YAM2: Yet another library for the $M_2$ variables using sequential quadratic programming

Chan Beom Park*

Center for Theoretical Physics of the Universe, Institute for Basic Science (IBS), Daejeon 34126, Korea

Abstract

The $M_2$ variables are devised to extend $M_{T2}$ by promoting transverse masses to Lorentz-invariant ones and making explicit use of on-shell mass relations. Unlike simple kinematic variables such as the invariant mass of visible particles, where the variable definitions directly provide how to calculate them, the calculation of the $M_2$ variables is undertaken by employing numerical algorithms. Essentially, the calculation of $M_2$ corresponds to solving a constrained minimization problem in mathematical optimization, and various numerical methods exist for the task. We find that the sequential quadratic programming method performs very well for the calculation of $M_2$, and its numerical performance is even better than the method implemented in the existing software package for $M_2$. As a consequence of our study, we have developed and released yet another software library, YAM2, for calculating the $M_2$ variables using several numerical algorithms.

Keywords— Collider physics, Kinematic variable, Constrained optimization, Sequential quadratic programming

*cbpark@ibs.re.kr
1 Introduction

The existence of invisible particles such as neutrino or dark matter candidate in the final state of physics processes poses a great challenge on physic analyses at hadron colliders, where the event information along the beam direction is limited, in particular. Detectors at collider experiments project the full phase space onto the space of visible particle momenta, and the projection is a non-invertible mapping. We can infer the existence of invisible particles only by the record of missing transverse momentum, or equivalently, the momentum imbalance of visible final-state particles. In response to the challenge, physicists have developed many useful methods and algorithms for extracting the information of the physics processes involved with the invisible particle as much as possible. As one of such endeavors, a set of kinematic variables, termed $M_2$, has been proposed for measuring the mass spectrum in the presence of invisible particles produced in a pair and providing a good approximation to the four-momenta of the invisible particles [1].

The $M_2$ variables are an extension of $M_T^2$ to the Lorentz invariant systems without projecting on the transverse plane. In the definition of $M_2$, the longitudinal momenta, as well as the transverse momenta of invisible particles, become the parameters of minimization under certain kinematic constraints. The value and the solution of $M_2$ are obtained from the optimality and feasibility conditions of the nonlinearly constrained minimization problem in mathematical optimization. The formulation and description of the $M_2$ variables will be given in Sec. 2.

Unlike simple kinematic variables such as the invariant mass of visible particles, where the variable definitions directly provide how to calculate them, the calculation of the $M_2$ variables is undertaken by employing numerical algorithms. It is due to the lack of analytic expressions for $M_2$, except for some special kinematic configurations. Currently, the only publicly available software package for calculating the $M_2$ variables is OPTIMASS [2].

As mentioned in the above, finding the $M_2$ value is to perform constrained minimization in essence, and there are a lot of numerical methods viable for the task. We will present a brief overview of some of such methods, the augmented Lagrangian method and the sequential quadratic programming, in Sec. 3. We have attempted to implement the numerical methods for comparing their performance of the $M_2$ calculation. As an outcome of our study, we release yet
another software library \textit{YAM2} for calculating \( M_2 \). The implementation of numerical algorithms in \textit{YAM2} and the benchmark study of comparing the algorithms are presented in Sec. 4. We find that the sequential quadratic programming method performs better than the other methods, including those implemented in \textit{OPTIMASS}. The instructions for installing and using \textit{YAM2} are described in Sec. 5, and then the last section is dedicated to summary and outlook.

2 The \( M_2 \) variables

In the physics process under consideration, if there exist particles moving beyond the outermost detector component without leaving a trace, it is recorded as the signal with missing energy. Reconstruction of missing energy events is one of the major obstacles to physics analyses at hadron colliders. The reconstruction is an attempt of inverse projection onto the full phase space, which is bound to be incomplete. Nevertheless, in many situations, it is possible to measure the mass spectrum or to reconstruct the missing energy events up to some ambiguity. One of the multitudes of methods aiming to resolve the missing energy problem is the method of the \( M_2 \) variables [1–3].

Before entering into the description of the \( M_2 \) variables, it would be better to start with looking into \( M_{T2} \), which is closely related to the definition of the \( M_2 \) variables and designed to tackle the same problem. The invention of the \( M_{T2} \) variable was devised to find a lower bound on the masses of superparticles decaying into the lightest neutral supersymmetric particle. Such examples include the pair productions of sleptons, \( \tilde{\ell}^+ + \tilde{\ell}^- \rightarrow \ell^+ \tilde{\chi}_1^0 + \ell^- \tilde{\chi}_1^- \) [4], charginos, \( \tilde{\chi}^+_1 + \tilde{\chi}^-_1 \rightarrow \pi^0 \tilde{\chi}^0_1 + \pi^0 \tilde{\chi}^0_1 \) [5,6], and gluinos, \( \tilde{g} + \tilde{g} \rightarrow q\bar{q}\tilde{\chi}^0_1 + q\bar{q}\tilde{\chi}^0_1 \) [7–10]. All of them fall into the symmetric decay topology of

\[
Y + \bar{Y} \rightarrow v_1(p_1)\chi(k_1) + v_2(p_2)\tilde{\chi}(k_2),
\]

(2.1)

where \( v_i \) are the sets of visible standard model particles, and \( \chi \) is the invisible particle. The method of the \( M_{T2} \) variable is indeed applicable to not only supersymmetric cases but any physics process that can be represented by the decay topology (2.1). It is defined as

\[
M_{T2} \equiv \min_{k_{1T}, k_{2T} \in \mathbb{R}^2} \left[ \max \left\{ M_T(p_{1T}, k_{1T}; \chi), M_T(p_{2T}, k_{2T}; \chi) \right\} \right]

\text{subject to } k_{1T} + k_{2T} = P_T,
\]

(2.2)

where \( P_T \) is the missing transverse momentum determined by the negative sum over all the visible particles momenta in the plane transverse to the beam axis,

\[
P_T = -\sum_i p_{iT}.
\]

(2.3)

If there are additional particles not involved with the hard process, such as initial state radiations, they are added to the sum as well. The invisible particle mass \( M_\chi \), which is unknown, is an input for the transverse masses,

\[
M^2_T(p_{iT}, k_{iT}; \chi) = \pi_T(p_i + k_i)^2.
\]

(2.4)

Here \( \pi_T \) is the projection operator from \((1+3)\)-dimensional space onto \((1+2)\)-dimensional one:

\[
\pi_T : p_i = (E_i, p_i) \rightarrow p_{iT} = (E_{iT}, p_{iT}),
\]

\[
\pi_T : k_i = (e_i, k_i) \rightarrow k_{iT} = (e_{iT}, k_{iT}),
\]

(2.5)
with $E_{iT} = (m_i^2 + \|p_{iT}\|^2)^{1/2}$ and $e_{iT} = (M_2^2 + \|k_{iT}\|^2)^{1/2}$. The $M_{T2}$ distribution has an endpoint at the parent particle mass $M_Y$ for the true value of $M_\chi$, that is,

$$M_{T2}(M_\chi = M_\chi^{\text{true}}) \leq M_Y. \quad (2.6)$$

Thus, it enables us to extract the mass spectrum information by identifying the position of the endpoint of the $M_{T2}$ distribution.

It does not stop there: the solution of the minimization in (2.2) provides an approximation to the transverse momenta of invisible particles,

$$\{\vec{k}_{iT}\} = \underset{k_{1T}+k_{2T}=P_T}{\text{argmin}} \left[ \max \left\{ M_T(p_{1T}, k_{1T}; M_\chi), M_T(p_{2T}, k_{2T}; M_\chi) \right\} \right]. \quad (2.7)$$

Then, for the approximation to the transverse momenta $\vec{k}_{iT}$, one can obtain the associated longitudinal momenta $k_{iL}$ of the invisible particles by using the on-shell mass relations of the parent particles:

$$(p_1 + \vec{k}_1)^2 = M_Y^2, \quad (p_2 + \vec{k}_2)^2 = M_Y^2. \quad (2.8)$$

It is called the $M_{T2}$-assisted on-shell (MAOS) method or the MAOS approximation for the invisible momenta [11]. The right-hand sides of the equations in the above need not be the parent particle mass. One can substitute the $M_{T2}$ value or the transverse mass $M_T$ in place of $M_Y$ [12–15]. In this way, the MAOS method serves an approximate reconstruction of the center-of-mass frame event by event. It can be used to measure the particle properties, such as spins and helicities, besides the mass spectrum information.

