Stochastic EM - A closely related cousin to MCEM

Wakeel Adekunle Kasali
wakeel.kasali@stat.ubc.ca
December 12, 2024

Abstract

In real-world genetic research, particularly in estimating gene frequencies, scientists often struggle with the challenge of incomplete data, where the true genotypes—the hidden threads of genetic identity—remain hard-to-catch and beyond direct observation. SEM is one of the possible open-ended computational methods to understanding this phenomenon. This study uses simulation-based approaches to comparing the behaviours of SEM under different conditions of its components. SEM is compared in its use with MCEM - a variant - to draw some commonalities. Contextual comparisons of the duo are presented to complement the limited simulations. For future research focus, SEM algorithm's performance for high-dimensional or complex missing data problems can be reflected-on for improvement under adaptive step size methods (e.g., Robbins-Monro sequences) as an extensive approach to the one implemented in this study.

1 Introduction

The intractability of the Expectation-Maximization (EM) algorithm arises primarily from challenges in computing the conditional expectations in the E-step, especially when faced with intractable conditional distributions, non-linear dependencies, non-smooth likelihoods, or high-dimensional latent variables. These complexities make exact computation of the E-step infeasible in practical applications, particularly in fields like genomics where missing data and latent variables are common [1].

The Monte Carlo EM (MCEM) algorithm addresses this challenge by replacing conditional expectations with Monte Carlo averages, effectively approximating the E-step. While this method reduces computational burdens, it introduces three primary issues, namely: (i) *determining the termination criterion*, (ii) *selecting the Monte Carlo sample size*, and (iii) *ensuring efficient simulation from the appropriate conditional distribution* [2].

To address the uncertainties in the convergence of MCEM, stochastic approximation algorithms provide an alternative approach. These iterative methods solve the same core problem but use a different update formula. Instead of fitting neatly into the EM framework, stochastic approximation generates a Monte Carlo sample at each iteration, approximating the E-step in a computationally efficient manner.

Stochastic Expectation-Maximization (SEM), as a specific instance of stochastic approximation, allows for the iterative approximation of expectations through sampling. By focusing on subsets of latent variables or imputed missing data points, SEM reduces computational demands while maintaining convergence properties.

This project examines the Stochastic Expectation-Maximization (SEM) algorithm as a closely related counterpart to the Monte Carlo Expectation-Maximization (MCEM) method. The focus is on highlighting SEM's practical applications and its connection to the classical EM method and MCEM. Pseudocode and examples are provided to clarify the stochastic E-step and iterative parameter updates in computationally intensive scenarios. Essential components, such as *step size*, *sampling rates*, and the *fraction of missing information*, are analyzed to understand their influence on convergence and stability using simulations. The computational trade-offs and advantages of SEM over MCEM are also explored, particularly in applications involving missing data.

2 Classical EM algorithm

Let $\mathbf{Y}$ represent the observed data and $\mathbf{X}$ the missing data, where $\mathbf{X}$ serves as a conceptual tool to analyze the complete data distribution. The joint distribution of $(\mathbf{Y}, \mathbf{X})$ is referred to as the "complete data distribution" with density f_c. The function f denotes the marginal density of $\mathbf{Y}$, while f_m represents the conditional density of $\mathbf{X}$ given $\mathbf{Y}$, commonly referred to as the "missing data distribution." Notably, f_m corresponds to the conditional distribution $f_m(\mathbf{X} \mid \mathbf{Y})$, rather than the marginal distribution of $\mathbf{X}$.

The log-likelihoods ℓ, ℓ_c, ℓ_m correspond to the observed, complete, and missing data distributions, respectively, with associated score vectors $\mathbf{S}, \mathbf{S}_c, \mathbf{S}_m$ and observed information matrices $\mathbf{I}, \mathbf{I}_c, \mathbf{I}_m$. These are parameterized by $\boldsymbol{\theta} \in \Theta \subseteq R^p$.

The Expectation-Maximization (EM) algorithm iteratively alternates between two steps. The first step, known as the "E-step," computes the conditional expectation of

the complete data log-likelihood:

$$Q(\theta \mid \theta_{k-1}) = E_{\theta_{k-1}} \left[\ell_c(\theta; \mathbf{Y}, \mathbf{X}) \mid \mathbf{Y} = \mathbf{y} \right], \tag{1}$$

where $\boldsymbol{\theta}_{k-1}$ is the parameter estimate from the previous iteration. The M-step maximizes this expectation in Equation 1 as

$$\theta_k = \arg\max_{\theta} Q(\theta \mid \theta_{k-1}),$$

yielding the updated parameter estimate $\boldsymbol{\theta}_k$. This iterative process continues until convergence.

