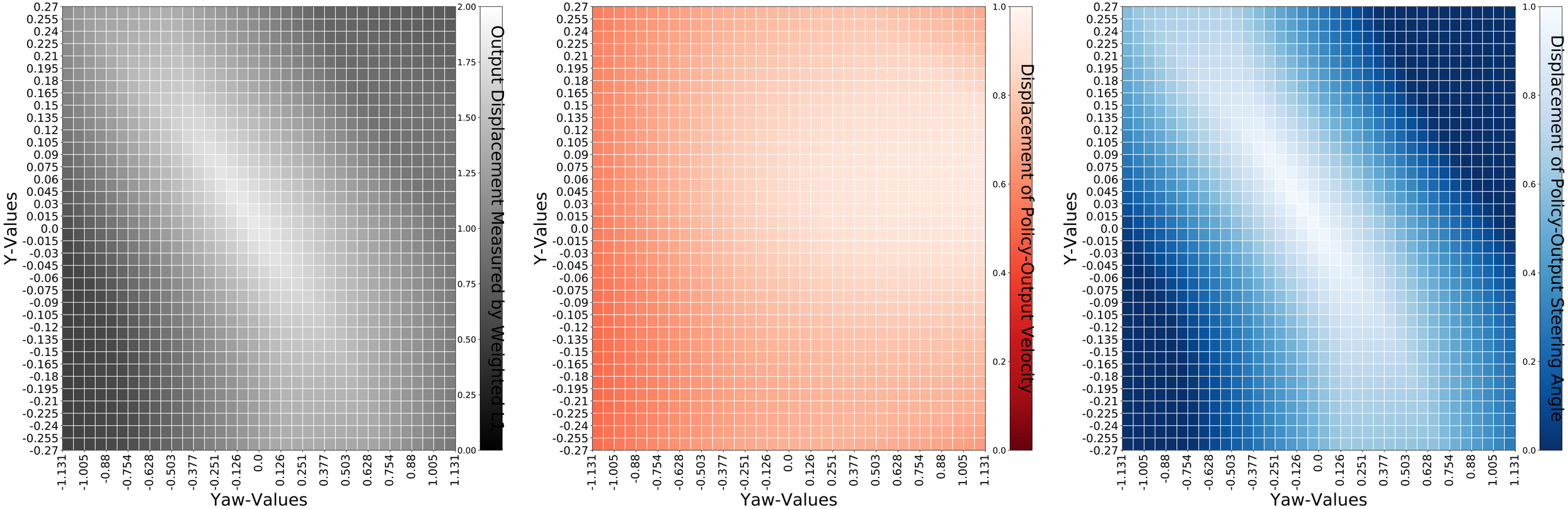


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0.415384, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 0f6e4fb2-f091-4584-8de9-22d789dc66ec
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_38_32_525821

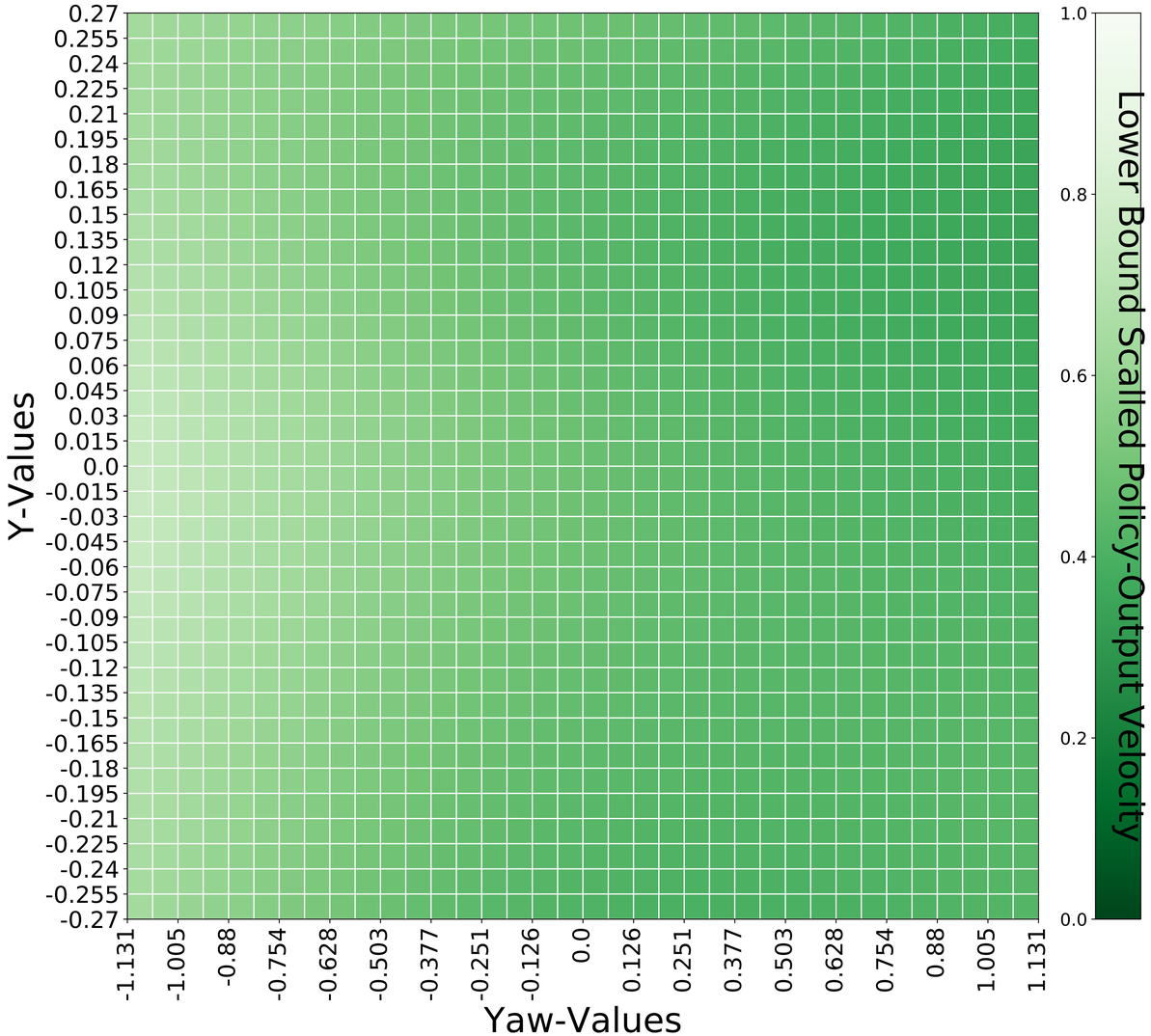


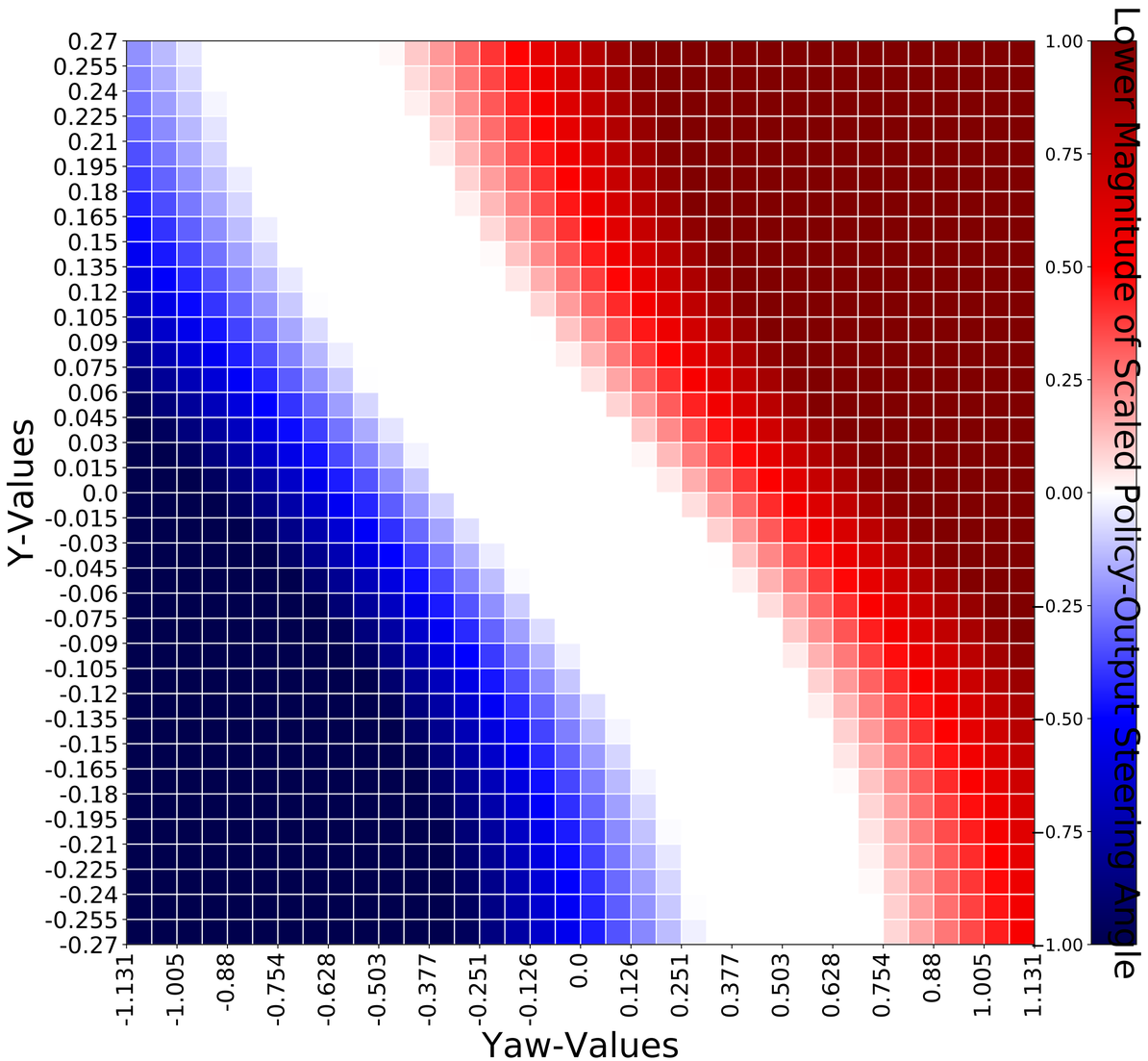


Line-Following Stability Results
xDot:0.0, yDot:0.415384, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: b4066a08-9850-44f6-bc49-43d4352ab40f
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_37_37_914413

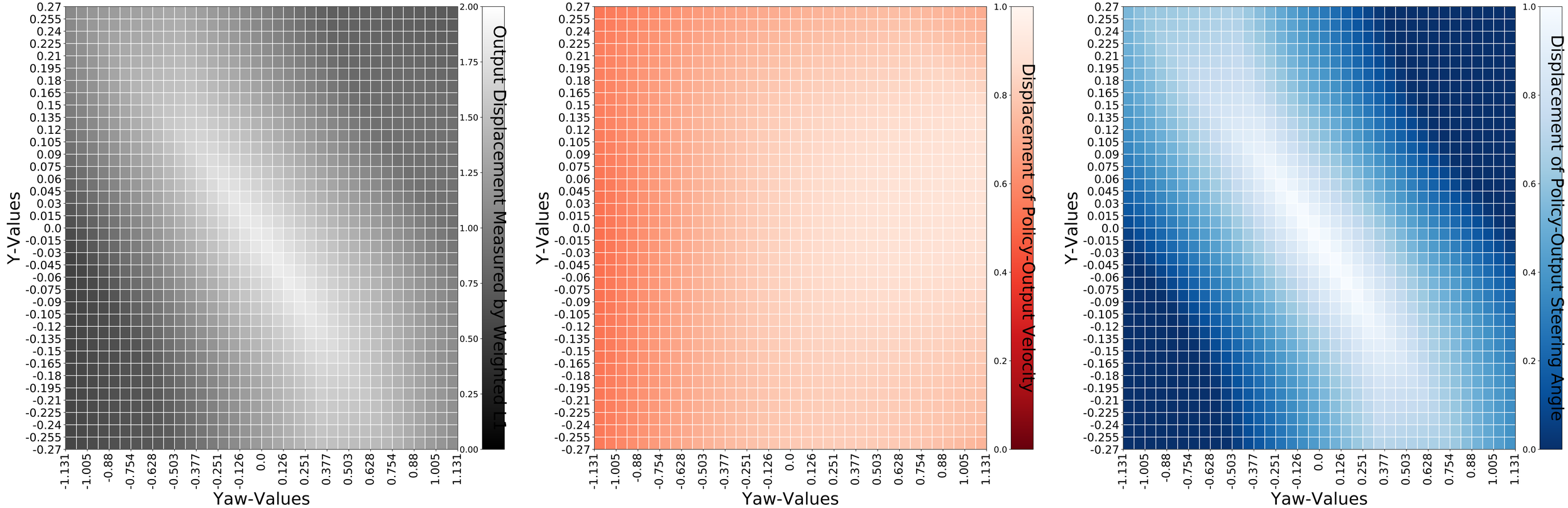


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0.415384, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 2b433809-53a2-4b56-b2ae-172c39a77d93
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_50_59_354750

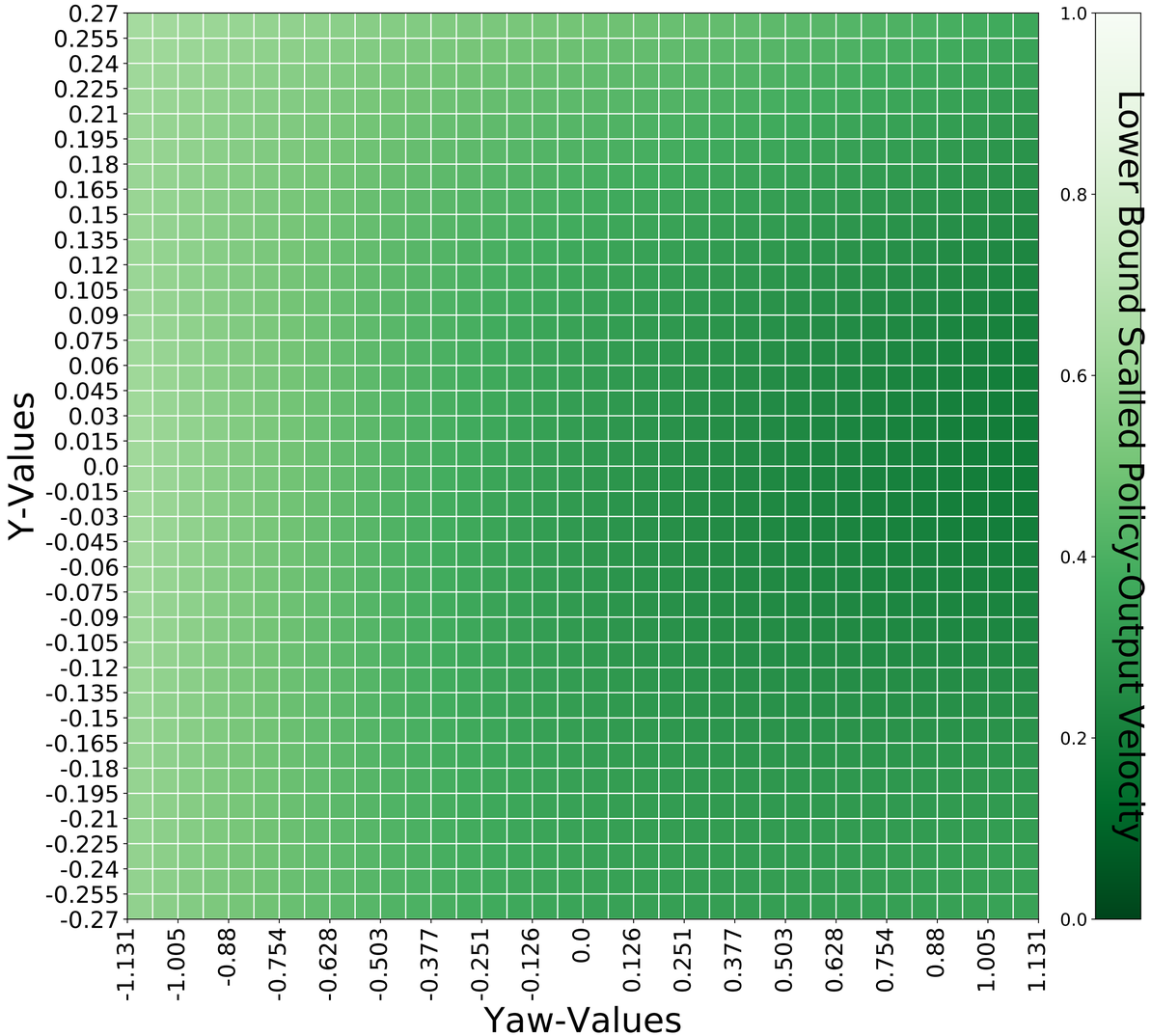


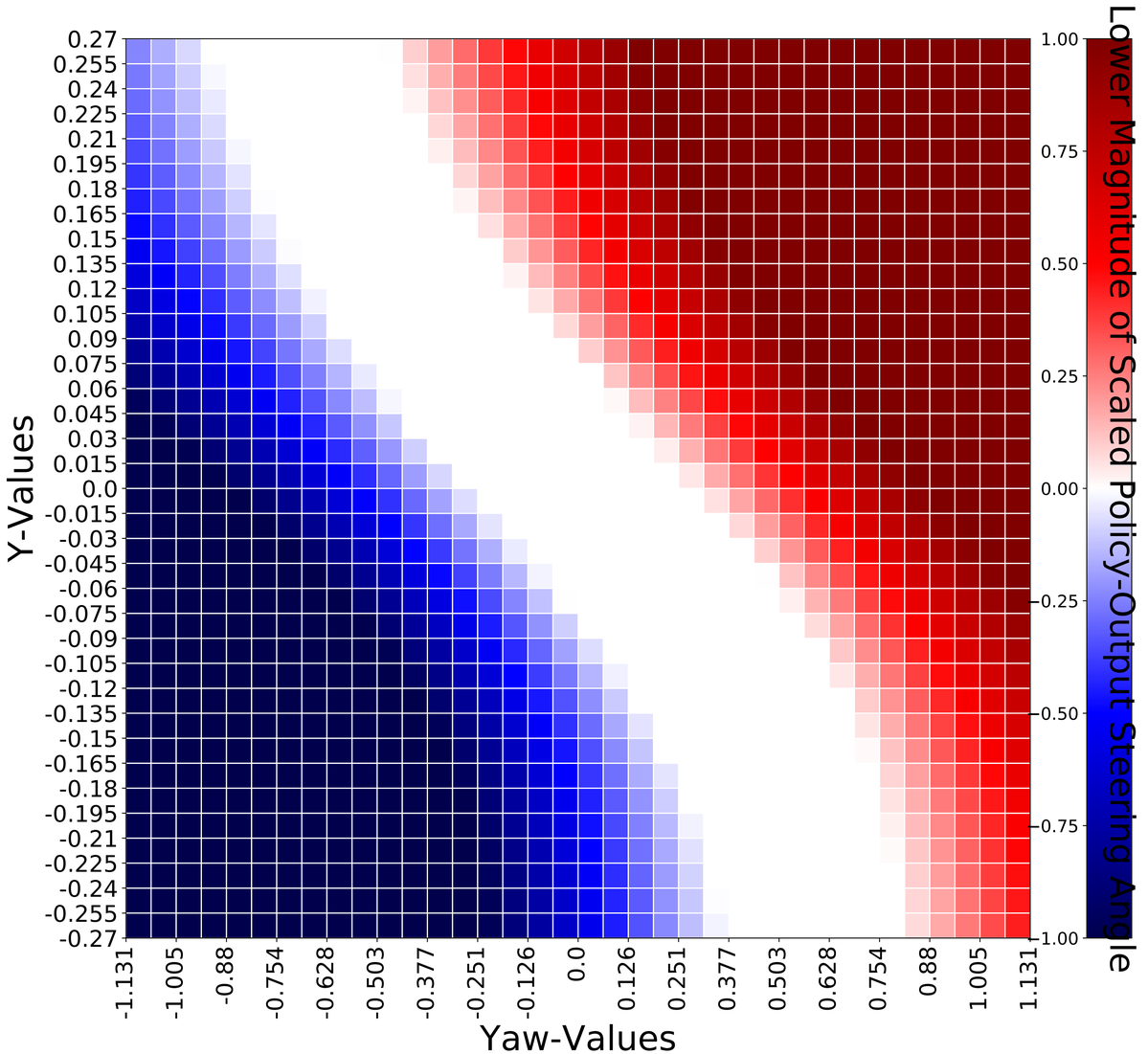


Line-Following Stability Results
xDot:0.0, yDot:0.415384, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 76ba96c5-baad-4006-8b2f-a5e17181cccf
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_19_49_55_486152

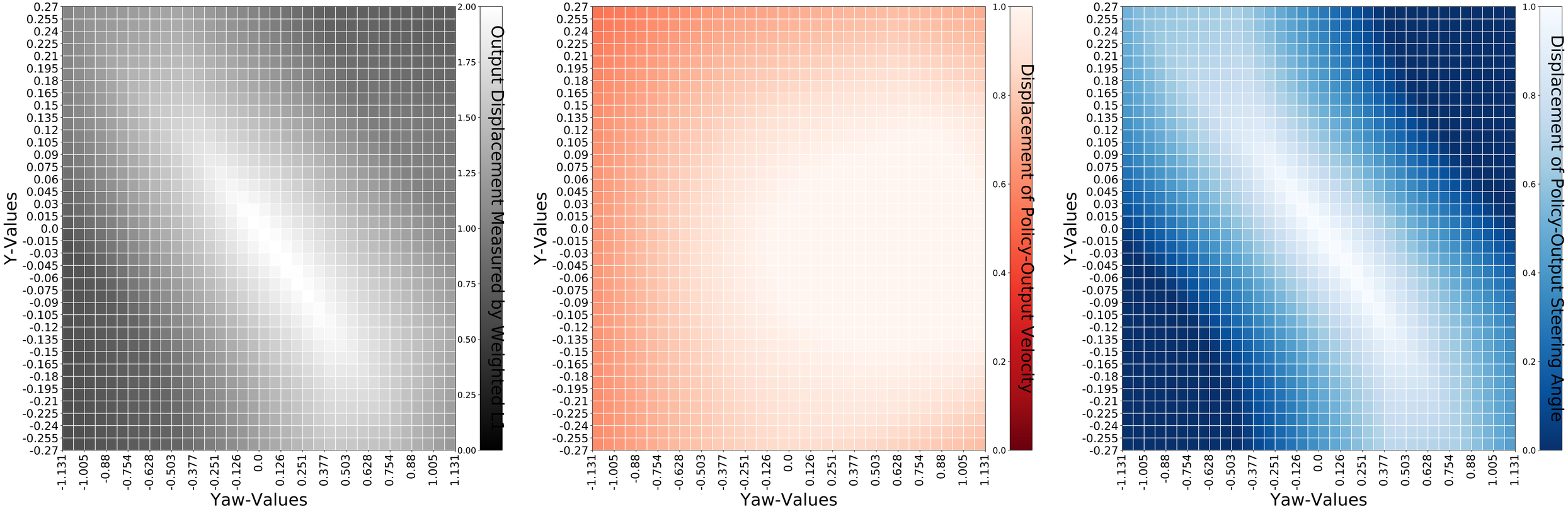


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0, yawDot: 0.153846, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 5f0eaf5b-d375-4c8b-9161-3f8d26df37c4
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_21_23_14_113518

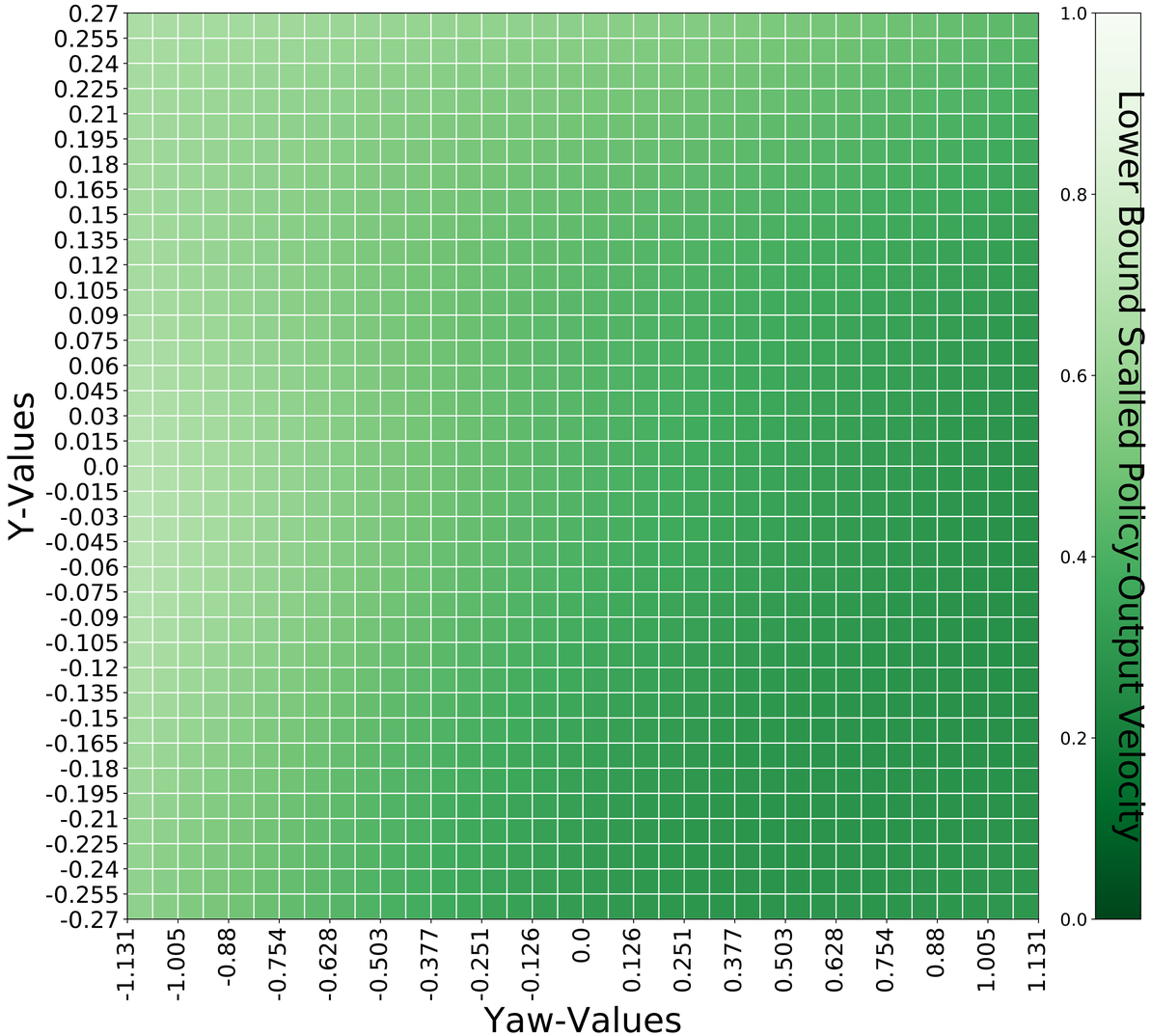


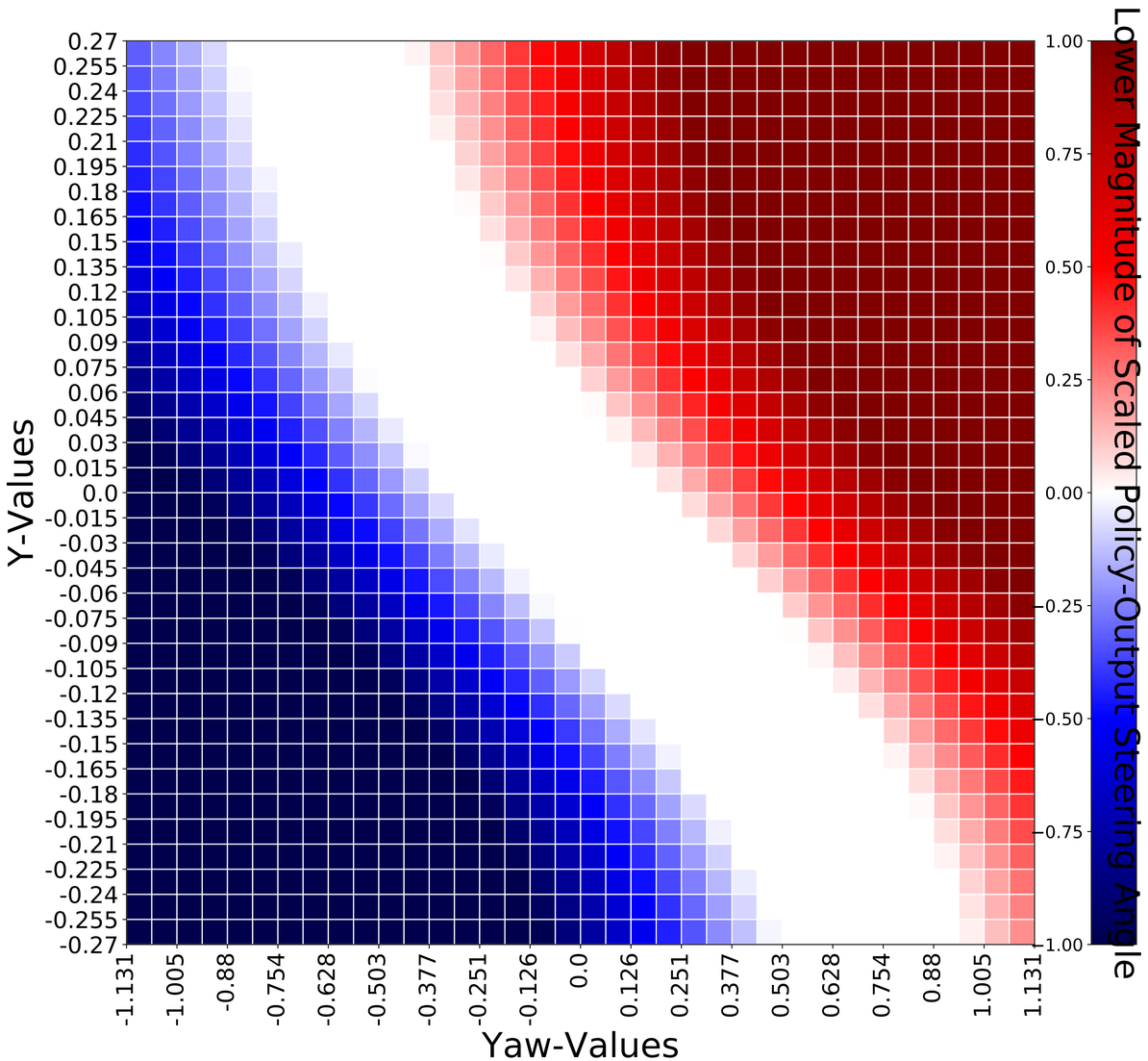


Line-Following Stability Results
xDot:0.0, yDot:0, yawDot: 0.153846, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 5c65c420-dfa3-4dbe-93c5-d838b7043917
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_21_22_13_946029

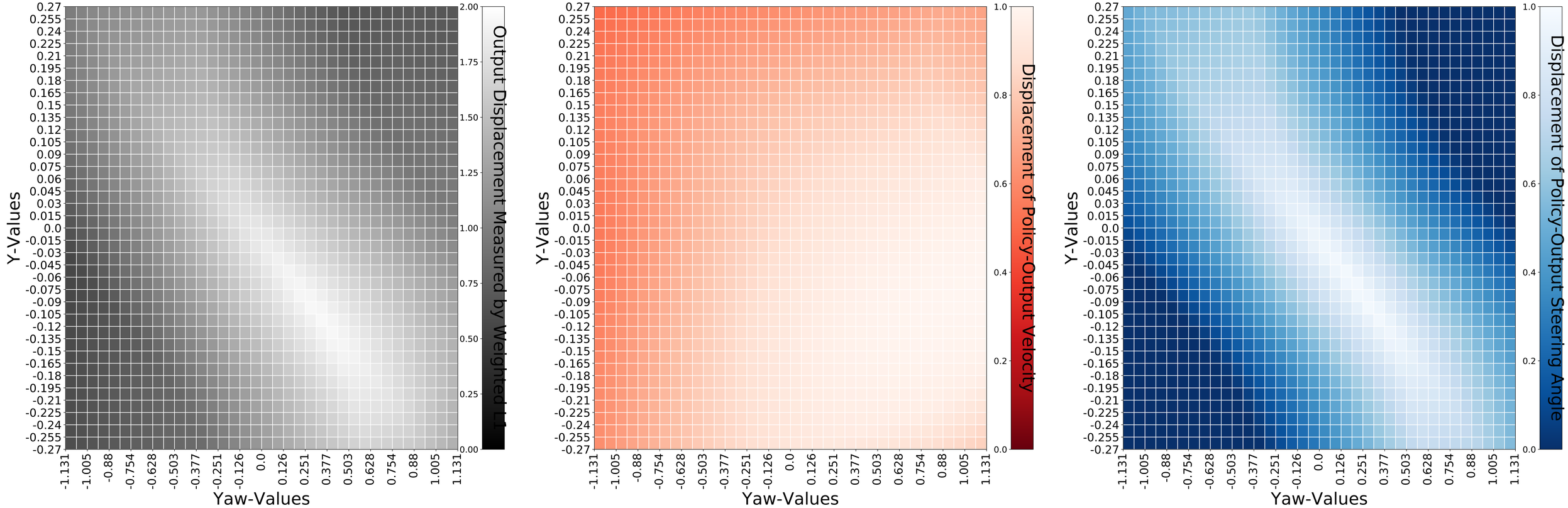


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0, yawDot: -1.07692, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 653b0a57-8afc-49ce-9101-9af98f16b04f
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_21_25_08_053684

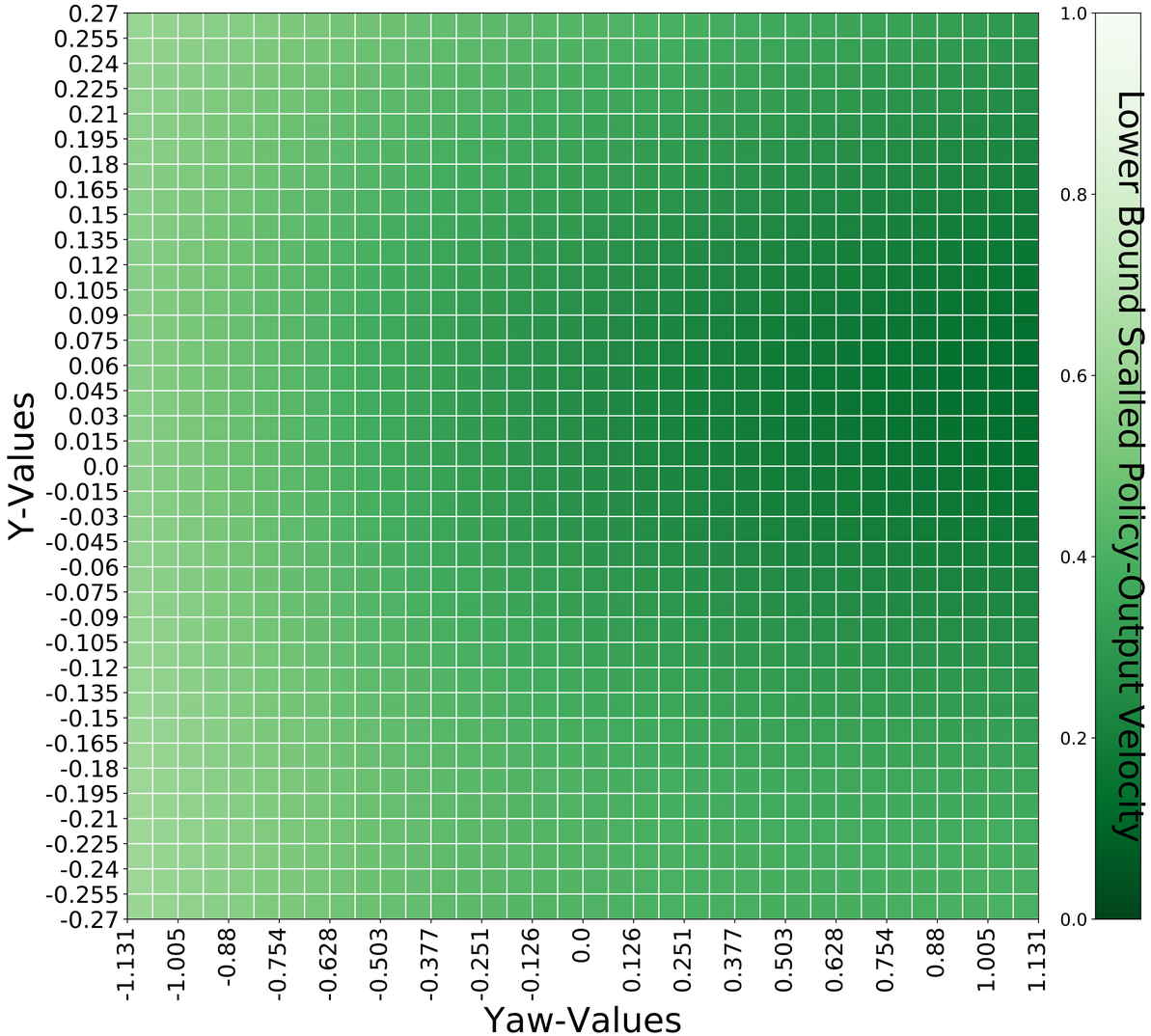


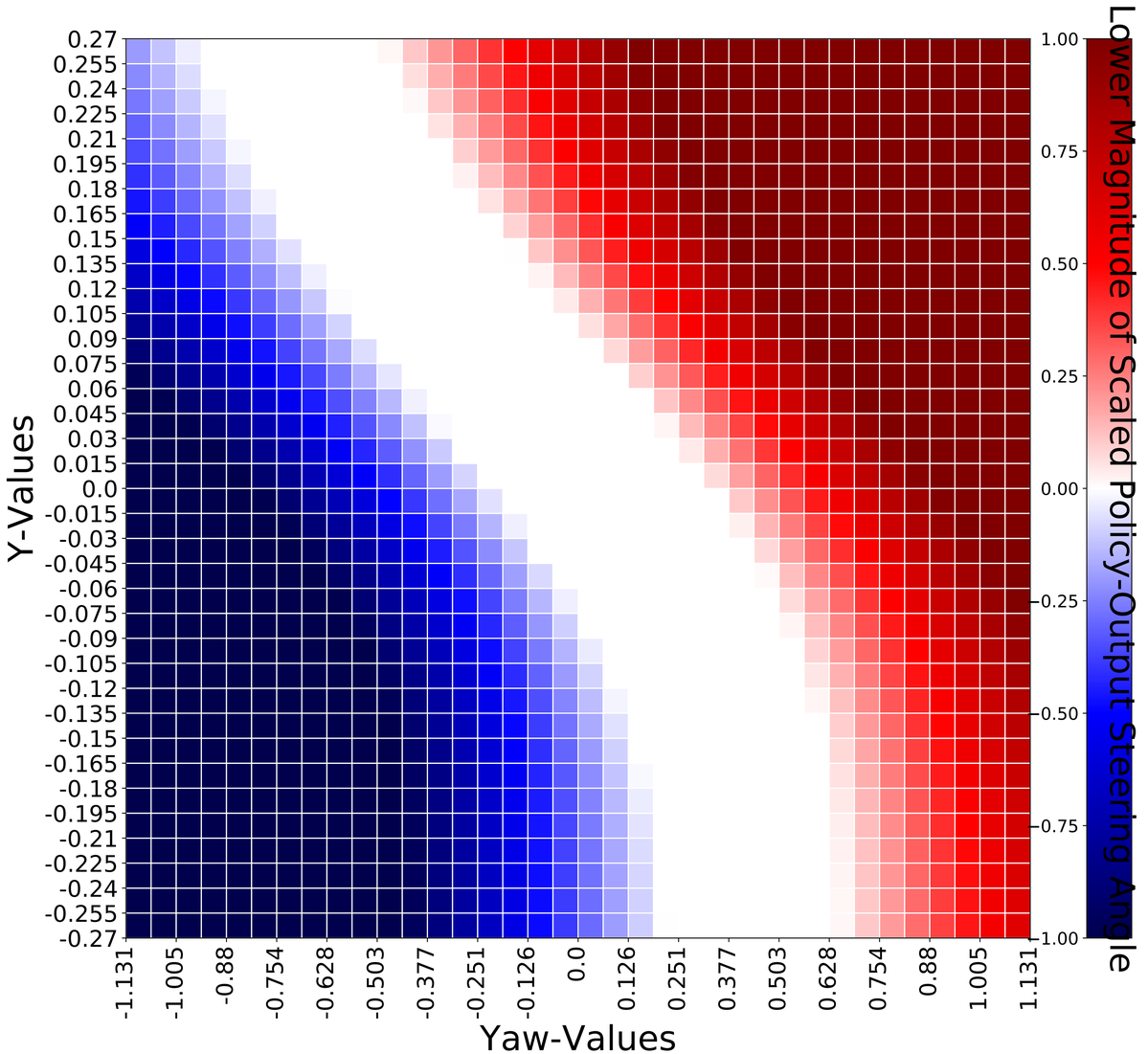


Line-Following Stability Results
xDot:0.0, yDot:0, yawDot: -1.07692, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 0fb393b2-69ed-4177-b23e-ea68fb2d6150
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_21_24_11_986971

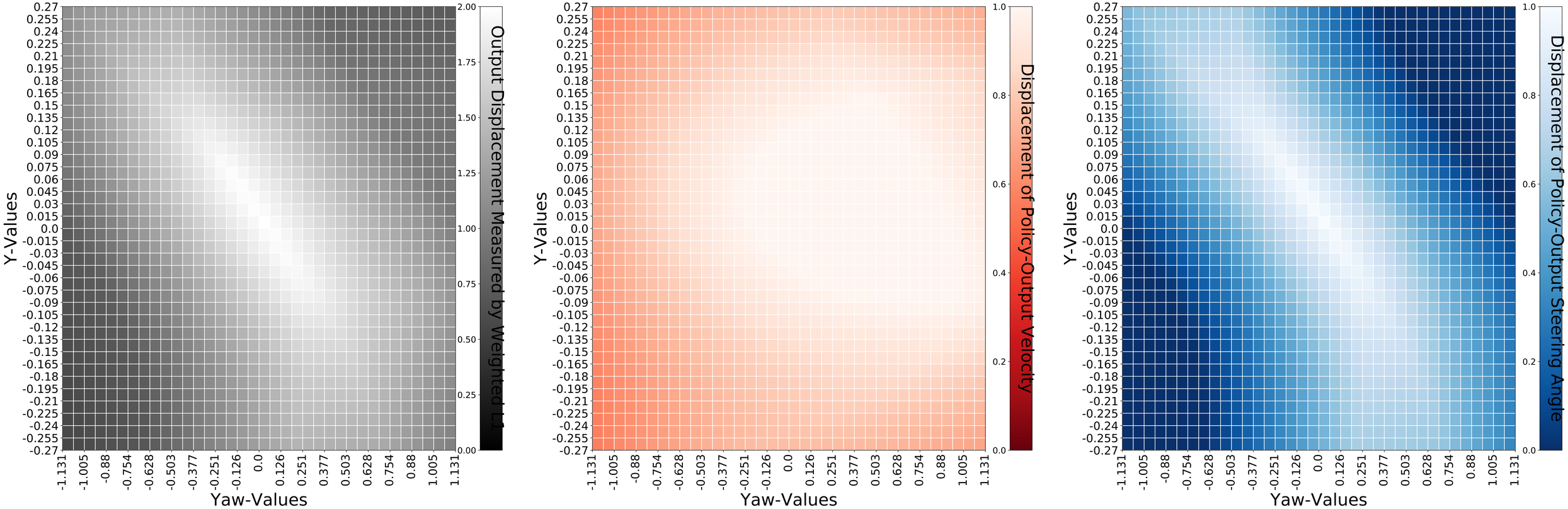


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0, yawDot: 1.384615, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 408f299c-8871-4244-a4b4-12e3ab9da7df
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_21_26_56_297794

