Eur. Phys. J. C (2020) 80:1106
https://doi.org/10.1140/epjc/s10052-020-08635-y

THE EUROPEAN PHYSICAL JOURNAL C

Special Article - Tools for Experiment and Theory

Xsec: the cross-section evaluation code

Andy Buckley[1], **Anders Kvellestad**[2,3], **Are Raklev**[3], **Pat Scott**[2,4], **Jon Vegard Sparre**[5], **Jeriek Van den Abeele**[3,a], **Ingrid A. Vazquez-Holm**[6]

[1] School of Physics and Astronomy, University of Glasgow, Glasgow G12 8QQ, UK
[2] Department of Physics, Imperial College London, South Kensington, London SW7 2AZ, UK
[3] Department of Physics, University of Oslo, 0316 Oslo, Norway
[4] School of Mathematics and Physics, The University of Queensland, St. Lucia, Brisbane, QLD 4072, Australia
[5] The Norwegian Labour and Welfare Administration, 0557 Oslo, Norway
[6] Institut de Physique Théorique, Université Paris-Saclay, CEA, CNRS, 91191 Gif-sur-Yvette, France

Received: 9 July 2020 / Accepted: 2 November 2020 / Published online: 2 December 2020

Abstract The evaluation of higher-order cross-sections is an important component in the search for new physics, both at hadron colliders and elsewhere. For most new physics processes of interest, total cross-sections are known at next-to-leading order (NLO) in the strong coupling α_s, and often beyond, via either higher-order terms at fixed powers of α_s, or multi-emission resummation. However, the computation time for such higher-order cross-sections is prohibitively expensive, and precludes efficient evaluation in parameter-space scans beyond two dimensions. Here we describe the software tool xsec, which allows for fast evaluation of cross-sections based on the use of machine-learning regression, using distributed Gaussian processes trained on a pre-generated sample of parameter points. This first version of the code provides all NLO Minimal Supersymmetric Standard Model strong-production cross-sections at the LHC, for individual flavour final states, evaluated in a fraction of a second. Moreover, it calculates regression errors, as well as estimates of errors from higher-order contributions, from uncertainties in the parton distribution functions, and from the value of α_s. While we focus on a specific phenomenological model of supersymmetry, the method readily generalises to any process where it is possible to generate a sufficient training sample.

Contents

1 Introduction

The determination of cross-sections beyond leading order (LO) is typically very computationally expensive because of the evaluation of tensorial loop integrals. This is especially so for hadronic interactions, where the loop integrals must themselves be numerically integrated over the relevant parton distribution functions (PDFs).

[a] e-mail: jeriekvda@fys.uio.no (corresponding author)

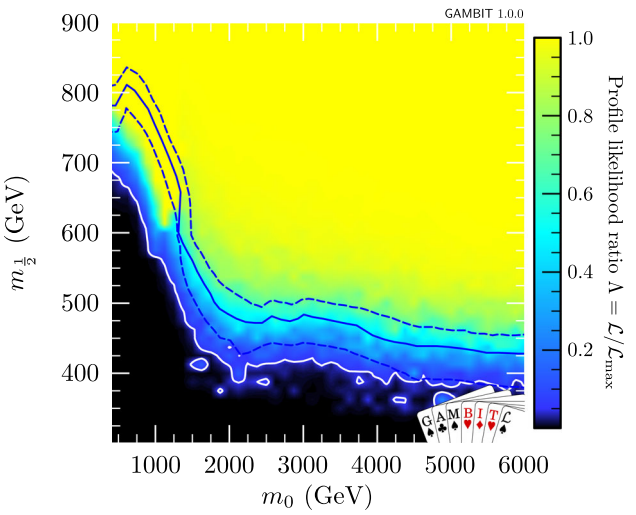


Fig. 1 Colour map showing the profile likelihood ratio $\mathcal{L}/\mathcal{L}_{\max}$ for a ColliderBit scan over the CMSSM model defined by m_0 and $m_{1/2}$, with $\tan\beta = 30$, $A_0 = -2m_0$ and $\mu > 0$, using the ATLAS zero-lepton supersymmetry search likelihood from $\sqrt{s} = 8$ TeV data [2]. The solid white line indicates the 95% CL exclusion contour found with GAMBIT 1.0 [3] using LO cross-sections. The solid blue line shows the corresponding ATLAS 95% CL observed exclusion limit with NLO+NLL cross-sections, with dashed blue lines showing the $\pm 1\sigma$ theoretical cross-section uncertainty. Reproduced from Fig. 3 in Ref. [1]

The computational cost of evaluation per parameter point restricts the usage of next-to-leading order (NLO) cross-sections to (simplified) new physics models with only one or two relevant parameters. However, higher-order contributions can be very important in many other models. This is especially true for strong interactions, where NLO contributions can be of comparable size to the LO contribution. The physics impact of this restriction is quite dramatic. In Fig. 1, reproduced from Ref. [1], we show as an example the significant differences between the limits resulting from a parameter scan of the Constrained Minimal Supersymmetric Standard Model (CMSSM) using LO rather than NLO cross-sections. This also highlights the importance of propagating theory uncertainties, e.g. from the PDFs and the scale dependence, through to the physical observables. As the uncertainties on LO hard-scattering cross-sections are typically very large, owing in part to the missing higher-order terms, such theory error propagation to the cross-sections can only really be considered representative from NLO accuracy onwards. Even at NLO, we can see in Fig. 1 that the impact is considerable, and needs to be taken into account when fitting models.

In this work we present xsec, an attempt at a general solution to the speed problem irrespective of the type of cross-section being evaluated, the size of the parameter space of the model, and the underlying precision of the calculation. We achieve this by performing machine-learning regression on a pre-generated training dataset consisting of cross-sections sampled from a model. In this first version of xsec, we will be focusing on strong-production cross-sections in the MSSM, but the selection of cross-sections and models will be extended in future versions.

The increasing sparsity of training data in models with larger numbers of free parameters means that a lot of care must go into performing the regression, and that reliable regression errors can only be determined on a point-by-point basis. We choose to use Gaussian process (GP) regression [4], a highly flexible Bayesian method for modelling functions. It is not restricted to pre-determined functional shapes with a set number of parameters, and is instead directly informed by the data together with some prior understanding of the underlying correlation structure of the function that it is used to model. Moreover, a point-specific regression uncertainty follows naturally from the posterior predictive distribution, which is straightforward to compute in closed form.

Training GP models involves the inversion of a matrix of the size of the number of training points. This means that while GPs are a very powerful regression tool, single GPs scale very badly with increasing training data, and training becomes infeasible beyond $\mathcal{O}(10^4)$ data points. To overcome this difficulty we use a model with factorised and distributed training, where the data are split into manageable subsets each assigned to a single GP. The regression prediction is subsequently determined by a combination of the predictions of the individual GPs [5,6].

Current codes that can evaluate a broad collection of MSSM cross-sections at NLO include Prospino 2.1 [7–12] and MadGraph 5 [13]. However, the evaluation time per parameter point on a modern CPU is in the range of minutes for a single final state, far too slow for large parameter scans. This does not include the calculation of scale, α_s, nor PDF errors, which in Prospino increases the evaluation time by orders of magnitude. In the MadGraph implementation, this can be done more efficiently through reweighting techniques. However, the NLO functionality for the MSSM has only recently been made publicly available in MadGraph, and after the computationally expensive sample generation campaign for the current version of xsec was completed.

A test of the calculation time per parameter point for all strong-production cross-sections in Prospino, including the evaluation of scale, α_s, and PDF errors, when Prospino has been heavily optimised with the ifort compiler, takes around 2.5 hours on a single Intel Xeon-Gold 6138 (Skylake) core running at 2.0 GHz.

Fast evaluation of strong-production cross-sections in the MSSM at NLO, with next-to-leading (NLL) and next-to-next-to-leading (NNLL) logarithmic resummation, already exists in the form of the NLL-fast and NNLL-fast codes [14–24], which add NLL and NNLL corrections to existing NLO results from Prospino. However, these results are restricted

to interpolation in a two-dimensional subspace spanned by the gluino mass and a common squark mass, and results are given as a sum over outgoing squark flavours.

In addition to the processes found in Prospino, cross-sections for gaugino and slepton pair production, as well as gaugino-gluino production, can be evaluated at NLO+NLL precision by the Resummino code [25–33]. Here, the total running time for all chargino and neutralino production processes at NLO (again on a single 2.0 GHz Intel Xeon-Gold 6138 Skylake core) is around three hours, with no evaluation of scale, α_s, nor PDF errors. Combined NLO+NLL precision takes four days.

Recently, the DeepXS package [34] appeared, which performs a fast evaluation of neutralino and chargino pair production using neural networks. This is currently limited to a small set of the most phenomenologically relevant processes in the MSSM-19, and does not include uncertainties arising from PDFs, or the value of α_s.

The xsec 1.0 code is currently designed to reproduce all the NLO results of Prospino 2.1 for strong production in the MSSM in a small fraction of the time, including scale errors, and with the addition of PDF and α_s errors based on a modern PDF set. It does not include NNLL or even NLL resummation as can be found in NNLL-fast and NLL-fast, however, unlike those codes it provides reliable results for non-degenerate squark masses, within the inherent limitations of the training set generated with Prospino. We intend to extend the code to also include NLL resummation in future releases.

Unlike NNLL-fast, xsec performs a separate evaluation of the cross-section for all distinct flavour combinations of the first two generations of squarks. The xsec code also treats sbottom and stop pair-production processes separately, but this has limited impact as the two cross-sections are the same at LO (for the same masses),[1] as Prospino – and therefore our training set – is limited to light quark initial states. A complete list of available processes can be found in Table 1.

The code is written in Python with heavy use of the NumPy [35] package for numerical calculations, and is compatible with both Python 2 and 3. It can be installed using the pip package manager and run as a set of functions from a self-contained module. We also provide interfaces through SLHA files [36] and a command-line tool.

The rest of this paper is structured as follows. We begin in Sect. 2 with an introduction to the GP regression framework that we employ. In Sect. 3 we describe our GP training regime, including how we compute the required training sets of NLO cross-sections. In Sect. 4, we perform a thorough validation of the results from xsec, with a comparison to results from existing codes. Section 5 covers the structure of the code and its interfaces, and we conclude in Sect. 6.

2 Regression framework

2.1 Gaussian process regression

The basic objective in regression is to estimate an unknown function value $f(\mathbf{x})$ at some new $\mathbf{x}$ point, given that we know the function values at some other $\mathbf{x}$ points. A Bayesian approach to this task is provided by Gaussian process regression, in which we express our degree of belief about any set of function values as a joint Gaussian pdf. We underline here that this pdf should be understood in a purely Bayesian sense – it does not imply any randomness in the true function we are approximating.

We begin by defining some notation and terminology, indicating in italics the terminology commonly used in the GP and machine learning literature. Each input point $\mathbf{x}$ has m components (*features*), which in our case will correspond to masses and mixing angles from the MSSM squark and gluino sector. We denote by $\mathbf{x}_*$ the new input point (*test point*) for which we will estimate the unknown, true function value $f_* \equiv f(\mathbf{x}_*)$, here an NLO production cross-section. We let $\mathbf{x}_i$ with $i = 1, \ldots, n$ denote the n input points (*training points*) at which we know the function values $f_i \equiv f(\mathbf{x}_i)$ (*targets*). The combined set $\mathcal{D} = \{\mathbf{x}_i, f_i\}_{i=1}^n$ is referred to as the *training set*. The complete set of input components in our training set can be expressed as an $n \times m$ matrix $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^{\mathrm{T}}$. Similarly, the complete set of known function values can be collected in a vector $\mathbf{f} = [f_1, \ldots, f_n]^{\mathrm{T}}$. Thus, our training set can also be expressed as $\mathcal{D} = \{X, \mathbf{f}\}$.

The starting point for GP regression is the formulation of a joint Gaussian prior pdf[2]

$$p(\mathcal{D}, f_* | \mathbf{x}_*) \equiv p(\mathbf{f}, f_* | X, \mathbf{x}_*), \tag{1}$$

which formally describes our degree of belief for possible function values at both the training points X and the test point $\mathbf{x}_*$, before we look at the training data. This prior is chosen indirectly by choosing a *mean function* $m(\cdot)$ and a *covariance function* or *kernel* $k(\cdot, \cdot)$, defined to specify the following expectation values for arbitrary input points:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \equiv \int_{-\infty}^{\infty} f p(f|\mathbf{x})\, \mathrm{d}f \tag{2}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]\,. \tag{3}$$

[1] Considering only QCD corrections, corrections that depend on the mixing angles contribute only through $t\tilde{t}_i\tilde{g}$ and $\tilde{t}_1\tilde{t}_1\tilde{t}_2\tilde{t}_2$ vertices (and similarly for sbottoms) at NLO and are typically small.

[2] We always treat the input $\mathbf{x}$ points in a dataset $\mathcal{D}$ as known. Thus, the pdf $p(\mathcal{D})$ should be understood as the pdf $p(f_1, \ldots, f_n | \mathbf{x}_1, \ldots, \mathbf{x}_n) = p(\mathbf{f}|X)$, and similarly for other pdfs involving $\mathcal{D}$.

We note that while the mean function and kernel are defined as functions of inputs in $\mathbf{x}$ space, the function values represent mean and covariance values in f space. Our joint Gaussian prior can then be expressed as

$$p\left(\begin{bmatrix}\mathbf{f}\\ f_*\end{bmatrix}\middle|\begin{bmatrix}X\\ \mathbf{x}_*\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}m(X)\\ m(\mathbf{x}_*)\end{bmatrix}, \begin{bmatrix}\Sigma & k(X,\mathbf{x}_*)\\ k(\mathbf{x}_*,X) & k(\mathbf{x}_*,\mathbf{x}_*)\end{bmatrix}\right), \tag{4}$$

where

$$m(X) \equiv [m(\mathbf{x}_i),\ldots,m(\mathbf{x}_n)]^{\mathrm{T}} \tag{5}$$
$$k(\mathbf{x}_*,X) \equiv [k(\mathbf{x}_*,\mathbf{x}_i),\ldots,k(\mathbf{x}_*,\mathbf{x}_n)] \tag{6}$$
$$k(X,\mathbf{x}_*) \equiv k(\mathbf{x}_*,X)^{\mathrm{T}} \tag{7}$$
$$\Sigma \equiv k(X,X),\ \text{i.e. } \Sigma_{ij} = k(\mathbf{x}_i,\mathbf{x}_j). \tag{8}$$

The choice and optimisation of the kernel and mean function constitute the main challenge in GP regression, and we will discuss these aspects in detail in the next sections.

Our goal is to obtain a predictive posterior pdf for the unknown function value f_* at $\mathbf{x}_*$. From the fully specified GP prior we can now find this simply by "looking at" the training data $\mathbf{f}$, i.e. by deriving from the GP prior $p(\mathcal{D}, f_*|\mathbf{x}_*)$ the conditional pdf

$$p(f_*|\mathcal{D},\mathbf{x}_*) = \mathcal{N}(\mu_*,\sigma_*^2). \tag{9}$$

The mean and variance of this univariate Gaussian can be expressed in closed form as

$$\mu_* = m(\mathbf{x}_*) + k(\mathbf{x}_*,X)\Sigma^{-1}(\mathbf{f}-m(X)) \tag{10}$$
$$\sigma_*^2 = k(\mathbf{x}_*,\mathbf{x}_*) - k(\mathbf{x}_*,X)\Sigma^{-1}k(X,\mathbf{x}_*). \tag{11}$$

The prediction μ_* for f_* is thus simply the prior mean $m(\mathbf{x}_*)$ plus a shift given by a weighted sum of the shifts of the known function values from their corresponding prior means, $\mathbf{f}-m(X)$. The weights are proportional to the covariances between the prediction at $\mathbf{x}_*$ and the known function values at the training points X, as set by the kernel $k(\mathbf{x}_*,X)$. The prediction variance σ_*^2 is given as the prior variance $k(\mathbf{x}_*,\mathbf{x}_*)$ reduced by a term representing the additional information provided by the training data about the function value at $\mathbf{x}_*$. This naturally depends only on the kernel. We will refer to the width σ_* simply as the regression error or GP prediction error, keeping in mind that it should be interpreted in a Bayesian manner.

2.2 Kernel choice and optimisation

Choosing the kernel, Eq. (3), is the main modelling step in GP regression. It effectively determines what types of functional structure the GP will be able to capture. In particular, it encodes the smoothness and the periodicity (if applicable) of the function that is being modelled, as it controls the expected correlation between function values at two different points. The choice of prior mean function, Eq. (2), is typically much less important, as we discuss at the end of this section.

The question of the optimal kernel choice is covered in more detail in Refs. [4,37]. The squared-exponential kernel

$$k\left(\mathbf{x},\mathbf{x}';\sigma_f^2,\ell\right) = \sigma_f^2\exp\left(-(\mathbf{x}-\mathbf{x}')^2/(2\ell^2)\right), \tag{12}$$

is the standard choice. It results in an exponentially decreasing correlation as the Euclidean distance between two input points increases with respect to a length-scale hyperparameter ℓ. The signal variance σ_f^2 is a hyperparameter containing information about the amplitude of the modelled function. This is a universal kernel [38], which means that it is in principle capable of approximating any continuous function given enough data. The infinite differentiability and exponential behaviour of this kernel typically result in a very smooth posterior mean.

However, for our purposes, the squared-exponential has some problems. Its sensitivity to changes in the function means that the length scale ℓ is usually determined by the smallest 'wiggle' in the function [37]. We hence consider also the Matérn kernel family: like the squared-exponential, these are universal and stationary, i.e., only functions of the relative positions of the two input points, but additionally incorporate a smoothness hyperparameter ν following the basic form

$$k_{\mathrm{M}}(\mathbf{x},\mathbf{x}';\nu,\ell) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\frac{|\mathbf{x}-\mathbf{x}'|}{\ell}\right)^{\nu}K_\nu\left(\sqrt{2\nu}\frac{|\mathbf{x}-\mathbf{x}'|}{\ell}\right), \tag{13}$$

where $\Gamma(\nu)$ is the gamma function and K_ν is a modified Bessel function of the second kind. For the modelling of cross-section functions, we adopt the Matérn kernel class on the basis of its superior performance. This has followed significant testing and cross-validation across a number of different problems [39–41]. During testing we found $\nu = \frac{3}{2}$ to be optimal for our purposes, in which case Eq. (13) simplifies to

$$k_{\mathrm{M}}(\mathbf{x},\mathbf{x}';\nu=\tfrac{3}{2},\ell) = \left(1+\sqrt{3}\frac{|\mathbf{x}-\mathbf{x}'|}{\ell}\right)\exp\left(-\sqrt{3}\frac{|\mathbf{x}-\mathbf{x}'|}{\ell}\right). \tag{14}$$

To account for the fact that some directions in the input space of masses and mixing angles may have more impact on the cross-section values than others, we use an anisotropic, multiplicative Matérn kernel,

$$k_{\mathrm{AMM}}(\mathbf{x}, \mathbf{x}'; \nu, \sigma_f^2, \boldsymbol{\ell}) = \sigma_f^2 \prod_{d=1}^{m} k_M(x^{(d)}, x'^{(d)}; \nu, \ell_d), \quad (15)$$

where we have also included a signal variance hyperparameter σ_f^2, similar to the one in Eq. (12). Here $x^{(d)}$ denotes the dth component of the input vector $\mathbf{x}$, and $\boldsymbol{\ell}$, with components ℓ_d, is a vector containing one length scale per $\mathbf{x}$ component. The product over the dimensions of the parameter space results in points only being strongly correlated if in each dimension, their distance is small with respect to the relevant length scale.

So far we have focused on the "noise-free" case, in which the training targets $\mathbf{f}$ are the exact values of the true function at the training points. In this case the predictive posterior $p(f_*|\mathcal{D}, \mathbf{x}_*)$ collapses to a delta function when $\mathbf{x}_*$ equals a training point. In theory this is a reasonable approach, since what we seek is a surrogate model for an expensive, but precise and deterministic numerical computation. In practice, however, allowing for some uncertainty also at the training points typically results in a more well-behaved and stable regression model. The main reason for this is that the additional wiggle-room in the modelling can ease the challenging matrix numerics of GP regression, as we will discuss in some detail in Sect. 2.3.

We therefore add a "white-noise" term,

$$k_{\mathrm{WN}}(\mathbf{x}, \mathbf{x}'; \sigma_\epsilon^2) = \delta_{\mathbf{x}\mathbf{x}'} \sigma_\epsilon^2, \quad (16)$$

to our kernel, where σ_ϵ^2 is the hyperparameter that sets the amount of "noise". The effect of this term is simply to add σ_ϵ^2 along the diagonal of the covariance matrix Σ, as well as to the prior variance at the test point, $k(\mathbf{x}_*, \mathbf{x}_*)$. It is known as *homoscedastic* noise, as it is the same for all data points.