The ascent property is fundamental to EM convergence, ensuring that each iteration does not decrease the observed data likelihood, $\ell(\theta; \mathbf{Y})$. Additionally, under regularity conditions, the EM algorithm facilitates the computation of the score vector $S(\theta; \mathbf{Y})$ and the observed information matrix $I(\theta)$ of the observed data likelihood $\ell(\theta; \mathbf{Y})$ directly from the complete data quantities.

3. MCEM and SEM

The Stochastic Expectation-Maximization (SEM) algorithm extends the traditional Expectation-Maximization (EM) framework by incorporating stochasticity into the iterative process. This adjustment addresses cases where exact computation of the expected complete-data log-likelihood is computationally infeasible due to the complexity of the underlying data or model.

In the standard EM algorithm, each iteration alternates between two steps. The **E-step** computes the expected value of the complete-data log-likelihood, given the observed data Y and the current parameter estimate $\theta^{(k)}$:

$$Q(\theta \mid \theta^{(k)}) = E_{X|Y,\theta^{(k)}} \left[\ell_c(\theta; \mathbf{Y}, \mathbf{X}) \right].$$

The **M-step** then maximizes this expectation with respect to the parameter θ. However, the computational demands of the E-step can become prohibitive in high-dimensional or complex models, as calculating the full expectation often requires integration over intractable conditional distributions.

The Monte Carlo Expectation-Maximization (MCEM) algorithm refines the E-step of the EM framework by replacing the exact expectation with a Monte Carlo approximation [3]. At each iteration, the MCEM generates samples from the conditional distribution of the missing data, given the observed data and the current parameter estimate [4].

For instance, in a simplified version of the Monte Carlo EM (MCEM) algorithm, the E-step may involve drawing multiple independent samples $X_1, \ldots, X_m$ from $f_m(X \mid Y; \theta^{(k)})$ and estimating the expectation by averaging over the corresponding complete-data log-likelihoods. This leads to an approximate E-step of the form:

$$Q(\theta \mid \theta^{(k)}) \approx \frac{1}{m} \sum_{j=1}^{m} \ell_c(\theta; \mathbf{Y}, \mathbf{X_j}), \tag{2}$$

where X_j represents the j-th sample from the conditional distribution.

However, SEM algorithm modifies the E-step by replacing the exact expectation with a stochastic approximation. Instead of calculating $Q(\theta \mid \theta^{(k)})$ directly, SEM samples from

the conditional distribution $P(\mathbf{X} \mid \mathbf{Y}, \theta^{(k)})$. Let the sampled value be $X^{(k)}$, the algorithm then uses $\ell_c(\theta; \mathbf{Y}, \mathbf{X^{(k)}})$ as a proxy for the complete-data log-likelihood.

When $m = 1$ in Equation (2), the SEM algorithm relies on a single sample per iteration, further reducing computational complexity at the cost of increased variability. Despite this variability, SEM retains the key property of iteratively improving the observed data likelihood under appropriate regularity conditions.