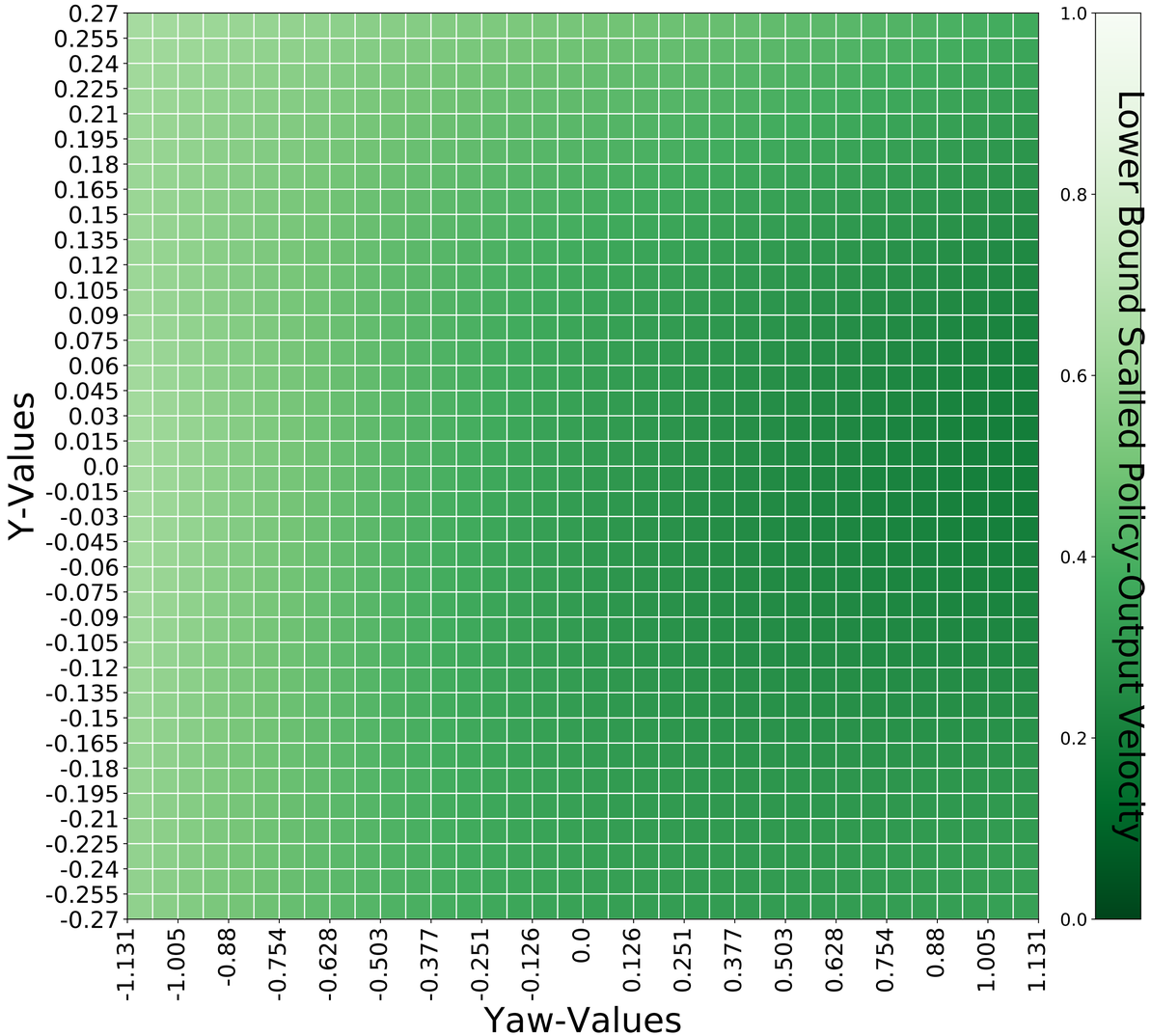


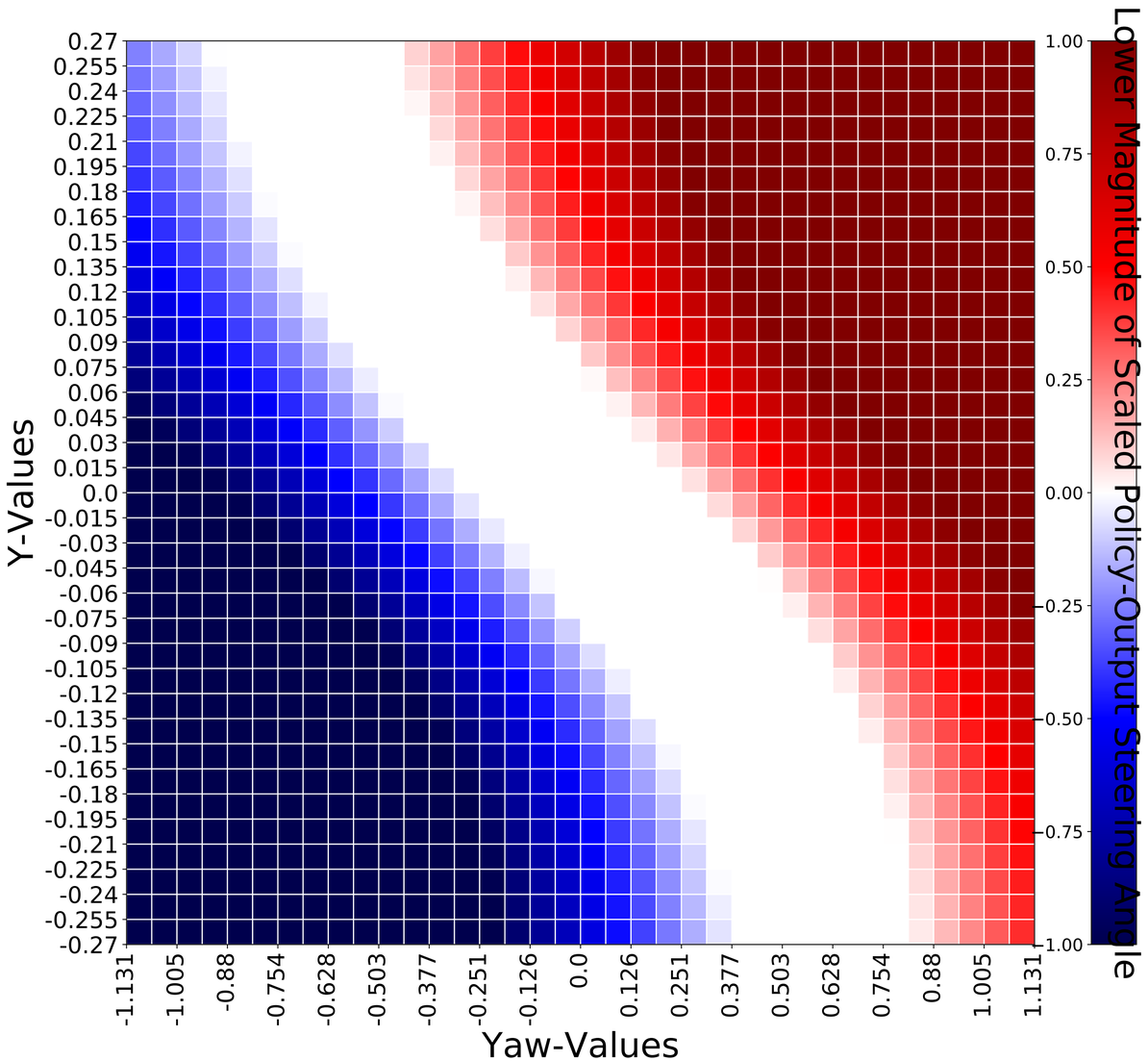


Line-Following Stability Results
xDot:0.0, yDot:0, yawDot: 1.384615, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 867d5f72-65ed-4184-b5f7-d040a1e21268
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_21_26_02_277737

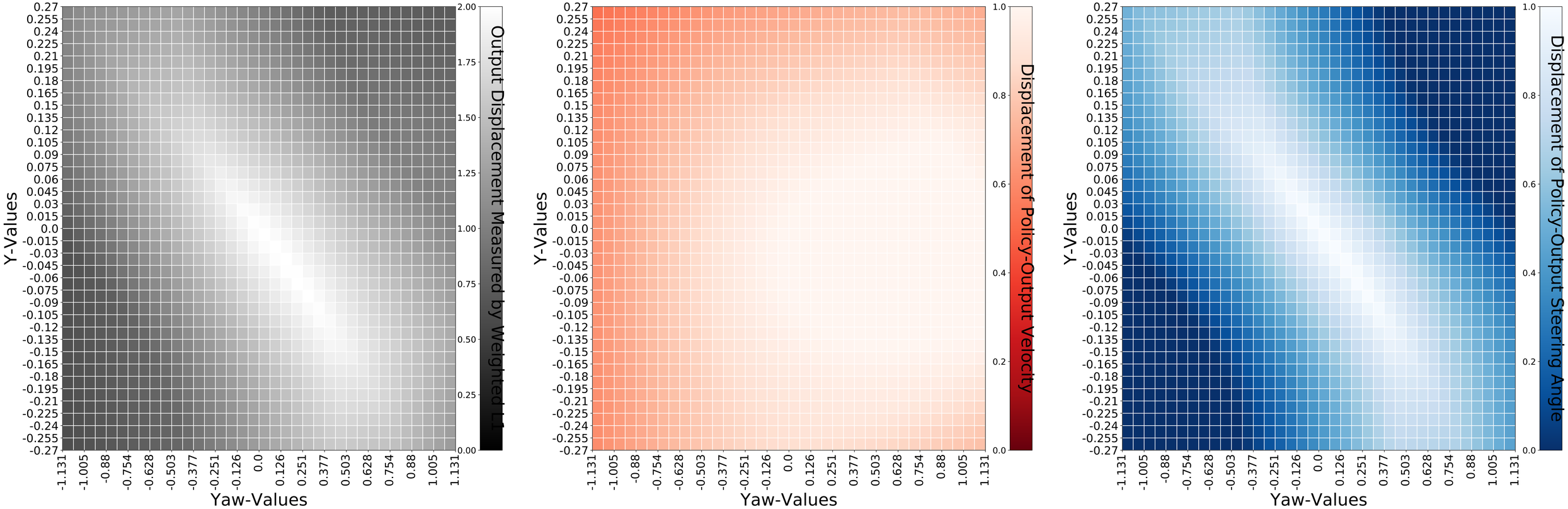


Line-Following Min. Magnitude Output Results
xDot:0.0, yDot:0, yawDot: 0, percentEpsilonToUse:0.015
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: c9840977-26b6-469f-979e-26ccde2bd9c4
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_21_39_00_951451



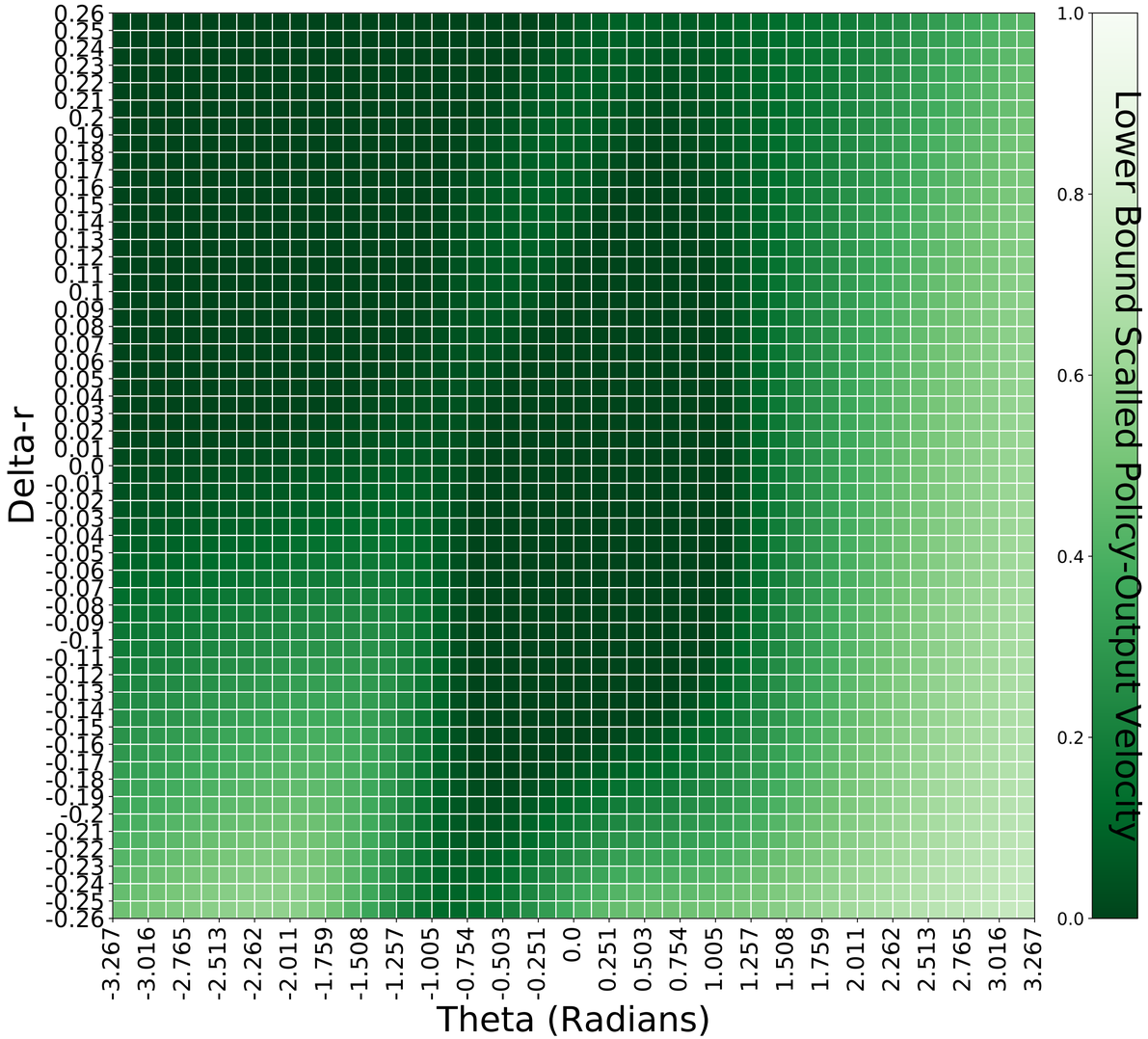


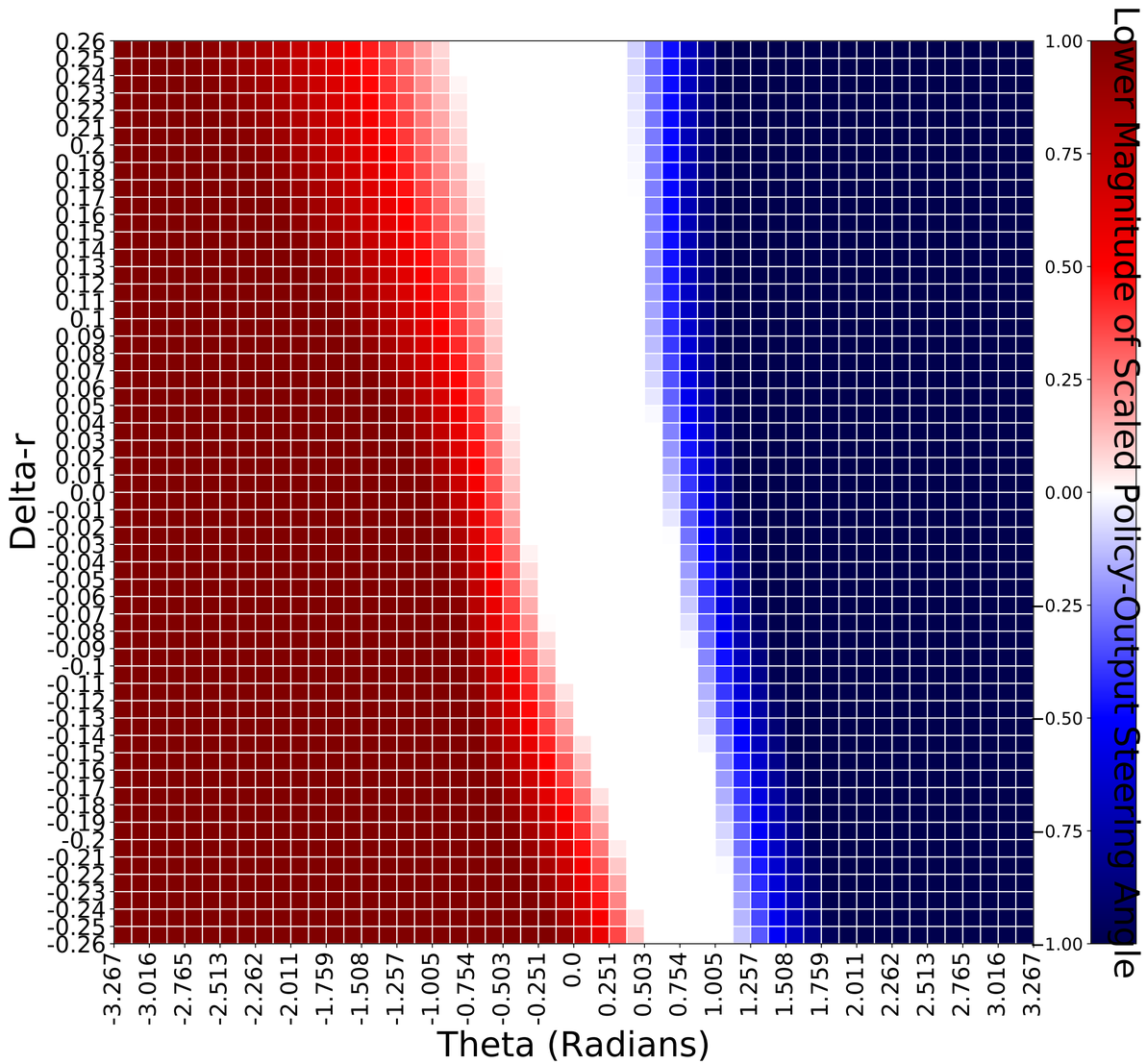
Line-Following Stability Results
xDot:0.0, yDot:0, yawDot: 0, percentEpsilonToUse:0.015
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: c5040b43-2531-4a18-9622-26c8bed3709e
git commit hash: 3593794c01d44d16caef2fa62af1695fdae9d36d
time plot formed: 2019_06_08_21_38_02_747843



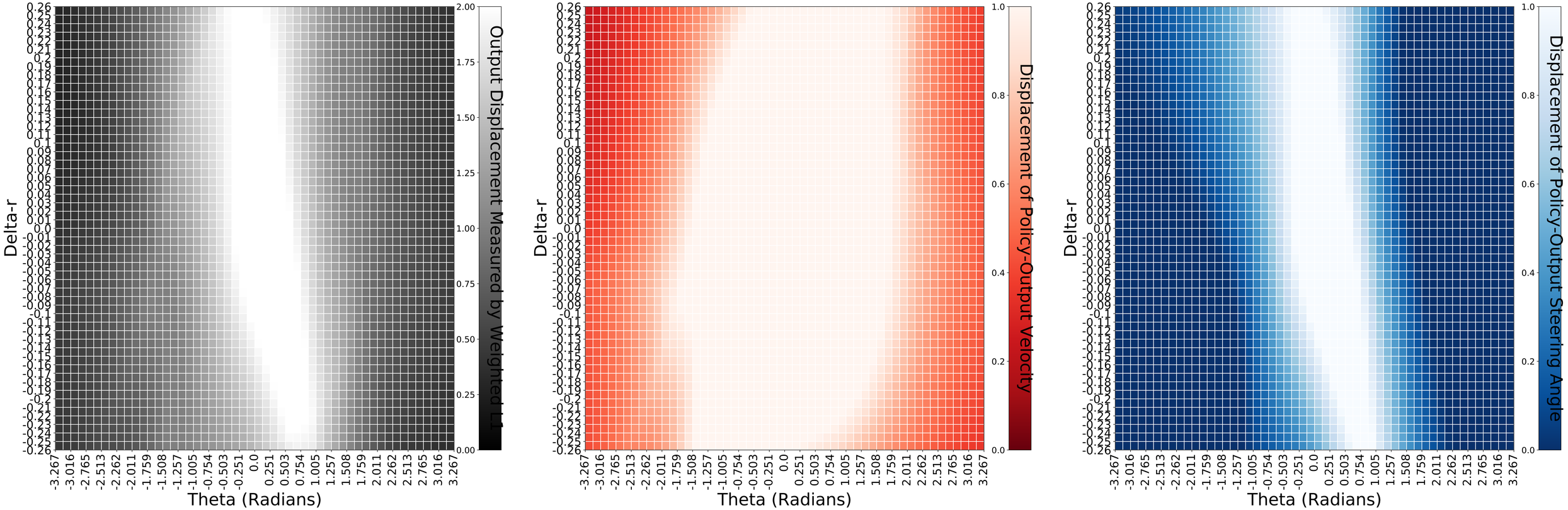
Circle-Following Policy Results

Circle-Following Min. Magnitude Output Results
deltaR-dot:0.115384, deltaTheta-dot:0.278846, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 71a3fe97-961f-4ea9-936e-05c42468fb95
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_22_59_46_127905

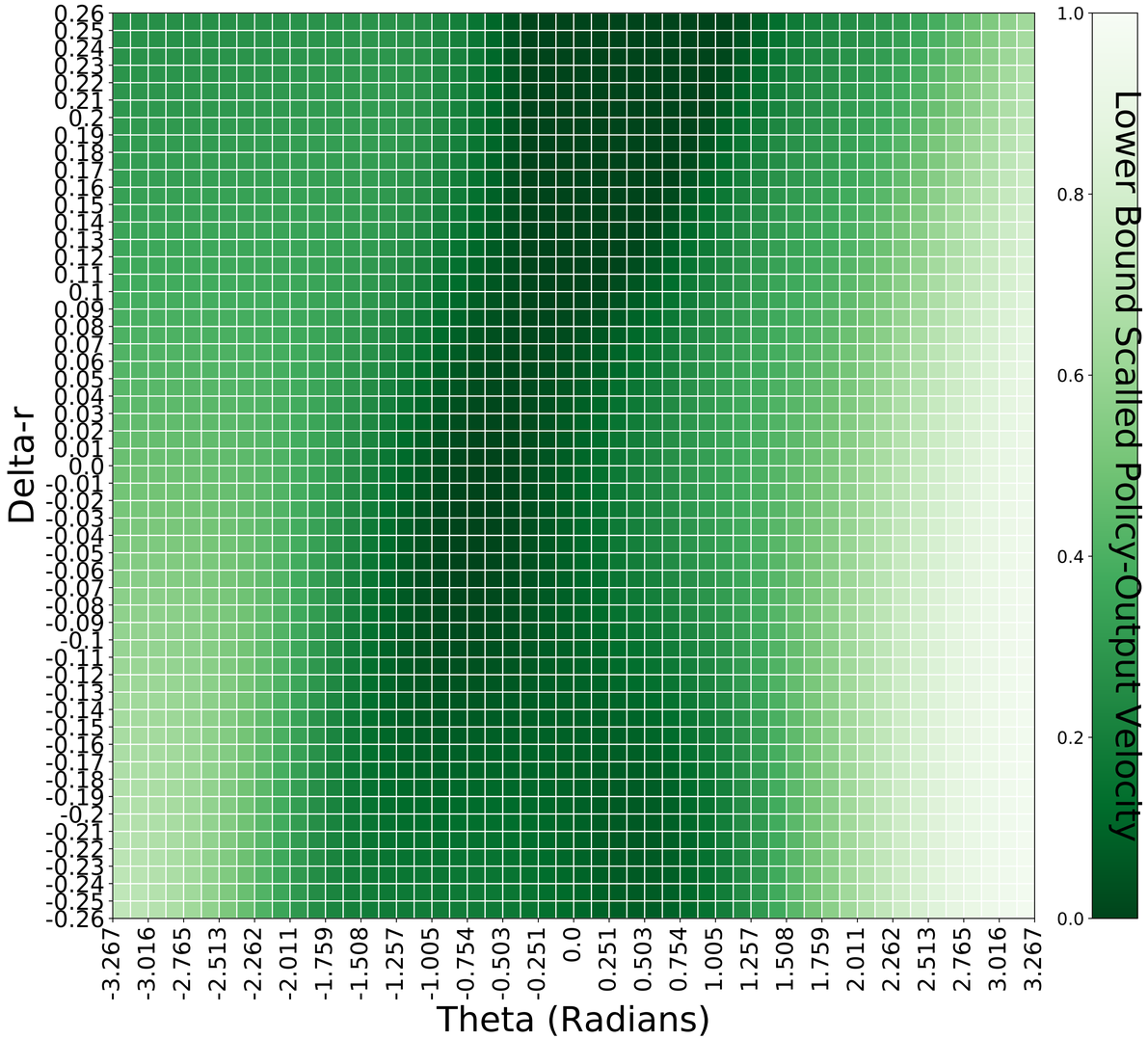


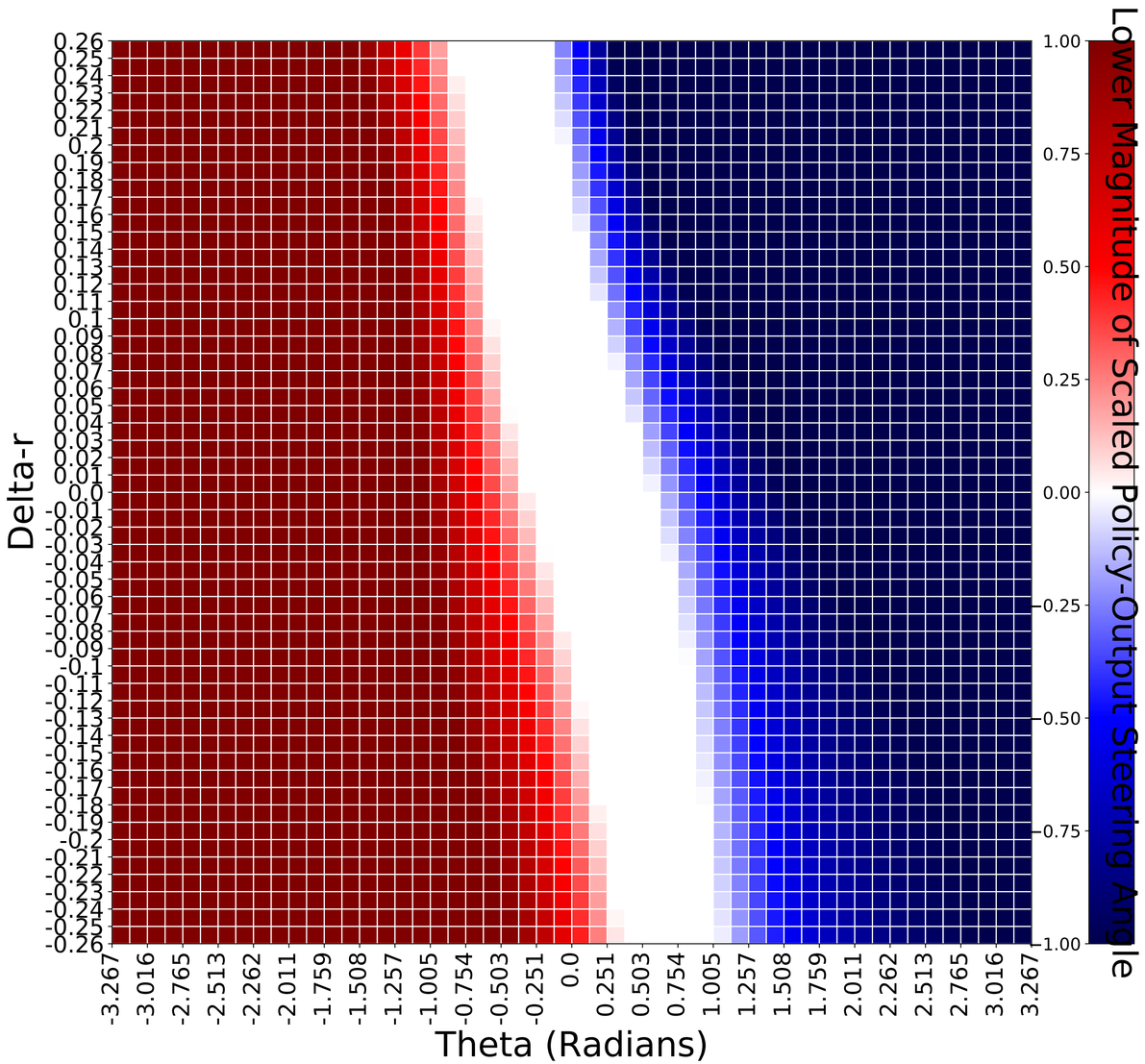


Circle-Following Stability Results
deltaR-dot:0.115384, deltaTheta-dot:0.278846, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: e3288b6e-b4b1-4de6-979d-48155b7d894d
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_22_57_47_057526

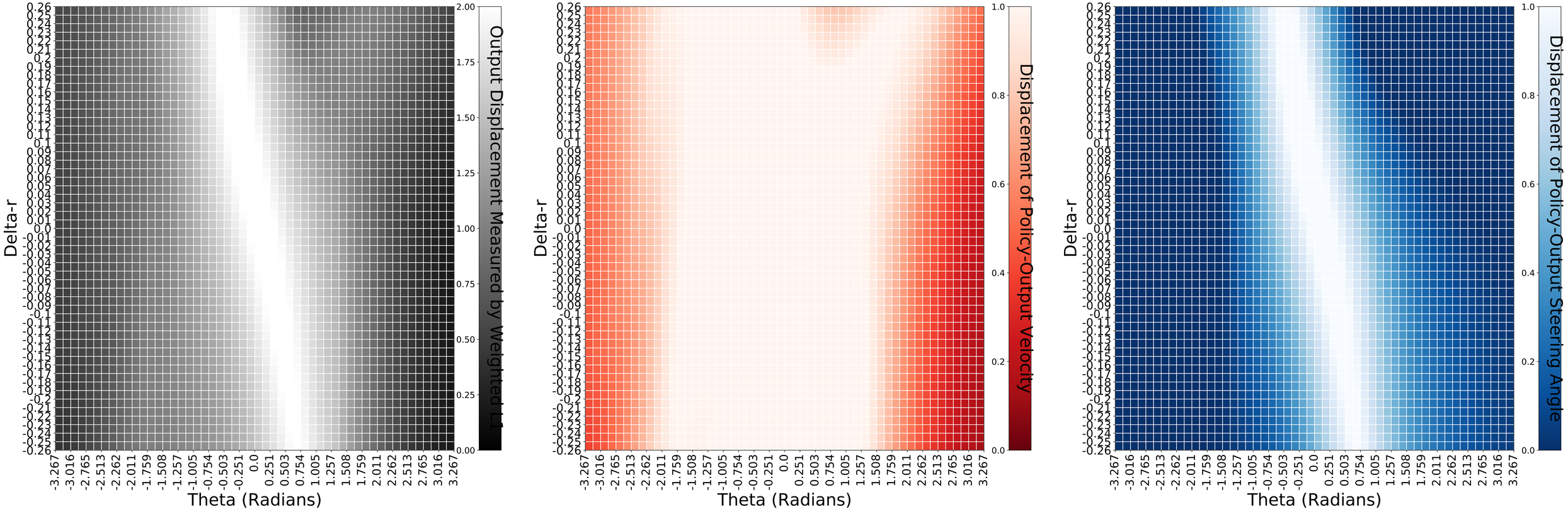


Circle-Following Min. Magnitude Output Results
deltaR-dot:0.115384, deltaTheta-dot:-1.95192, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 8d9514b6-264c-44c9-b3d0-2c98bd41afdd
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_03_45_172603

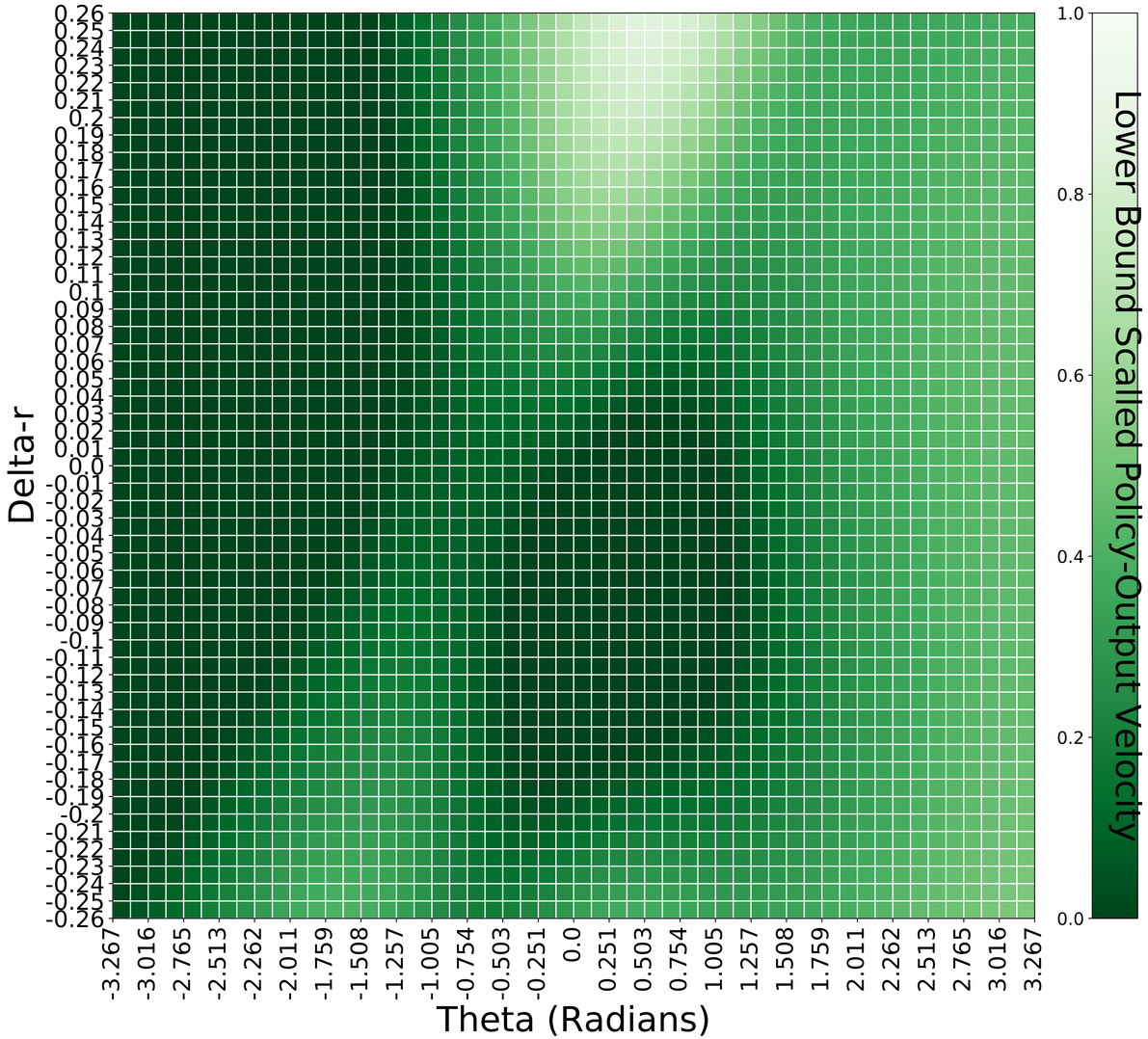


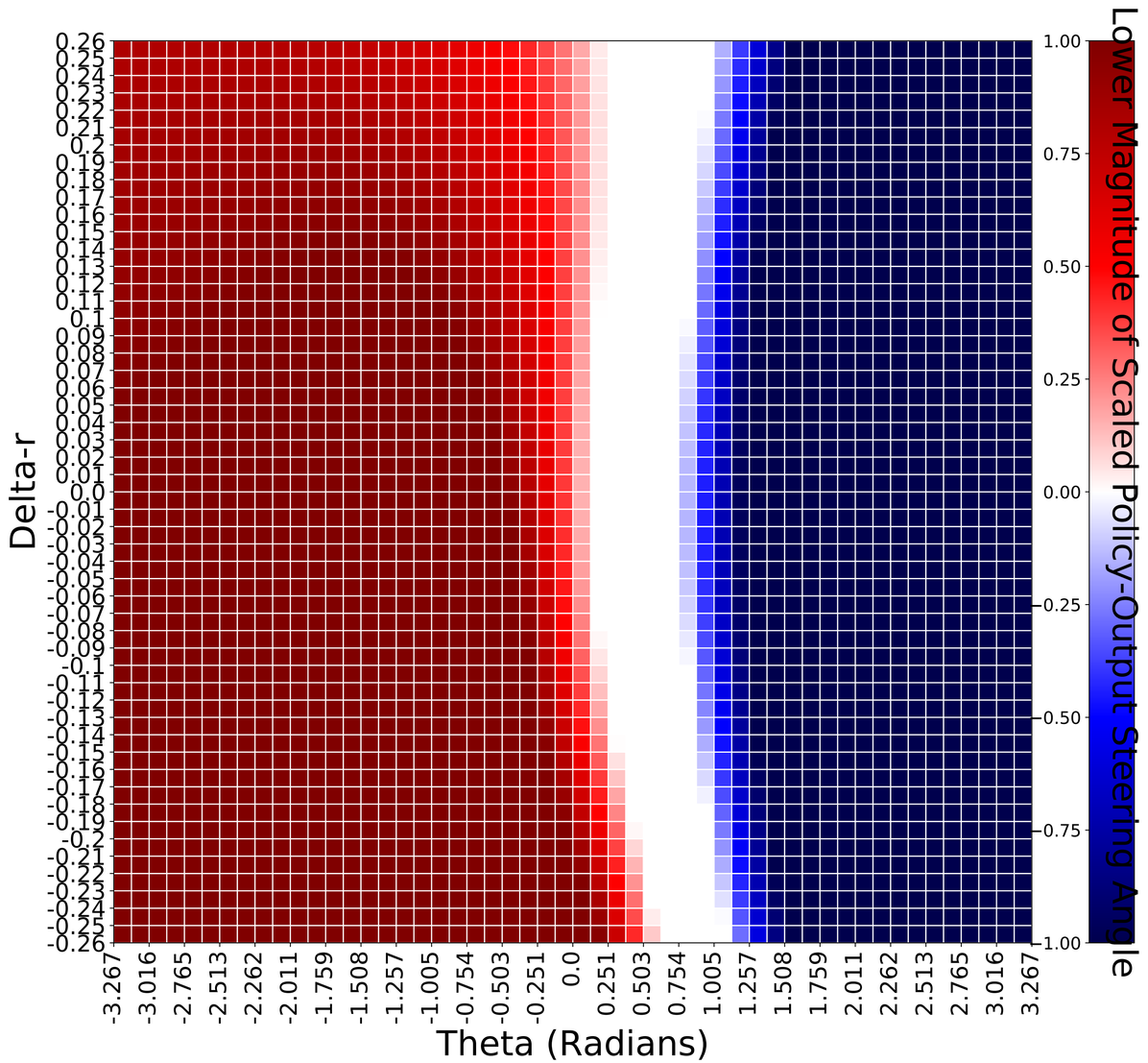


Circle-Following Stability Results
deltaR-dot:0.115384, deltaTheta-dot:-1.95192, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 41533675-4409-4824-be69-ae2336566607
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_01_47_376141

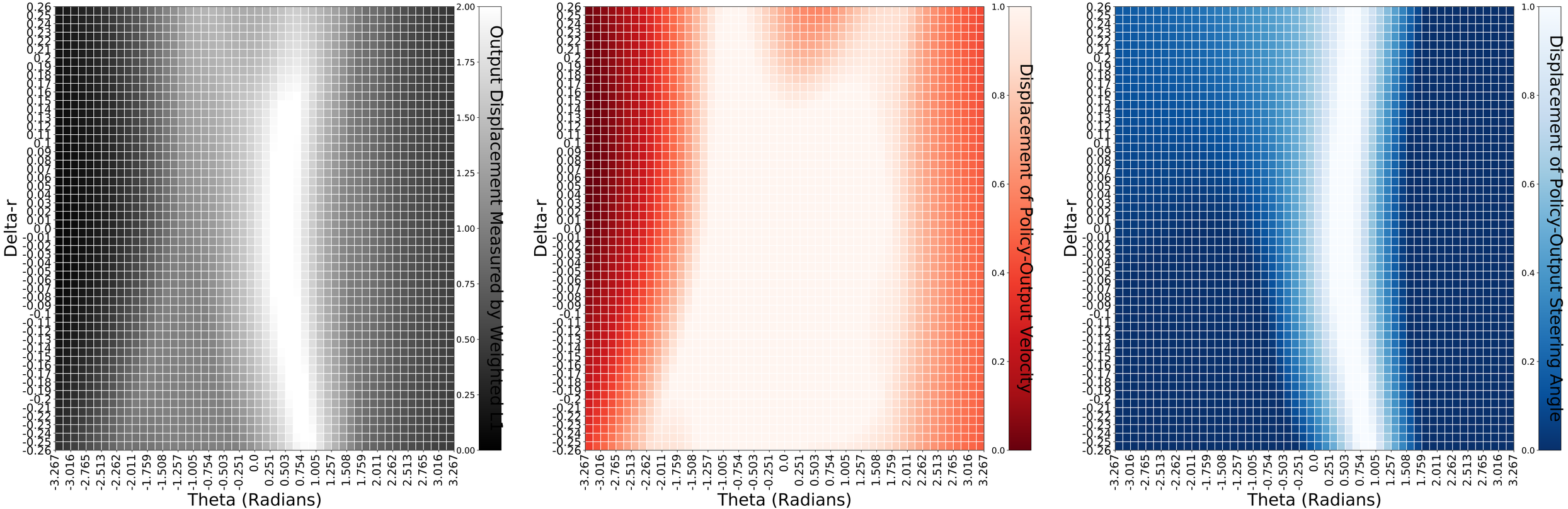


Circle-Following Min. Magnitude Output Results
deltaR-dot:0.115384, deltaTheta-dot:2.509615, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 782391d5-f593-4c2e-8860-8cec6d58aed9
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_07_44_246933

