EPJ Web of Conferences **258**, 09003 (2022) https://doi.org/10.1051/epjconf/202225809003
A Virtual Tribute to Quark Confinement and the Hadron Spectrum (vConf21)

A machine learning pipeline for autonomous numerical analytic continuation of Dyson-Schwinger equations

Andreas Windisch[1,2,4,*], *Thomas* Gallien[3,4,**], and *Christopher* Schwarzlmüller[3,4,***]

[1]Department of Physics, Washington University in St. Louis, MO 63130, USA
[2]Know-Center GmbH, Inffeldgasse 13, 8010 Graz, Austria
[3]Silicon Austria Labs GmbH, Inffeldgasse 33, 8010 Graz, Austria
[4]RL Community, AI AUSTRIA, Wollzeile 24/12, 1010 Vienna, Austria

Abstract. Dyson-Schwinger equations (DSEs) are a non-perturbative way to express n-point functions in quantum field theory. Working in Euclidean space and in Landau gauge, for example, one can study the quark propagator Dyson-Schwinger equation in the real and complex domain, given that a suitable and tractable truncation has been found. When aiming for solving these equations in the complex domain, that is, for complex external momenta, one has to deform the integration contour of the radial component in the complex plane of the loop momentum expressed in hyper-spherical coordinates. This has to be done in order to avoid poles and branch cuts in the integrand of the self-energy loop. Since the nature of Dyson-Schwinger equations is such, that they have to be solved in a self-consistent way, one cannot analyze the analytic properties of the integrand after every iteration step, as this would not be feasible. In these proceedings, we suggest a machine learning pipeline based on deep learning (DL) approaches to computer vision (CV), as well as deep reinforcement learning (DRL), that could solve this problem autonomously by detecting poles and branch cuts in the numerical integrand after every iteration step and by suggesting suitable integration contour deformations that avoid these obstructions. We sketch out a proof of principle for both of these tasks, that is, the pole and branch cut detection, as well as the contour deformation.

1 Introduction

Computing Green's functions in the complex domain numerically turns out to be a tedious task, even when working in Euclidean space and when considering perturbative calculations at 1-loop level. The main complications one has to deal with are accounting for the poles and branch cuts that arise in the complex plane of the radial component of the square of the loop momentum, when the latter is expressed in hyper-spherical coordinates. Since the loop momentum is a 4-momentum, and because there is an external momentum flowing into the diagram, as well as the internal (loop-) momentum, two of the hyper-spherical angles can be integrated trivially, and one has to consider a two-fold, non-trivial integration over the square of the loop momentum, and over (the cosine of) the angle between the external and

*e-mail: windisch@physics.wustl.edu
**e-mail: thomas.gallien@silicon-austria.com
***e-mail: christopher.schwarzlmueller@silicon-austria.com

the internal momentum. As discussed e.g. in [1], branch cuts are induced by the angular integration and in conjunction with the current value of the external and internal momentum square, and they appear in the complex plane of the radial integration variable. Because these branch cuts often intersect the positive real axis, the original integration contour of the radial part, that starts at the origin (or a small infrared (IR) cut-off) and runs along the positive real half-axis to some ultraviolet (UV) cut-off, has to be deformed around the obstructive branch cut to connect the origin and the UV cut-off, and without picking up the residue of any pole in the plane that could appear between the deformed contour and the line between the origin and the UV cut-off. Put into other words, the integration contour has to be continuously deformable from the original contour to the new, deformed contour that avoids the branch cuts. While for perturbative loop diagrams usually a full, explicit form of the integrand is known, this normally does not hold true for DSEs (see e.g. [2] for a thorough treatment of the DSE formalism), as these are of the second Fredholm type and require e.g. iterative strategies to find a numerical solution. The dressing functions that result from a successful numerical solution of a Dyson-Schwinger equation appear on both sides of the equation, that is, explicitly on the left hand side, and in the integrand of the self-energy loop on the right hand side of the equation. The dressing functions are usually initialized to some real value, and given this as an initial condition, one could in principle analyze the integrand to find the poles and cuts for this given situation. One could then, in principle, find suitable contour deformations and avoid the cuts and poles, perform one iteration step, analyze the resulting, updated dressing functions, and try to deduce the newly arisen analytic obstructions for the next iteration step. Then, new contour deformations could be sought, and another iteration step could be performed. However, this is not feasible because of mainly two reasons. One, once we allow the dressing functions to take complex values for the external momentum square, the complex momenta will produce different branch cuts for each of its values, which results in a plethora of differently parametrized integration contours that have to be found. Usually one can cover all necessary contours by introducing a few parametrizations for different regions of the complex plane, but it is still a tedious task. The second reason why this is not feasible in practice is simply the fact that the system's solution is sought by an iterative approach, that is, after every iteration step such a full analysis would have to be conducted.