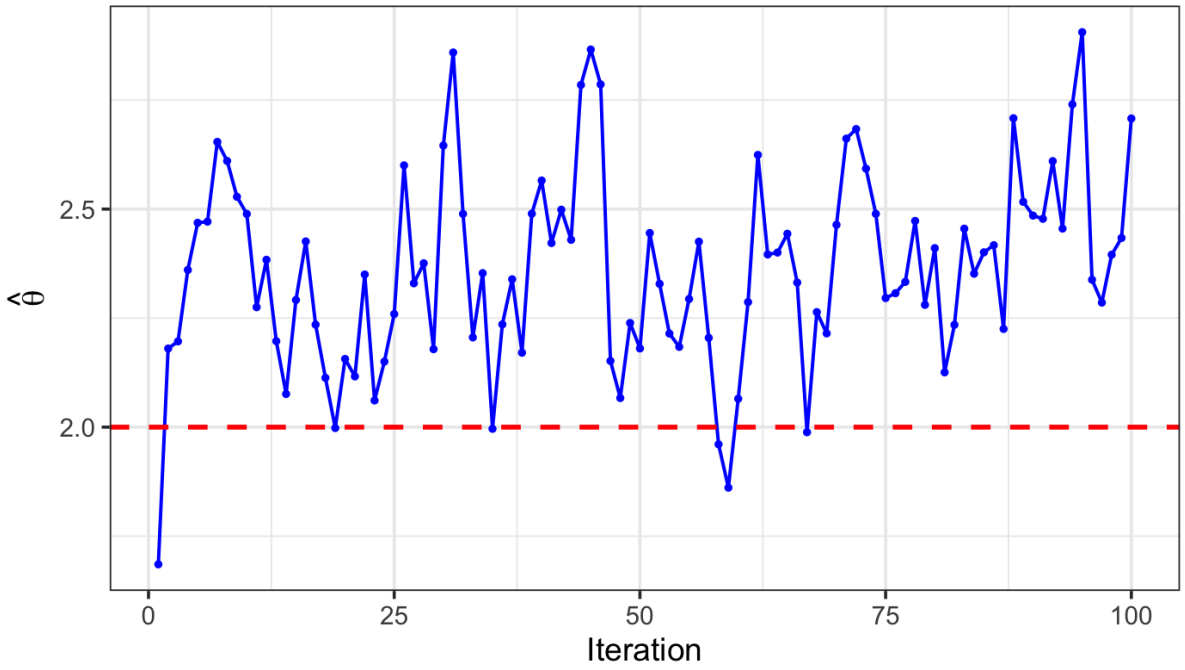

Figure 1: Convergence of SEM Algorithm: Fluctuating estimates due to stochasticity around the true value over iterations.

This approach introduces randomness into the parameter updates, forming a Markov chain that converges to the maximum likelihood estimator (MLE) under suitable regularity conditions. By focusing on a single sample at each iteration, SEM differs from the Monte Carlo EM (MCEM) algorithm, which typically relies on averaging over multiple Monte Carlo samples to approximate the expectation in the E-step. While MCEM provides a more precise estimate (See Figure 2), it often requires greater computational resources. SEM, by contrast, trades off precision (See Figure 1) for computational efficiency, making it especially useful in scenarios where resources are limited or the model is highly complex.

In cases where the complete data likelihood is from an exponential family of distributions, specific assumptions simplify the convergence analysis of MCEM and SEM [4]. Let $f(x; \theta)$ denote the complete data likelihood and $p(x \mid \theta)$ the conditional density of the missing data. The first key assumption is that Θ is an open subset of R^{d_θ}, and the family $\{f(\cdot; \theta)\}_{\theta \in \Theta}$ is an exponential family of positive functions on $\mathcal{X}$, such that:

$$f(x; \theta) = \exp\left[\psi^\top(\theta) S(x) - c(\theta)\right] h(x),$$

where $\psi : R^{d_\theta} \to R^{d_s}$, $S : \mathcal{X} \to R^{d_s}$, $c : \Theta \to R$, and $h : \mathcal{X} \to R_+$. This structure facilitates the computation of sufficient statistics $S(x)$, which are critical in approximating

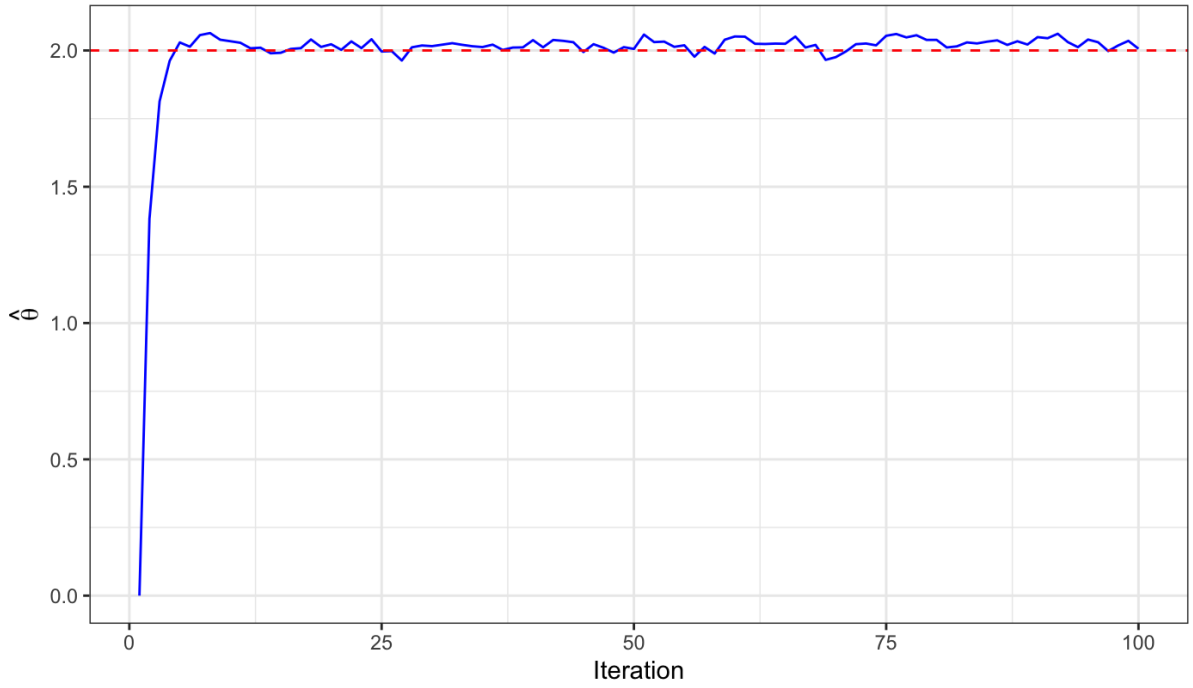

Figure 2: Convergence of the MCEM Algorithm: Parameter estimates approach the true value with increasing iterations, demonstrating stability and rapid convergence.

