Bioinformatics Advances, 2026, **6**(1), vbag053
https://doi.org/10.1093/bioadv/vbag053
Published: 15 February 2026
Original Article | Proteomics and metabolomics

OXFORD

Protein abundance inference via expectation-maximization in fluorosequencing

Javier Kipen[1,*], Matthew Beauregard Smith[2], Thomas Blom[2], Sophia Bailing Zhou[3], Edward M. Marcotte[4], Joakim Jaldén[1]

[1]Department of Intelligent Systems, Division of Information Science and Engineering, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden
[2]Erisyon, Inc., Austin, TX 78752, United States
[3]Technical University of Munich, Munich 80333, Germany
[4]Department of Molecular Biosciences, The University of Texas at Austin, Austin, TX 78712, United States
*Corresponding author. Department of Intelligent Systems, Division of Information Science and Engineering, KTH Royal Institute of Technology, Malvinas väg 10, 114 28 Stockholm, Sweden. E-mail: kipen@kth.se.
Associate Editor: Nurcan Tunçbağ

Abstract

Summary: **Fluorosequencing generates millions of single peptide reads, yet a principled route to quantitative protein abundances has been lacking. We present a probabilistic framework that adapts expectation–maximization (EM) to the fluorosequencing measurement process, using posterior peptide probabilities from existing classifiers to estimate relative protein abundances. The algorithm iteratively updates abundances to maximize the likelihood of observed reads. We first evaluate five-protein simulations with realistic labeling and system errors. A simple Python implementation processes one million reads in under ten seconds on a standard workstation and reduces the mean absolute error by over an order of magnitude relative to a uniform-abundance guess, indicating robust performance in small-scale settings. We also assess scalability with full human-proteome simulations (20 642 proteins). Ten million reads are processed in under four hours on an NVIDIA DGX with a single Tesla V100 GPU, confirming tractability at proteome scale. Under current fluorosequencing error rates, the method yields modest accuracy gains, but when error rates are reduced, estimation error drops markedly, indicating that chemistry improvements would translate directly into more accurate quantitative proteomics. Overall, EM-based inference provides a scalable, model-driven bridge from peptide-level classification to protein-level quantification in fluorosequencing. Furthermore, the framework can also serve as a refinement step within other inference methods.**

Availability and implementation: **The code and data utilized to produce all the results of this paper is at https://github.com/JavierKipen/ProtInfGPU.**

1 Introduction

Next-generation sequencing has transformed genetic analysis, enabling rapid, low-cost whole-genome sequencing (Eren *et al.* 2022). In contrast, protein sequencing technologies have lagged behind, despite proteins being the primary functional molecules in cells and often more directly indicative of biological state. Many biological and medical applications, from cancer biomarker discovery to the analysis of complex signaling pathways, rely not only on identifying proteins but also on quantifying their abundances with high precision. Existing approaches, such as mass spectrometry and affinity-based assays, have made considerable progress, yet they often struggle to simultaneously achieve high sensitivity, digital quantification, and multiplexed measurement within a single platform (Timp and Timp 2020).

Single-molecule protein sequencing (SMPS) seeks to address this gap (Restrepo-Pérez *et al.* 2018, Alfaro *et al.* 2021). Inspired by next-generation DNA sequencing, these techniques aim to identify and quantify individual protein molecules directly, without amplification, and at scale. While recent advances in nanopore-based methods (Brinkerhoff *et al.* 2021, Zhang *et al.* 2021, Yu *et al.* 2023) and emerging commercial platforms (Reed *et al.* 2022) hold significant promise, fluorosequencing (Swaminathan *et al.* 2015, 2018, Mapes *et al.* 2023) offers a compelling SMPS method that combines amino-acid labeling, Edman degradation, and imaging to extract partial peptide sequence information.

Some of the recent progress in fluorosequencing includes estimations of experimental error rates (Smith *et al.* 2024), analysis and reduction of dye-dye interactions (Bachman *et al.* 2022) and algorithms for classifying peptides from fluorescence readouts (Kipen and Jaldén 2023, Smith *et al.* 2023). These latter methods are used as algorithmic components in this paper.

Received: 1 September 2025. **Revised:** 17 January 2026. **Accepted:** 27 January 2026

Although protein inference is well studied for mass spectrometry (Nesvizhskii *et al*. 2003, Huang *et al*. 2012, The *et al*. 2016, Audain *et al*. 2017), analogous methods for fluorosequencing remain underexplored. In the case of mass spectrometry, protein inference is often considered independently of protein quantification, although some work has considered both at once, e.g. Spivak *et al*. (2012); Winkler (2021); He *et al*. (2016). An important distinction for fluorosequencing is that, as a single-molecule sequencing technique, identification and quantification are directly coupled, with each identification corresponding to one discrete peptide molecule in the sample, and hence an individual molecule of the source protein. We thus sought an integrated approach in which protein identifications, based on (single molecule) peptide sequences, were considered in the same statistical framework as their abundances.

In this work, we adapt Expectation-Maximization (EM), a well-established probabilistic inference technique, to fluorosequencing. We present a framework that estimates protein abundances from peptide-level posteriors, produced by peptide-level classifiers from Whatprot (Smith *et al*. 2023), Probeam (Kipen and Jaldén 2023) and an artificial oracle which has access to the true peptide. Here, "peptide posterior probability" denotes the model's estimated likelihood that a read originates from a given peptide after observing the fluorescence data. This algorithm iteratively refines the estimates of protein abundances to maximize the likelihood of the observed fluorescence data. The approach is principled, computationally efficient, and scalable, aligning with the throughput potential of SMPS.

We validate the framework on simulated datasets, showing robust abundance recovery under realistic errors for small protein mixtures and tractable runtimes at proteome scale. Although improvements in proteome-wide abundance estimates are modest under current fluorosequencing error rates, our simulations demonstrate that reductions in these errors lead to markedly more accurate protein-level inference. While these results are encouraging, further methodological advances will be necessary to fully meet the demands of large-scale proteomic analysis in practical applications.