In GP terminology, to include this additional variance term corresponds to going from the noise-free case to a scenario with noisy training data. The targets are then considered measurements $y_i \equiv y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$, with the noise ϵ_i, introduced in the process of performing the ith measurement, modelled by a Gaussian distribution $\mathcal{N}(0, \sigma_\epsilon^2)$. However, we remind the reader that for our case this Gaussian pdf represents an adopted effective Bayesian degree of belief in the accuracy of the training data, rather than an expression of actual random noise.

Conceptually we should then make the substitution $f \to y$ in our definitions from Sect. 2.1. Our training set becomes $\mathcal{D} = \{X, \mathbf{y}\}$, with $\mathbf{y} = [y_1, \ldots, y_n]^{\mathrm{T}}$, and the GP prior becomes a joint pdf for $\mathbf{y}$ and y_*:

$$p(\mathcal{D}, y_*|\mathbf{x}_*) = p(\mathbf{y}, y_*|X, \mathbf{x}_*). \quad (17)$$

The prior mean function and kernel now specify expectation values in y space,

$$m(\mathbf{x}) = \mathbb{E}[y(\mathbf{x})] \quad (18)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(y(\mathbf{x}) - m(\mathbf{x}))(y(\mathbf{x}') - m(\mathbf{x}'))], \quad (19)$$

where we note that $\mathbb{E}[y(\mathbf{x})] = \mathbb{E}[f(\mathbf{x})]$ since the Gaussian noise term has zero mean. Likewise, the predictive posterior pdf becomes

$$p(y_*|\mathcal{D}, \mathbf{x}_*) = \mathcal{N}(\mu_*, \sigma_*^2), \quad (20)$$

with mean and variance

$$\mu_* = m(\mathbf{x}_*) + k(\mathbf{x}_*, X)\Sigma^{-1}(\mathbf{y} - m(X)) \quad (21)$$

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, X)\Sigma^{-1}k(X, \mathbf{x}_*). \quad (22)$$

Our complete kernel is then given by

$$k(\mathbf{x}, \mathbf{x}') = k_{\mathrm{AMM}}(\mathbf{x}, \mathbf{x}'; \nu, \sigma_f^2, \boldsymbol{\ell}) + k_{\mathrm{WN}}(\mathbf{x}, \mathbf{x}'; \sigma_\epsilon^2). \quad (23)$$

Fixing $\nu = \frac{3}{2}$, as discussed above, we are left with the set $\boldsymbol{\theta} = \{\sigma_f^2, \boldsymbol{\ell}, \sigma_\epsilon^2\}$ of undetermined hyperparameters. To be fully Bayesian, one would introduce a prior pdf $p(\boldsymbol{\theta})$ for the hyperparameters and obtain the GP posterior $p(y_*|\mathcal{D}, x_*)$ by marginalising over $\boldsymbol{\theta}$,

$$\begin{aligned} p(y_*|\mathcal{D}, x_*) &= \int p(y_*, \boldsymbol{\theta}|\mathcal{D}, x_*)\, \mathrm{d}\boldsymbol{\theta} \\ &= \int p(y_*|\boldsymbol{\theta}, \mathcal{D}, x_*)\, p(\boldsymbol{\theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta} \\ &\propto \int p(y_*|\boldsymbol{\theta}, \mathcal{D}, x_*)\, p(\mathcal{D}|\boldsymbol{\theta})\, p(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}. \end{aligned} \quad (24)$$

In our high-dimensional case with large datasets, such integration would come at a steep computational expense, even with MCMC methods. We therefore follow the common approach of using a point estimate for the hyperparameters, found by maximising the log-likelihood function [4]

$$\begin{aligned} \log p(\mathcal{D}|\boldsymbol{\theta}) &= \log p(\mathbf{y}|X, \boldsymbol{\theta}) \\ &= -\frac{1}{2}(\mathbf{y} - m(X))^{\mathrm{T}}\Sigma^{-1}(\mathbf{y} - m(X)) \\ &\quad -\frac{1}{2}\log|\Sigma| - \frac{n}{2}\log 2\pi. \end{aligned} \quad (25)$$

Finding an adequate set of hyperparameters constitutes the model training step in the GP approach. It is complicated by the fact that each optimisation step requires the computation of the inverse and determinant of the $n \times n$ covariance matrix Σ, which scales poorly with the number of training points n. To increase speed and numerical stability, Σ is generally

not directly inverted in practice, and its Cholesky decomposition is used instead. In an attempt to avoid local optima, we employ the SciPy implementation of the differential evolution method [42,43], rather than performing a gradient-based search.

Recent work has demonstrated that the theoretical prediction error σ_*^2 in Eq. (22) systematically underestimates the mean-squared prediction error when the hyperparameters are learned from the data [44]. As proposed there, we account for the uncertainty on the point estimate of the hyperparameter by adding a correction term to σ_*^2, derived from the Hybrid Cramér–Rao Bound. In our case, with a constant prior mean function, this extra term amounts to

$$\Delta\sigma_*^2 = \left(1 - \mathbf{1}\Sigma^{-1}k(X, \mathbf{x}_*)\right)^2 \Big/ \sum_{i,j=1}^{n} [\Sigma^{-1}]_{ij}, \tag{26}$$

where $\mathbf{1} \equiv [1, \ldots, 1]$. In particular, this increases the prediction error at test points far from the training data.

Compared to the choice of kernel, the choice of the prior mean function, Eq. (18), is typically less important. Following conditioning on a sufficiently large training set, the prior gets overpowered and the posterior mean is primarily influenced by the training data through the second term in Eq. (21). For this reason the prior mean function is commonly taken to be zero everywhere. Nevertheless, it is sensible to incorporate our knowledge of the mean, and we therefore use the sample mean of the target values $\mathbf{y}$ as a prior mean function that is constant in $\mathbf{x}$.

2.3 Regularisation of the covariance matrix

A practical challenge when training GPs is to ensure numerical stability when inverting the covariance matrix Σ. The precision of the result is controlled by the condition number κ of Σ, which can be considered a measure of the sensitivity of the inversion to roundoff error. It is computed as the ratio $\lambda_{\max}/\lambda_{\min}$ between the highest and lowest eigenvalues of Σ, and becomes infinite for a singular matrix. The loss of numerical precision at high κ becomes most obvious when the predictive variance, computed according to Eq. (22), evaluates to a negative number. In order to prevent this problem, it is essential to understand how to control κ.

When the target values of training points are strongly correlated, their corresponding rows and columns in Σ are nearly identical. This leads to eigenvalues close to zero and a very large condition number. It has been shown that in the worst case, κ can grow linearly with the number of training points and quadratically with the signal-to-noise ratio $\mathrm{SNR} = \sigma_f/\sigma_\epsilon$ [45].

Increasing the noise level improves numerical stability, as a larger diagonal contribution σ_ϵ^2 to Σ enhances the difference between otherwise similar rows and columns. Therefore, we add a term to the log likelihood in Eq. (25) that penalises hyperparameter choices with extremely high signal-to-noise ratios, as suggested in Ref. [45]. Our objective function for training GPs then becomes

$$\log p(\mathbf{y}|X, \boldsymbol{\theta}) - \left(\frac{\log(\sigma_f/\sigma_\epsilon)}{\log(\mathrm{SNR}_{\max})}\right)^{50}. \tag{27}$$

The large exponent guarantees that situations where $\mathrm{SNR} > \mathrm{SNR}_{\max}$ are the only ones where the penalty term has a significant effect. We use $\mathrm{SNR}_{\max} = 10^4$.

In some cases the likelihood penalty in Eq. (27) does not decrease the condition number sufficiently to stabilise the inversion. However, choosing a lower overall value for $\mathrm{SNR}_{\max}$ dilutes the information in the training data to an extent that it can sometimes be fitted by noise, even when unnecessary. We therefore check the condition number after the optimisation with the penalty term, and proceed to increase the homoscedastic noise σ_ϵ^2 just for the inversion step until the condition number drops below a reasonable value $\kappa_{\max}$ [46]:

$$\Delta\sigma_\epsilon^2 = \frac{\lambda_{\max} - \kappa_{\max}\lambda_{\min}}{\kappa_{\max} - 1}. \tag{28}$$

We set $\kappa_{\max} = 10^9$, roughly corresponding to a maximal loss of nine digits accuracy from the total of 16 in a 64-bit double-precision floating-point number.

These measures may seem to deteriorate the performance of our regression model, but they are necessary to ensure the numerical stability. The underlying reason is that we have essentially noiseless data, and are hitting the limits of floating-point precision in the process of calculating the GP predictions. In comparison to the scale and PDF uncertainties on the cross-sections, the resulting regression errors nevertheless remain small, as we demonstrate in Sect. 4.

2.4 Distributed Gaussian processes and prediction aggregation

With n training points, the complexity of the matrix inversion operations in Eqs. (21) and (22) scales as n^3, making standard GP regression unsuitable for problems that require large training sets. To overcome this challenge we construct a regression model based on *distributed Gaussian processes* (DGPs) [5]: we partition the total training set $\mathcal{D}$ into d manageable subsets $\mathcal{D}_i$, and for each $\mathcal{D}_i$ we train a new GP $\mathcal{M}_i$. These GPs are referred to as *experts*. The prediction from our regression model is obtained by aggregating the predictions from the individual experts. For this prediction aggregation we follow the approach know as the *Generalised Robust*

Bayesian Committee Machine (GRBCM) [6], for which we summarise the main steps below.

First we construct a data subset $\mathcal{D}_1 \equiv \mathcal{D}_c$, randomly chosen from $\mathcal{D}$ without replacement, which will be used to train a single *communication expert* $\mathcal{M}_c$. Next, we partition the remaining data into subsets $\{\mathcal{D}_i\}_{i=2}^d$, each of which will serve to train one expert $\mathcal{M}_i$. Following Refs. [5,6], all experts are then trained simultaneously, such that they share a common set of hyperparameters.

The GRBCM approach places no restrictions on how to partition the data to form the subsets $\{\mathcal{D}_i\}_{i=2}^d$. However, empirical studies have shown that some clustering of the data can help the experts to become sensitive to local, short-scale variability of the target function [6,47]. Compared to using a simple random partition, we have noticed minor improvements with a disjoint partition, where the data is split into local subsets based on the mass parameter with the smallest length-scale hyperparameter. Tests with k-means clustering did not indicate further improvements in our case, nor did tests with sorting on less dominant features.

The special role of the communication expert $\mathcal{M}_c$ becomes evident at the prediction stage. For each of the experts $\{\mathcal{M}_i\}_{i=2}^d$, we construct an improved expert $\mathcal{M}_{+i}$ by replacing the corresponding dataset $\mathcal{D}_i$ with the extended set $\mathcal{D}_{+i} = \{\mathcal{D}_i, \mathcal{D}_c\}$. That is, for prediction the communication dataset $\mathcal{D}_c$ is shared by all the experts $\mathcal{M}_{+i}$. The communication expert $\mathcal{M}_c$ serves as a common baseline to which the experts $\mathcal{M}_{+i}$ can be compared. In the final combination, the prediction from expert $\mathcal{M}_{+i}$ will be weighted according to the differential entropy difference between its predictive distribution and that of $\mathcal{M}_c$.

The central approximation that allows for computational gains in DGPs and related approaches is an assumption that the individual experts can be treated as independent, which corresponds to approximating the kernel matrix of the combined problem, i.e. without partition into experts, as block-diagonal. In the GRBCM approach, this approximation is expressed as the conditional independence assumption $\mathcal{D}_i \perp \mathcal{D}_j | \mathcal{D}_c, y_*, \mathbf{x}_*$ for $2 \leq i \neq j \leq d$, which enables the approximation $p(\mathcal{D}_i|\mathcal{D}_j, \mathcal{D}_c, y_*, \mathbf{x}_*) \approx p(\mathcal{D}_i|\mathcal{D}_c, y_*, \mathbf{x}_*)$. That is, when the information contained in the communication set $\mathcal{D}_c$ is known, we assume that the predictive distribution for points in subset $\mathcal{D}_i$ should not be strongly influenced by the additional information contained in subset $\mathcal{D}_j$.

Using Bayes' theorem and the above independence assumption, the exact predictive distribution $p(y_*|\mathcal{D}, \mathbf{x}_*)$ can now be approximated as

$$
\begin{aligned}
& p(y_*|\mathcal{D}, \mathbf{x}_*) \\
& \propto\ p(y_*|\mathbf{x}_*)\, p(\mathcal{D}_c|y_*, \mathbf{x}_*) \prod_{i=2}^{d} p(\mathcal{D}_i|\mathcal{D}_1, \ldots \mathcal{D}_{i-1}, y_*, \mathbf{x}_*) \\
& \approx\ p(y_*|\mathbf{x}_*)\, p(\mathcal{D}_c|y_*, \mathbf{x}_*) \prod_{i=2}^{d} p^{\beta_i}(\mathcal{D}_i|\mathcal{D}_c, y_*, \mathbf{x}_*) \\
& =\ p(y_*|\mathbf{x}_*)\, p(\mathcal{D}_c|y_*, \mathbf{x}_*) \prod_{i=2}^{d} \frac{p^{\beta_i}(\mathcal{D}_i, \mathcal{D}_c|y_*, \mathbf{x}_*)}{p^{\beta_i}(\mathcal{D}_c|y_*, \mathbf{x}_*)} \\
& =\ \frac{p(y_*|\mathbf{x}_*) \prod_{i=2}^{d} p^{\beta_i}(\mathcal{D}_{+i}|y_*, \mathbf{x}_*)}{p^{\beta_1}(\mathcal{D}_c|y_*, \mathbf{x}_*)},
\end{aligned}
\tag{29}
$$

where we have introduced the weights β_i for the predictions from different experts, and defined $\beta_1 \equiv -1 + \sum_{i=2}^d \beta_i$. By applying Bayes' theorem again, we can express our approximation for $p(y_*|\mathcal{D}, \mathbf{x}_*)$ in terms of the corresponding predictive distributions from the individual experts, $p_{+i}(y_*|\mathcal{D}_{+i}, \mathbf{x}_*)$ and $p_c(y_*|\mathcal{D}_c, \mathbf{x}_*)$. Leaving out normalisation factors, the distribution for the aggregated prediction becomes

$$
p_A(y_*|\mathcal{D}, \mathbf{x}_*) \propto \frac{\prod_{i=2}^{d} p_{+i}^{\beta_i}(y_*|\mathcal{D}_{+i}, \mathbf{x}_*)}{p_c^{\beta_1}(y_*|\mathcal{D}_c, \mathbf{x}_*)},
\tag{30}
$$

with mean μ_{DGP} and variance σ^2_{DGP} at $\mathbf{x}_*$ given by

$$
\sigma^{-2}_{\mathrm{DGP}}(\mathbf{x}_*) = -\beta_1 \sigma_c^{-2}(\mathbf{x}_*) + \sum_{i=2}^{d} \beta_i \sigma_{+i}^{-2}(\mathbf{x}_*)
\tag{31}
$$

$$
\begin{aligned}
\frac{\mu_{\mathrm{DGP}}(\mathbf{x}_*)}{\sigma^2_{\mathrm{DGP}}(\mathbf{x}_*)} &= -\beta_1 \sigma_c^{-2}(\mathbf{x}_*) \mu_c(\mathbf{x}_*) \\
&\quad + \sum_{i=2}^{d} \beta_i \sigma_{+i}^{-2}(\mathbf{x}_*) \mu_{+i}(\mathbf{x}_*).
\end{aligned}
\tag{32}
$$

Following Ref. [6], we set the weights β_i to

$$
\beta_2 = 1,
\tag{33}
$$

$$
\beta_{i \geq 3} = 0.5 \left[\log \sigma_c^2(\mathbf{x}_*) - \log \sigma_{+i}^2(\mathbf{x}_*) \right].
\tag{34}
$$

The reason for assigning weight $\beta_2 = 1$ for expert $\mathcal{M}_{+2}$ is that the transition

$$
p(\mathcal{D}_i|\mathcal{D}_1, \ldots \mathcal{D}_{i-1}, y_*, \mathbf{x}_*) \to p^{\beta_i}(\mathcal{D}_i|\mathcal{D}_c, y_*, \mathbf{x}_*)
\tag{35}
$$

in Eq. (29) is exact for $i = 2$, $\beta_2 = 1$. For each remaining expert $\mathcal{M}_{+i \geq 3}$, the weight is taken to be the difference in differential entropy between the baseline predictive distribution of the communication expert, $p_c(y_*|\mathcal{D}_c, \mathbf{x}_*)$, and that of the given expert, $p_{+i}(y_*|\mathcal{D}_{+i}, \mathbf{x}_*)$. Thus, if an expert $\mathcal{M}_{+i}$ provides little additional predictive power over $\mathcal{M}_c$, its relative influence on the aggregated prediction is low.

Requiring the experts to share a common set of hyperparameters effectively disfavours overfitting of individual experts. Moreover, the risk of overfitting is alleviated by the fact that after training, each expert is extended with the communication dataset $\mathcal{D}_c$ that it did not see during training, and its weight in the prediction aggregation is regularised through the comparison to the communication expert.

The GRBCM split of the dataset into d experts reduces the complexity of training from n^3 to $\mathcal{O}(d(n/d)^3 = n^3 d^{-2})$. The memory, storage space, and evaluation all depend directly on the size of the matrix, and scale as $\mathcal{O}(n^2)$ for a regular GP, but as $\mathcal{O}(n^2/d)$ in the GRBCM approach.

3 Training

3.1 Sample generation

We generate the inputs for our training data calculations by sampling the physical gluino and squark masses and (for third-generation squarks) the angles describing mass mixing between gauge eigenstates. The only other parameters involved in the production of gluinos and squarks to NLO QCD are the strong coupling α_s and the SM quark masses. The cross-sections depend on α_s both through the matrix element and the PDF. To capture the cross-section variation due to the uncertainty on α_s, we generate separate input points with α_s set to 0.1180 (central value), 0.1165 (1σ lower value) and 0.1195 (1σ upper value), using the corresponding PDF sets, and train separate GPs on the ratio of cross-sections obtained with the central and $\pm 1\sigma$ values. For the SM masses we use a fixed value for the bottom and top quark masses, and assume the other quark masses to be zero.

In the sample generation we do not simply sample over a regular grid of parameter values, for three reasons:

1. Grid sampling is inefficient when one parameter is more important than the others: too many evaluations are spent on varying the less influential parameters whilst keeping the important one at a fixed value.
2. The curse of dimensionality renders this sampling technique infeasible for processes that depend on more than three or four parameters.
3. The complexity of sampling and cross-section calculation for the large number of processes that we consider (in terms of final-state squark flavours) means that it is more efficient to evaluate multiple cross-sections for every parameter combination than to generate separate samples for each final state.

We sample individual baseline masses for the gluino, $m_{\tilde{g}}$, for the first and second-generation (gauge eigenstate) squarks, $m_{\tilde{u}_L}$, $m_{\tilde{d}_L}$, $m_{\tilde{c}_L}$, $m_{\tilde{s}_L}$, $m_{\tilde{u}_R}$, $m_{\tilde{d}_R}$, $m_{\tilde{c}_R}$, and $m_{\tilde{s}_R}$, and for the third-generation (mass eigenstate) squarks $m_{\tilde{b}_1}$, $m_{\tilde{t}_1}$, $m_{\tilde{b}_2}$, and $m_{\tilde{t}_2}$. We do this in two different ways: either drawing from a uniform distribution on the interval [50, 3500] GeV, or from a hybrid distribution uniform on the interval [50, 150] GeV and logarithmic on the interval [150, 3500] GeV. We order the third-generation squarks in mass after sampling, so that $\tilde{t}_1$ and $\tilde{b}_1$ are by definition the lightest. We intentionally choose these sampling ranges to be slightly beyond our final claimed region of validity for the regression – typically 200–3000 GeV – in order to try to avoid large regression errors at the edges. The regions where our regression has been validated more than cover the ranges of masses of interest for the LHC. We sample the cosines of the sbottom and stop mixing angles, $\cos\theta_{\tilde{b}}$ and $\cos\theta_{\tilde{t}}$, uniformly on the interval $[-1, 1]$.

