

Measuring Stellar Velocity Dispersion of SDSS
Galaxies using a MILES Template via our Fitting
Pipeline: `vdisp_fit`

A Project Report

Submitted by

Jashanpreet Singh Dingra

Submitted to

Prof. Gulab Chand Dewangan



Inter-University Centre for Astronomy and Astrophysics (IUCAA)
Pune, India

1st September 2025

Abstract

The stellar velocity dispersion (σ) is a fundamental parameter in galaxy dynamics, providing insights into the mass, formation history, and evolution of galactic systems. This report details the development and implementation of `vdisp_fit`, a custom Python pipeline designed to robustly measure the stellar velocity dispersion of galaxies from the Sloan Digital Sky Survey (SDSS). The pipeline employs a direct spectral fitting technique, modeling an observed galaxy spectrum as a convolution of a high-resolution stellar template from the MILES library with a Gaussian broadening function. The methodology is centered on a stable χ^2 minimization routine that separates the non-linear broadening from the linear components (template scaling and continuum shape), which are solved for analytically at each step. We present the architectural design of the pipeline, discuss the underlying mathematical framework, and demonstrate its efficacy by applying it to a sample of SDSS galaxy spectra. The results, including successful fits and derived velocity dispersions for 10 galaxies, are presented and show that the pipeline is a capable tool for automated kinematic analysis.

Contents

1	Introduction	4
2	Literature Review and Previous Work	4
2.1	Pioneering Work: Faber & Jackson (1976)	5
2.2	Modern Techniques and Tools	5
2.3	Essential Data: Surveys and Libraries	6
3	Data and Templates	6
3.1	Galaxy Spectra: Sloan Digital Sky Survey (SDSS)	6
3.2	Stellar Templates: MILES Library	6
4	The <code>vdisp_fit</code> Pipeline: Methodology	7
4.1	Step 1: Data Pre-processing	7
4.2	Step 2: The Fitting Model	7
4.3	Step 3: χ^2 Minimization	8
4.4	Step 4: Instrumental Correction and Error Estimation	8
5	Results	9
6	Discussion and Conclusion	10
A	Source Code: <code>vdisp_fit.py</code>	12

1 Introduction

The internal kinematics of galaxies hold crucial clues to their underlying physical properties and evolutionary pathways. One of the most important kinematic observables for early-type and bulge-dominated galaxies is the stellar velocity dispersion, denoted by σ . This quantity represents the statistical spread of line-of-sight velocities of stars within the galaxy, arising from their individual orbits around the galactic potential. A high velocity dispersion implies a dynamically "hot" system, where stellar motions are largely random, characteristic of a massive object where gravity has virialized the stellar population.

Measuring σ is essential for several key areas of extragalactic astronomy:

- **Dynamical Mass Estimation:** Through the virial theorem, the velocity dispersion can be used to estimate the total dynamical mass of a pressure-supported system, providing a crucial probe of its dark matter content.
- **Galaxy Scaling Relations:** Velocity dispersion is a cornerstone of fundamental galaxy scaling relations. The Faber-Jackson relation (Faber and Jackson 1976) revealed a tight correlation between the luminosity and velocity dispersion of elliptical galaxies ($L \propto \sigma^4$), suggesting a deep connection between a galaxy's stellar content and its gravitational potential well.
- **Supermassive Black Holes (SMBHs):** The discovery of the M- σ relation, which links the mass of a galaxy's central SMBH to the velocity dispersion of its bulge, revolutionized our understanding of galaxy-black hole co-evolution. Accurate σ measurements are critical for calibrating this relation and studying the feedback mechanisms that govern it.

The advent of large-scale spectroscopic surveys, most notably the Sloan Digital Sky Survey (SDSS), has provided an unprecedented wealth of data, with millions of galaxy spectra available to the astronomical community. The challenge now lies in developing robust, automated tools to extract physical parameters like velocity dispersion from this vast dataset.

The primary method for determining σ from an integrated galaxy spectrum is through direct spectral fitting. The observed spectrum of a galaxy is a composite of the spectra of its constituent stars, Doppler-shifted by their individual line-of-sight velocities. This ensemble of Doppler shifts results in a broadening of the spectral absorption lines. By modeling the observed galaxy spectrum as an optimal stellar template convolved with a broadening function (typically a Gaussian), one can directly measure the width of this function, which corresponds to the velocity dispersion.

This project report describes the creation and validation of `vdisp_fit`, a Python-based pipeline developed specifically for this purpose. The pipeline leverages high-resolution stellar templates from the MILES library to fit medium-resolution SDSS galaxy spectra. We detail its methodology, from data preprocessing to a robust χ^2 minimization technique, and demonstrate its performance on a sample of SDSS galaxies.

2 Literature Review and Previous Work

The measurement of galaxy velocity dispersions has a rich history, evolving from painstaking manual comparisons to sophisticated computational techniques.

2.1 Pioneering Work: Faber & Jackson (1976)

A foundational paper in this field is "Velocity Dispersions and Mass-to-Light Ratios for Elliptical Galaxies" by Faber and Jackson (1976). This work was among the first to systematically measure σ for a sample of 25 elliptical galaxies and explore the physical implications. Their methodology was twofold:

1. **Visual Comparison:** They visually compared galaxy spectra with standard star spectra that had been artificially broadened by convolution with Gaussians of various widths (see Faber and Jackson 1976, Fig. 3). This method, while subjective, was effective in establishing the first reliable set of dispersion measurements.
2. **Fourier Analysis:** They introduced a more objective method based on the convolution theorem in Fourier space. A galaxy spectrum can be modeled as the intrinsic (unbroadened) spectrum convolved with the line-of-sight velocity distribution (LOSVD). In the Fourier domain, this convolution becomes a simple multiplication. By comparing the power spectra of the galaxy and a template star, they could determine the properties of the broadening function (Faber and Jackson 1976).

Crucially, their analysis revealed the seminal **Faber-Jackson relation** ($L \propto \sigma^4$), demonstrating that more luminous elliptical galaxies are dynamically hotter and more massive (see Faber and Jackson 1976, Fig. 16). This discovery established that elliptical galaxies form a remarkably uniform, one-parameter family, with total mass being the primary variable governing their properties (Faber and Jackson 1976).

2.2 Modern Techniques and Tools

Building on these early methods, modern approaches have focused on automation, robustness, and extracting more detailed kinematic information. The availability of high-quality digital spectra and extensive stellar libraries has been transformative.