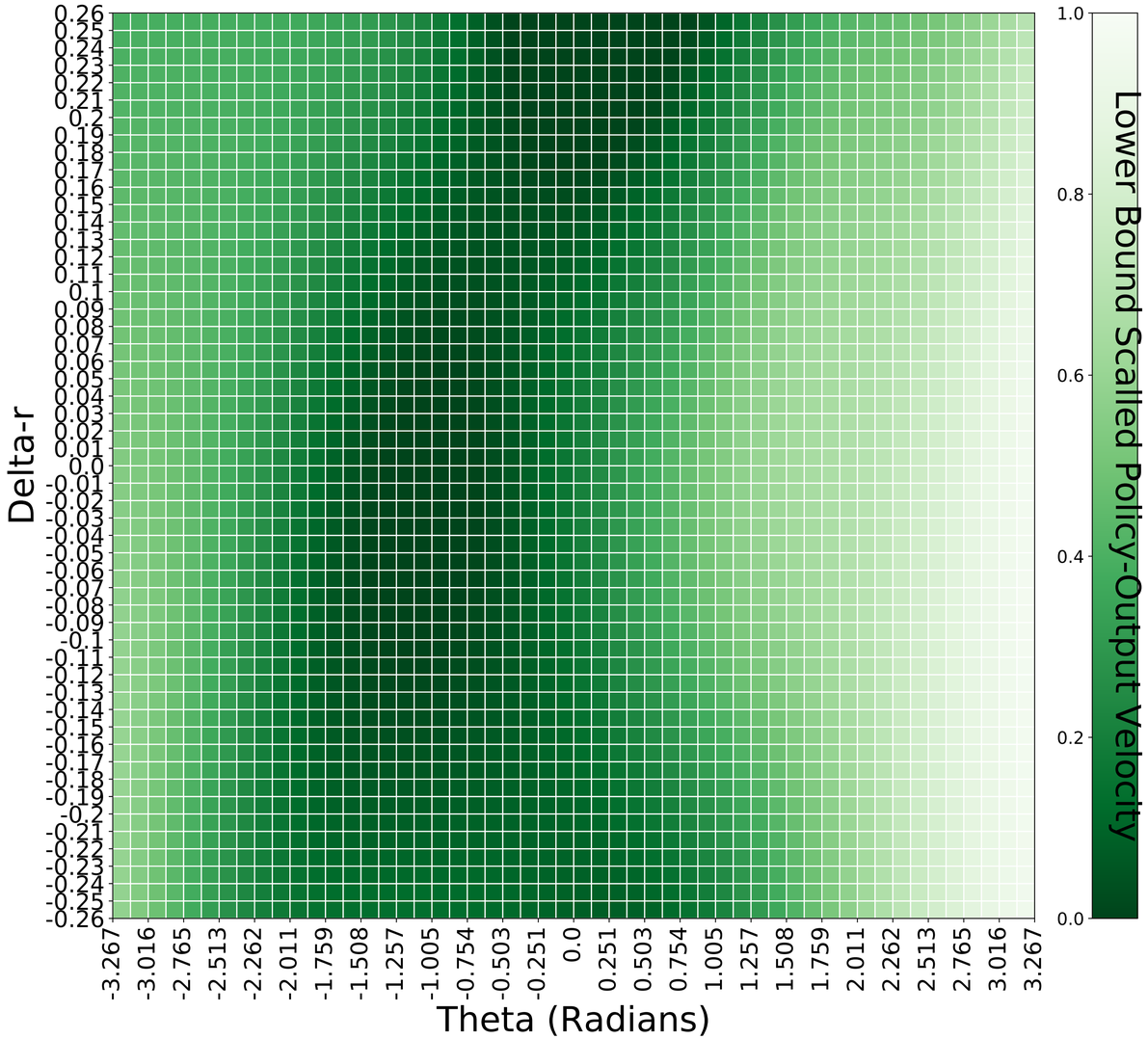


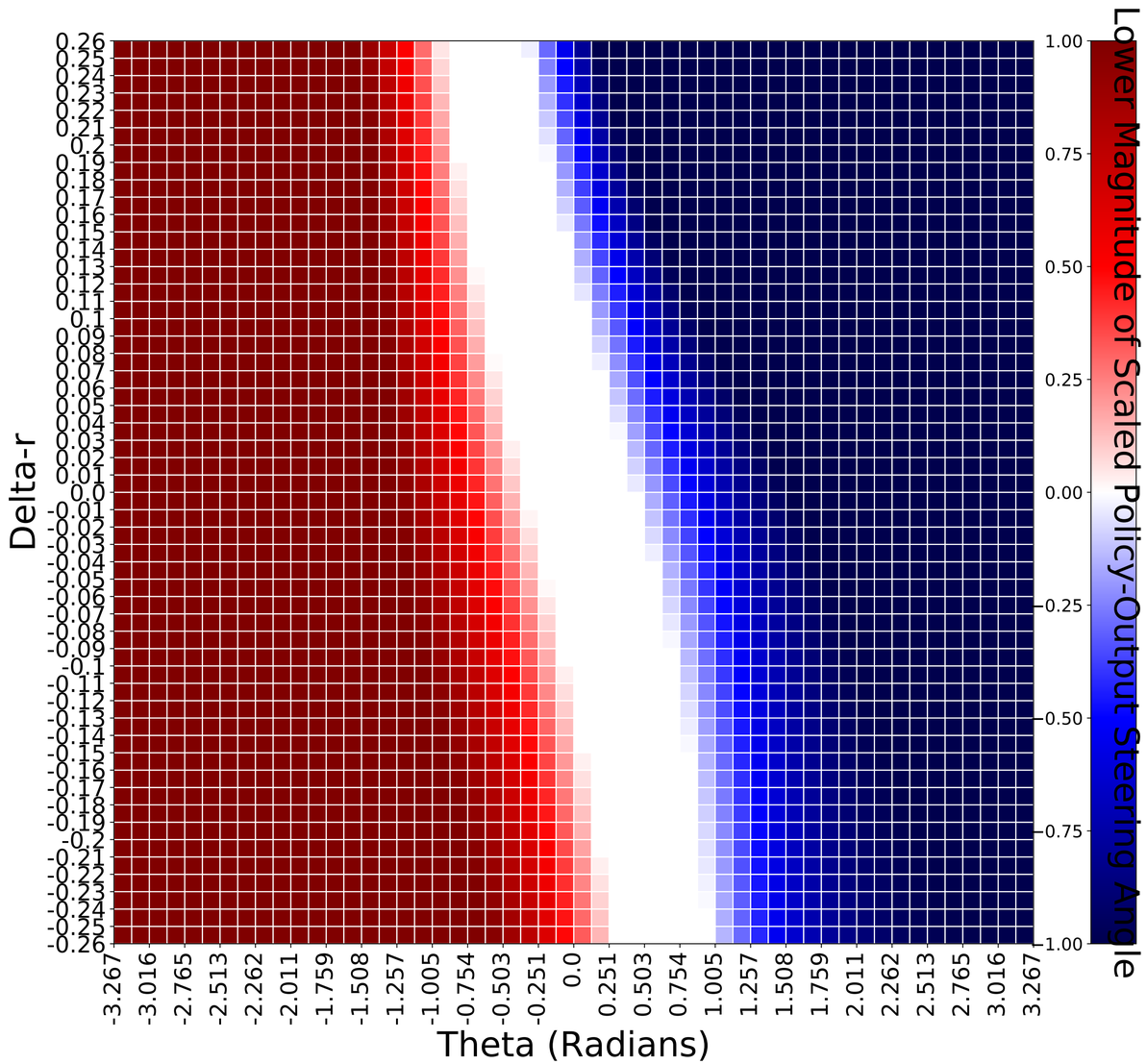


Circle-Following Stability Results
deltaR-dot:0.115384, deltaTheta-dot:2.509615, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: f51274da-608a-4349-914b-2ce3282e5e1a
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_05_47_835606

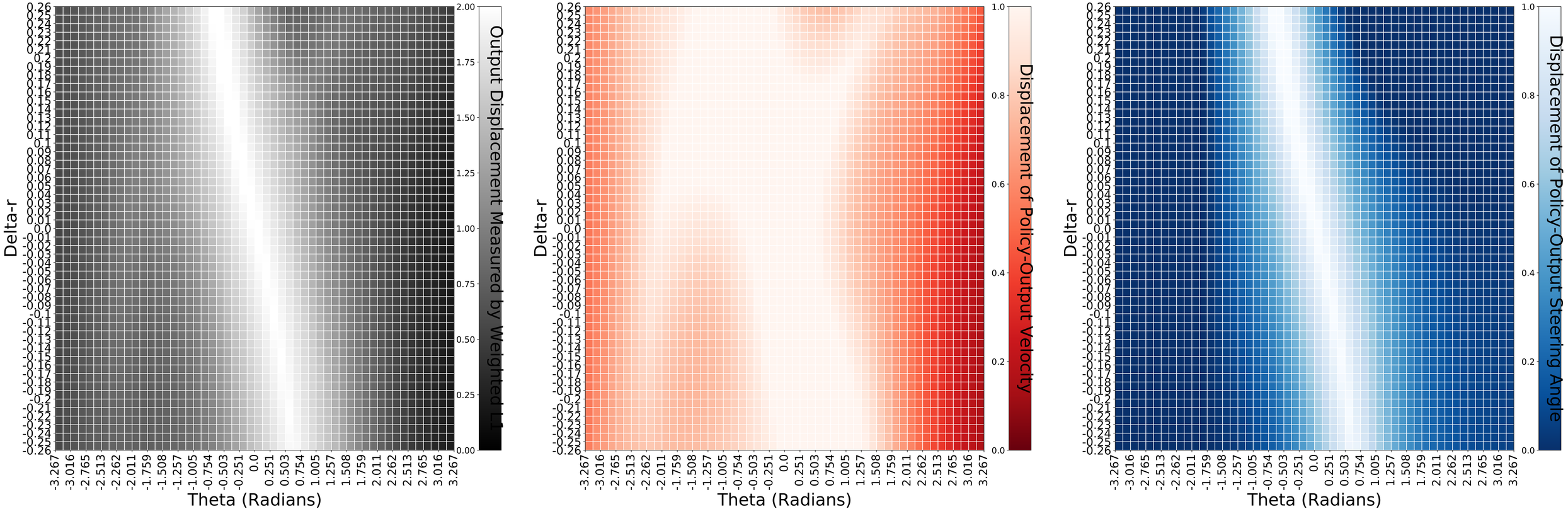


Circle-Following Min. Magnitude Output Results
deltaR-dot:0.115384, deltaTheta-dot:-3.06730, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: f67e937d-54cd-45a2-9b2f-c642d0363973
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_11_33_979756

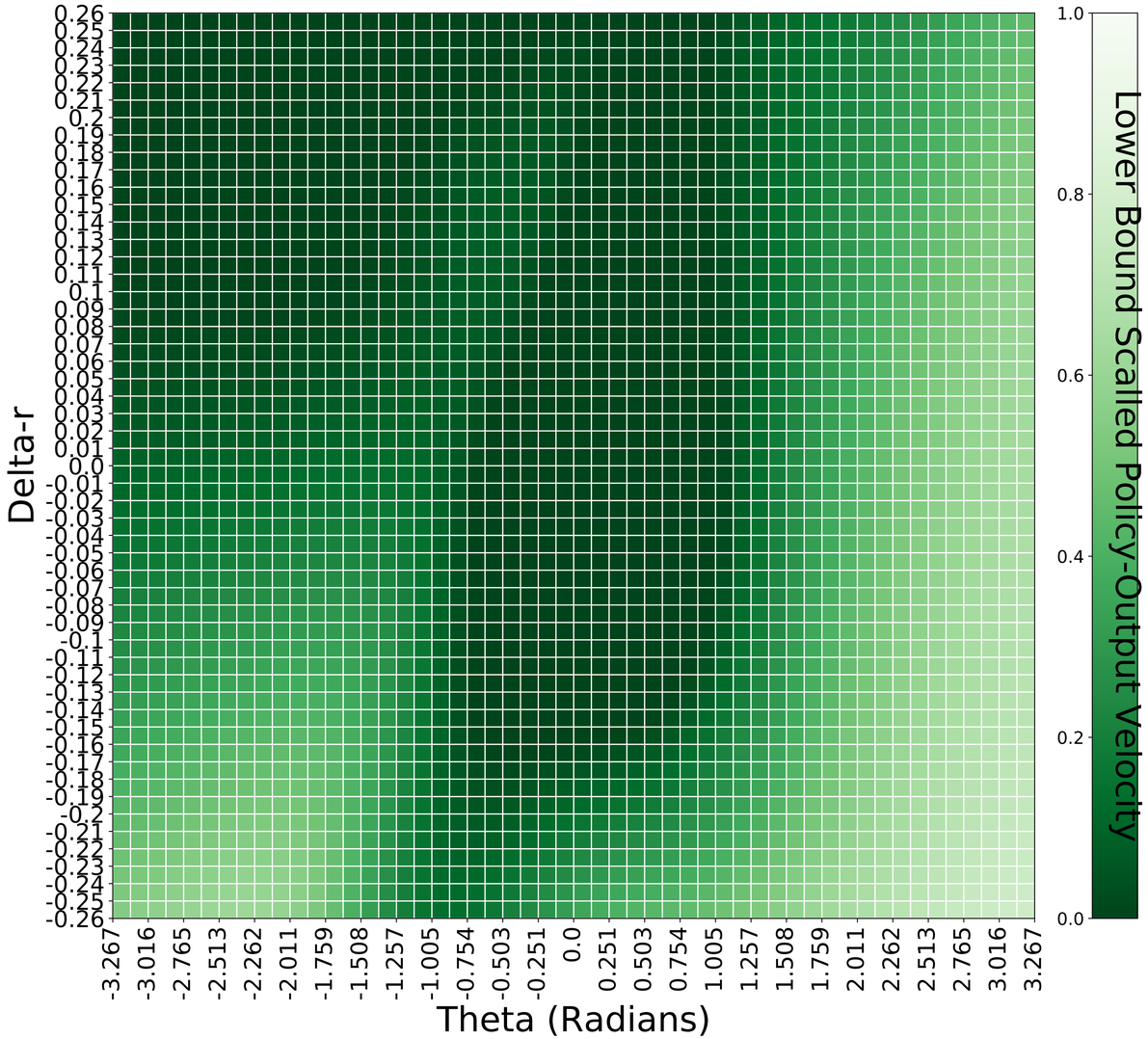


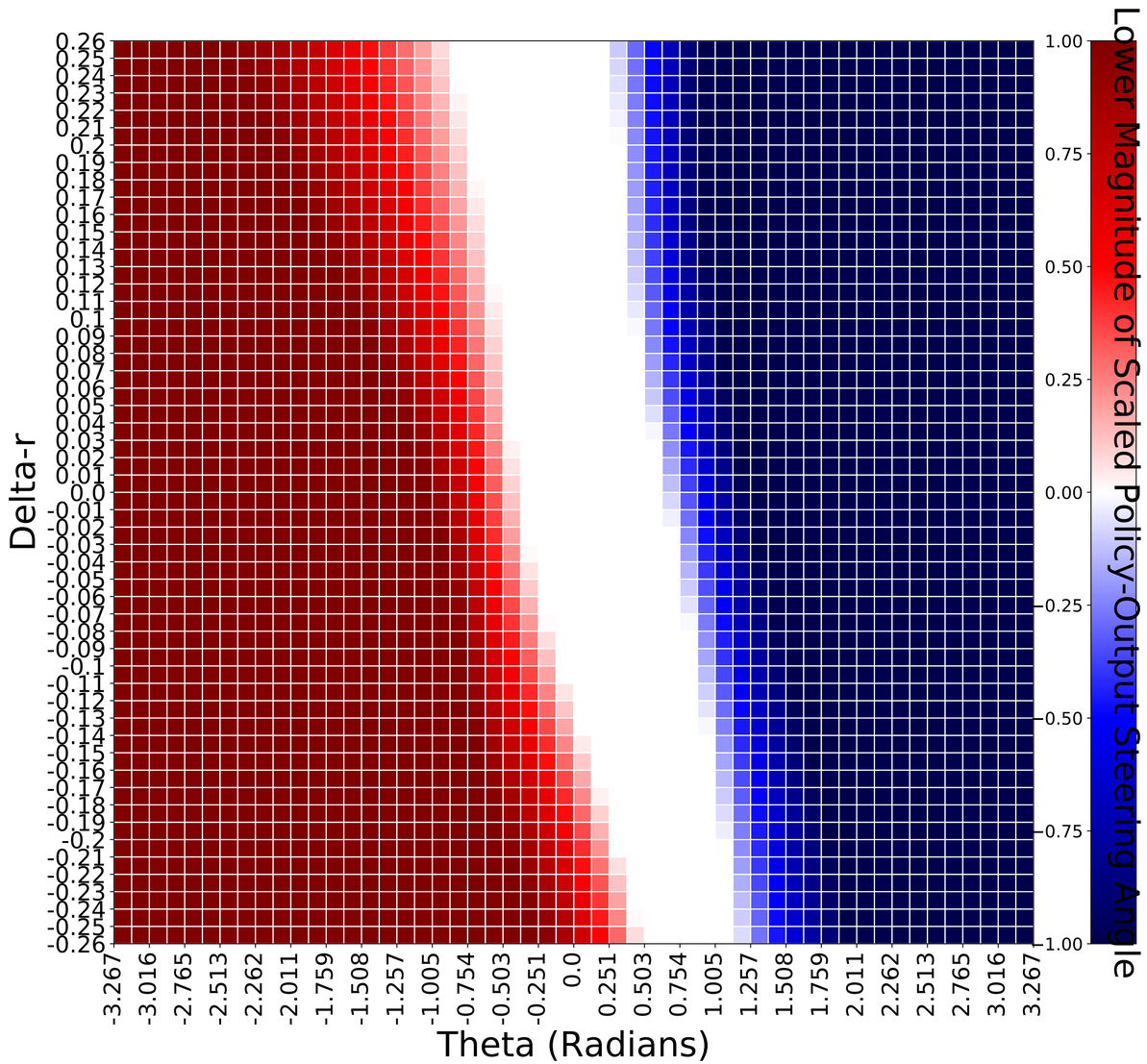


Circle-Following Stability Results
deltaR-dot:0.115384, deltaTheta-dot:-3.06730, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: dca3ea17-b17b-4ec6-8b1b-8f253546374a
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_09_36_612672

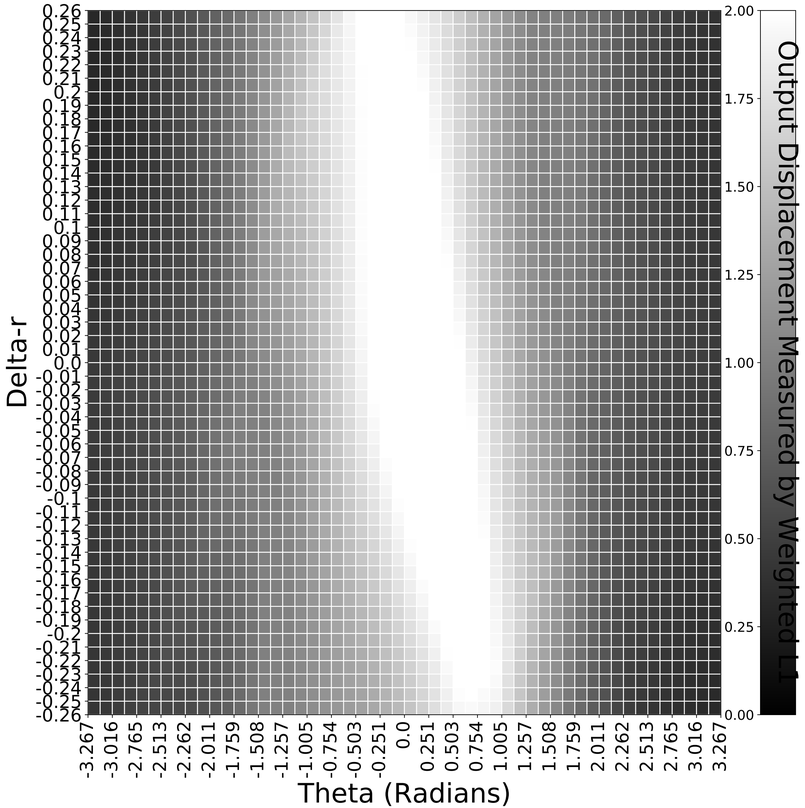


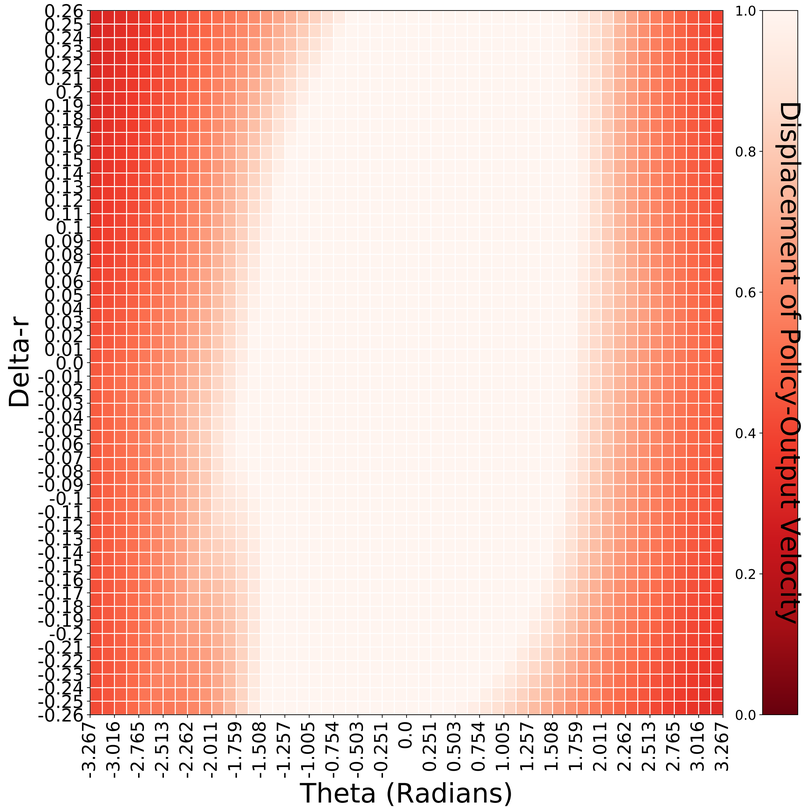
Circle-Following Min. Magnitude Output Results
deltaR-dot:0.115384, deltaTheta-dot:0, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 1bec6847-c758-45ee-bd61-a9f216c77272
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_41_51_515235

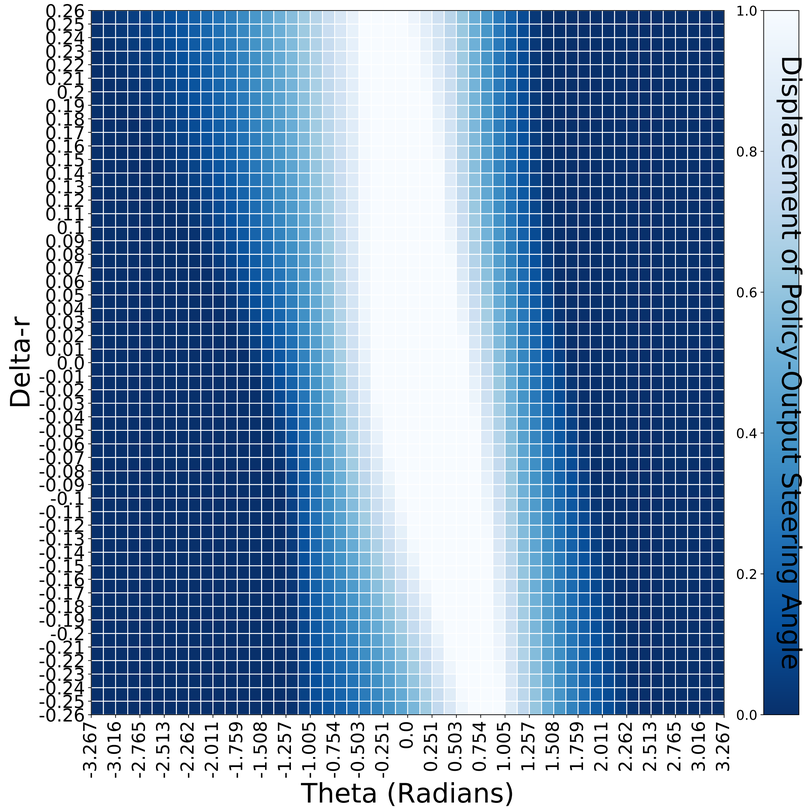




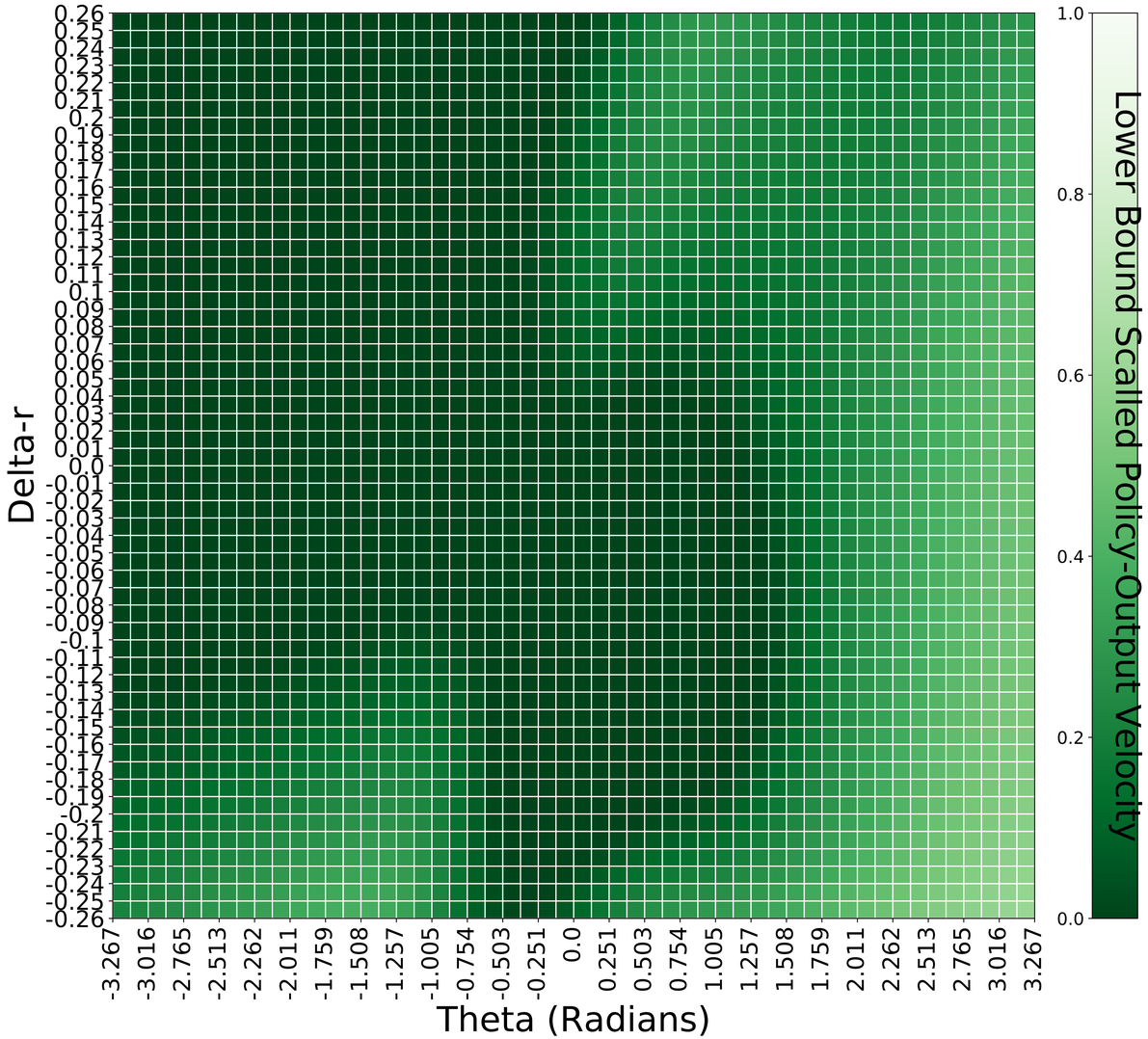
Circle-Following Stability Results
deltaR-dot:0.115384, deltaTheta-dot:0, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: dcffb755-df72-4d11-b450-724c24a01d35
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_39_40_145887

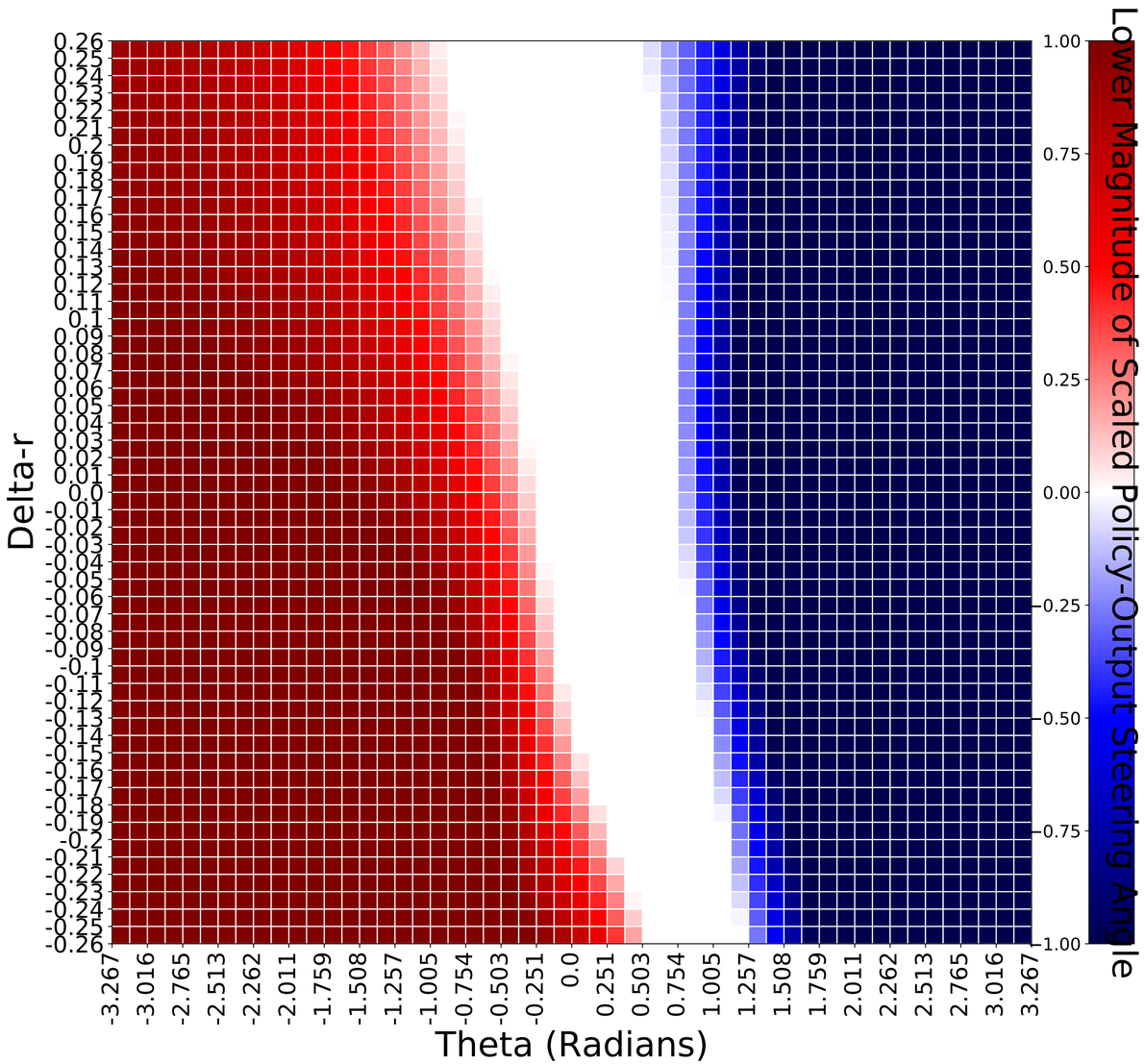




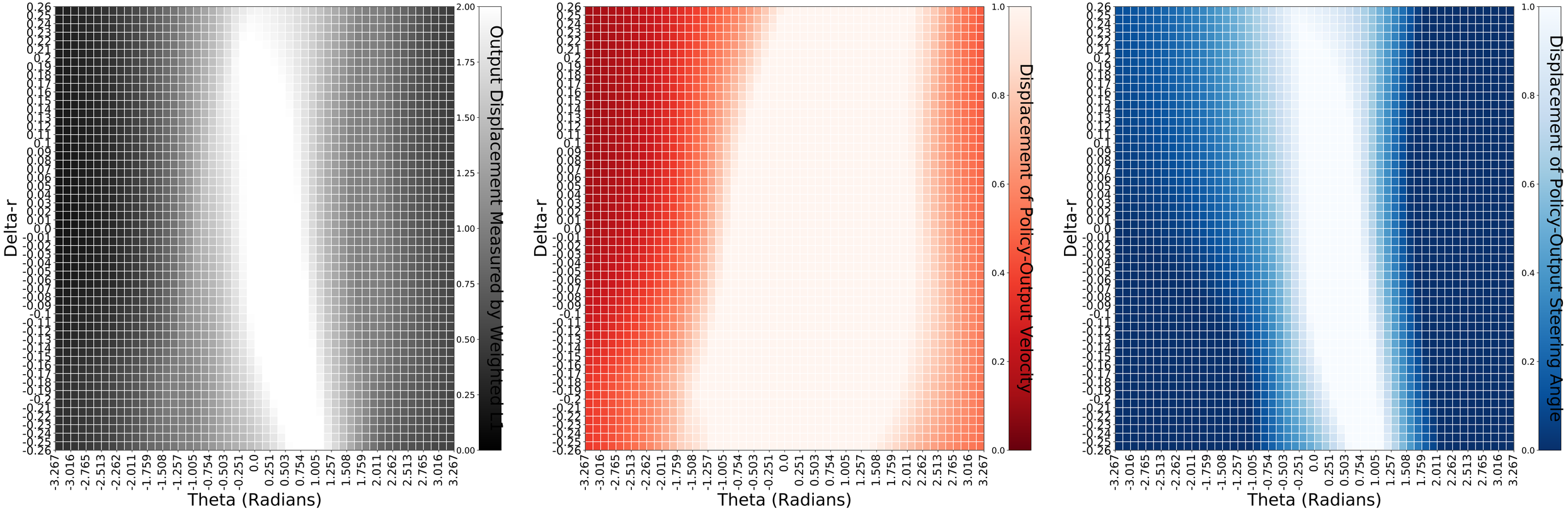


Circle-Following Min. Magnitude Output Results
deltaR-dot:-0.80769, deltaTheta-dot:0.278846, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 8385a772-0d02-4f22-8e5f-7c683e3a67a4
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_46_22_370331

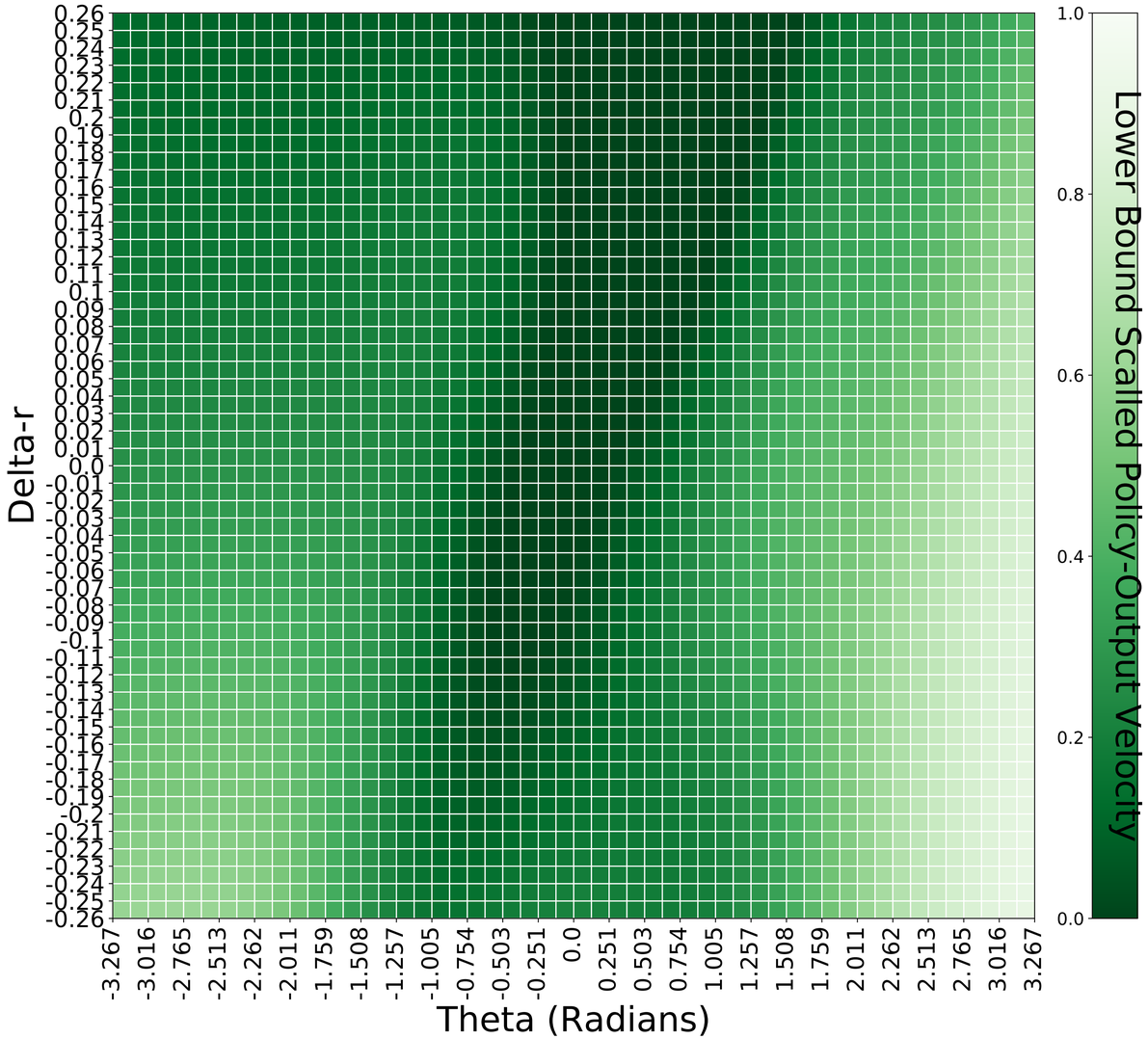


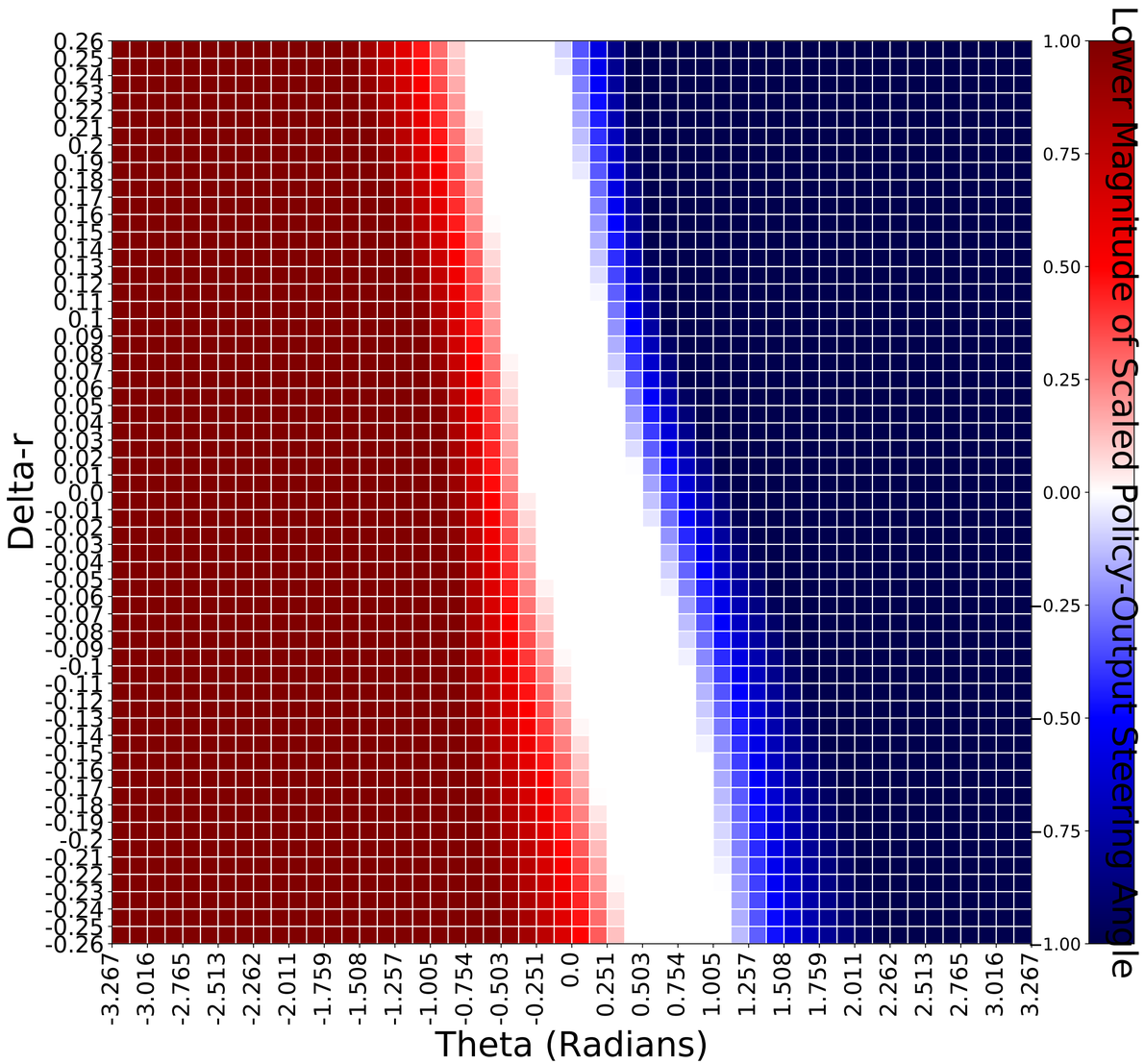


Circle-Following Stability Results
deltaR-dot:-0.80769, deltaTheta-dot:0.278846, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: b289a471-43ee-4b0f-8fc7-2a0f9a065f0c
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_44_05_647295

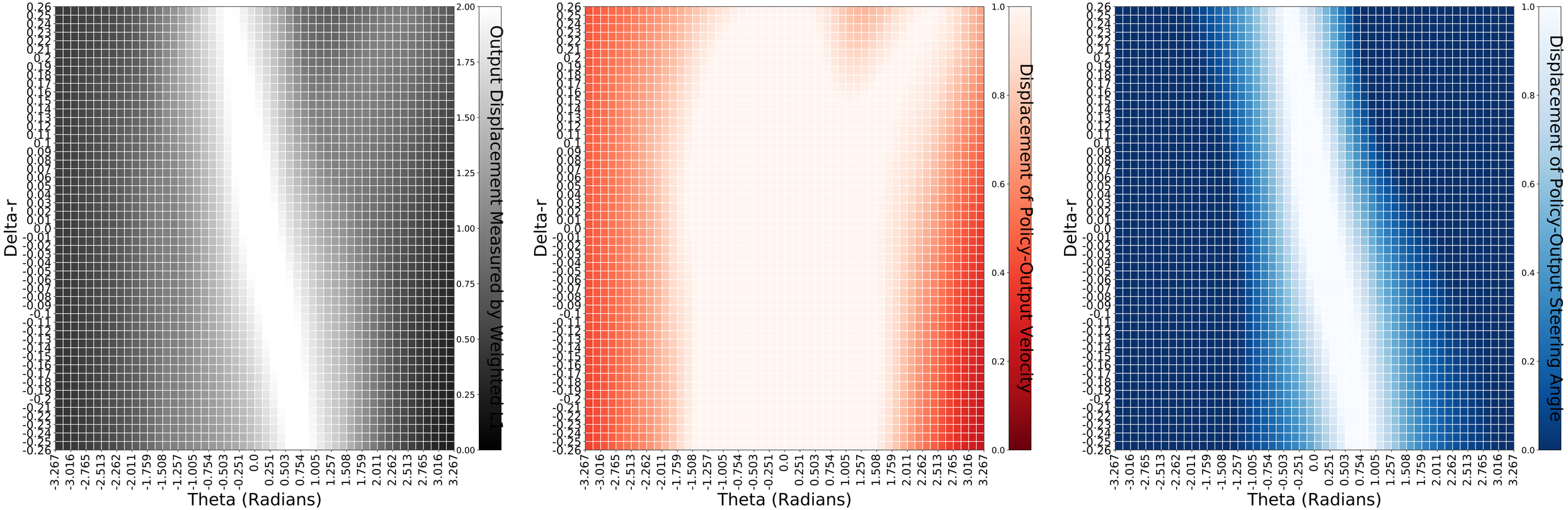


Circle-Following Min. Magnitude Output Results
deltaR-dot:-0.80769, deltaTheta-dot:-1.95192, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: dca69d35-6694-4184-8a1d-72f9494f2a49
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_50_47_462979