2 Method

2.1 Fluorosequencing

Fluorosequencing is a single-molecule protein sequencing (SMPS) technique inspired by methods used in DNA and RNA sequencing (Swaminathan *et al*. 2015, 2018). The process begins by denaturing proteins and proteolytically cleaving them into peptides, which are then chemically labeled with fluorescent dyes. Millions of these labeled peptides are immobilized in a flow cell and imaged using total internal reflection fluorescence (TIRF) microscopy.

Edman degradation is then performed in cycles, sequentially removing one N-terminal amino acid from each peptide. After each cycle, the fluorescence intensities of the peptides are measured. The pattern of fluorescence intensity drops across cycles is then analyzed by a peptide inference algorithm, which estimates the likelihoods of possible peptides generating that pattern. The possible peptides are known because the proteins in the sample, the labeling chemistry, and the proteolysis rules governing peptide cleavage are all defined.

Several systematic error sources affect inference, including incomplete Edman degradation, dye loss between cycles, and the non-observability of some peptides. These factors have been characterized previously (Smith *et al*. 2023, 2024).

Once peptide sequences and their probabilities are inferred from the fluorescence reads, they can be mapped back to proteins, enabling protein abundance estimation. This protein-level inference has not previously been described for fluorosequencing and is the focus of the present work.

2.2 Protein abundance inference in fluorosequencing

Let $\boldsymbol{X}$ denote the fluorosequencing dataset comprising N_r reads. Each read X_k is a matrix of fluorescence intensities with rows for dye channels and columns for Edman degradation cycles (observations $\boldsymbol{x}$ and x_k denote realizations).

The reads X_k are generated by different peptides attached to the wall. Certain peptides are mutually indistinguishable by fluorosequencing, and the indistinguishable groups are referred to as fluorescence strings. The number of possible fluorescence strings is finite and is determined by several factors: the proteins utilized, the enzymes used to digest the proteins (e.g. trypsin), and the specific amino acids labeled with fluorophores. We introduce the random variable F, representing a uniformly drawn fluorescence string in the experiment, with the domain of all possible fluorescence strings denoted as $\mathcal{D}_\mathrm{F}$. The distribution of F is given by $P_F(f)$ and peptides lacking labeled residues map to the null fluorescence string f_{null}.

The purpose of protein abundance inference is to be able to estimate the protein distribution present in a solution. Let $Y \in \mathcal{D}_\mathrm{Y}$ be a random variable representing a uniformly drawn protein, and $P_Y(y)$ the distribution of proteins, which is the target for estimation. Given that the number of possible proteins N_P is known for our experiments, we can represent $P_Y(y)$ as $P_Y(y) = p_y$, where p_y represents the relative abundance of the yth protein, and $\sum_y p_y = 1$.

Given a known protein distribution $P_Y(y)$, the fluorescence string distribution $P_F(f)$ can be inferred a priori from known cleavage and labeling scheme.

The protein inference problem can be formulated as estimating the protein distribution that maximizes the likelihood of the observed dataset:

$$\widehat{P}_Y = \underset{P_Y}{\operatorname{argmax}}\, P_{\boldsymbol{X}}(\boldsymbol{x}) \tag{1}$$

Solving (1) is a highly complex problem, and in this paper, we present a method to obtain practical estimates $\widehat{P}_Y$. While not necessarily optimal, the proposed method provides meaningful estimations and can be also used to refine other inference techniques.

2.3 Notation

To apply EM to the protein inference, we introduce auxiliary variables. A key challenge is that proteins and reads are not in one-

to-one correspondence: each protein may generate multiple fluorescence strings, some unobservable. This mismatch violates standard EM assumptions on having the same number of hidden and observed variables, a problem that will be solved with our framework.

Let F° denote an *observable* fluorescence string; it shares the domain of F but satisfies $P_{F^{\circ}}(f_{\text{null}}) = 0$ where f_{null} is the fluorescence string that represents no dyes on the string. The probability of dyes not attaching and the difference with f_{null} results in $P_{F^{\circ}}(f) \neq P_F(f)$, and an example is provided in the Appendix, available as supplementary data at *Bioinformatics Advances* online to further clarify the variable F°. However, we can compute $P_{F^{\circ}}(f)$ from a transformation of $P_F(f)$:

$$P_{F^{\circ}}(f) = K(1 - m^{N_{\text{d}}(f)})P_F(f) \tag{2}$$

where m is the dye miss rate, N_{d} is a function that returns the number of amino acids that are ideally labeled and K is a normalization constant ($K^{-1} = \sum_{f \in \mathcal{D}_F}[(1 - m^{N_{\text{d}}(f)})P_F(f)]$). Notice that $N_{\text{d}}(f_{\text{null}}) = 0$, and then the probability of observing the null fluorescence string is zero too. The advantage of F° is that there is one realization per read in the dataset, and that $P_{X_k|F_k}(x_k|f) = P_{X_k|F_k^{\circ}}(x_k|f)$ as the fluorescence string itself is unchanged. Finally, we define F_k° as the experimentally observed fluorescence string that generated the kth read.

Let I be the *protein indicator*, i.e. the protein of origin for a read, with mass function $P_I(y) = q_y$. For clarity, we denote I_k as the protein indicator for the kth read. An example is provided in the Appendix, available as supplementary data at *Bioinformatics Advances* online to further illustrate the concept of I. Both I_k and F_k° are hidden variables, while X_k is observable. The relationship between these variables is illustrated in Fig. 1.

Each probability q_y is related to the original protein distribution $P_Y(y)$ as:

$$q_y = P_I(y) = KE_{F^{\circ}}(y)P_Y(y) = KE_{F^{\circ}}(y)p_y \tag{3}$$

where $E_{F^{\circ}}(y)$ is the average number of observable fluorescence strings produced by protein y, and K is a normalizing factor ($K^{-1} = \sum_{y \in \mathcal{D}_Y}[E_{F^{\circ}}(y)P_Y(y)]$). Equation (3) shows that knowledge of P_Y uniquely determines P_I, and also knowing P_I allows for the reconstruction of P_Y, and an example is provided in the Appendix, available as supplementary data at *Bioinformatics Advances* online to further clarify this expression.