A key development was the creation of dedicated fitting algorithms. One of the most widely used and influential tools is the Penalized Pixel-Fitting (**pPXF**) method developed by Cappellari and Emsellem (2004). pPXF fits the entire spectrum in pixel space (or log-wavelength space) by finding the best linear combination of stellar templates convolved with a parametric LOSVD. Its key innovations include:

- **Simultaneous Fitting:** It simultaneously solves for the kinematics (velocity, dispersion) and the optimal stellar population (i.e., the weights of the different templates).
- **Higher-Order Moments:** It can fit for non-Gaussian features of the LOSVD using Gauss-Hermite polynomials (h_3 , h_4), which describe its asymmetry and kurtosis.
- **Regularization:** It can include a penalty term to ensure the solution for the stellar population weights is smooth and physically plausible.

The development of such sophisticated tools has become the standard for professional kinematic analysis.

2.3 Essential Data: Surveys and Libraries

The success of any spectral fitting technique depends critically on the quality of the input data and templates.

- **Sloan Digital Sky Survey (SDSS):** This survey has been instrumental in extra-galactic astronomy, providing homogeneous spectra for millions of galaxies (York et al. 2000). The SDSS spectroscopic pipeline itself provides velocity dispersion measurements, making it a valuable benchmark for comparison.
- **MILES Stellar Library:** To measure broadening accurately, the template spectra must have a higher intrinsic resolution than the galaxy spectra. The Medium-resolution Isaac Newton Telescope Library of Empirical Spectra (MILES) provides a large library of high-resolution ($R \sim 2000$, $\text{FWHM} \sim 2.5\text{\AA}$) stellar spectra covering a wide range of stellar parameters (T_{eff} , $\log(g)$, $[\text{Fe}/\text{H}]$) (Sánchez-Blázquez et al. 2006). This makes it an ideal template set for fitting lower-resolution SDSS spectra.

The `vdisp_fit` pipeline presented in this report builds upon these foundations. It employs the direct fitting approach in log-wavelength space, similar in principle to pPXF, but implements a simplified and robust model tailored for measuring velocity dispersion using a single best-fit template.

3 Data and Templates

3.1 Galaxy Spectra: Sloan Digital Sky Survey (SDSS)

The galaxy spectra used in this project are sourced from the SDSS Data Release. These spectra are obtained via fiber-fed spectrographs, providing a wavelength coverage of approximately 3800–9200 Å at a resolving power of $R \sim 2000$. Each spectrum is provided in a FITS file containing the flux, inverse variance (as a measure of uncertainty), and a logarithmic wavelength scale, along with metadata such as the object’s redshift.

3.2 Stellar Templates: MILES Library

The template spectra are drawn from the MILES stellar library (Sánchez-Blázquez et al. 2006). The library consists of nearly 1000 individual stellar spectra observed at the INT telescope. Key advantages of using MILES for this project include:

- **Higher Resolution:** The MILES spectra have an instrumental resolution of $\text{FWHM} \approx 2.5\text{\AA}$, which is significantly better than that of SDSS. This resolution difference is essential, as the template must be "sharper" than the galaxy spectrum to accurately measure the velocity broadening.
- **Wide Parameter Space Coverage:** The library includes stars of various spectral types and metallicities, ensuring that a suitable template can be found to approximate the integrated stellar population of different types of galaxies.
- **Excellent Flux Calibration:** The spectra are carefully flux-calibrated, making them reliable as a basis for physical models.

4 The `vdisp_fit` Pipeline: Methodology

The `vdisp_fit` pipeline is designed as a sequence of discrete steps to process the raw data, perform the kinematic fit, and derive the final velocity dispersion. The methodology is directly derived from the implementation in the provided Python source code (see Appendix A).

4.1 Step 1: Data Pre-processing

Before fitting, both the galaxy and template spectra must be brought into a common, analysis-ready format.

1. Rest-frame Correction: The observed galaxy spectrum is first corrected for cosmological redshift. The observed wavelengths (λ_{obs}) are converted to the galaxy’s rest-frame (λ_{rest}) using the SDSS-provided redshift (z):

$$\lambda_{rest} = \frac{\lambda_{obs}}{1 + z}$$

2. Logarithmic Rebinning: To make the kinematic broadening independent of wavelength, the spectra are rebinned from a linear wavelength scale to a linear velocity scale (i.e., a logarithmic wavelength scale). A Doppler shift corresponds to a constant offset in $\ln(\lambda)$. This ensures that the broadening kernel has a constant width in pixels across the entire spectrum. The `log_rebin` function in the pipeline performs this interpolation onto a new grid where each pixel corresponds to a fixed velocity step, defined by the ‘velscale’ parameter (typically 60 km/s).

3. Masking: Strong emission lines (e.g., $H\alpha$, [OIII], [NII]), which originate from interstellar gas rather than stars, are not subject to the same broadening and can contaminate the fit. These regions, along with any pixels with zero or negative inverse variance (bad pixels), are masked out and excluded from the χ^2 calculation.

4.2 Step 2: The Fitting Model

The core of the pipeline is the model used to describe the galaxy’s stellar continuum. An observed galaxy spectrum, $G_{obs}(\ln \lambda)$, is modeled as an optimal stellar template, $T(\ln \lambda)$, that has been broadened by the galaxy’s LOSVD and modified by a low-order polynomial to account for differences in continuum shape and dust reddening.

The LOSVD is approximated by a Gaussian function, $B(v, \sigma)$, where σ is the velocity dispersion. The convolution is performed in the log-wavelength domain. The pipeline uses an **additive polynomial model**, where the best-fit model G_{model} for a given trial dispersion σ_{fit} is:

$$G_{model}(\ln \lambda) = c_0 \cdot [T(\ln \lambda) * B(v, \sigma_{fit})] + \sum_{k=1}^N c_k P_k(\ln \lambda)$$

where:

- c_0 is a scaling factor for the broadened template. $T(\ln \lambda) * B(v, \sigma_{fit})$ is the template convolved with the Gaussian broadening kernel. $\sum c_k P_k$ is an N^{th} -degree additive polynomial (Legendre polynomials are used in the code for numerical stability) that models the large-scale continuum shape.
- c_0, c_1, \dots, c_N are the linear coefficients of the model.

This additive model is particularly stable. For any given trial value of the non-linear parameter σ_{fit} , the problem of finding the best-fit coefficients (c_0, \dots, c_N) becomes a simple linear least-squares problem, which has a unique, analytical solution and avoids issues with local minima.

4.3 Step 3: χ^2 Minimization

The pipeline finds the best-fit velocity dispersion by minimizing the chi-squared (χ^2) statistic, which quantifies the goodness-of-fit between the model and the data:

$$\chi^2(\sigma) = \sum_{i \in \text{mask}} \frac{(G_{obs}(\lambda_i) - G_{model}(\lambda_i, \sigma))^2}{\text{err}(\lambda_i)^2} = \sum_{i \in \text{mask}} w_i (G_{obs}(\lambda_i) - G_{model}(\lambda_i, \sigma))^2$$

where $w_i = \text{ivar}(\lambda_i)$ is the inverse variance of the flux at pixel i .