Except for some special cases [8, 9, 13, 16–22], the analytic expression for the $M_{T2}$ variable and the solution (2.7) in general cases are unknown. Instead, one makes use of numerical optimization algorithms to calculate them.\footnote{In this article, the term “optimization” is interchangeable with “minimization.”} Note that $M_{T2}$ can be written as

$$M_{T2} = \min_{k_{1T} \in \mathbb{R}^2} \left[ \max \left\{ M_T(p_{1T}, k_{1T}; M_\chi), M_T(p_{2T}, (e_{2T}, P_T - k_{1T}); M_\chi) \right\} \right], \quad (2.9)$$

where $k_{2T}$ has been eliminated by the constraint on the missing transverse momentum, and $e_{2T} = (M_2^2 + \|P_T - k_{1T}\|^2)^{1/2}$. Therefore, finding the $M_{T2}$ value for a given event corresponds to performing \textit{unconstrained} minimization on a function of two variables, $k_{1T} = (k_{1x}, k_{1y})$. In the past, a combination of Migrad and Simplex methods included in the \texttt{Minuit2} library [23] of \texttt{ROOT} was used for the minimization [24]. The Migrad algorithm is a variable-metric method that depends on the first derivative of the objective function to be minimized [25, 26], while the Simplex algorithm, also called the Nelder-Mead method [27], does not use the derivative information. For a brief review of the methods, see Appendix A of Ref. [2]. Afterward, it was realized that $M_{T2}$ could be understood as the boundary of the mass region $(M_\chi, M_Y)$ consistent with the minimal kinematic constraints, \textit{i.e.}, the on-shell mass relations,

$$(p_1 + k_1)^2 = (p_2 + k_2)^2 = M_Y^2, \quad (2.10)$$

and the missing transverse momentum constraint [28]. The kinematically allowed region for $k_{1T}$ forms an ellipse, and $M_{T2}$ can be calculated by investigating the scaling behavior of two ellipses for the two decay chains in (2.1). In the implementation, it is used the bisection method in conjunction with the Sturm sequence for the quartic polynomial, converted from two quadratic equations, to test if two ellipses intersect. See Subsec. 2.3 and Appendix A of Ref. [28] for the
Here the subscript \( \text{min} \) indicates that the minimization over four variables. However, it again turned out to be that constrained minimization is equivalent to \( \chi \), the parent particles, as given in (2.10). Thus, we can impose the on-shell mass relation as a constraint in addition to the missing transverse momentum constraint. Thus, finding the decay topology (2.1), the longitudinal momenta and transverse masses \( M(T) \) becomes identical to the transverse masses \( M(T(p_{1T}, k_{1T}) \).

There are additional considerations worth examining further. If the physics process has the decay topology (2.1), the longitudinal momenta \( k_{iz} \) satisfy the on-shell mass relations for the parent particles, as given in (2.10). Thus, we can impose the on-shell mass relation as a constraint in addition to the missing transverse momentum constraint,

\[
M_{2C} \equiv \min_{k_1, k_2 \in \mathbb{R}^3} \left[ \max \left\{ M(p_1, k_1; M_\chi), M(p_2, k_2; M_\chi) \right\} \right]
\]

subject to \( k_{1T} + k_{2T} = P_T \),

(2.12)

Here the subscript \( \text{C} \) denotes the additional constraint on the invariant masses. We do not use \( M_Y \) since it is unknown and is to be determined. It is similar to the constrained mass variable in Refs. [31, 32], where a constraint on the mass difference \( M_Y - M_\chi \) has imposed further. After eliminating \( k_{2T} \) by using the missing transverse momentum as above, it has the form of \( \chi \) minimized over four variables. However, it again turned out to be that \( M_{2C} \) is equivalent to \( M_{T2} \) [1, 22]. Recall that \( M_{T2} \) can be defined as the boundary of the consistent mass region, subject to the kinematic constraints, including the on-shell mass relations for the parent particles in Eq. (2.10). In other words, \( M_{T2} \) already utilizes the on-shell mass information in an implicit way. Together with the fact that \( M_2 = M_{T2} \), we also find that \( M_{2C} = M_2 \).

The \( M_2 \) variable becomes distinct from \( M_{T2} \) when taking into account on-shell intermediate particles in the decay chains:

\[
A_1 + A_2 \rightarrow a_1 B_1 + a_2 B_2 \\
\rightarrow a_1(p_{a_1}) b_1(p_{b_1}) C_1(k_1) + a_2(p_{a_2}) b_2(p_{b_2}) C_2(k_2).
\]

(2.13)
In the final state, $a_i$ and $b_i$ are visible particles, and $C_i$ are invisible particles responsible for the missing energy. $B_i$ are the intermediate states decaying to $b_iC_i$. As in Ref. [1], we assume that the decay chains are symmetric, i.e., $M_{A_1} = M_{A_2} = M_A$, $M_{B_1} = M_{B_2} = M_B$, and $M_{C_1} = M_{C_2} = M_C$. For the sake of notational simplicity, we express the visible momenta by
\[
    p_i \equiv p_{a_i} + p_{b_i}, \quad q_i \equiv p_{b_i}.
\]
(2.14)

Note that we have already used the relation that $k_1^2 = k_2^2 = M_C^2$, since $M_C$ enters as an input to the invariant masses of visible + invisible particle systems in $M_2$. Then, the remaining on-shell mass constraints yet to be used for the decay topology (2.13) are
\[
    (p_1 + k_1)^2 = (p_2 + k_2)^2, \quad (q_1 + k_1)^2 = (q_2 + k_2)^2.
\]
(2.15)

Depending on the on-shell mass constraint to use (or not to use) for minimization, there are four types of the $M_2$ variables:

\[
    M_{2XX} \equiv \min_{k_1, k_2 \in \mathbb{R}^3} \left[ \max \left\{ M(p_1, k_1; M_C), M(p_2, k_2; M_C) \right\} \right]
\]
subject to $k_{1T} + k_{2T} = P_T$,

\[
    M_{2CX} \equiv \min_{k_1, k_2 \in \mathbb{R}^3} \left[ \max \left\{ M(p_1, k_1; M_C), M(p_2, k_2; M_C) \right\} \right]
\]
subject to \[
    \begin{align*}
    k_{1T} + k_{2T} &= P_T, \\
    (p_1 + k_1)^2 &= (p_2 + k_2)^2, \\
    (q_1 + k_1)^2 &= (q_2 + k_2)^2.
\end{align*}
\]
(2.16)

\[
    M_{2XC} \equiv \min_{k_1, k_2 \in \mathbb{R}^3} \left[ \max \left\{ M(p_1, k_1; M_C), M(p_2, k_2; M_C) \right\} \right]
\]
subject to \[
    \begin{align*}
    k_{1T} + k_{2T} &= P_T, \\
    (q_1 + k_1)^2 &= (q_2 + k_2)^2, \\
    (p_1 + k_1)^2 &= (p_2 + k_2)^2.
\end{align*}
\]
(2.17)

\[
    M_{2CC} \equiv \min_{k_1, k_2 \in \mathbb{R}^3} \left[ \max \left\{ M(p_1, k_1; M_C), M(p_2, k_2; M_C) \right\} \right]
\]
subject to \[
    \begin{align*}
    k_{1T} + k_{2T} &= P_T, \\
    (q_1 + k_1)^2 &= (q_2 + k_2)^2.
\end{align*}
\]
(2.18)

In each definition, the first subscript of $M_2$ refers to the on-shell mass constraint on the parent particles $A_i$, and the second does to that on the intermediate particles $B_i$. The subscript “$C$” (“$X$”) means that the corresponding constraint is (not) imposed. One can also construct the $M_2$ variables for other subsystems of the visible particles, e.g., the subsystem of $\{b_i\}$. Here we confine ourselves to the $\{a, b_i\}$ system. One can see that $M_{2XX} = M_2$ in (2.11), and $M_{2CX} = M_{2C}$ in (2.12). Therefore, we have
\[
    M_{2XX} = M_{2CX} = M_{T2}.
\]
(2.20)

On the other hand, it has been found that $M_{2XC}$ and $M_{2CC}$ are different from $M_{T2}$, and further, they have the following hierarchy [1]:
\[
    M_{2XX} = M_{2CC} \leq M_{2XC} \leq M_{2CC} \leq M_Y
\]
(2.21)

for events with the decay topology (2.13). The last inequality holds for $M_C = M_C^{\text{true}}$. The consequence of the hierarchical structure is that the event densities of the $M_{2XC}$ and $M_{2CC}$ distributions will be populated more toward the parent particle mass than that of $M_{T2}$.  