In these proceedings we propose a possible way to overcome these obstacles by introducing a machine learning pipeline that analyzes the current analytical properties of the integrand, and finds suitably deformed integration contours autonomously. In more concrete terms, here and in the following, we provide evidence for the feasibility of such a pipeline by providing proof (or proof of concept) for each and every of the following steps:

1. **PROOF (A):** Suitability of numerical contour deformation to compute the analytic properties of Green's functions

2. **PROOF OF CONCEPT (B):** Feasibility of determining the analytic properties of numerically computed functions by means of deep learning and computer vision

3. **PROOF OF CONCEPT (C):** Feasibility of autonomous contour deformation by means of a deep reinforcement learning agent

Here, the difference between **PROOF** and **PROOF OF CONCEPT** is, that the former has been successfully applied in a broad variety of contexts, which can thus be regarded as a successful strategy (and not as a proof in the mathematical sense). The latter means, that these steps have been introduced with the very application discussed here in mind, and we have published (and yet unpublished) knowledge that strongly suggests that the respective strategies will be successful. These proceedings are organized as follows. In the following

three sections we discuss the points **(A)**, **(B)**, and **(C)** in detail, and then conclude with a summary and outlook.

2 Contour deformation for numerical computation of Green's functions in the complex domain (A)

In order to establish numerical proof that such a deformation can be obtained, we look at this study [3], in which a correlator of so-called i-particles was considered. These i-particles are discussed in [4], where also an exact solution of the analytic properties of the correlator, that has been treated numerically in [3], is presented. The exact solutions of [4], and the numerical results found in [3] by means of contour deformation, are in perfect agreement. While [3] is concerned with showing agreement between the analytic and numerical approaches, there is an abundance of studies in which this strategy has been deployed, see e.g. [5–8] for an exemplary but incomplete list of such studies.

3 Pole and branch cut detection with deep learning (B)

In this section we will discuss the pole and branch cut detection in numerical data by means of deep learning. In principle, and at least as far as pole detection is concerned, one could also do this by more conventional means, such as computing contours around possible pole locations in order to check for a non-vanishing residue, see e.g. [9]. This doesn't resolve the detection of branch cuts, though, which is why we chose a different approach here. Using a so-called U-Net, that has been originally developed for biomedical image segmentation [10], we created a similar pipeline for detecting poles and branch cuts in snippets of numerically computed functions. The U-Net architecture is comprised of a convolutional neural network (CNN) [11] with a contracting and an expanding path. In between, feature maps are being cropped and copied from the contracting to the expanding path. While images (in our case numerically computed snippets of functions in the complex plane) are being fed into the network, at its other end a segmentation mask is being produced, which, again in our case, marks the position of the poles and branch cuts. The network is trained by comparing the segmentation mask produced by the model with the ground truth that is provided as a 'label' for this learning setup.

