

Special Topics in Applied Mathematics I

Solution 1

XIA JIAHAN

January 21, 2026

Question 1

Iterative eigenvalue methods: an intuitive story

Think of a matrix A as a machine that acts on vectors: it can rotate, shear, and stretch them. Most directions are changed in complicated ways, but some special directions come back pointing the same way after applying A . Those directions are *eigenvectors* $v \neq 0$ satisfying

$$Av = \lambda v,$$

where λ is the corresponding *eigenvalue* (the factor by which A stretches v). When matrices are large, computing all eigenvalues exactly (e.g., from a characteristic polynomial) is impractical and numerically fragile. Iterative methods instead exploit one operation we can often do efficiently: repeatedly multiply by A (or solve linear systems involving A).

Power iteration: repeated multiplication reveals the dominant direction

The simplest method is *power iteration*. Choose any nonzero starting vector b_0 and repeat:

$$\begin{aligned} c_{k+1} &= Ab_k, \\ b_{k+1} &= \frac{c_{k+1}}{\|c_{k+1}\|}. \end{aligned}$$

Normalization “resets the length” so we can watch the direction. The method tends to align b_k with the eigenvector associated with the eigenvalue of largest magnitude.

Why it converges (qualitative). Assume A is diagonalizable with eigenpairs (λ_i, v_i) and $|\lambda_1| > |\lambda_2| \geq \dots$. Write the starting vector as a combination of eigenvectors:

$$b_0 = \sum_{i=1}^n \alpha_i v_i \quad (\alpha_1 \neq 0).$$

Then

$$A^k b_0 = \sum_{i=1}^n \alpha_i \lambda_i^k v_i = \lambda_1^k \left(\alpha_1 v_1 + \sum_{i=2}^n \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^k v_i \right).$$

As k grows, the ratios $\left(\frac{\lambda_i}{\lambda_1} \right)^k$ shrink toward 0, so the direction of $A^k b_0$ becomes dominated by v_1 . The convergence speed is governed by the spectral gap; a common rule of thumb is that the directional error shrinks like

$$\left| \frac{\lambda_2}{\lambda_1} \right|^k.$$

Estimating the eigenvalue: Rayleigh quotient

Once we have an approximate eigenvector x , a natural estimate of its eigenvalue is the *Rayleigh quotient*

$$R(x) = \frac{x^\top A x}{x^\top x}.$$

If x is exactly an eigenvector v , then $R(v) = \lambda$ because $v^\top A v = v^\top (\lambda v) = \lambda v^\top v$. When x is close to an eigenvector, $R(x)$ typically gives a good scalar approximation of the associated eigenvalue, so one can monitor the sequence $R(b_k)$ during power iteration.

Example: Consider a diagonal matrix with an irrational eigenvalue:

$$M = \begin{pmatrix} \pi & 0 \\ 0 & 1 \end{pmatrix}, \quad \lambda_1 = \pi \approx 3.14159.$$

Let's estimate λ_1 using power iteration starting with $x_0 = (1, 1)^\top$. The Rayleigh quotient $R(x_k)$ provides the estimate at each step.

- **Step 0:** $x_0 = (1, 1)^\top$.

$$R(x_0) = \frac{1 \cdot \pi + 1 \cdot 1}{1 + 1} = \frac{\pi + 1}{2} \approx 2.0708.$$

- **Step 1:** $x_1 = Mx_0 = (\pi, 1)^\top$.

$$R(x_1) = \frac{\pi^3 + 1}{\pi^2 + 1} \approx \frac{31.006 + 1}{9.870 + 1} \approx 2.9446.$$

- **Step 2:** $x_2 = Mx_1 = (\pi^2, 1)^\top$.

$$R(x_2) = \frac{\pi^5 + 1}{\pi^4 + 1} \approx \frac{306.02 + 1}{97.41 + 1} \approx 3.1198.$$

- **Step 3:** $x_3 = Mx_2 = (\pi^3, 1)^\top$.

$$R(x_3) = \frac{\pi^7 + 1}{\pi^6 + 1} \approx \frac{3020.29 + 1}{961.39 + 1} \approx 3.1394.$$

After three steps, the estimate 3.1394 is very close to $\pi \approx 3.1416$.

Question 2

Mathematical Foundation of Monte Carlo Integration

Monte Carlo (MC) methods provide a stochastic framework for approximating high-dimensional integrals that are analytically intractable. Consider the problem of evaluating the integral of a function $f : \Omega \rightarrow \mathbb{R}$ over a domain $\Omega \subset \mathbb{R}^d$:

$$I = \int_{\Omega} f(x) dx.$$

By introducing a probability density function $p(x)$ that is strictly positive on Ω , we can reformulate I as an expectation:

$$I = \int_{\Omega} \frac{f(x)}{p(x)} p(x) dx = \mathbb{E} \left[\frac{f(X)}{p(X)} \right],$$

where X is a random variable with density p . This transformation allows the use of statistical sampling to estimate the value of the integral.