the E-step expectations. Furthermore, the function $L(\theta)$, defined as the integral of the complete data likelihood, is assumed to be positive and continuous over Θ, ensuring well-defined parameter updates during iterations.

Another essential assumption is that the expected sufficient statistic $\bar{S}(\theta)$, given by:

$$\bar{S}(\theta) = \int S(x)p(x \mid \theta)\lambda(dx),$$

is finite and continuous over Θ. Additionally, there exists a subset $S \subset R^{d_s}$ that contains the convex hull of $S(x)$. For any $s \in S$, the mapping $\theta \mapsto \psi^{\top}(\theta)s - c(\theta)$ has a unique global maximum $\hat{\theta}(s) \in \Theta$. These assumptions guarantee that the optimization procedure in both MCEM and SEM converges under suitable conditions, providing a solid theoretical foundation for their use in applications involving missing data .

4. SEM approach and theoretical justification

SEM algorithm is a vital extension of the traditional EM framework, introducing a stochastic component to the iterative process. SEM is beneficial when it is infeasible to compute the complete data log-likelihood directly because of data complexity. Its fundamental method is to approximate the expectation in the E-step via random sampling. SEM provides a computationally efficient method of approximating expectations by generating stochastic realizations rather than integrating across the whole space of missing data. Because the amount of missing information greatly influences convergence

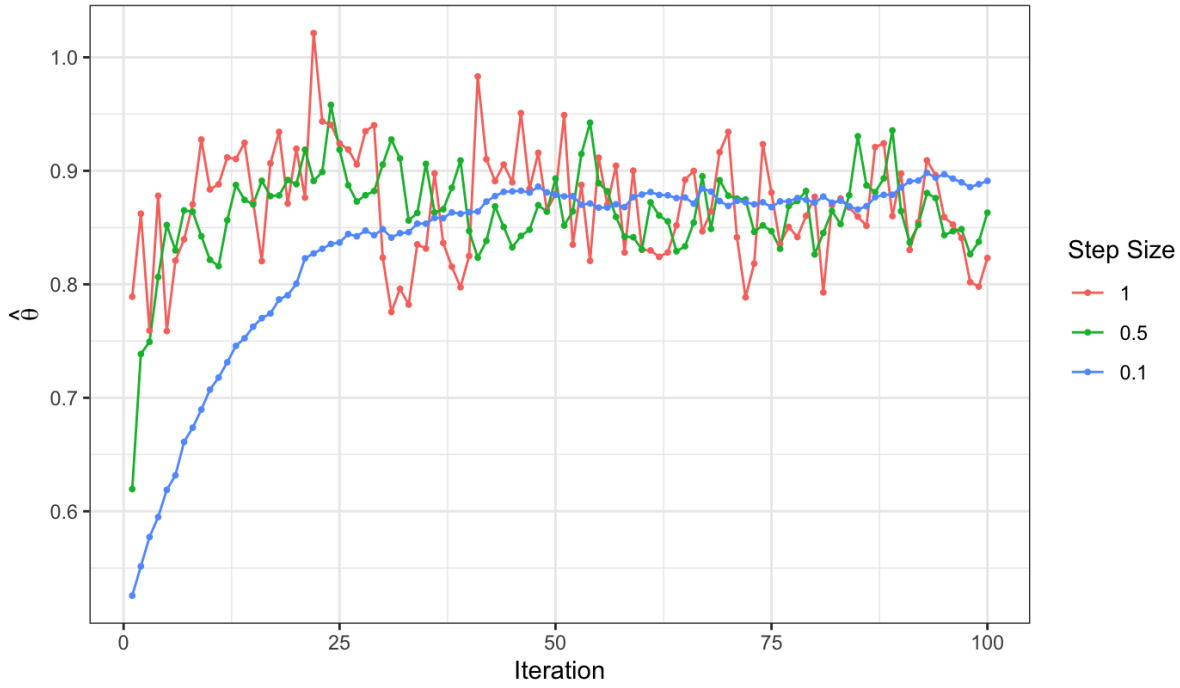

Figure 3: Convergence of the MCEM Algorithm: Parameter estimates approach the true value with increasing iterations, demonstrating stability and rapid convergence.

behaviour and computing efficiency, this stochastic adjustment makes applying SEM to issues involving missing data easier.

