Optimization Algorithms: Theory and Software Implementation

Prof. Thirumulanathan D

Department of Mathematics

Indian Institute of Technology Kanpur

Lecture: 19

Hello everyone, this is the fourth lecture of week four. Please recall that in previous lectures, we learned about the gradient descent algorithm and its drawbacks. Specifically, when minimizing a quadratic function where the Hessian matrix has eigenvalues with a very small ratio between the smallest and largest eigenvalues, the number of iterations required for gradient descent becomes excessively large.

To address this limitation, we first explored the coordinate descent algorithm. Subsequently, we implemented a change of coordinates, aligning our axes with the eigenvectors of the Hessian matrix, as discussed previously.

However, this eigenvector-based approach presents two significant challenges. First, computing all eigenvectors of the Hessian matrix is computationally expensive. We therefore seek alternative methods that avoid this high computational cost. Second, we aim to extend these optimization techniques to non-quadratic functions, which requires further consideration.

We will examine these issues in this lecture and the next. To illustrate, recall our example minimization problem with the Hessian matrix

H = [[2, 1],

[1, 2]].

For this problem, we selected search directions corresponding to scalar multiples of the eigenvectors (1, 1) and (1, -1).

Note : $g_k = g^k$, $\beta_k = \beta^k$, $d_k = d^k$

Before addressing more sophisticated questions, we should first consider whether the solution can be attained using directions other than the eigenvectors. Let us examine the Hessian matrix H = [[2, 1], [1, 2]] with initial point x^0 = (4, -5), as in previous illustrations.

We choose the first search direction as d^0 = (1, 0).

The gradient at x^0 is computed as:

$g^0 = H x^0$ = [[2, 1], [1, 2]] · (4, -5) = (2·4 + 1·(-5), 1·4 + 2·(-5)) = (3, -6)

The optimal step size α^0 is calculated as follows:

$g^{0T} d^0$ = (3, -6) · (1, 0) = 3

$d^{0T} H d^0$ = (1, 0) · [[2, 1], [1, 2]] · $(1, 0)^T$ = (1, 0) · (2, 1) = 2

$\alpha^0 = -(g^{0T} d^0) / (d^{0T} H d^0) = -3/2 = -1.5$

The updated point becomes:

$x^1 = x^0 + \alpha^0 d^0 = (4, -5) + (-1.5)\cdot(1, 0) = (2.5, -5)$

For the second step, we choose $d^1 = (1, -2)$. The gradient at x^1 is:

$g^1 = H x^1 = [[2, 1], [1, 2]] \cdot (2.5, -5) = (0, -7.5)$

The optimal step size α^1 is calculated as:

$g^{1T} d^1 = (0, -7.5) \cdot (1, -2) = 15$

$d^{1T} H d^1 = (1, -2) \cdot [[2, 1], [1, 2]] \cdot (1, -2)^T$

First, compute $H d^1 = [[2, 1], [1, 2]] \cdot (1, -2) = (0, -3)$

Then $d^{1T} (H d^1) = (1, -2) \cdot (0, -3) = 6$

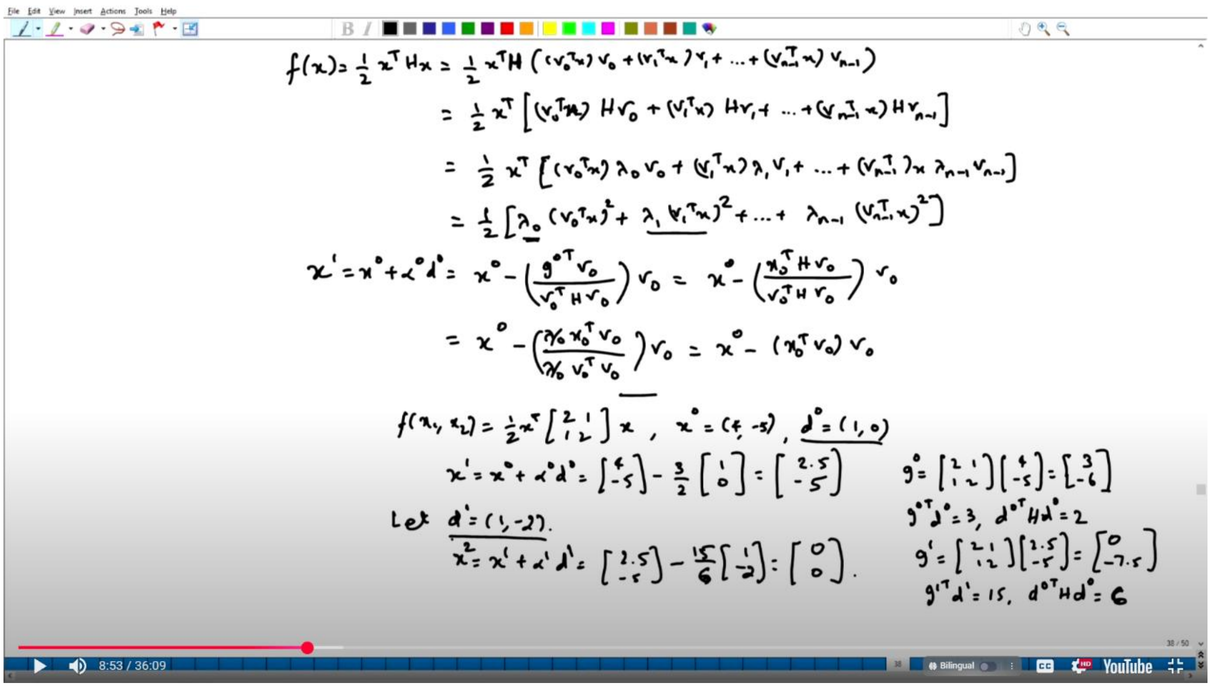
$\alpha^1 = -(g^{1T} d^1) / (d^{1T} H d^1) = -15/6 = -2.5$