On top of our baseline sampling, we employ further subsampling of the particle masses in order to properly include cross-section resonances within our training set. This ensures that our training dataset is more densely sampled where the cross-sections of interest vary the most. To do this, we generate and then combine six different training sets:

(i) All masses sampled from the uniform prior.
(ii) All masses sampled from the hybrid prior.
(iii) Employing the uniform prior for both the mass of the gluino and for a common squark mass scale, and taking a Gaussian prior with width 50 GeV for the difference between the common squark mass scale and the masses of the individual squarks.
(iv) Employing the uniform prior for the mass of the gluino, the hybrid prior for a common squark mass scale, and the same Gaussian prior as in (iii) to draw values for the squark masses around the common scale.
(v) Employing the uniform prior for a common mass scale, and using the same Gaussian prior as in (iii) to draw a gluino mass and the individual squark masses around the common scale.
(vi) Varying the gluino mass and a common squark mass independently on a two-dimensional grid spanning from 60 to 3000 GeV, with steps of 60 GeV.[3]

By joining these samples, we are able to achieve both good sampling of low masses (where cross-sections are large) via the logarithmic mass prior, and acceptable sampling of large masses via the uniform prior.

Because R-parity is assumed in the MSSM, there are no new s-channel resonances in the mass range where we train our GPs, and SM resonances are too light to have any impact. Therefore, we do not have to worry directly about sampling densely in the region of resonances. However, at LO there

[3] This set was only used to improve the regression results for the stop and sbottom pair production cross-sections.

are potential effects for gluino pair production near threshold from destructive interference between diagrams with s-channel gluons and those with t-channel squarks. For squark production, the chirality of the final-state squarks affects the contribution from t-channel gluino exchange, so that for example in squark–squark production for equal chiralities the matrix element has a zero at $m_{\tilde{g}} \to 0$, and a maximum around $m_{\tilde{g}} \simeq \bar{m}_{\tilde{q}}$, where $\bar{m}_{\tilde{q}}$ is the average mass of the first and second-generation squarks. Both these effects can lead to non-monotonous behaviour of the cross-section as a function of masses (bumps and dips). In Sect. 4 we shall see this particularly for gluino pair production. This necessitates the separate samples with (near) degenerate masses, where such effects are larger.

Even though all non-degenerate squark masses enter into the LO cross-sections, to the precision of our training sample (see Sect. 3.2), only two mass parameters enter into the NLO corrections to gluino and first/second-generation squark production: the gluino mass $m_{\tilde{g}}$ and the averaged first and second-generation squark mass $\bar{m}_{\tilde{q}}$. Therefore, any additional interference structures present in the NLO corrections to these processes must be visible in the $(m_{\tilde{g}}, \bar{m}_{\tilde{q}})$ slice of the parameter space. For stop/sbottom production, such structures should be similarly visible in the $(m_{\tilde{g}}, m_{\tilde{t}_1/\tilde{b}_1})$ slice. As a result, we shall spend some time below in validation (Sect. 4) looking at gluino and first/second-generation squark production cross-sections in terms of $m_{\tilde{g}}$ and $\bar{m}_{\tilde{q}}$, and third-generation squark production in terms of $m_{\tilde{g}}$ and $m_{\tilde{t}_1}$.

Table 1 gives the list of cross-sections available from xsec, and their parameter dependencies. The reader may wonder at this point why we train our GPs on the total cross-section in terms of all the non-degenerate masses, rather than simply on the NLO corrections (in terms of just $m_{\tilde{g}}$ and $\bar{m}_{\tilde{q}}$ for gluino/first/second-generation squark production, supplemented with the three relevant third-generation parameters each for stop and sbottom production). One reason is that we have designed xsec to be more general than Prospino; in future releases we intend to make use of training data from other tools able to move beyond the degenerate-squark-mass approximation. The other reason is that we feel it is more convenient for a user to simply obtain full LO+NLO (and future +NLL+NNLL+…) cross-sections from xsec, rather than needing to install the correct LO cross-section calculator and PDF set, call them, and then combine the results. Similarly, training on and returning the full cross-section will make simultaneously including cross-sections from different calculators (chosen e.g. as most appropriate for different theories or processes) far more straightforward.

Table 1 List of all available cross-sections in xsec. Explicit first and second-generation squarks $\tilde{q}_i$ can be $\tilde{u}_L, \tilde{d}_L, \tilde{c}_L, \tilde{s}_L, \tilde{u}_R, \tilde{d}_R, \tilde{c}_R$ or $\tilde{s}_R$, whilst $\bar{m}_{\tilde{q}}$ denotes the average over all eight of these masses. Where the charge-conjugate process is distinct from the original, xsec returns the sum of the cross-section for the process and its conjugate (as indicated; for $q_i\tilde{q}_j^*$, the conjugate is only distinct when $i \neq j$)

Final state	Variables
$\tilde{g}\tilde{g}$	$m_{\tilde{g}}, \bar{m}_{\tilde{q}}, m_{\tilde{u}_L}, m_{\tilde{d}_L}, m_{\tilde{c}_L}, m_{\tilde{s}_L}, m_{\tilde{u}_R}, m_{\tilde{d}_R}, m_{\tilde{c}_R}, m_{\tilde{s}_R}$
$\tilde{g}\tilde{q}_i$ + c.c.	$m_{\tilde{g}}, \bar{m}_{\tilde{q}}, m_{\tilde{q}_i}$
$\tilde{q}_i\tilde{q}_j$ + c.c.	$m_{\tilde{g}}, \bar{m}_{\tilde{q}}, m_{\tilde{q}_i}, m_{\tilde{q}_j}$
$\tilde{q}_i\tilde{q}_j^*$(+ c.c.)	$m_{\tilde{g}}, \bar{m}_{\tilde{q}}, m_{\tilde{q}_i}, m_{\tilde{q}_j}$
$\tilde{b}_i\tilde{b}_i^*$	$m_{\tilde{g}}, \bar{m}_{\tilde{q}}, m_{\tilde{b}_1}, m_{\tilde{b}_2}, \cos\theta_{\tilde{b}}$
$\tilde{t}_i\tilde{t}_i^*$	$m_{\tilde{g}}, \bar{m}_{\tilde{q}}, m_{\tilde{t}_1}, m_{\tilde{t}_2}, \cos\theta_{\tilde{t}}$

3.2 Calculation of NLO training cross-sections

We use Prospino 2.1 to generate cross-sections for our training samples. This calculates, amongst other things, NLO cross-sections for strong production processes in the MSSM for proton-proton collisions at a given centre-of-mass (CoM) energy, and for a choice of renormalisation/factorisation scales. We have modified the code to set α_s accordingly, and to accept generic PDF sets from LHAPDF 6.2 [48]. For the current version of xsec we have used the PDF4LHC15_nlo_30_pdfas symmetric Hessian NLO PDF set with 30 eigenvector members and two members with varied strong coupling $\alpha_s(m_Z) = 0.1180 \pm 0.0015$ [49].

Prospino performs the PDF integral of the partonic process using the VEGAS [50] importance sampling algorithm for Monte Carlo integration. The convergence criterion leaves some numerical noise in the result, typically of the order of 10^{-3} relative to the central cross-section value.

In order to obtain K-factors for gluino and first/second-generation squark production, Prospino first calculates the LO and NLO cross-sections using a single squark mass, obtained as the average over the masses of all first and second-generation squarks, i.e. eight in total; this mass is employed even for any internal third-generation squark propagators. The ratio of the LO and NLO results gives the K-factor for the process in question. Prospino then recomputes the cross-section at LO, without the assumption of an average mass, and the corresponding NLO value is found by multiplying this LO result by the K-factor calculated for the average squark mass. The calculation proceeds similarly for stop and sbottom production, except that the third-generation squark masses are kept non-degenerate for all steps of the calculation, meaning that first and second-generation masses are still averaged in the K-factor calculation. The final cross-sections should thus be viewed as an approximation to a fully non-degenerate squark mass NLO calculation. The effect of this assumption was investigated in Ref. [51] and found to be relatively small in most parts of parameter space.

In total, Prospino allows 141 final states containing gluinos and different flavour squarks. However, due to charge conjugation relations and NLO QCD identities in the cross-section (relating certain combinations of left and right-handed final-state squarks under the exchange of masses), only 49 distinct final states are needed to represent all processes calculated by Prospino in our training sample. These relationships are handled internally in xsec, and therefore need not directly concern the user. A list of the available cross-sections from the user's side can be found in Table 1.

For every parameter point and every process in the training sample, we calculate 34 different cross-section values. These include: (i) a central value, (ii) 30 values using the PDF eigenvectors, (iii) one value where α_s is taken to its lower value, (iv) one where α_s is taken to its upper value, (v) a value where the renormalisation/factorisation scale is doubled, and, (vi) a value where the renormalisation/factorisation scale is halved.

From the values in (i) and (ii) we follow the PDF4LHC guidelines to calculate the symmetric PDF uncertainty on the central value, understood to be a 68% confidence level bound.[4] Similarly, we follow the PDF4LHC prescription to calculate the 68% confidence level bound from varying α_s alone, using the results from (iii) and (iv). We do not add the PDF and α_s errors, but train different DGPs for the two errors. We note that these errors should be added in quadrature after evaluation if the user wishes to obtain a single 68% confidence bound incorporating both effects.

Further, we follow the standard lore of estimating an uncertainty associated with missing higher-order corrections by calculating the spread in cross-section values under scale variation coming from (v) and (vi). In xsec this is referred to as the scale uncertainty. Choosing exactly how to interpret this asymmetric uncertainty as a probability distribution (flat, Gaussian or otherwise), and whether/how to combine it with the (Gaussian) α_s and PDF uncertainties, is left to the user.

In total this leaves us with seven cross-sections: the central value and the two-sided PDF, α_s and scale uncertainties.

3.3 Training implementation details

For each final state, we train one large DGP with multiple experts on the central cross-section value, and five smaller regular GPs: one each on the upper and lower values arising from regularisation and factorisation scale variation, one each on the cross-sections for the variation of α_s (and corresponding PDFs) to its upper and lower limit in its 68% confidence interval (these values are later symmetrised), and a single GP on the symmetric 68% confidence level PDF variation.

To improve the training, we try to simplify the target model. The span of some ten orders of magnitude in cross-sections across the sampled parameter space means that it is numerically challenging to train the DGPs on the raw cross-section numbers. Before training, we scale out part of the dependence on the final-state masses by first multiplying the central input cross-section by the square of the (average) final-state mass, and then take the logarithm of the result. For the remaining five 'error' values, we train on the logarithm of the ratio to the central cross-section, such that after the reverse transformation, the final predictions for the ratios are strictly positive. As noted in Sect. 2.2, we choose a constant prior mean equal to the sample average of the transformed target values, which results in GPs effectively modelling the deviation from this mean value. We also rescale all input parameters to the interval [0, 1] before training, to improve numerical stability and precision.

The transformations during training are automatically recorded and reversed at the time of evaluation. As the GPs are trained on the logarithm of the (mass-rescaled) cross-section, the resulting predictive distribution for the absolute cross-section is a log-normal distribution. Due to the positive skew of this distribution, we base the central cross-section estimate on the median. Specifically, let $\mathcal{N}(\mu_{\mathrm{DGP}}(\mathbf{x}_*), \sigma^2_{\mathrm{DGP}}(\mathbf{x}_*))$ denote the aggregated DGP predictive distribution for the log-transformed cross-section at point $\mathbf{x}_*$, after accounting for the constant prior shift and the mass scaling applied to the training data. The central cross-section value y_{xsec} returned by xsec is then

$$y_{\mathsf{xsec}}(\mathbf{x}_*) = \exp[\mu_{\mathrm{DGP}}(\mathbf{x}_*)], \tag{36}$$

with associated asymmetric regression uncertainties Δ^-_{xsec} and Δ^+_{xsec}. These are constructed from the bounds of the 1σ credible interval $\exp(\mu_{\mathrm{DGP}} \pm \sigma_{\mathrm{DGP}})$[5]:

$$\begin{aligned} \Delta^-_{\mathsf{xsec}}(\mathbf{x}_*) &= \exp[\mu_{\mathrm{DGP}}(\mathbf{x}_*)] \\ &\quad - \exp[\mu_{\mathrm{DGP}}(\mathbf{x}_*) - \sigma_{\mathrm{DGP}}(\mathbf{x}_*)], \\ \Delta^+_{\mathsf{xsec}}(\mathbf{x}_*) &= \exp[\mu_{\mathrm{DGP}}(\mathbf{x}_*) + \sigma_{\mathrm{DGP}}(\mathbf{x}_*)] \\ &\quad - \exp[\mu_{\mathrm{DGP}}(\mathbf{x}_*)]. \end{aligned} \tag{37}$$

More generally, the range $y_{\mathsf{xsec}} \pm m\Delta^{\pm}_{\mathsf{xsec}}$ corresponds to an $n\sigma$ credible interval, where n is given by

$$n = \frac{\log\left(1 \pm \frac{m\Delta^{\pm}_{\mathsf{xsec}}}{y_{\mathsf{xsec}}}\right)}{\log\left(1 \pm \frac{\Delta^{\pm}_{\mathsf{xsec}}}{y_{\mathsf{xsec}}}\right)}, \tag{38}$$

[4] Due to the extreme calculational cost, we do not use the recommended Monte Carlo PDF sets with 100 replicas, but use the symmetric Hessian set with 30 eigenvector members.

[5] As detailed in Sect. 5, xsec returns these regression errors in the form of the signed, relative errors $(-\Delta^-_{\mathsf{xsec}}/y_{\mathsf{xsec}})$ and $(\Delta^+_{\mathsf{xsec}}/y_{\mathsf{xsec}})$.

so that $n = m$ to first order in $(m\Delta^{\pm}_{\text{xsec}}/y_{\text{xsec}})$ and $(\Delta^{\pm}_{\text{xsec}}/y_{\text{xsec}})$.

We train the GPs with the union of the training sets discussed in Sect. 3.1. In order to choose the optimal training parameters (relative sizes of the five different training samples, total number of training points, and experts per process), we have carried out a long programme of testing and cross-validation, in which we optimised training parameters separately for each process type, e.g. separately for gluino pair production and gluino–squark production. Our goal was to achieve good performance in terms of the regression error estimated by the DGP, while keeping disk size, memory footprint, training and evaluation time down to acceptable levels.

The processor time spent on training the GPs varies for each process and depends critically on the number of training points per expert because of the GP scaling relations, as well as on optimiser settings like the population size and convergence thresholds for the differential evolution. In practice, the total time we spent on training a single process ranges from 4.5 CPU hours for a simple case with only three parameters, such as $\tilde{c}^*_L\tilde{c}_L$ production, to 141 CPU hours for gluino pair production, which has ten parameters at NLO. In these most extreme examples, the use of parallel processing for the matrix operations, in particular for the different experts, reduced the actually elapsed wall-clock time by a factor 8–12, compared to the CPU time.

4 Validation

To validate the xsec cross-section results we generate three main test sets: $\mathcal{D}_{\text{test}}$, $\mathcal{D}_{\text{test-tb}}$ and $\mathcal{D}_{\text{MSSM-24}}$. The set $\mathcal{D}_{\text{test}}$ is used for testing all cross-sections except for the stop/sbottom pair-production processes, which are tested with the set $\mathcal{D}_{\text{test-tb}}$. The sets $\mathcal{D}_{\text{test}}$ and $\mathcal{D}_{\text{test-tb}}$ contain 10,000 and 5000 points, respectively, and in both cases the input points are sampled in the same way as for the corresponding training sets. The third set, $\mathcal{D}_{\text{MSSM-24}}$, contains 19,000 points. We use this in the validation of all cross-sections. Here we draw the input points from the MSSM-24, as defined at $Q = 1\,\text{TeV}$, using uniform priors for the MSSM parameters. Our definition of the MSSM-24 follows that in Ref. [3], except that we parameterise the Higgs sector using the higgsino mass parameter μ and the tree-level mass of the A^0 boson m_{A^0}, instead of the soft-breaking mass parameters $m^2_{H_u}$ and $m^2_{H_d}$. For the samples in $\mathcal{D}_{\text{MSSM-24}}$ we use SoftSUSY 4.0 to calculate the physical mass spectrum from the MSSM input parameters [52,53]. In addition to the three main test sets, we also generate a number of process-specific two-dimensional parameter grids for further validation.

As part of the validation we will discuss two simple error measures and their distributions for our sets of test points. The first measure is just the relative error,

$$\epsilon(\mathbf{x}_*) = \frac{y_{\text{prosp}}(\mathbf{x}_*) - y_{\text{xsec}}(\mathbf{x}_*)}{y_{\text{prosp}}(\mathbf{x}_*)}, \tag{39}$$

which measures the relative deviation of the xsec result y_{xsec} from the true Prospino value y_{prosp}. By assuming some sampling prior $\pi(\mathbf{x}_*)$ for the test points $\mathbf{x}_*$ and looking at the resulting distribution of $\epsilon(\mathbf{x}_*)$ values, we can get a *global* picture of how well xsec performs for the different supersymmetric production processes included in xsec.

One of the strengths of GP regression is that the method directly provides *point-wise* uncertainty estimates, here in terms of $\Delta^-_{\text{xsec}}(\mathbf{x}_*)$ and $\Delta^+_{\text{xsec}}(\mathbf{x}_*)$, which are based on the width $\sigma_{\text{DGP}}(\mathbf{x}_*)$ of the DGP predictive distribution (see Eq. 37). As discussed in Sect. 2, the point-wise regression uncertainty is connected to a Bayesian degree of belief regarding the unknown function value at $\mathbf{x}_*$, given the particular training set $\mathcal{D}$ and the modelling choices made in constructing and optimising the kernel.

An interesting question is then how this point-wise uncertainty compares to the *actual* deviation between y_{xsec} and the true Prospino value y_{prosp} across the input feature space. That is, we should also investigate the distribution of the standardised residual

$$z(\mathbf{x}_*) = \frac{y_{\text{prosp}}(\mathbf{x}_*) - y_{\text{xsec}}(\mathbf{x}_*)}{\Delta^{\pm}_{\text{xsec}}(\mathbf{x}_*)}, \tag{40}$$

in our sets of test points. Here the notation $\Delta^{\pm}_{\text{xsec}}$ is shorthand for using Δ^+_{xsec} when $y_{\text{prosp}} - y_{\text{xsec}} > 0$ and Δ^-_{xsec} when $y_{\text{prosp}} - y_{\text{xsec}} < 0$. By studying the distribution of $z(\mathbf{x}_*)$ values for our test sets, and comparing to the unit normal distribution, we will obtain a global picture of the extent to which the point-wise regression errors are conservative or not, compared to the true deviations.

In cases where the main source of GP regression uncertainty is actual random noise in the training data, and we learn this noise level from the data by including a white-noise term, Eq. (16), in the GP kernel, we expect the residual in Eq. (40) to be distributed as $\mathcal{N}(0, 1)$.[6] We can understand this from a simple example with single-component input points x. Assume that the target data are noisy measurements of the form $y(x) = f(x) + \delta$, where the noise δ is distributed as $p(\delta) = \mathcal{N}(0, \sigma^2_{\text{noise}})$. Let the GP posterior predictive distribution for $y(x_*)$ be given by $\mathcal{N}(\mu_{\text{GP}}(x_*), \sigma^2_{\text{GP}}(x_*))$. If the noise is the dominant uncertainty we have $\sigma_{\text{GP}}(x_*) \approx \sigma_{\text{noise}}$, and we can express the residual as

$$\begin{aligned} z(x_*) &= \frac{y(x_*) - \mu_{\text{GP}}(x_*)}{\sigma_{\text{GP}}(x_*)} \\ &\approx \frac{y(x_*) - f(x_*)}{\sigma_{\text{noise}}} + \frac{f(x_*) - \mu_{\text{GP}}(x_*)}{\sigma_{\text{GP}}(x_*)}. \end{aligned} \tag{41}$$

[6] That is, at least up to log-normal corrections implied by Eq. (38) for $n \neq 1$.

Given the generative model for the data, the first term will follow an $\mathcal{N}(0, 1)$ distribution. The second term is the number of standard deviations (as measured by the GP's own uncertainty) by which the GP prediction μ_{GP} differs from the true value of the underlying function f. While this term can in general not be expected to be normally distributed, the assumption of noise-dominated uncertainty implies that its contribution to the residual is $\ll 1$, and hence, that the z distribution is close to $\mathcal{N}(0, 1)$.

On the other hand, if the true noise level is tiny and some other source of uncertainty dominates σ_{GP}, i.e., if $y \approx f$ and $\sigma_{\text{GP}} \gg \sigma_{\text{noise}}$, we get

$$z(x_*) \approx \frac{f(x_*) - \mu_{\text{GP}}(x_*)}{\sigma_{\text{GP}}(x_*)}. \tag{42}$$

Thus, in this limit it is the generally small, and potentially non-Gaussian, second term from Eq. (41) that dominates the residual.