The theoretical justification of SEM stems from its convergence to the MLE under mild regularity conditions. This is achieved through its connection to the Robbins-Monro stochastic approximation framework, which ensures that the algorithm iteratively refines parameter estimates by approximating the roots of a mean-field function. Specifically, the SEM algorithm can be analyzed in terms of its updates being noisy observations of the expected complete-data sufficient statistics, which converge to a fixed point corresponding to the MLE. This convergence is guaranteed as long as the step sizes, γ_i, satisfy conditions such as

$$\sum \gamma_i = \infty \quad \text{and} \quad \sum \gamma_i^2 < \infty.$$

Important to the SEM approach is the control of **step size** (γ_i), which influences the stability and rate of convergence. Larger step sizes improve initial exploration of the parameter space but may introduce excess noise, while smaller step sizes enhance precision but risk premature convergence (See Figure 3 . To balance these trade-offs, the choice of a slowly decreasing sequence, such as

$$\gamma_i = i^{-0.6},$$

is often recommended, as it ensures both efficient convergence and stability. The theoretical results, supported by Lyapunov functions [4], demonstrate that SEM inherits the ascent property of EM, where each iteration guarantees a non-decreasing likelihood value.

Sampling rates, or the number of Monte Carlo iterations used in each E-step, are another critical factor in the SEM framework. Increasing the sampling rate improves the

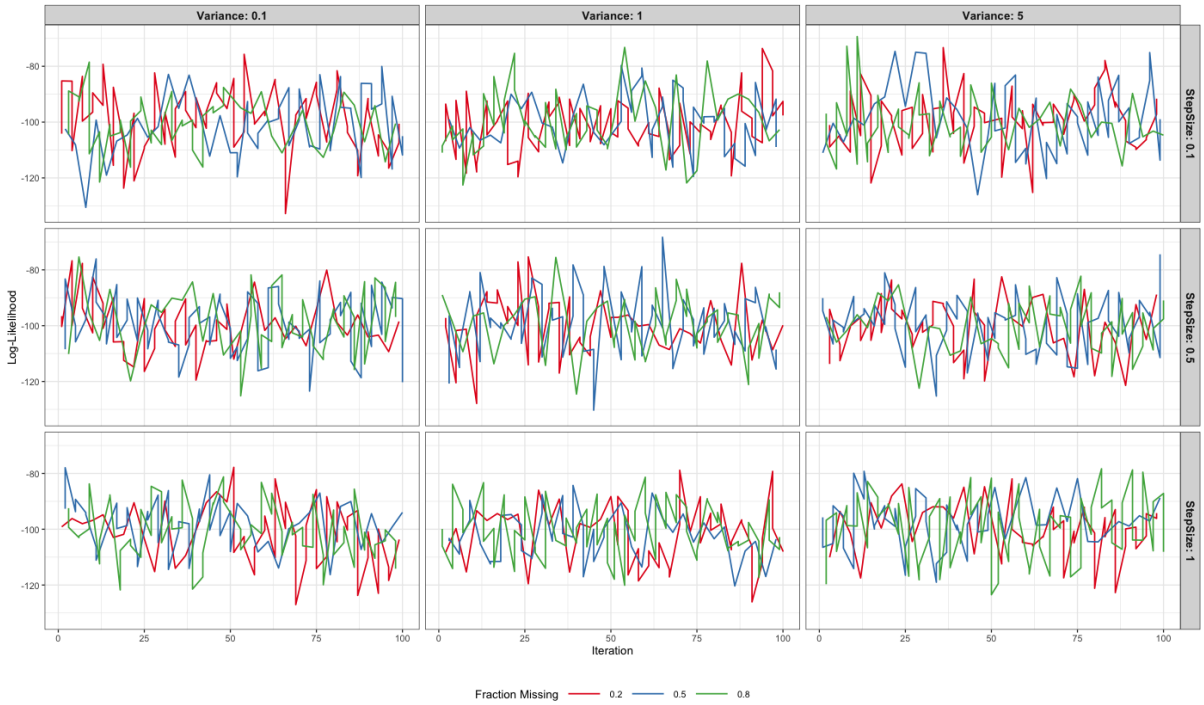

Figure 4: Log likelihood convergence for varying step sizes, variances and varying fraction of missing information.

approximation of the expectation in the E-step, thereby reducing bias in parameter estimates. However, this comes at the expense of increased computational cost. Theoretical justifications for SEM suggest that even with a moderate sampling rate, the algorithm converges effectively due to its reliance on ergodic Markov chains [5]. This property ensures that the sampled values progressively represent the true distribution of the missing data as the number of iterations increases.

