Updates to the One-loop Provider NLOX

Diogenes Figueroa[a,1], Seth Quackenbush[a,2], Laura Reina[a,3], Christian Reuschle[b,4]

[a]*Physics Department, Florida State University, Tallahassee, FL 32306-4350, U.S.A.*

[b]*Department of Astronomy and Theoretical Physics, Lund University, SE-223 62 Lund, Sweden*

Abstract

Here we describe the 1.2 update to NLOX, a computer program for calculations in high-energy particle physics. New features since the 1.0 release and other changes are described, along with usage documentation.

Keywords: NLO QCD and EW automation, one-loop provider, higher-order calculations

---

NEW VERSION PROGRAM SUMMARY

Program Title: NLOX
Licensing provisions: CC BY NC 3.0
Programming language: `C++`. `Fortran` interface available, and `Fortran` compiler required for dependencies.
Does the new version supersede the previous version?: Yes
Reasons for the new version: New features; performance and stability improvements.
Summary of revisions: New expansion modes; color-correlated matrix elements; runtime parameter setting; improved cross-checks and stability.
Nature of problem: The computation of higher-order terms in the coupling expansion of Standard Model scattering amplitudes is required for precision studies in collider experiments. Techniques for computing the first corrections are well-known, and are now suited to automation. We wish to provide code that calculates virtual (one-loop) quantum chromodynamics and electroweak corrections for desired amplitudes using a package that automates the production of this code.
Solution method: We use `Python` scripts and a computer algebra system, FORM, to reduce virtual amplitudes to `C++` code and data based on Feynman rules of the Standard Model. The scripts perform a tensor decomposition of the one loop integral to reduce the amplitude to dependence on tensor integral coefficients. These coefficients are called at runtime by the provided library `TRed`, which performs tensor reduction into base (scalar) coefficients at runtime. The scripts identify repeated structures to be calculated once in the produced code for efficiency. The tensor reduction code is designed such that needed tensor coefficients need to be computed only once per evaluation of the desired amplitude, and are built recursively from other needed coefficients.
Additional comments including Restrictions and Unusual features: The code-producing scripts are not pro-

---

[1]dfigueroa@hep.fsu.edu

[2]squackenbush@hep.fsu.edu

[3]reina@hep.fsu.edu

[4]creuschle@hep.fsu.edu, christian.reuschle@thep.lu.se

vided in this release, only fixed libraries such as `TRed` and required interface code for pre-generated processes. Some processes are provided with this release, with others available upon request.
Required External Dependencies: QCDLoop 1.95, OneLOop 3.6 (available for download in utility tarball).
Required Compilers: `gcc` 4.6 or higher. Interface to certain optional libraries requires `C++` 11 support found in `gcc` 4.7 or higher.
Operating System: Linux, MacOS.

1. Introduction

NLOX is a one-loop provider that allows for the automated calculation of one-loop QCD and electroweak (EW) corrections to Standard Model (SM) scattering amplitudes. Based on Feynman-diagram techniques, it has been developed to optimize the manipulation of symbolic and tensor structures recurring in one-loop amplitudes. The first release of the code (v1.0) [1] consisted of the main tensor-integral reduction library, `TRed`, and several pre-generated processes. Further processes have been produced upon demand for various projects. All core functionalities of NLOX v1.0 have been presented in detail in Ref. [1] to which we refer.

Since v1.0 the code has seen a substantial number of important additions aimed at improving the numerical stability of the one-loop amplitudes and the interface to external codes. Among the most important additions are improved stability checks in the tensor reduction performed by `TRed` and new routines that check at runtime the IR pole structure of virtual amplitudes against the IR pole structure of the corresponding real emission, order by order in the QCD+EW couplings. These new IR checks are implemented using color-correlated amplitudes that are now also independently obtainable from the code. Given the impact that these improvements will have on the succesful use of the code, we have included them in the new NLOX v1.2 that is being released.

In the following, Sections 2 and 3 present a more technical description of all the code's new features and improvements, while future developments are outlined in Section 4.

2. New features

2.1. New modes for perturbative expansions

NLOX now has interface support for automatic compilation of all sub-amplitudes that contribute to the same overall power of α_s (QCD) and α_e (QED) in the interference between tree level and tree level or one-loop amplitudes for any given sub-process. This *α-mode* complement the existing possibility of compiling sub-amplitudes by specifying the g_s ($g_s^2 = 4\pi\alpha_s$) and e ($e^2 = 4\pi\alpha_e$) power of the interfering tree level and tree level or one-loop amplitudes [1].