Next, the distribution of experimental fluorescence strings for a given set of protein abundances can be expressed using the above introduced terms. This formulation is essential for generating the datasets used in testing:

$$P_{F^{\circ}}(f) = \sum_{y=1}^{N_{\text{P}}} P_{F^{\circ}I}(f, y) = \sum_{y=1}^{N_{\text{P}}} P_{F^{\circ}|I}(f|y)P_I(y) \tag{4}$$

where $P_I(y)$ is obtained from the protein distribution with (3) and $P_{F^{\circ}|I}(f|y)$ is known. Finally, we define $\widehat{q} = \widehat{P}_I(y)$ as the estimated probabilities of the distribution of protein indicators.

In summary, the protein indicator I is a hidden random variable with a realization for each observed read. This setup allows us to formulate an EM algorithm to maximize the likelihood of the observed data and estimate $\widehat{P}_I$. After fitting, the final step is to apply the inverse transformation from (3) to obtain the estimated $\widehat{P}_Y$.

2.4 Expectation-maximization iteration

The EM algorithm iteratively increases the data likelihood to obtain local maximum-likelihood estimates. We apply EM to update the protein distribution from $\widehat{P}_Y$ to $\widehat{P}_Y'$ such that

$$P_{\boldsymbol{X}}(\boldsymbol{x}|\widehat{P}_Y') \geq P_{\boldsymbol{X}}(\boldsymbol{x}|\widehat{P}_Y). \tag{5}$$

Because P_Y uniquely determines P_I via (3), the EM iteration can be rewritten as:

$$P_{\boldsymbol{X}}(\boldsymbol{x}|\widehat{q}') = P_{\boldsymbol{X}}(\boldsymbol{x}|\widehat{P}_I') \geq P_{\boldsymbol{X}}(\boldsymbol{x}|\widehat{P}_I) = P_{\boldsymbol{X}}(\boldsymbol{x}|\widehat{q}) \tag{6}$$

where $\widehat{q}$ represents the parameters of the current estimated distribution $\widehat{P}_I$ and $\widehat{q}'$ represents the parameters of the updated estimate $\widehat{P}_I'$. The inequality in (6) is guaranteed by our algorithm (proof in Leijon and Henter 2012). The parameter update equation (derived in the Appendix, available as supplementary data at *Bioinformatics Advances* online) is

$$\widehat{q}'_y = \frac{1}{N_{\text{r}}}\sum_{k=1}^{N_{\text{r}}} \gamma_{y,k} = \frac{1}{N_{\text{r}}}\sum_{k=1}^{N_{\text{r}}} \frac{P_{X_k|I_k}(x_k|y)\widehat{q}_y}{\sum_{l=1}^{N_{\text{P}}} P_{X_k|I_k}(x_k|l)\widehat{q}_l}. \tag{7}$$

Intuitively, $\gamma_{y,k}$ can be seen as a joint probability between the kth read X_k and the relative protein abundance of a protein y: I_k, when assuming the abundance of proteins $\widehat{q}$ in I_k. Then averaging over the reads in (7) results in new estimate of the protein indicator distribution: $\widehat{q}'$. Updating from $\widehat{q}$ to $\widehat{q}'$ therefore shifts the protein proportions toward those that consistently explain the observed reads.

The likelihoods $P_{X_k|I_k}(x_k|y)$ can be expressed in terms of the intermediate random variable F_k° (derivation in the Appendix, available as supplementary data at *Bioinformatics Advances* online), yielding

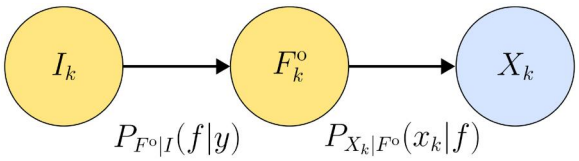

Figure 1 Relationship between the hidden protein indicator I_k, the hidden experimentally observable fluorescence string F_k° and the observable read X_k for the kth read.

$$P_{X_k|I_k}(x_k|y) = \sum_{f \in \mathcal{D}_F(y)} P_{X_k|F_k^o}(x_k|f) P_{F_k^o|I_k}(f|y) \tag{8}$$

2.5 Using the fluorescence string classification for the EM iteration

Algorithms such as Whatprot and Probeam allow us to estimate the distribution of fluorescence strings for a given read, but we use posterior likelihoods in EM. For any classifier that assumes equally distributed fluorescence strings ($P^{u}_{F_k^o}(f)$) and provides a posterior estimate $\widehat{P}_{F_k^o|X_k}(f|x_k)$, we can express

$$\begin{aligned}\widehat{P}_{F_k^o|X_k}(f|x_k) &= \frac{\widehat{P}_{X_k|F_k^o}(x_k|f) P^{u}_{F_k^o}(f)}{\sum_{g \in \mathcal{D}_F} \widehat{P}_{X_k|F_k^o}(x_k|g) P^{u}_{F_k^o}(g)} \\ &= \frac{\widehat{P}_{X_k|F_k^o}(x_k|f)}{\sum_{g \in \mathcal{D}_F} \widehat{P}_{X_k|F_k^o}(x_k|g)}\end{aligned} \tag{9}$$

where the denominator can be written as a normalizing constant C_k. Therefore $\widehat{P}_{X_k|F_k^o}(x_k|f) = C_k \widehat{P}_{F_k^o|X_k}(f|x_k)$. This result is important because it allows us to use the classifier's estimates to compute $P_{X_k|F_k^o}(x_k|f)$ for the protein inference. The multiplicative term C_k is canceled for protein inference. Using this result and (8) we can rewrite the EM update of (7) as

$$\begin{aligned}\widehat{q}'_y &= \frac{1}{N_r} \sum_{k=1}^{N_r} \frac{\eta_{y,k} \widehat{q}_y}{\sum_{l=1}^{N_P} \eta_{l,k} \widehat{q}_l} \\ \eta_{y,k} &= \sum_{f \in \mathcal{D}_F(y)} \widehat{P}_{F_k^o|X_k}(f|x_k) P_{F_k^o|I_k}(f|y)\end{aligned} \tag{10}$$

This equation shows how the updated protein distribution parameters $\widehat{q}'_y$ are obtained by summing over the reads and using the posterior probabilities from the classifier to update the EM steps.