5
In the left panel of Fig. 1, we show the $M_2$ distributions for the di-leptonic top pair process
\[ t + \bar{t} \rightarrow bW^+ + \bar{b}W^- \rightarrow b\ell^+\nu + b\ell^−\bar{\nu} \quad (2.22) \]
at truth level with the LHC beam condition. All distributions of the $M_2$ variables have endpoint shape near the parent particle mass $m_t$, and the distributions of $M_2^{XX}$ and $M_2^{CX}$ are identical to each other. In particular, we can confirm the hierarchy relation (2.21) by seeing the peak locations.

Another important outcome of the $M_2$ variables is that we can obtain an approximation for the invisible particle momenta as the consequence of the minimization:
\[
\{ \tilde{k}_i \} = \arg \min_{k_1, k_2 \in \mathbb{R}^3} \left[ \max \left\{ M(p_1, k_1; M_C), M(p_2, k_2; M_C) \right\} \right] \quad \text{subject to constraints.} \quad (2.23)
\]
It appears to be taking a similar approach to the MAOS method: cf. Eq. (2.7). However, the methodology is substantially different. In the MAOS method, one obtains the transverse components of the invisible momenta from the $M_T^2$ solution and then employs the on-shell mass relations to get the longitudinal components. On the other hand, in the $M_2$ variables, the on-shell mass relations act as the constraints in the minimization step, which eventually emits the components of invisible momenta all together. In result, the invisible momentum solutions of $M_2^{XC}$ and $M_2^{CC}$ are uniquely determined, while each longitudinal momentum of the MAOS method is determined up to two-fold ambiguity. The comparison study of the $M_2$ and MAOS methods has been performed in Ref. [3], where it is shown that the $M_2$ variables provide the better approximation to the invisible momenta. In the right panel of Fig. 1, we show the reconstructed $W$-boson mass using the $M_2$ solution to the neutrino momenta,
\[
\tilde{M}_W^2 = (q_i + \tilde{k}_i)^2. \quad (2.24)
\]
\footnote{There is a combinatorial ambiguity on pairing the $b$ quarks and charged leptons. In this article, we do not concern the ambiguity, but use the correct pair. Interestingly, the $M_2$ variables can be used to resolve the combinatorial problem [33].}
The peak of the $\tilde{M}_W$ distribution for $M_{2CC}$ is located near the $W$-boson mass, thus allowing additional mass measurement, that is, measuring $m_W$ using $\tilde{M}_W$ as well as $m_t$ using $M_2$.

3 Algorithms for constrained minimization

As in the case of $M_{72}$, the calculation of the $M_2$ variables resorts to numerical algorithms because analytic expression for general cases is unknown. The calculations for $M_2$ are essentially multi-dimensional constrained minimization, except for $M_{2XX}$ in Eq. (2.16). Currently, the only publicly available software package for calculating $M_2$ is OPTIMASS [2]. The algorithm of choice in OPTIMASS is the augmented Lagrangian (AUGLAG) method [34,35] with the Migrad and Simplex algorithms from the Minuit2 library of ROOT. In this section, we present a brief review of the formulation of constrained minimization problems and numerical algorithms for calculating the $M_2$ variables.

A general formulation for constrained minimization problems is

$$
\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \begin{cases} 
c_i(x) = 0, \\
d_j(x) \geq 0,
\end{cases}
$$

(3.1)

where $f(x)$ is an objective function, and $c_i(x)$ and $d_j(x)$ are equality and inequality constraints, respectively. The objective function and constraints are all smooth, real-valued functions: $f, c_i, d_j : \mathbb{R}^n \rightarrow \mathbb{R}$. In the case of $M_2$ variables, the objective function is

$$
f(k) = \max \left\{ M(p_1, k_1; M_C), M(p_2, k_2; M_C) \right\},
$$

(3.2)

and $c_i$ are on-shell mass relations. The $M_2$ variables do not have inequality constraints, $j \in \emptyset$. The unknown variables are $k_{1x}, k_{1y}, k_{1z},$ and $k_{2z}$, after eliminating $k_{2x}$ and $k_{2y}$ by using the missing transverse momentum condition. If a certain point $x^*$ is in the feasible set for the system of constraints,

$$
x^* \in \Omega = \{ x \in \mathbb{R}^n | c_i(x) = 0, d_j(x) \geq 0 \},
$$

(3.3)

and $f(x) \geq f(x^*)$ for a neighborhood of $x^*$ on the feasible set $\Omega$, $x^*$ is termed a local solution (or minimizer) of the problem. For the $M_2$ variables, the $M_2$ value corresponds to $f(\hat{k}_i)$, with $\hat{k}_i$ being the $M_2$ solution to the invisible momenta event-by-event. We refer to Ref. [36] for the more complete explanations of the constrained optimization problems and conditions for the local solution.

As mentioned earlier, the software package OPTIMASS employs the AUGLAG method, which is well known and widely used algorithm for constrained optimization problems. The AUGLAG method constructs a Lagrangian function on top of the quadratic penalty function, in which the penalty terms are the squares of constraints. For the description of the AUGLAG method, we begin by considering the penalty method for solving constrained optimization problems. Ignoring the inequality constraints, the quadratic penalty function for the problem (3.1) is given by

$$Q(x; \mu) = f(x) + \frac{\mu}{2} \sum_i c_i^2(x),$$

(3.4)

where $\mu > 0$ is the penalty parameter. The constraint violations, or infeasibility, are penalized by increasing $\mu$. We can minimize the penalty function $Q(x; \mu)$ instead of the objective function.
By constructing the penalty function, the problem (3.1) has been transformed from constrained to unconstrained minimization. In the implementation of the algorithm, we increase the $\mu$ value, and then seek the solution $x^*$ of $Q(x; \mu)$ in each iteration. The iterations proceed until some convergence test has been satisfied. However, it turns out that the solution $x^*$ does not satisfy the feasibility conditions $c_i(x^*) = 0$, but instead, it is given by

$$c_i(x^*) \approx \frac{\lambda_i^*}{\mu}$$  \hspace{1cm} (3.5)

for some fixed parameter $\lambda_i^*$. The feasibility conditions are satisfied in the limit of $\mu \to \infty$, but then the penalty function will be dominated by huge penalty terms that may lead to inaccuracy in the numerical calculation. Therefore, we need an algorithm where the solutions more nearly satisfy the constraints even for moderate values of $\mu$.

The AUGLAG method extends the quadratic penalty method by adding Lagrangian multipliers to the objective function,

$$L_A(x, \lambda; \mu) = f(x) - \sum_i \lambda_i c_i(x) + \frac{\mu}{2} \sum_i c_i^2(x).$$  \hspace{1cm} (3.6)

The Lagrangian multiplier vector $\lambda$ is an input at each iteration step, not to be determined as in the conventional Lagrangian multiplier method. For a point $x_k$, the feasibility condition is now given by

$$c_i(x_k) \approx -\frac{1}{\mu_k} \left( \lambda_i^* - \lambda_i^k \right),$$  \hspace{1cm} (3.7)

thus, the infeasibility will be much smaller than $1/\mu_k$ for $\lambda_i^k \to \lambda_i^*$. Namely, the convergence of the AUGLAG method can be assured without taking $\mu$ to be increasing indefinitely. At the $k$th step of iterations, the Lagrangian multiplier vector is updated as

$$\lambda_i^k = \lambda_i^{k-1} - \mu_{k-1} c_i(x_{k-1})$$  \hspace{1cm} (3.8)

using the approximation solution $x_{k-1}$ at the previous step.