The final point is:

$x^2 = x^1 + \alpha^1 d^1 = (2.5, -5) + (-2.5)\cdot(1, -2) = (0, 0)$

This demonstration shows that directions other than eigenvectors, such as $d^0 = (1, 0)$ and $d^1 = (1, -2)$, can also achieve convergence to the minimum in two steps for a quadratic function. This indicates that multiple sets of directions can lead to the solution, and eigenvectors are not the only valid choice.

(Refer Slide Time 8:54)



$$f(x) = \frac{1}{2} x^T H x = \frac{1}{2} x^T H \left((v_0^T x) v_0 + (v_1^T x) v_1 + \dots + (v_{n-1}^T x) v_{n-1} \right)$$

$$= \frac{1}{2} x^T \left[(v_0^T x) H v_0 + (v_1^T x) H v_1 + \dots + (v_{n-1}^T x) H v_{n-1} \right]$$

$$= \frac{1}{2} x^T \left[(v_0^T x) \lambda_0 v_0 + (v_1^T x) \lambda_1 v_1 + \dots + (v_{n-1}^T x) \lambda_{n-1} v_{n-1} \right]$$

$$= \frac{1}{2} \left[\underline{\lambda_0} (v_0^T x)^2 + \underline{\lambda_1 (v_1^T x)}^2 + \dots + \lambda_{n-1} (v_{n-1}^T x)^2 \right]$$

$$x^1 = x^0 + \alpha^0 d^0 = x^0 - \left(\frac{g^{0T} v_0}{v_0^T H v_0} \right) v_0 = x^0 - \left(\frac{x_0^T H v_0}{v_0^T H v_0} \right) v_0$$

$$= x^0 - \left(\frac{\cancel{\lambda_0} x_0^T v_0}{\cancel{\lambda_0} v_0^T v_0} \right) v_0 = x^0 - (x_0^T v_0) v_0$$

$$f(x_1, x_2) = \frac{1}{2} x^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} x, \quad x^0 = (4, -5), \quad \underline{d^0 = (1, 0)}$$

$$x^1 = x^0 + \alpha^0 d^0 = \begin{bmatrix} 4 \\ -5 \end{bmatrix} - \frac{3}{2} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2.5 \\ -5 \end{bmatrix} \qquad g^0 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ -5 \end{bmatrix} = \begin{bmatrix} 3 \\ -6 \end{bmatrix}$$

$$g^{0T} d^0 = 3, \quad d^{0T} H d^0 = 2$$

Let $\underline{d^1 = (1, -2)}$.

$$x^2 = x^1 + \alpha^1 d^1 = \begin{bmatrix} 2.5 \\ -5 \end{bmatrix} - \frac{15}{6} \begin{bmatrix} 1 \\ -2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \qquad g^1 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 2.5 \\ -5 \end{bmatrix} = \begin{bmatrix} 0 \\ -7.5 \end{bmatrix}$$

$$g^{1T} d^1 = 15, \quad d^{0T} H d^0 = 6$$

A key question arises: how can such directions be systematically constructed? The answer lies in the following principle. For minimizing a quadratic function with a positive definite Hessian H, the algorithm converges to the solution in at most n steps if we choose directions d^0, d^1, ..., d^{n-1} such that:

$d_i^T H d_j = 0$ for all $i \neq j$

This condition must be combined with exact line search at each iteration to compute the optimal step size.