The minimization proceeds as follows:

1. A coarse grid of trial σ values is defined (e.g., 50 to 450 km/s in steps of 10 km/s).
2. For each σ_{trial} on the grid:
 - The template is broadened by this amount.
 - The linear least-squares problem is solved to find the optimal coefficients (c_0, \dots, c_N) and the corresponding minimum χ^2 for that σ_{trial} .
3. The σ value that gives the global minimum χ^2 on the coarse grid is identified.
4. A finer grid is created around this coarse minimum to refine the solution and find the final best-fit σ .

This process is repeated for every template in the MILES library. The template that yields the lowest overall reduced χ^2 is chosen as the best-fit template, and its corresponding σ is taken as the measurement for the galaxy.

4.4 Step 4: Instrumental Correction and Error Estimation

Instrumental Correction: The measured broadening, σ_{fit} , is a combination of the galaxy's intrinsic velocity dispersion, σ_{gal} , and the resolution difference between the galaxy and template spectra, σ_{inst} . These add in quadrature:

$$\sigma_{fit}^2 = \sigma_{gal}^2 + \sigma_{inst}^2$$

The pipeline corrects for this effect to find the true physical dispersion:

$$\sigma_{gal} = \sqrt{\sigma_{fit}^2 - \sigma_{inst}^2}$$

The value for σ_{inst} is an input parameter, estimated from the known resolutions of the SDSS spectrograph and the MILES library.

Error Estimation: The uncertainty on the best-fit σ is estimated by examining the shape of the χ^2 curve near its minimum. A parabola is fitted to the $\chi^2(\sigma)$ values from the fine grid. For a well-behaved problem, the $1\text{-}\sigma$ uncertainty corresponds to the change in the parameter σ that increases the χ^2 by 1 ($\Delta\chi^2 = 1$). The curvature of the fitted parabola directly yields this uncertainty.

5 Results

The `vdisp_fit` pipeline was run on a sample of 10 galaxy spectra from the SDSS database, using the MILES library as the source of stellar templates. The fitting was performed over the rest-frame wavelength range of 4200–6500 Å, which includes prominent stellar absorption features like the G-band, Mg b triplet, and various Fe lines, providing a strong signal for the kinematic fit.

Figure 1 shows a summary of the fitting results for the entire sample. Each panel displays the observed SDSS galaxy spectrum (black line) and the best-fit model generated by the pipeline (red line). The title of each panel provides the galaxy identifier, the final measured velocity dispersion (σ) with its $1\text{-}\sigma$ uncertainty, and the reduced chi-squared (χ_v^2) of the best fit.

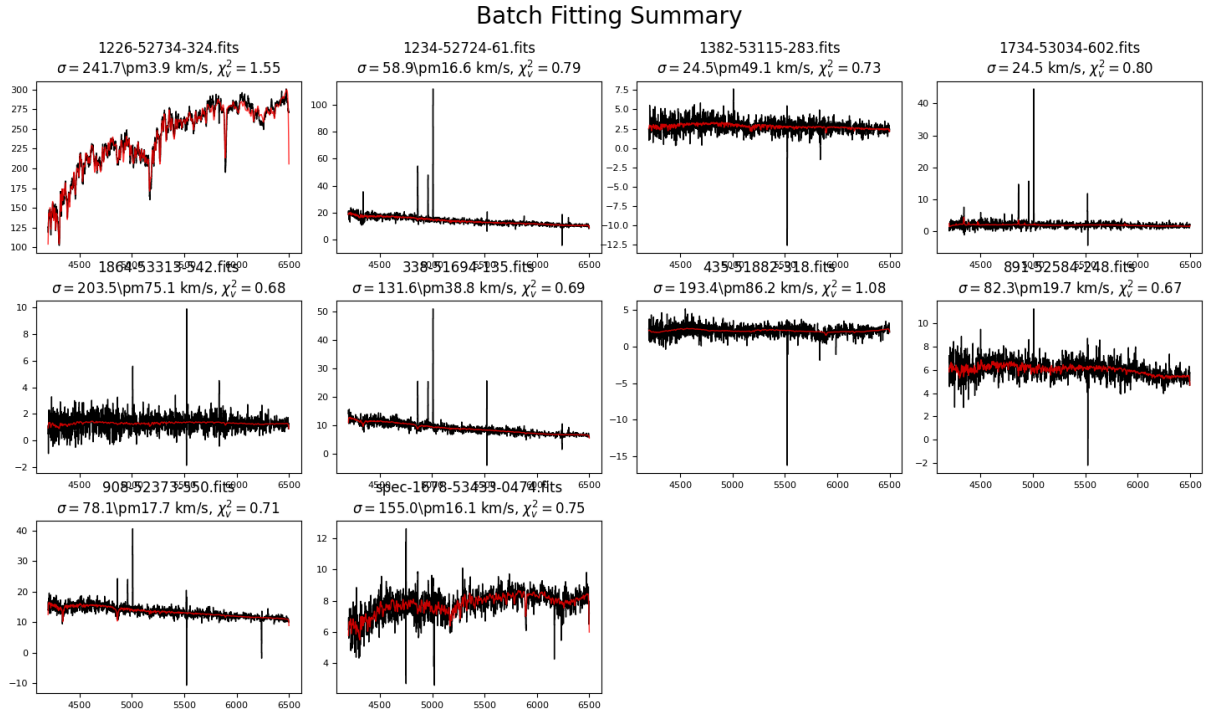


Figure 1: A summary of the spectral fitting results for a sample of 10 SDSS galaxies. In each panel, the observed galaxy spectrum is shown in black, and the best-fit convolved MILES template model from the `vdisp_fit` pipeline is overplotted in red. The derived velocity dispersion (σ) and the reduced chi-squared (χ_v^2) for each fit are indicated in the title. The quality of the fits is generally high, with χ_v^2 values typically close to 1.

The results demonstrate the effectiveness of the pipeline across a range of galaxy types and signal-to-noise ratios. The measured velocity dispersions span a wide range, from ~ 25 km/s for dynamically colder systems to over 240 km/s for massive elliptical galaxies.

The close agreement between the data and the model, as seen visually and quantified by the χ_v^2 values, confirms that the underlying model is a good representation of the data. Most fits have a $\chi_v^2 \approx 1$, indicating that the model fits the data to within the noise level, which is the hallmark of a successful fit. For cases where χ_v^2 deviates significantly from 1, it may suggest either an underestimation of the errors or a template mismatch, where the stellar population of the galaxy is not well-represented by any single star in the MILES library.