Moreover, the **fraction of missing information**, denoted by $I_m(\theta)/I_o(\theta)$, is a critical factor for SEM's performance. Theoretical studies highlight that a higher fraction of missing information necessitates a larger number of simulations to achieve the same level of precision in parameter estimates. This relationship arises because missing information increases uncertainty in the parameter updates, requiring additional computational effort to approximate the expectation in the E-step accurately. Through simulations, it has been demonstrated that SEM maintains its efficiency even in high-missing-information scenarios by leveraging its stochastic nature to balance computational demands and precision.

In 4 combining the dimensions including variance of the estimators, it could be observed that high variance and large fractions of missing information have a compounding destabilizing effect on the convergence process, especially when coupled with larger step sizes. For instance, in the bottom-right subplot (step size = 1, variance = 5), the trajectories are the most erratic, highlighting the need for careful calibration of step size and variance under such conditions.

Theoretically, SEM is supported by strong asymptotic guarantees. The convergence of SEM to the MLE is rooted in the properties of Markov chains used in the E-step

sampling process. As the number of iterations grows, the Markov chain transitions toward its stationary distribution, ensuring that the generated samples accurately reflect the missing data accurate distribution. This foundation allows SEM to exhibit consistency and asymptotic normality in parameter estimation, with well-characterised variance terms under regularity conditions. These theoretical properties position SEM as a reliable algorithm for solving complex statistical inference problems.

5. Contextual comparison between MCEM and SEM

The SEM algorithm surpasses MCEM in computational efficiency by using a single sample in its stochastic E-step instead of multiple samples. MCEM averages multiple Monte Carlo samples, whereas SEM updates parameter estimates using a single sampled realization. Hence, SEM's reduced computational demand makes it ideal for resource-constrained scenarios, simplifying workflows in complex models while maintaining convergence under regularity conditions..

In contexts involving complex latent structures or high-dimensional parameter spaces, SEM often outperforms MCEM in efficiency [6]. For instance, when working with hierarchical models or non-linear dependencies where exact integration in the E-step is infeasible, SEM's single-sample approach updates parameters effectively without the computational overhead of averaging. This makes SEM an ideal candidate for applications in genomics, finance, or machine learning, where rapid iterations and scalability are critical. Furthermore, the Markovian nature of SEM's updates ensures that it can handle missing data scenarios with fewer assumptions.

However, SEM's stochastic nature leads to higher variability in parameter updates compared to MCEM. The single-sample approach of SEM reduces computational demand but increases noise, leading to slower and less stable convergence, especially if the step size (γ_i) is not well-tuned. Mitigating this SEM's variability involves hyperparameter tuning, such as step size adjustment (e.g., $\gamma_i = i^{-0.6}$) or adaptive sampling, to enhance convergence stability.

Moreover, SEM is sensitive to initialization and sampling strategy, where poor choices can amplify variability and lead to suboptimal convergence. In contrast, MCEM reduces instability by averaging multiple samples. Practical strategies to address these challenges in SEM include post-convergence averaging, where the final parameter estimates are smoothed over several iterations. Additionally, combining SEM with other Monte Carlo methods, such as importance sampling, can enhance the efficiency of the stochastic E-step.

From a **theoretical perspective**, SEM and MCEM share many similarities in their stochastic approximation frameworks, but they differ significantly in their trade-offs between computational efficiency and precision. While MCEM provides more precise approximations at each iteration by averaging multiple samples, SEM focuses on computational feasibility, making it better suited for large-scale applications. The choice between SEM and MCEM ultimately depends on the model complexity and available computational resources. In scenarios where exact computation is infeasible but approximate solutions are acceptable, SEM stands out as a practical alternative.

In summary, the **contextual comparison between SEM and MCEM** reveals a balanced view of their strengths and limitations. SEM's low computational demand suits resource-constrained settings, but its variability and slower convergence require careful

tuning and adaptive techniques. However, its stochastic nature introduces variability and slower convergence rates, which require thoughtful tuning and adaptive techniques to address. By linking these theoretical insights to practical implications, this comparison reflects SEM's role as a computationally efficient yet flexible alternative to MCEM, particularly for complex latent variable models that challenge traditional EM approaches.

6. Conclusion

Compared to the Monte Carlo EM (MCEM) technique, the SEM approach presents significant computational trade-offs. SEM's stochastic E-step simplifies this process by depending on one or a few simulations per iteration. In contrast, MCEM employs full integration via Monte Carlo simulations in each iteration.