3.1 U-Net based segmentation

In order to generate a data set used for training the U-Net for pole and branch-cut segmentation, we again considered the problem of the i-particle correlator mentioned above [4]. The numerical data has been produced by solving the angular integral for complex internal momenta (y) and complex external momenta (x) in equation (22) of [3] for $x, y \in [-10, 10] \times i[-10, 10]$. Once the cut has been produced, we added a random number of poles (between 0 and 9), with random positions and random residues additively to the function with the cut. The segmentation mask is a matrix with the same size as the data with the cut and poles (128×128 pixels), with its entries being ternary numbers. Using base-3 covers the three possible cases of every positional value, namely the 'pixel' being a pole location, belonging to a branch cut, or being a point that is neither of these two cases. The pole locations can be directly entered in the segmentation mask once the random location has been produced for a given training sample. The branch cut position can be marked by using the cut parametrization equation (25) in [3].

In Figure 1 we show an example of a training sample used to train the U-Net.

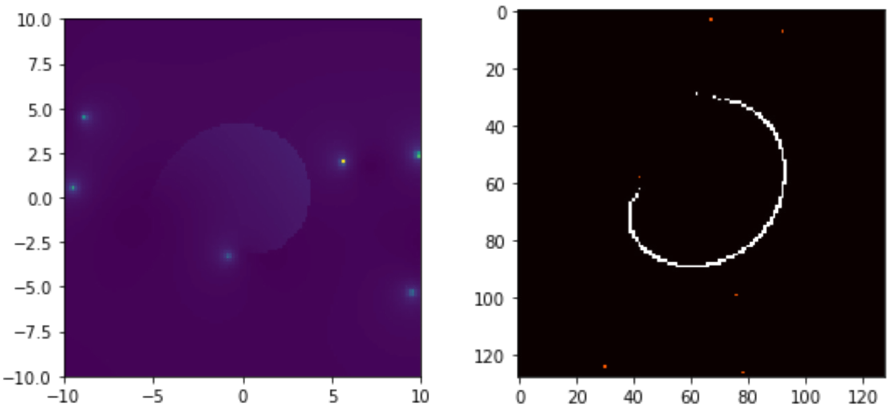

Figure 1. Left: A training sample with a branch cut and poles, produced as described in the text. The horizontal and vertical axis correspond to the real and imaginary part of the loop momentum square, y, expressed in GeV^2. Right: The corresponding segmentation mask. Note that the segmentation mask appears rotated here, because it has been expressed in pixel space rather than in the physical coordinate system.

In Figure 2 we show preliminary results of the first, trained U-Net using a training set generated as described above. We are currently investigating various extensions to our approach and will thus not provide any hyper parameters of the model at this point, as we are preparing a separate publication for the pole and cut detection with deep learning at the moment. The results of this first attempt look very promising, and suggest that this approach can be used in a Dyson-Schwinger setup to detect the cuts and poles in the integrand automatically. There is, of course, the problem of generalizability that is still to be addressed. In order to employ this approach to a Dyson-Schwinger setup, the variability encountered in the course of the iterative procedure must be captured by the distribution of the data set the U-Net has been trained with, otherwise it will most likely fail to recognize the respective patterns. This problem mostly affects the branch cuts, as poles are point-like and do not possess any relevant sub-structure, while cuts are essentially 1-dimensional objects. One possible way to generate cuts of arbitrary shape could be to produce a finite-length cut using a logarithm, and then apply an arbitrary distortion to the resulting cut-line to allow for arbitrary shapes. Another issue that is relevant in the context of the approach as described here is the sparsity of the ternary masks. Since there are only a few poles, and since also the branch cuts do not occupy a large amount of pixels, there is a class imbalance between pixels labeled as poles, cuts, or neither of the two. This skewness can become an issue when computing the loss function when training the system. To this end, one could compress the labels by deploying a compressive sensing approach, as discussed in [12, 13] in the context of biomedical imaging. We are currently also investigating this option, but do not have results yet.