2.6 Optimizing memory and processing time with sparsity on the fluorescence string classification

Parameter updates at proteome scale are costly. We exploit the sparsity of the posteriors $\widehat{P}_{F_k^o|X_k}(f|x_k)$: while there may be a large number of possible fluorescence strings (approximately 15010^3 for the human proteome), only a small subset typically has non-negligible likelihood for any given read.

We model this sparsity as follows:

$$\begin{aligned}\widehat{P}^{s}_{F_k^o|X_k}(f|x_k) &= \widehat{P}_{F_k^o|X_k}(f|x_k) \delta_{\mathcal{G}_k}(f) + r_k \\ \delta_{\mathcal{G}_k}(f) &= \begin{cases} 1 & f \in \mathcal{G}_k \\ 0 & f \notin \mathcal{G}_k \end{cases}.\end{aligned} \tag{11}$$

The set $\mathcal{G}_k$ contains the N_b fluorescence strings with the highest posterior probabilities for the read k. The indicator function therefore selects only the highest posterior probabilities for each read k.

The normalization constant r_k: $r_k = |\mathcal{D}_F|^{-1}\left(1 - \sum_f \widehat{P}_{F_k^o|X_k}(f|x_k)\delta_{\mathcal{G}_k}(f)\right)$ ensures that the remaining probability mass is uniformly distributed among all possible fluorescence strings. This sparsification strategy significantly reduces the computational and memory requirements for estimating $\eta_{y,k}$.

Since we only need to consider fluorescence strings that are both produced by a given protein y and among the top N_b estimated posterior probabilities for a read k, we define the set $\mathcal{D}_F(y,k) = \{f \in (\mathcal{D}_F(y) \cap \mathcal{G}_k)\}$ as the fluorescence strings that must be considered when approximating $\eta_{y,k}$. The sparsified form of the $\eta_{y,k}$ computation becomes:

$$\begin{aligned}\widehat{q}'_y &= \frac{1}{N_r} \sum_{k=1}^{N_r} \frac{\eta^{s}_{y,k} \widehat{q}_y}{\sum_{l=1}^{N_P} \eta^{s}_{l,k} \widehat{q}_l} \\ \eta_{y,k} \approx \eta^{s}_{y,k} &= r_k + \sum_{f \in \mathcal{D}_F(y,k)} \widehat{P}_{F_k^o|X_k}(f|x_k) P_{F_k^o|I_k}(f|y).\end{aligned} \tag{12}$$

After comparing different sparsification and normalization schemes (see Appendix, available as supplementary data at *Bioinformatics Advances* online), the variant presented above showed the best performance empirically on small protein datasets. The behavior might differ at larger scales, but it does effectively reduce computational burden.

2.7 Posterior estimates

This section describes the different methods to obtain estimates of the posterior probabilities $\widehat{P}_{F_k^o|X_k}(f|x_k)$

2.8 Oracle

For benchmarking, we use an oracle posterior with access to the true fluorescence string f_k^t for the kth read. With error rate e, it assigns probability $1-e$ to f_k^t and spreads e uniformly over all fluorescence strings:

$$\widehat{P}^{o}_{F_k^o|X_k}(f|x_k) = (1-e)\delta_{f_k^t}(f) + \frac{e}{|\mathcal{D}_F|} \tag{13}$$

When $e = 0$, the oracle is a perfect classifier and works as an upper bound on downstream abundance accuracy. For example, if the error-free oracle fails to yield accurate protein abundance estimates under certain experimental configurations (number of Edman degradation cycles, labeling strategies, and protease choices), then real-world posterior estimates, which inevitably introduce error, are unlikely to perform adequately under those conditions.

2.9 Whatprot

Whatprot (Smith *et al*. 2023) is a two-stage classifier: a k-nearest neighbors step narrows candidate fluorescence strings, followed by a Hidden Markov Model (HMM) that computes the read likelihood given the fluorescence string. While the HMM in Whatprot allows for direct computation of $P_{X_k,F_k^o}(x_k|f)$, doing so for every possible fluorescence string remains computationally expensive. We used Whatprot to validate the accuracy of Probeam's predictions and subsequently used Probeam for the protein inference

tasks, as its code allowed straightforward output of full posterior distributions.

2.10 Probeam

Probeam (Kipen and Jaldén 2023) is a faster alternative for fluorescence string classification, using a novel state-space and beam search to estimate posteriors. It trades modest accuracy for substantial speed gains.

Unlike Whatprot, Probeam does not model the blocking effect (Smith *et al*. 2024); therefore, we excluded blocking from our simulations. We made this choice for computational efficiency and workflow compatibility: Probeam enables substantially faster peptide-level inference, and the outputs could be easily modified to run the protein inference.

We acknowledge that omitting blocking can affect classification accuracy; fully incorporating it, either by extending Probeam or adapting Whatprot's outputs, would require significant engineering effort and is beyond the scope of this work. Initial investigations show that the accuracy of Whatprot decreases from 9.8% to 7.4% in the peptide inference when considering the blocking effect; we expect Probeam to exhibit a similar behavior. If the effect was incorporated, it would not alter significantly the conclusions of this work.

2.11 Error metric

A standard approach for assessing parameter improvement in an EM implementation is to track the log-likelihood $P_{\boldsymbol{X}}(\boldsymbol{x}|\widehat{q})$ across successive iterations, but this is computationally expensive in our setting. Instead, we report a distributional distance: the MAE, chosen for its simplicity and ease of interpretation since lower values directly indicate a better fit:

$$\mathrm{MAE}(\widehat{P}_Y) = \frac{1}{N_\mathrm{P}} \sum_{y=1}^{N_\mathrm{P}} |\widehat{P}_Y(y) - P_Y(y)| \tag{14}$$

3 Results

3.1 Datasets

Simulated datasets were generated using the Whatprot framework (Smith *et al*. 2023) to assess our algorithm's performance. We used simulations to verify our method since real proteome-scale experimental data is not yet available, and simulations provide the necessary ground-truth protein distributions for comparison, which would be extremely challenging to obtain under real-world conditions.