To verify that eigenvectors satisfy this condition, consider eigenvectors d_i and d_j with corresponding eigenvalues λ_i and λ_j. Since H is symmetric, its eigenvectors are orthogonal:

$d_i^T d_j = 0$ for $i \neq j$

Then:

$d_i^T H d_j = d_i^T (\lambda_j d_j) = \lambda_j (d_i^T d_j) = 0$

Thus, eigenvectors naturally satisfy the conjugacy condition $d_i^T H d_j = 0$, which is why they yield convergence in n steps.

In the example with $d^0 = (1, 0)$ and $d^1 = (1, -2)$, we can verify:

$H d^1 = [[2, 1], [1, 2]] \cdot (1, -2) = (0, -3)$

$d^{0T} (H d^1) = (1, 0) \cdot (0, -3) = 0$

This confirms that d^0 and d^1 are conjugate with respect to H, explaining why convergence occurred in two steps.

The central question then becomes: how can we generate such conjugate directions for an arbitrary symmetric positive definite matrix H without explicitly computing eigenvectors? This leads to methods like the Conjugate Gradient algorithm, which iteratively constructs conjugate directions using the gradients observed at each step, avoiding the computational expense of eigenvector decomposition.

The central challenge is to efficiently generate a set of H-conjugate directions (d^0, d^1, ..., d^{n-1}) for an arbitrary symmetric positive definite Hessian matrix H, such that $d_i^T H d_j = 0$ for all $i \neq j$. Using eigenvectors satisfies this condition but is computationally prohibitive for large-scale problems.

The solution is the Conjugate Gradient algorithm, specifically the Fletcher-Reeves formulation. This method iteratively constructs the conjugate directions using the gradient information from each step, avoiding the explicit calculation of eigenvectors or the full Hessian.

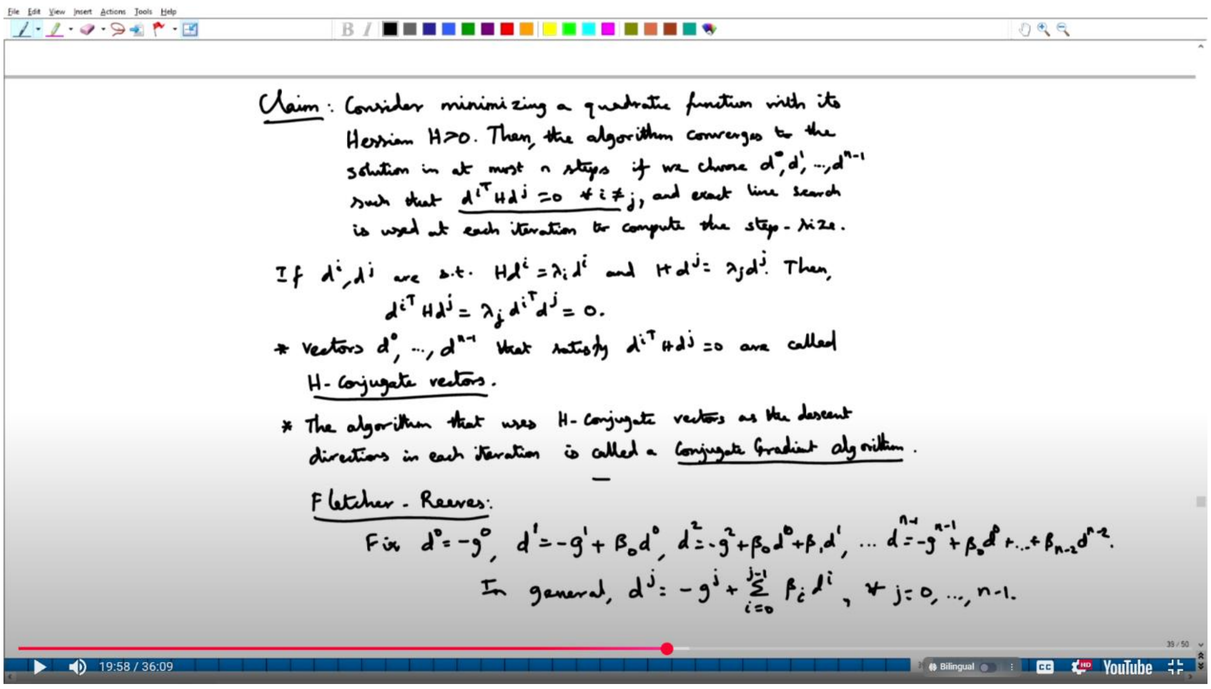
The algorithm is initialized with the standard steepest descent direction:

$d^0 = -g^0$

Subsequent directions are constructed using a recurrence relation that incorporates information from all previous directions:

$d^j = -g^j + \sum_{i=0}^{j-1} \beta_i d_i$

(Refer Slide Time 20:00)



Claim: Consider minimizing a quadratic function with its Hessian $H \succ 0$. Then, the algorithm converges to the solution in at most n steps if we choose $d^0, d^1, \dots, d^{n-1}$ such that $\underline{d^{i^T} H d^j = 0 \ \forall i \neq j}$, and exact line search is used at each iteration to compute the step-size.

If d^i, d^j are s.t. $H d^i = \lambda_i d^i$ and $H d^j = \lambda_j d^j$. Then,

$$d^{i^T} H d^j = \lambda_j d^{i^T} d^j = 0.$$

* Vectors $d^0, \dots, d^{n-1}$ that satisfy $d^{i^T} H d^j = 0$ are called H-Conjugate vectors.

* The algorithm that uses H-Conjugate vectors as the descent directions in each iteration is called a Conjugate Gradient algorithm.

Fletcher - Reeves:

$$\text{Fix } d^0 = -g^0,\ d^1 = -g^1 + \beta_0 d^0,\ d^2 = -g^2 + \beta_0 d^0 + \beta_1 d^1,\ \dots\ d^{n-1} = -g^{n-1} + \beta_0 d^0 + \dots + \beta_{n-2} d^{n-2}.$$

$$\text{In general, } d^j = -g^j + \sum_{i=0}^{j-1} \beta_i d^i,\ \forall j = 0, \dots, n-1.$$

The critical step is to choose the coefficients β_i to enforce the conjugacy condition ($d_l^T H d^j = 0$ for $l < j$). Substituting the expression for d^j into this condition yields:

$d_l^T H d^j = -d_l^T H g^j + \sum_{i=0}^{j-1} \beta_i (d_l^T H d_i)$. Note $(d_l^T = (d^l)^T$

For $l < j$, the conjugacy of the previous directions ($d_l^T H d_i = 0$ for $l \neq i$) simplifies the sum. The only non-zero term in the summation occurs when $i = l$:

$d_l^T H d^j = -d_l^T H g^j + \beta_l (d_l^T H d_l)$

Setting this equal to zero to enforce conjugacy gives the formula for the coefficient β_l:

$\beta_l = (d_l^T H g^j) / (d_l^T H d_l) = (g^{jT} H d_l) / (d_l^T H d_l)$

Therefore, the direction update rule becomes:

$d^j = -g^j + \sum_{i=0}^{j-1} [(g^{jT} H d_i) / (d_i^T H d_i)] d_i$

This formulation ensures that each new direction d^j is conjugate to all previous directions $d^0, \dots, d^{j-1}$.

The term $-g^j$ represents the steepest descent direction, while the summation term modifies it to maintain conjugacy with the past search directions, leading to much faster convergence than standard gradient descent.

The Fletcher-Reeves conjugate gradient algorithm constructs search directions using the recurrence relation:

$d^j = -g^j + \sum_{i=0}^{j-1} \beta_i d_i$

where the coefficients β_i are chosen to enforce conjugacy: $d_i^T H d^j = 0$ for all $i < j$. This ensures the directions are H-conjugate.

To simplify this expression, we use the relation between consecutive gradients. From the update $x^{i+1} = x^i + \alpha^i d^i$, we have:

$g^{i+1} = g^i + \alpha^i H d^i$

Rearranging gives:

$H d^i = (1/\alpha^i)(g^{i+1} - g^i)$

Substituting this into the expression for β_i eliminates the explicit Hessian:

$\beta_i = [g^{jT} (g^{i+1} - g^i)] / [d^{iT} (g^{i+1} - g^i)]$

This simplification is possible because the scalar $1/\alpha^i$ cancels in the numerator and denominator. The resulting formula for the direction becomes:

$d^j = -g^j + \sum_{i=0}^{j-1} [g^{jT} (g^{i+1} - g^i) / d^{iT} (g^{i+1} - g^i)] d^i$

This form avoids direct computation of H, relying only on gradient differences. Further simplification arises from the orthogonality properties of gradients and directions in conjugate gradient methods, which ultimately reduce the summation to a single term, leading to the efficient recurrence:

$d^j = -g^j + \beta^{j-1} d^{j-1}$

with:

$\beta^{j-1} = (g^{jT} g^j) / (g^{j-1T} g^{j-1})$

This final form is computationally efficient and ensures convergence within n steps for quadratic functions.