The most popular software package for the practical AUGLAG method is LANCELOT [37, 38], and the implementation of OPTIMASS is inspired in part by the package. Note that, in each iteration, unconstrained minimization is performed for given values of $\lambda$ and $\mu$ updated by the AUGLAG algorithm. In OPTIMASS, a combination of Migrad and Simplex algorithms is adopted for the unconstrained minimization: Simplex finds a local solution at first, and then Migrad makes use of the solution as an initial guess for searching a minimum.

We have briefly looked over the AUGLAG method, which is the chosen algorithm of OPTIMASS for calculating $M_2$. However, we note that the AUGLAG method is not the only available way for performing constrained minimization. There are other well-known methods such as the sequential quadratic programming (SQP) [39–43] and the interior-point method [44, 45] that can be applied to the same problem. In Ref. [2], it was shown that the implementation using the AUGLAG method effectively achieved the minimization well. But, it did not mention the particular reason for the choice of the algorithm, nor show benchmark studies for comparisons to other methods. This motivates us to study other numerical methods for calculating the $M_2$ variables.

For comparison with the AUGLAG method, we employ the SQP method. We have chosen the SQP method since it is known to be the most efficient unless the number of free variables is

---

4$\lambda^*$ is the Lagrange multiplier vector satisfying the first-order necessary conditions for optimality, also known as Karush-Kuhn-Tucker conditions. See Eq. (3.10).
too large, and it requires fewer function evaluations in comparison with AUGLAG methods [36]. We will compare their performance in the next section.

The SQP method can be derived by applying Newton’s method for solving nonlinear equations to the condition for the local solution of constrained minimization problems. Here we closely follow the description of the SQP method given in Ref. [36]. Considering only equality constraints, we define the Lagrangian function for the problem (3.1) as

$$\mathcal{L}(x, \lambda) = f(x) - \sum \lambda_i c_i(x).$$  \hspace{1cm} (3.9)

For continuously differentiable functions $f$ and $c_i$, and a local solution $x^*$ to the problem, there exists a Lagrangian multiplier vector $\lambda^*$ such that the following conditions are satisfied:

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0,$$

$$c_i(x^*) = 0.$$  \hspace{1cm} (3.10)

They are known as the first-order necessary conditions for optimality, or the Karush-Kuhn-Tucker (KKT) conditions.

Newton’s method is an algorithm widely used for finding the roots of real-valued equations. It successively improves the approximation to the roots using the first derivatives of the equations. For a continuously differentiable vector function $F(x) : \mathbb{R}^n \rightarrow \mathbb{R}^n$, the iteration is given by

$$x_{k+1} = x_k + p_k,$$  \hspace{1cm} (3.11)

where the search direction $p_k$ is the solution of

$$J(x_k)p_k = -F(x_k).$$  \hspace{1cm} (3.12)

Here $J(x)$ is the Jacobian matrix of $F(x)$, $J(x) = \nabla F(x)$.

We now consider an equality-constrained minimization problem,

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad c(x) = 0,$$  \hspace{1cm} (3.13)

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$. The KKT conditions (3.10) of the problem can be written as

$$F(x, \lambda) = \begin{pmatrix} \nabla f(x) - A(x)^T \lambda \\ c(x) \end{pmatrix} = 0.$$  \hspace{1cm} (3.14)

It has $n+m$ unknown parameters, $x$ and $\lambda$. $A$ is an $m \times n$ matrix, the Jacobian of the constraints:

$$A(x) = (\nabla c_1(x), \ldots, \nabla c_m(x))^T.$$  \hspace{1cm} (3.15)

And, the Jacobian matrix of $F(x, \lambda)$ is given by

$$J(x, \lambda) = \begin{pmatrix} \nabla^2_{xx} \mathcal{L}(x, \lambda) & -A(x)^T \\ A(x) & 0 \end{pmatrix}.$$  \hspace{1cm} (3.16)

For the iterate $(x_k, \lambda_k)$, applying Newton’s method to the KKT conditions (3.14) gives us the next iterate,

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} + \begin{pmatrix} p_x \\ p_\lambda \end{pmatrix},$$  \hspace{1cm} (3.17)
where \( p_x \) and \( p_{\lambda} \) are the solutions of

\[
\begin{pmatrix}
\nabla^2_{xx} L(x, \lambda) & -A(x)^T \\
A(x) & 0
\end{pmatrix}
\begin{pmatrix}
p_x \\
p_{\lambda}
\end{pmatrix}
= -\begin{pmatrix}
\nabla f(x) - \nabla A(x)^T \lambda \\
c(x)
\end{pmatrix}.
\tag{3.18}
\]

It turns out that the iterate generated by the application of Newton’s method is equivalent to modeling the problem (3.13) using the quadratic subproblem at the iterate \((x_k, \lambda_k)\),

\[
\min_{p \in \mathbb{R}^n} \left[ L(x_k, \lambda_k) + \nabla_x L(x_k, \lambda_k)^T p + \frac{1}{2} p^T \nabla^2_{xx} L(x_k, \lambda_k) p \right]
\]

subject to \( A(x_k) p + c(x_k) = 0 \).
\tag{3.19}

The objective function in the above is a quadratic approximation of the Lagrangian function. Therefore, it can be argued that we have replaced the constrained minimization problem (3.13) by the problem of minimizing the quadratic approximation of the Lagrangian function subject to the linear approximation of the constraints. This is the SQP framework, which enables us to derive the SQP algorithm for nonlinearly constrained minimization problems. The SQP framework can easily be extended to optimization problems with inequality constraints.

The SQP framework has a unique solution \((p_k, \lambda_k')\) satisfying

\[
\nabla^2_{xx} L(x_k, \lambda_k) p_k + \nabla f(x_k) - A(x_k)^T \lambda_k' = 0,
\]

\[
A(x_k) p_k + c(x_k) = 0,
\tag{3.20}
\]

if the following assumptions hold:

(a) The Jacobian matrix of the constraints \( A(x) \) has full row rank, i.e., the constraint gradients are linearly independent.

(b) The Hessian matrix of the Lagrangian function \( \nabla^2_{xx} L(x, \lambda) \) is positive definite on the tangent space of the constraints, \( d^T \nabla^2_{xx} L d > 0 \) for all \( d \neq 0 \) such that \( A(x)d = 0 \).

After solving the equations, the new iterate \((x_{k+1}, \lambda_{k+1})\) are given by

\[
x_{k+1} = x_k + p_k, \quad \lambda_{k+1} = \lambda_k'.
\tag{3.21}
\]

As we have seen, the SQP method needs the derivative information of the objective and constraint functions. In the case of the AUGLAG method, the dependence on the derivatives follows the minimization algorithm to solve the subproblem. For example, as mentioned earlier, the Simplex algorithm does not use the derivative information, while the Migrad algorithm depends on the gradient of the objective function. The gradients of functions in the \( M_2 \) variables can be analytically obtained event by event. To present the gradients, we define invariant mass functions as

\[
M_{(i)}^2 \equiv (P_i + k_i)^2.
\tag{3.22}
\]

The gradients of the mass functions at \( k = (k_{1x}, k_{1y}, k_{1z}, k_{2z}) \) are given by

\[
\nabla_k M_{(1)}(P_1, k_1) = \frac{1}{M_{(1)} e_i} \begin{pmatrix}
E_1 k_{1x} - e_1 P_{1x} \\
E_1 k_{1y} - e_1 P_{1y} \\
E_1 k_{1z} - e_1 P_{1z} \\
0
\end{pmatrix},
\]
\[ \nabla_k M_2(P_2, k_2) = \frac{1}{M_2 c_2} \begin{pmatrix} E_2(k_{1x} - P_{x}) + e_2 P_{2x} \\ E_2(k_{1y} - P_{y}) + e_2 P_{2y} \\ 0 \\ E_2k_{2z} - e_2 P_{2z} \end{pmatrix}. \] (3.23)