The Estimator and its Statistical Properties

Given N independent and identically distributed (i.i.d.) samples $\{X_i\}_{i=1}^N$ drawn from $p(x)$, the Monte Carlo estimator \hat{I}_N is defined as the sample mean:

$$\hat{I}_N = \frac{1}{N} \sum_{i=1}^N \frac{f(X_i)}{p(X_i)}.$$

The estimator is *unbiased*, meaning $\mathbb{E}[\hat{I}_N] = I$. Its convergence is guaranteed by the Law of Large Numbers (LLN), while the Central Limit Theorem (CLT) provides the asymptotic distribution of the error:

$$\sqrt{N}(\hat{I}_N - I) \xrightarrow{d} \mathcal{N}(0, \sigma^2), \quad \text{where } \sigma^2 = \text{Var} \left(\frac{f(X)}{p(X)} \right).$$

Consequently, the probabilistic error bound is $O(\sigma/\sqrt{N})$. This rate is independent of the dimension d , which is the primary advantage of Monte Carlo methods over deterministic quadrature rules (e.g., Simpson's rule), where the error scales as $O(N^{-k/d})$ and suffers from the “curse of dimensionality.”

The Necessity of Variance Reduction

The $O(N^{-1/2})$ convergence rate implies that to gain one additional decimal digit of accuracy (a 10-fold error reduction), the sample size N must be increased by a factor of 100. For complex simulations where each function evaluation is computationally expensive, this requirement is often prohibitive.

Variance reduction techniques are essential to improve computational efficiency by decreasing the constant σ in the error term without altering the expectation I . By strategically choosing $p(x)$ (Importance Sampling), exploiting correlations (Antithetic Variates), or utilizing auxiliary information (Control Variates), one can achieve the desired precision with significantly fewer samples, effectively making the stochastic approach viable for high-precision scientific computing.

Numerical Implementation: Control Variates

To demonstrate the efficiency of variance reduction, we implemented a Monte Carlo simulation in Python. The script estimates the integral of $f(x) = e^x$ on $[0, 1]$ using $g(x) = 1 + x$ as a control variate.

Listing 1: Monte Carlo with Control Variates Implementation

```

1 import numpy as np
2
3 def f(x):
4     return np.exp(x)
5
6 def g(x):
7     return 1 + x
8
9 def main():
10    np.random.seed(42)
11    N = 10000
12    U = np.random.rand(N)
13    Y = f(U)
14    C = g(U)
15    mu_c = 1.5
16
17    # Standard Monte Carlo
18    mc_estimate = np.mean(Y)
19    mc_variance = np.var(Y, ddof=1) / N
20
21    # Control Variates
22    cov_matrix = np.cov(Y, C)
23    c_star = cov_matrix[0, 1] / cov_matrix[1, 1]
24    Y_cv = Y - c_star * (C - mu_c)
25    cv_estimate = np.mean(Y_cv)
26    cv_variance = np.var(Y_cv, ddof=1) / N
27
28    print(f"Standard Monte Carlo Estimate: {mc_estimate:.6f}")
29    print(f"Standard MC Variance: {mc_variance:.10f}")
30    print("-" * 30)
31    print(f"Control Variates Estimate: {cv_estimate:.6f}")
32    print(f"Control Variates Variance: {cv_variance:.10f}")

```

```
33     print(f"Variance Reduction Factor: {mc_variance / cv_variance:.2f}x")
34
35 if __name__ == "__main__":
36     main()
```

Simulation Output:

```
Standard Monte Carlo Estimate: 1.707932
Standard MC Variance: 0.0000239217
-----
Control Variates Estimate: 1.717782
Control Variates Variance: 0.0000003921
Variance Reduction Factor: 61.02x
```

The results show a variance reduction of approximately 61x, meaning the control variate estimator achieves the same precision with 61 times fewer samples than the standard approach.

A Complete Conversation Log

This appendix contains the full record of the interaction and development process for this solution.

USER: in question2, d:\file\MATH4665\HW1\Sol1.tex, describe the algorithms of Monte Carlo precisely, you should state when the algorithms is used and how, also state why the variance reduction is needed?

ASSISTANT: [Added comprehensive Monte Carlo description including definitions, use cases, and the theoretical need for variance reduction.]

USER: too many subpoints, you should reorganize it as a professional math report, state the MC in math clearly.

ASSISTANT: [Refactored the Monte Carlo section into a formal mathematical report format, emphasizing the Law of Large Numbers, Central Limit Theorem, and the $O(N^{-1/2})$ convergence rate.]

USER: write a python program to simulate a simple estimator and compare sample variance with and without control variates, run your code in virtual env.

ASSISTANT: [Created a virtual environment, installed dependencies, and developed mc_control_variates.py to simulate e^x integration. Demonstrated a 61.02x variance reduction using control variates.]

USER: add the original algorithms code to the d:\file\MATH4665\HW1\Sol1.tex, also add the output.

ASSISTANT: [Configured LaTeX listings package and embedded the full Python source code and its numerical execution output into the document.]