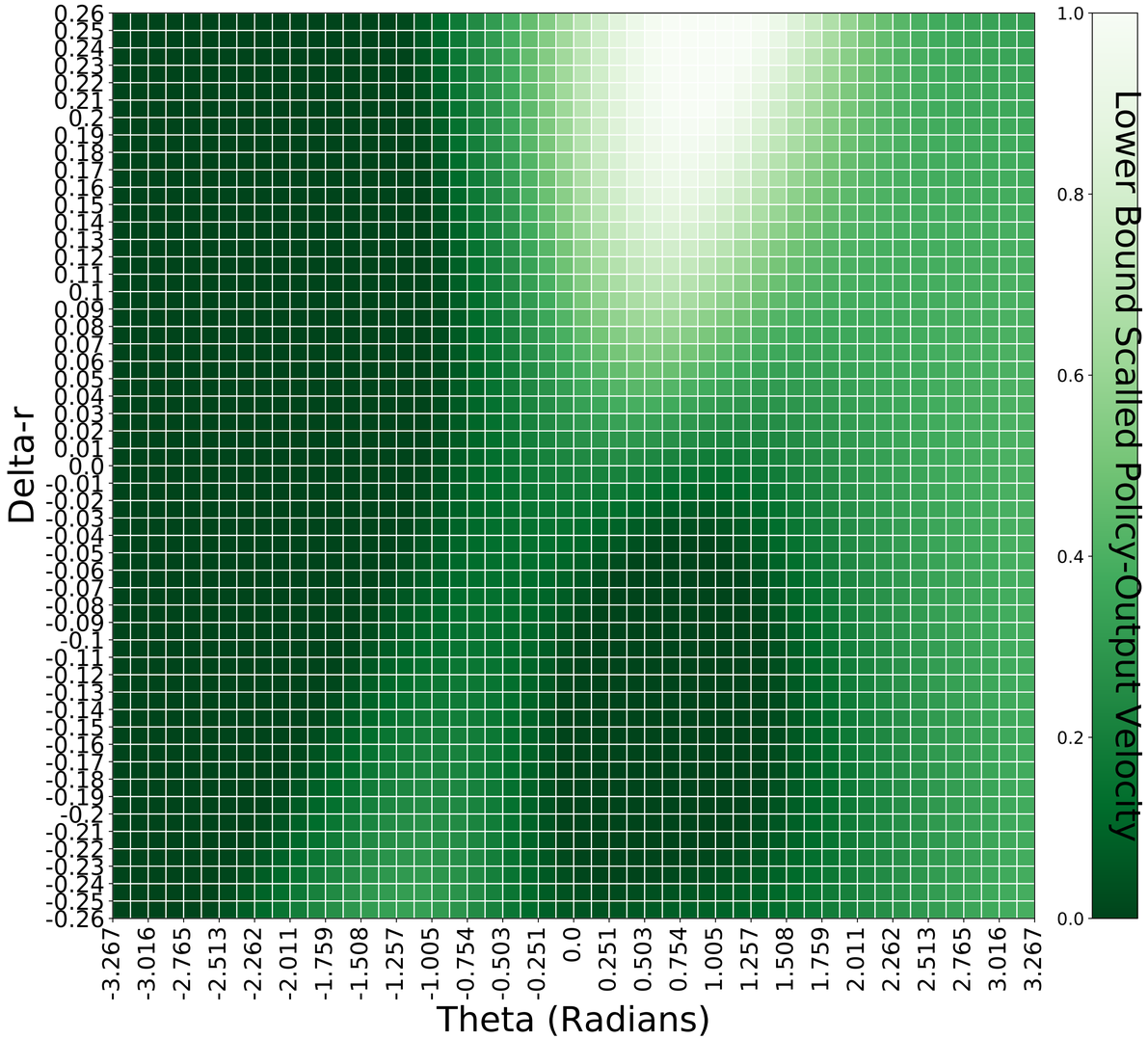


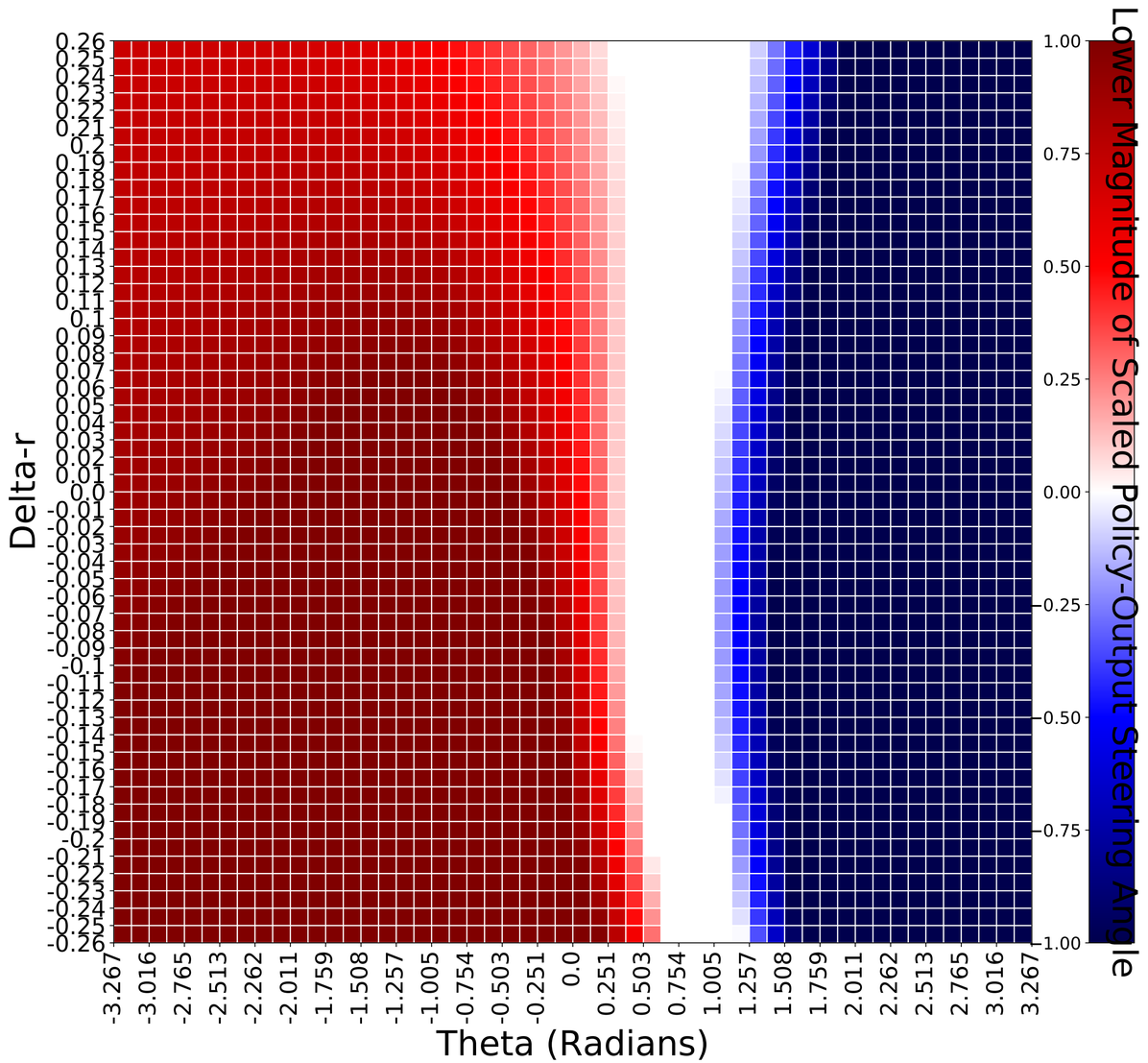


Circle-Following Stability Results
deltaR-dot:-0.80769, deltaTheta-dot:-1.95192, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 223352b5-96ca-4dc7-b868-ec7003d19474
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_48_34_460899

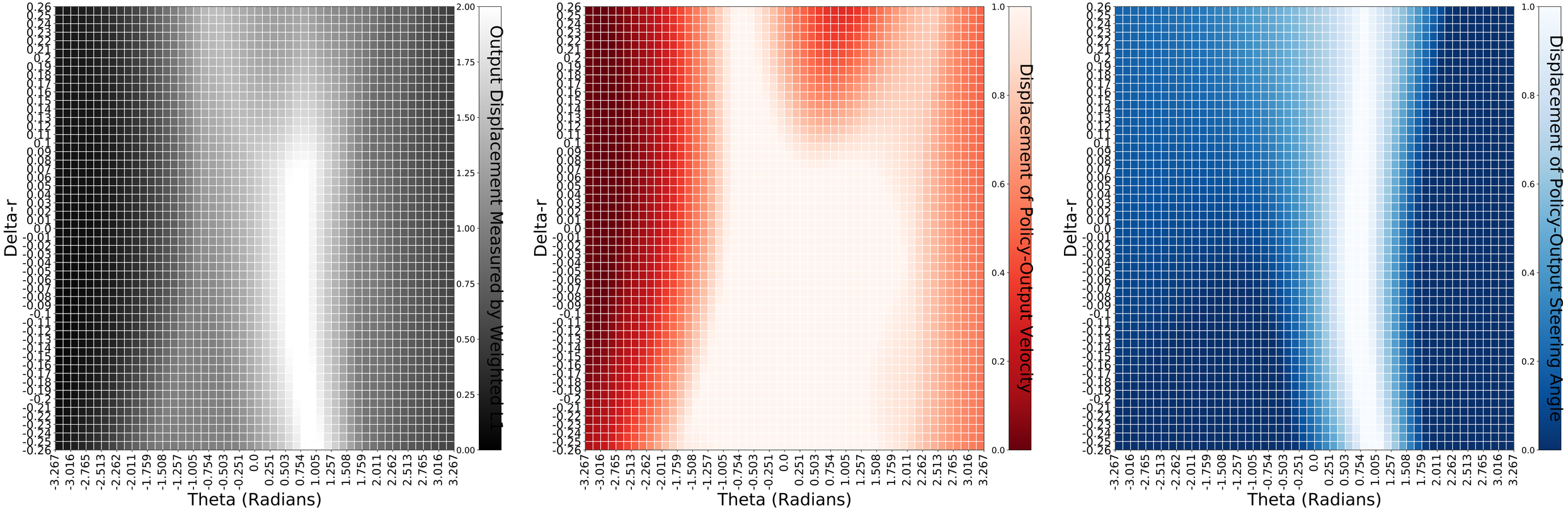


Circle-Following Min. Magnitude Output Results
deltaR-dot:-0.80769, deltaTheta-dot:2.509615, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: e83d3f9c-ef13-4e4e-ad4d-89d240ac066d
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_55_10_246214

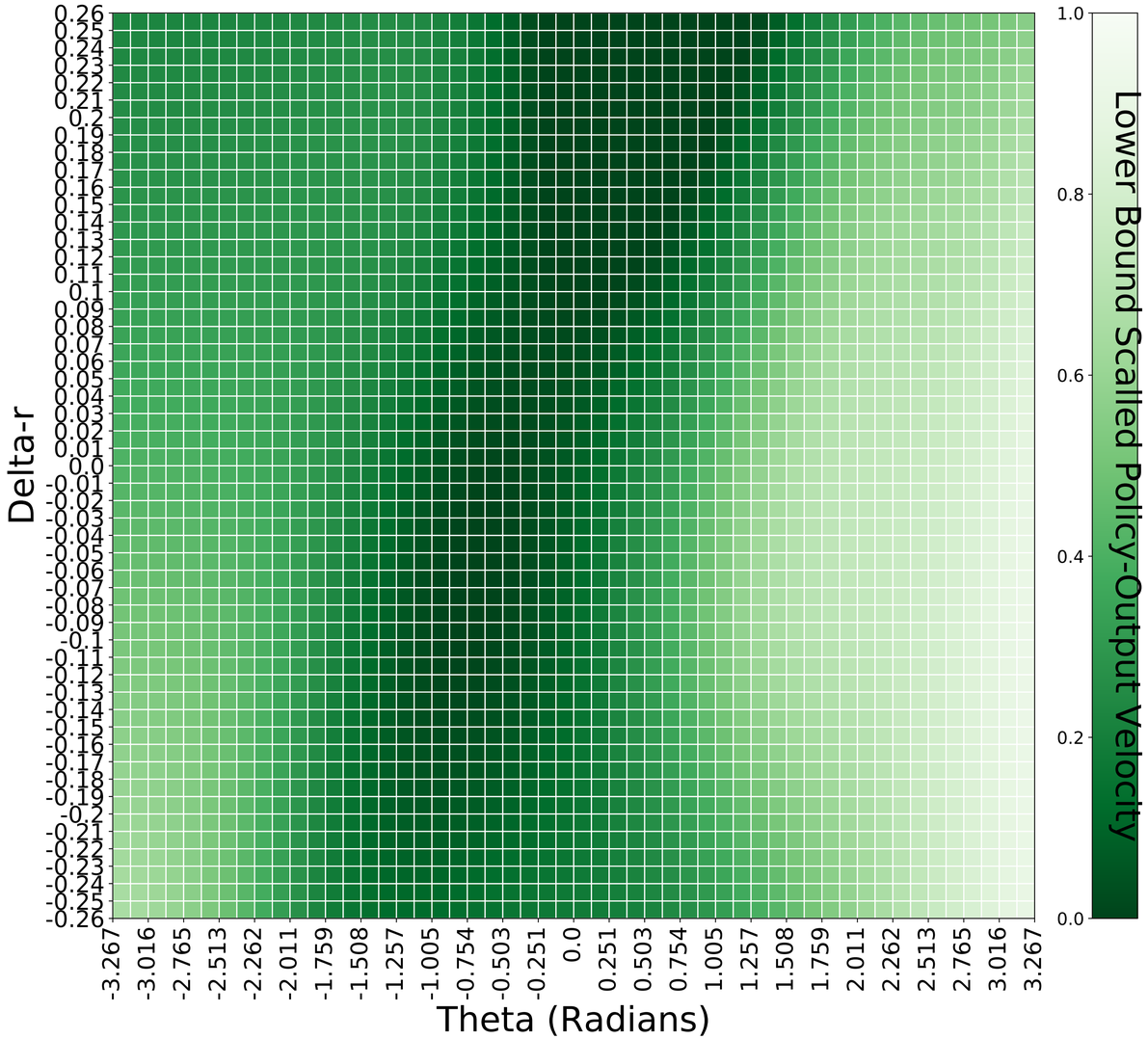


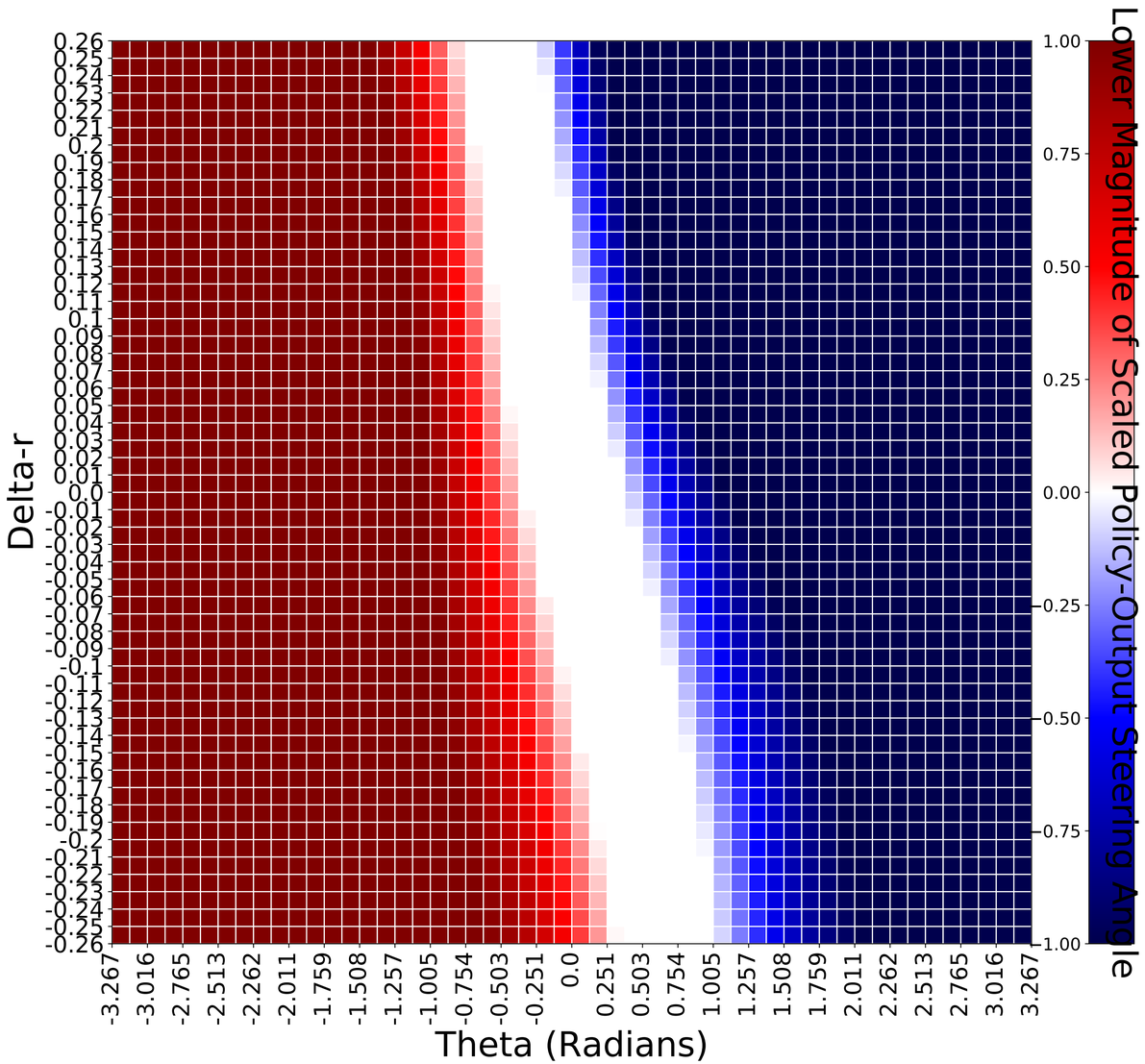


Circle-Following Stability Results
deltaR-dot:-0.80769, deltaTheta-dot:2.509615, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 5bfbcd41-c6c7-4d06-bf00-562bd96eab5c
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_52_57_249384

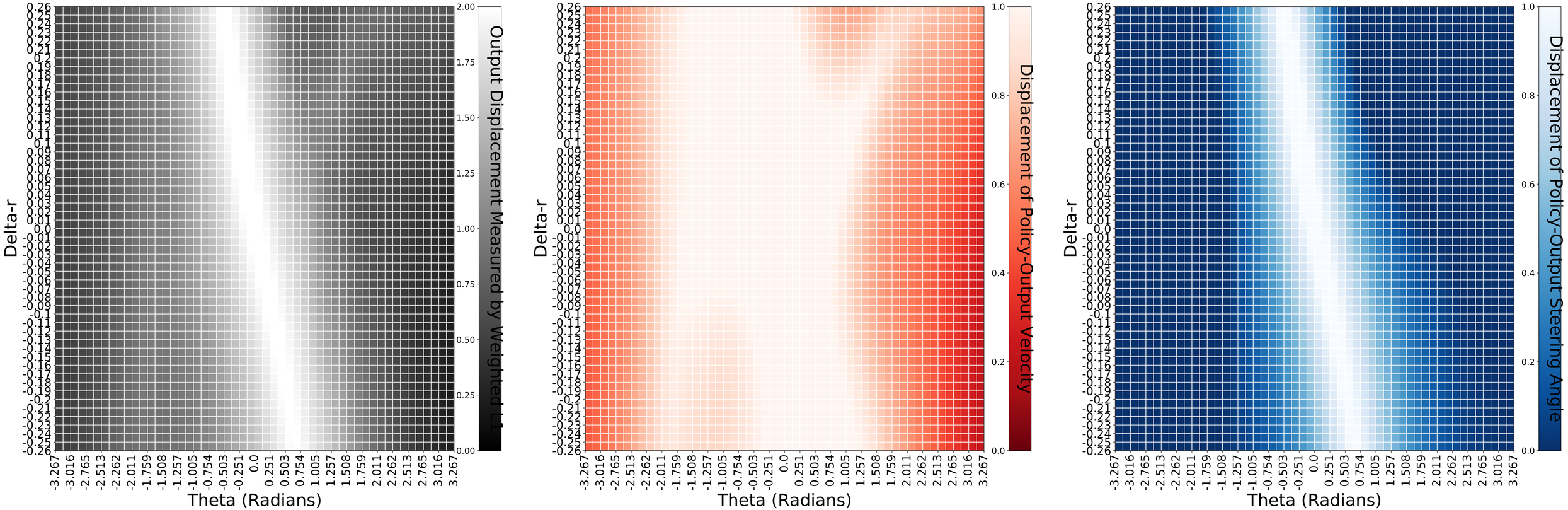


Circle-Following Min. Magnitude Output Results
deltaR-dot:-0.80769, deltaTheta-dot:-3.06730, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 6b2e1728-777e-4970-aee7-d87a5ba573c6
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_59_32_392614

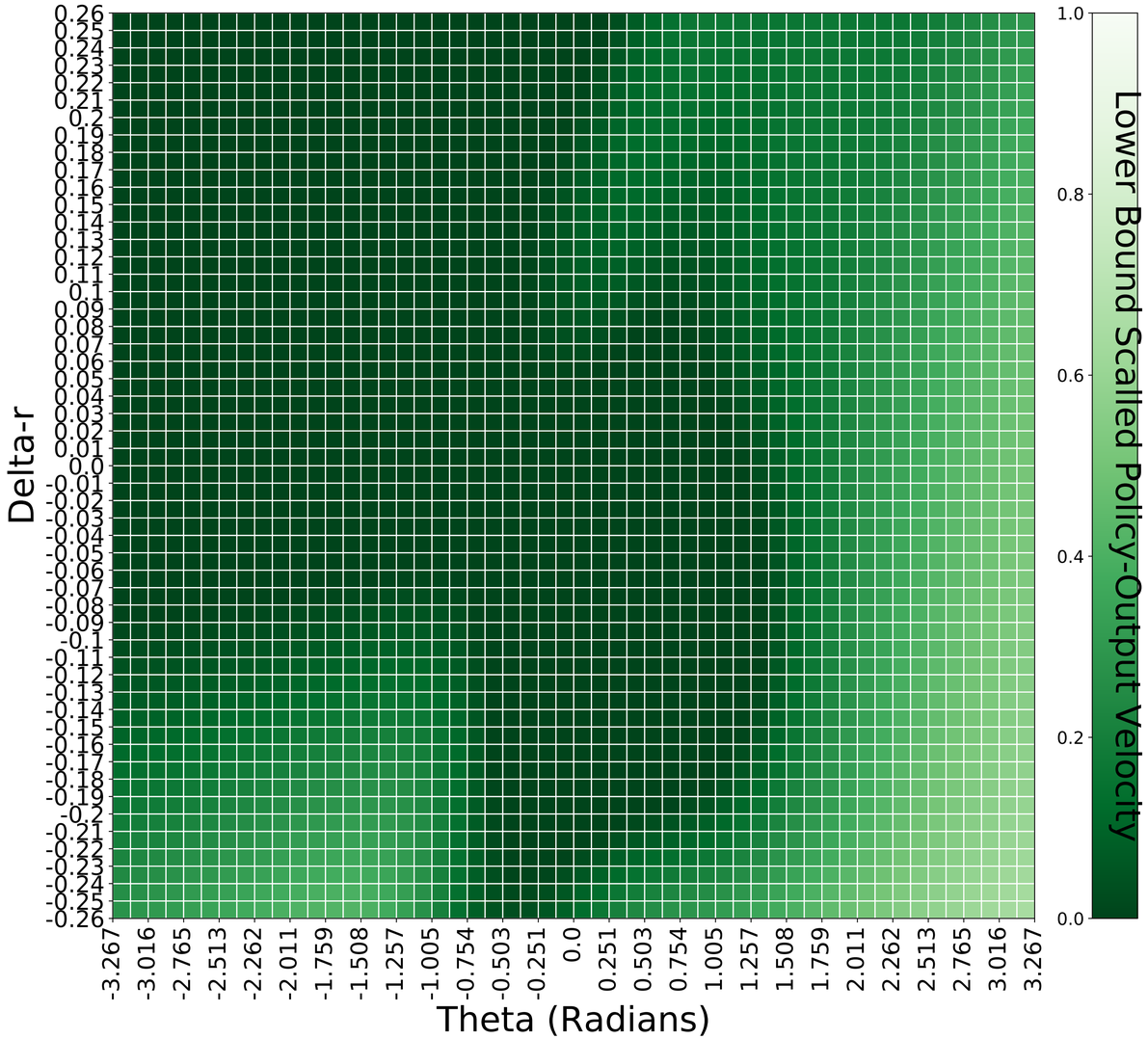


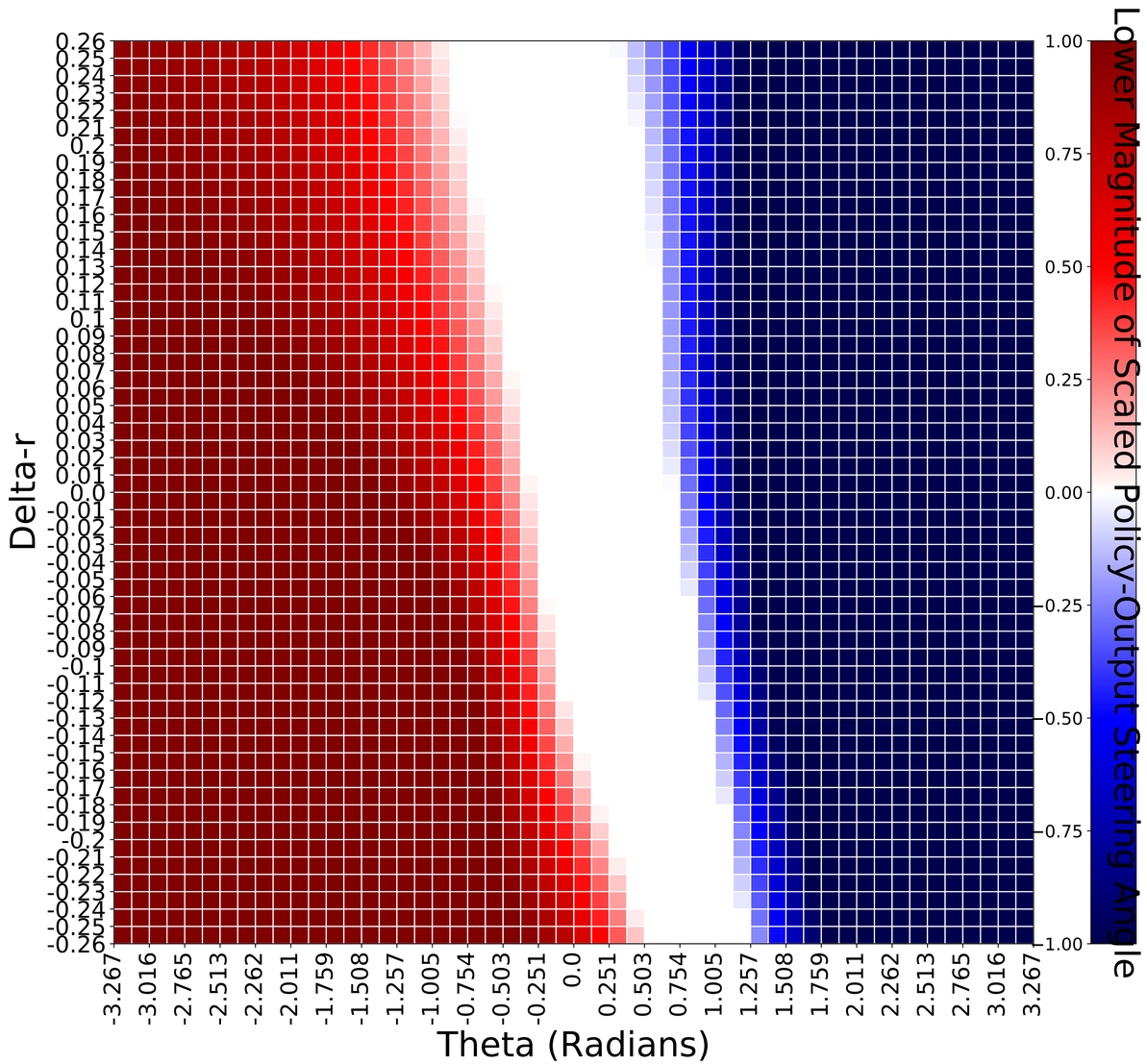


Circle-Following Stability Results
deltaR-dot:-0.80769, deltaTheta-dot:-3.06730, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 5e55e827-cd26-47b3-877b-16b63dabcad2
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_07_23_57_20_960957

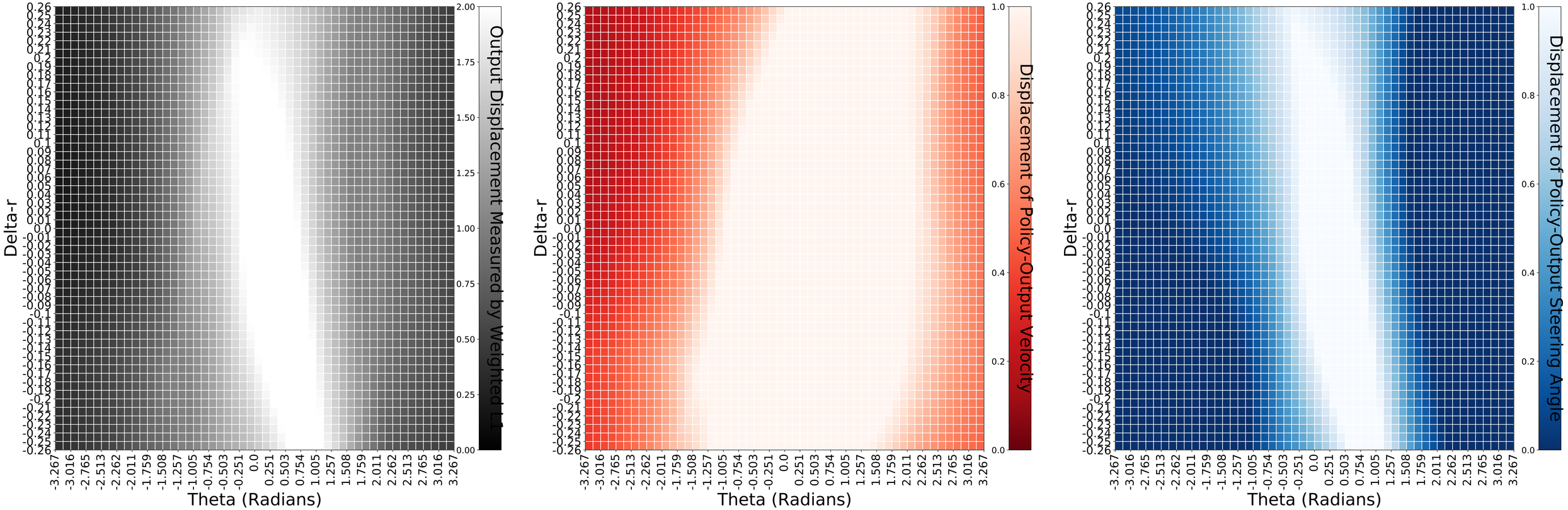


Circle-Following Min. Magnitude Output Results
deltaR-dot:-0.80769, deltaTheta-dot:0, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 4f77f23a-5f80-4e78-8cf6-43039d676985
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_00_30_21_659332

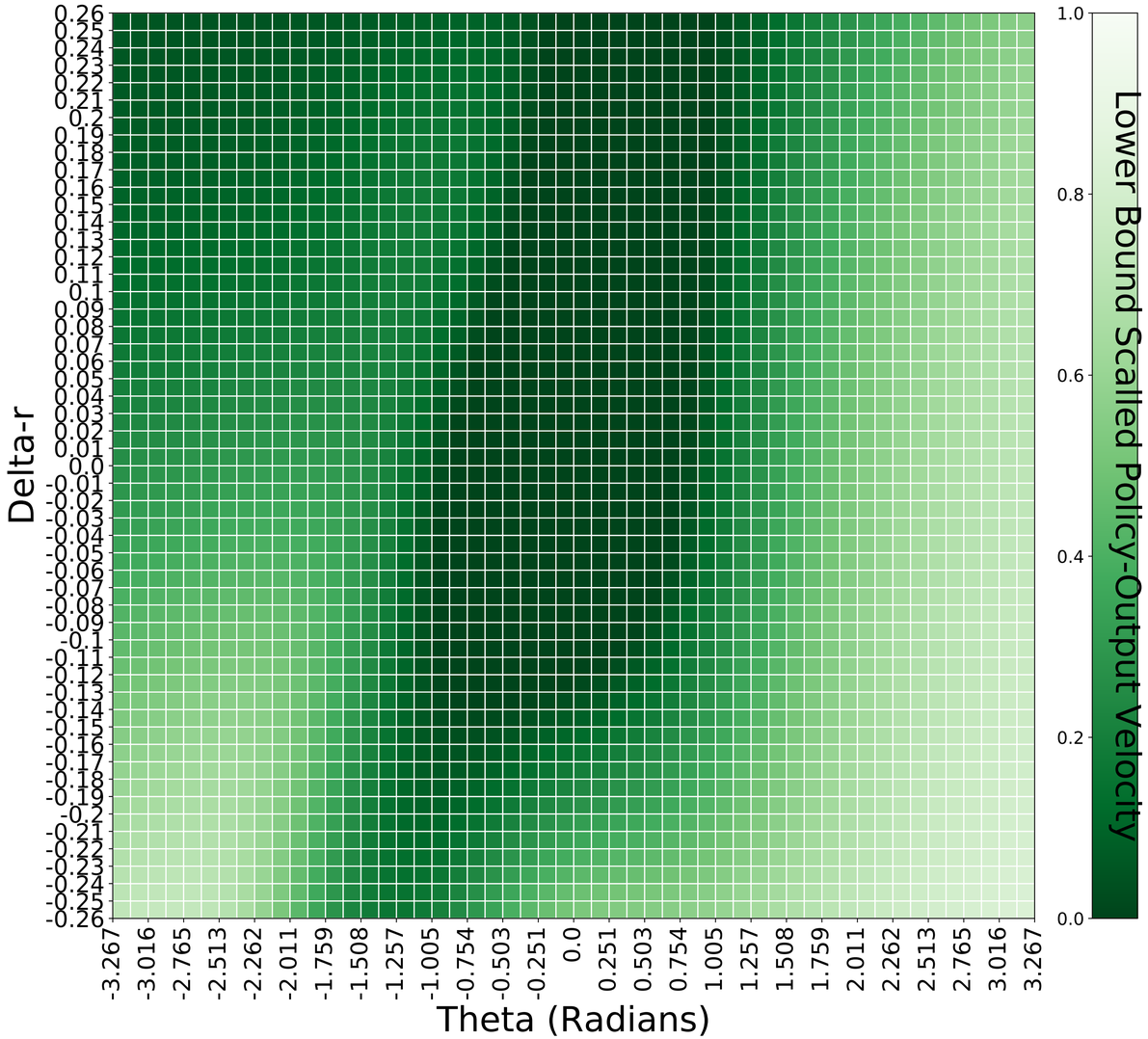


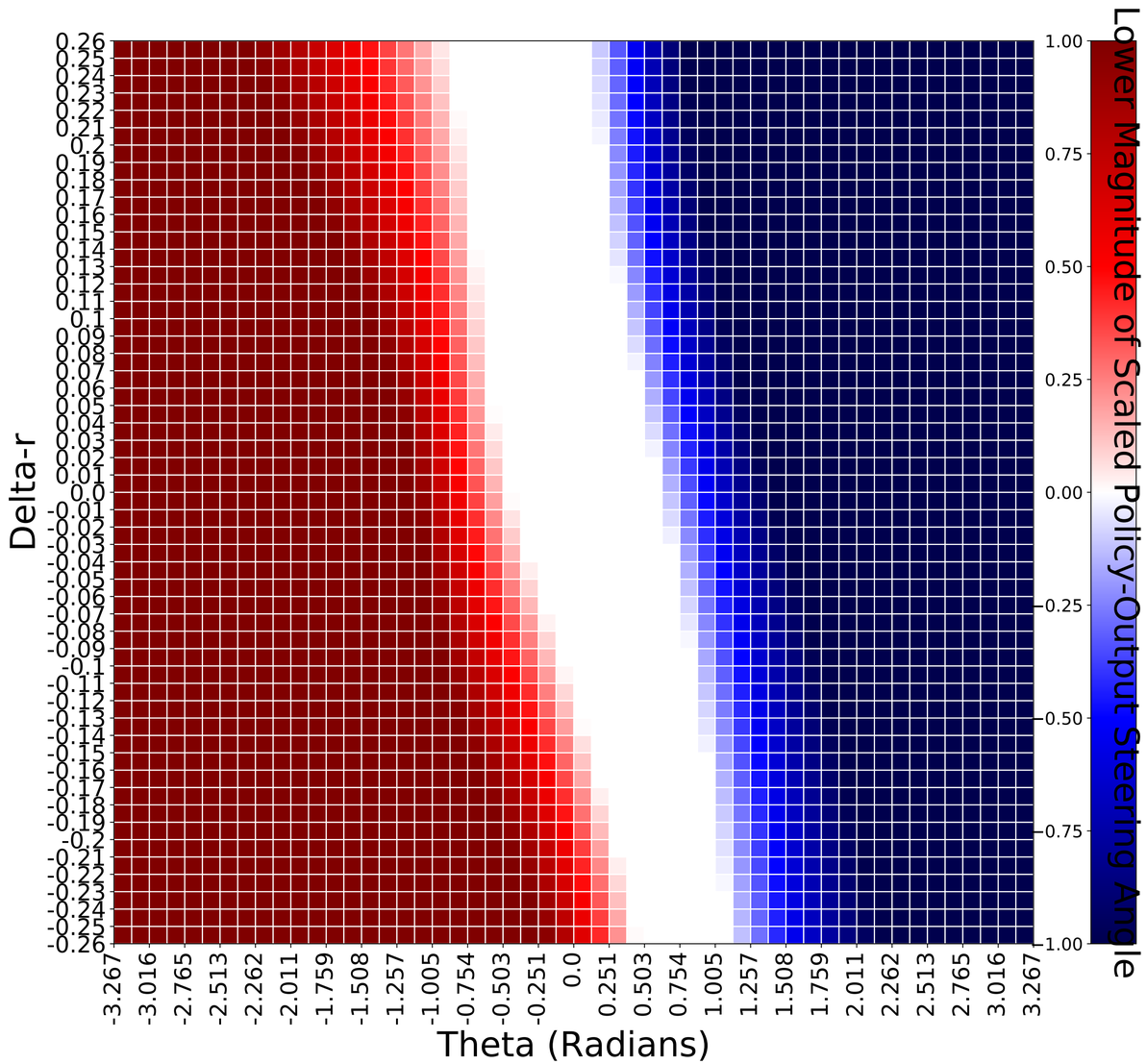


Circle-Following Stability Results
deltaR-dot:-0.80769, deltaTheta-dot:0, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 166d284f-6f00-47da-9cd0-d4d766a70122
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_00_28_10_989085

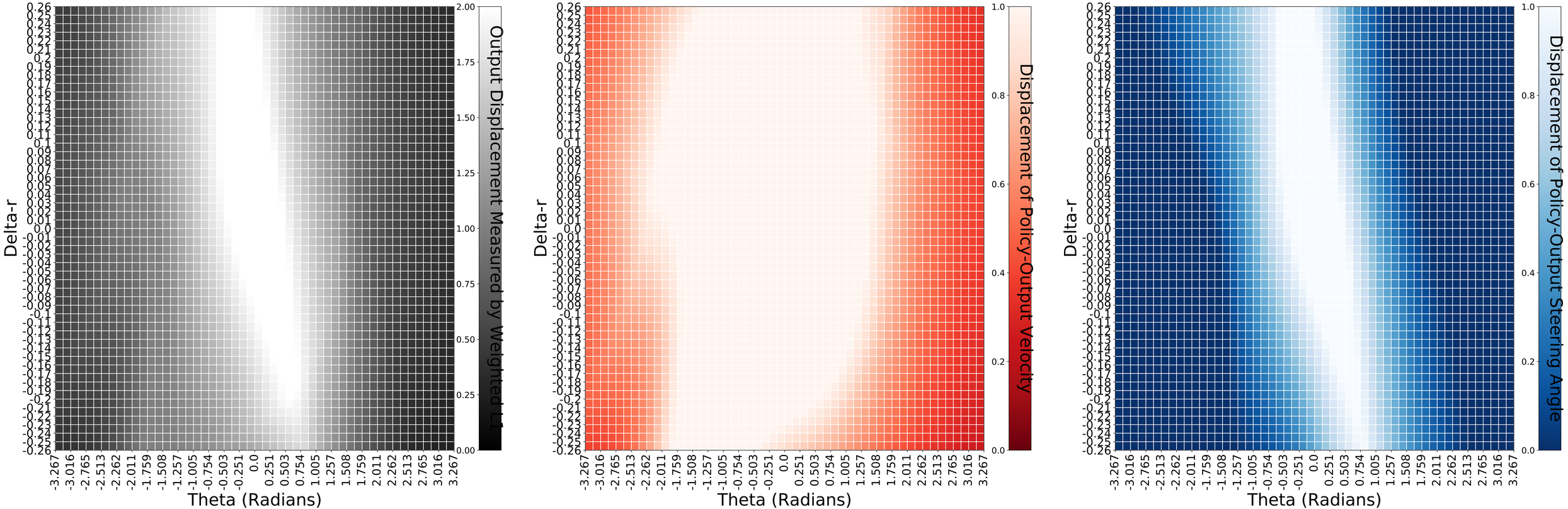


Circle-Following Min. Magnitude Output Results
deltaR-dot:1.038461, deltaTheta-dot:0.278846, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: f83bc70c-9357-454e-ae24-36b1d42f3528
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_00_34_39_489823