The latter scenario is most similar to the case we have for the xsec residual in Eq. (40). The actual noise level in the Prospino training data is very small, as typically is the additional error in Eq. (26) accounting for uncertainty in the hyperparameter choice. Assuming some reasonably uninformative sampling prior $\pi(\mathbf{x}_*)$, this means that for most points the widths $\Delta^{\pm}_{\text{xsec}}(\mathbf{x}_*)$ are dominated by the homoscedastic error contribution that we include to stabilise the numerics (see Sect. 2.3). As this is a global error contribution, we can expect the resulting regression error for most test points to be larger than the actual error, $y_{\text{prosp}}(\mathbf{x}_*) - y_{\text{xsec}}(\mathbf{x}_*)$. We therefore in general expect the $z(\mathbf{x}_*)$ distributions to be narrow compared to $\mathcal{N}(0, 1)$, and not necessarily Gaussian. For comparison, we will include a graph of $\mathcal{N}(0, 1)$ in all our plots of residual distributions.

In the coming subsections, much of our focus will be on the regression errors and the related relative error ϵ and residual z. However, we remind the reader that the regression errors that we find are typically far subdominant to the cross-section uncertainties coming from the scale and PDF errors.

4.1 Gluino pair production

The gluino pair-production cross-section to NLO in QCD depends on the gluino mass and all the other squark masses. Naturally, the gluino mass is the dominant feature of the DGP after training. The mass-averaging approximation that Prospino uses (see Sect. 3.2) means that the average first/second-generation squark mass $\bar{m}_{\tilde{q}}$ is a strong predictor of the NLO contribution. We therefore provide it to the GPs as a separate feature, i.e. as if it were part of the vector of parameters $\mathbf{x}$. The importance of the individual first and second-generation squark masses roughly follows the PDF

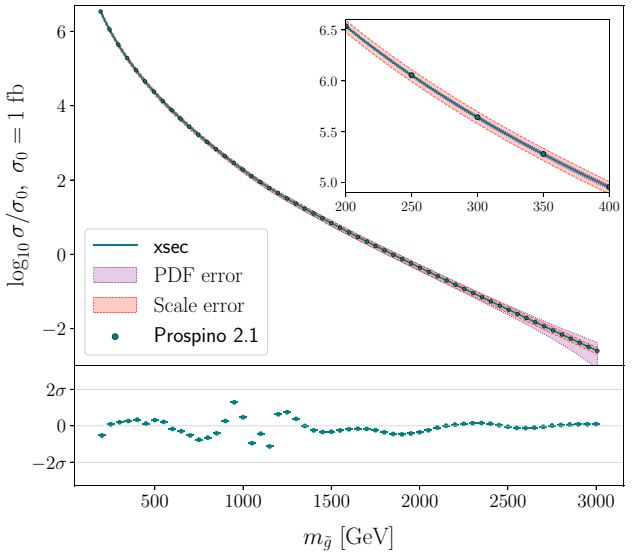


Fig. 2 Gluino pair-production cross-section as a function of gluino mass, with all squark masses fixed at 1 TeV. The central value is shown in light green, the scale error in pink, and the PDF error in violet. The α_s error is too small to be visible on the scale of the plot. Superimposed on the prediction are the Prospino values (dots). Inset is a close-up of the region at low gluino mass, and below we show the residual between the xsec prediction and the Prospino values

contributions from their corresponding quarks, due to LO t-channel squark exchange diagrams.

Given that the mass range of parameters (features) in the training samples is $[100, 3500]$ GeV, we validate the cross-section on the sub-interval $[200, 3000]$ GeV for both gluino and squark masses, where the cross-section regression has solid support from training data and the regression error is small.

In Fig. 2 we compare the gluino pair-production cross-sections predicted by xsec, presented as a function of the gluino mass, with values taken directly from Prospino (but not in the training set). For this comparison we fix the squark masses to a common value of 1 TeV. We also show the associated xsec-predicted uncertainty from the renormalisation scale and the PDFs as bands. The uncertainties from the regression provided by xsec and from α_s are too small to be visible on the logarithmic scale. However, below the plot we show the residual between the xsec prediction and the Prospino value (Eq. 40), as a multiple of the xsec regression uncertainty. We observe good agreement overall with the Prospino result. As expected, the scale error dominates at low gluino masses, and the PDF error at high masses.

A particular phenomenon occurs in gluino pair production due to destructive interference between LO diagrams when $m_{\tilde{g}} \approx m_{\tilde{q}}$, resulting in a vanishing partonic cross-section at threshold [7]. This can be found as a significant dip in the total pair-production cross-section when one or more of the squark masses become degenerate with the gluino. We show

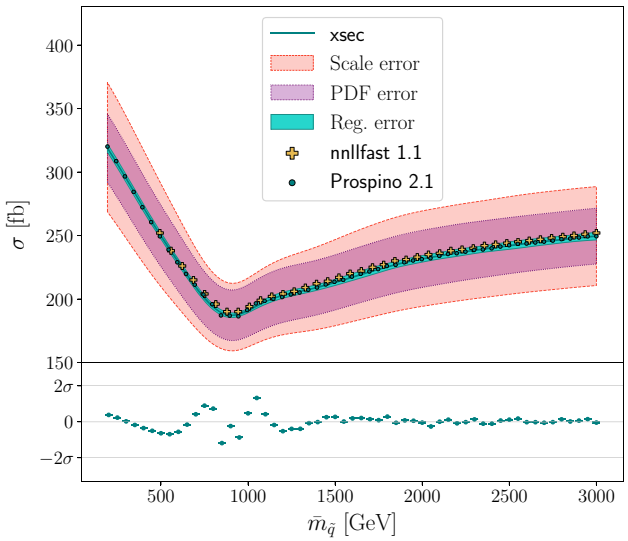


Fig. 3 Gluino pair-production cross-section as a function of average first and second-generation squark mass. The gluino mass is fixed at 1 TeV. Shown are the central xsec prediction (solid line), the 1σ regression error band (light green), the scale error (pink), the PDF error (violet), the Prospino values (dots) and the corresponding NNLL-fast NLO result (crosses)

that xsec reproduces this behaviour to very good precision in Fig. 3. Here the gluino mass is fixed to 1 TeV while the squark masses are run together as a common squark mass $\bar{m}_{\tilde{q}}$. This figure also clearly demonstrates how subdominant the regression error is compared to the scale and PDF errors. Finally, Fig. 3 compares the results of xsec to the corresponding NLO result from NNLL-fast based on the same PDF set. We observe a slight systematic difference at the $\sim$1% level. This is at the level of the interpolation error quoted by NNLL-fast.

The same gluino pair-production cross-section as a function of a common squark mass for a selection of different gluino masses can be found in Fig. 4, showing that xsec reproduces the feature across the whole assumed range of validity for the regression. As expected, the regions in which the cross-section changes most rapidly are the most difficult to predict, but we see that with one exception the prediction is always within 2σ of the Prospino value, over a large number of test points.

In Figs. 5 and 6, we show the potential importance of being able to deal with non-degenerate squark masses in xsec. Here, we vary the $\tilde{u}_L$ and $\tilde{d}_R$ mass alone, respectively, with all other squarks fixed at 1 TeV and the gluino at 1.5 TeV for

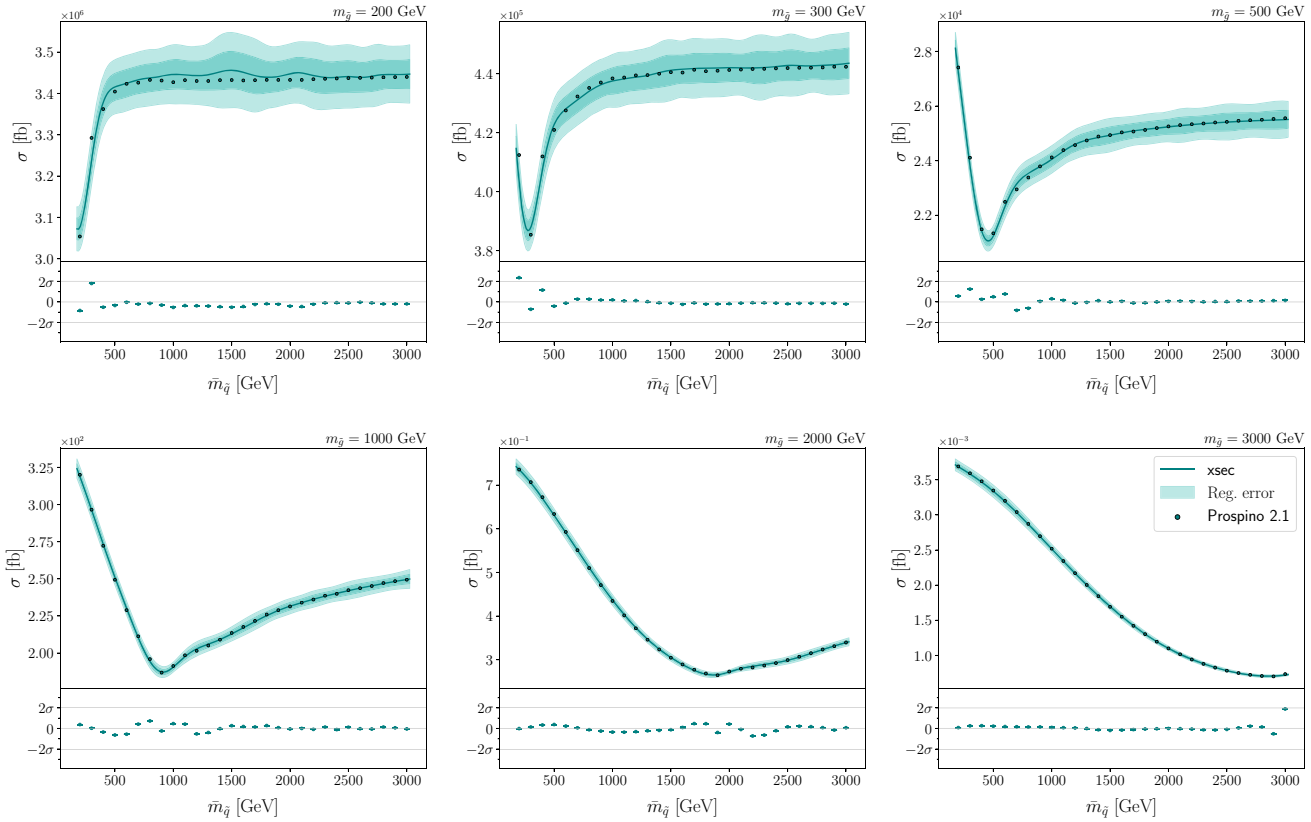


Fig. 4 Gluino pair-production cross-section as a function of average first and second-generation squark mass for a set of different gluino masses. Shown is the central xsec prediction (solid line), the 1σ and 2σ regression error bands (shaded regions), and the Prospino values (dots). We also show the residuals of the comparison to Prospino

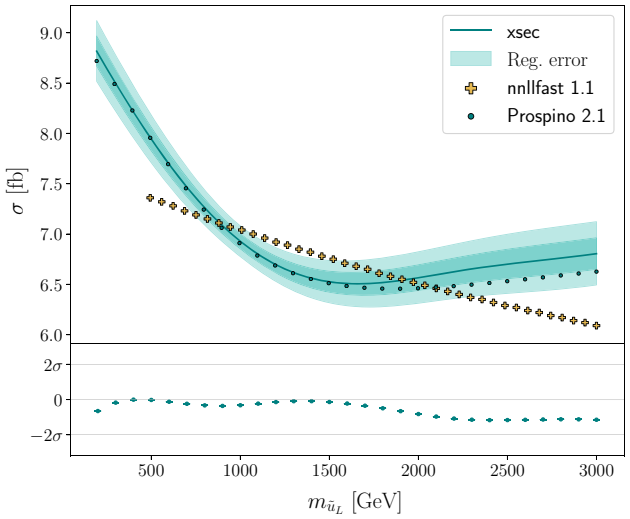


Fig. 5 Gluino pair-production cross-section as a function of the $\tilde{u}_L$ mass. All other squark masses are fixed at 1 TeV and the gluino mass is set to 1.5 TeV. Shown is the central xsec prediction (solid line), the 1σ and 2σ regression error bands (shaded regions), the Prospino values (dots) and the corresponding NNLL-fast NLO result (crosses)

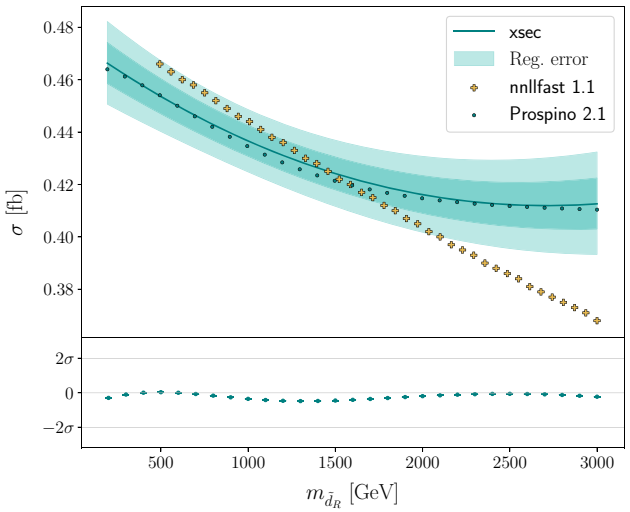


Fig. 6 Gluino pair-production cross-section as a function of the $\tilde{d}_R$ mass. All other squark masses are fixed at 1 TeV and the gluino mass is set to 2 TeV. Shown is the central xsec prediction (solid line), the 1σ and 2σ regression error bands (shaded regions), the Prospino values (dots) and the corresponding NNLL-fast NLO result (crosses)

$\tilde{u}_L$ and 2 TeV for the plot with $\tilde{d}_R$. While less pronounced, qualitatively the same dip feature due to the t-channel interference as discussed above can be seen here as well. In this example, the NNLL-fast NLO result fails to reproduce the feature, as expected from its inherent assumption of degenerate squark masses.

Figure 7 shows the distributions of the relative error (left) and the residual (right) for the central cross-section value using the test sets $\mathcal{D}_{\text{test}}$ and $\mathcal{D}_{\text{MSSM-24}}$. All distributions are normalised to unity, and the y-axis range is set to show all bins with non-zero values. In both sets the relative error is well below 5% for most points, and in fact there is only a single point, in the $\mathcal{D}_{\text{MSSM-24}}$ set, with an error greater than 5%. From the comparison to the unit normal distribution we see that, as expected, the xsec regression error is somewhat larger than the true error, but with no apparent bias. The xsec prediction is also robust under a change of the test sample to the MSSM-24.

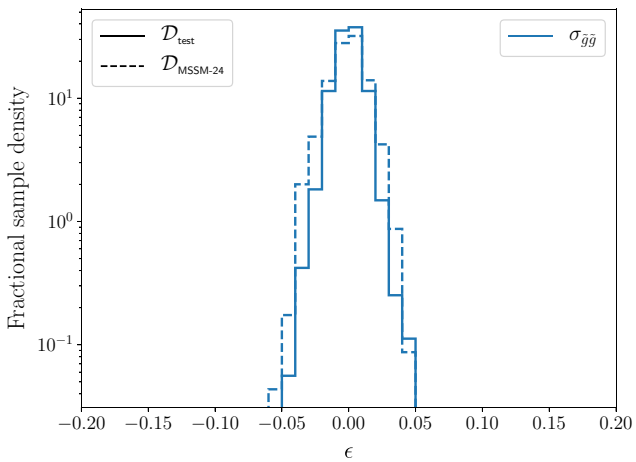
It is also instructive to perform two-dimensional grid scans of mass planes to show the relative error and residual of the central cross-section value as a function of two of the features at a time. Two examples of this are found in Fig. 8, where we show the result in the planes of $(m_{\tilde{g}}, \bar{m}_{\tilde{q}})$ and $(m_{\tilde{g}}, m_{\tilde{d}_R})$. We see that the relative errors and residuals are correlated in the mass planes, as should be expected from Gaussian processes when the dominant uncertainty is not due to random noise in the training data, but rather due to the lack of information in regions where the function is changing quickly. We also note that the regression uncertainty is largest when $m_{\tilde{g}} \approx m_{\tilde{q}}$. This shows that the destructive interference dip seen in Figs. 3, 4 and 5 is the part of this cross-section function that is the most challenging to capture in the regression. Further improvements could be made by adding extra training points, but only at significant cost to the evaluation speed, which seems unwarranted given the small regression errors compared to the other errors. Naive counting of the number of bins in the residual plots (right panels) above the 1, 2, and 3σ levels indicates that the quoted xsec regression error is in general conservative compared to the actual error (i.e. fewer bins show large residuals than expected from Gaussian statistics).

In addition to the regression errors, xsec also predicts scale, PDF and α_s errors from separate GPs performing regression on the relative size of the respective error bands, see Sect. 3.3. The performance of these GPs in the case of gluino pair-production is shown in Fig. 9 in terms of the resulting relative error on the errors, compared to values calculated using Prospino for the $\mathcal{D}_{\text{test}}$ and $\mathcal{D}_{\text{MSSM-24}}$ test sets. To be precise, we compute the analogue of Eq. (39) for each individual type of error:

$$\epsilon(\mathbf{x}_*) = \frac{\delta_{\text{prosp}}(\mathbf{x}_*) - \delta_{\text{xsec}}(\mathbf{x}_*)}{\delta_{\text{prosp}}(\mathbf{x}_*)}, \tag{43}$$

where $\delta_{\text{prosp}}(\mathbf{x}_*)$ represents the true width of the scale, PDF, or α_s error band at the test point $\mathbf{x}_*$, as computed with Prospino, and $\delta_{\text{xsec}}(\mathbf{x}_*)$ is the corresponding xsec estimate.

While PDF and α_s errors computed by xsec are symmetric by default (see Sect. 5.3), the scale errors are not. We therefore symmetrise these before adding them in quadrature to the PDF and α_s errors, to obtain a combined error. We find that the relative error on this combined error is below 10% in over 90% of the test points.