4 Automated contour deformation through deep reinforcement learning (C)

In this section we discuss the next step in the pipeline, where we assume that the previous step has been successfully conducted and a segmentation mask that contains the information about the poles and cuts for the situation under consideration has been produced. Given this

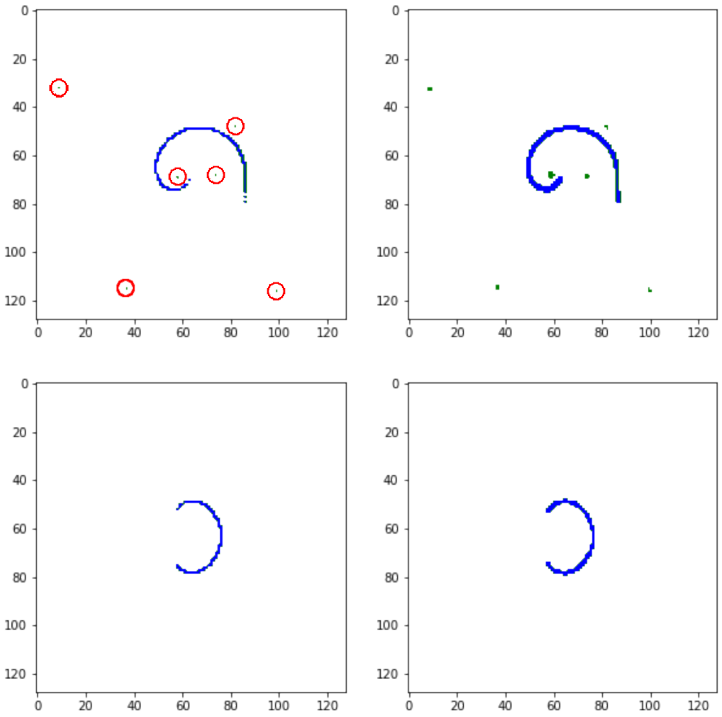

Figure 2. Preliminary results with the first U-Net attempt. Left column: Ground truth of segmentation masks. Since each pole only occupies a pixel in the mask, they have been marked by small circles for readability. Right column: Predictions produced by the trained U-Net.

information, we must now find a suitable integration contour that avoids the poles (i.e. is continuously deformable from the line along $\mathbb{R}_+ \cup \{0\}$), as well as the branch cuts. To this end we trained a deep reinforcement learning agent to perform this task (see [14]). Deep reinforcement learning is a learning approach inspired by behavioral science, in which a learning entity (the agent) observes and lives in an environment, conducts actions in this environment, and receives a reward or punishment by means of a scalar signal (see e.g. [15, 16] for an overview and a text book on the subject). The desired behavior of the agent can then be learned by maximizing the expected, cumulative reward, and the trained agent is then in possession of a policy, that takes in the state of the environment, and produces the most suitable action for a given situation.

4.1 Deep Reinforcement Learning agent for contour deformation

In [14], we considered a continuous scenario, in which we again used the i-particle correlator of [4]. This correlator possesses a branch cut, as well as two complex conjugate poles. Using a proximal policy optimization based approach [17], we successfully trained an agent that could produce meaningful contours in this environment, see Figure 3. The environment in this case consists of the complex plane of the radial integration variable, and the agent perceives this environment by means of a vector that holds relevant information, such as where the cut is, where the poles are located, in which direction the cut opens up, etc. In principle, one

could also train a deep reinforcement learning agent solely on the numerical data as input, however, we decided against this approach, as we would lose a substantial amount of control over the whole setup, which, in addition, would be much more involved as well. In order to deploy this approach for DSEs, the DRL agent will be re-implemented in a discrete setup, as the continuous environment is harder to solve and not necessary for the relatively small sizes of discrete points the correlator will be solved for.