The compiled result with a fixed power of both α_s and α_e can be accessed via the OLP function `NLOX_OLP_EvalSubProcess` by passing the corresponding string (`cp`) specifying the coupling-power combination. In a `C++` program this OLP function takes the form:

```
NLOX_OLP_EvalSubProcess(isub, type, cp, p, next, mu, rval, acc)
```

where the arguments follow the BLHA standard [2]:

- `isub`: Integer number specifying the index of the desired sub-process as defined in the sub-process list in `nlox_process.h`.
- `type`: Character string specifying the interference type, both `tree_tree` and `tree_loop` are supported.
- `cp`: Character string specifying the coupling-power combination of the interference type in question, both `gXeY_gX′eY′` and `asWaeZ` are supported.
- `p`: Array of double-precision numbers specifying the external momenta and masses, each momenta must be followed by its mass.
- `next`: Integer number specifying the number of external particles.
- `mu`: Double-precision number specifying the renormalization scale (in GeV).
- `rval`: Return value pointer, the container must allow for 3 double-precision numbers to be stored corresponding to double pole, single pole, and finite part.
- `acc`: Accuracy pointer, the container must allow for a single double-precision value to be stored.

In a `Fortran` program this OLP function takes extra arguments specifying the length of the interference type and of the coupling power combination:

```
NLOX_OLP_EvalSubProcess(isub, type, ltyp, cp, lcp, p, next, mu, rval, acc)
```

where the extra arguments of this function are:

- `ltyp`: Integer specifying the length of the interference type string.
- `lcp`: Integer specifying the length of the coupling-power combination string.

A detail worth mentioning is that for NLOX the two contributions `gXeY_gX′eY′` and `gX′eY′_gXeY` are different contributions that one must collect and sum if one decides to use the {`g,e`} specification for coupling powers, while in the α-mode all sub-amplitudes with the same overall power of α_s and α_e are summed over, and so these two will be summed up by the OLP function automatically.

2.2. Color-correlated output

NLOX is now capable of providing color-correlated Born-level matrix elements $\mathbf{B}_{ij}$. The normalization convention used is:

$$\mathbf{B}_{ij} =< \mathcal{M}^{C_1,\ldots,C_i,\ldots,C_j,\ldots,C_m}|(\mathbf{T}^a)_{C_iD_i}(\mathbf{T}^a)_{C_jD_j}|\mathcal{M}^{C_1,\ldots,D_i,\ldots,D_j,\ldots,C_m} >$$

where the matrices $\mathbf{T}^a$ denote the generators of the $SU(3)$ algebra and the indices $\{C_i, D_i\}$ are generic indices associated to the $SU(3)$ representation under which the i-th particle transforms. More specifically [3] the $(\mathbf{T}^a)_{C_iD_i}$ are defined as:

$$(\mathbf{T}^a)_{C_iD_i} = \begin{cases} (T^a)_{\alpha\beta} & \text{for initial quarks and final anti-quarks,} \\ -(T^a)_{\beta\alpha} & \text{for initial anti-quarks and final quarks,} \\ if^{cab} & \text{for gluons,} \end{cases}$$

where $(T^a)_{\alpha\beta}$ $(\alpha, \beta = 1, 2, 3)$ are the Gell-Mann matrices and f^{abc} $(a, b, c = 1, \ldots, 8)$ are the $SU(3)$ structure constants. For direct evaluation, there is a dedicated OLP level function that calls for the evaluation of color-correlated matrix elements:

```
NLOX_OLP_EvalSubProcess_CC(isub, type, cp, p, next, mu, borncc, acc)
```

which can only be called for Born-level matrix elements. The container that `borncc` points to must be able to store the uncorrelated Born-level matrix element as well as the $\mathbf{B}_{ij}$ components for which $j > i$, while the components with $i > j$ can be obtained via symmetry from the $\mathbf{B}_{ij}$. The first element in the container is reserved for the uncorrelated Born, and the subsequent $[n(n-1)]/2$ elements correspond to the components of $\mathbf{B}_{ij}$ using lexicographical order on (i, j).

For `Fortran` programs the OLP function takes extra arguments specifying the length of the interference type as well as the coupling power combination:

```
NLOX_OLP_EvalSubProcess_CC(isub, type, ltyp, cp, lcp, p, next, mu, rval, acc)
```

where the extra arguments of this function are:

- `ltyp`: Integer specifying the length of the interference type string.
- `lcp`: Integer specifying the length of the coupling-power combination string.

For debugging purposes the generation of color-correlated Born matrix elements can be deactivated in the input file by setting:

```
generateColorCorrelations = false
```

By default `generateColorCorrelations` is set to `true`.