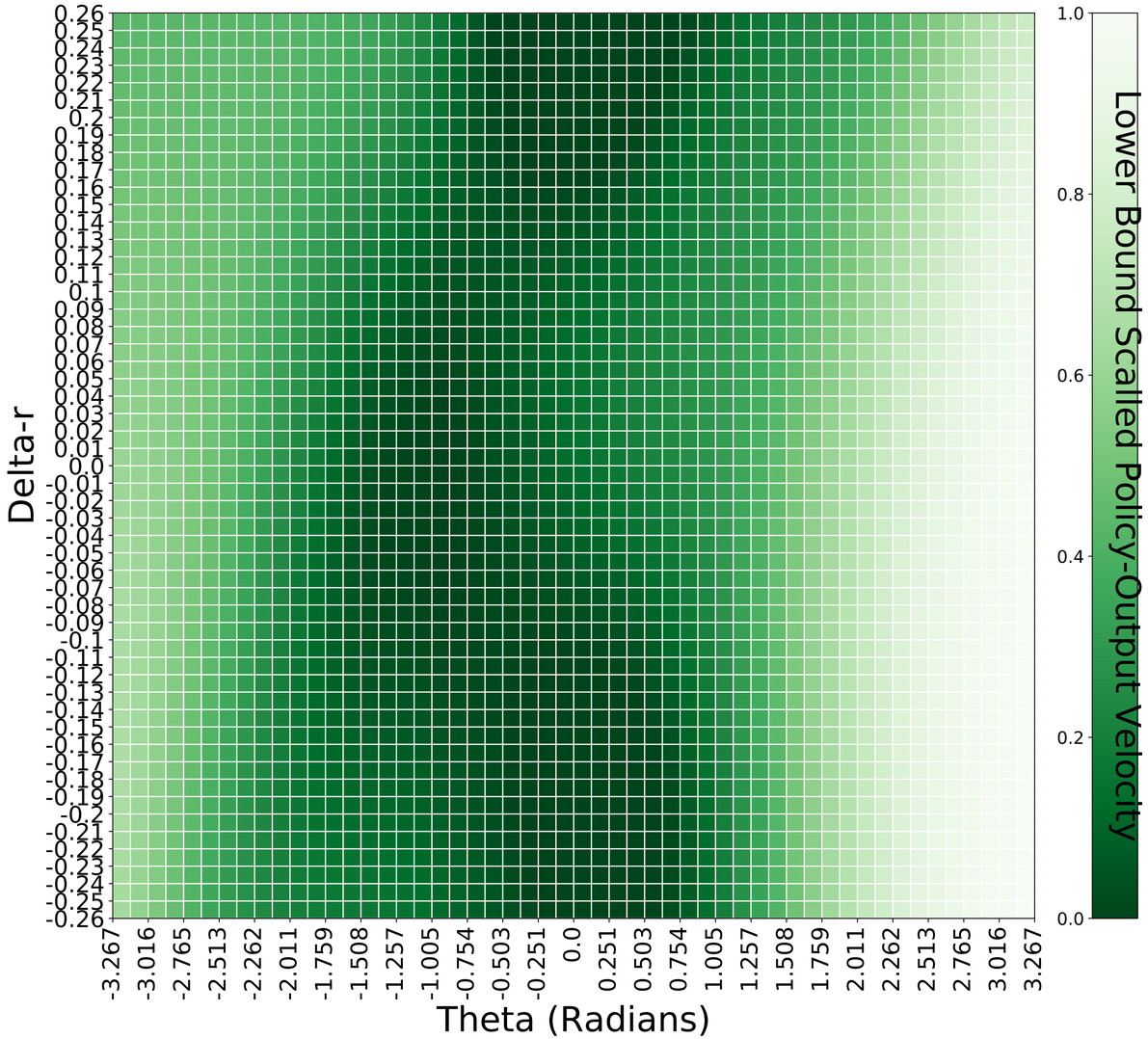


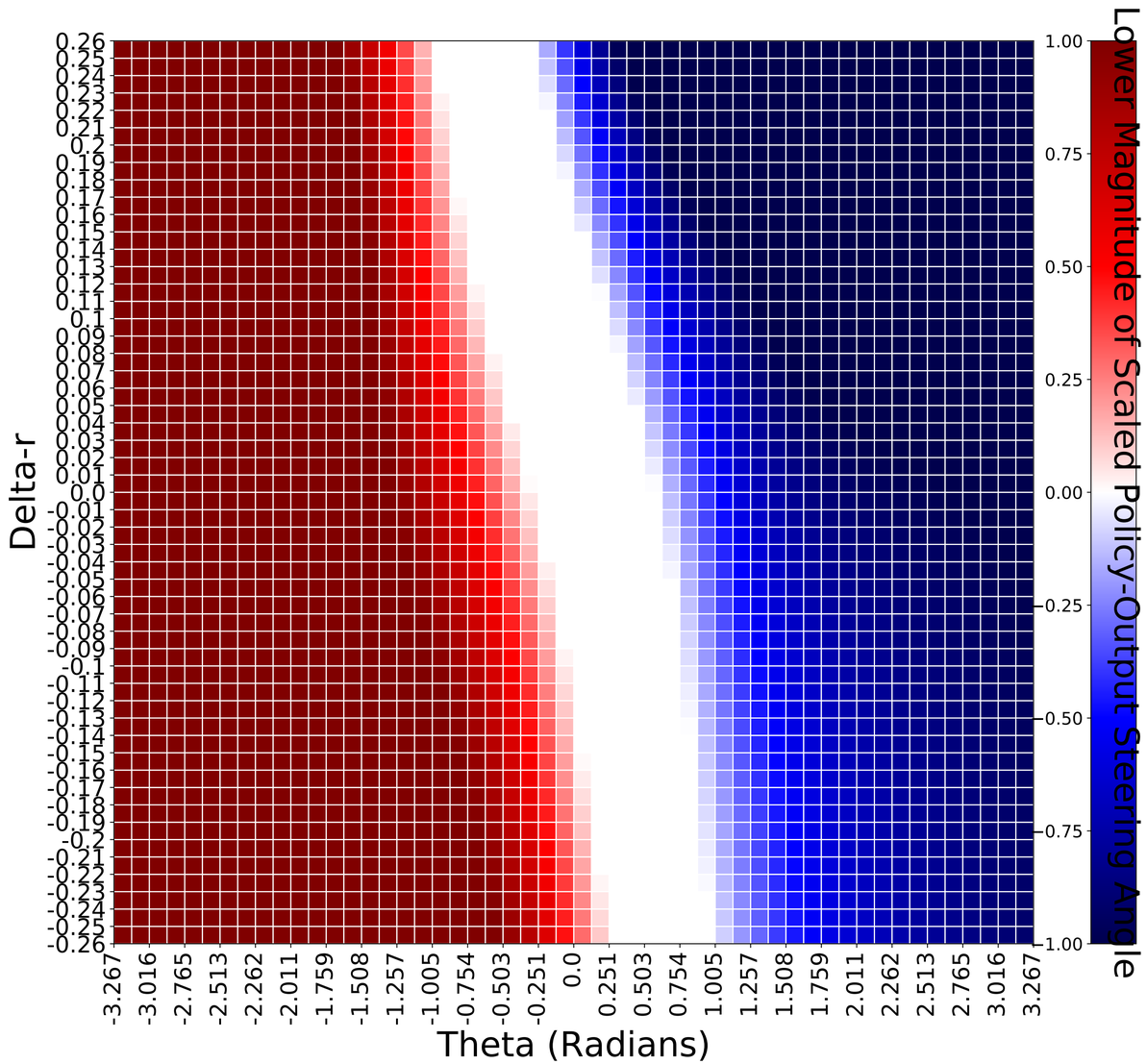


Circle-Following Stability Results
deltaR-dot:1.038461, deltaTheta-dot:0.278846, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 90eec2d6-8ccd-483a-81c0-fb3b2324a062
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_00_32_29_524901

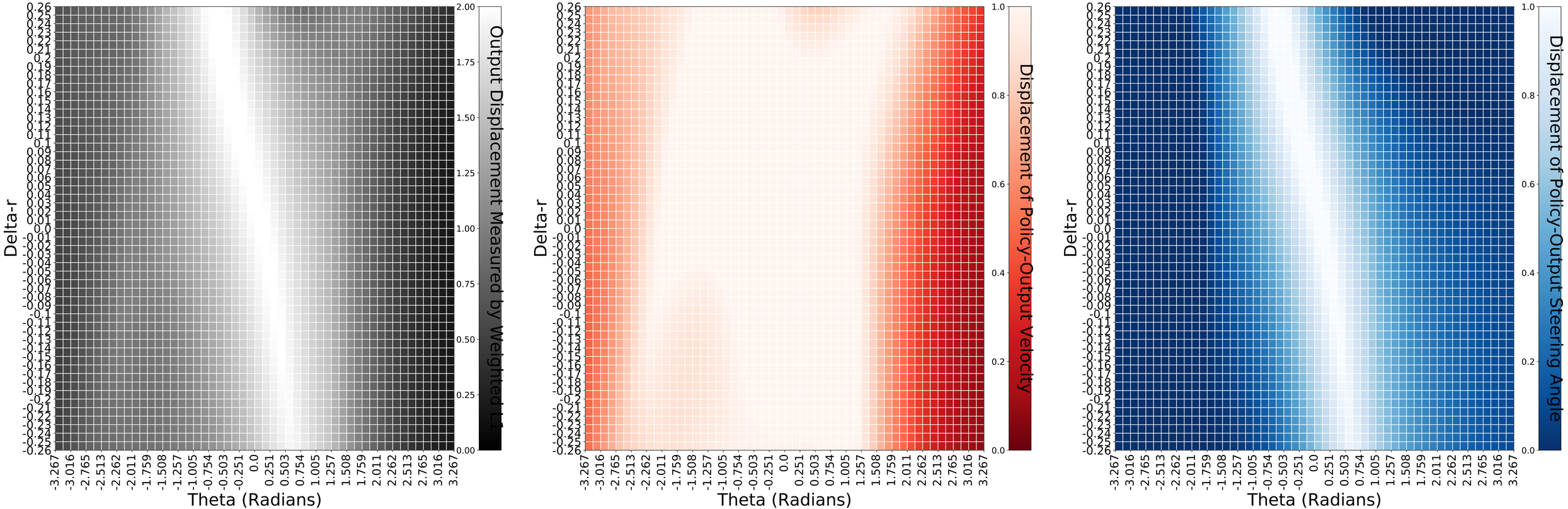


Circle-Following Min. Magnitude Output Results
deltaR-dot:1.038461, deltaTheta-dot:-1.95192, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 0cba550a-8ef2-4b45-99b0-dc0cfdc8025c
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_00_38_59_410556

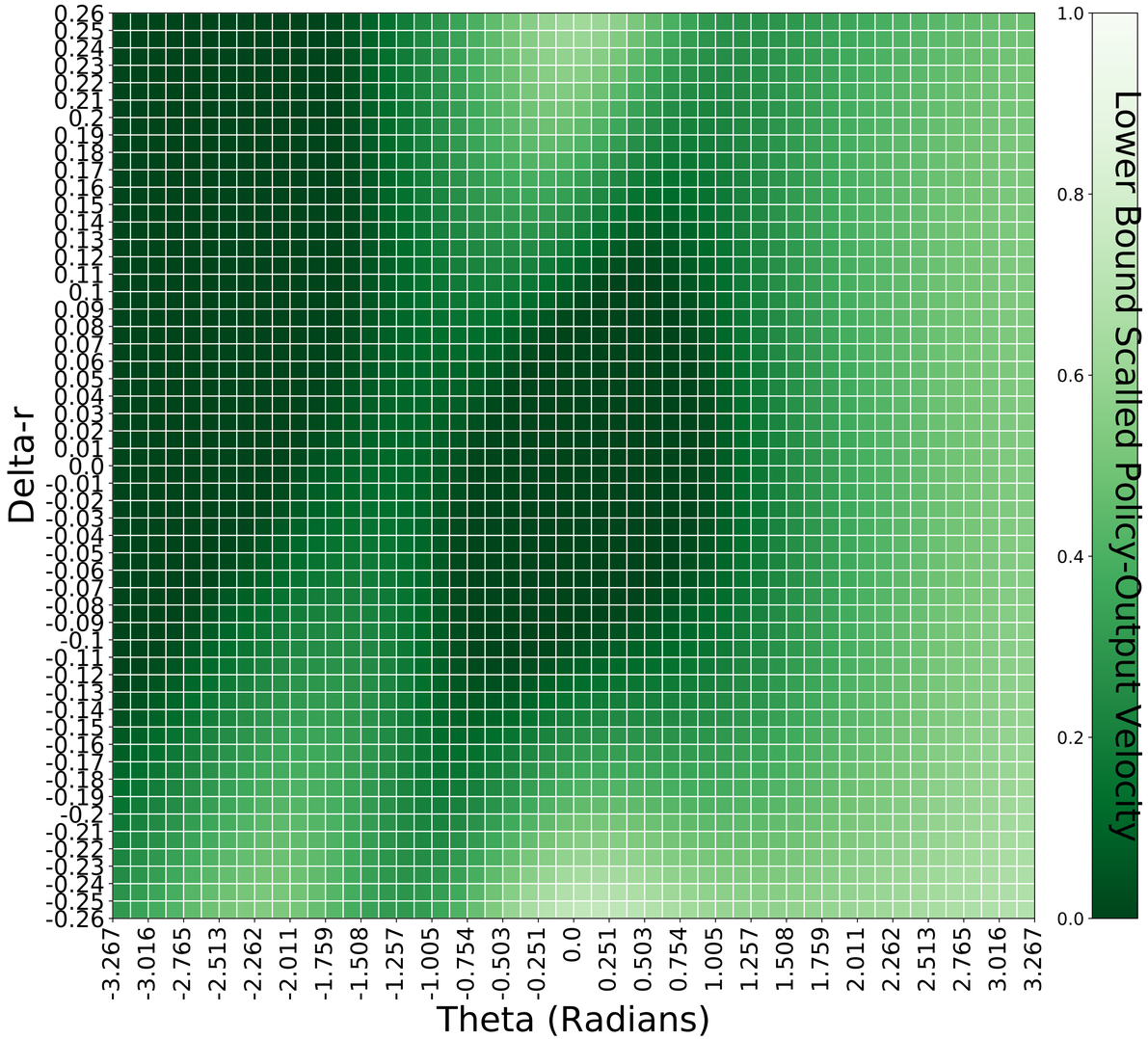


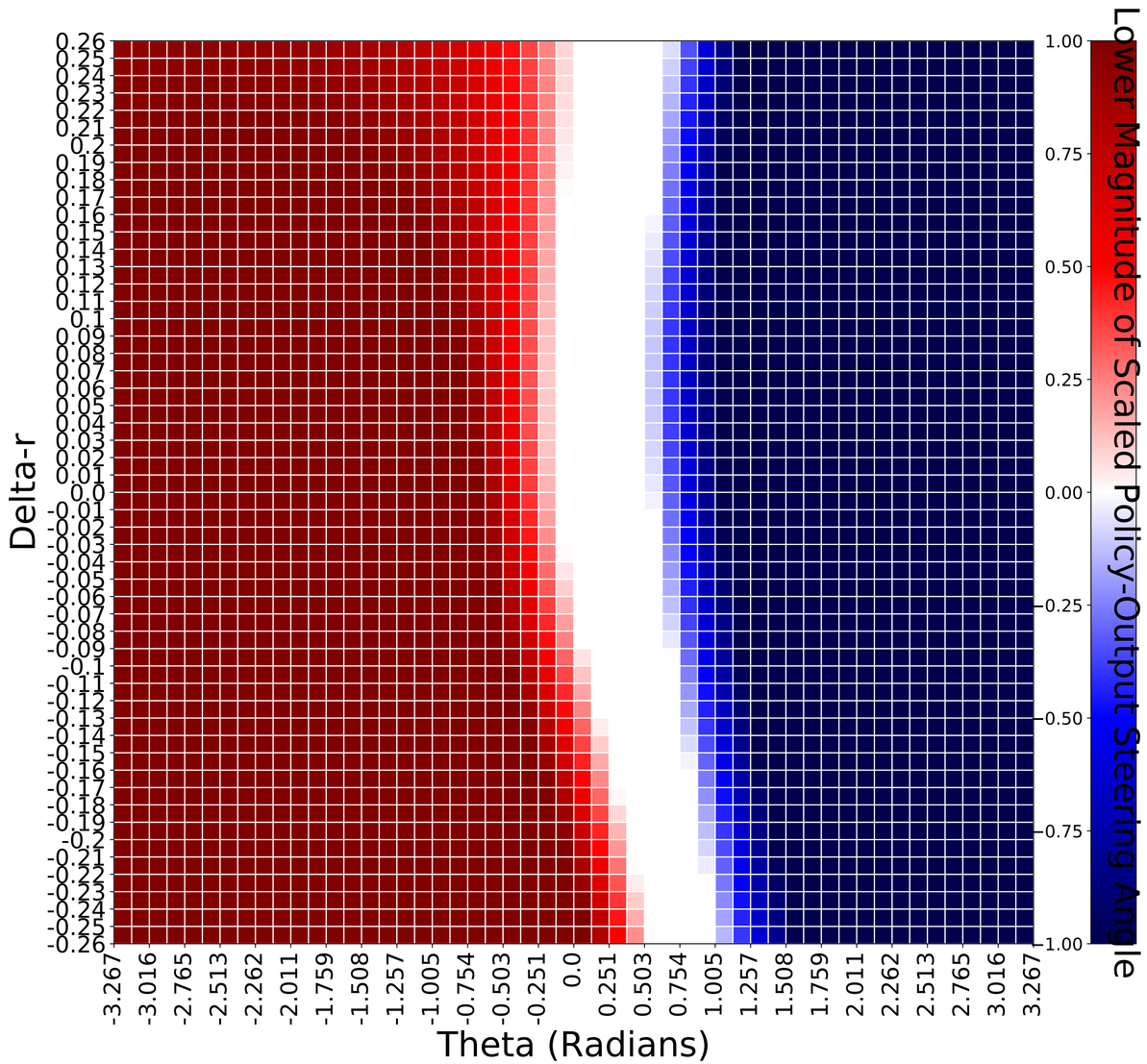


Circle-Following Stability Results
deltaR-dot:1.038461, deltaTheta-dot:-1.95192, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: a5a4eb57-e50d-4885-a471-5f8c9499add6
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_00_36_48_156732

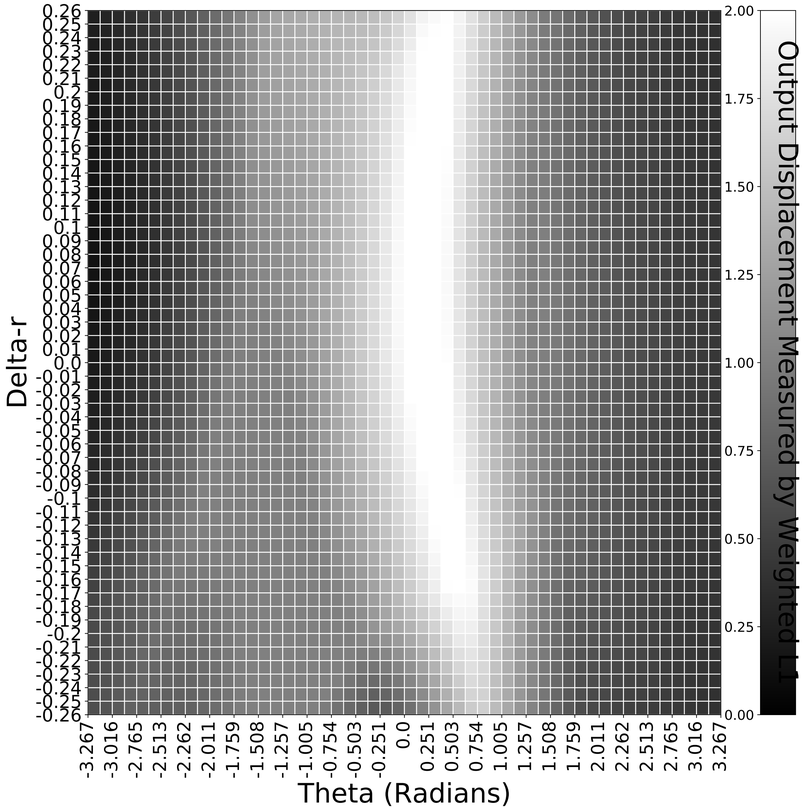


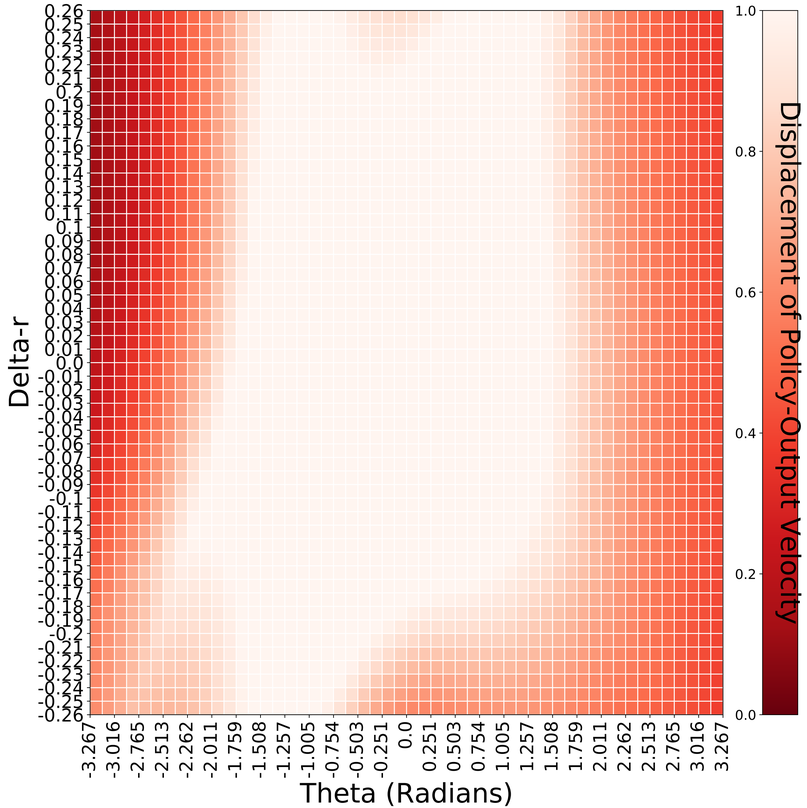
Circle-Following Min. Magnitude Output Results
deltaR-dot:1.038461, deltaTheta-dot:2.509615, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: cdda06f7-1fc0-45fb-bf8b-d2cd57979ab2
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_00_43_19_989309

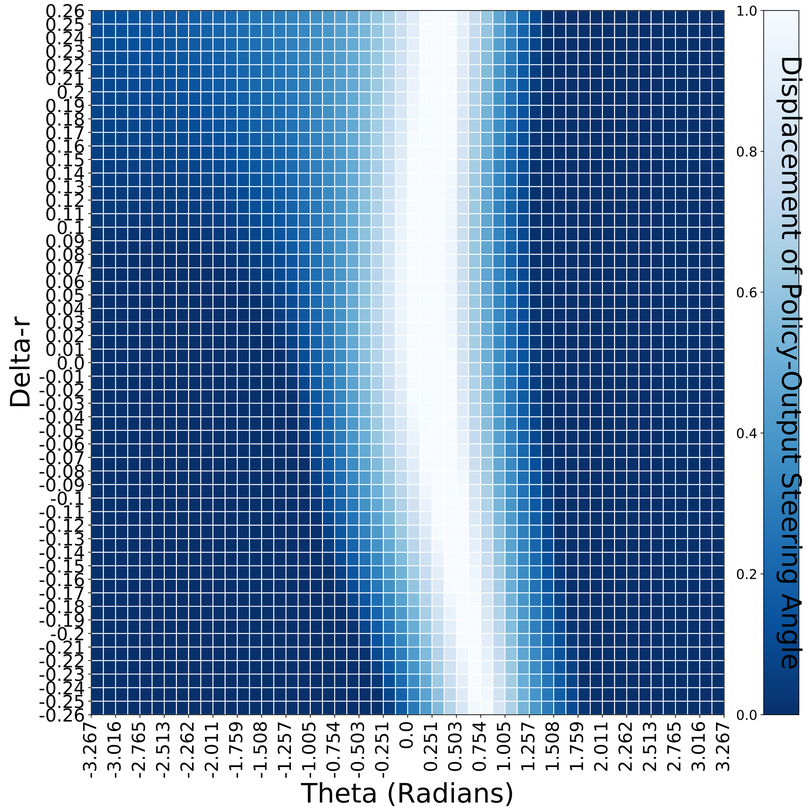




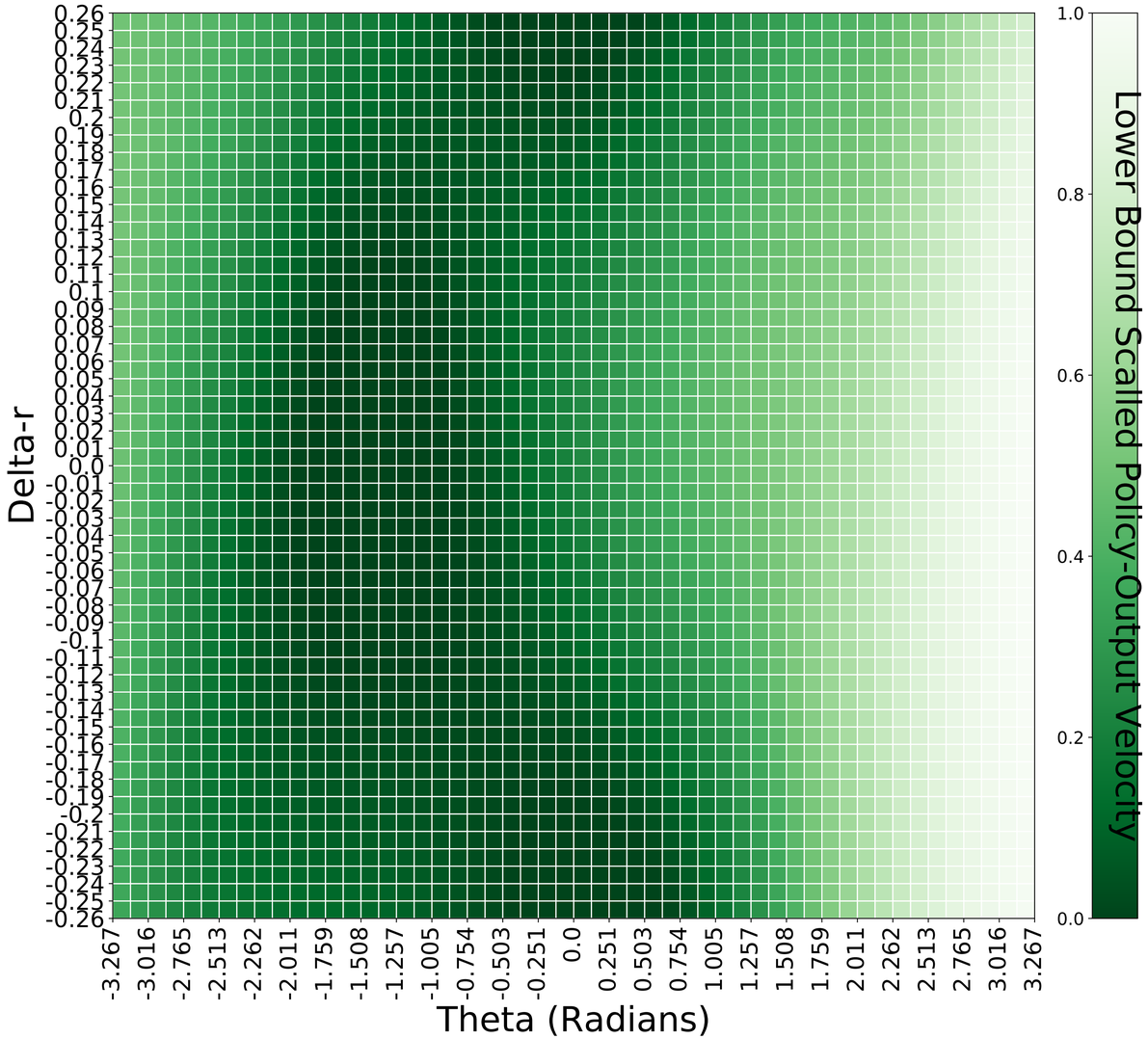
Circle-Following Stability Results
deltaR-dot:1.038461, deltaTheta-dot:2.509615, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: dd1ece31-3a0b-434f-aa32-2f20de2eff28
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_00_41_08_313737

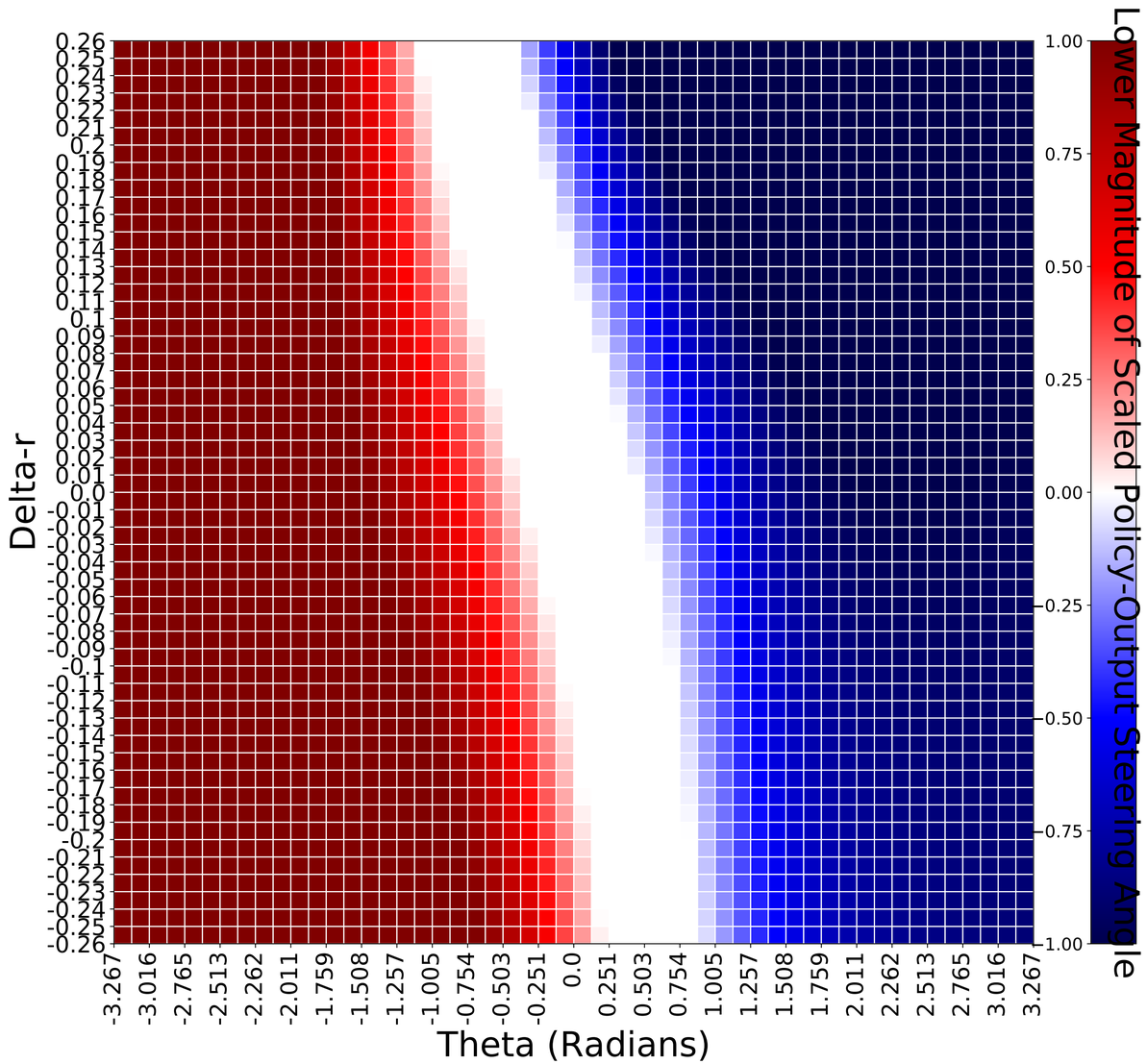




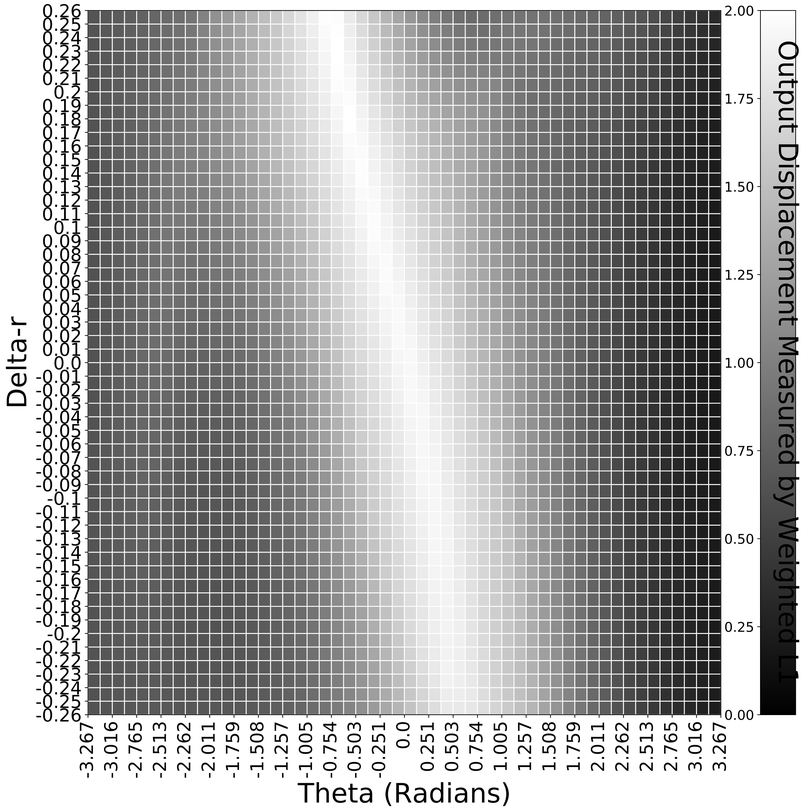


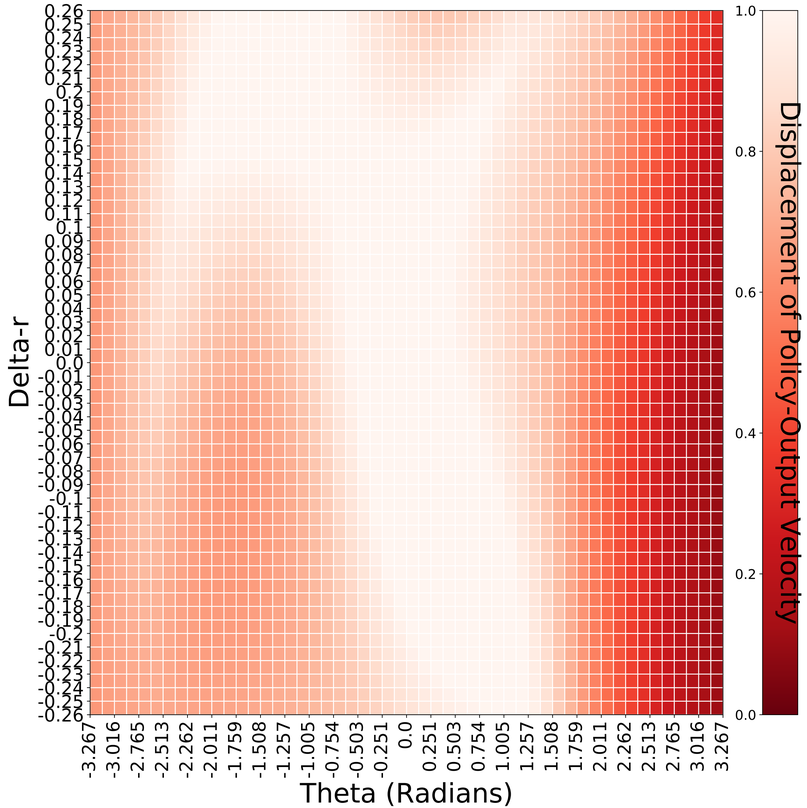
Circle-Following Min. Magnitude Output Results
deltaR-dot:1.038461, deltaTheta-dot:-3.06730, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 87fc4d97-6ee8-4b0c-a18a-be9268e77947
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_00_47_38_705821

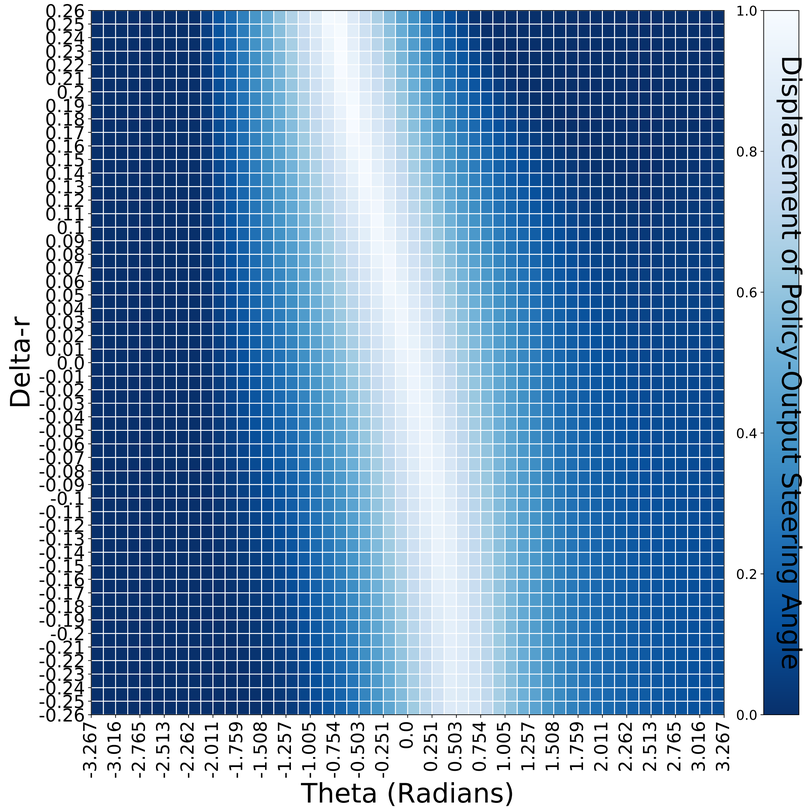




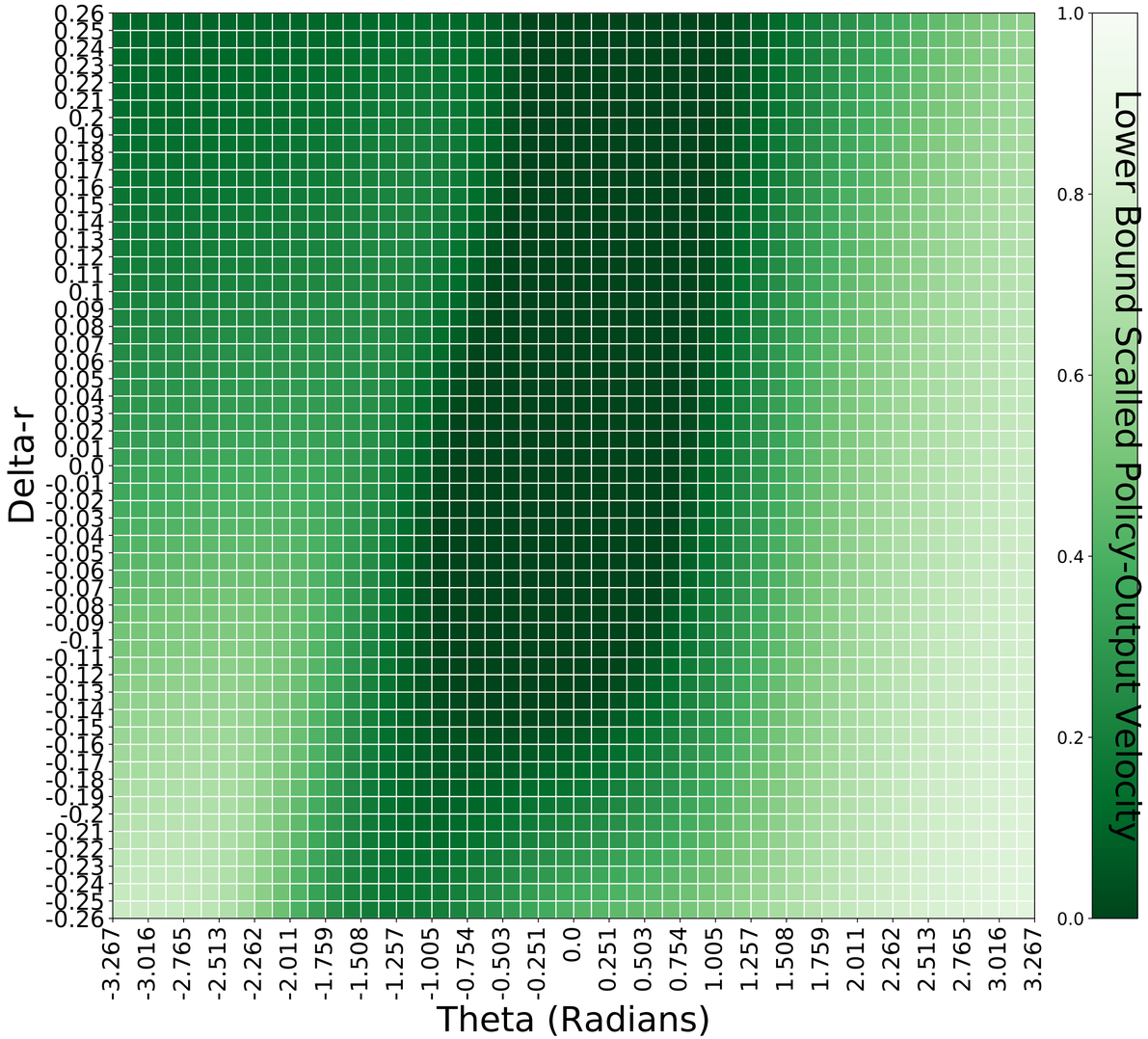
Circle-Following Stability Results
deltaR-dot:1.038461, deltaTheta-dot:-3.06730, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: e3059b97-1fe2-4d67-a2c5-054eefdfbbbe
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_00_45_28_587419

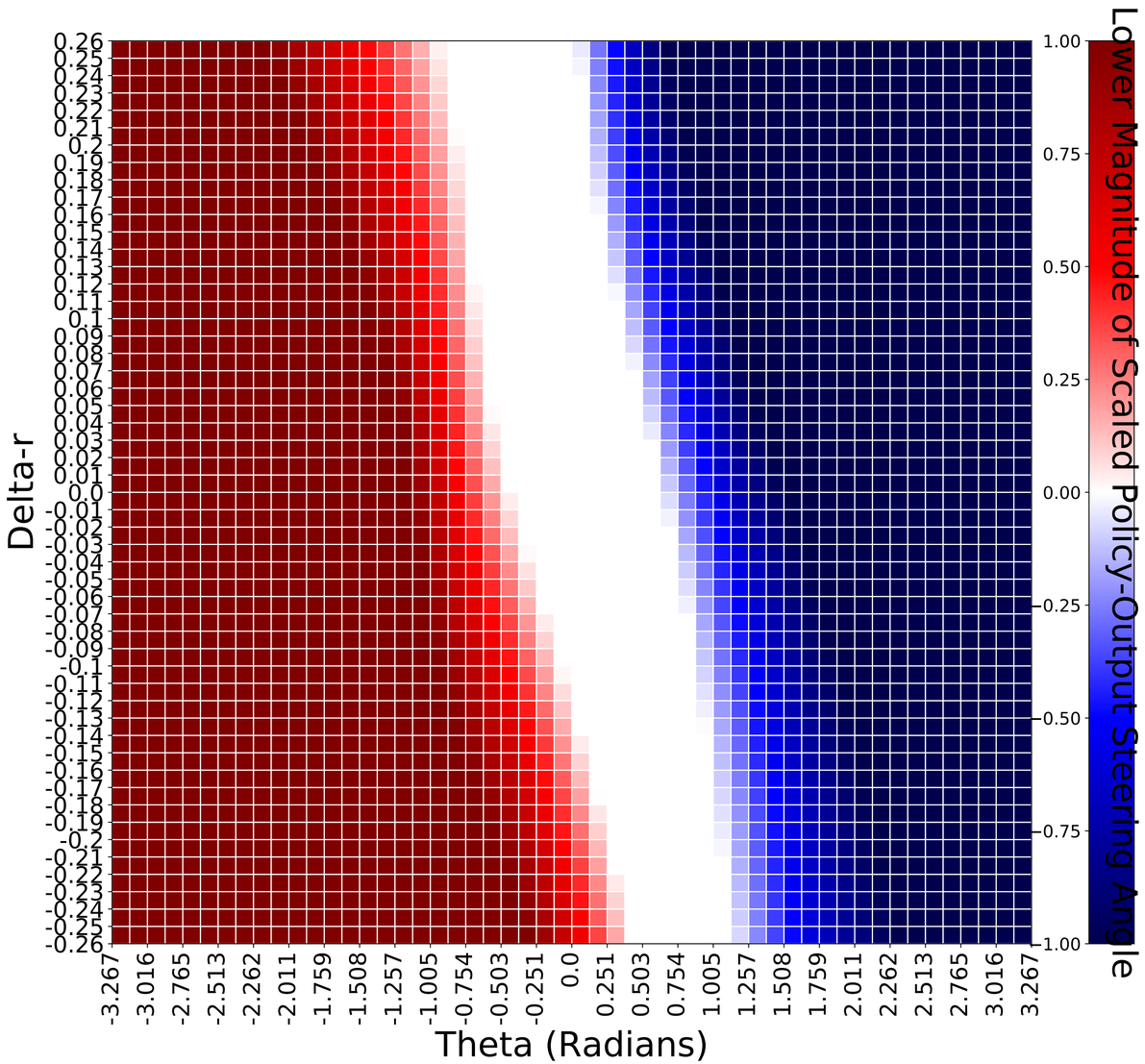




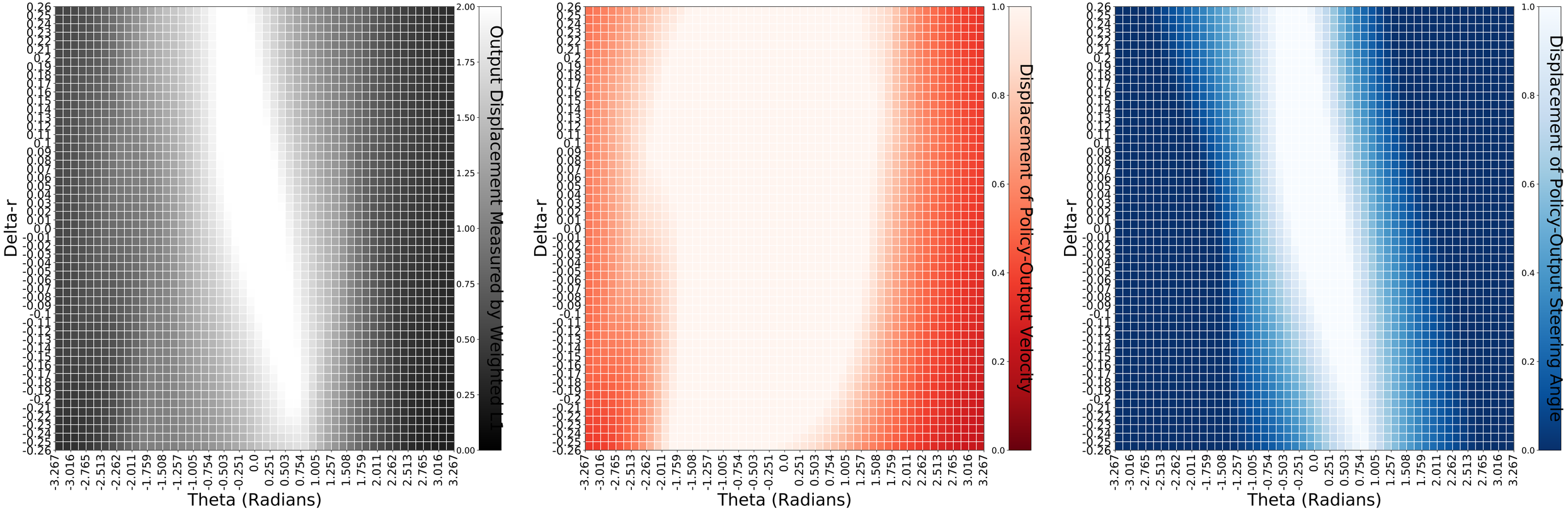


Circle-Following Min. Magnitude Output Results
deltaR-dot:1.038461, deltaTheta-dot:0, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: fc802f14-04ec-483c-a696-abd749b63264
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_18_06_277484