Using the invariant mass functions, the gradient of the objective function of the \( M_2 \) variables (3.2) is written as follows:

\[ \nabla_k f(k) = \begin{cases} \nabla_k M_1(p_1, k_1) & \text{if } M_1(p_1, k_1) \geq M_2(p_2, k_2), \\ \nabla_k M_2(p_2, k_2) & \text{if } M_1(p_1, k_1) < M_2(p_2, k_2). \end{cases} \] (3.24)

Furthermore, the on-shell mass constraints can be expressed by

\[ c_A = M_1(p_1, k_1) - M_2(p_2, k_2), \]
\[ c_B = M_1(q_1, k_1) - M_2(q_2, k_2), \] (3.25)

for \( A_i \) and \( B_i \), respectively. Therefore, it is straightforward to obtain the gradients of the constraints by using \( \nabla_k M_{(i)} \) in (3.23) as well,

\[ \nabla_k c_A = \nabla_k M_1(p_1, k_1) - \nabla_k M_2(p_2, k_2), \]
\[ \nabla_k c_B = \nabla_k M_1(q_1, k_1) - \nabla_k M_2(q_2, k_2). \] (3.26)

Meanwhile, the exact form of the Hessian matrix of the Lagrangian \( \nabla^2_{xx} \mathcal{L} \) is not necessary if we employ quasi-Newton approximation, where only the gradient information is required. In the quasi-Newton method, the approximation of the Hessian matrix \( B_k \) satisfies the so-called secant condition,

\[ B_{k+1}(x_{k+1} - x_k) = \nabla \mathcal{L}(x_{k+1}, \lambda_{k+1}) - \nabla \mathcal{L}(x_k, \lambda_k). \] (3.27)

The strategy is that we compute \((x_{k+1}, \lambda_{k+1})\) for a given \( B_k \), and then update \( B_k \) as

\[ B_{k+1} = B_k + U_k. \] (3.28)

To determine \( U_k \) uniquely, it is necessary to impose additional conditions, which differ by algorithms. The most popular and powerful method is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, and we will use the algorithm in our implementation for the \( M_2 \) variables.

### 4 Implementation and performance comparisons

We are now in a position to describe our implementation for calculating the \( M_2 \) variables. For employing the numerical minimization algorithms, we use NLopt, a publicly available software library for nonlinear optimization [46]. Since NLopt includes several numerical algorithms with a uniform interface, we have also tested the other algorithms not examined in the previous section. Among them, we find that the SQP and AUGLAG methods with the BFGS update perform the best in terms of accuracy and speed for calculating \( M_2 \). In NLopt, the SQP algorithm is based on the implementation described in Refs. [47, 48], and the implementation of the AUGLAG algorithm follows Refs. [37, 49]. Our studies on the numerical algorithms have brought yet another library for the \( M_2 \) variables, which we dub YAM2 [50]. The instructions for installation and usage of YAM2 are given in Sec. 5. For comparison, we have also added an interface for using the AUGLAG method with the Nelder-Mead Simplex algorithm to the library.

Currently, the set of the algorithms employed in YAM2 is as follows:
• **SQP + BFGS**: SQP algorithm with the BFGS update,

• **AUGLAG + BFGS**: AUGLAG algorithm with the BFGS update,

• **AUGLAG + Simplex**: AUGLAG algorithm with the Nelder-Mead Simplex method,

• a combination of the above algorithms.

In the combination of the algorithms, the $M_2$ solution is given by comparing the minima found by **SQP + BFGS** and **AUGLAG + BFGS**. If both algorithms have failed to find a minimum, the **AUGLAG + Simplex** algorithm is used. We used the combination of the algorithms for the distributions in Fig. 1.

We must choose an initial guess for the unknown variables $k = (k_{1x}, k_{1y}, k_{1z}, k_{2z})$ as an input to the algorithms. The initial guess is important because the numerical minimization algorithms perform the best in general if the guess is near the solution, or at least, is not very far from the solution. We have tested two kinds of initial guesses: one is that $k_{1x} = p_{1x}/2$, $k_{1y} = p_{1y}/2$, and $k_{1z} = k_{2z} = 0$, and the other is the momentum configuration that minimizes the total invariant mass of the final state, i.e.,

$$\arg\min_{k_1, k_2} (p_1 + p_2 + k_1 + k_2)^2 \text{ subject to } k_{1T} + k_{2T} = P_T. \quad (4.1)$$

It corresponds to the solution of $\hat{s}_{\text{min}}$ in Ref. [51]. We find that both guesses work well, but the latter is slightly better to find the minimum. In **YAM2**, we use the solution of the total invariant mass given above as the initial guess for all the numerical algorithms.

One of the most important parameters that can be adopted by the user input is the error tolerance. It serves as a stopping criterion for the iterations of the algorithms as well as a measure of the error relative to the solution. The tolerance can be set for the absolute or relative values of objective and constraint functions. One subtlety for setting the tolerance is that the objective and constraint functions of the $M_2$ variables are mass-dimensionful quantities whose scales vary event by event. In order to remove the scale dependence, we rescale all the masses and momenta of particles by a scale parameter $s$,

$$p_i \rightarrow s^{-1} p_i, \quad q_i \rightarrow s^{-1} q_i, \quad P_T \rightarrow s^{-1} P_T, \quad MC \rightarrow s^{-1} MC, \quad (4.2)$$

so that all the quantities become dimensionless. After the algorithm has completed minimization, the solution will be scaled back to have the right mass dimension: $M_2 \rightarrow sM_2$ and $\hat{k} \rightarrow s\hat{k}$. In **YAM2**, we have heuristically taken the scale parameter $s$ for a given event to be

$$s = 8\sqrt{E_1^2 + E_2^2 + \|P_T\|^2 + 2M_C^2}, \quad (4.3)$$

where $E_i^2 = p_i^2 + \|p_i\|^2$ are the squared energies of visible particles. On the rescaled objective and constraint functions, we set the tolerance conditions as follows:

- for constraints $c_i(x)$,

  $$|c_i(x_k)| < \varepsilon, \quad (4.4)$$

- for objective function $f(x)$,

  $$|f(x_k) - f(x_{k-1})| < \varepsilon \times 10^{-3}$$

  or

  $$|f(x_k) - f(x_{k-1})| < \frac{1}{2} \left| \frac{f(x_k) + f(x_{k-1})}{2} \right| \times \varepsilon \times 10^{-3} \quad (4.5)$$

  at the $k$th step.
In *YAM2*, the default value of the $\varepsilon$ parameter is $10^{-3}$. Users can feed a different value to that. On occasion, the numerical algorithms fail to find a minimum. In this case, we find that loosening the tolerance condition is helpful. Whenever the algorithm throws failure, we increase the $\varepsilon$ parameter to be ten times larger and then restart the algorithm.

It would also be wise to set a maximal number of iterations in order to avoid increasing the execution time indefinitely. The maximal number is set to be 5,000. However, in our experience of numerical studies, a large number of iterations is a typical symptom that the algorithm is failing to find the solution. In this situation, it is encouraged to adjust the tolerance or initial guess rather than increasing the maximal number of iterations.

To demonstrate and compare the performance of the algorithms, we consider the dileptonic top pair process of \((2.22)\), which has the decay topology of \((2.13)\). We have generated Monte Carlo event samples using Pythia 8 [52], and analyzed the parton-level data. The center-of-mass energy of proton-proton collision has been set to be $\sqrt{s} = 13$ TeV. The total number of analyzed event samples is 200k.

In Fig. 2, we show the differences between the $M_2$ variables calculated with the SQP and the AUGLAG methods. In the case of $M_{2XX}$, the two methods are the same because there is no on-shell mass constraints. In the other cases, the $M_2$ values mostly match within 1–2 GeV, while the SQP method is slightly better for $M_{2CX}$ and $M_{2CC}$. Here, saying the better means that the method finds the deeper local minimum. We also find that tolerance values smaller than $\varepsilon = 10^{-3}$ do not improve the result much, so it appears to be an effective choice. However, it is always worthwhile to check the result by changing the tolerance before producing the final result in practical physics analyses.