Metrics of convergence in SEM typically increase for a properly chosen step size. For a reliable parameter estimation, moderate sampling rates are often enough to sustain that. In the context of missing data, when fractions of missing information is high, this causes increament in the uncertainty. Additionally, the variance usually persist across iterations but does not diminish the accuracy of the estimates in capturing the true parameter. These flexibilities in properties enable SEM to be adjustable for a better a system particularly in applications involving missing data.

References

[1] Y. L. Qiu, H. Zheng, and O. Gevaert, "Genomic data imputation with variational auto-encoders," *GigaScience*, vol. 9, no. 8, p. giaa082, 2020.

[2] W. Ruth, "A review of monte carlo-based versions of the em algorithm," 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:266725382

[3] G. Fort and É. Moulines, "Convergence of the monte carlo expectation maximization for curved exponential families," *Annals of Statistics*, vol. 31, pp. 1220–1259, 2003. [Online]. Available: https://api.semanticscholar.org/CorpusID:121911625

[4] O. Cappé, E. Moulines, T. Rydén, S. N. S. Mathematics, S. eBooks 2005 English International, and E. Central, "Inference in hidden markov models."

[5] S. Nielsen, "The stochastic em algorithm: estimation and asymptotic results," *Bernoulli*, vol. 6, pp. 457–489, 2000. [Online]. Available: https://api.semanticscholar.org/CorpusID:15921077

[6] C. Gilles and D. Jean, "On stochastic versions of the em algorithm," 1995.

Appendix

```
    # SEM - Figure 1

# A simple censored data problem

```{r}
set.seed(123)

n <- 100
theta.true <- 2
censor.pt <- 1.5  # Censoring threshold

X <- rexp(n, rate = 1/theta.true)
censored <- X > censor.pt       # Identify censored values
Y <- ifelse(censored,
            censor.pt, X)  # Observed data (censored values truncated)
```


Stochastic E-step

```{r}
stochastic.EStep <- function(Y,
                             censored, theta) {
  # Simulate from the conditional distribution
  Xsim <- Y
  Xsim[censored] <- rexp(sum(censored),
                         rate = 1/theta) + censor.pt
  return(Xsim)
}
```

Stochastic M-step

```{r}
Mstep <- function(Xsim) {
  # Maximum likelihood estimator for exponential mean
  theta.new <- mean(Xsim)
  return(theta.new)
}
```



```{r}
# Stochastic EM Algorithm
stochastic.EM <- function(Y, censored,
```
```

```
                          theta.init,
                          max.iter = 100, tol = 1e-6) {
  theta <- theta.init
  theta.history <- numeric(max.iter)

  for (iter in 1:max.iter) {
    # E-step: Simulate missing data
    Xsim <- stochastic.EStep(Y, censored, theta)

    # M-step: Update parameter estimate
    theta.new <- Mstep(Xsim)

    theta.history[iter] <- theta.new

    # Check convergence
    if (abs(theta.new - theta) < tol) {
      message("Convergence achieved after ", iter, " iterations.")
      theta.history <- theta.history[1:iter]  # Trim history
      break
    }

    # Update parameter
    theta <- theta.new
  }

  return(list(theta.est = theta.new,
              theta.history = theta.history))
}
```



```{r}
# Initial parameter guess
theta.init <- 1.0


result <- stochastic.EM(Y, censored, theta.init)

theta.est <- result$theta.est
theta.history <- result$theta.history

theta.data <- data.frame(
  Iteration = 1:length(theta.history),
  ParamEstimate = theta.history
)

sem_plot <- ggplot(theta.data, aes(x = Iteration,
```

```
                                    y = ParamEstimate)) +
  geom_line(color = "blue",
            linewidth = 0.7) +
  geom_point(color = "blue",
             size = 1) +
  geom_hline(yintercept = theta.true,
             color = "red",
             linetype = "dashed", size = 1) +
  labs(
    title = "",
    x = "Iteration",
    y = expression(hat(theta))
  ) +
  theme_bw(base_size = 14)
```

 # MCEM: Figure 2

```{r}
# Define the log-likelihood function
logLikelihood <- function(theta, data) {
  sum(dnorm(data,
            mean = theta,
            sd = 1, log = TRUE))
}
```

```{r}
# Generate incomplete data for the simulation
set.seed(42)
n <- 100  # Number of data points
theta.true <- 2
data <- rnorm(n,
              mean = theta.true,
              sd = 1)