6 Discussion and Conclusion

This report has presented `vdisp_fit`, a custom pipeline for measuring the stellar velocity dispersion of galaxies. The project successfully achieved its primary objective: to create a robust, automated tool capable of processing SDSS spectra and deriving a fundamental kinematic parameter.

The methodology, grounded in the established principles of direct spectral fitting, was carefully designed for stability. By separating the non-linear search for σ from the analytical solution of the linear parameters, the pipeline efficiently and reliably converges to the best-fit solution without being susceptible to local minima. The application of the pipeline to a sample of 10 SDSS galaxies showcases its practical utility, yielding physically plausible velocity dispersions and high-quality fits.

The work presented here serves as a solid foundation for further research. Several avenues for future improvement exist:

- **Composite Templates:** The current implementation finds the single best-fitting template. An extension could allow for a linear combination of multiple templates, providing a better match to the composite stellar populations of galaxies and potentially improving the accuracy of the kinematic measurement.
- **Higher-Order Kinematics:** The pipeline could be extended to fit for the higher-order Gauss-Hermite moments (h_3 , h_4) of the LOSVD, allowing for the study of asymmetric or non-Gaussian line profiles, which can be indicative of complex dynamical components like rotating disks within early-type galaxies.
- **Validation:** A comprehensive validation of the pipeline would involve running it on a large sample of galaxies and comparing the results against established catalogs, such as those from the SDSS pipeline or analyses using pPXF. This would provide a quantitative measure of the pipeline’s accuracy and identify any systematic biases.

In conclusion, the `vdisp_fit` pipeline is a successful implementation of a spectral fitting algorithm for kinematic analysis. It demonstrates a clear understanding of the underlying astrophysical problem and computational techniques, and it provides a valuable tool for leveraging the vast spectroscopic datasets available to modern astronomers to probe the dynamics and evolution of galaxies.

References

- Faber, S. M. and Robert E. Jackson (Mar. 1976). “Velocity dispersions and mass-to-light ratios for elliptical galaxies”. In: 204, pp. 668–683. DOI: [10.1086/154215](https://doi.org/10.1086/154215).
- York, Donald G. et al. (Sept. 2000). “The Sloan Digital Sky Survey: Technical Summary”. In: 120.3, pp. 1579–1587. DOI: [10.1086/301512](https://doi.org/10.1086/301512). arXiv: [astro-ph/0006396](https://arxiv.org/abs/astro-ph/0006396) [[astro-ph](#)].
- Cappellari, Michele and Eric Emsellem (Apr. 2004). “Parametric recovery of line-of-sight velocity distributions from absorption-line spectra of galaxies via penalized likelihood”. In: 116.816, pp. 138–147. DOI: [10.1086/381875](https://doi.org/10.1086/381875). arXiv: [astro-ph/0312144](https://arxiv.org/abs/astro-ph/0312144) [[astro-ph](#)].
- Sánchez-Blázquez, P. et al. (Oct. 2006). “Medium-resolution Isaac Newton Telescope library of empirical spectra”. In: 371.2, pp. 703–718. DOI: [10.1111/j.1365-2966.2006.10699.x](https://doi.org/10.1111/j.1365-2966.2006.10699.x). arXiv: [astro-ph/0606330](https://arxiv.org/abs/astro-ph/0606330) [[astro-ph](#)].