For both the five-protein and whole-proteome settings, we generated ten datasets with distinct abundance profiles by sampling N_P independent and identically distributed exponential variables and normalizing to obtain valid distributions. Throughout this paper, "protein" refers to gene-level entries in the proteome database. In the proteome case, this produces a dynamic range in which the most abundant protein is typically $\sim 10^6$ times more abundant than the least abundant.

For each experiment we created a shared pool of reads per fluorescence string (1000 for the five-protein case; 100 for the proteome case). Each of the ten datasets was then formed by sampling from this pool according to the fluorescence-string distribution in (4). The error metric used to assess performance was Mean Absolute Error (MAE), defined in (14); plots show mean MAE across datasets as points, with error bands indicating standard deviation.

Throughout the Results section, we refer to the output of peptide-level classifiers such as Whatprot and Probeam as peptide inference. While this output is formally defined in the Methods section as fluorescence string classification, we adopt the term peptide inference here for clarity and to remain consistent with terminology commonly used in related literature. We also used an oracle-based peptide inference method to set a lower bound, and a uniform guess over the protein estimations as the upper bound.

We used Probeam to infer the posterior probabilities of peptides for each read, as described in (9). This choice was primarily motivated by the flexibility of Probeam's in-house implementation, which allowed us to modify the code to output the top N_b peptides rather than only the most likely one. Although prior work suggests that Probeam performs slightly worse than Whatprot for the peptide-inference task (Kipen and Jaldén 2023), we show in the Appendix, available as supplementary data at *Bioinformatics Advances* online that both classifiers yield comparable accuracy on the datasets used in this study.

Whole-proteome runs used GPU acceleration on an NVIDIA DGX with Tesla V100 GPUs (implementation details in the Appendix, available as supplementary data at *Bioinformatics Advances* online); the five-protein experiments used a simple CPU-based Python script to facilitate reproducibility.

3.2 Five proteins abundance estimation

While the fluorosequencing platform is ultimately intended for whole-proteome abundance estimation, it is also well-suited for early-stage development and use cases involving smaller sets of proteins. To explore this regime, we simulate a simplified scenario comprising only five proteins (PSME4, HBA2, HBB, PSME3, and PSMB10), as might arise in the case of a mixture of isolated proteins or a highly enriched, partially purified proteomics sample.

In this experiment, we use two peptide inference strategies to compute the posterior estimates needed for EM: an oracle and a modified version of Probeam, both explained in section Posterior estimates. The error rates of the simulations were parameters used in previous publications and are specified in the Appendix, available as supplementary data at *Bioinformatics Advances* online.

3.3 Comparison of abundance estimation

A direct comparison of convergence curves across all methods is presented in Fig. 2B, which includes the oracle, full Probeam, and sparse Probeam ($N_\mathrm{b} = 40$). The presented computing time on the legend takes into consideration the whole ten datasets. While both Probeam variants converge similarly and lag behind the oracle in terms of final accuracy, the sparse version requires less computations and memory. For five proteins the differences

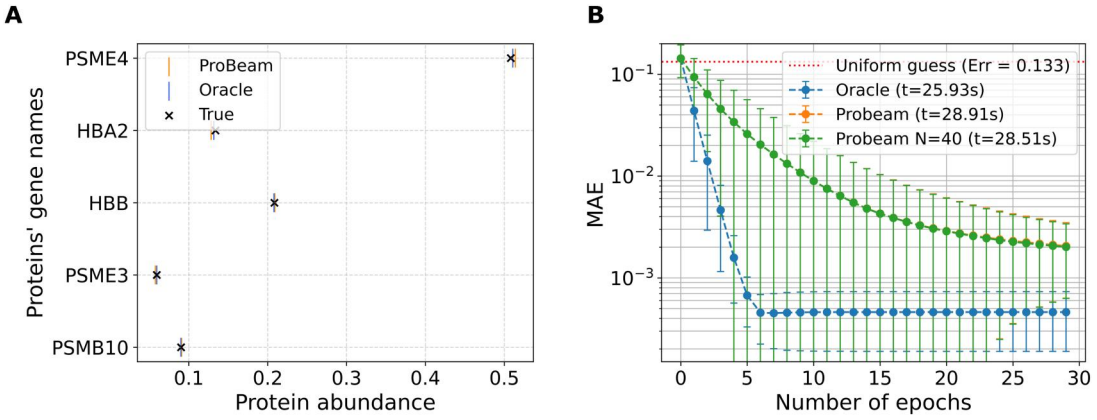

Figure 2 Performance comparison across different posterior estimates for a 5-protein experiment. (A) Protein abundance estimation from single peptide reads. (B) EM curves by inference method.

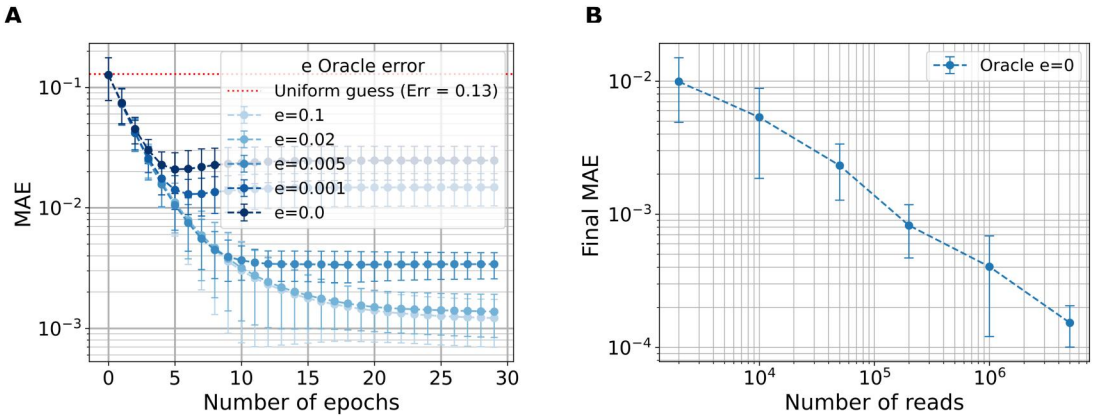

Figure 3 Sensitivity to oracle error rate and number of reads in a 5-protein experiment classified with an oracle. (A) Convergence curves for different oracle classification error rates e ($N_r = 10^6$). (B) Final MAE for different N_r.

between the sparse and non-sparse are not significant, but the sparse method is more scalable in terms of the amount of proteins in the dataset.