```

```{r}
# Introduce missingness
missing.fraction <- 0.3
missing.indices <- sample(1:n,
                          size = round(missing.fraction * n))
observed.data <- data[-missing.indices]
```

```
# MCEM Implementation
MCEM <- function(observed.data,
                 theta.init,
                 max.iter,
                 mSimulations) {
  theta.est <- numeric(max.iter)
  theta.est[1] <- theta.init

  for (k in 2:max.iter) {
    # Monte Carlo E-step
    simulatedData <- replicate(mSimulations, {
      missingData <- rnorm(length(missing.indices),
                           mean = theta.est[k - 1],
                           sd = 1)
      complete_data <- c(observed.data, missingData)
      complete_data
    })


    simulatedData <- t(simulatedData)

    # Approximate Q-function
    Q_theta <- apply(simulatedData,
                     1,
                     logLikelihood,
                     theta = theta.est[k - 1])

    # M-step
    theta.est[k] <- mean(apply(simulatedData, 1, mean))
  }

  return(theta.est)
}
```


```{r}
theta.init <- 0
max.iter <- 100
mSimulations <- 10


theta.mcem <- MCEM(observed.data,
                   theta.init,
                   max.iter,
                   mSimulations)

iterations <- 1:max.iter
```

```
mcem.results <- data.frame(Iteration = iterations,
                           Estimate = theta.mcem)

MCEM <- ggplot(mcem_results, aes(x = Iteration,
                         y = Estimate)) +
  geom_line(color = "blue") +
  geom_hline(yintercept = theta.true,
             linetype = "dashed", color = "red") +
  labs(
    title = "",
    x = "Iteration",
    y = expression(hat(theta))
  ) +
  theme_bw()
```

 # Step size - Figure 3

```{r}
# Stochastic EM function
stochasticEM <- function(Y,
                         censored,
                         theta.init,
                         max.iter = 100, step_size = 1) {
  theta <- theta.init
  theta.history <- numeric(max.iter)

  for (iter in 1:max.iter) {
    # Stochastic E-step
    simulatedData <- ifelse(censored,
                            rexp(length(Y),
                                 rate = 1/theta), Y)

    # M-step
    theta.new <- mean(simulatedData)

    # Including step size
    theta <- theta + step_size * (theta.new - theta)

    # Store theta in history
    theta.history[iter] <- theta
  }

  return(list(theta.final = theta,
              theta.history = theta.history))
}
```

```{r}
simulatStepSizes <- function(stepSizes,
                             Y, censored,
                             theta.init, max.iter = 100) {
  results <- lapply(stepSizes, function(size) {
    res <- stochasticEM(Y,
                        censored,
                        theta.init,
                        max.iter,
                        step_size = size)
    data.frame(
      Iteration = 1:max.iter,
      Theta = res$theta.history,
      StepSize = factor(size)
    )
  })

  # Combine results into a single data frame
  results_df <- bind_rows(results)
  return(results_df)
}
```

```{r}
set.seed(123)
n <- 100
true_theta <- 1.5
Y <- rexp(n, rate = 1/true_theta)
censored <- Y > 2
Y[censored] <- 2
theta.init <- 0.5
stepSizes <- c(1, 0.5, 0.1)


results_df <- simulatStepSizes(stepSizes,
                               Y, censored,
                               theta.init,
                               max.iter = 100)

```

```{r}
ggplot(results_df, aes(x = Iteration,
                       y = Theta,
                       color = StepSize,
```

```
                        group = StepSize)) +
  geom_line() +
  geom_point(size = 0.7) +
  labs(
    title = "",
    x = "Iteration",
    y = expression(hat(theta)),
    color = "Step Size"
  ) +
  theme_bw() +
  theme(legend.position = "right")
```

 # SEM components - Figure 4

```{r}
library(ggplot2)
#library(dplyr)
#library(tidyr)

df <- data.frame(
  Iteration = rep(1:100, 15),
  LogLikelihood = rnorm(1500,
                        mean = -100,
                        sd = 10),
  StepSize = rep(c(0.1, 0.5, 1), each = 500),
  Variance = rep(c(0.1, 1, 5), times = 100),
  FractionMissing = rep(c(0.2, 0.5, 0.8), each = 50)
)

```



```{r}
df <- df %>%
  mutate(StepSize = as.factor(StepSize),
         Variance = as.factor(Variance),
         FractionMissing = as.factor(FractionMissing))


ggplot(df, aes(x = Iteration,
               y = LogLikelihood,
               color = FractionMissing,
               group = interaction(FractionMissing,
                                   Variance))) +
  geom_line(size = 0.3) +
  facet_grid(StepSize ~ Variance,
             labeller = label_both) +
```

```
  scale_color_brewer(palette = "Set1") +
  labs(
    title = "",
    x = "Iteration",
    y = "Log-Likelihood",
    color = "Fraction Missing"
  ) +
  theme_bw(base_size = 5) +
  theme(
    strip.text = element_text(size = 5, face = "bold"),
    legend.position = "bottom"
  )
```