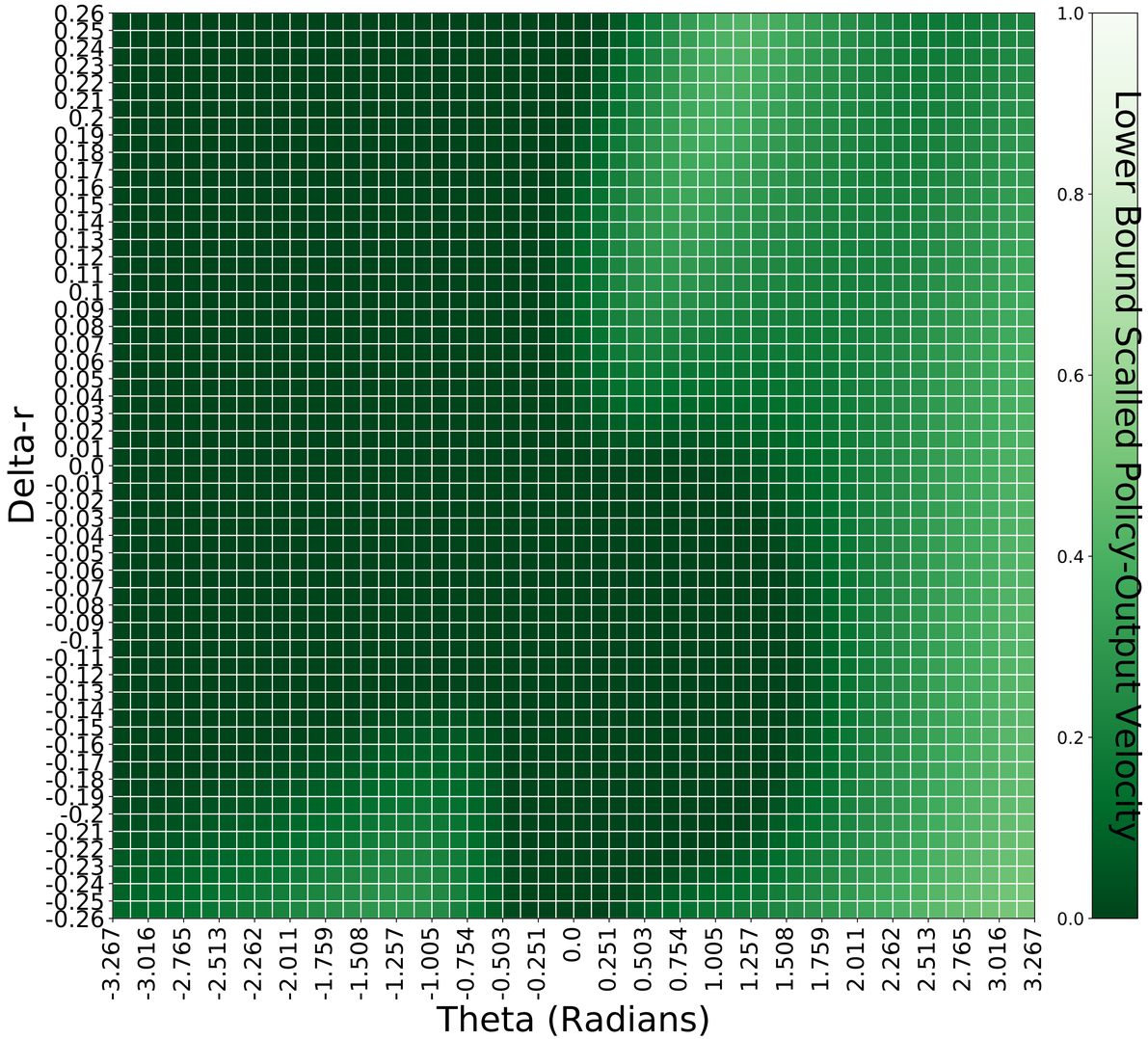


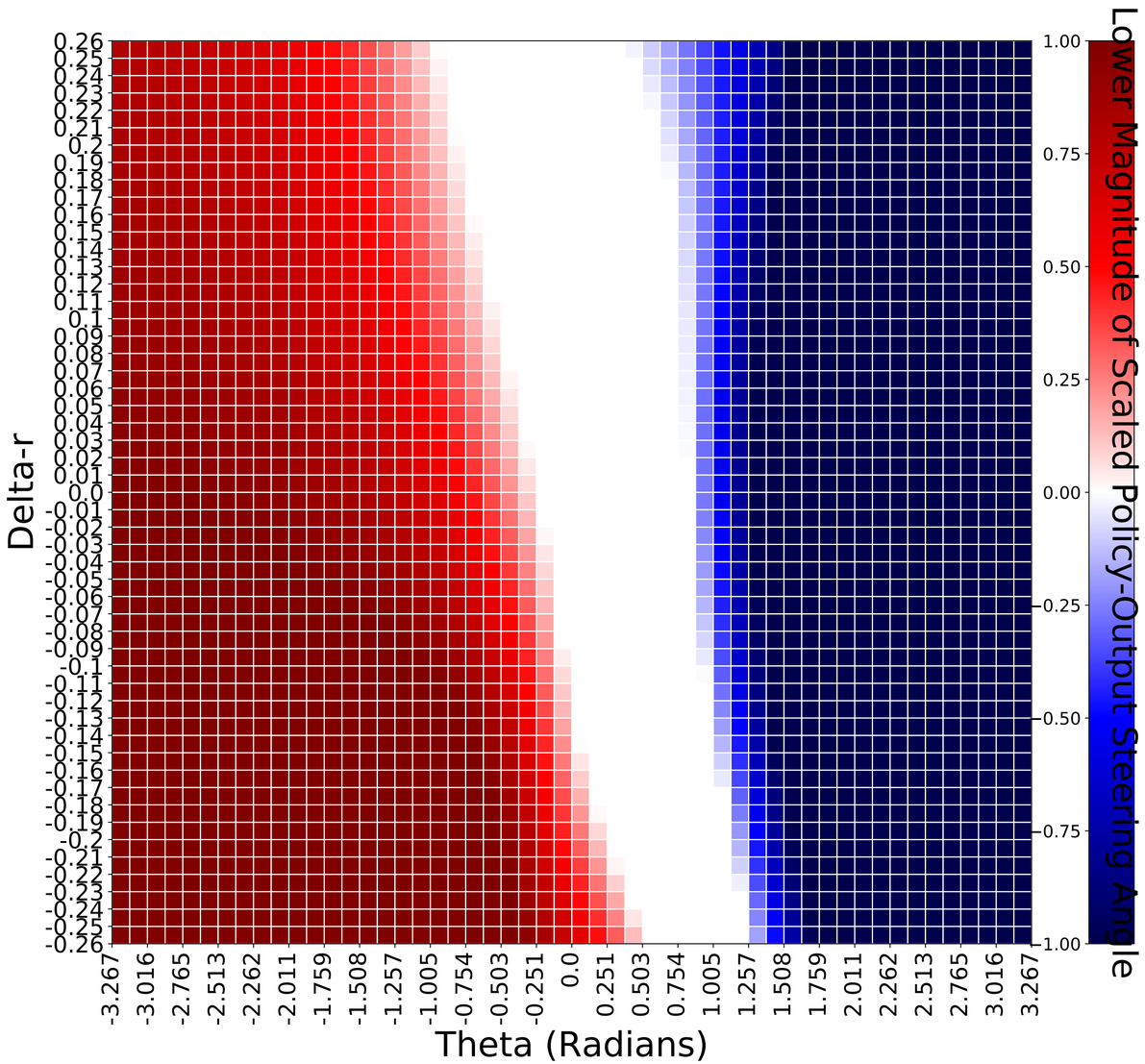


Circle-Following Stability Results
deltaR-dot:1.038461, deltaTheta-dot:0, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 2eb4ab9c-9397-46ce-bdac-97fc77430f01
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_15_55_901938

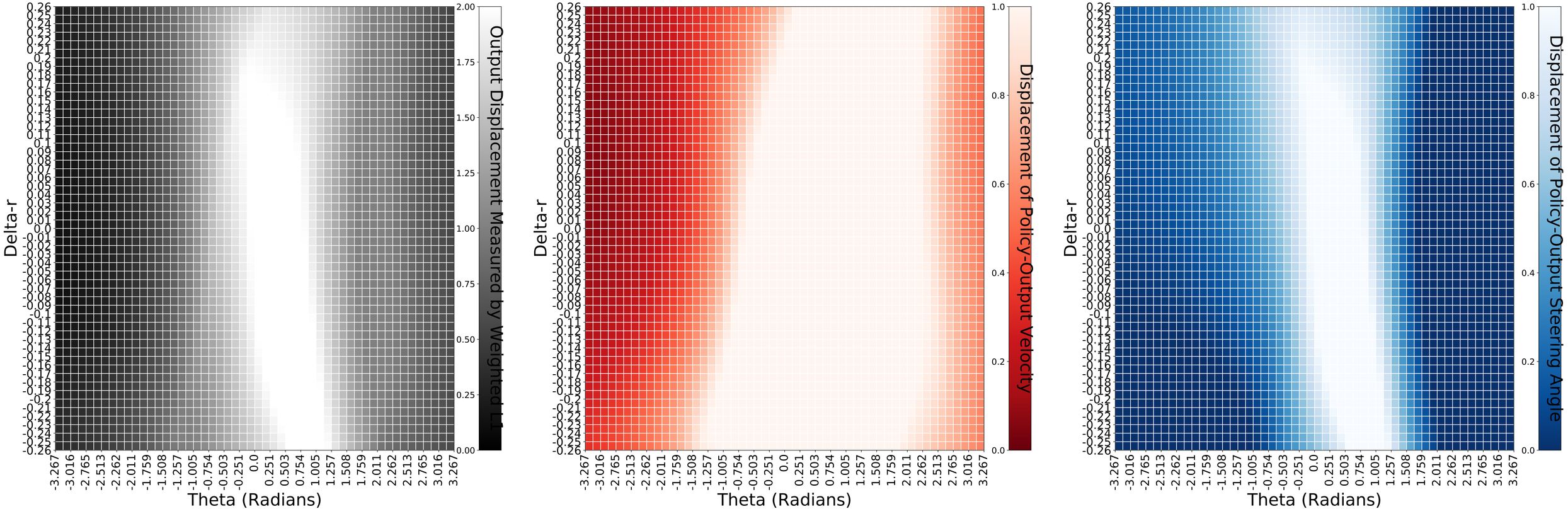


Circle-Following Min. Magnitude Output Results
deltaR-dot:-1.26923, deltaTheta-dot:0.278846, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 3822ac66-5845-4bce-a13b-70737c09fc06
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_22_25_505480

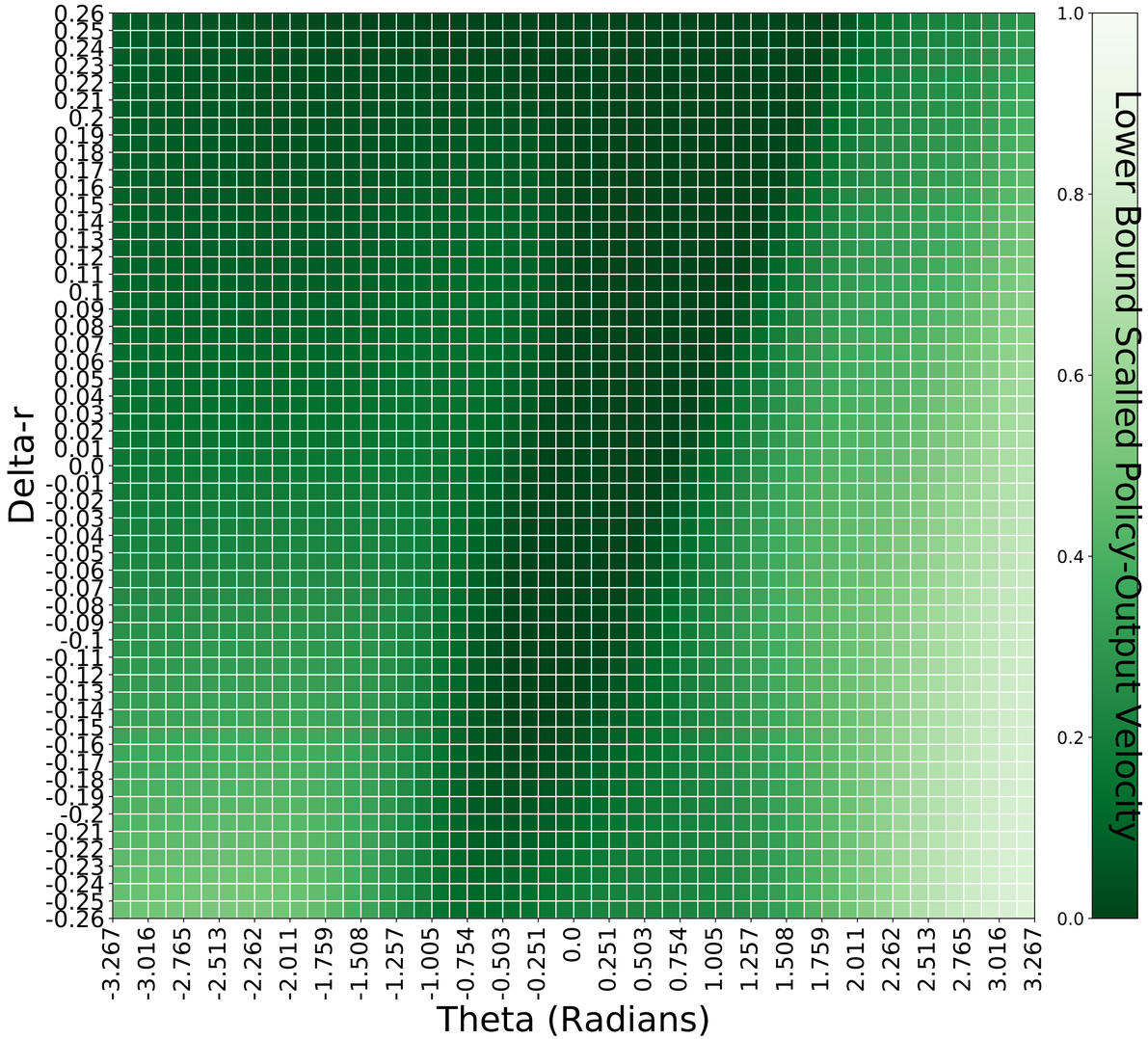


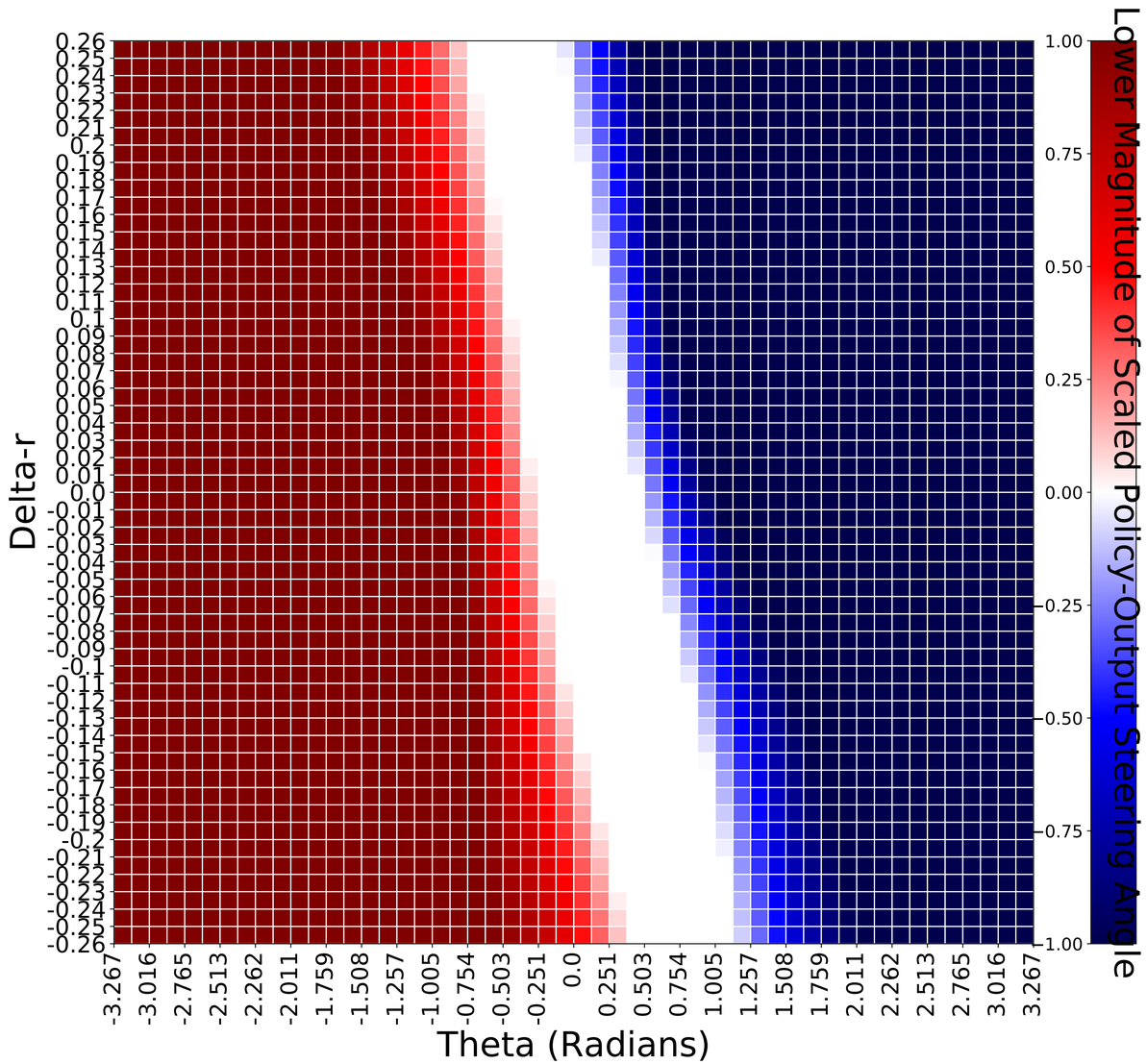


Circle-Following Stability Results
deltaR-dot:-1.26923, deltaTheta-dot:0.278846, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 7e0e0de5-bd12-427d-959c-95d984924ae0
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_20_14_716086

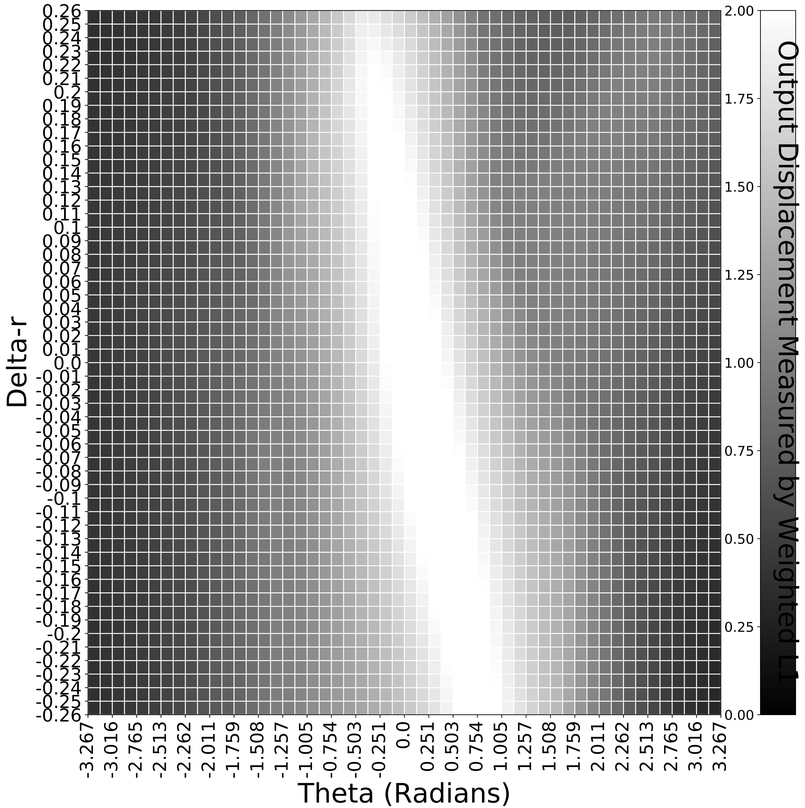


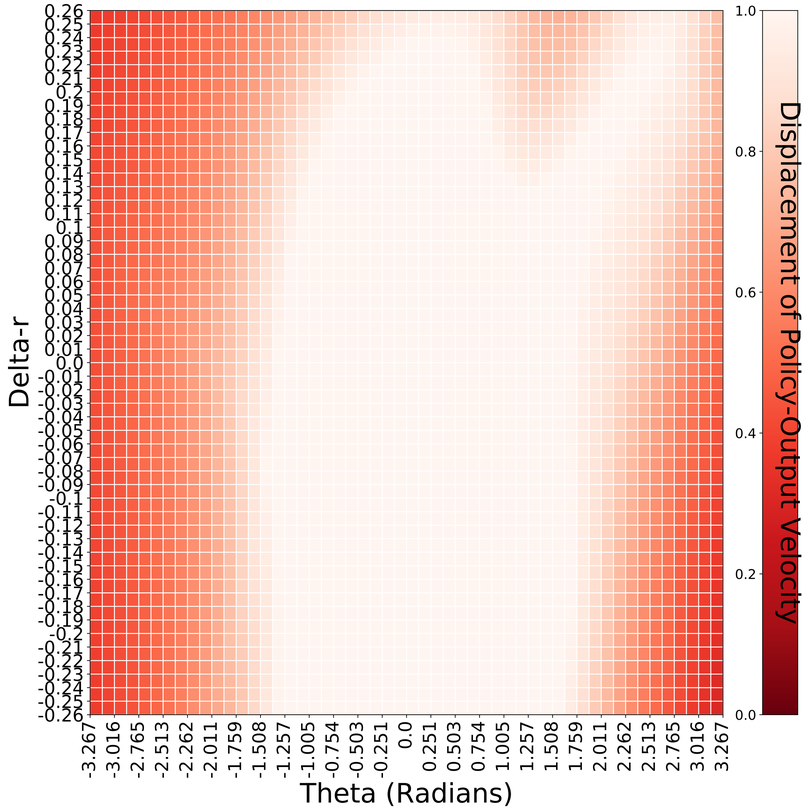
Circle-Following Min. Magnitude Output Results
deltaR-dot:-1.26923, deltaTheta-dot:-1.95192, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 51147150-a679-4d75-831b-f4fc9105e2c9
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_26_49_691931

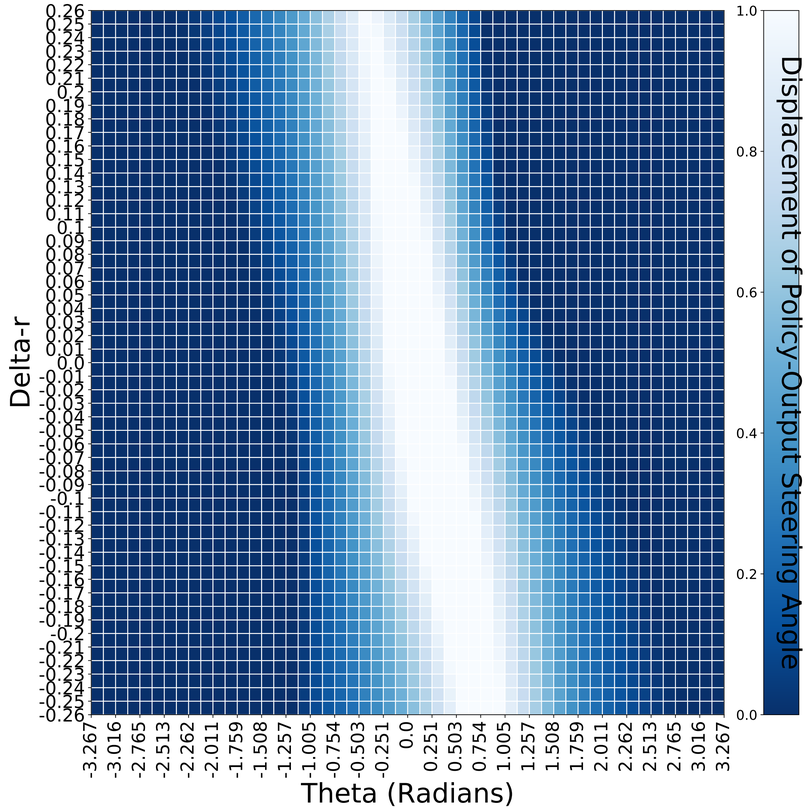




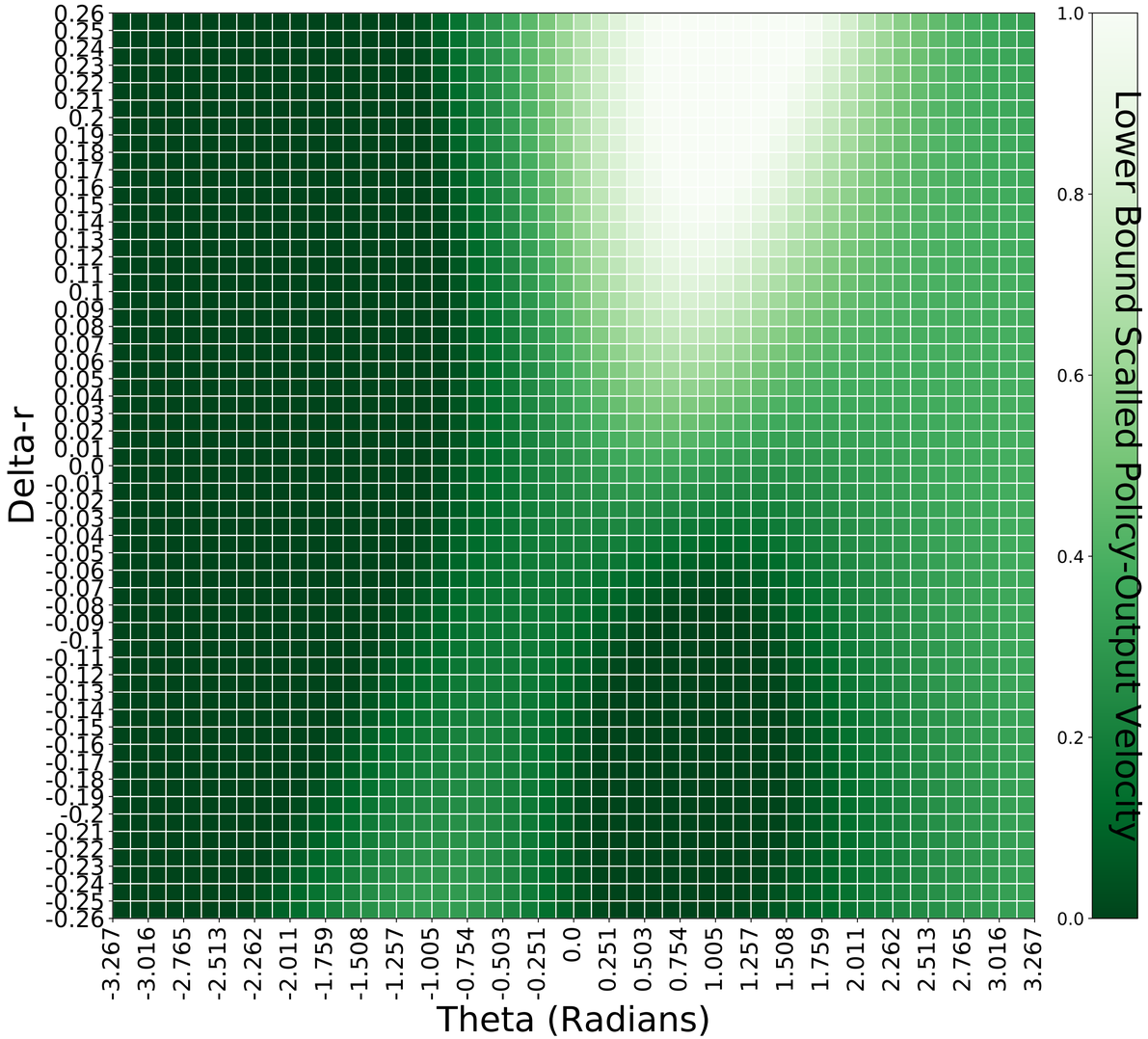
Circle-Following Stability Results
deltaR-dot:-1.26923, deltaTheta-dot:-1.95192, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: c262b35d-07bb-4ce4-9cbd-4cc754204b2c
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_24_33_725364

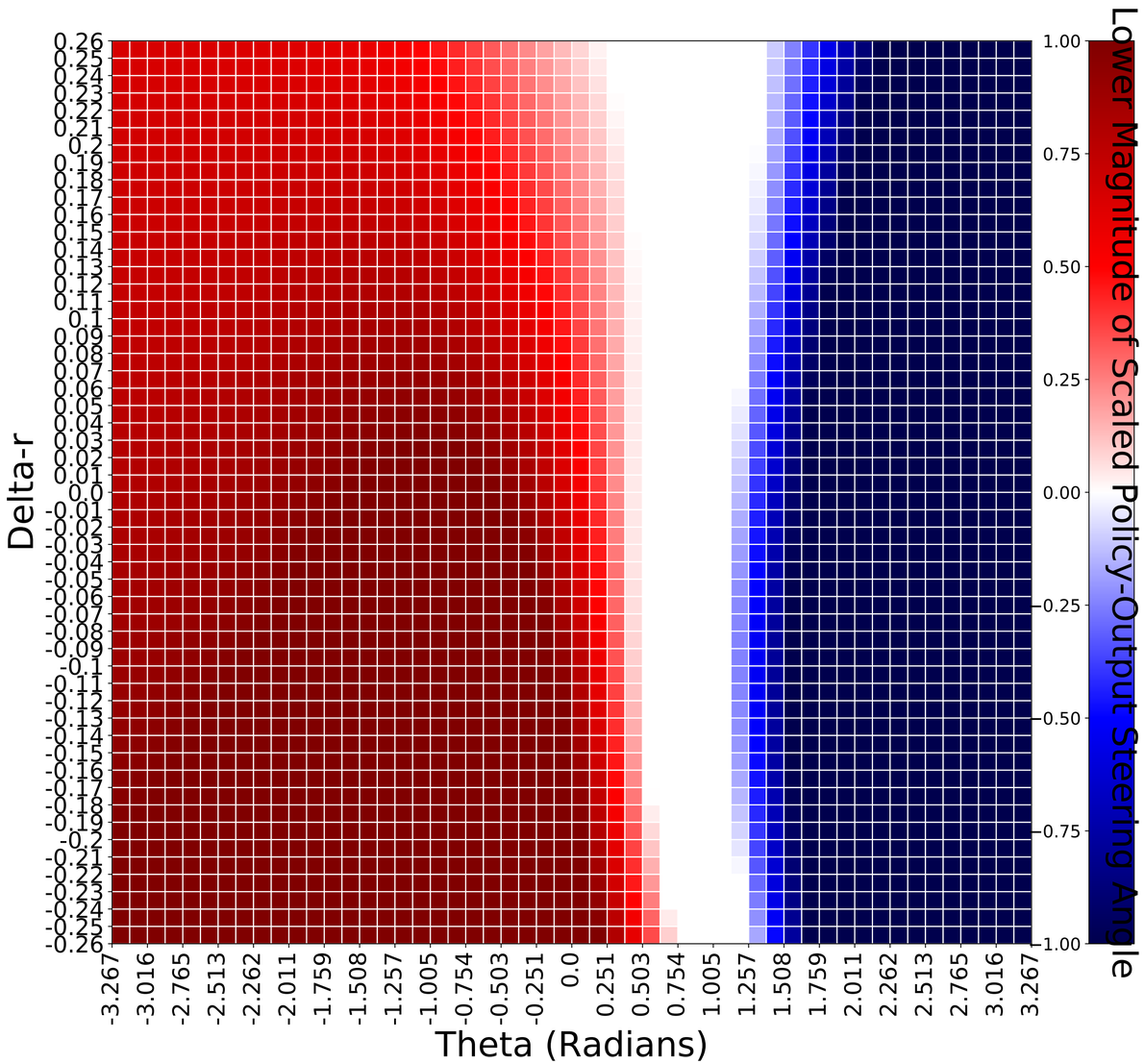




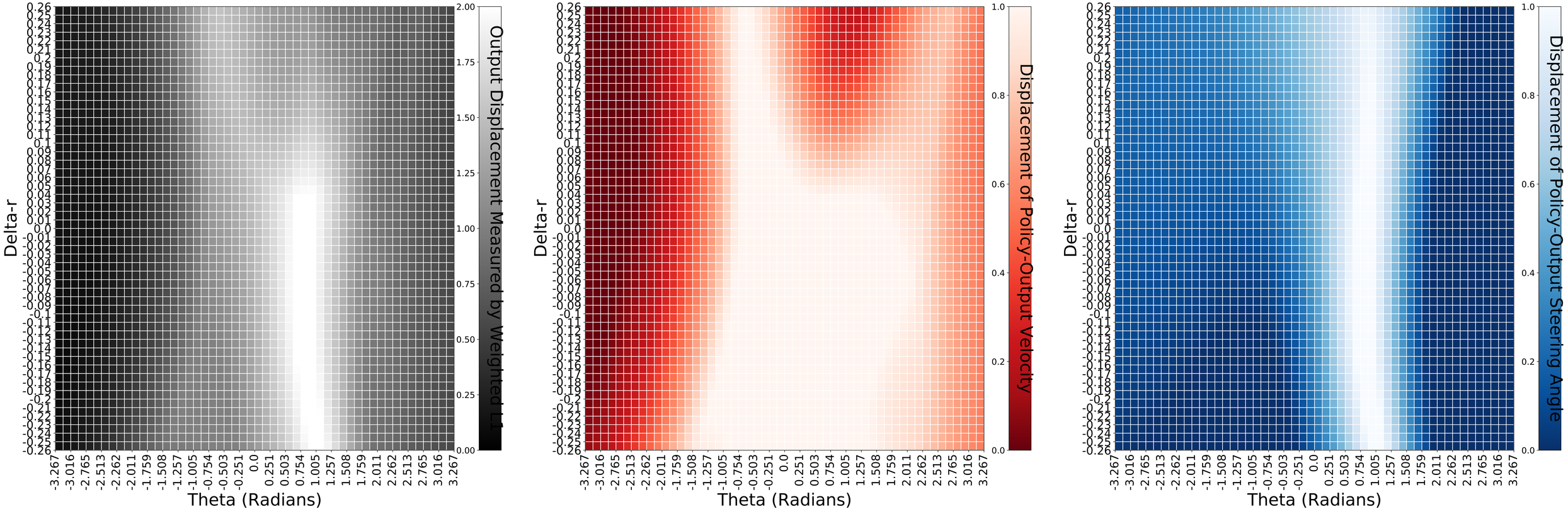


Circle-Following Min. Magnitude Output Results
deltaR-dot:-1.26923, deltaTheta-dot:2.509615, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 775cf63a-6cab-4d65-b1af-a3f407f19fe6
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_31_20_104708

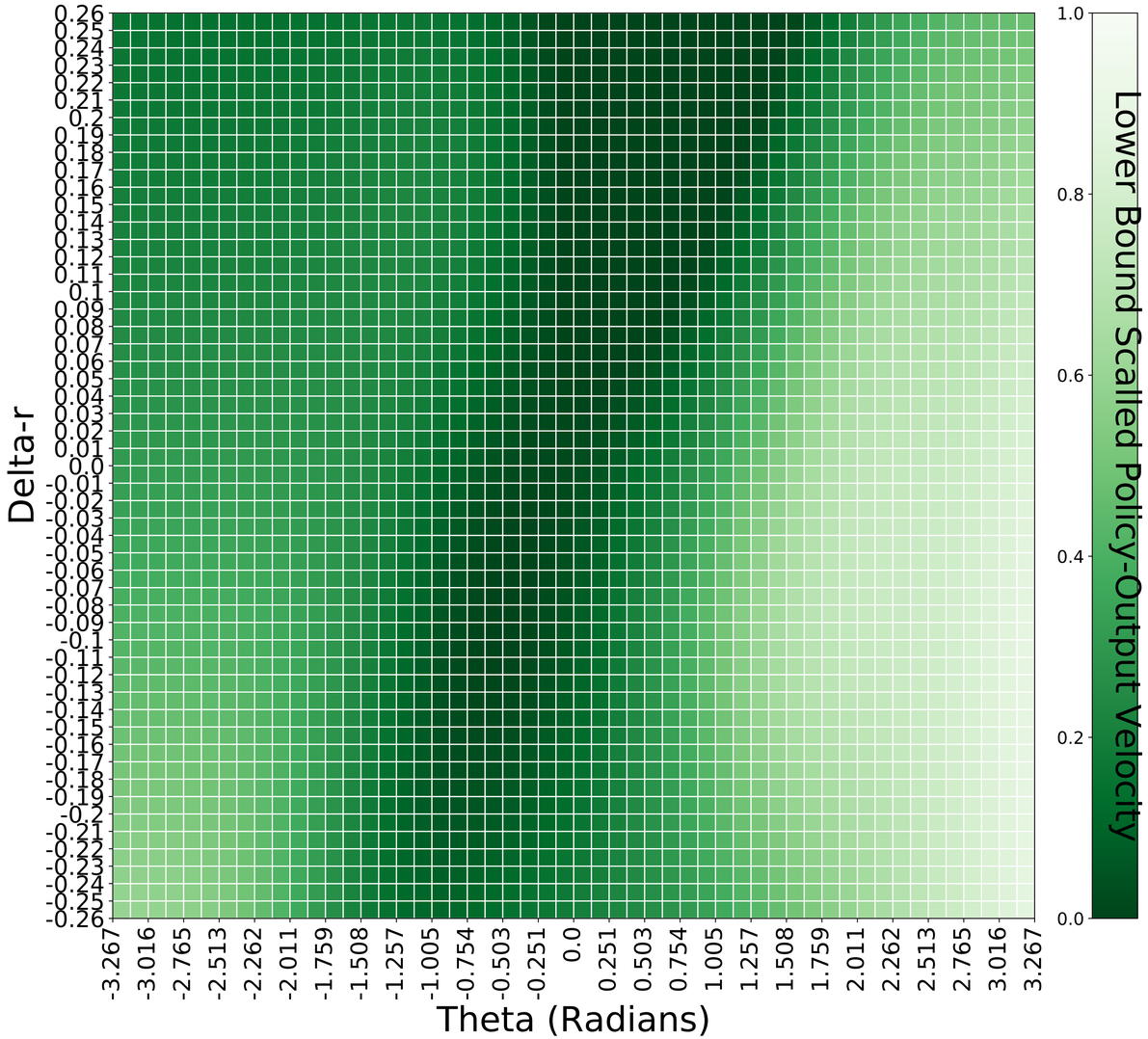


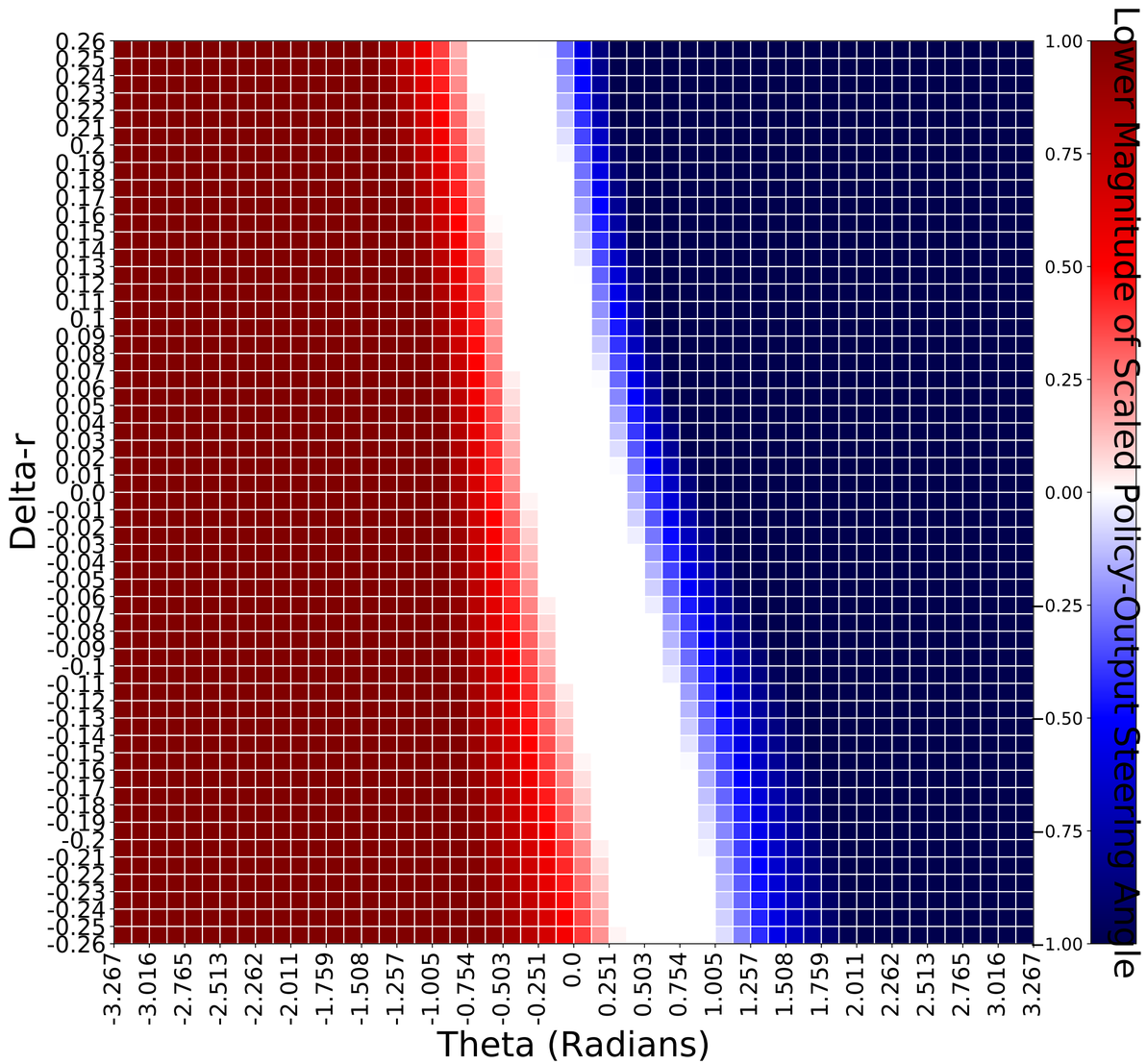


Circle-Following Stability Results
deltaR-dot:-1.26923, deltaTheta-dot:2.509615, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 1a72142d-3c35-4c41-82fa-352048f9ddb5
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_29_03_930325