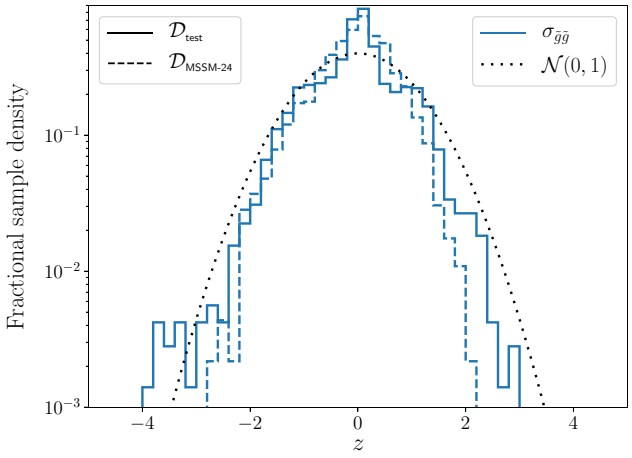


Fig. 7 The relative error (left) and residual (right) distributions for the gluino pair-production cross-section in the test sets $\mathcal{D}_{\text{test}}$ (solid) and $\mathcal{D}_{\text{MSSM-24}}$ (dashed). The input points in $\mathcal{D}_{\text{test}}$ are sampled from the same distribution as the training set, while the points in $\mathcal{D}_{\text{MSSM-24}}$ are sampled from the MSSM-24 using flat priors for the MSSM parameters. All distributions are normalised to unity. The unit normal distribution is shown for comparison as a dotted black line

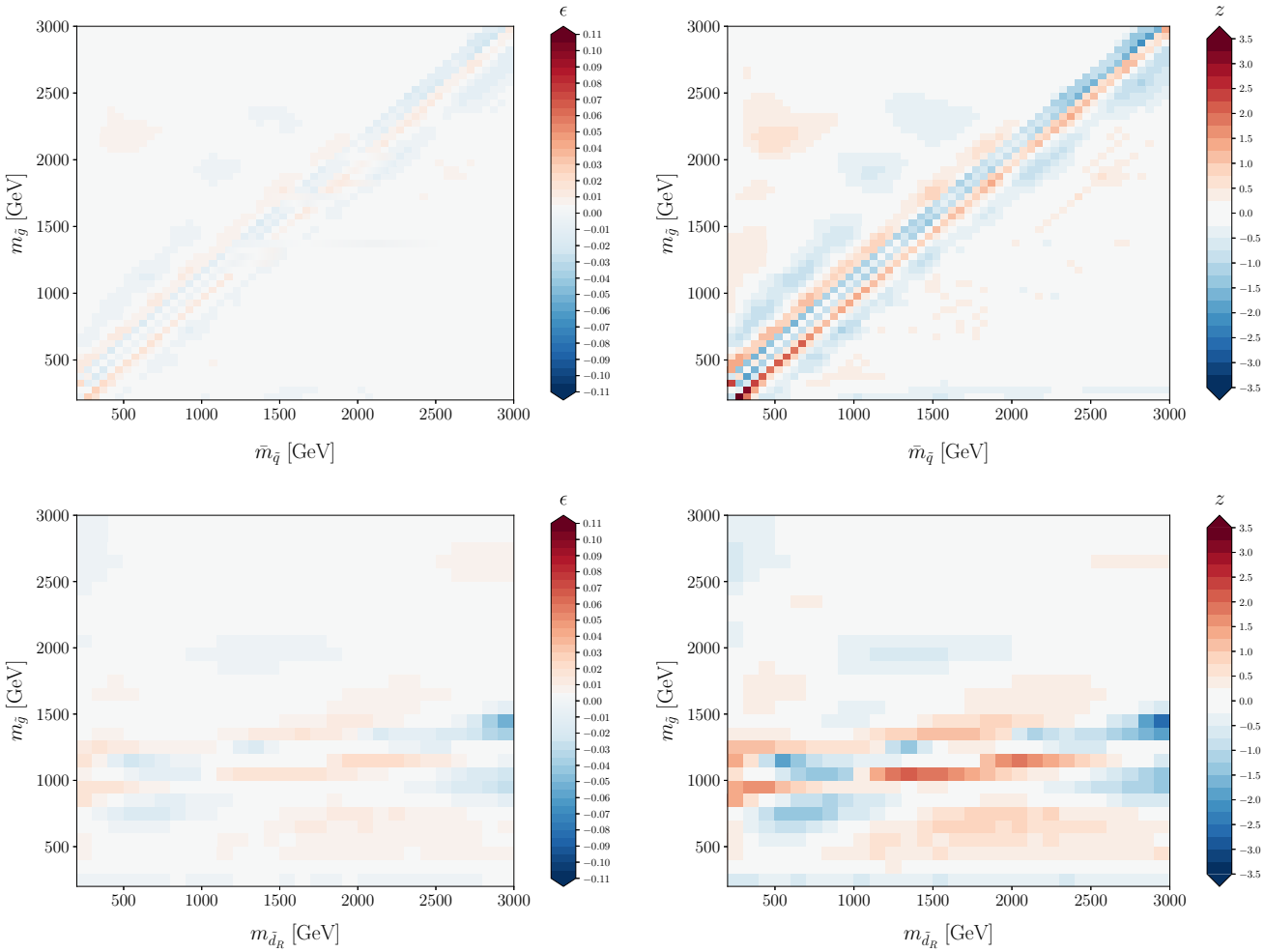


Fig. 8 The relative error (left) and residual (right) for the gluino pair-production cross-section as a function of the gluino mass versus a common squark mass (top) and the $\tilde{d}_R$ mass (bottom). All other masses are fixed at 1 TeV

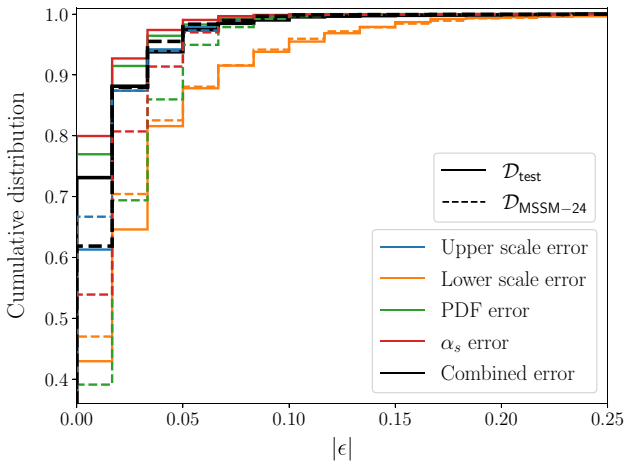


Fig. 9 Cumulative distribution of the relative error magnitude for the upper and lower scale errors, the PDF error and the α_s error for the gluino pair-production process

Given that the absolute magnitude of the cross-section is much larger than the absolute magnitude of the individual errors, these errors on errors are largely insignificant. Very similar conclusions can be reached for the scale, PDF and α_s errors for the other processes included in xsec, and for the sake of brevity we do not show the relative errors on the errors for other processes.

4.2 Pair production of gluinos with first or second-generation squarks

The data that we employ for training xsec 1.0 are limited by the fact that Prospino assumes flavour conservation and neglects heavy quarks in the proton PDFs. It therefore offers gluino–squark pair-production cross-sections only for processes with first and/or second-generation squarks in the final state. However, cross-sections for gluino–squark production with sbottoms or stops in the final state are expected to be very small. We also note that Prospino returns the cross-section for the sum over charge-conjugate final states, i.e. $\tilde{g}\tilde{q}_i + \tilde{g}\tilde{q}_i^*$, making it pointless to train xsec separately on the two final states. In addition to these limitations, at NLO QCD the numerical value of the cross-section is identical for left and right-handed squark final states, as long as their masses are identical. Although we use this fact internally in xsec to reduce the total file size of the DGPs, the user can freely request any first or second-generation final-state squark.

The sizes of the gluino–squark production cross-sections are naturally dominated by the gluino and final-state squark masses. Because of flavour conservation, to the level of approximation used in Prospino's K-factor calculation, the only additional property of the model parameter space (i.e. feature) used by xsec is the average squark mass $\bar{m}_{\tilde{q}}$. The range of sparticle masses over which we assume this cross-section evaluation to be valid is the same as for gluino pair production, i.e. [200, 3000] GeV.

In Fig. 10 we show the predicted gluino–squark production cross-sections as a function of the gluino mass (left) and the individual $\tilde{q}_L$ masses (right). For the individual squark masses we keep all other masses at 1 TeV and change only the mass of the final-state squark. Also shown are the predicted regression, PDF and scale errors, and the residual between the xsec predictions and the Prospino values calculated for the same parameters. We see that xsec reliably predicts the contribution from individual squark final-state flavours. This is even the case in the region of very low final-state squark masses, which tests xsec on the arguably strange scenario in which the particular final-state squark is much lighter than the average mass of the first and second-generation squarks.

For the $\tilde{g}\tilde{d}_L$ and $\tilde{g}\tilde{u}_L$ processes, the scale error is the dominant uncertainty across the full mass range in both the gluino mass and the final-state squark mass. For the $\tilde{g}\tilde{s}_L$ and $\tilde{g}\tilde{c}_L$ processes it is generally the PDF error that dominates the uncertainty, except at low gluino masses, where the scale error is more important. The fact that the regression error residuals comparing the xsec and Prospino values seem correlated between the four processes shown is due to the same training sample being used for all processes, causing the distances to the nearest, most influential training points to be the same in all cases.

In Fig. 11 we show the distributions of the relative error between the xsec prediction and the corresponding Prospino results, as well as the residual, for each individual flavour final state. We use the same two test sets, $\mathcal{D}_{\text{test}}$ and $\mathcal{D}_{\text{MSSM-24}}$, as for the gluino pair-production cross-section in Fig. 7. All the distributions are normalised to unity.

The relative errors and residuals are similar across all squark flavours. There are no obvious differences in performance with the two sets of test points. The relative error distributions show that for almost all points in the test sets, the true regression error is below 10%, and xsec tends to overestimate the Prospino cross-section by a few percent. Comparing the residual and $\mathcal{N}(0, 1)$ distributions, we see that the predicted xsec regression uncertainty is conservative; indeed, notably more so than for the gluino pair-production cross-section (Fig. 7).

We can also compare the relative error across mass planes, which is shown in Fig. 12 separately for all $\tilde{g}\tilde{q}_L$ processes, in terms of the gluino mass and the average first and second-generation squark mass. Here the final-state squark mass for each plotted cross-section is set equal to the average squark mass. We see that the regression error is below 8% across this plane for all four of the $\tilde{g}\tilde{d}_L$, $\tilde{g}\tilde{u}_L$, $\tilde{g}\tilde{s}_L$, and $\tilde{g}\tilde{c}_L$ production cross-sections.

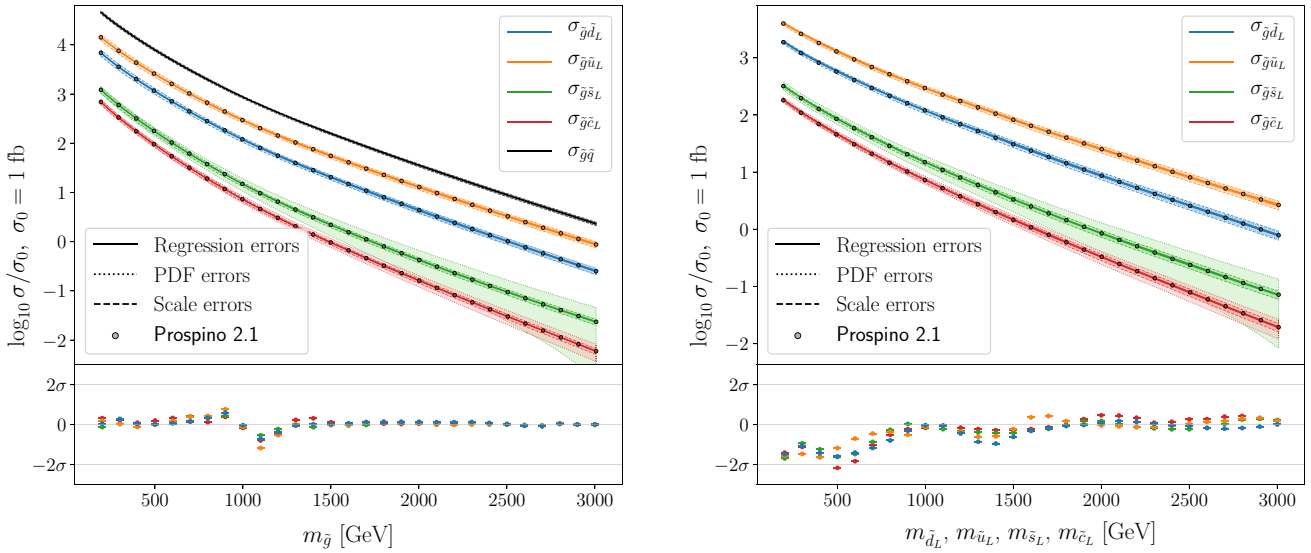


Fig. 10 Gluino–squark pair-production cross-section as a function of gluino mass (left) and squark masses (right), for production of first and second-generation squarks. Shown are individual (left-handed) squark final states (colours) and the sum of all first and second-generation final states (black). In the left-hand plot all squark masses are fixed at 1 TeV. In the right-hand plot all masses except for the final-state squark mass are fixed at 1 TeV. The central-value xsec prediction is shown with error bands from regression (solid line), scale error (dashed) and PDF error (dotted). The α_s error is too small to be visible. Also shown are the Prospino values (dots)

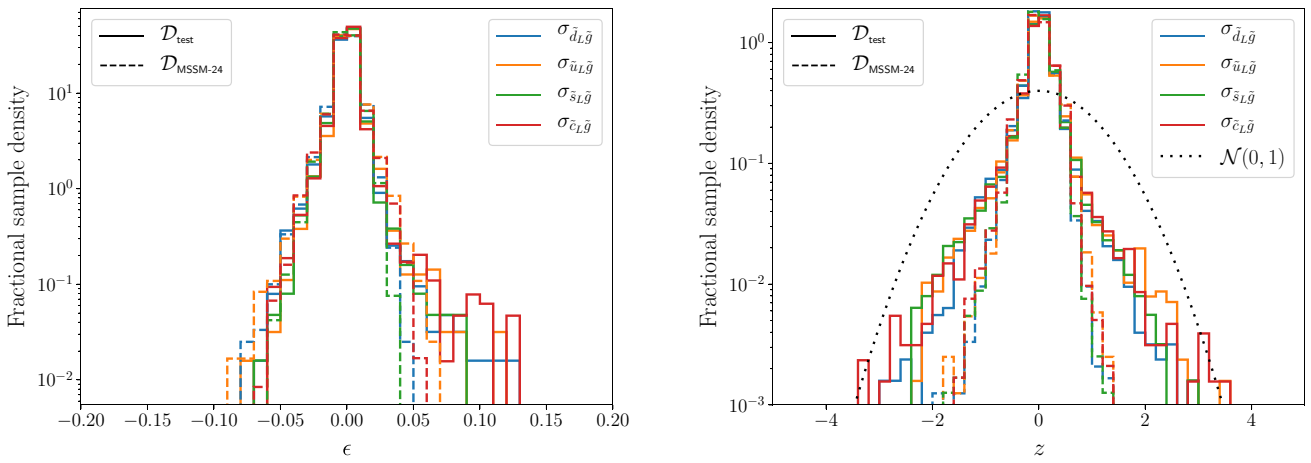


Fig. 11 The relative error (left) and residual (right) distributions for the first and second-generation gluino-squark cross-sections, for the test sets $\mathcal{D}_{\text{test}}$ (solid) and $\mathcal{D}_{\text{MSSM-24}}$ (dashed). All distributions are normalised to unity. The unit normal distribution is shown as a dotted black line for comparison to the residual distributions

4.3 First and second-generation squark–anti-squark pair production

We now look at the production cross-sections for squark–anti-squark pairs $\tilde{q}_{L/R}\tilde{q}^{(\prime)*}_{L/R}$. The flavours of the pair may be identical or different, and all four combinations of squark handedness are treated as separate processes. If the flavours of the two squarks are different, the process is assumed to include the charge-conjugate state. In this section we discuss only final states with first and second-generation (anti-)squarks, where the final-state flavour may come from first and second-generation (anti-)quarks sampled from the proton.

Within the limitations set by the training data from Prospino, the LO first and second-generation squark–anti-squark cross-sections depend on the masses of the gluino and the final-state squark(s), and the NLO corrections further on the mean mass of all first and second-generation squarks. Again, we validate the cross-section on the sub-interval [200, 3000] GeV of the training data (in both gluino

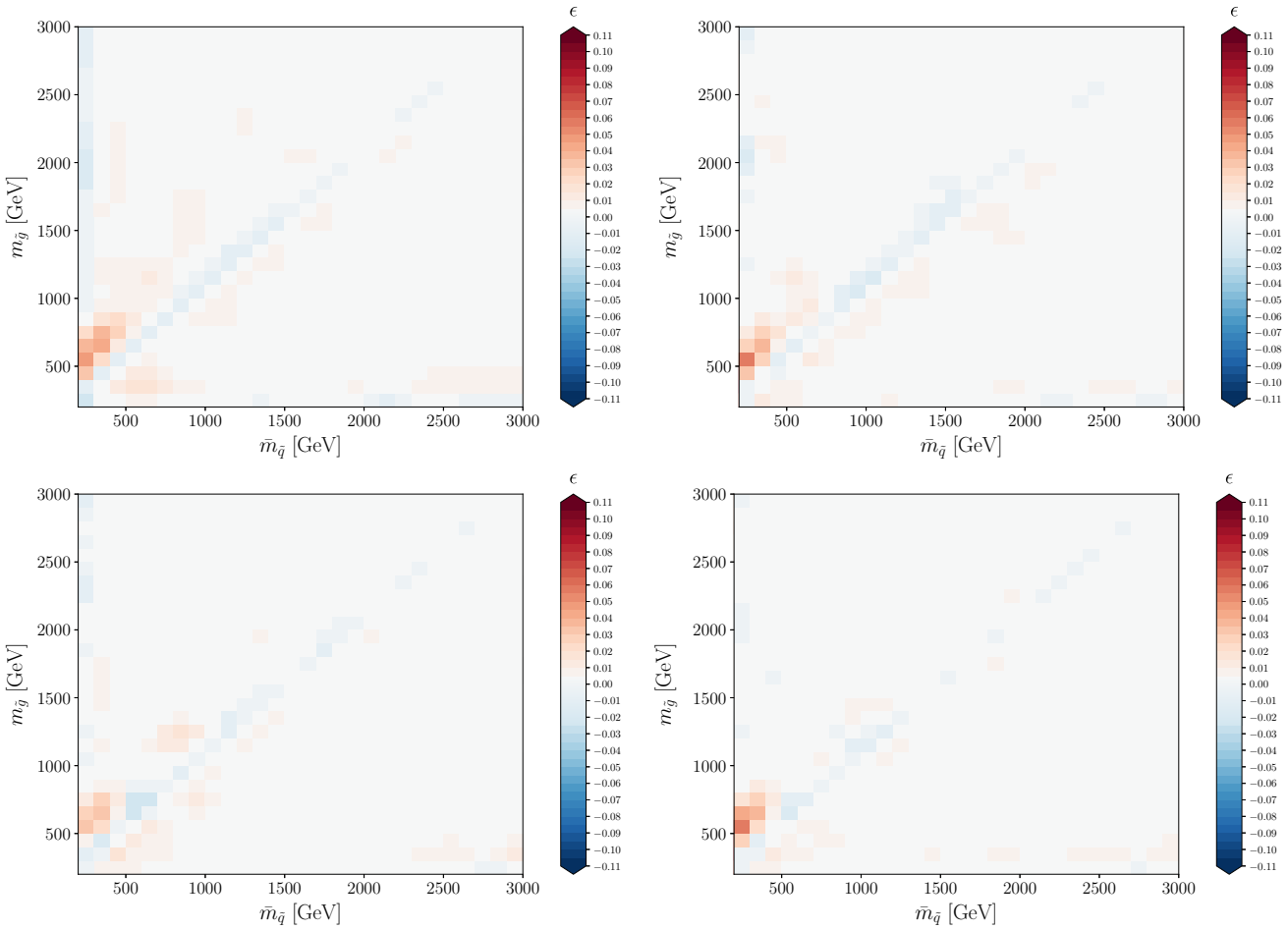


Fig. 12 The relative error of the production cross-section for gluinos and first or second-generation squarks, as a function of the average mass of first and second-generation squarks and the mass of the gluino. Shown are results for the production of $\tilde{g}\tilde{d}_L$ (top left), $\tilde{g}\tilde{u}_L$ (top right), $\tilde{g}\tilde{s}_L$ (bottom left), and $\tilde{g}\tilde{c}_L$ (bottom right). The final-state squark mass for each process is set equal to the average squark mass

and squark mass). For squark–anti-squark production, one should keep in mind that masses below the lower end of this range may be affected by resonant production through Z and W, and while xsec's reported regression error increases below 200 GeV, it cannot take these resonances into account, as they are not included in its training data. The resulting cross-sections reported by xsec must thus be seen as wholly unreliable for squark masses below 50 GeV.

In NLO QCD, cross-sections for the two sets of process pairs $(\tilde{q}_L\tilde{q}_L^*, \tilde{q}_R\tilde{q}_R^*)$ and $(\tilde{q}_R\tilde{q}_L'^*, \tilde{q}_R'\tilde{q}_L^*)$ differ within each set only by an exchange of the appropriate squark masses. Removing also charge-conjugate states, an initial number of 64 independent processes $\tilde{q}_{L/R}\tilde{q}_{L/R}'^*$ therefore reduces to 20 unique cross-sections. To save training time and user disk space, we reuse the DGPs for the identical processes in xsec simply by employing symbolic links and mapping the masses accordingly. This is however invisible to the user.

In Fig. 13 we show the predicted first and second-generation squark–anti-squark production cross-sections as a function of the mean first and second-generation squark mass $\bar{m}_{\tilde{q}}$ (left) and the gluino mass (right). We show results for a selection of sub-processes, and for the total cross-section (black, rescaled for readability). All other masses are kept at 1 TeV. We see that xsec reliably predicts the contribution from individual squark final states, although at high squark masses the PDF error (dotted line) for some processes is consistent with zero cross-section. We also see that xsec correctly captures the contribution of the gluino t-channel diagram, which controls the cross-section when the final-state squarks have different chirality, leading to the peak in Fig. 13 (right) for L-R combinations when $m_{\tilde{g}} \simeq m_{\tilde{q}}$.

In Fig. 14 we show the distributions of the relative regression error and residual (Eqs. (39) and (40)) for the points in the test sets $\mathcal{D}_{\text{test}}$ and $\mathcal{D}_{\text{MSSM-24}}$. We have normalised all distributions to unity. The comparison to the unit normal distribution included in Fig. 14 (right column) shows that for both test sets, and all processes, the xsec regression error is conservative with respect to the true error.