To compare the result with *OPTIMASS*, we show the differences between the $M_2$ variables calculated by using the SQP method and *OPTIMASS* in Fig. 3. We have used the *OPTIMASS* version 1.0.3 with the default setup of parameters included in the package. The results mostly match to
each other, and deviations are only within 1 GeV. For $M_{2CC}$, the SQP method finds a slightly better minimum in some events. Therefore, we find that the numerical methods employed in YAM2 perform well enough.

We now consider the computational cost of the $M_2$ calculations using the numerical algorithms. In real situations, we may have to deal with a tremendous amount of data for physics analyses. The computational cost is of particular importance, as the integrated luminosity of the current LHC experiment increases by order of magnitude, and the High-Luminosity LHC project is on the horizon [53]. Given limited computing resources and human time, a cost-effective way without damaging or deteriorating the results will be the most preferable. Studies of numerical algorithms should be accompanied with a measure of the execution time. In Fig. 4, the accumulated execution time of calculating the $M_2$ variables is exhibited. For a fair comparison, we have used the same routines for parsing event data, and the execution time has been measured by using the \texttt{std::chrono} library of C++. We did not use run-time parallelization libraries. For 200k events, the calculation of $M_{2CC}$ using the SQP method with the BFGS update takes about 25 seconds, while it does about 10 minutes when using \texttt{OPTIMASS}. Thus, the SQP method implemented in YAM2 is faster than \texttt{OPTIMASS} by $O(20)$-times. The combination of the SQP and AUGLAG methods takes about 1 minute, which is still $O(10)$-times faster than \texttt{OPTIMASS}. We have repeated the time measurement a thousand times and found that the speed upgrade is stable. Fig. 4 also shows that the SQP method is faster than the AUGLAG method, and the derivative-dependent algorithm such as the BFGS is faster than the derivative-free one for all the cases. We have used the analytic expressions for the gradients given in Sec. 3 for the derivative-dependent algorithms.\footnote{In \texttt{OPTIMASS}, the gradients are calculated numerically by using finite difference method instead of the analytic expression.}

Another mean for comparing numerical algorithms is to measure the number of function evaluations. We have counted the number of objective function evaluations for each event, and the result is shown in Fig. 5. We can see that the number for the SQP method is much less than that of AUGLAG method. The methods adopting the BFGS update evaluate the objective function $O(10)-O(100)$ times per event, while the number of evaluations is an order of magnitude larger in the case of the AUGLAG method with the Simplex algorithm. In \texttt{OPTIMASS}, the number $\gtrsim O(10^4)$ is much larger than the others. We have also measured the amount of system memory used while executing the analysis codes. For calculating $M_{2CC}$, the executables from the YAM2 codes consumed about 25 MB memory for processing whole 200k events, while
the OPTIMASS codes consumed about 160 MB. Therefore, we conclude that the implementation of YAM2 can calculate the $M_2$ variables in a much faster and more cost-effective way.

5 Installation and usage

YAM2 is free software under the license specified in the source code. It is distributed via [https://github.com/cbpark/YAM2](https://github.com/cbpark/YAM2).

For compiling and linking the source code, it is required to have a C++ compiler, supporting the features of the C++17 revision, and the NLopt library. Detailed instructions for installing NLopt by building the source code are given in Ref. [54]. In some Linux distributions, it can be installed by using system package manager. We have tested our codes with the NLopt version 2.6.2.

The source code of the YAM2 library can be built by using the build automation tool make. If the path to NLopt is /usr/local, append the path to the make command:

```
NLOPT=/usr/local make
```

The command will build all the source codes, and then generate a static library file, libYAM2.a, in the lib directory. If the shared library is necessary, run make lib. See Makefile for the detail of the compilation flags and path settings. The header and library files can also be installed to the other destination path outside the build directory. If the path to be installed is /usr/local, run the command as follows.

```
DESTDIR=/usr/local make install
```

The interfaces for using YAM2 are defined in the header file yam2.h. Users have to add the header to their analysis code through include directive.

```
#include <yam2.h>
```

The type signature of the function for calculating $M_{2\text{CC}}$ can be seen in the following function declaration.

```
std::optional<M2Solution> m2CSQP(
    const std::optional<InputKinematics> &inp,
    double eps = EPS, int neval = NEVAL);
```

The name of the function is descriptive. It will calculate $M_{2\text{CC}}$ using the SQP method. For $M_{2\text{XC}}$, the function to use is m2XCSQP. The function for calculating $M_{2\text{CC}}$ using the AUGLAG
method with the BFGS update is \texttt{m2CCAugLagBFGS}. The corresponding function using the combination of the SQP and AUGLAG methods is named \texttt{m2CC}. In the function declaration given above, one can see that the return type of the function is \texttt{std::optional} of \texttt{M2Solution}. The class template \texttt{std::optional} causes a null value if the function has failed, or otherwise, it returns the contained value, that is, \texttt{M2Solution} in our case. The function fails if the input is incorrect or the function has eventually failed to find a minimum. Once the calculation of the function is successful, the result can be extracted by the \texttt{value} method of \texttt{std::optional}.

\begin{verbatim}
const auto m2sol = yam2::m2CCSQP(input.value());
if (!m2sol) {
    std::cerr << "Failed.\n";
} else {
    std::cout << "M2CC = " << m2sol.value().m2() << 'n'
              << "solution:\n"
              << " k1: " << m2sol.value().k1() << 'n'
              << " k2: " << m2sol.value().k2() << 'n';
}
\end{verbatim}

As can be seen in the code snippet, the \texttt{M2Solution} class contains three methods: \texttt{m2} for the \(M_2\) value, \texttt{k1} and \texttt{k2} for the \(M_2\) solution to the invisible particle momenta. All the functions and classes are in the namespace of \texttt{yam2}. Once the minimization is successful, the \(M_2\) solution can be used to calculate other collider variables such as \(M_{AT}\) in Ref. [55], after converting it into a suitable object. For example, the instance of \texttt{TLorentzVector} in \texttt{ROOT} can be constructed from the solution as follows.

\begin{verbatim}
const auto k1 = m2sol.value().k1(), k2 = m2sol.value().k2();
const TLorentzVector inv1{k1.px(), k1.py(), k1.pz(), k1.e()};
const TLorentzVector inv2{k2.px(), k2.py(), k2.pz(), k2.e()};
\end{verbatim}

There are three inputs to the functions for calculating \(M_2\). The first one is an instance of \texttt{InputKinematics}, which is for the particle momentum configuration of the given event. It can be constructed by using the \texttt{mkInput} function.

\begin{verbatim}
std::optional<InputKinematics> mkInput(
    const std::vector<FourMomentum> &as,
    const std::vector<FourMomentum> &bs,
    const TransverseMomentum &ptmiss, const Mass &minv);
\end{verbatim}

Here \texttt{as} and \texttt{bs} correspond to the four-momenta of the visible particles \(a_i\) and \(b_i\). See the decay topology given in (2.13). The convention is

\begin{equation}
\texttt{as} = (p_{a_1}, p_{a_2}), \quad \texttt{bs} = (p_{b_1}, p_{b_2}).
\end{equation}

We stress that the order of the particle momenta should be set with care since it is not checked by the program: \(a_i\) must be produced before having \(b_i\) in the decay chain. In addition to them, users have to insert the missing transverse momentum and the invisible particle mass into \texttt{ptmiss} and \texttt{minv}, respectively. Schematic structures of the momentum classes are

\begin{verbatim}
class FourMomentum = FourMomentum (E, P_x, P_y, P_z),
class TransverseMomentum = TransverseMomentum (P_x, P_y).
\end{verbatim}

See \texttt{momentum.h} for the class definitions of \texttt{FourMomentum, TransverseMomentum, and Mass}. The input momentum configuration should be validated before substituting it into the functions for calculating \(M_2\). An example code snippet using the \texttt{mkInput} is given below.
```cpp
const auto input =
    yam2::mkInput({a1, a2}, {b1, b2}, ptmiss, yam2::Mass{m_invis});
if (!input) {
    std::cerr << "Invalid input.\n";
}
const auto m2sol = yam2::m2CCSQP(input.value());
```

The other optional inputs to the m2CCSQP function in the above are the tolerance (eps) and the maximal number of iterations (neval). These will be set to the default values defined in yam2.h unless users supply any input. In the current version of YAM2, their default values are \( \text{EPS} = 10^{-3} \) and \( \text{NEVAL} = 5000 \). We recommend users to read the example analysis code enclosed with YAM2, examples/m2.cc, before starting to write their analysis code for the \( M_2 \) variables.