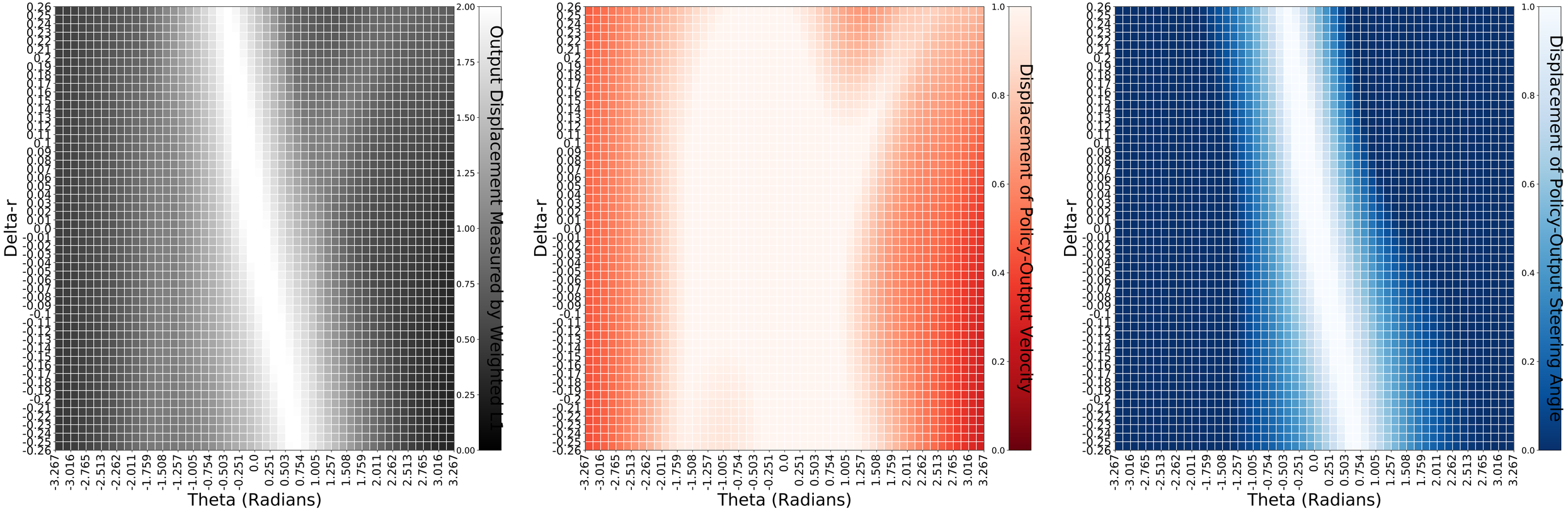


Circle-Following Min. Magnitude Output Results
deltaR-dot:-1.26923, deltaTheta-dot:-3.06730, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 55d4aeeb-aa9a-4cf3-81ea-2b6c76f2e0ca
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_35_50_070747

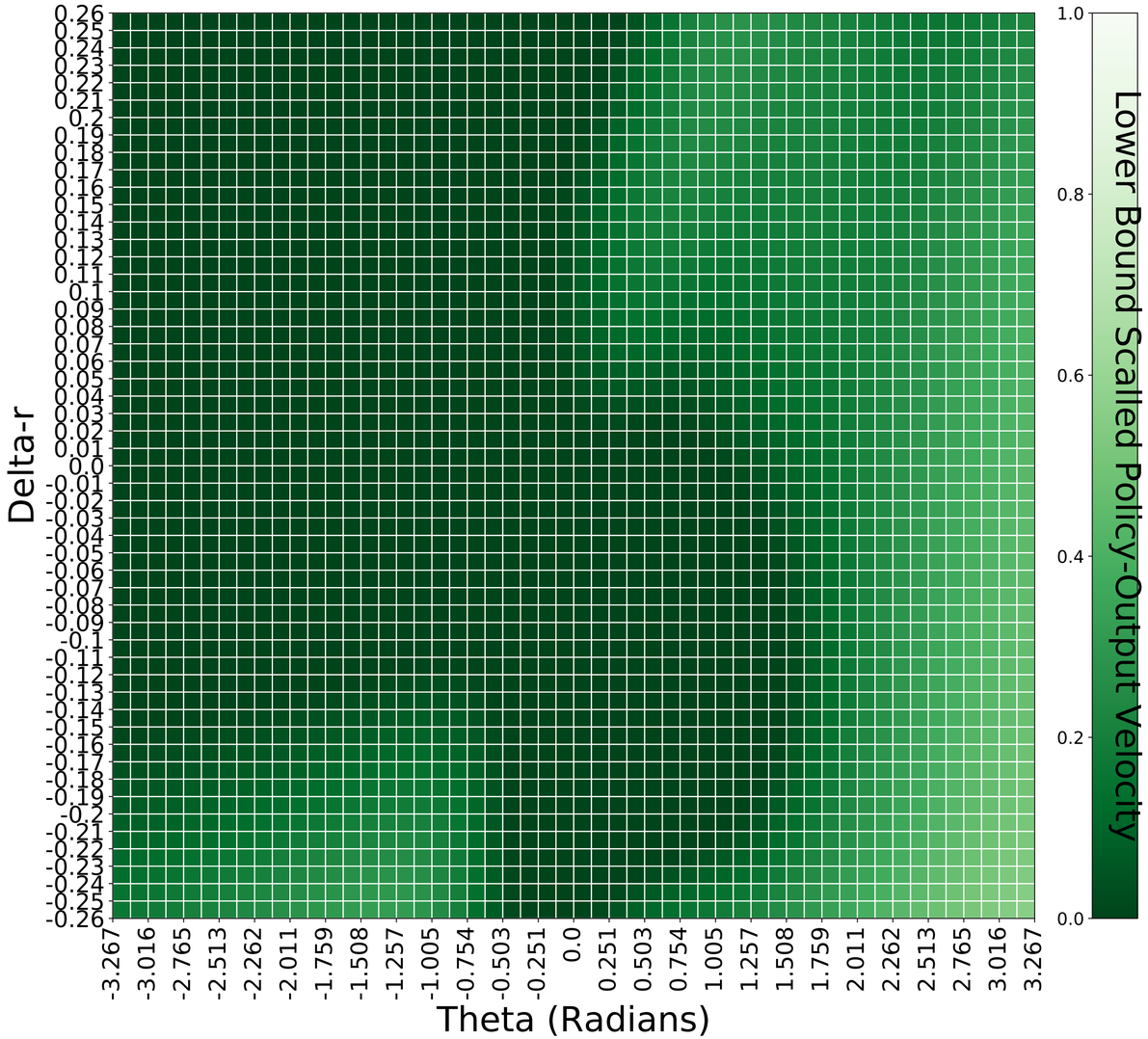


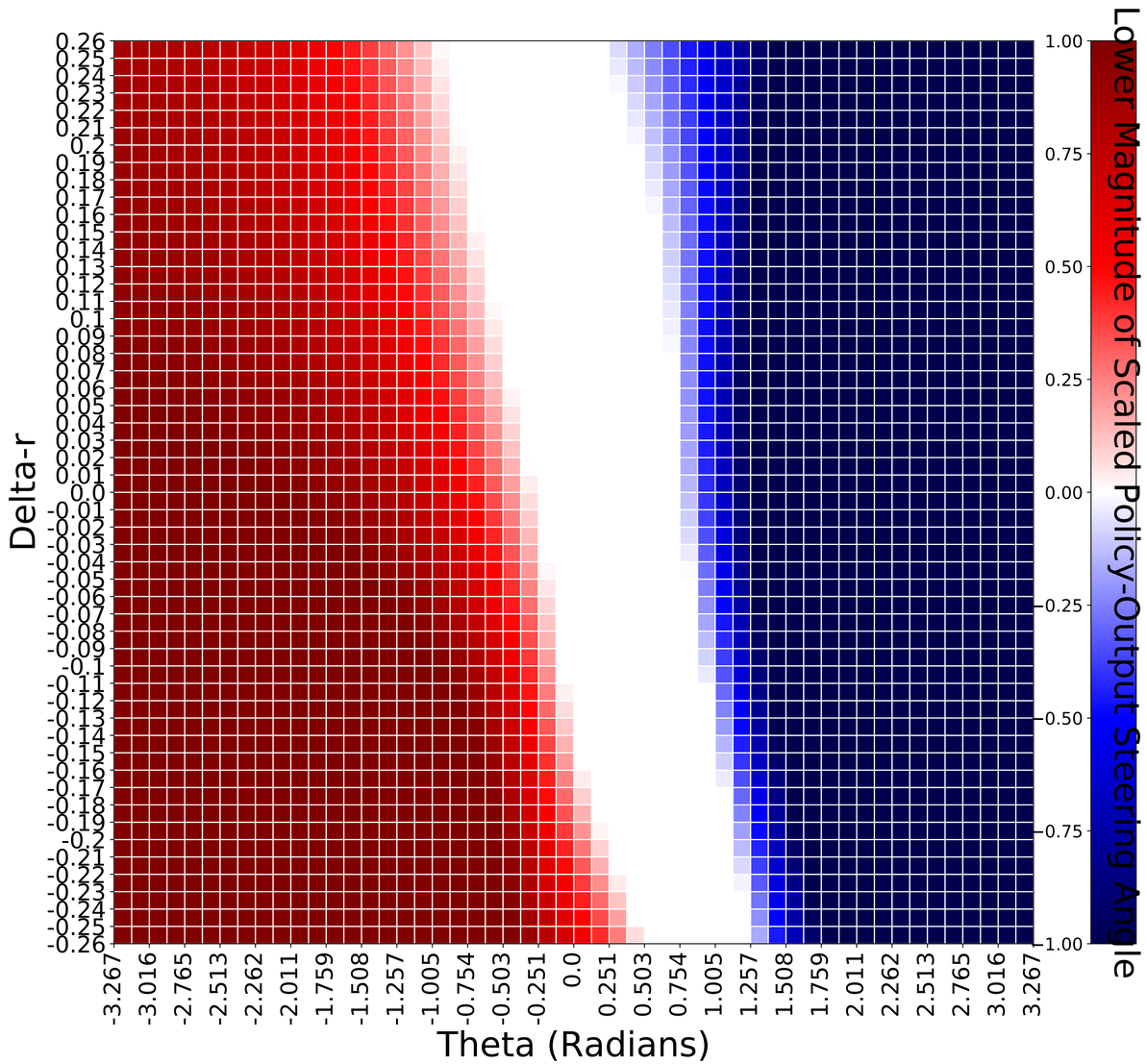


Circle-Following Stability Results
deltaR-dot:-1.26923, deltaTheta-dot:-3.06730, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 2dde0aec-0b1b-4e18-bef5-d46e5d5b3df7
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_01_33_33_845293

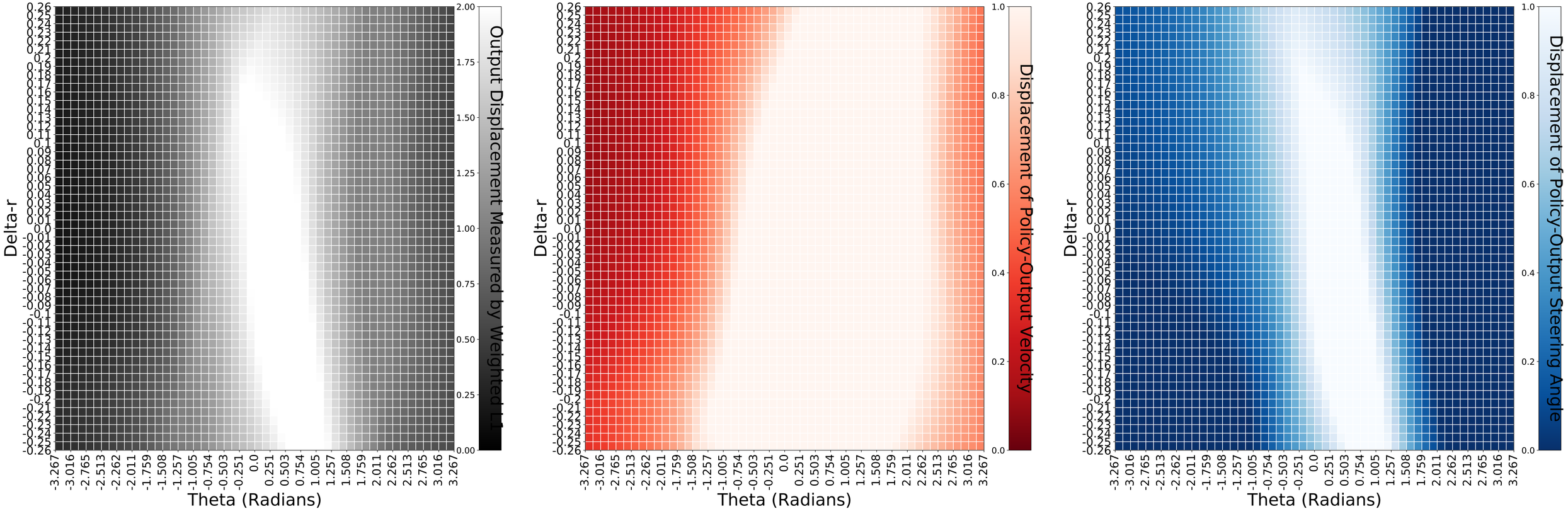


Circle-Following Min. Magnitude Output Results
deltaR-dot:-1.26923, deltaTheta-dot:0, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 60b111ef-9bbb-4e98-a8b0-d97945257aae
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_07_14_562863

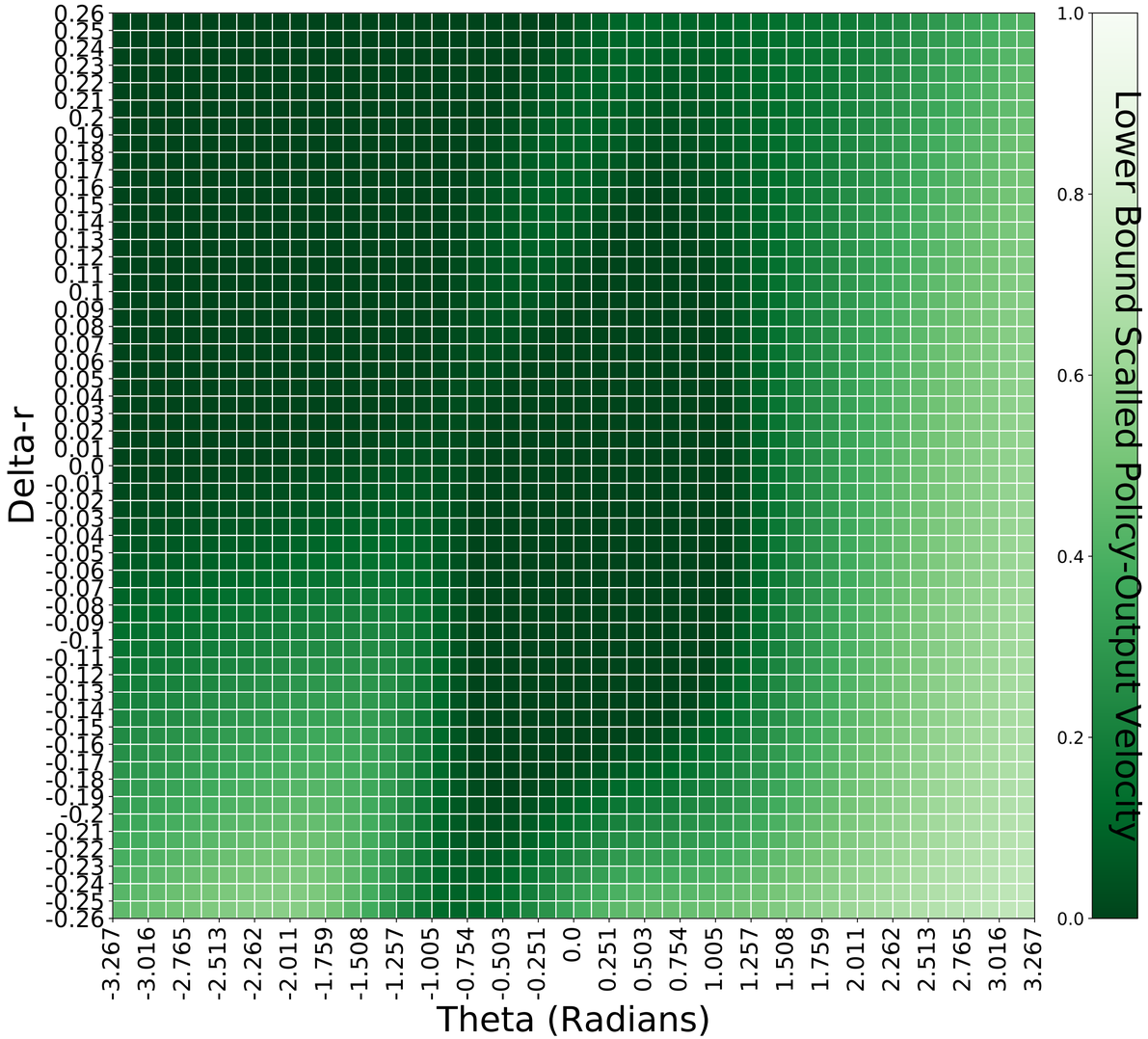


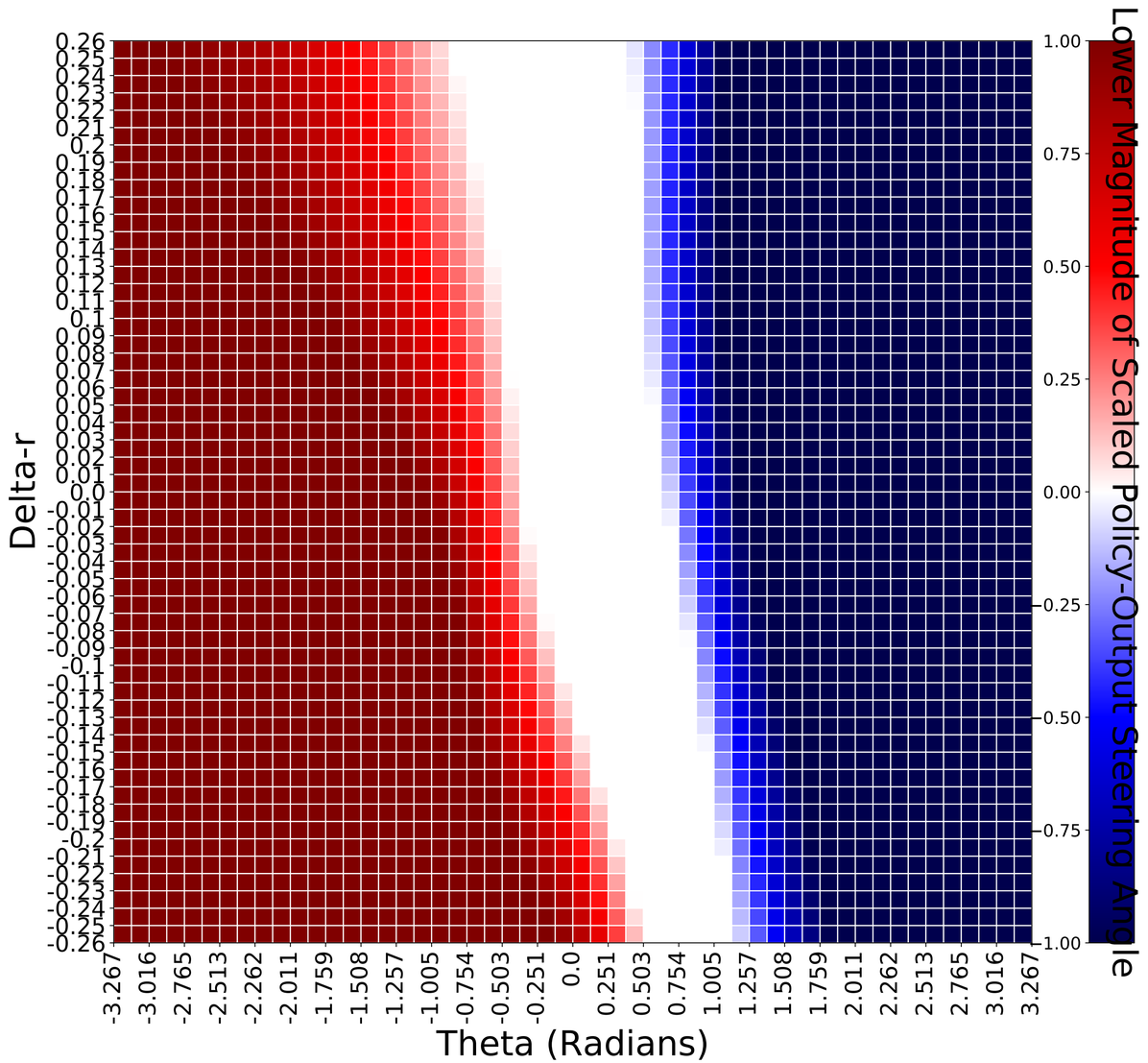


Circle-Following Stability Results
deltaR-dot:-1.26923, deltaTheta-dot:0, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 9f5c3f1b-c229-4916-b70e-cebeb88a2467
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_02_04_57_832781

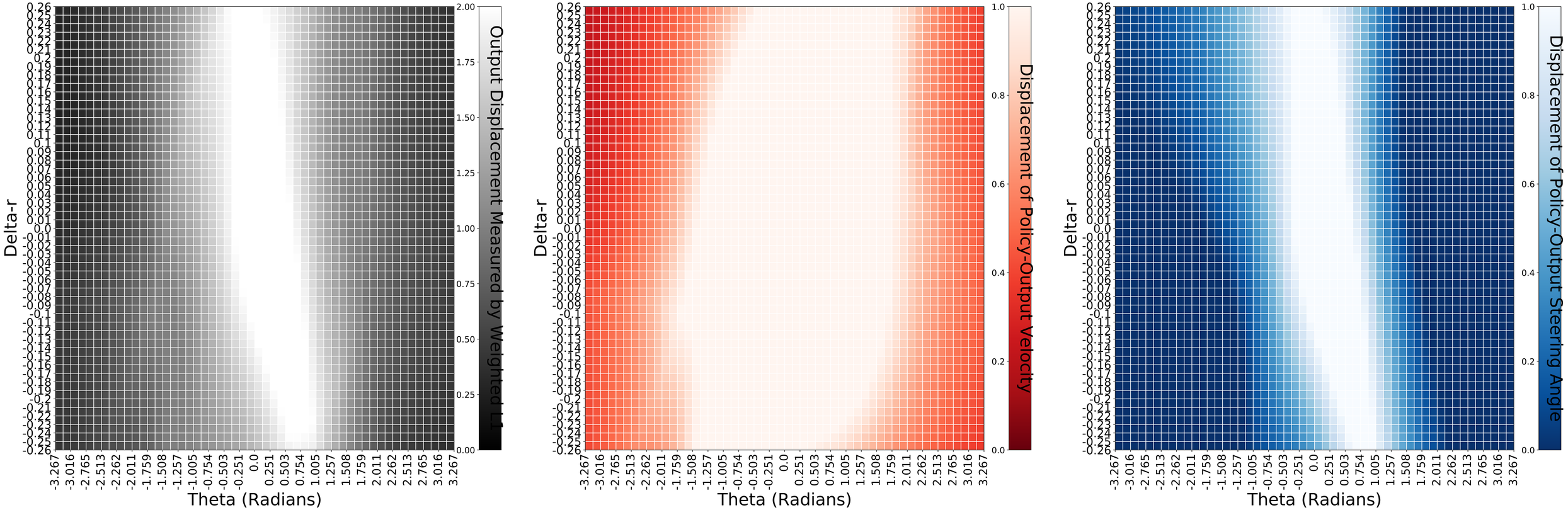


Circle-Following Min. Magnitude Output Results
deltaR-dot:0, deltaTheta-dot:0.278846, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 4ab9118d-120e-42e8-aeb4-1b4a187e6682
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_41_35_630722

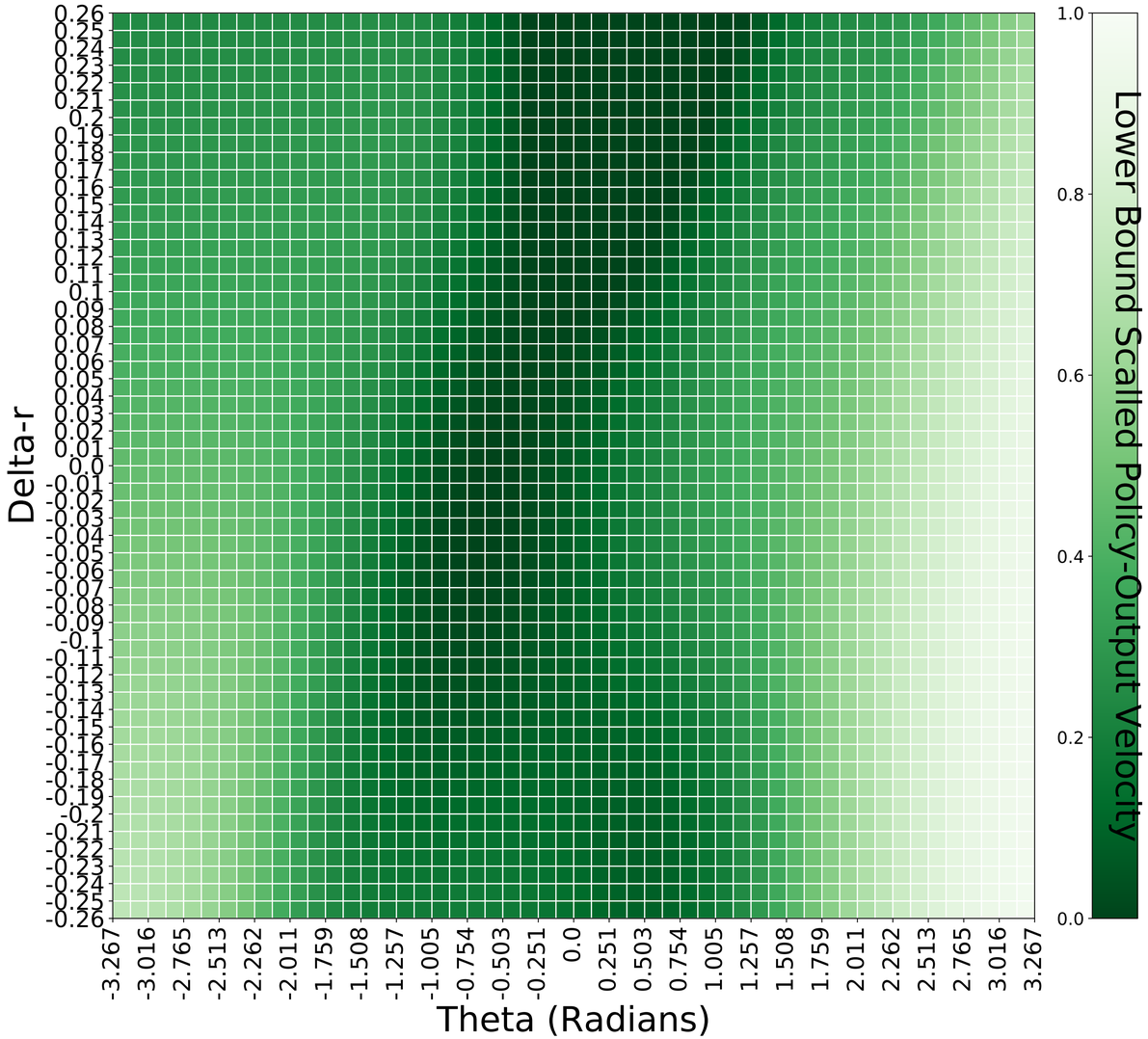


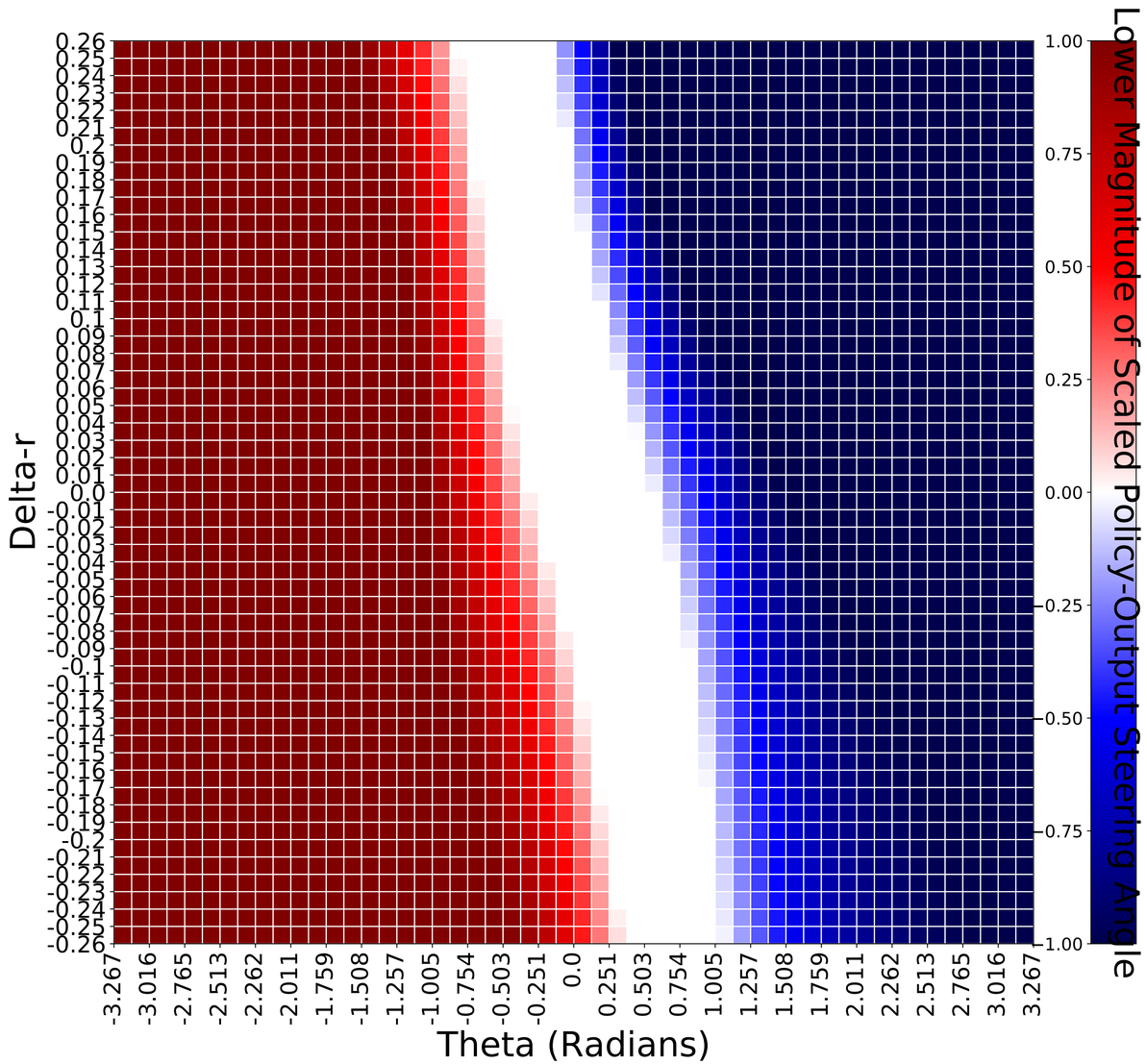


Circle-Following Stability Results
deltaR-dot:0, deltaTheta-dot:0.278846, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: b186deca-0009-4a03-b212-ac91c7328396
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_39_35_405542

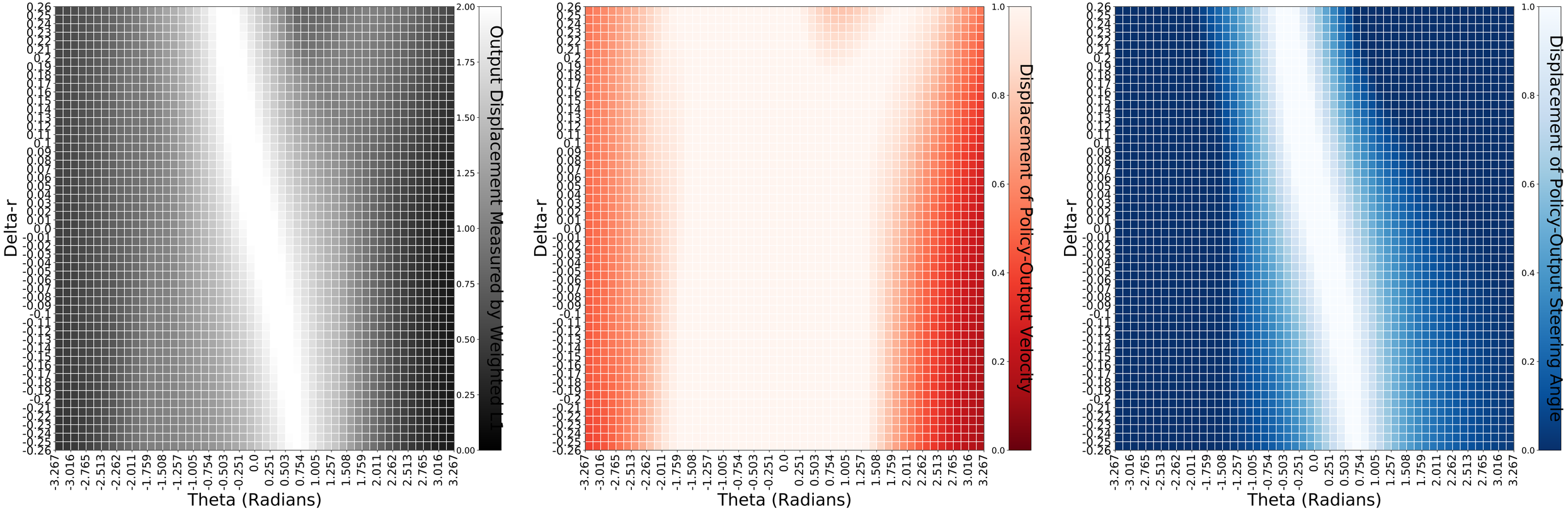


Circle-Following Min. Magnitude Output Results
deltaR-dot:0, deltaTheta-dot:-1.95192, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 9d6d836b-4e42-4596-ab5d-5280e4eef859
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_45_22_046726

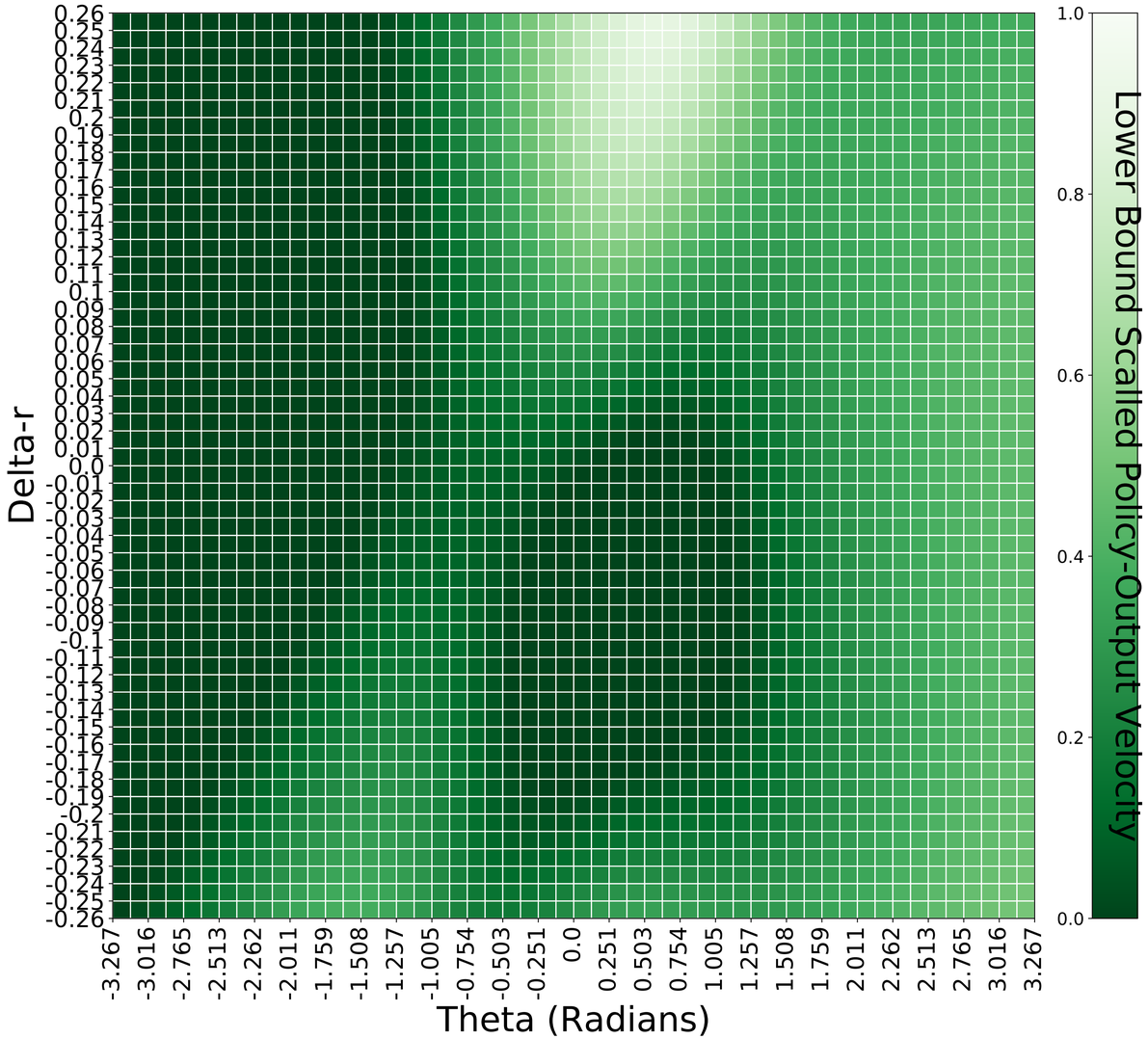


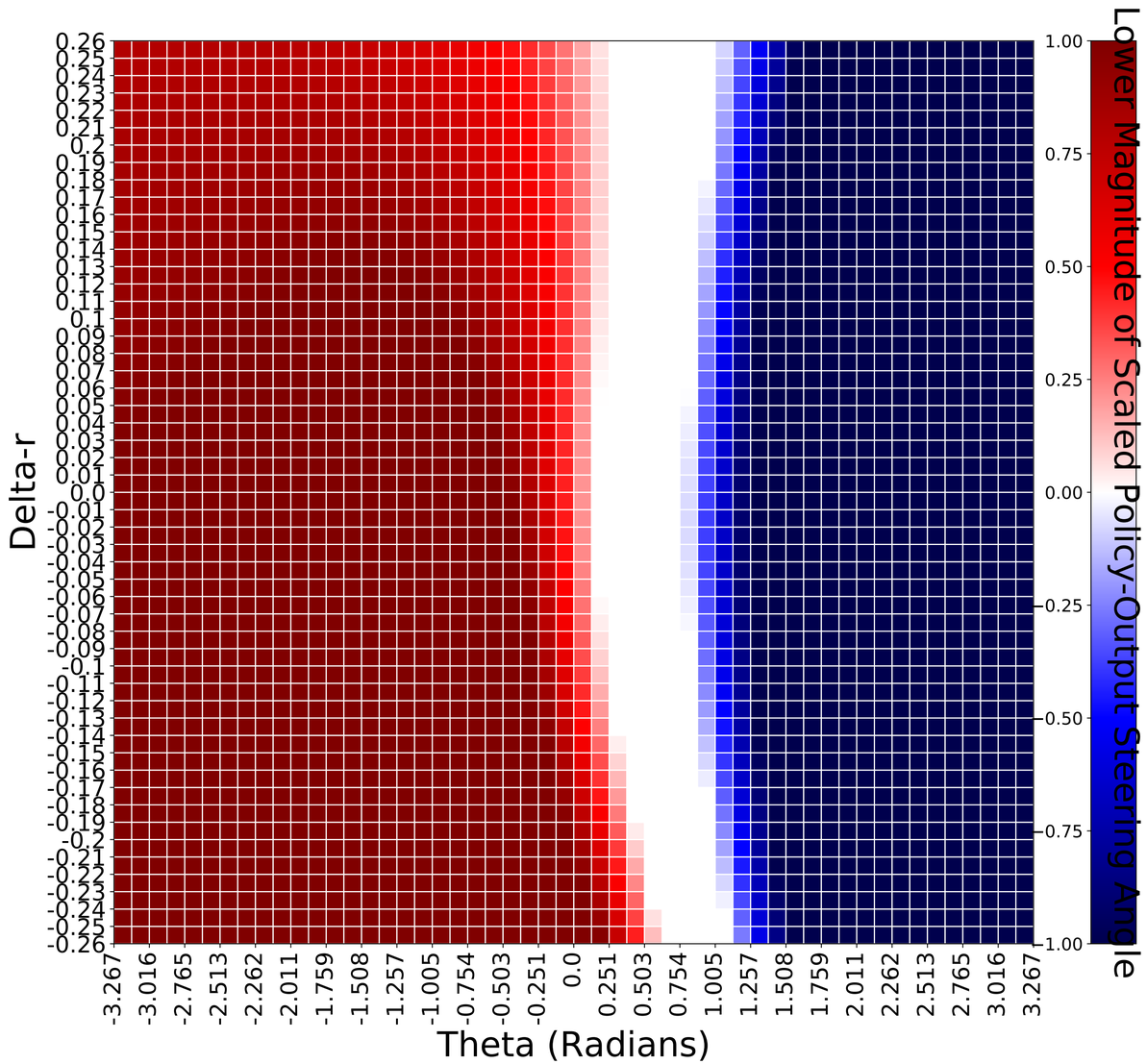


Circle-Following Stability Results
deltaR-dot:0, deltaTheta-dot:-1.95192, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 5de756e8-2219-4439-a805-67246adee5d8
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_43_37_747165

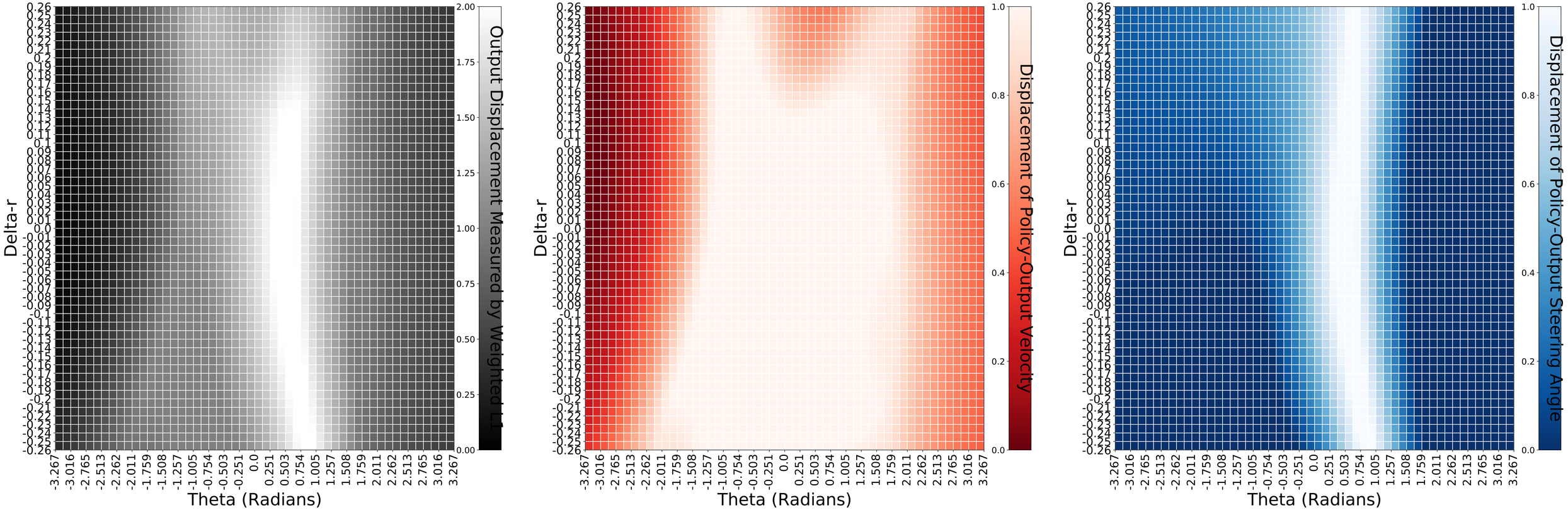


Circle-Following Min. Magnitude Output Results
deltaR-dot:0, deltaTheta-dot:2.509615, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: a33e571b-9deb-4ca7-9bf2-0c7ca72c06bc
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_49_31_757496