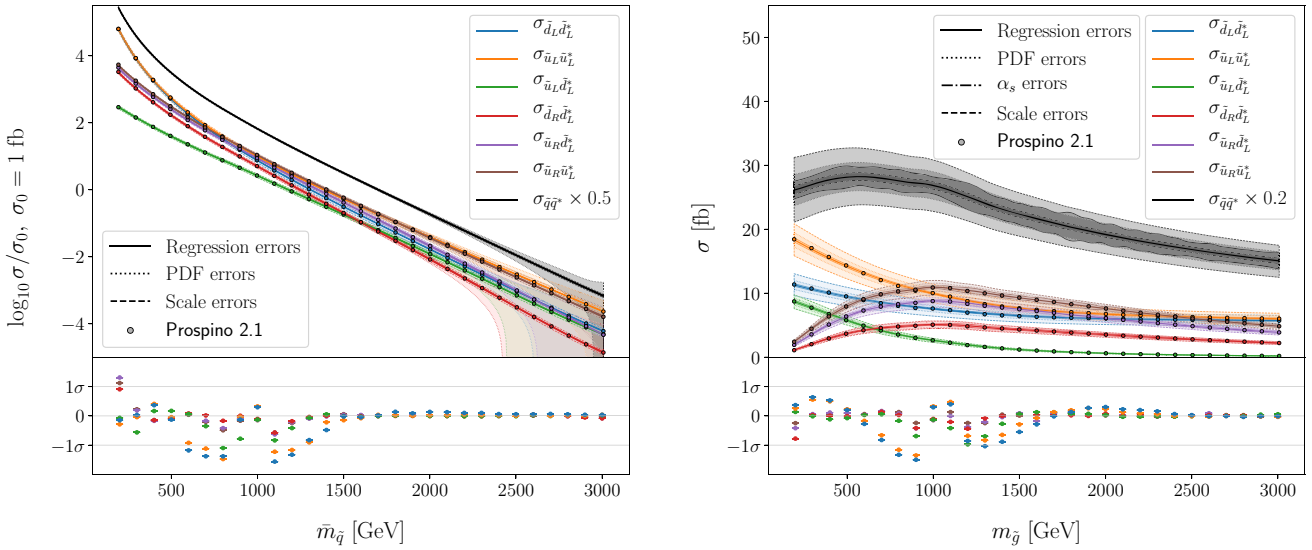


Fig. 13 Squark–anti-squark pair-production cross-sections for first and second-generation squarks. Panels show cross-sections as a function of the average of all first and second-generation squark masses (left) and gluino mass (right), for a selection of final states (colours) and the total cross-section for production of first and second-generation squark–anti-squark pairs (black)

The relative regression error in Fig. 14 (left column) is below 10% for all processes for the vast majority of test points. There is again a slight tendency for xsec to overestimate the Prospino cross-section values, in particular for the $\mathcal{D}_{\text{MSSM-24}}$ test set. As this set has individual flat priors for all the squark (soft) masses, which only a subset of $\mathcal{D}_{\text{test}}$ has, we expect it to be more challenging to reproduce as its points are more likely to lie on the outskirts of the validation region.

The relative error distributions are most narrow for the processes producing a squark–anti-squark pair of the same type, i.e., for the four $\tilde{q}_L^*\tilde{q}_L$ processes and the corresponding $\tilde{q}_R^*\tilde{q}_R$ processes (not shown). These cross-sections are easier to model, as the final state involves only a single mass parameter. The $\tilde{s}_L^*\tilde{s}_L$ and $\tilde{c}_L^*\tilde{c}_L$ processes have particularly small relative errors. This is likely due to the smallness of the proton PDFs for the s and c quarks, which effectively makes the gluino t-channel diagram irrelevant and thus further simplifies the parameter dependence.

4.4 First and second-generation squark pair production

This section looks at the validation of DGPs trained to predict cross-sections for squark–squark pair production, $\tilde{q}_{L/R}\tilde{q}_{L/R}^{(\prime)}$. As should be clear from the notation, the flavours of the pair may be identical or different, and all four combinations of squark handedness are treated as separate processes. The processes are always assumed to include the charge-conjugate state. Again, we discuss only final states with first and second-generation squarks, where at LO the final-state flavour comes from first and second-generation quarks in the proton.

As for squark–anti-squark production, within the limitations set by the design of Prospino, the LO first and second-generation squark–squark pair-production cross-sections depend on the mass(es) of the final-state squarks and the gluino mass, and the NLO corrections further depend on the mean mass of the first and second-generation squarks. We again validate our cross-sections on the sub-interval [200, 3000] GeV of the training data, for both gluino and squark masses.

If the squark masses are interchanged appropriately, the cross-sections for the process pairs $(\tilde{q}_L\tilde{q}_L, \tilde{q}_R\tilde{q}_R)$ are identical in Prospino, as are those for the process pairs $(\tilde{q}_R\tilde{q}'_L, \tilde{q}'_R\tilde{q}_L)$. The 64 independent processes $\tilde{q}_{L/R}\tilde{q}'_{L/R}$ can therefore be reduced to 20 unique cross-sections. Again we use symbolic links to reuse DGPs for processes with identical cross-sections.

In Fig. 15 we show the predicted first and second-generation squark–squark production cross-sections for a selection of sub-processes, as a function of $\bar{m}_{\tilde{q}}$ and $m_{\tilde{g}}$. All other masses are kept at 1 TeV. For readability, the total cross-section is rescaled. We see that xsec reliably predicts the contribution from individual squark final-state flavours, and captures the contribution of the gluino t-channel diagram, which depends on the chirality of the final-state squarks, giving the qualitatively different behaviour of the cross-section as a function of the gluino mass seen in the right panel of Fig. 15.

In Fig. 16 we show the corresponding distributions of the relative error and residual between the xsec prediction and the Prospino central cross-section values in $\mathcal{D}_{\text{test}}$ and $\mathcal{D}_{\text{MSSM-24}}$, for each individual squark–squark process. The

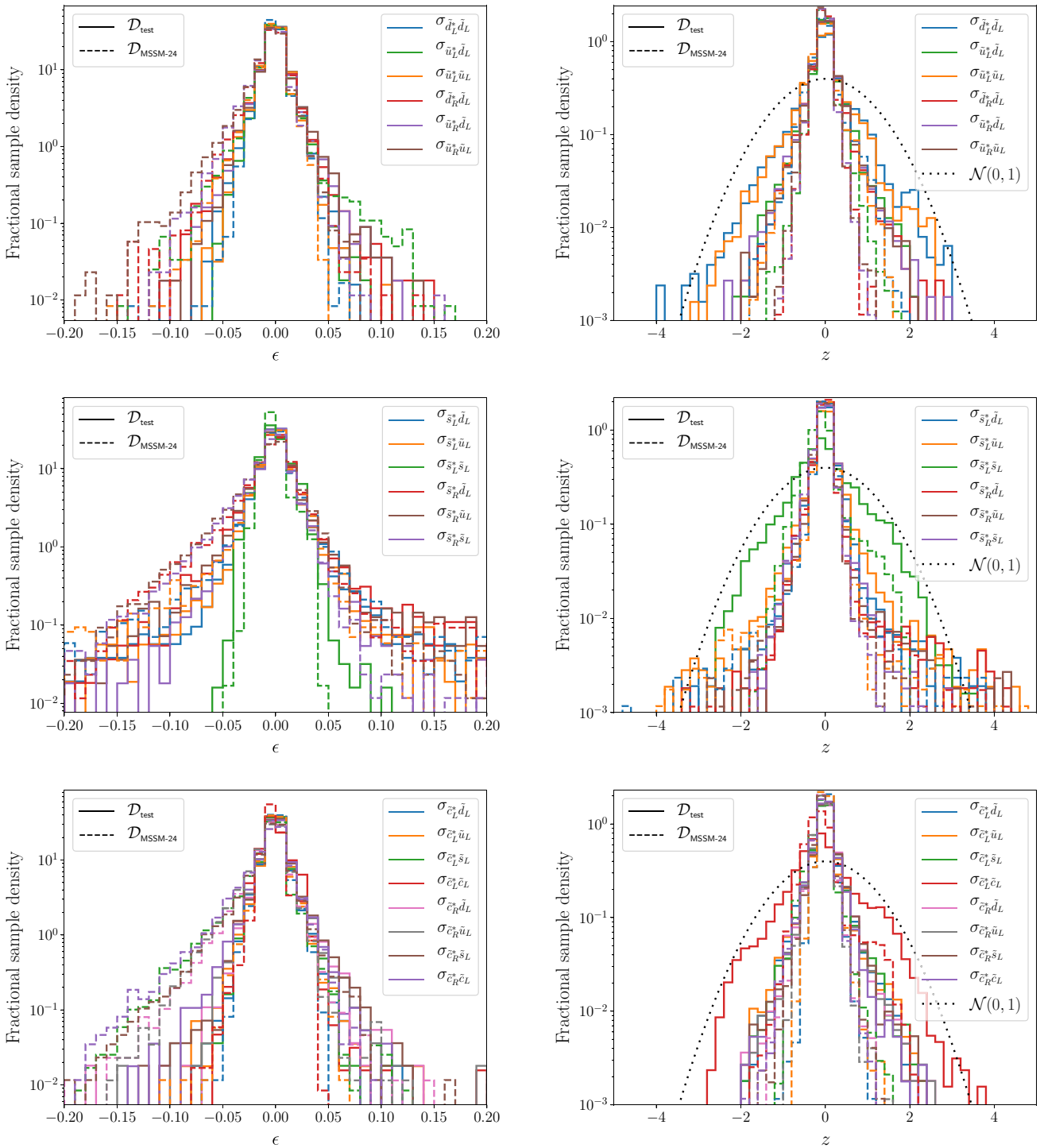


Fig. 14 The relative error (left) and residual (right) distributions for the first and second-generation squark–anti-squark cross-sections for the test sets $\mathcal{D}_{\text{test}}$ (solid) and $\mathcal{D}_{\text{MSSM-24}}$ (dashed)

residuals indicate that the regression errors from xsec across all squark–squark processes are generally conservative compared to the actual difference between the true and predicted cross-sections. The relative errors are below 10% for the large majority of test points, across all processes. There is some interesting process-dependent structure, however. For processes with two identical final-state squarks, e.g. $\tilde{u}_L\tilde{u}_L$ or $\tilde{s}_L\tilde{s}_L$, the relative error is quite small, due to the dependence of the cross-section upon only three instead of four masses, making it easier to predict reliably. Processes with chirally-

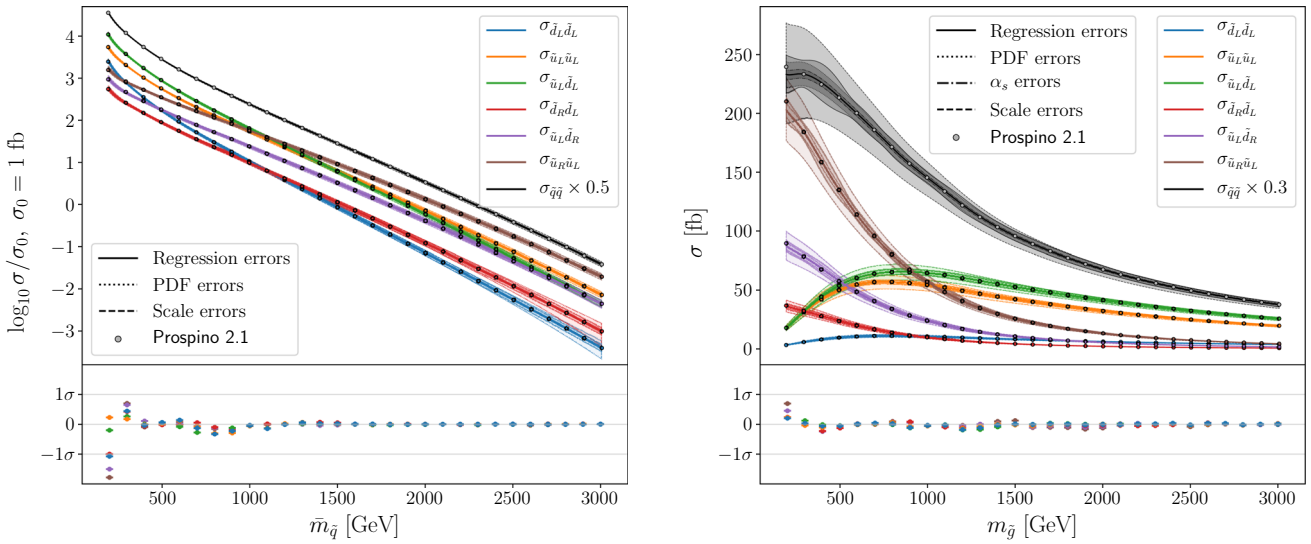


Fig. 15 First and second-generation squark–squark pair-production cross-section as a function of the average first and second-generation squark mass (left) and gluino mass (right), for a selection of final states (colours) and the (rescaled) total cross-section for first and second-generation squark–squark production (black)

matched squarks, i.e. LL or RR final states, have smaller errors than LR final states. This is due to a difference in the LO matrix element, where the LR final states depend on $(t-m_{\tilde{q}}^2)(u-m_{\tilde{q}}^2)-m_{\tilde{q}}^2 s$, whereas LL and RR final states are proportional to $m_{\tilde{g}}^2 s$ [39]. This more complicated kinematic dependence means that the LR final states are harder to train and show larger relative errors.

4.5 Stop and sbottom pair production

Prospino includes PDF contributions only from light quarks and gluons. This means that production of third-generation squarks is purely s-channel in our training samples, so only flavour-neutral squark–anti-squark final states are available, i.e. there are no processes with third-generation squarks in gluino–squark nor squark–squark production. However, the third-generation mass eigenstates used by Prospino include left-right mixing, quantified through the sbottom and stop mixing angles $\cos\theta_{\tilde{b}}$ and $\cos\theta_{\tilde{t}}$, where $\cos\theta_{\tilde{t}} = 1$ implies that $\tilde{t}_1, \tilde{t}_2 = \tilde{t}_L, \tilde{t}_R$, and similarly for sbottoms. Furthermore, Prospino does not calculate cross-sections for the production of mixed $\tilde{t}_1\tilde{t}_2^* + \text{c.c.}$ (or $\tilde{b}_1\tilde{b}_2^* + \text{c.c.}$) states, as their cross-sections are of order α_s^4, resulting in negligibly small rates [9]. As a result, xsec is limited to training the third-generation processes $\tilde{b}_1\tilde{b}_1^*$, $\tilde{b}_2\tilde{b}_2^*$, $\tilde{t}_1\tilde{t}_1^*$, and $\tilde{t}_2\tilde{t}_2^*$.

For the stop and sbottom pair-production cross-sections, only the final-state mass enters at LO as a parameter (feature). To NLO in QCD, i.e. up to $\mathcal{O}(\alpha_s^3)$, the cross-sections depend on the stop or sbottom final-state mass, the gluino mass, the averaged first and second-generation squark mass, and the stop or sbottom mixing angle, in roughly descending order of importance. Unlike the processes involving first and second-generation squarks, the sbottom and stop cross-sections also include the dependence on all the individual sbottom and stop masses in loops. However, for sbottom (stop) final states, the dependence on the stop (sbottom) masses was found to be very small, and we do not use them as features in the GPs.

For the NLO contributions involving heavy-quark loops, we have used top and bottom masses of $m_t = 172.0\,\text{GeV}$ and $m_b = 4.6\,\text{GeV}$. However, we have checked that the effect of changing these within current experimental uncertainties is numerically irrelevant.

Due to the symmetry of the cross-section expressions at NLO, $\sigma_{\tilde{t}_1\tilde{t}_1^*} \leftrightarrow \sigma_{\tilde{t}_2\tilde{t}_2^*}$ under the combined interchange

$$\begin{Bmatrix} m_{\tilde{t}_1} \\ \cos\theta_{\tilde{t}} \\ \sin\theta_{\tilde{t}} \end{Bmatrix} \leftrightarrow \begin{Bmatrix} m_{\tilde{t}_2} \\ -\sin\theta_{\tilde{t}} \\ \cos\theta_{\tilde{t}} \end{Bmatrix}, \tag{44}$$

and similarly for sbottoms. There is also very little difference between the sbottom and stop cross-sections for the same masses. Nevertheless, xsec contains separate DGPs for all four non-zero stop and sbottom pair-production processes included in Prospino, in preparation for future extensions. Here we will focus our validation tests on the $\tilde{t}_1\tilde{t}_1^*$ cross-sections, and only discuss the other cross-sections when there are significant differences. We validate the xsec output for the third-generation squarks over a slightly larger mass range than for the other processes ($[100,\ 3000]\,\text{GeV}$), as there is still considerable interest in light stops.

In Fig. 17, we show the stop pair-production cross-section predicted by xsec as a function of the stop mass, which is by

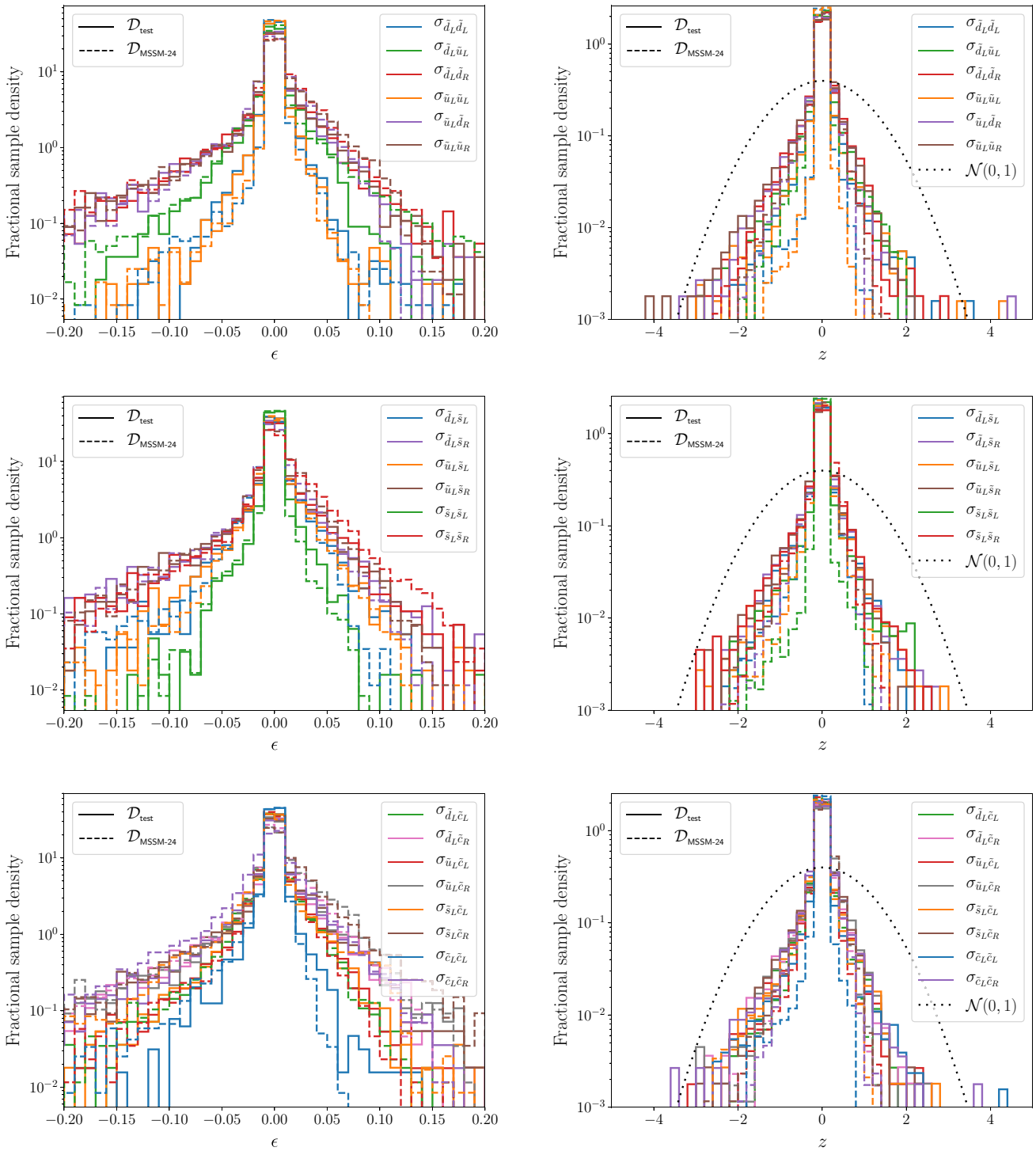


Fig. 16 The relative error (left) and residual (right) distributions for first and second-generation squark–squark pair-production cross-sections for the test sets $\mathcal{D}_{\text{test}}$ (solid) and $\mathcal{D}_{\text{MSSM-24}}$ (dashed)

far the dominant parameter in determining the cross-section. We also show the predicted scale and PDF uncertainties, and compare to values taken directly from Prospino. In generating the plot, we have set all stop and sbottom masses degenerate, $\cos\theta_{\tilde{t}} = 1$, and all other masses to 1 TeV. At low stop masses we can see that the scale error dominates, while at high masses it is the PDF error, as expected. Throughout the validation range the regression error is subdominant.