Finally, Fig. 2A illustrates the ground truth versus the estimated protein abundances for each method in one of the ten datasets. All estimates closely align with the true values, confirming that even approximate posteriors derived from practical peptide inferences with currently achievable experimental errors can provide useful and reliable abundance estimates for datasets with few proteins.

In summary, this section demonstrates that the proposed EM-based inference method performs robustly in small-scale settings. For applications involving few proteins -whether for prototyping, targeted studies, or constrained biological contexts- this approach offers a practical and effective solution for protein abundance estimation.

3.4 Oracle performance

We analyze how an imperfect peptide classifier oracle with error rate e affects the protein inference quality. As shown in Fig. 3A, increasing the error rate leads to a marked degradation in the accuracy of the protein abundance estimates. This behavior is expected: inaccurate posterior probabilities of peptides result in larger deviations from the true protein abundance. Additionally, the convergence curves do not necessarily decrease monotonically. This is not contradictory to EM theory, which guarantees monotonically increasing data likelihood, not necessarily monotonic improvement in an external metric such as the MAE.

Next, we evaluate the impact of the number of reads on estimation accuracy, as shown in Fig. 3B. With a perfect oracle ($e = 0$), increasing the number of reads consistently enhances the estimation performance. This behavior shows that a larger dataset corresponds to improved abundance accuracy when the peptides are perfectly distinguishable.

3.5 Whole proteome protein inference

This section presents an evaluation of our algorithm's ability to estimate protein abundances at the whole-proteome scale. The dataset was constructed from the human proteome as defined in UniProt, excluding only 18 proteins that did not generate any observable tryptic peptides, resulting in N_p=20 642 proteins.

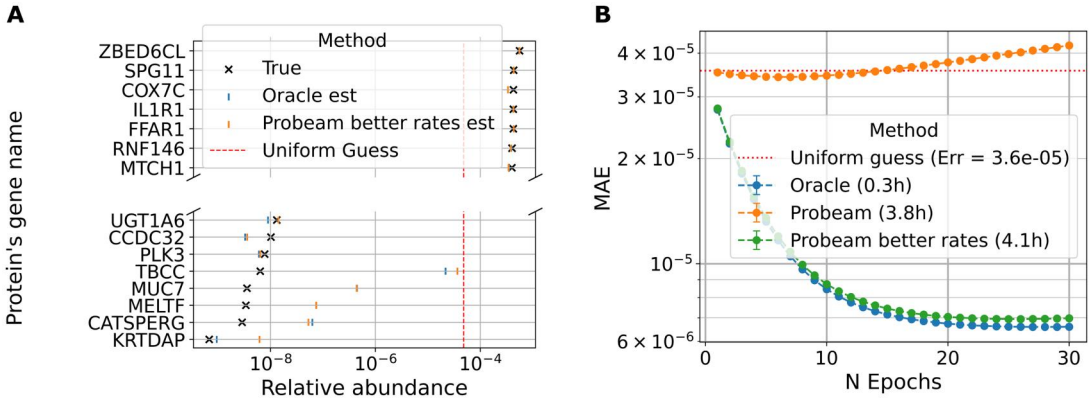

Figure 4 Protein inference on whole proteome ($N_P = 20642$, $N_r = 10^7$). (A) Protein abundance estimation from single peptide reads. (B) EM curves by inference method.

3.6 Comparison of abundance estimation

We now compare protein inference results obtained using the oracle and sparse Probeam ($N_b = 1000$), under both typical and reduced experimental error conditions and 10^6 reads. The sparsity value was chosen so that the computation times remained tractable, and the convergence characteristic was illustrated correspondingly. Figure 4B shows the convergence behavior of the EM algorithm for each case, with runtimes reported as the average per dataset.

Using the method described in the Appendix, available as supplementary data at *Bioinformatics Advances* online, we estimate that each of the ten datasets contains approximately 3×10^5 protein molecules. This method also allows us to estimate the expected number of reads contributed by each protein, and we verified that, on average, 99.9% of proteins were represented by at least one read in each dataset.

The oracle achieves the lowest estimation error, establishing a best-case reference for inference performance. Under standard experimental conditions, Probeam initially performs better than random guessing; however, its estimation error increases with successive EM iterations. This degradation is consistent with the behavior observed in high-error oracle classifiers and can be attributed to the posterior estimates of Probeam not being sparse enough (see Appendix, available as supplementary data at *Bioinformatics Advances* online). To mitigate this degradation, we stop updates at the epoch that minimizes MAE, which yields small but consistent improvements over a uniform baseline. In real experimental data, this criterion is not directly available because the ground-truth abundances are unknown; thus, early stopping should be viewed as a pragmatic heuristic rather than a principled stopping rule. In practice, a reasonable stopping epoch can be calibrated from simulation studies under plausible error regimes.

When experimental error rates are artificially reduced, Probeam produces substantially more accurate estimates. In this regime, the EM algorithm converges to protein abundance distributions that more closely reflect the ground truth. The simulation parameters used to evaluate this improved performance are provided in the Appendix, available as supplementary data at *Bioinformatics Advances* online. A sensitivity analysis with respect to error rates is also presented in the Appendix, available as supplementary data at *Bioinformatics Advances* online. The oracle results can also be interpreted as the limiting case of Probeam as experimental error rates approach zero.

Finally, Fig. 4B also highlights that oracle-based inference requires significantly less computational time than Probeam. Nevertheless, our EM algorithm remains computationally efficient and can be executed at whole-proteome scale using standard GPU resources.