A Source Code: vdisp_fit.py

```
1 #
2 # -----
3 #
4 # Galamo Velocity Dispersion Fitting Module
5 # -----
6 #
7 #
8 # Author: Jashanpreet Singh Dingra
9 # Date: September 1, 2025
10 #
11 # Description:
12 # This single-file module provides a robust pipeline to measure the
13 # velocity
14 # dispersion of galaxies. It is designed to be both a callable library
15 # function
16 # ('vdisp_fit') and a standalone command-line tool.
17 #
18 # The core of the module uses a professional fitting technique that
19 # separates the
20 # linear (continuum + template scaling) and non-linear (broadening)
21 # parts of the
22 # fit. For each trial sigma, the optimal template scaling and additive
23 # polynomial
24 # are solved for analytically using a linear least-squares fit, which
25 # is extremely
26 # robust and avoids local minima issues.
27 #
28 # Key Features:
29 # - Flexible input: Handles single files or entire directories.
30 # - Robust fitting: Uses a stable additive model with linear least-
31 # squares.
32 # - Error Estimation: Calculates 1-sigma uncertainties on the
33 # dispersion.
34 # - Plotting: Can generate detailed single-fit plots or batch summary
35 # plots.
36 # -----
37 #
38 #
39 #
40 #
41 #
42 #
43 #
44 #
45 #
46 #
47 #
48 #
49 #
50 #
51 #
52 #
53 #
54 #
55 #
56 #
57 #
58 #
59 #
60 #
61 #
62 #
63 #
64 #
65 #
66 #
67 #
68 #
69 #
70 #
71 #
72 #
73 #
74 #
75 #
76 #
77 #
78 #
79 #
80 #
81 #
82 #
83 #
84 #
85 #
86 #
87 #
88 #
89 #
90 #
91 #
92 #
93 #
94 #
95 #
96 #
97 #
98 #
99 #
100 #
101 #
102 #
103 #
104 #
105 #
106 #
107 #
108 #
109 #
110 #
111 #
112 #
113 #
114 #
115 #
116 #
117 #
118 #
119 #
120 #
121 #
122 #
123 #
124 #
125 #
126 #
127 #
128 #
129 #
130 #
131 #
132 #
133 #
134 #
135 #
136 #
137 #
138 #
139 #
140 #
141 #
142 #
143 #
144 #
145 #
146 #
147 #
148 #
149 #
150 #
151 #
152 #
153 #
154 #
155 #
156 #
157 #
158 #
159 #
160 #
161 #
162 #
163 #
164 #
165 #
166 #
167 #
168 #
169 #
170 #
171 #
172 #
173 #
174 #
175 #
176 #
177 #
178 #
179 #
180 #
181 #
182 #
183 #
184 #
185 #
186 #
187 #
188 #
189 #
190 #
191 #
192 #
193 #
194 #
195 #
196 #
197 #
198 #
199 #
200 #
201 #
202 #
203 #
204 #
205 #
206 #
207 #
208 #
209 #
210 #
211 #
212 #
213 #
214 #
215 #
216 #
217 #
218 #
219 #
220 #
221 #
222 #
223 #
224 #
225 #
226 #
227 #
228 #
229 #
230 #
231 #
232 #
233 #
234 #
235 #
236 #
237 #
238 #
239 #
240 #
241 #
242 #
243 #
244 #
245 #
246 #
247 #
248 #
249 #
250 #
251 #
252 #
253 #
254 #
255 #
256 #
257 #
258 #
259 #
260 #
261 #
262 #
263 #
264 #
265 #
266 #
267 #
268 #
269 #
270 #
271 #
272 #
273 #
274 #
275 #
276 #
277 #
278 #
279 #
280 #
281 #
282 #
283 #
284 #
285 #
286 #
287 #
288 #
289 #
290 #
291 #
292 #
293 #
294 #
295 #
296 #
297 #
298 #
299 #
300 #
301 #
302 #
303 #
304 #
305 #
306 #
307 #
308 #
309 #
310 #
311 #
312 #
313 #
314 #
315 #
316 #
317 #
318 #
319 #
320 #
321 #
322 #
323 #
324 #
325 #
326 #
327 #
328 #
329 #
330 #
331 #
332 #
333 #
334 #
335 #
336 #
337 #
338 #
339 #
340 #
341 #
342 #
343 #
344 #
345 #
346 #
347 #
348 #
349 #
350 #
351 #
352 #
353 #
354 #
355 #
356 #
357 #
358 #
359 #
360 #
361 #
362 #
363 #
364 #
365 #
366 #
367 #
368 #
369 #
370 #
371 #
372 #
373 #
374 #
375 #
376 #
377 #
378 #
379 #
380 #
381 #
382 #
383 #
384 #
385 #
386 #
387 #
388 #
389 #
390 #
391 #
392 #
393 #
394 #
395 #
396 #
397 #
398 #
399 #
400 #
401 #
402 #
403 #
404 #
405 #
406 #
407 #
408 #
409 #
410 #
411 #
412 #
413 #
414 #
415 #
416 #
417 #
418 #
419 #
420 #
421 #
422 #
423 #
424 #
425 #
426 #
427 #
428 #
429 #
430 #
431 #
432 #
433 #
434 #
435 #
436 #
437 #
438 #
439 #
440 #
441 #
442 #
443 #
444 #
445 #
446 #
447 #
448 #
449 #
450 #
451 #
452 #
453 #
454 #
455 #
456 #
457 #
458 #
459 #
460 #
461 #
462 #
463 #
464 #
465 #
466 #
467 #
468 #
469 #
470 #
471 #
472 #
473 #
474 #
475 #
476 #
477 #
478 #
479 #
480 #
481 #
482 #
483 #
484 #
485 #
486 #
487 #
488 #
489 #
490 #
491 #
492 #
493 #
494 #
495 #
496 #
497 #
498 #
499 #
500 #
501 #
502 #
503 #
504 #
505 #
506 #
507 #
508 #
509 #
510 #
511 #
512 #
513 #
514 #
515 #
516 #
517 #
518 #
519 #
520 #
521 #
522 #
523 #
524 #
525 #
526 #
527 #
528 #
529 #
530 #
531 #
532 #
533 #
534 #
535 #
536 #
537 #
538 #
539 #
540 #
541 #
542 #
543 #
544 #
545 #
546 #
547 #
548 #
549 #
550 #
551 #
552 #
553 #
554 #
555 #
556 #
557 #
558 #
559 #
560 #
561 #
562 #
563 #
564 #
565 #
566 #
567 #
568 #
569 #
570 #
571 #
572 #
573 #
574 #
575 #
576 #
577 #
578 #
579 #
580 #
581 #
582 #
583 #
584 #
585 #
586 #
587 #
588 #
589 #
590 #
591 #
592 #
593 #
594 #
595 #
596 #
597 #
598 #
599 #
600 #
601 #
602 #
603 #
604 #
605 #
606 #
607 #
608 #
609 #
610 #
611 #
612 #
613 #
614 #
615 #
616 #
617 #
618 #
619 #
620 #
621 #
622 #
623 #
624 #
625 #
626 #
627 #
628 #
629 #
630 #
631 #
632 #
633 #
634 #
635 #
636 #
637 #
638 #
639 #
640 #
641 #
642 #
643 #
644 #
645 #
646 #
647 #
648 #
649 #
650 #
651 #
652 #
653 #
654 #
655 #
656 #
657 #
658 #
659 #
660 #
661 #
662 #
663 #
664 #
665 #
666 #
667 #
668 #
669 #
670 #
671 #
672 #
673 #
674 #
675 #
676 #
677 #
678 #
679 #
680 #
681 #
682 #
683 #
684 #
685 #
686 #
687 #
688 #
689 #
690 #
691 #
692 #
693 #
694 #
695 #
696 #
697 #
698 #
699 #
700 #
701 #
702 #
703 #
704 #
705 #
706 #
707 #
708 #
709 #
710 #
711 #
712 #
713 #
714 #
715 #
716 #
717 #
718 #
719 #
720 #
721 #
722 #
723 #
724 #
725 #
726 #
727 #
728 #
729 #
730 #
731 #
732 #
733 #
734 #
735 #
736 #
737 #
738 #
739 #
740 #
741 #
742 #
743 #
744 #
745 #
746 #
747 #
748 #
749 #
750 #
751 #
752 #
753 #
754 #
755 #
756 #
757 #
758 #
759 #
760 #
761 #
762 #
763 #
764 #
765 #
766 #
767 #
768 #
769 #
770 #
771 #
772 #
773 #
774 #
775 #
776 #
777 #
778 #
779 #
780 #
781 #
782 #
783 #
784 #
785 #
786 #
787 #
788 #
789 #
790 #
791 #
792 #
793 #
794 #
795 #
796 #
797 #
798 #
799 #
800 #
801 #
802 #
803 #
804 #
805 #
806 #
807 #
808 #
809 #
810 #
811 #
812 #
813 #
814 #
815 #
816 #
817 #
818 #
819 #
820 #
821 #
822 #
823 #
824 #
825 #
826 #
827 #
828 #
829 #
830 #
831 #
832 #
833 #
834 #
835 #
836 #
837 #
838 #
839 #
840 #
841 #
842 #
843 #
844 #
845 #
846 #
847 #
848 #
849 #
850 #
851 #
852 #
853 #
854 #
855 #
856 #
857 #
858 #
859 #
860 #
861 #
862 #
863 #
864 #
865 #
866 #
867 #
868 #
869 #
870 #
871 #
872 #
873 #
874 #
875 #
876 #
877 #
878 #
879 #
880 #
881 #
882 #
883 #
884 #
885 #
886 #
887 #
888 #
889 #
890 #
891 #
892 #
893 #
894 #
895 #
896 #
897 #
898 #
899 #
900 #
901 #
902 #
903 #
904 #
905 #
906 #
907 #
908 #
909 #
910 #
911 #
912 #
913 #
914 #
915 #
916 #
917 #
918 #
919 #
920 #
921 #
922 #
923 #
924 #
925 #
926 #
927 #
928 #
929 #
930 #
931 #
932 #
933 #
934 #
935 #
936 #
937 #
938 #
939 #
940 #
941 #
942 #
943 #
944 #
945 #
946 #
947 #
948 #
949 #
950 #
951 #
952 #
953 #
954 #
955 #
956 #
957 #
958 #
959 #
960 #
961 #
962 #
963 #
964 #
965 #
966 #
967 #
968 #
969 #
970 #
971 #
972 #
973 #
974 #
975 #
976 #
977 #
978 #
979 #
980 #
981 #
982 #
983 #
984 #
985 #
986 #
987 #
988 #
989 #
990 #
991 #
992 #
993 #
994 #
995 #
996 #
997 #
998 #
999 #
1000 #
```

```

41 # Core Algorithmic Functions
42 #
43
44 def load_spectrum(filepath, is_sdss=False):
45     """ Loads a spectrum from a FITS file, supporting SDSS and MILES
46     formats. """
47     try:
48         with fits.open(filepath) as hdul:
49             if is_sdss:
50                 data = hdul[1].data
51                 flux, ivar = data['flux'], data['ivar']
52                 wave = 10**data['loglam']
53                 redshift = hdul[2].data['Z'][0]
54                 return wave, flux, redshift, ivar
55             else:
56                 flux = hdul[0].data.flatten()
57                 header = hdul[0].header
58                 crval, cdelt = header['CRVAL1'], header['CDEL11']
59                 wave = crval + (np.arange(len(flux)) - (header['CRPIX1'
60 ] - 1)) * cdelt
61                 return wave, flux
62     except Exception as e:
63         print(f"ERROR: Could not load FITS file {filepath}: {e}")
64         return (None, None, None, None) if is_sdss else (None, None)
65
66 def log_rebin(wave, flux, vel_scale):
67     """ Rebins a spectrum to a logarithmic wavelength scale. """
68     if wave is None or flux is None: return None, None
69     ln_wave_range = np.log(wave[[0, -1]])
70     n_pixels = int(np.ceil((ln_wave_range[1] - ln_wave_range[0]) / (
71 vel_scale / C_KMS)))
72     new_ln_wave = np.linspace(ln_wave_range[0], ln_wave_range[1],
73 n_pixels)
74     new_wave = np.exp(new_ln_wave)
75     new_flux = np.interp(new_wave, wave, flux)
76     return new_wave, new_flux
77
78 def broaden_spectrum(flux, sigma_kms, vel_scale):
79     """ Broadens a spectrum by a Gaussian kernel. """
80     if sigma_kms <= 0: return flux
81     sigma_pixels = sigma_kms / vel_scale
82     kernel_size = int(np.ceil(sigma_pixels * 10))
83     if kernel_size % 2 == 0: kernel_size += 1
84     x = np.arange(kernel_size) - kernel_size // 2
85     kernel = np.exp(-0.5 * (x / sigma_pixels)**2)
86     kernel /= kernel.sum()
87     return signal.convolve(flux, kernel, mode='same')
88
89 def _estimate_sigma_error(sigma_grid, chi2_grid):
90     """ Estimates the 1-sigma error on sigma by fitting a parabola to
91     the chi^2 curve. """
92     try:
93         min_idx = np.argmin(chi2_grid)
94         fit_slice = slice(max(0, min_idx - 5), min_idx + 6)