2.3. Runtime amplitude-level pole checks

Routines that calculate the IR pole structure of the real emission associated to a given Born amplitude have been newly implemented within NLOX, but are only computed in the *α-mode* of coupling-power specification. The virtual IR poles are computed independently in the combination of loop and counterterm diagrams, and must cancel the real IR poles in the G_μ input scheme.[5] If real and virtual IR poles must cancel against each other, checking for the goodness of zero of their sum can be used to provide a check on the stability of the tensor reduction algorithm. The user has access to this new feature via the `acc` flag of the `NLOX_OLP_EvalSubProcess` function. This flag has different behavior depending on the particular type of `cp` the user specifies:

- `cp=gXeY_gWeZ`: `acc` is either 0 for success or -1 for failure, where failure is internally determined by `TRed` as described in 3.1.

[5]Because NLOX only allows nonzero masses for the b and t quarks, EW renormalization in the $\alpha(0)$ scheme formally produces IR poles arising from massless fermions that remain after combination with those arising from real radiation, and must be compensated externally. Thus, in the $\alpha(0)$ cheme, the IR pole-checking feature is only useful for strictly QCD corrections, while it works for both QCD and EW corrections in the G_μ scheme (`GmuScheme = true`).

- `cp=asWaeZ`: `acc` is -1 for a `TRed` failure. In case of a `TRed` success the `acc` flag returns the accuracy of the virtual IR single pole relative to the real IR single pole.

The calculation of the real IR poles is performed using the dipole-subtraction formalism and it requires the internal evaluation of color-correlated Born matrix elements [3]. Due to this fact if the generation of color-correlated amplitudes is turned off via the input file (see Sec. 2.2) this will also have the effect of turning off the amplitude level IR pole checks.

2.4. Runtime parameter setting

In order to change physical parameters at runtime, NLOX now supports the Les Houches [2] standard function `OLP_SetParameter()`:

```
NLOX_OLP_SetParameter(para, re, im, ierr)
```

whose arguments are:

- `para`: Character array, the name of the parameter to be changed. The currently supported names are:
  - `"alpha_e"`: α_e, the QED fine structure constant.
  - `"alpha_s"`: α_s, the QCD coupling constant.
  - `"mb", "mt", "mH", "mW", "mZ"`: masses of supported massive particles.
  - `"wH", "wW", "wZ"`: widths of supported particles. At present unstable fermions are not officially supported, and giving a nonzero width to an external particle will result in undetermined behavior.
  - `"nlf"` the number of light quark flavors to be used for renormalization purposes.
  - `"nhf"` the number of heavy quark flavors to be used for renormalization purposes.
  - `"mu"` the renormalization scale.

  It is recommended that one change these values only as often as necessary. For masses and widths in particular, changing the value can result in expensive reallocations and computations.
- `re`: Pointer to a double precision number holding the new real part of the value to be set.
- `im`: Pointer to a double precision number. This should be a dummy variable, as NLOX does not currently support setting parameters to complex values through this function.
- `ierr`: Pointer to an integer to store the return flag of the function. Set to 1 if successful and 2 if the name is unknown.

For `Fortran` programs the `SetParameter` function takes an extra argument:

```
NLOX_OLP_SetParameter(para, lpara, re, im, ierr)
```

where the extra argument of this function is:

- `lpara`: Integer specifying the length of the parameter name string.

3. Improvements and other changes

3.1. `TRed` *stability checks*

Since its original release, the `TRed` library has internally checked the stability of the tensor-integral reduction. Its primary check is to compare the single pole term (the coefficient c_1 in the expansion $c_2\epsilon^{-2} + c_1\epsilon^{-1} + c_0$ of a given tensor-integral coefficient) to an internal fast-evaluating library for coefficients that are infrared finite. This is a direct cross-check, that allows early correction of individual coefficients through recalculation at higher precision, and acts as a first-pass filter of potential instabilities, and has now been complemented by the IR-pole check described in Section 2.3. In this new release of NLOX we have improved its effectiveness as described in the following.

The driver of instability for the primary reduction method, Passarino-Veltman [4], is unstable matrix inversion involving the Gram matrix $G_{ij} \equiv 2p_i \cdot p_j$ for loop internal momenta p_i. In particular, the determinant of this matrix becomes small as it becomes degenerate, and as it appears explicitly in the denominator of reduction formulae, an imprecise cancellation in the numerator of the reduction can result in large incorrect results. To guard against this problem, in previous versions `TRed` has compared the size of the Gram determinant to an appropriate-scaling quantity, $E^{2(N-1)}$ for E a scale in the problem. However, one can improve on the sensitivity and specificity of this test with more information, reducing the failures for acceptable phase space points and catching others that may have been problematic. To this end, we have improved the test to more directly check the instability of the linear system.