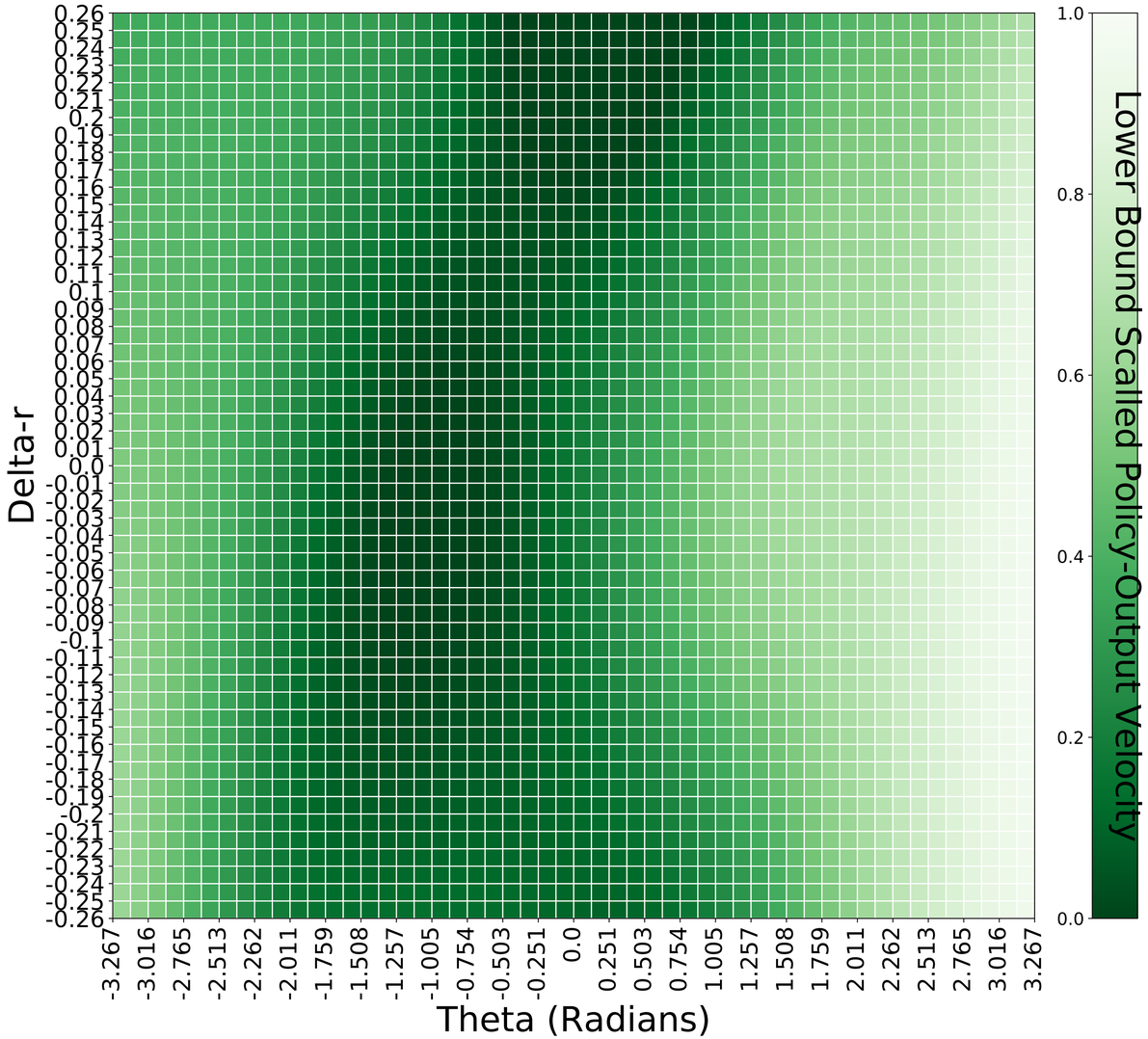


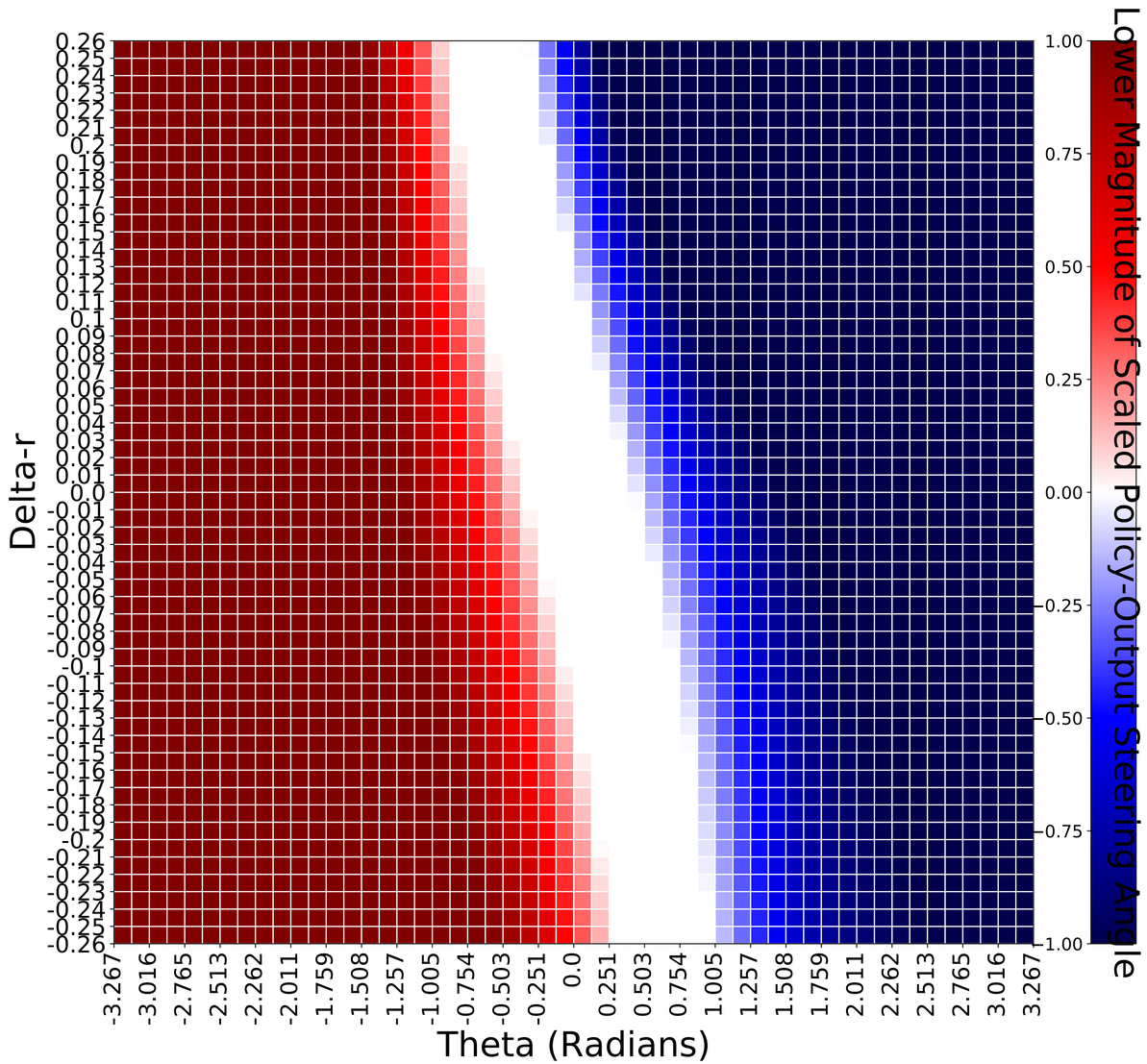


Circle-Following Stability Results
deltaR-dot:0, deltaTheta-dot:2.509615, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 426780a2-3ff2-49d1-931e-e51d44c223f8
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_47_25_071460

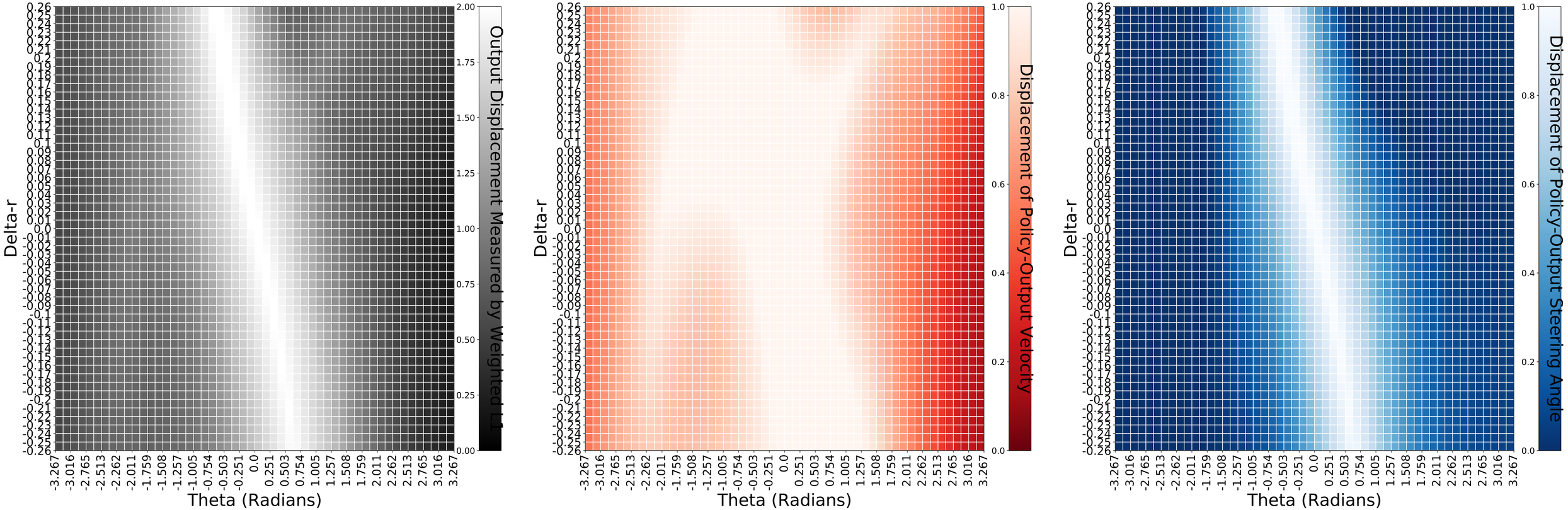


Circle-Following Min. Magnitude Output Results
deltaR-dot:0, deltaTheta-dot:-3.06730, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 34f2ba85-5b1f-4071-a3db-9c18bdf2d475
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_53_36_495975

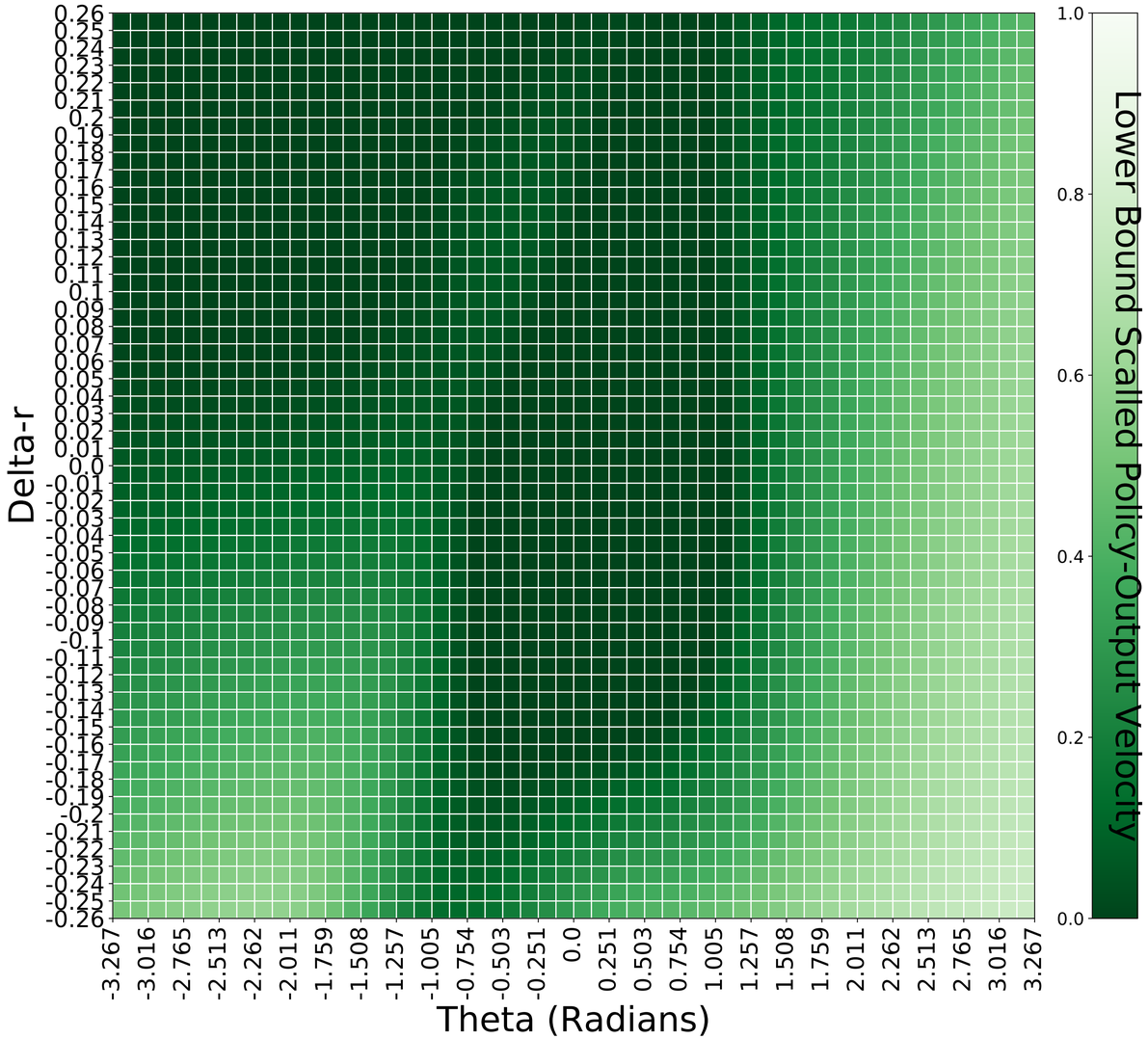


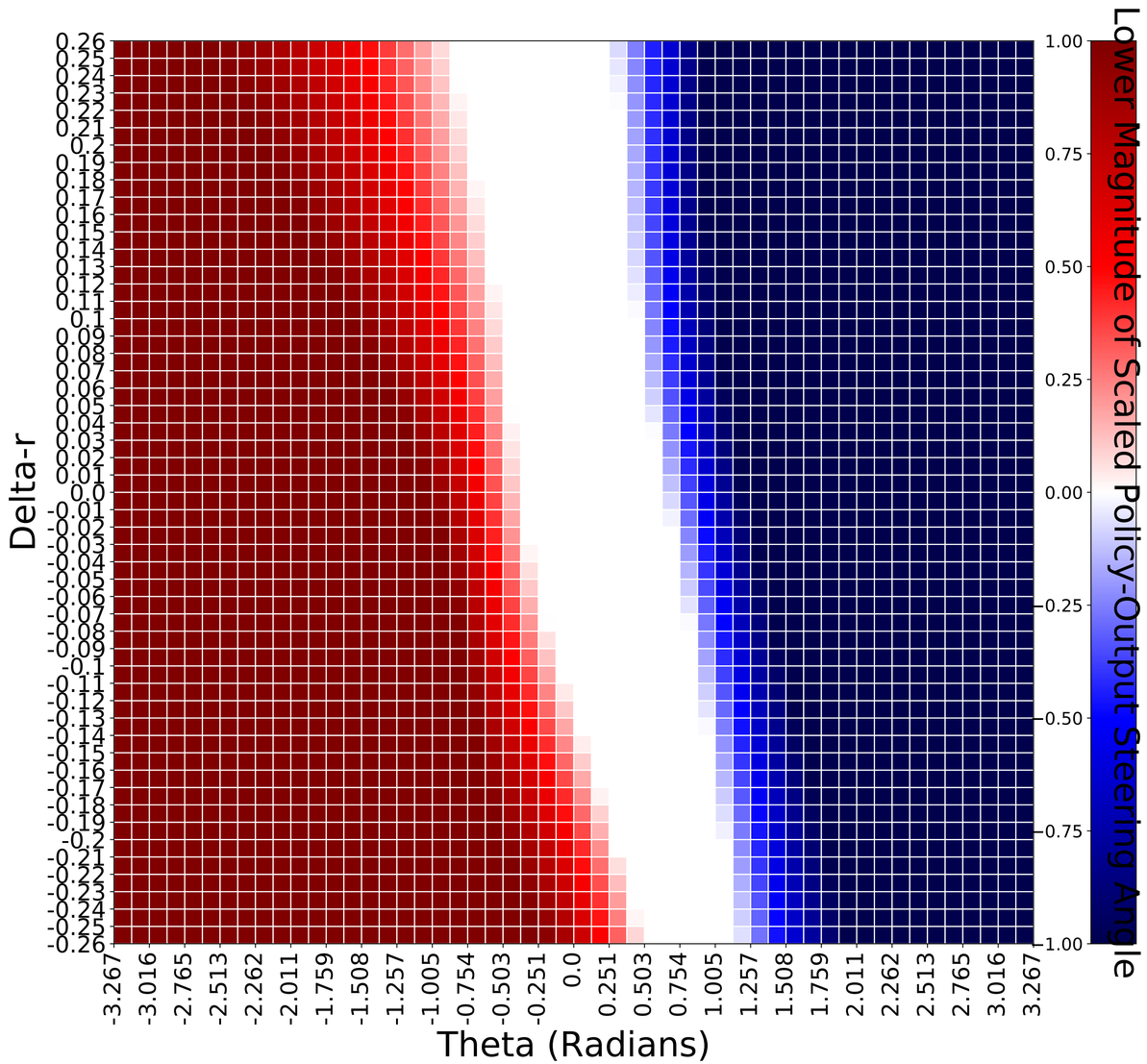


Circle-Following Stability Results
deltaR-dot:0, deltaTheta-dot:-3.06730, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 86b12c93-a811-4894-bc33-76501dc30df5
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_06_51_35_050665

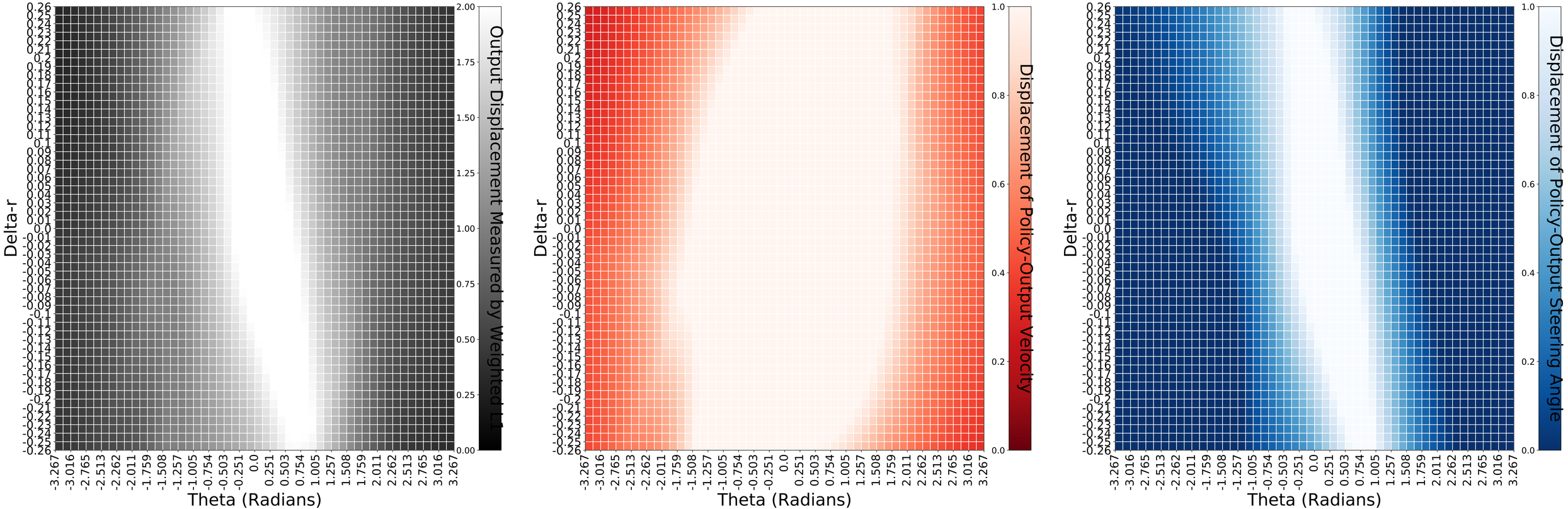


Circle-Following Min. Magnitude Output Results
deltaR-dot:0, deltaTheta-dot:0, percentEpsilonToUse:0.01
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 9237d402-dfae-4f1b-bb71-fbab48291f1d
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_07_22_47_842761



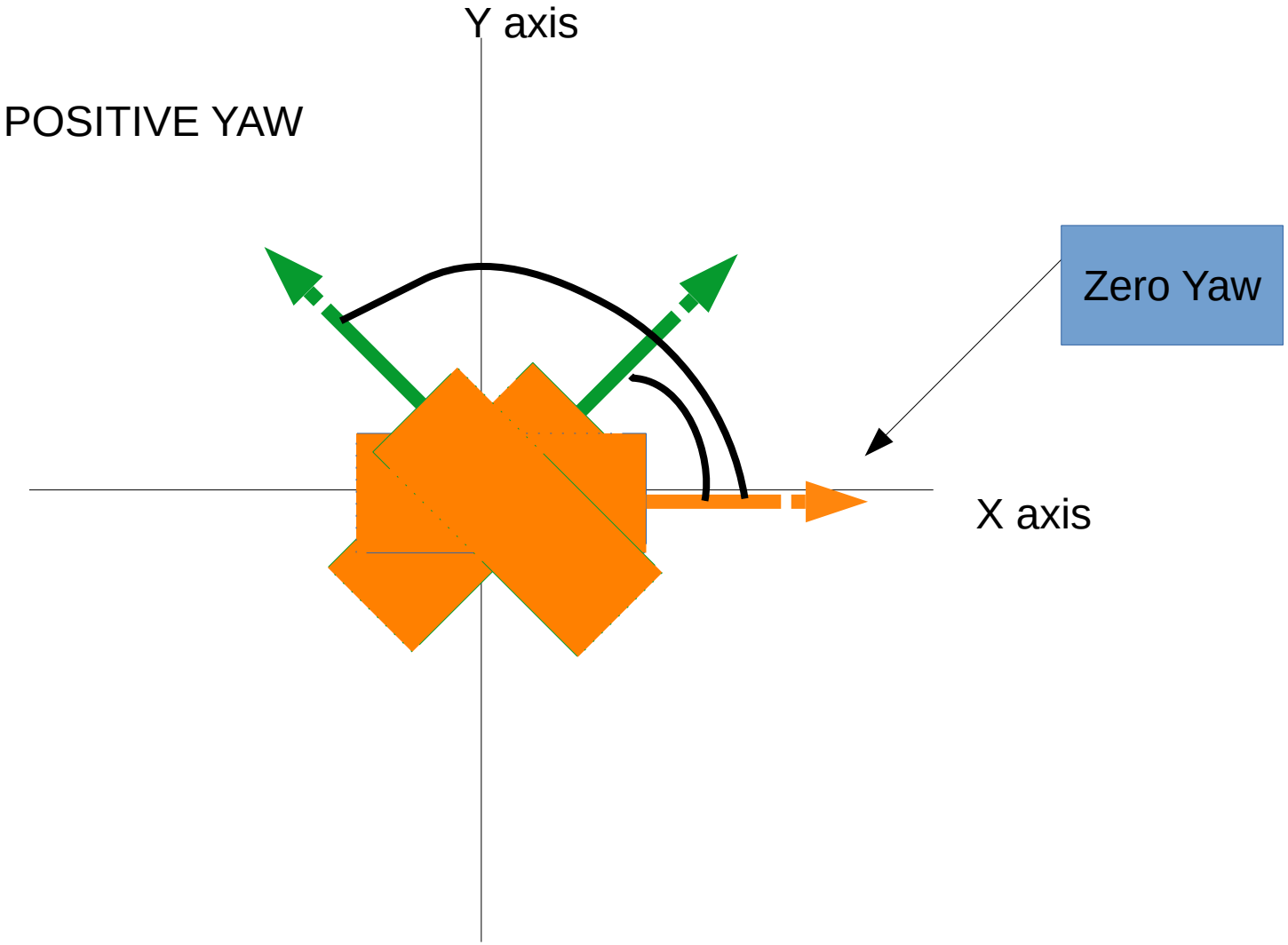


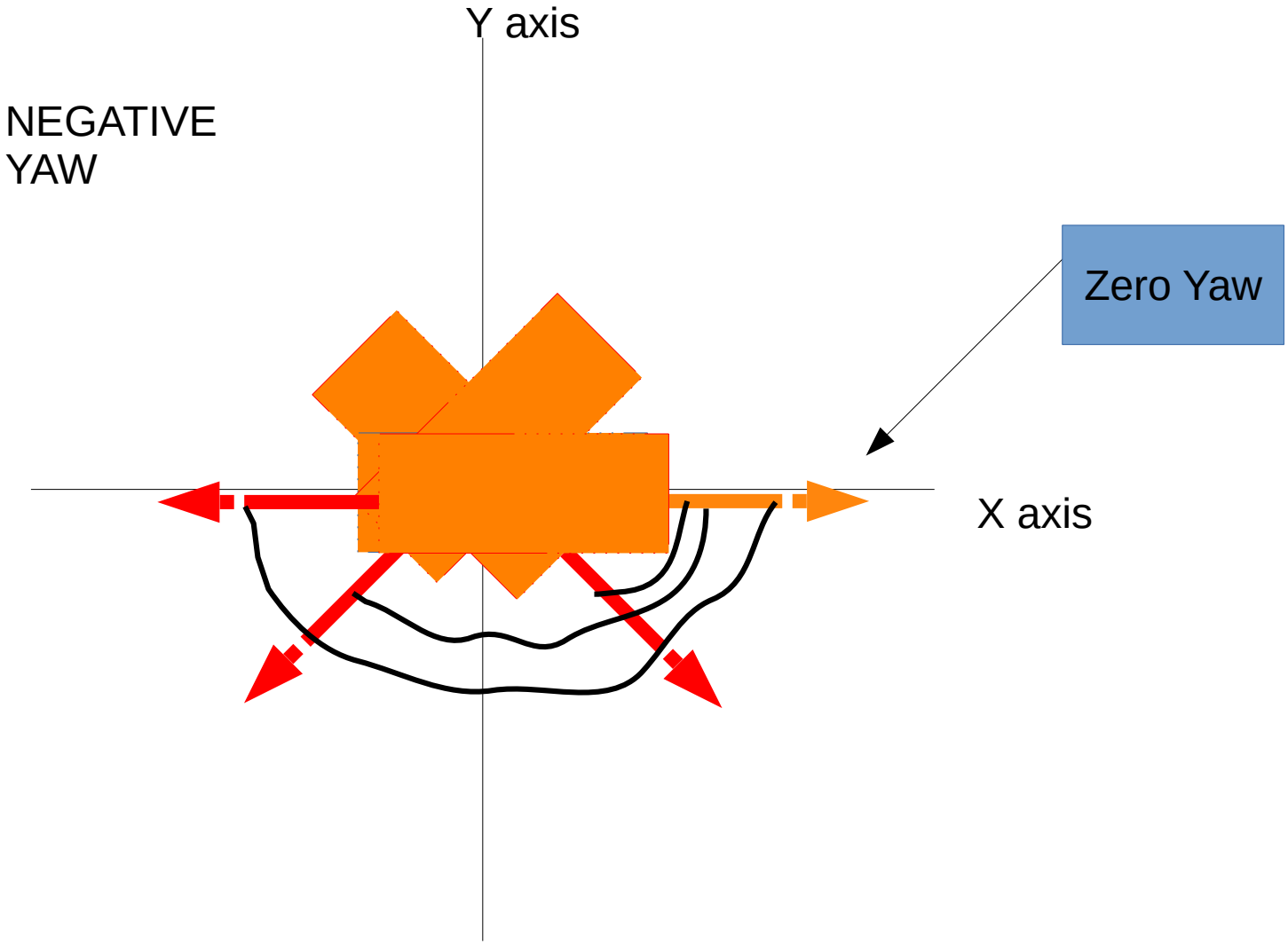
Circle-Following Stability Results
deltaR-dot:0, deltaTheta-dot:0, percentEpsilonToUse:0.01
Grey:displacement as L1-norm normalized per variable displayed
Red: displacement of policy-output velocity, Blue: displacement of policy-output steering angle
plot uuid: 0d26825d-facf-468c-8b8b-e705c4e4b678
git commit hash: 585f4b3e78d6c658322b2d70a37bc7bdc13e0d17
time plot formed: 2019_06_08_07_20_35_931726

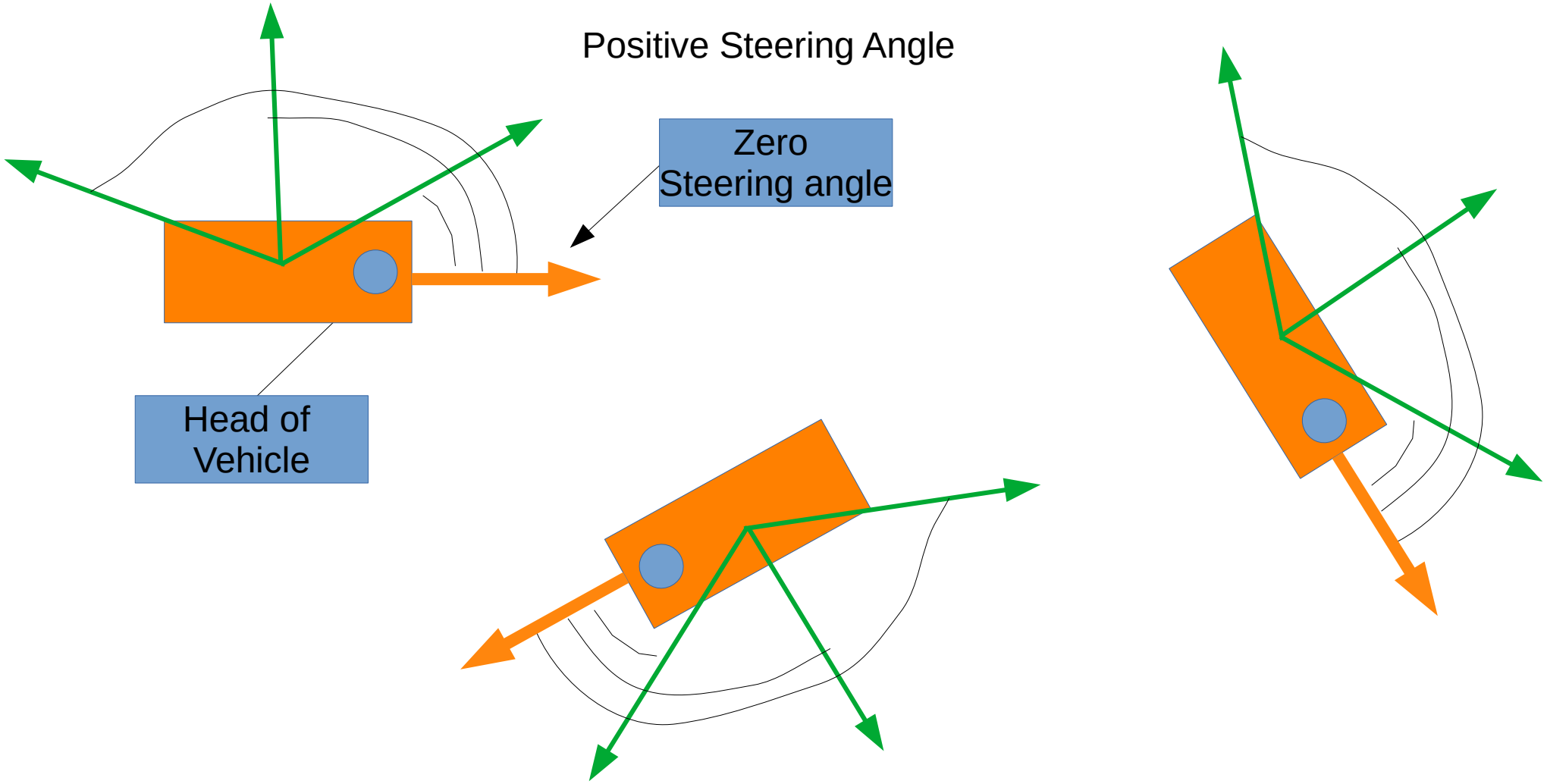


Checking Variable Value Interpretations

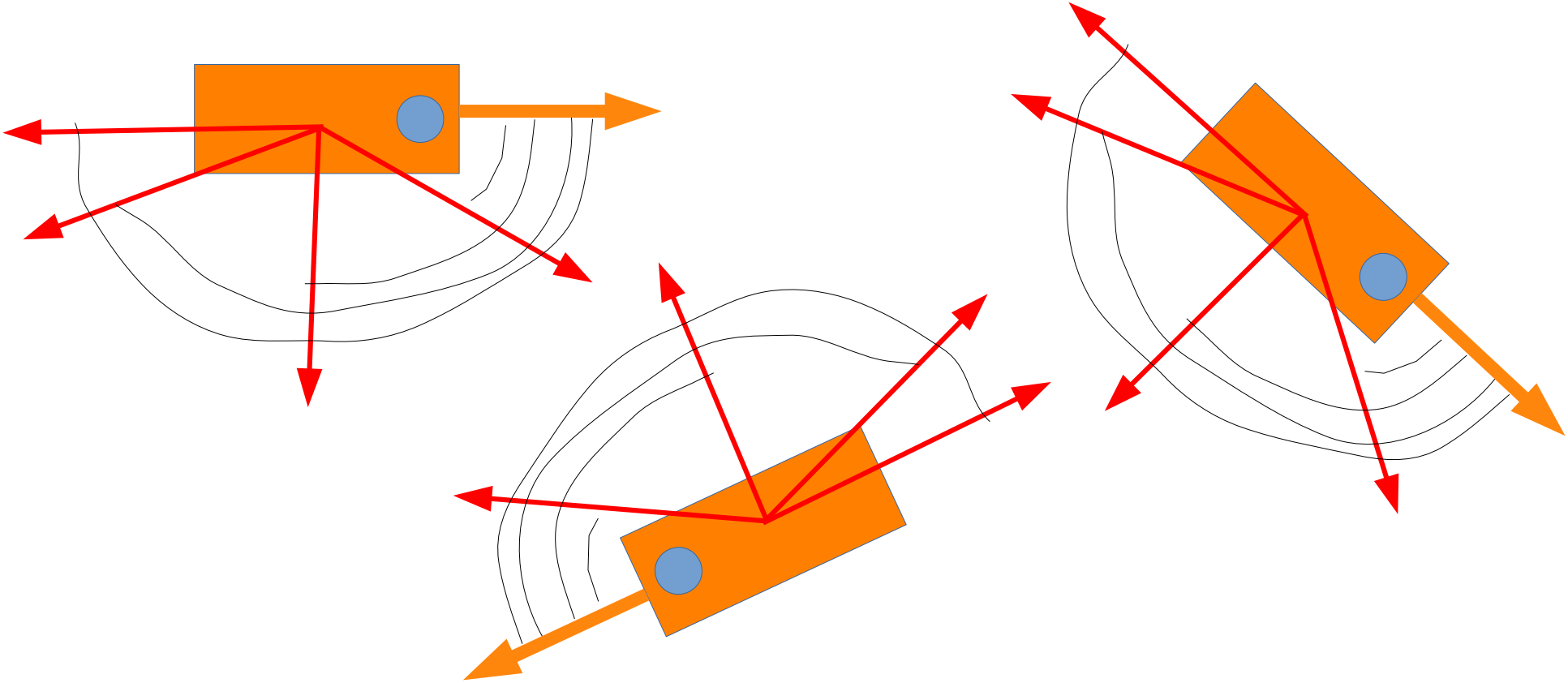
- David Bayani, day9 month6 year2019,
- Confirming with Edward Ahn the interpretation of some model variables
- Notation based on document shared by Edward Ahn via Google drives located at: https://drive.google.com/drive/u/0/folders/1aDm1zsTE-CtOc5XqnI16WZmrB5S4q0T9
- This document UUID: ebc32bd9-3d55-4013-9d02-a6897cac2c09


Y axis
POSITIVE YAW
Zero Yaw
X axis

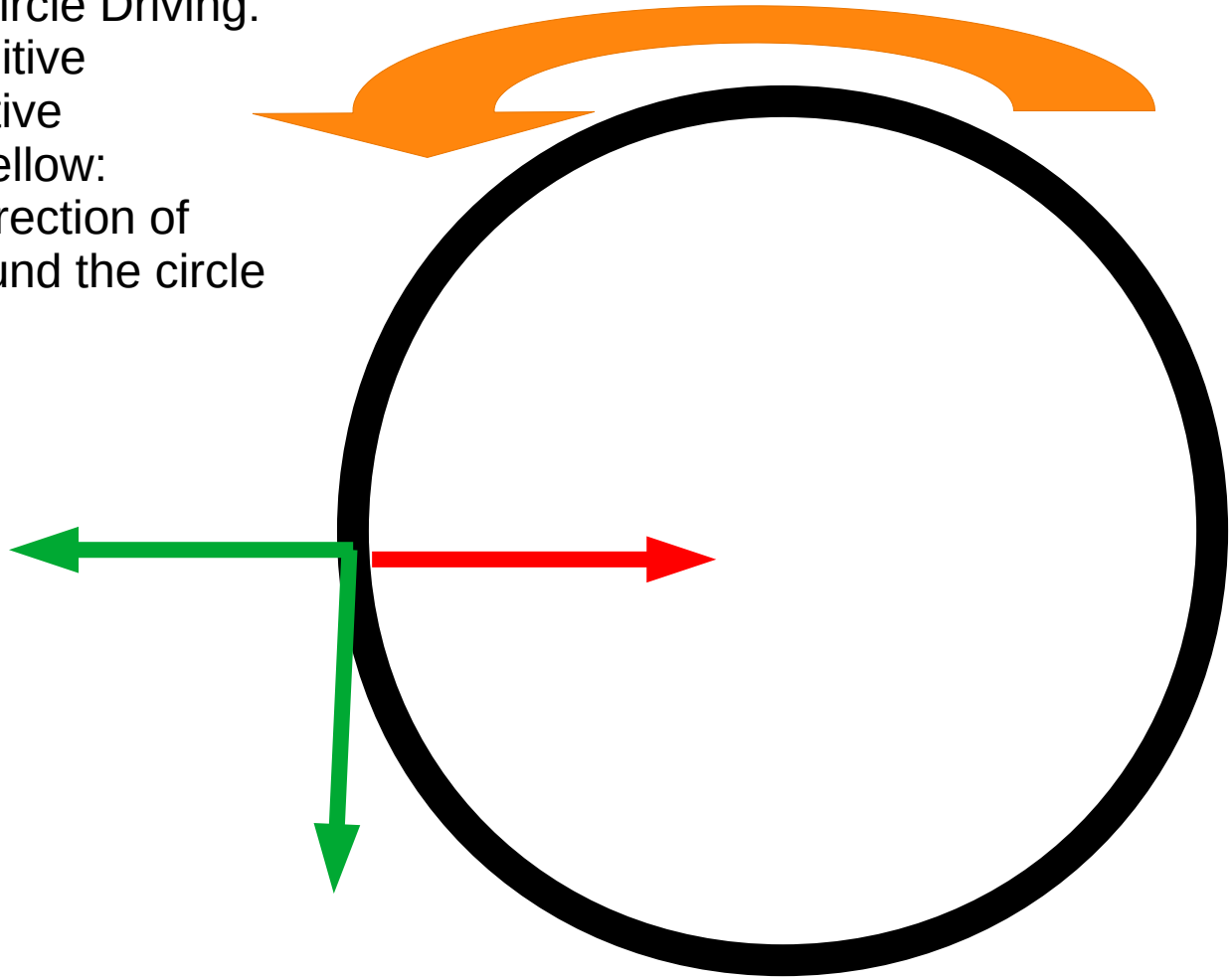

Y axis
NEGATIVE
YAW
Zero Yaw
X axis

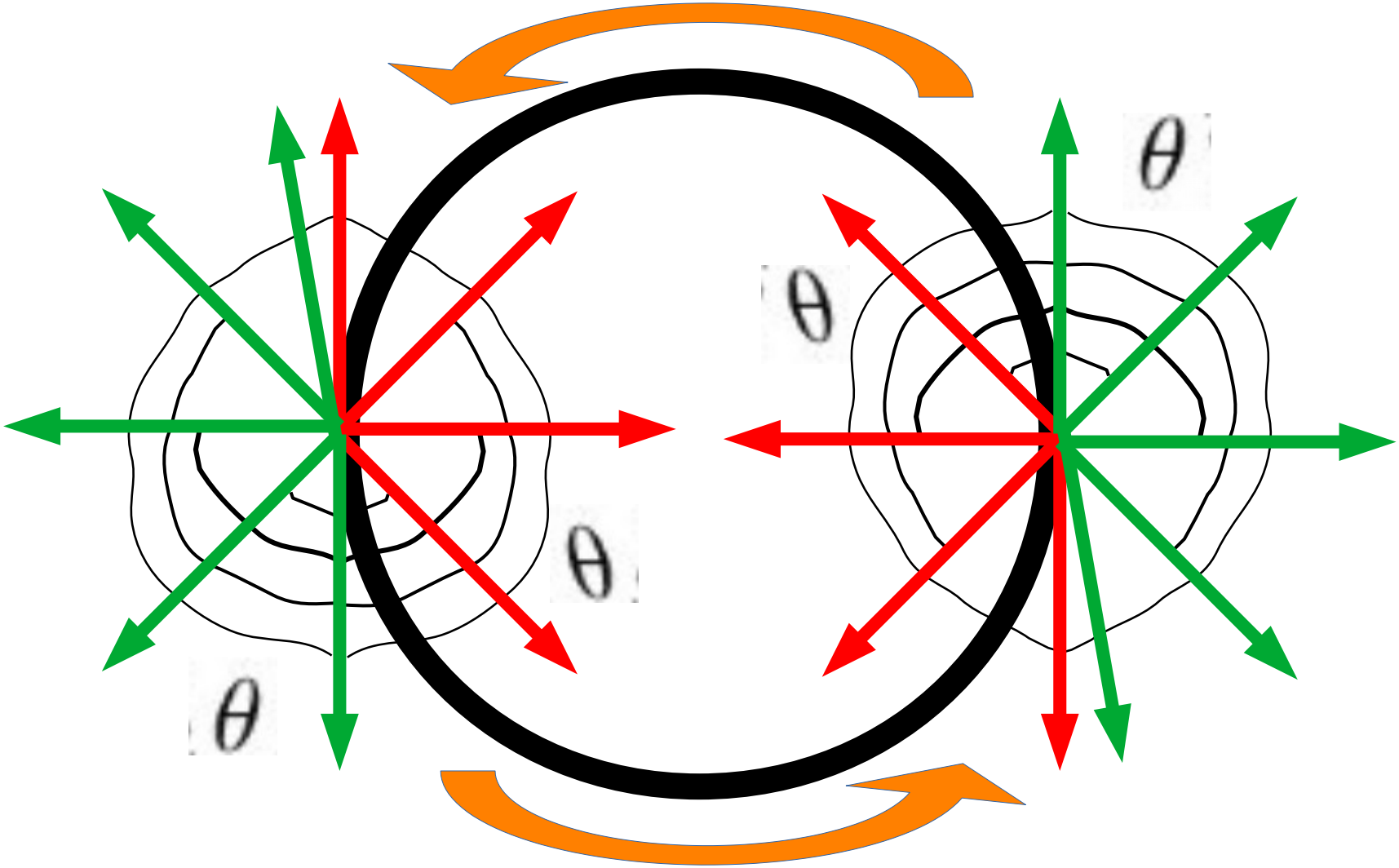

Positive Steering Angle
Zero
Steering angle
Head of
Vehicle

Negative Steering Angle



Theta for Circle Driving:
Green: Positive
Red: Negative
Orange / Yellow:
intended direction of
driving around the circle




θ
θ
θ
θ