Finally, we show an example command for building an analysis code using YAM2. Supposing that the name of the analysis code is m2.cc and the path to YAM2 is /usr/local, the command is as follows.

```
c++ -o m2.exe m2.cc -I/usr/local/include/YAM2 \\
    -L/usr/local/lib -lYAM2 -lnlopt
```

### 6 Summary and outlook

The \( M_2 \) variables are an extension of \( M_{T2} \) by promoting the transverse masses to Lorentz-invariant ones and making explicit use of on-shell mass relations. Depending on the on-shell mass relations, the \( M_2 \) variables have a hierarchical structure, which results in higher event density of the distribution near the parent particle mass. Finding the \( M_2 \) value and solution corresponds to solving a constrained minimization problem. Due to the lack of general analytic expression for the \( M_2 \) value and solution, the calculation relies on numerical minimization algorithms. Up to the present time, the only publicly available software package for calculating \( M_2 \) is OPTIMASS, where the augmented Lagrangian method with Migrad and Simplex algorithms has been employed. We note that there exist various numerical methods for solving constrained optimization problems. Among them, we have chosen the sequential quadratic programming method and the derivative-dependent BFGS algorithm. The method has been codified by using the implementation of the numerical algorithms in the NLopt library. The new library, YAM2, also includes other numerical algorithms for calculating \( M_2 \), such as the augmented Lagrangian method with the BFGS update.

By using YAM2, we performed a benchmark study for checking the performance of the numerical algorithms. It turned out that the sequential quadratic programming method correctly identified the local minimum for \( M_2 \), and it is more efficient than the other numerical methods, as well as OPTIMASS. We release YAM2 as publicly available free software, to help physicists interested in the \( M_2 \) variables for applying them to physics analyses.

There are many rooms for upgrading and adding more features to YAM2. We list a few of them:

- As supposed in Ref. [1], the \( M_2 \) variables can be defined for various subsystems of visible particles. It is also possible to calculate \( M_2 \) for different subsystems using YAM2, but the interface is not very transparent. We will improve the interface in the upcoming release of the upgrade.

- Other than the sequential quadratic programming, one of the most popular algorithms for constrained optimization problems is the interior-point method. As there exists a publicly
available software, Ipopt [56], for the interior-point method, it would be straightforward to test the method.

- Another interesting variable related with $M_2$ is $M_{2\text{Cons}}$, which is defined as [57,58]:

$$M_{2\text{Cons}} \equiv \min_{k_1, k_2 \in \mathbb{R}^3} \left[ \max \left\{ M \left( p_1, k_1; M_X \right), M \left( p_2, k_2; M_X \right) \right\} \right]$$

subject to

$$k_{1T} + k_{2T} = P_T,$$

$$(p_1 + p_2 + k_1 + k_2)^2 = M_X^2. \quad (6.1)$$

It can be useful for measuring the masses of on-shell intermediate particles produced in a pair from a resonance,

$$X \rightarrow Y + \bar{Y} \rightarrow v_1(p_1)\chi(k_1) + v_2(p_2)\bar{\chi}(k_2), \quad (6.2)$$

where the resonance mass $M_X$ is known $\text{a priori}$. The $M_{2\text{Cons}}$ distribution is bounded from above by $M_Y$. We can implement the $M_{2\text{Cons}}$ variable in the same way as in $M_2$.

- Since the source code of YAM2 is written in C++, it can directly be used in analysis codes written in C++. However, we expect that providing a C wrapper for YAM2 will greatly help to use it in the codes written in the other programming languages through foreign function interface.

**Acknowledgments**

The author is grateful to Doyoun Kim and Seodong Shin for their useful comments on the manuscript. This work was supported by IBS under the project code, IBS-R018-D1.

**References**

[1] W. S. Cho, J. S. Gainer, D. Kim, K. T. Matchev, F. Moortgat, L. Pape, and M. Park, “On-shell constrained $M_2$ variables with applications to mass measurements and topology disambiguation,” *JHEP* **08** (2014) 070, [arXiv:1401.1449 [hep-ph]].

[2] W. S. Cho, J. S. Gainer, D. Kim, S. H. Lim, K. T. Matchev, F. Moortgat, L. Pape, and M. Park, “OPTIMASS: A Package for the Minimization of Kinematic Mass Functions with Constraints,” *JHEP* **01** (2016) 026, [arXiv:1508.00589 [hep-ph]].

[3] D. Kim, K. T. Matchev, F. Moortgat, and L. Pape, “Testing Invisible Momentum Ansätze in Missing Energy Events at the LHC,” *JHEP* **08** (2017) 102, [arXiv:1703.06887 [hep-ph]].

[4] C. Lester and D. Summers, “Measuring masses of semi invisibly decaying particles pair produced at hadron colliders,” *Phys. Lett. B* **463** (1999) 99–103, [arXiv:hep-ph/9906349].

[5] A. Barr, C. Lester, M. A. Parker, B. Allanach, and P. Richardson, “Discovering anomaly mediated supersymmetry at the LHC,” *JHEP* **03** (2003) 045, [arXiv:hep-ph/0208214].

[6] A. Barr, C. Lester, and P. Stephens, “$m_T^2$: The Truth behind the glamour,” *J. Phys. G* **29** (2003) 2343–2363, [arXiv:hep-ph/0304226].
[7] W. S. Cho, K. Choi, Y. G. Kim, and C. B. Park, “Gluino Stransverse Mass,” *Phys. Rev. Lett.* **100** (2008) 171801, arXiv:0709.0288 [hep-ph].

[8] A. J. Barr, B. Gripaios, and C. G. Lester, “Weighing Wimps with Kinks at Colliders: Invisible Particle Mass Measurements from Endpoints,” *JHEP* **02** (2008) 014, arXiv:0711.4008 [hep-ph].

[9] W. S. Cho, K. Choi, Y. G. Kim, and C. B. Park, “Measuring superparticle masses at hadron collider using the transverse mass kink,” *JHEP* **02** (2008) 035, arXiv:0711.4526 [hep-ph].

[10] M. M. Nojiri, K. Sakurai, Y. Shimizu, and M. Takeuchi, “Handling jets + missing $E_T$ channel using inclusive $m_T^2$,” *JHEP* **10** (2008) 100, arXiv:0808.1094 [hep-ph].

[11] W. S. Cho, K. Choi, Y. G. Kim, and C. B. Park, “$M_T^2$-assisted on-shell reconstruction of missing momenta and its application to spin measurement at the LHC,” *Phys. Rev. D* **79** (2009) 031701, arXiv:0810.4853 [hep-ph].

[12] K. Choi, S. Choi, J. S. Lee, and C. B. Park, “Reconstructing the Higgs boson in dileptonic $W$ decays at hadron collider,” *Phys. Rev. D* **80** (2009) 073010, arXiv:0908.0079 [hep-ph].

[13] W. S. Cho, K. Choi, Y. G. Kim, and C. B. Park, “Mass and Spin Measurement with $M_T^2$ and MAOS Momentum,” *Nucl. Phys. B Proc. Suppl.* **200-202** (2010) 103–112, arXiv:0909.4853 [hep-ph].

[14] K. Choi, J. S. Lee, and C. B. Park, “Measuring the Higgs boson mass with transverse mass variables,” *Phys. Rev. D* **82** (2010) 113017, arXiv:1008.2690 [hep-ph].

[15] C. B. Park, “Reconstructing the heavy resonance at hadron colliders,” *Phys. Rev. D* **84** (2011) 096001, arXiv:1106.6087 [hep-ph].