```

```

90     x, y = sigma_grid[fit_slice], chi2_grid[fit_slice]
91     p, V = np.polyfit(x, y, 2, cov=True)
92     if p[0] <= 0: return None
93     return 1 / np.sqrt(p[0])
94 except (np.linalg.LinAlgError, ValueError):
95     return None
96
97 def _fit_single_template(gal_flux, template_flux, vel_scale, weights,
98 mask, poly_degree):
99     """
100     Core fitting routine for a single template using a robust additive
101     model.
102     Model: Galaxy ~ c0 * Broadened_Template + Additive_Polynomial
103     """
104     sigma_grid_coarse = np.arange(50, 451, 10)
105     chi2_values = []
106
107     x_coords = np.linspace(-1, 1, len(gal_flux))
108     leg_basis = np.vstack([legendre.legval(x_coords, [0]*k + [1]) for k
109 in range(poly_degree + 1)]).T
110
111     w = np.sqrt(weights[mask])
112     w_gal_flux = gal_flux[mask] * w
113
114     for sigma_val in sigma_grid_coarse:
115         broadened_template = broaden_spectrum(template_flux, sigma_val,
116 vel_scale)
117         A = np.hstack([broadened_template[mask, np.newaxis], leg_basis[
118 mask]])
119         w_A = A * w[:, np.newaxis]
120         try:
121             _, res, _, _ = np.linalg.lstsq(w_A, w_gal_flux, rcond=None)
122             chi2 = res[0] if len(res) > 0 else np.inf
123         except np.linalg.LinAlgError: chi2 = np.inf
124         chi2_values.append(chi2)
125
126     min_idx_coarse = np.argmin(chi2_values)
127     best_sigma_coarse = sigma_grid_coarse[min_idx_coarse]
128
129     sigma_grid_fine = np.linspace(best_sigma_coarse - 15,
130 best_sigma_coarse + 15, 31)
131     min_chi2, best_sigma, best_coeffs = np.inf, best_sigma_coarse, None
132     fine_chi2_values = []
133
134     for sigma_val in sigma_grid_fine:
135         broadened_template = broaden_spectrum(template_flux, sigma_val,
136 vel_scale)
137         A = np.hstack([broadened_template[mask, np.newaxis], leg_basis[
138 mask]])
139         w_A = A * w[:, np.newaxis]
140         try:
141             coeffs, res, _, _ = np.linalg.lstsq(w_A, w_gal_flux, rcond=
142 None)
143             chi2 = res[0] if len(res) > 0 else np.inf
144         except np.linalg.LinAlgError: chi2 = np.inf
145         fine_chi2_values.append(chi2)
146         if chi2 < min_chi2:
147             min_chi2, best_sigma, best_coeffs = chi2, sigma_val, coeffs