Figure 4A visualizes the inferred protein distribution versus the true distribution for a single dataset. In this figure, proteins are sorted by their true abundances (y-axis), while the x-axis displays the true abundance values and the corresponding EM-based estimates after thirty epochs. The results show clearly that all classifiers outperform random guessing, with the oracle providing the closest match to the ground truth. For visualization clarity, only the improved-error version of Probeam is shown. An additional visualization of this example is included in the Appendix, available as supplementary data at *Bioinformatics Advances* online.

Although relative abundance estimates are less accurate in this large-scale setting than in the five-protein case, the algorithm still provides meaningful improvement over baseline methods. Importantly, our EM framework enhances inference for any initial abundance estimate. Therefore, it can be combined with other methods to achieve a more accurate protein inference on whole-proteome scale.

3.7 Oracle performance

We also analyze the performance of the oracle classifier in this large-scale setting. Figure 5B shows how the final MAE after 30 epochs varies with the number of reads, assuming a perfect peptide sequencing. Figure 5A illustrates the impact of different oracle's classification error probabilities e on protein abundance estimation.

As observed in the five-protein case, increasing the number of reads improves the quality of the abundance estimates. Similarly, increasing the classification error degrades performance, with convergence curves again showing non-monotonic behavior. One key difference in this full-proteome setting is that the error bars in the plots are negligible and not visually apparent, indicating low variance across different datasets.

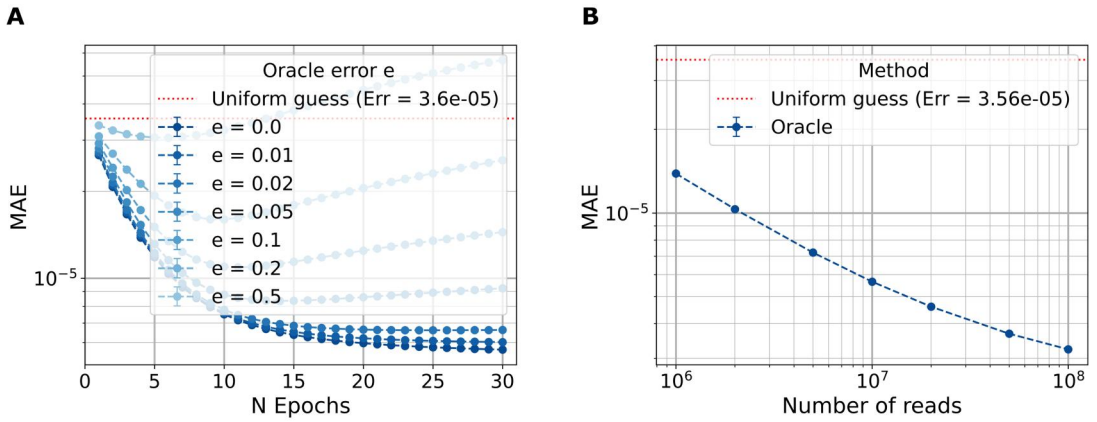

Figure 5 Sensitivity of the algorithm with respect to the number of reads and the classifier error rate. (A) EM for different errors on oracle ($N_r = 10^7$). (B) Final MAE for different N_r.

Overall, the perfect oracle represents an idealized case in which peptide classification is error-free. As expected, it leads to significantly more accurate abundance estimation and serves as a reference point for evaluating also protease and labeling schemes.

4 Discussion

We presented an EM-based protein inference framework tailored to fluorosequencing. Simulations show robust abundance estimates under realistic error rates in small-scale settings and tractable runtime at proteome scale, indicating compatibility with high throughput demands of fluorosequencing. However, this study has limitations such as its reliance on simulations in the absence of proteome scale experimental data based on a reliable ground truth, and the modest estimation improvements at proteome scale under current error rates, although it establishes a framework that can benefit directly from future improvements in chemistry and classifiers.

The scalability of our algorithm is critical for single-molecule protein sequencing, where throughput exceeds that of traditional protein sequencing methods. This approach is in principle applicable to related technologies that obtain peptide-level information, including nanopore-based methods (Brinkerhoff *et al*. 2021, Zhang *et al*. 2021). These technologies can provide peptide-level posteriors; with minor adjustments to how peptides are generated (e.g. removing the fluorosequencing-specific non-observability of dye-free fluorescence strings), our EM framework can convert their read-level outputs into protein abundance estimates.

From an accuracy standpoint, future work could explore the integration of expert prior knowledge or alternatively sourced information into the prior distributions on proteins, as is, e.g. done with TPM data in IsoBayes (Bollon *et al*. 2025). In addition, this EM-based method could serve as a refinement step for other inference pipelines by starting the EM protein inference from an initial abundance estimate rather than from a uniform protein abundance, as we did in this work. Another avenue would be to devise efficient approximations of the full data likelihood to apply a multi-start approach in the EM. Such computations would make it feasible to run EM from different initial protein distributions and, after convergence, rank the resulting solutions according to their likelihood, retaining the one that achieves the highest local maximum.

On the performance side, working on optimizing sparsification of posteriors, EM acceleration (e.g. variational or stochastic EM, quasi-Newton updates), and lightweight partial/batch updates of the protein abundances could reduce memory usage and runtime further.

Finally, the integration of deep learning into this pipeline presents a compelling opportunity. One option could be to train neural networks to predict protein abundances directly from the raw fluorescence reads, by passing some of the intermediate steps entirely. Alternatively, one could learn the η values in (10) used in the EM updates using a neural network, allowing the EM framework to function in conjunction with learned components. Both approaches could be trained with supervision, minimizing a loss with the estimation against the ground truth obtained in simulations, and potentially using architectures like Convolutional Neural Networks and attention mechanisms. Such hybrid or full data-driven approaches may not only accelerate inference but also increase robustness to modeling mismatches and experimental noise, effectively adapting the model to the data.

Acknowledgements

ChatGPT was used to improve the cohesion, style, and vocabulary of the text, and Github Copilot was used for the development of the code.