The gluino mass can be important for contributions that appear at NLO. In Fig. 18 we show the dependence of the

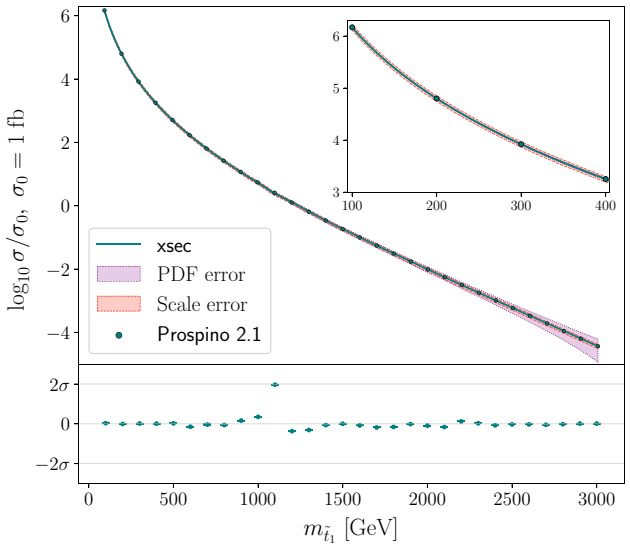


Fig. 17 Stop pair-production cross-section as a function of stop mass. The central-value xsec prediction is shown in light green, the scale error in pink, and the PDF error in violet. The 1σ α_s and regression error bands are too small to be visible. Superimposed on the prediction are the Prospino values (dots). Inset is a close-up of the region at low stop masses

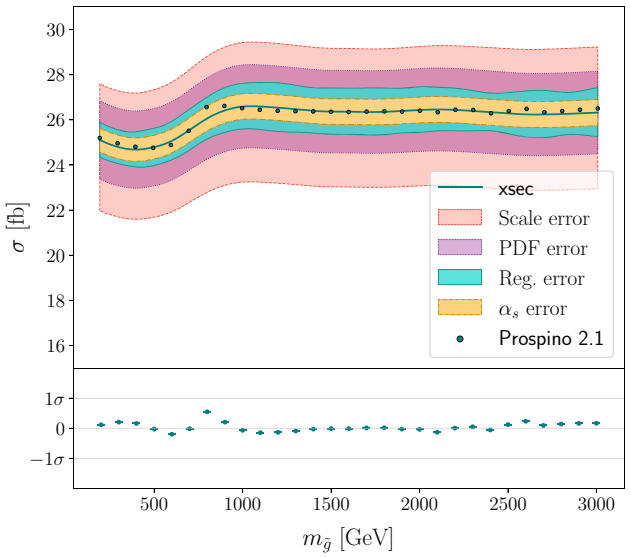


Fig. 18 Stop pair-production cross-section as a function of gluino mass for $m_{\tilde{t}_1} = 800$ GeV, with all other squark masses set to 1 TeV. The central-value xsec prediction and the 1σ regression error band is shown in light green, the scale error in pink, the PDF error in violet, and the α_s error in yellow. Below we show the residual between the xsec and Prospino results

$\tilde{t}_1\tilde{t}_1^*$ cross-section on the gluino mass, adopting a stop mass of $m_{\tilde{t}_1} = 800$ GeV, and 1 TeV for all other masses. Clearly, xsec fully captures the variation in the cross-section due to the gluino contribution at low gluino masses, although this dependence is rather small compared to the scale, PDF, and even α_s uncertainties, which we also show. The Prospino

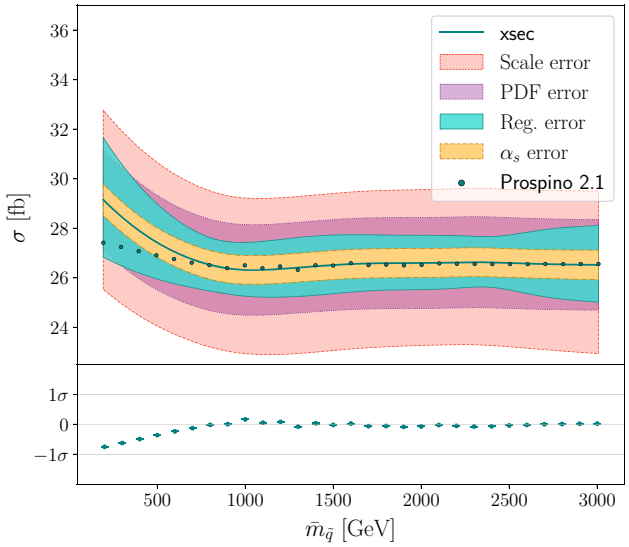


Fig. 19 Stop pair-production cross-sections as a function of the average of the first and second-generation squark masses, for $m_{\tilde{t}_1} = 800$ GeV. The xsec prediction and 1σ regression error is in light green, the scale error in pink, the PDF error in violet, and the α_s error in yellow. The Prospino values are shown as dots, and below we show the residual between the xsec regression result and Prospino

result seems to have some numerical jitter at high gluino masses, although the effect lies well within the regression uncertainty band.

In Fig. 19, we show the stop pair-production cross-section for $m_{\tilde{t}_1} = m_{\tilde{t}_2} = 800$ GeV as function of the average of the first and second-generation squark masses. To maximize the potential effect from squark loops, the sbottom masses are fixed to the same value as the first and second-generation squarks. We fix the mixing angle to $\cos\theta_{\tilde{t}} = 1$, while the gluino is decoupled at 3 TeV to remove the more dominant NLO contributions from gluino exchange. We see that the dependence of the cross-section on the other squark masses is so weak that the DGP is relatively insensitive to it. However, we again observe that the Prospino results have some jitter at high squark masses. After some investigation of the sampling, we found that this jitter increases as the gluino mass is increased, and we tentatively interpret this as a numerical problem in the gluino-decoupling regime in Prospino.

We present $\tilde{t}_1\tilde{t}_1^*$ and $\tilde{t}_2\tilde{t}_2^*$ production cross-sections as a function of the mixing angle $\theta_{\tilde{t}}$ in Fig. 20, for $m_{\tilde{t}_1} = m_{\tilde{t}_2} = 800$ GeV and all other sparticle masses set to 1 TeV. We can see that the dependence on the mixing angle is so small – of the order of 2% – that the limited resolution of the regression means that the DGPs cannot currently capture this behaviour. The error is nonetheless well within that reported by xsec.

Looking at the stop and sbottom residual distributions in $\mathcal{D}_{\text{test-tb}}$ and $\mathcal{D}_{\text{MSSM-24}}$ (right panel of Fig. 21), we see that they have somewhat longer tails than other process types. The most notable feature is at large negative z, indicating

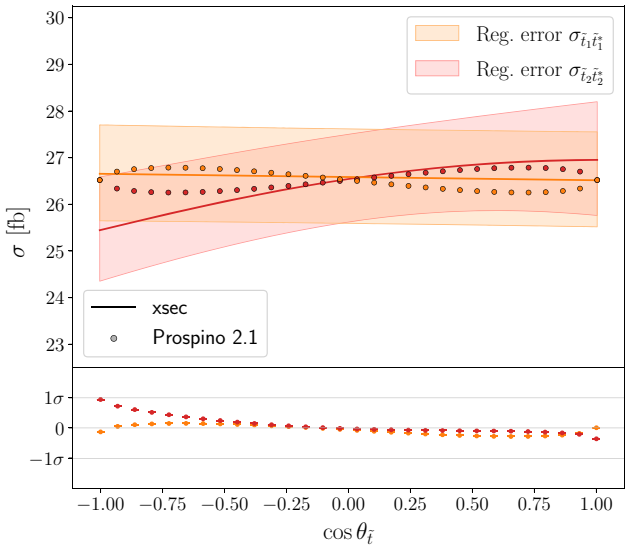


Fig. 20 Stop pair-production cross-sections as a function of the stop mixing angle $\cos\theta_{\tilde{t}}$. The central-value xsec prediction and the regression error band is shown in orange ($\tilde{t}_1\tilde{t}_1^*$) and red ($\tilde{t}_2\tilde{t}_2^*$). Superimposed on the prediction are the Prospino values (dots)

a slight tendency to overestimate the cross-section reported by Prospino. This is driven by points with large (> 2 TeV) gluino masses, where we earlier observed the jitter in the training data. Overall, however, the regression errors returned by xsec for the stop and sbottom processes are conservative. The corresponding relative errors are below 10% for the vast majority of test points, with some tails.

Finally, in Fig. 22, we show the relative error between the xsec-predicted $\tilde{t}_1\tilde{t}_1^*$ production cross-section and the Prospino value, in the planes of common stop mass and gluino mass (left), and stop mass and mixing angle (right). In these plots we set all the other masses to 1 TeV, and in the left-hand plot we fix $\cos\theta_{\tilde{t}} = 1$.

We find that xsec predicts the values with better than 10% accuracy in the majority of this space. However, the left-hand panel shows again the problem with numerical jitter from Prospino at large gluino (and stop) masses. We also observe a small localised area of underestimated cross-sections (positive relative error) for $m_{\tilde{t}_1} \sim 1100$ GeV and $m_{\tilde{g}} \sim 1000$ GeV, seen in both plots. This region can also be found in Fig. 17 as a single point where the residual is at the 2σ level.

5 Code structure and usage

5.1 Speeding up cross-section evaluation

To achieve fast cross-section predictions the xsec code comes with fully trained GP models. These contain all relevant information about the training points, such as their inverse covariance matrix and the optimised kernel hyperparameter values. The main program, written in Python, provides a simple user interface for predicting cross-sections and their uncertainties at a given set of input parameters.

With the time-consuming processes of sampling and training carried out beforehand, two important steps remain in order to obtain predictions for a new parameter point:

1. Performing the matrix-vector multiplications given in Eqs. (21) and (22) to compute predictions from each individual GP expert;
2. Weighting and combining the individual predictions according to the GRBCM prescription, leading to Eq. (31).

As the aggregation step requires no matrix algebra, the first step dominates the computational cost of prediction. It scales

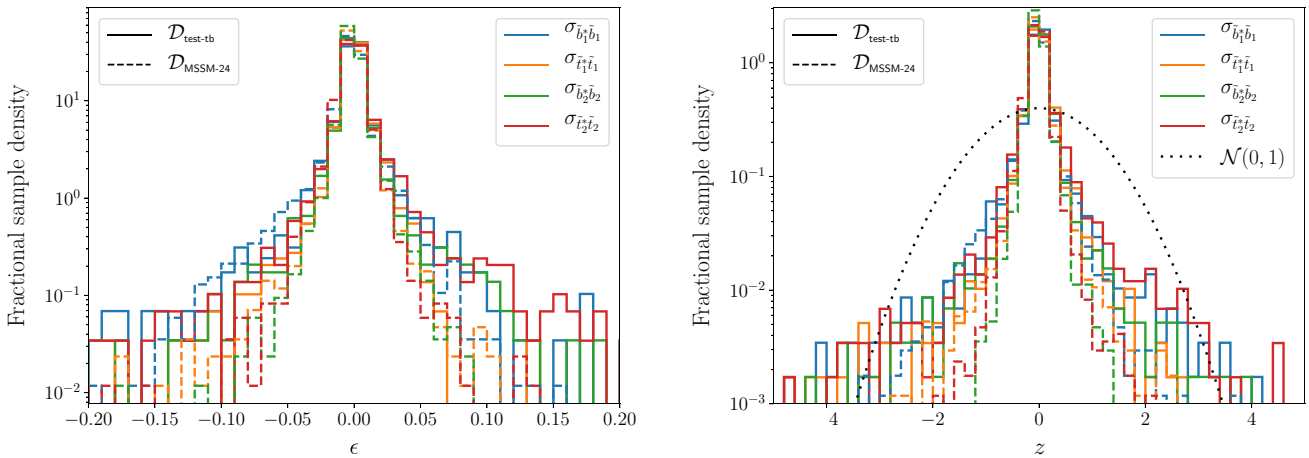


Fig. 21 Distributions for the relative error (left) and residual (right) for the stop and sbottom pair-production cross-sections for the test sets $\mathcal{D}_{\text{test-tb}}$ (solid) and $\mathcal{D}_{\text{MSSM-24}}$ (dashed)

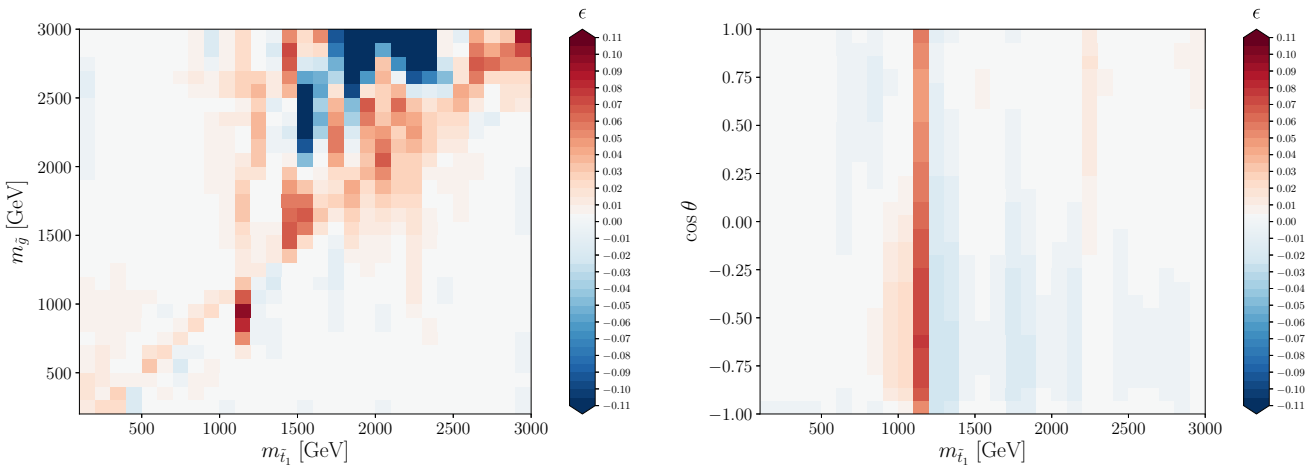


Fig. 22 The relative error between the predicted stop pair-production cross-section in xsec and the Prospino value as a function of the stop mass versus gluino mass (left) and stop mass versus mixing angle (right)

in complexity as the square of the number of training points of the GP expert.

The code relies on NumPy [35] functionality for these matrix operations. Significant speed gains are therefore possible if NumPy is properly linked to optimised numerical algebra libraries, like BLAS and LAPACK. Large performance differences exist between different BLAS implementations. Popular ones like Intel MKL, OpenBLAS and ATLAS support multi-threaded calculations, enabling NumPy to take advantage of multi-core machines and xsec to achieve the highest evaluation speeds. The NumPy package provides a function `sho_config()` to list the numerical libraries it detected in the system it was built on.

The xsec code is compatible with Python 2 and 3. While the main program consists of the six interdependent modules shown in Fig. 23, the intended usage requires no explicit knowledge of this underlying structure. All high-level functions of relevance to the user are immediately accessible upon installing and importing the xsec package, as detailed in the following sections.

We have not included the code used to train the GP models in the current version. Interested parties with specific use cases, or researchers trying to reproduce our results, will be given access to the private repository for this code upon request.

5.2 Installation

The xsec program is available from the Python Package Index (PyPI) and can be installed immediately with the pip package manager, by executing

```
pip install xsec
```

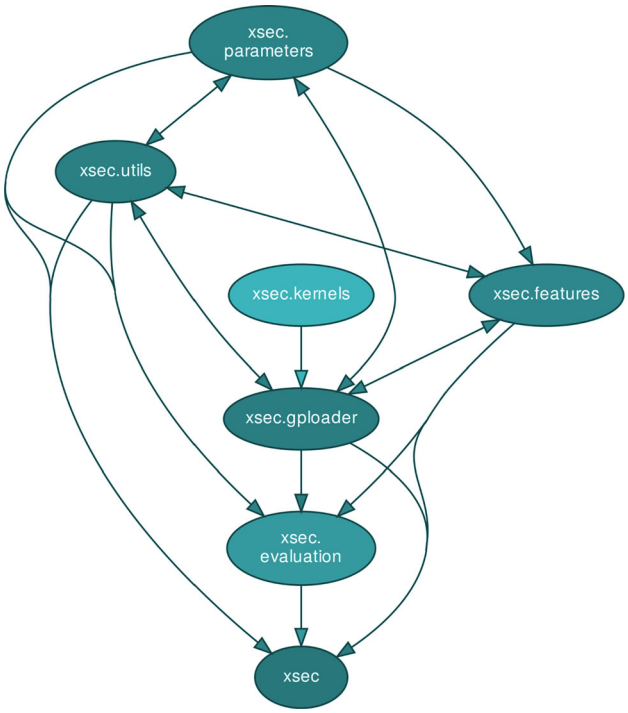


Fig. 23 Module dependency graph of xsec

in a terminal. Alternatively, the user may wish to access the code directly from its GitHub repository[7] and install a local copy:

```
git clone https://github.com/jeriek/
    xsec.git
pip install ./xsec
```

Due to the size of the trained GP datafiles, these are not contained in the repository, nor provided directly by `pip`.

[7] https://github.com/jeriek/xsec.

Instead, they are available as separate binaries in the GitHub release. We provide a script for their automatic download and extraction into a directory of choice, called by running

```
xsec-download-gprocs -g gp_dir -t
    process_type
```

on the command line. Here, *gp_dir* is the chosen target directory; a new directory `gprocs` is created in the current working directory if this argument is omitted. The other optional argument, *process_type*, specifies the final-state type for which data will be downloaded: `gg` (gluino pair production), `sg` (1st/2nd gen. squark–gluino pair production), `ss` (1st/2nd gen. squark pair production), `sb` (1st/2nd gen. squark–anti-squark pair production), `tb` (3rd gen. squark–anti-squark pair production), or `all` (default).

5.3 Python interface

The primary envisaged use of xsec is as a component in a parameter scan code, where the user chooses the desired particle production processes and varies the input parameters. Appendix A contains a simple example script in Python where these parameters are set by hand. Examples of SLHA file input and usage within a loop over sparticle masses are available in the GitHub repository.

To ensure that any time-consuming computations are only run when strictly necessary, the prediction process is split into four steps:

1. Initialisation
   During the installation (see Sect. 5.2), GP model files for all included processes are downloaded to a local directory *gp_dir* of choice. At the start of a Python script, xsec must first be pointed to that directory:

   ```
   import xsec
   xsec.init(data_dir="gp_dir")
   ```

2. Process selection
   Subsequently, the centre-of-mass energy must be set (in GeV) through

   ```
   xsec.set_energy(13000)
   ```

   Currently, only the 13 TeV dataset is available. We expect to add further energies in the near future. Then, one or more sparticle production processes can be specified. A single process is identified by a tuple

   ```
   process = (pid1, pid2)
   ```

   containing the PDG codes of its final-state particles. A list of such tuples is required as the argument to the loading function:

   ```
   xsec.load_processes([process1, process2,
       ...])
   ```

   This function call decompresses the GP model files (serialised into Python pickles to save space) for the specified processes, and loads them into memory. This step only needs to happen once per process. It may take some time, as it involves a matrix multiplication to reconstruct the inverse covariance matrix of the training points from its Cholesky decomposition.

   To show a list of all available trained processes, one can use

   ```
   xsec.list_all_xsec_processes()
   ```

3. Parameter input
   The next step is to set the values of all relevant parameters for the selected processes. The relevant parameters are shown in Table 1. There are three ways to enter the parameters: manually setting their values one by one, setting multiple values at once with a dictionary, or reading all parameters from an SLHA file describing the supersymmetric spectrum.