A geometric interpretation of a determinant is that it is the volume of the parallelepiped spanned by the matrix's row (or column) vectors. As the vectors become linearly dependent, the volume of this space shrinks to zero, and the space spanned by them loses a dimension. Therefore, a direct measure of the smallness of the determinant compares it to the maximal volume such a space would have with the same-sized vectors, i.e., if they were orthogonal. Our new test compares $\det G_{ij}$ to $\prod_j |\vec{G}_i|$. The specificity of this test has allowed us to pass many configurations that would previously have been thrown out, even while preventing unhandled instabilities that could occur in collinear configurations for some processes.

Since `TRed` relies on external scalar integral libraries for the endpoint of its reduction, it is only as good as the precision of those libraries. In particular, in the small determinant case above, combinations of scalar integrals appearing may cancel in the limit of degenerate configurations. While `TRed` will attempt to do the reduction in higher precision, if the scalars only match to a lower precision, this can result in poor behavior regardless. Therefore we have reconfigured the installation of the default scalar library, OneLOop [5], to install quadruple-precision versions of its functions, and adjusted `TRed` to make use of them as appropriate.

3.2. Code size optimizations

Expressions resulting from the algebraic construction of the expression for a process squared amplitude naturally can be large for complicated processes, especially if many masses are involved or the theory has non-trivial expressions for interactions, as in the case of the broken electroweak theory. We have focused on simplifying expressions, especially finding overall factors, and finding common structures such that the resulting code size is smaller. These optimizations can result in a reduction of code size of as much as a factor of two in extreme situations compared to the version 1.0 release. These optimizations improve runtime speed as well if the speed is process-cache or memory-speed limited.

Since much of the resulting algebraic data has been removed from code generation entirely, and placed into process data files, this is the dominant contribution to a process package size. We have eliminated some empty files and redundant structures to further reduce the size of a process package.

4. Outlook

We have described the important changes to NLOX since its original release. The main focus has been on numerical stability. We have reorganized internal integral stability checks for accuracy and efficiency. We have also added a new stability check by implementing the dipole subtraction formalism, which contains the singular part of the real radiation contribution to next-to-lowest-order amplitudes and allows us to exploit the finiteness of combined next-to-lowest-order amplitudes to detect problems at the level of the final answer.

The implementation of the dipole formalism has also exposed a new functionality to the user, that of Born-level color-correlated amplitudes, which may be of use to users wishing to interface NLOX to Monte Carlo simulators requesting these amplitudes.

This release has also seen the implementation of runtime parameter setting, further implementing the Les Houches Accord next-to-lowest-order function interface and making the process of interfacing NLOX to external programs more convenient.

While this 1.2 release and its predecessor, 1.1, have focused on stability and ease of use, we anticipate the next releases will focus on speed and size efficiency, while also continuing to improve the ease of use and organization of the non-public process archive generation features so that the full package may be released.

To access the host `URL` for the NLOX package, please go to `http://www.hep.fsu.edu/~nlox/v1.2.0/`.

5. Acknowledgments

This work has been supported by the U.S. Department of Energy under grant DE-SC0010102. C.R. acknowledges support by the European Union's Horizon 2020 research and innovative programme, under grant agreement No. 668679. L.R. is grateful to the Aspen Center for Physics for the hospitality offered while parts of this work were being completed. The work performed at the Aspen Center for Physics is supported in part by the National Science Foundation under grant NSF PHY-1607611.

References

[1] S. Honeywell, S. Quackenbush, L. Reina, C. Reuschle, NLOX, a one-loop provider for Standard Model processes, Comput. Phys. Commun. 257 (2020) 107284. arXiv:1812.11925, doi:10.1016/j.cpc.2020.107284.

[2] S. Alioli, et al., Update of the Binoth Les Houches Accord for a standard interface between Monte Carlo tools and one-loop programs, Comput. Phys. Commun. 185 (2014) 560–571. `arXiv:1308.3462`, `doi:10.1016/j.cpc.2013.10.020`.

[3] S. Catani, M. H. Seymour, A General algorithm for calculating jet cross-sections in NLO QCD, Nucl. Phys. B485 (1997) 291–419, [Erratum: Nucl. Phys.B510,503(1998)]. `arXiv:hep-ph/9605323`, `doi:10.1016/S0550-3213(96)00589-5,10.1016/S0550-3213(98)81022-5`.

[4] G. Passarino, M. J. G. Veltman, One Loop Corrections for e+ e- Annihilation Into mu+ mu- in the Weinberg Model, Nucl. Phys. B160 (1979) 151–207. `doi:10.1016/0550-3213(79)90234-7`.

[5] A. van Hameren, OneLOop: For the evaluation of one-loop scalar functions, Comput.Phys.Commun. 182 (2011) 2427–2438. `arXiv:1007.4716`, `doi:10.1016/j.cpc.2011.06.011`.