[16] C. Lester and A. Barr, “$m_{T\text{Gen}}$: Mass scale measurements in pair-production at colliders,” *JHEP* **12** (2007) 102, arXiv:0708.1028 [hep-ph].

[17] M. Burns, K. Kong, K. T. Matchev, and M. Park, “Using Subsystem $M_T^2$ for Complete Mass Determinations in Decay Chains with Missing Energy at Hadron Colliders,” *JHEP* **03** (2009) 143, arXiv:0810.5576 [hep-ph].

[18] P. Konar, K. Kong, K. T. Matchev, and M. Park, “Dark Matter Particle Spectroscopy at the LHC: Generalizing $M_T^2$ to Asymmetric Event Topologies,” *JHEP* **04** (2010) 086, arXiv:0911.4126 [hep-ph].

[19] K. Agashe, D. Kim, D. G. Walker, and L. Zhu, “Using $M_T^2$ to Distinguish Dark Matter Stabilization Symmetries,” *Phys. Rev. D* **84** (2011) 055020, arXiv:1012.4460 [hep-ph].

[20] C. G. Lester, “The transverse mass, $M_T^2$, in special cases,” *JHEP* **05** (2011) 076, arXiv:1103.5682 [hep-ph].

[21] C. H. Lally and C. G. Lester, “Properties of $M_T^2$ in the massless limit,” arXiv:1211.1542 [hep-ph].
[22] R. Mahbubani, K. T. Matchev, and M. Park, “Re-interpreting the Oxbridge transverse mass variable $M_{T2}$ in general cases,” *JHEP* **03** (2013) 134, arXiv:1212.1720 [hep-ph].

[23] F. James and M. Roos, “Minuit: A System for Function Minimization and Analysis of the Parameter Errors and Correlations,” *Comput. Phys. Commun.* **10** (1975) 343–367.

[24] C. G. Lester, “MT2 / Stransverse Mass / Oxbridge Kinetics Library,” https://www.hep.phy.cam.ac.uk/~lester/mt2/.

[25] R. Fletcher and M. J. D. Powell, “A Rapidly Convergent Descent Method for Minimization,” *Comput. J.* **6** no. 2, (1963) 163–168.

[26] W. C. Davidon, “VARIABLE METRIC METHOD FOR MINIMIZATION,” ANL-5990-Rev. 2, ANL-5990.

[27] J. Nelder and R. Mead, “A Simplex Method for Function Minimization,” *Comput. J.* **7** (1965) 308–313.

[28] H.-C. Cheng and Z. Han, “Minimal Kinematic Constraints and $m_{T2}$,” *JHEP* **12** (2008) 063, arXiv:0810.5178 [hep-ph].

[29] C. G. Lester and B. Nachman, “Bisection-based asymmetric $M_{T2}$ computation: a higher precision calculator than existing symmetric methods,” *JHEP* **03** (2015) 100, arXiv:1411.4312 [hep-ph].

[30] A. Barr, T. Khoo, P. Konar, K. Kong, C. Lester, K. Matchev, and M. Park, “Guide to transverse projections and mass-constraining variables,” *Phys. Rev. D* **84** (2011) 095031, arXiv:1105.2977 [hep-ph].

[31] G. G. Ross and M. Serna, “Mass determination of new states at hadron colliders,” *Phys. Lett. B* **665** (2008) 212–218, arXiv:0712.0943 [hep-ph].

[32] A. J. Barr, G. G. Ross, and M. Serna, “The Precision Determination of Invisible-Particle Masses at the LHC,” *Phys. Rev. D* **78** (2008) 056006, arXiv:0806.3224 [hep-ph].

[33] D. Debnath, D. Kim, J. H. Kim, K. Kong, and K. T. Matchev, “Resolving Combinatorial Ambiguities in Dilepton $t\bar{t}$ Event Topologies with Constrained $M_2$ Variables,” *Phys. Rev. D* **96** no. 7, (2017) 076005, arXiv:1706.04995 [hep-ph].

[34] M. R. Hestenes, “Multiplier and gradient methods,” *J. Optimiz. Theory. App.* **4** (1969) 303–320.

[35] M. J. D. Powell, “A method for nonlinear constraints in minimization problems,” in *Optimization*, R. Fletcher, ed., pp. 283–298. Academic Press, New York, 1969.

[36] J. Nocedal and S. Wright, *Numerical Optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2006.

[37] A. R. Conn, N. I. M. Gould, and P. Toint, “A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds,” *SIAM J. Numer. Anal.* **28** no. 2, (1991) 545–572.

[38] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, 2000.
[39] R. B. Wilson, *A simplicial algorithm for concave programming*. PhD thesis, Graduate School of Business Administration, Harvard University, 1963.

[40] U. M. G. Palomares and O. L. Mangasarian, “Superlinearly convergent quasi-newton algorithms for nonlinearly constrained optimization problems,” *Math. Program.* 11 (1976) 1–13.

[41] S. P. Han, “Superlinearly convergent variable metric algorithms for general nonlinear programming problems,” *Math. Program.* 11 (1976) 263–282.

[42] S. P. Han, “A globally convergent method for nonlinear programming,” *J. Optimiz. Theory. App.* 22 (1977) 297–309.

[43] M. J. D. Powell, “Algorithms for nonlinear constraints that use lagrangian functions,” *Math. Program.* 14 (1978) 224–248.

[44] A. Forsgren, P. E. Gill, and M. H. Wright, “Interior methods for nonlinear optimization,” *SIAM Rev.* 44 (2002) 525–597.

[45] N. Gould, D. Orban, and P. Toint, “Numerical methods for large-scale nonlinear optimization,” *Acta Numer.* 14 (2005) 299–361.

[46] S. G. Johnson, “The NLopt nonlinear-optimization package,” [https://github.com/stevengj/nlopt](https://github.com/stevengj/nlopt).

[47] D. Kraft, *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988.

[48] D. Kraft, “Algorithm 733: TOMPÅȘ-Fortran modules for optimal control calculations,” *Acm. T. Math. Software* 20 no. 3, (1994) 262–281.

[49] E. G. Birgin and J. M. MartÅłnez, “Improving ultimate convergence of an Augmented Lagrangian method,” *Optim. Method. Softw.* 23 no. 2, (2011) 177–195.

[50] C. B. Park, “YAM2: Yet another library for the $M_2$ variables,” [https://github.com/cbpark/YAM2](https://github.com/cbpark/YAM2), 2020.

[51] P. Konar, K. Kong, and K. T. Matchev, “$\sqrt{s_{\text{min}}}$: A Global inclusive variable for determining the mass scale of new physics in events with missing energy at hadron colliders,” *JHEP* 03 (2009) 085, arXiv:0812.1042 [hep-ph].

[52] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, “An Introduction to PYTHIA 8.2,” *Comput. Phys. Commun.* 191 (2015) 159–177, arXiv:1410.3012 [hep-ph].

[53] G. Apollinari, O. Brüning, T. Nakamoto, and L. Rossi, “High Luminosity Large Hadron Collider HL-LHC,” *CERN Yellow Rep.* no. 5, (2015) 1–19, arXiv:1705.08830 [physics.acc-ph].

[54] S. G. Johnson, “NLopt Installation,” [https://nlopt.readthedocs.io/en/latest/NLopt_Installation/](https://nlopt.readthedocs.io/en/latest/NLopt_Installation/).
[55] C. B. Park, “A singular way to search for heavy resonances in missing energy events,” *JHEP* **07** (2020) 089, arXiv:2005.12297 [hep-ph].

[56] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Math. Program.* **106** (2006) 25–57.

[57] P. Konar and A. K. Swain, “Mass reconstruction with $M_2$ under constraint in semi-invisible production at a hadron collider,” *Phys. Rev. D* **93** no. 1, (2016) 015021, arXiv:1509.00298 [hep-ph].

[58] P. Konar and A. K. Swain, “Reconstructing semi-invisible events in resonant tau pair production from Higgs,” *Phys. Lett. B* **757** (2016) 211–215, arXiv:1602.00552 [hep-ph].