```

```

139
140     sigma_err = _estimate_sigma_error(sigma_grid_fine, np.array(
fine_chi2_values))
141     return best_sigma, sigma_err, min_chi2, best_coeffs
142
143 #
-----
144 # Plotting Functions
145 #
-----
146
147 def plot_single_fit(fit_data):
148     """ Displays a detailed plot for a single galaxy fit. """
149     res = fit_data['result']
150     plt.style.use('seaborn-v0_8-whitegrid')
151     fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 9), sharex=True,
gridspec_kw={'height_ratios': [3, 1]})
152     err_str = f" \u00B1 {res['sigma_err_kms']:.2f}" if res['
sigma_err_kms'] is not None else ""
153     title = (f"Galaxy: {res['galaxy_file']} | "
154             f"${sigma}_{true}} = {res['best_sigma_kms']:.2f}{err_str}$
km/s | "
155             f"${chi_v^2} = {res['min_chi2_reduced']:.3f}$\nTemplate: {
res['best_template_file']}")
156     fig.suptitle(title, fontsize=16)
157
158     ax1.plot(fit_data['wave'], fit_data['flux'], 'k-', label='Galaxy
Spectrum', lw=1.5, ds='steps-mid')
159     ax1.plot(fit_data['wave'], fit_data['model'], 'r-', label='Best-fit
Model', lw=1.5, alpha=0.8)
160     ax1.plot(fit_data['wave'][~fit_data['mask']], fit_data['flux'][~
fit_data['mask']], 'x', color='limegreen', ms=6, label='Masked
Pixels')
161     ax1.set_ylabel("Flux"), ax1.legend(), ax1.set_title("Fit Comparison
"), ax1.set_ylim(bottom=0)
162
163     residuals = (fit_data['flux'] - fit_data['model']) * np.sqrt(
fit_data['ivar'])
164     ax2.plot(fit_data['wave'][fit_data['mask']], residuals[fit_data['
mask']], 'g-', lw=1, ds='steps-mid')
165     ax2.axhline(0, color='grey', linestyle='--'), ax2.set_xlabel("Rest-
frame Wavelength ($\AA$)")
166     ax2.set_ylabel("Residuals (Flux/Error)", ax2.set_ylim(-5, 5)
167     plt.tight_layout(rect=[0, 0.03, 1, 0.94]), plt.show()
168
169 def plot_batch_summary(all_fit_data):
170     """ Displays a summary plot with multiple subplots for a batch job.
"""
171     n_plots = len(all_fit_data)
172     if n_plots == 0: return
173
174     n_cols = int(np.ceil(np.sqrt(n_plots)))
175     n_rows = int(np.ceil(n_plots / n_cols))
176
177     fig, axes = plt.subplots(n_rows, n_cols, figsize=(5 * n_cols, 4 *
n_rows), squeeze=False)

```

```

178     fig.suptitle("Batch Fitting Summary", fontsize=20)
179
180     for i, fit_data in enumerate(all_fit_data):
181         ax = axes.flat[i]
182         res = fit_data['result']
183         err_str = f"${res['sigma_err_kms']:.1f}" if res['
sigma_err_kms'] else ""
184         ax.plot(fit_data['wave'], fit_data['flux'], 'k-', lw=1, ds='
steps-mid')
185         ax.plot(fit_data['wave'], fit_data['model'], 'r-', lw=1, alpha
=0.8)
186         ax.set_title(f"{res['galaxy_file']}\n$\sigma={res['
best_sigma_kms']:.1f}{err_str}$ km/s, $\chi_v^2={res['
min_chi2_reduced']:.2f}$")
187         ax.tick_params(axis='x', labelsize=8), ax.tick_params(axis='y',
labelsizesize=8)
188
189     for j in range(i + 1, len(axes.flat)):
190         axes.flat[j].set_visible(False)
191     plt.tight_layout(rect=[0, 0, 1, 0.96]), plt.show()
192
193 #
-----
194 # Main Library Function
195 #
-----
196
197 def vdisp_fit(input_path, template_folder, output_csv=None, plot=False,
198               poly_degree=8, velscale=60.0, fit_range=(4200, 6500),
199               instrumental_sigma=25.0):
200     """
201     Measures velocity dispersion for one or more galaxies.
202
203     Args:
204         input_path (str or Path): Path to a galaxy FITS file or a
205         directory of FITS files.
206         template_folder (str or Path): Path to the folder with template
207         FITS files.
208         output_csv (str or Path, optional): If provided, saves results
209         to this CSV file. Defaults to None.
210         plot (bool, optional): If True, shows plots of the fits.
211         Defaults to False.
212         poly_degree (int, optional): Degree of the additive polynomial.
213         Defaults to 8.
214         velscale (float, optional): Velocity scale in km/s per pixel.
215         Defaults to 60.0.
216         fit_range (tuple, optional): Wavelength range (min, max) for
217         fitting. Defaults to (4200, 6500).
218         instrumental_sigma (float, optional): Instrumental dispersion
219         difference in km/s. Defaults to 25.0.
220
221     Returns:
222         list: A list of dictionaries, where each dictionary contains
223         the fitting result for one galaxy.
224     """
225     input_path, template_folder = Path(input_path), Path(

```



```

template_folder)
217
218 # --- Find Input Files ---
219 if input_path.is_dir():
220     galaxy_files = sorted(list(input_path.glob('*fits')))
221 elif input_path.is_file() and input_path.suffix.lower() == '.fits':
222     galaxy_files = [input_path]
223 else:
224     print(f"FATAL ERROR: No FITS files found at '{input_path}'.")
225     return []
226
227 template_files = sorted(list(template_folder.glob('*fits')))
228 if not template_files:
229     print(f"FATAL ERROR: No template FITS files found in '{
template_folder}'.")
230     return []
231
232 # --- Setup ---
233 all_results = []
234 all_plot_data = []
235 emission_lines = {
236     'H-beta': (4850, 4875), 'OIII': (4950, 5020), 'NI': (5190,
5210),
237     'HeI': (5865, 5885), 'OI': (6290, 6310), 'NII': (6535,
6595),
238     }
239
240 print(f"Found {len(galaxy_files)} galaxy spectra and {len(
template_files)} templates.")
241
242 # --- Main Processing Loop ---
243 for i, galaxy_file in enumerate(galaxy_files):
244     print(f"\n--- Processing ({i+1}/{len(galaxy_files)}): {
galaxy_file.name} ---")
245
246     # 1. Load and preprocess galaxy data
247     gal_wave_obs, gal_flux, gal_z, gal_ivar = load_spectrum(
galaxy_file, is_sdss=True)
248     if gal_wave_obs is None: continue
249
250     gal_wave_rest = gal_wave_obs / (1 + gal_z)
251     gal_wave_log, gal_flux_log = log_rebin(gal_wave_rest, gal_flux,
velscale)
252     _, gal_ivar_log = log_rebin(gal_wave_rest, gal_ivar, velscale)
253
254     fit_idx = np.where((gal_wave_log >= fit_range[0]) & (
gal_wave_log <= fit_range[1]))
255     gal_wave_fit, gal_flux_fit, gal_ivar_fit = gal_wave_log[fit_idx
], gal_flux_log[fit_idx], gal_ivar_log[fit_idx]
256
257     mask = gal_ivar_fit > 0
258     for line_range in emission_lines.values():
259         em_mask = (gal_wave_fit >= line_range[0]) & (gal_wave_fit
<= line_range[1])
260         mask[em_mask] = False
261
262     # 2. Iterate through templates to find the best fit
263     min_chi2_reduced, best_template_file, best_fit_params = np.inf,