   ```
   xsec.set_parameter("name", value)
   xsec.set_parameters(
       {"name1": value1, "name2": value2,
           ...}
     )
   xsec.import_slha("filepath")
   ```

   The example script in Appendix A lists all available parameters. The SLHA interface is based on the PySLHA package [54], which is a dependency of xsec. The program currently works with the SLHA1 standard, because we give the decomposition of cross-sections with squarks in terms of their flavour eigenstates (with the exception of the third generation[8]).
   Checking and clearing the current value of one or more parameters is possible with

   ```
   xsec.get_parameter("name")
   xsec.get_parameters(["name1", "name2",
       ...])
   xsec.clear_parameter("name")
   xsec.clear_parameters(["name1", "name2",
       ...])
   xsec.clear_parameters()
   ```

   The last line resets all parameter values.
4. Cross-section prediction
   At this point, the cross-section can be evaluated by simply calling

   ```
   xsec.eval_xsection()
   ```

   This function has two optional keywords: `verbose` (default `2`) and `check_consistency` (default `True`). Setting `verbose=0` makes sure nothing is printed to the

[8] To avoid confusing conventions regarding the third-generation mixing angles $\theta_{\tilde{t}}$ and $\theta_{\tilde{b}}$, we use the (1, 1) components of the relevant mixing matrices (`STOPMIX` and `SBOTMIX` in SLHA files) as parameters.

screen; `verbose=1` prints a single line per process that lists, in order: the PDG codes of the two final-state particles, the central cross-section value, and the regression, scale, PDF and α_s error bands (two values each); `verbose=2` prints a full description of the results. The consistency check will stop the evaluation if the features are outside the range of validity, discussed in Sect. 4, or have not been set.

The returned errors are always relative to the central cross-section, and a minus sign in the print-out makes it easy to distinguish the lower from the upper error bounds. Note that currently in xsec the PDF and α_s errors are symmetric by definition, following the PDF4LHC recommendations [49], whereas the regression and scale errors are not. In order to return a symmetric α_s error, we average the outputs from our two GPs trained separately on the upper and lower errors. This can be changed in the code by more advanced users if desired (for example, to allow α_s to be treated more accurately as a nuisance parameter to be scanned over). The regression error is asymmetric because it comes from a log-normal distribution, while the lower (upper) scale error derives from the minimum (maximum) cross-section value obtained by doubling and halving the renormalisation/factorisation scale.

The results from an evaluation for a given parameter point can be added to an existing SLHA file in the `XSECTION` block[9]:

```
results = xsec.eval_xsection()
xsec.write_slha("slha_path", results)
```

The `XSECTION` structure does not allow for storing the regression error, so this is omitted. Furthermore, we do not predict individual cross-section values for all the different members of the PDF set used. Thus, in order to provide the PDF error, we follow the PDF4LHC guidelines [49] and give the lower (upper) bound of the 68% confidence interval by incrementing the central PDF set index by 1 (2) in the `XSECTION` block.

Finally, it is recommended to run

```
xsec.finalise()
```

after all evaluations have been completed. This creates a BibTeX file in the current working directory, listing references to all original work that has been used to provide the requested results.

For advanced users, we include an option to write the decompressed GP files to disk and memory-map them for quicker access. This can reduce the memory load when many processes are requested simultaneously. The cache option can be activated by specifying

```
xsec.init(data_dir="gp_dir", use_cache=
    True, cache_dir="cache_dir")
```

in the initialisation step. The keyword specifying the cache directory is optional; by default a temporary directory with a random name will be created. The cache directory is removed when `finalise()` is called.

Another option to decrease the memory load when using xsec for many final states is to load the data for only a few processes at a time, make predictions, and clean the memory before loading new processes. The latter can be done by

```
xsec.unload_processes([process1, process2,
...])
```

If called without arguments, the data for all previously loaded processes will be cleared from memory.

5.4 Command-line interface

We also provide a command-line interface, where the user can supply a set of two final-state PDG codes and the values for the relevant features (parameters and averaged first and second-generation squark mass) involved in the corresponding cross-section. This is invoked as

```
xsec 13000 1000021 1000001 -p 1000
    500 500  -g gp_dir
```

This example calculates the cross-section for $\tilde{g}\tilde{u}_L$ production at $\sqrt{s} = 13$ TeV, using the GP data directory `gp_dir` (see Sect. 5.2). The features in the call and their order (in this case $m_{\tilde{g}}$, $m_{\tilde{q}_i}$ and $\bar{m}_{\tilde{q}}$, all in GeV) can be gleaned from

```
xsec 13000 1000021 1000001 --show-
    input
```

Alternatively, the parameters can be read from an SLHA file located at `slha_path`:

```
xsec 13000 1000021 1000001 -r
    slha_path -g gp_dir
```

Executing `xsec -help` in a terminal provides more details on all possible inputs. We do not recommend using the command-line interface in time-sensitive applications requiring multiple cross-section evaluations for the same process. Every evaluation from the command line invokes the (inevitably slow) `load_processes()` step, while a lot of time can be saved by writing a Python script where this function is called only once.

5.5 Code structure

For the interested user we briefly describe the structure of the code and the content of the various modules. Our design objective was to keep the user interface intuitive and simple

[9] https://phystev.cnrs.fr/wiki/2013:groups:tools:slha.

enough to be readily integrated within a larger code, even in another programming language via binding libraries.

The `parameters` module manages a global dictionary of parameter values, and has functions that allow for setting, getting and clearing those values. Furthermore, there is a function that checks the internal consistency of the entered values, ensuring that the mean squark mass and the centre-of-mass energy are set correctly. SLHA interface functions are also collected in this module, enabling one to read parameter values from an SLHA1 file and to write the cross-section results to an `XSECTION` block in that file.

The `features` module keeps track of the set of parameter values relevant to each specific production process, using a global dictionary and functions that access it.

The `gploader` module contains functions and global variables related to initialisation settings and to loading pre-trained GPs into memory from pickled files. When using the cache option described in Sect. 5.3, this module is responsible for managing the cache directory on the disk.

The implementation of the GP kernels resides within the `kernels` module. These are mostly reproduced from the scikit-learn v0.19.2 [55] source code under the New BSD License, apart from some new kernels and a function that reconstructs kernel function objects from the kernel parameters stored in the pickled GP files.

All functions responsible for GP predictions and combining results from multiple GPs using the GRBCM prescription are collected in the `evaluation` module. Each production process and cross-section type has its own data directory, containing a `transform` module used by the `evaluation` module, and relevant datafiles. The `transform` module reverses the specific transformations applied to the target values during training.

Internal helper functions are defined in the `utils` module. These are mostly related to the internal naming system for Gaussian process model files, but also include utility functions for outputting results to screen and file, as well as the collection of BibTeX references relevant to the requested processes.

6 Conclusions

In this paper, we have presented a new phenomenology code xsec, which allows for the fast evaluation of higher-order cross-sections through regression on pre-generated training datasets. The regression in xsec is based on Generalised Robust Bayesian Committee Machines, a method that employs distributed Gaussian processes. In addition to the central value and the related regression variance provided by the Gaussian processes, we have trained separate Gaussian processes on the scale, PDF and α_s errors, providing a complete ecosystem for the evaluation of cross-sections and their uncertainties.

The current version of xsec 1.0 includes all MSSM strong-production cross-sections at $\sqrt{s} = 13\,\text{TeV}$ at NLO precision in QCD, separated into individual squark flavour final states, and allows for non-degenerate squark masses. We plan to make future updates of xsec that will extend the code both in terms of the included energies and processes, as well as with higher-order corrections. The method used here can also be extended to other models, so long as appropriate training sets can be generated. The production of supersymmetric particles in the MSSM serves as a good starting point however, as the absence of s-channel resonances simplifies the training.

We would like to emphasise that the xsec code described here is not a new calculation of the included cross-sections. It is a regression estimate based on a pre-generated sample of cross-sections taken from existing results. Thus, users of this code should also reference the original physics results on which the results of xsec are based when using them in publications. We provide functionality within the code for easily identifying the relevant references.

Acknowledgements This work has been supported by the Research Council of Norway (FRIPRO 230546/F20) and a NOTUR (Norway; NN9284K) Grant for computation time on the Abel cluster at the University of Oslo. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Action Innovative Training Networks MCnetITN3 and SAGEX (Grant agreement nos. 722104 and 764850), the Royal Society under Grant UF160548, and the Australian Research Council under grant FT190100814. AB wishes to thank Donatas Zaripovas and Laurynas Mince for their contributions to earlier versions of this work, and AK and PS wish to thank Ben Farmer, Nicholas Reed and Iza Veliščeck for many useful discussions on Gaussian Processes.

Data Availability Statement This manuscript has no associated data or the data will not be deposited. [Authors' comment: As detailed in the paper, the source code for the presented software tool is available in a public repository. The data it requires is an integral part of the software release, and the software contains a dedicated tool for downloading this data as described in the instructions for use.]

Funded by SCOAP[3].

Appendix A: Code example

Below we show a minimal Python script running the xsec program both by specifying masses and other input parameters by hand, and by loading values from an SLHA input file. The script also demonstrates how to store the results in an SLHA file in terms of XSECTION blocks.

```
import xsec

# Set data directory
xsec.init(data_dir='gprocs')

# Set centre-of-mass energy (in GeV)
xsec.set_energy(13000)

# Load GP models for specified process(es)
xsec.load_processes([(1000021, 1000021)])

# Evaluate cross-section with given inputs
xsec.set_parameters({
    'm1000021': 1000.0,
    'm1000001': 500.0,
    'm1000002': 500.0,
    'm1000003': 500.0,
    'm1000004': 500.0,
    'm1000005': 500.0,
    'm1000006': 500.0,
    'm2000001': 500.0,
    'm2000002': 500.0,
    'm2000003': 500.0,
    'm2000004': 500.0,
    'm2000005': 500.0,
    'm2000006': 500.0,
    'sbotmix11': 0.0,
    'stopmix11': 0.0,
    'mean': 500.0,
})
xsec.eval_xsection()

# Evaluate cross-section with SLHA input
xsec.import_slha('sps1a.slha')
results = xsec.eval_xsection()

# Write results to XSECTION block in SLHA
    file
xsec.write_slha('sps1a.slha', results)

# Create references file
xsec.finalise()
```

References

1. GAMBIT collaboration, C. Balázs et al., ColliderBit: a GAMBIT module for the calculation of high-energy collider observables and likelihoods. Eur. Phys. J. C **77**, 795 (2017). arXiv:1705.07919
2. ATLAS collaboration, G. Aad et al., Search for squarks and gluinos with the ATLAS detector in final states with jets and missing transverse momentum using $\sqrt{s} = 8$ TeV proton–proton collision data. JHEP **09**, 176 (2014). arXiv:1405.7875
3. GAMBIT collaboration, P. Athron et al., GAMBIT: the global and modular beyond-the-standard-model inference tool. Eur. Phys. J. C **77**, 784 (2017). arXiv:1705.07908
4. C.E. Rasmussen, C.K.I. Williams, *Gaussian Processes for Machine Learning* (The MIT Press, Cambridge, 2006)
5. M.P. Deisenroth, J.W. Ng, Distributed Gaussian Processes. arXiv:1502.02843
6. H. Liu, J. Cai, Y. Wang, Y.-S. Ong, Generalized robust Bayesian Committee machine for large-scale Gaussian process regression, in *Proceedings of ICML* (2018). arXiv:1806.00720
7. W. Beenakker, R. Hopker, M. Spira, P.M. Zerwas, Squark and gluino production at hadron colliders. Nucl. Phys. B **492**, 51–103 (1997). arXiv:hep-ph/9610490
8. W. Beenakker, R. Hopker, M. Spira, PROSPINO: A Program for the production of supersymmetric particles in next-to-leading order QCD. arXiv:hep-ph/9611232
9. W. Beenakker, M. Krämer, T. Plehn, M. Spira, P.M. Zerwas, Stop production at hadron colliders. Nucl. Phys. B **515**, 3–14 (1998). arXiv:hep-ph/9710451
10. W. Beenakker, M. Klasen, M. Krämer, T. Plehn, M. Spira, P.M. Zerwas, The production of charginos/neutralinos and sleptons at hadron colliders. Phys. Rev. Lett. **83**, 3780–3783 (1999). arXiv:hep-ph/9906298
11. M. Spira, Higgs and SUSY particle production at hadron colliders, in *Supersymmetry and unification of fundamental interactions. Proceedings, 10th International Conference, SUSY'02, Hamburg, Germany, June 17-23, 2002*, pp. 217–226, 2002. arXiv:hep-ph/0211145
12. T. Plehn, Measuring the MSSM Lagrangean. Czech. J. Phys. **55**, B213–B220 (2005). arXiv:hep-ph/0410063
13. S. Frixione, B. Fuks, V. Hirschi, K. Mawatari, H.-S. Shao, P.A. Sunder et al., Automated simulations beyond the Standard Model: supersymmetry. JHEP **12**, 008 (2019). arXiv:1907.04898
14. A. Kulesza, L. Motyka, Threshold resummation for squark–antisquark and gluino-pair production at the LHC. Phys. Rev. Lett. **102**, 111802 (2009). arXiv:0807.2405
15. A. Kulesza, L. Motyka, Soft gluon resummation for the production of gluino–gluino and squark–antisquark pairs at the LHC. Phys. Rev. D **80**, 095004 (2009). arXiv:0905.4749
16. W. Beenakker, S. Brensing, M. Krämer, A. Kulesza, E. Laenen, I. Niessen, Soft-gluon resummation for squark and gluino hadroproduction. JHEP **12**, 041 (2009). arXiv:0909.4418
17. W. Beenakker, S. Brensing, M. Krämer, A. Kulesza, E. Laenen, I. Niessen, Supersymmetric top and bottom squark production at hadron colliders. JHEP **08**, 098 (2010). arXiv:1006.4771
18. W. Beenakker, S. Brensing, M. Krämer, A. Kulesza, E. Laenen, L. Motyka et al., Squark and gluino hadroproduction. Int. J. Mod. Phys. A **26**, 2637–2664 (2011). arXiv:1105.1110
19. W. Beenakker, S. Brensing, M. Krämer, A. Kulesza, E. Laenen, I. Niessen, NNLL resummation for squark–antisquark pair production at the LHC. JHEP **01**, 076 (2012). arXiv:1110.2446
20. W. Beenakker, T. Janssen, S. Lepoeter, M. Krämer, A. Kulesza, E. Laenen et al., Towards NNLL resummation: hard matching coefficients for squark and gluino hadroproduction. JHEP **10**, 120 (2013). arXiv:1304.6354
21. W. Beenakker, C. Borschensky, M. Krämer, A. Kulesza, E. Laenen, V. Theeuwes et al., NNLL resummation for squark and gluino production at the LHC. JHEP **12**, 023 (2014). arXiv:1404.3134
22. W. Beenakker, C. Borschensky, M. Krämer, A. Kulesza, E. Laenen, S. Marzani et al., NLO + NLL squark and gluino production cross-sections with threshold-improved parton distributions. Eur. Phys. J. C **76**, 53 (2016). arXiv:1510.00375
23. W. Beenakker, C. Borschensky, R. Heger, M. Krämer, A. Kulesza, E. Laenen, NNLL resummation for stop pair-production at the LHC. JHEP **05**, 153 (2016). arXiv:1601.02954

24. W. Beenakker, C. Borschensky, M. Krämer, A. Kulesza, E. Laenen, NNLL-fast: predictions for coloured supersymmetric particle production at the LHC with threshold and Coulomb resummation. JHEP **12**, 133 (2016). arXiv:1607.07741
25. G. Bozzi, B. Fuks, M. Klasen, Transverse-momentum resummation for slepton-pair production at the CERN LHC. Phys. Rev. D **74**, 015001 (2006). arXiv:hep-ph/0603074
26. G. Bozzi, B. Fuks, M. Klasen, Threshold resummation for slepton-pair production at hadron colliders. Nucl. Phys. B **777**, 157–181 (2007). arXiv:hep-ph/0701202
27. G. Bozzi, B. Fuks, M. Klasen, Joint resummation for slepton pair production at hadron colliders. Nucl. Phys. B **794**, 46–60 (2008). arXiv:0709.3057
28. J. Debove, B. Fuks, M. Klasen, Transverse-momentum resummation for gaugino-pair production at hadron colliders. Phys. Lett. B **688**, 208–211 (2010). arXiv:0907.1105
29. J. Debove, B. Fuks, M. Klasen, Threshold resummation for gaugino pair production at hadron colliders. Nucl. Phys. B **842**, 51–85 (2011). arXiv:1005.2909
30. J. Debove, B. Fuks, M. Klasen, Joint resummation for gaugino pair production at hadron colliders. Nucl. Phys. B **849**, 64–79 (2011). arXiv:1102.4422
31. B. Fuks, M. Klasen, D.R. Lamprea, M. Rothering, Gaugino production in proton–proton collisions at a center-of-mass energy of 8 TeV. JHEP **10**, 081 (2012). arXiv:1207.2159
32. B. Fuks, M. Klasen, D.R. Lamprea, M. Rothering, Precision predictions for electroweak superpartner production at hadron colliders with Resummino. Eur. Phys. J. C **73**, 2480 (2013). arXiv:1304.0790
33. B. Fuks, M. Klasen, M. Rothering, Soft gluon resummation for associated gluino-gaugino production at the LHC. JHEP **07**, 053 (2016). arXiv:1604.01023
34. S. Otten, K. Rolbiecki, S. Caron, J.-S. Kim, R. Ruiz De Austri, J. Tattersall, DeepXS: fast approximation of MSSM electroweak cross sections at NLO. Eur. Phys. J. C **80**, 12 (2020). arXiv:1810.08312
35. T.E. Oliphant, *A guide to NumPy*, vol. 1 (Trelgol Publishing, New York, 2006)
36. P.Z. Skands et al., SUSY Les Houches accord: interfacing SUSY spectrum calculators, decay packages, and event generators. JHEP **07**, 036 (2004). arXiv:hep-ph/0311123
37. D. Duvenaud, *Automatic model construction with Gaussian processes*. Ph.D. thesis, University of Cambridge (2014)
38. C.A. Micchelli, Y. Xu, H. Zhang, Universal kernels. J. Mach. Learn. Res. **7**, 2651–2667 (2006)
39. I.A.V. Holm, *Gaussian processes for cross section evaluation*, Master's thesis, Department of Physics, University of Oslo (2018)
40. I. Vellšček, *Machine learning for fast simulations of physics beyond the standard model*, Master's thesis, Department of Physics, Imperial College London (2019)
41. N. Reed, *Machine learning for fast collider simulations*, Master's thesis, Department of Physics, Imperial College London (2019)
42. R. Storn, K. Price, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997)
43. P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau et al., Scipy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods **17**, 261–272 (2020). arXiv:1907.10121
44. J. Wågberg, D. Zachariah, T. Schön, P. Stoica, Prediction performance after learning in Gaussian process regression, in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research* vol. 54 ed. by A. Singh, J. Zhu (Fort Lauderdale, FL, USA), pp. 1264–1272, PMLR, 20–22 Apr (2017)
45. M.P. Deisenroth, Y. Luo, M.V.D. Wilk, A Practical Guide to Gaussian Processes. https://drafts.distill.pub/gp/
46. H. Mohammadi, R.L. Riche, N. Durrande, E. Touboul, X. Bay, An analytic comparison of regularization methods for Gaussian processes. arXiv:1602.00853
47. D. Rullière, N. Durrande, F. Bachoc, C. Chevalier, Nested Kriging predictions for datasets with large number of observations. arXiv:1607.05432
48. A. Buckley, J. Ferrando, S. Lloyd, K. Nordström, B. Page, M. Rüfenacht et al., LHAPDF6: parton density access in the LHC precision era. Eur. Phys. J. C **75**, 132 (2015). arXiv:1412.7420
49. J. Butterworth et al., PDF4LHC recommendations for LHC Run II. J. Phys. G **43**, 023001 (2016). arXiv:1510.03865
50. G.P. Lepage, A new algorithm for adaptive multidimensional integration. J. Comput. Phys. **27**(5), 192–203 (1978)
51. D. Gonçalves-Netto, D. López-Val, K. Mawatari, T. Plehn, I. Wigmore, Automated squark and gluino production to next-to-leading order. Phys. Rev. D **87**, 014002 (2013). arXiv:1211.0286
52. B.C. Allanach, SOFTSUSY: a program for calculating supersymmetric spectra. Comput. Phys. Commun. **143**, 305–331 (2002). arXiv:hep-ph/0104145
53. B.C. Allanach, S.P. Martin, D.G. Robertson, R.R. de Austri, The inclusion of two-loop SUSYQCD corrections to gluino and squark pole masses in the minimal and next-to-minimal supersymmetric standard model: SOFTSUSY3.7. arXiv:1601.06657
54. A. Buckley, PySLHA: a Pythonic interface to SUSY Les Houches Accord data. Eur. Phys. J. C **75**, 467 (2015). arXiv:1305.4194
55. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel et al., Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)