Author contributions

Javier Kipen (Conceptualization [lead], Formal analysis [lead], Investigation [lead], Methodology [lead], Software [lead], Validation [lead], Writing—original draft [lead], Writing—review & editing [lead]), Matthew Beauregard Smith (Conceptualization [supporting], Investigation [supporting], Methodology [supporting], Software [supporting], Visualization [supporting], Writing—review & editing [equal]), Thomas Blom (Conceptualization

[supporting], Data curation [supporting], Formal analysis [supporting], Resources [equal], Software [supporting], Writing—review & editing [equal]), Sophia Bailing Zhou (Conceptualization [supporting], Formal analysis [supporting], Investigation [supporting], Software [supporting], Writing—review & editing [equal]), Edward M. Marcotte (Conceptualization [supporting], Funding acquisition [supporting], Resources [supporting], Supervision [supporting], Writing—review & editing [equal]), and Joakim Jaldén (Formal analysis [equal], Funding acquisition [lead], Supervision [equal], Validation [supporting], Writing—review & editing [equal])

Supplementary material

Supplementary material is available at *Bioinformatics Advances* online.

Conflicts of interest

M.B.S. and T.B. are affiliated with Erisyon, Inc., as employees or shareholders. E.M.M. is a co-founder and shareholder of Erisyon, Inc. and serves on the scientific advisory board.

Funding

E.M.M. acknowledges grant support from the Welch Foundation [F-1515] and U.S. National Institutes of Health [R35 GM122480]. Angela Bardo and Thomas Blom (from Erisyon Inc.) work was supported by the Cancer Prevention and Research Institute of Texas [DP250149]. This work was supported by the Swedish Research Council [VR Research Environment Grant 2018-06169; QuantumSense], and the Swedish Foundation for Strategic Research [SSF Grant ITM17-0049].

Data availability

The code and data utilized to produce all the results of this paper is at https://github.com/JavierKipen/ProtInfGPU.

References

Alfaro JA, Bohländer P, Dai M *et al.* The emerging landscape of single-molecule protein sequencing technologies. *Nat Methods* 2021;**18**:604–17.

Audain E, Uszkoreit J, Sachsenberg T *et al.* In-depth analysis of protein inference algorithms using multiple search engines and well-defined metrics. *J Proteomics* 2017;**150**:170–82.

Bachman JL, Wight CD, Bardo AM *et al.* Evaluating the effect of dye–dye interactions of xanthene-based fluorophores in the fluorosequencing of peptides. *Bioconjug Chem* 2022;**33**:1156–65.

Bollon J, Shortreed MR, Jeffery E *et al.* Isobayes: a Bayesian approach for single-isoform proteomics inference. *Bioinformatics* 2025;**41**:btaf450.

Brinkerhoff H, Kang AS, Liu J *et al.* Multiple rereads of single proteins at single-amino acid resolution using nanopores. *Science* 2021;**374**:1509–13.

Eren K, Taktakoğlu N, Pirim I. Dna sequencing methods: from past to present. *Eurasian J Med* 2022;**54**:47–56.

He Z, Huang T, Liu X *et al.* Protein inference: a protein quantification perspective. *Comput Biol Chem* 2016;**63**:21–9.

Huang T, Wang J, Yu W *et al.* Protein inference: a review. *Brief Bioinform* 2012;**13**:586–614.

Kipen J, Jaldén J. Beam search decoder for enhancing sequence decoding speed in single-molecule peptide sequencing data. *PLoS Comput Biol* 2023;**19**:e1011345.

Leijon A, Henter GE. *Pattern Recognition: Fundamental Theory and Exercise Problems*. Stockholm, Sweden: School of Electrical Engineering, KTH Royal Institute of Technology; 2012.

Mapes JH, Stover J, Stout HD *et al.* Robust and scalable single-molecule protein sequencing with fluorosequencing. bioRxiv, 2023, preprint: not peer reviewed.

Nesvizhskii AI, Keller A, Kolker E *et al.* A statistical model for identifying proteins by tandem mass spectrometry. *Anal Chem* 2003;**75**:4646–58.

Reed BD, Meyer MJ, Abramzon V *et al.* Real-time dynamic single-molecule protein sequencing on an integrated semiconductor device. *Science (1979)* 2022;**378**:186–92.

Restrepo-Pérez L, Joo C, Dekker C. Paving the way to single-molecule protein sequencing. *Nat Nanotechnol* 2018;**13**:786–96.

Smith MB, Simpson ZB, Marcotte EM. Amino acid sequence assignment from single molecule peptide sequencing data using a two-stage classifier. *PLoS Comput Biol* 2023;**19**:e1011157.

Smith MB, VanderVelden K, Blom T *et al.* Estimating error rates for single molecule protein sequencing experiments. *PLoS Comput Biol* 2024;**20**:e1012258.

Spivak M, Weston J, Tomazela D *et al.* Direct maximization of protein identifications from tandem mass spectra. *Mol Cell Proteomics* 2012;**11**:M111.012161.

Swaminathan J, Boulgakov AA, Marcotte EM. A theoretical justification for single molecule peptide sequencing. *PLoS Comput Biol* 2015;**11**:e1004080.

Swaminathan J, Boulgakov AA, Hernandez ET *et al.* Highly parallel single-molecule identification of proteins in zeptomole-scale mixtures. *Nat Biotechnol* 2018;**36**:1076–82.

The M, MacCoss MJ, Noble WS *et al.* Fast and accurate protein false discovery rates on large-scale proteomics data sets with percolator 3.0. *J Am Soc Mass Spectrom* 2016;**27**:1719–27.

Timp W, Timp G. Beyond mass spectrometry, the next step in proteomics. *Sci Adv* 2020;**6**:eaax8978.

Winkler R. Protyquant: comparing label-free shotgun proteomics datasets using accumulated peptide probabilities. *J Proteomics* 2021;**230**:103985.

Yu L, Kang X, Li F *et al.* Unidirectional single-file transport of full-length proteins through a nanopore. *Nat Biotechnol* 2023;**41**:1130–9.

Zhang S, Huang G, Versloot RCA *et al.* Bottom-up fabrication of a proteasome–nanopore that unravels and processes single proteins. *Nat Chem* 2021;**13**:1192–9.

Bioinformatics Advances, 2026, 6, 1–9
https://doi.org/10.1093/bioadv/vbag053
Original Article