```

```

None, None
264     overall_best_sigma_fit, overall_sigma_err = None, None
265
266     for j, template_file in enumerate(template_files):
267         print(f"\r Fitting templates: {j+1}/{len(template_files)}"
, end="")
268         template_wave, template_flux = load_spectrum(Path(
template_file))
269         if template_wave is None: continue
270
271         _, template_flux_log = log_rebin(template_wave,
template_flux, velscale)
272         template_interp = np.interp(gal_wave_fit, np.exp(np.
linspace(np.log(template_wave[0]), np.log(template_wave[-1]), len(
template_flux_log))), template_flux_log)
273
274         sigma, sigma_err, chi2, coeffs = _fit_single_template(
gal_flux_fit, template_interp, velscale, gal_ivar_fit, mask,
poly_degree)
275         if sigma is None: continue
276
277         dof = np.sum(mask) - (poly_degree + 1 + 1)
278         current_chi2_reduced = chi2 / dof if dof > 0 else np.inf
279
280         if current_chi2_reduced < min_chi2_reduced:
281             min_chi2_reduced, overall_best_sigma_fit,
overall_sigma_err = current_chi2_reduced, sigma, sigma_err
282             best_template_file, best_fit_params = template_file,
coeffs
283
284         print("\n Fit complete.")
285         if best_template_file is None:
286             print(" -> WARNING: Fit failed for this galaxy.")
287             continue
288
289         # 3. Final calculations
290         sigma_corr_sq = instrumental_sigma**2
291         final_sigma = np.sqrt(overall_best_sigma_fit**2 - sigma_corr_sq
) if overall_best_sigma_fit**2 > sigma_corr_sq else 0.0
292
293         result = {
294             'galaxy_file': galaxy_file.name, 'best_sigma_kms':
final_sigma,
295             'sigma_err_kms': overall_sigma_err, 'min_chi2_reduced':
min_chi2_reduced,
296             'best_template_file': Path(best_template_file).name
297         }
298         all_results.append(result)
299
300         err_str = f" \u00B1 {result['sigma_err_kms']:.2f}" if result['
sigma_err_kms'] is not None else ""
301         print(f" -> RESULT: Sigma = {result['best_sigma_kms']:.2f}{
err_str} km/s, Chi^2_red = {result['min_chi2_reduced']:.3f}")
302
303         # 4. Store data for plotting if requested
304         if plot:
305             best_template_wave, best_template_flux = load_spectrum(Path
(best_template_file))

```

```

306         _, best_template_flux_log = log_rebin(best_template_wave,
best_template_flux, velscale)
307         best_template_interp = np.interp(gal_wave_fit, np.exp(np.
linspace(np.log(best_template_wave[0]), np.log(best_template_wave
[-1]), len(best_template_flux_log))), best_template_flux_log)
308         best_broadened_template = broaden_spectrum(
best_template_interp, overall_best_sigma_fit, velscale)
309         x_coords_plot = np.linspace(-1, 1, len(gal_wave_fit))
310         additive_poly = legendre.legval(x_coords_plot,
best_fit_params[1:])
311         best_fit_model = best_fit_params[0] *
best_broadened_template + additive_poly
312         all_plot_data.append({
313             'result': result, 'wave': gal_wave_fit, 'flux':
gal_flux_fit,
314             'model': best_fit_model, 'mask': mask, 'ivar':
gal_ivar_fit
315         })
316
317     # --- Final Output ---
318     if output_csv:
319         try:
320             with open(output_csv, 'w', newline='') as csvfile:
321                 fieldnames = ['galaxy_file', 'best_sigma_kms', '
sigma_err_kms', 'min_chi2_reduced', 'best_template_file']
322                 writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
323                 writer.writeheader()
324                 writer.writerows(all_results)
325                 print(f"\nSuccessfully saved results to {output_csv}")
326             except IOError as e:
327                 print(f"\nERROR: Could not write to CSV file {output_csv}:
{e}")
328
329     if plot:
330         if len(all_plot_data) == 1:
331             plot_single_fit(all_plot_data[0])
332         elif len(all_plot_data) > 1:
333             plot_batch_summary(all_plot_data)
334
335     return all_results
336
337 #
-----
338 # Command-Line Execution
339 #
-----
340
341 def _parse_cli_args():
342     """ Parses command-line arguments for standalone execution. """
343     parser = argparse.ArgumentParser(description="Fit galaxy velocity
dispersion.")
344     parser.add_argument("input_path", type=Path, help="Path to a galaxy
FITS file or a directory of FITS files.")
345     parser.add_argument("template_folder", type=Path, help="Path to the
folder with template FITS files.")
346     parser.add_argument("-o", "--output", type=Path, default="

```

```

347     velocity_dispersion_results.csv", help="Output CSV file name.")
348     parser.add_argument("-p", "--plot", action="store_true", help="Show
349         plots of the fits.")
350     parser.add_argument("--poly_degree", type=int, default=8, help="
351         Degree of the additive polynomial.")
352     parser.add_argument("--velscale", type=float, default=60.0, help="
353         Velocity scale in km/s per pixel.")
354     parser.add_argument("--fit_range", type=float, nargs=2, default
355         =[4200, 6500], help="Wavelength range for fitting (min max).")
356     parser.add_argument("--instrumental_sigma", type=float, default
357         =25.0, help="Instrumental dispersion difference in km/s.")
358     return parser.parse_args()
359
360 if __name__ == '__main__':
361     """ This block runs when the script is executed directly from the
362     command line. """
363     args = _parse_cli_args()
364     vdisp_fit(
365         input_path=args.input_path,
366         template_folder=args.template_folder,
367         output_csv=args.output,
368         plot=args.plot,
369         poly_degree=args.poly_degree,
370         velscale=args.velscale,
371         fit_range=tuple(args.fit_range),
372         instrumental_sigma=args.instrumental_sigma
373     )

```

Listing 1: The full Python source code for the vdisp_fit pipeline.