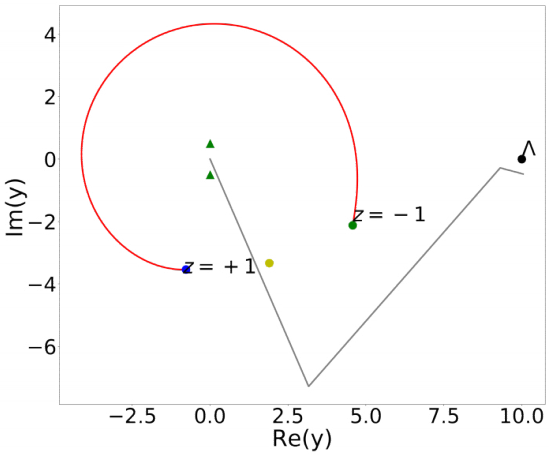

Figure 3. A successful contour found by the agent published in [14]. The red line represents the branch cut, the green triangles the complex conjugate poles located at $y = \pm\frac{i}{2}$. Note that this contour is continuously deformable from the line along the positive real half-axis to the cutoff, here represented by the Λ. The yellow point that is passed closely by the contour represents the direction of the opening of the cut, an information made available to the agent.

5 Conclusions and outlook

In the previous sections we provided proof, as well as proof of concept, for the main ingredients needed to set up an autonomous machine learning pipeline to perceive the analytic properties of the numerically computed, radial integration plane of the self energy loop in a DSE, as well as to produce the corresponding deformed integration contours. The first step, identifying the non-analyticities, will be implemented using either a U-Net architecture, or a regular CNN with dense labels produced by a compressive sensing approach. Once this information is available, it will be prepared for the DRL agent to observe, which will then produce a suitable contour. This could be done in every iteration step of the DSE, thereby solving the equation, while respecting the constraints as imposed by the non-analyticities. In order to study the quark propagator Dyson-Schwinger equation in Landau gauge in Euclidean space in the complex domain, we plan to deploy a suitable truncation scheme that establishes minimal constraints through the analytic properties of the self-energy loop.

Acknowledgements

AW acknowledges support through the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under Award Number #DE-FG-02-05ER41375.

References

[1] A. Windisch, M.Q. Huber, R. Alkofer, Acta Phys. Polon. Supp. **6**, 887 (2013), 1304.3642
[2] R. Alkofer, L. von Smekal, Phys. Rept. **353**, 281 (2001), hep-ph/0007355
[3] A. Windisch, R. Alkofer, G. Haase, M. Liebmann, Comput. Phys. Commun. **184**, 109 (2013), 1205.0752
[4] L. Baulieu, D. Dudal, M.S. Guimaraes, M.Q. Huber, S.P. Sorella, N. Vandersickel, D. Zwanziger, Phys. Rev. **D82**, 025021 (2010), 0912.5153
[5] C.S. Fischer, M.Q. Huber, Phys. Rev. D **102**, 094005 (2020), 2007.11505
[6] G. Eichmann, P. Duarte, M.T. Peña, A. Stadler, Phys. Rev. **D100**, 094001 (2019), 1907.05402
[7] A.S. Miramontes, H. Sanchis-Alepuz, Eur. Phys. J. **A55**, 170 (2019), 1906.06227
[8] A. Windisch, M.Q. Huber, R. Alkofer, Phys. Rev. **D87**, 065005 (2013), 1212.2175
[9] A. Windisch, Phys. Rev. **C95**, 045204 (2017), 1612.06002
[10] O. Ronneberger, P. Fischer, T. Brox, *U-net: Convolutional networks for biomedical image segmentation* (2015), 1505.04597
[11] K. Fukushima, Biological Cybernetics **36**, 193 (1980)
[12] Y. Xue, N. Ray, CoRR **abs/1708.03307** (2017), 1708.03307
[13] Y. Xue, G. Bigras, J. Hugh, N. Ray, IEEE Transactions on Medical Imaging **38**, 2632 (2019)
[14] A. Windisch, T. Gallien, C. Schwarzlmüller, Phys. Rev. E **101**, 033305 (2020), 1912.12322
[15] V. François-Lavet, P. Henderson, R. Islam, M.G. Bellemare, J. Pineau, CoRR **abs/1811.12560** (2018), 1811.12560
[16] R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, 2018)
[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, CoRR **abs/1707.06347** (2017), 1707.06347