The Fletcher-Reeves conjugate gradient algorithm efficiently constructs conjugate directions through mathematical simplifications that leverage key properties of the method. The process begins with the general form:

$d^j = -g^j + \sum_{i=0}^{j-1} [g^{jT} (g^{i+1} - g^i) / d^{iT} (g^{i+1} - g^i)] d^i$

This expression is simplified using fundamental orthogonal properties of the conjugate gradient method. Consider the relationship between iterates:

$x^k = x^j + \sum_{i=j}^{k-1} \alpha_i d_i$

Applying the Hessian H and adding b (where $g(x) = Hx + b$) yields:

$Hx^k + b = Hx^j + b + \sum_{i=j}^{k-1} \alpha_i H d_i$

This simplifies to:

$g^k = g^j + \sum_{i=j}^{k-1} \alpha_i H d_i$

Now, multiplying both sides by d_l^T for some $l < k$:

$g^{kT} d_l = g^{jT} d_l + \sum_{i=j}^{k-1} \alpha_i d_l^T H d_i$

Due to the conjugacy condition ($d_l^T H d_i = 0$ for $l \neq i$), the summation reduces to the single term where $i = l$:

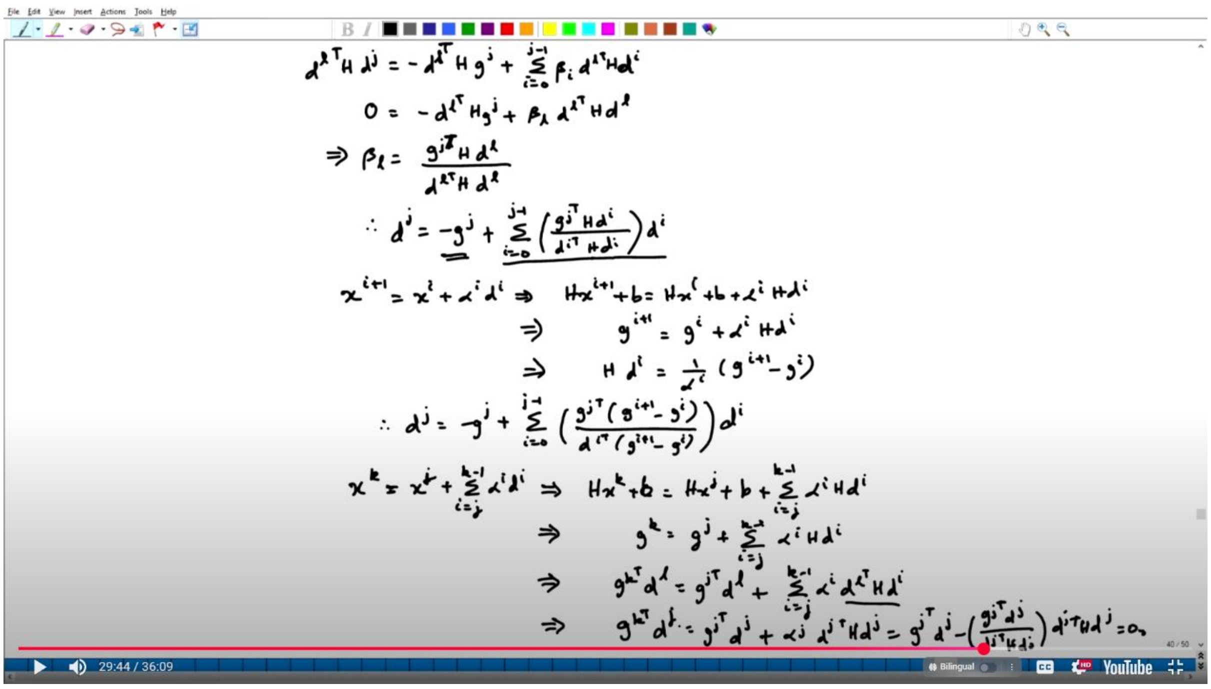
$g^{kT} d_l = g^{jT} d_l + \alpha_l d_l^T H d_l$

For the specific case where $l = j$ and $k > j$, the orthogonality property $g^{kT} d^j = 0$ (a result of exact line search) holds. Substituting $\alpha_j = - (g^{jT} d^j) / (d^{jT} H d^j)$ into the equation:

$0 = g^{jT} d^j - [(g^{jT} d^j) / (d^{jT} H d^j)] d^{jT} H d^j = 0$

This confirms the orthogonality.

(Refer Slide Time 29:44)



$$d^{lT} H d^j = -d^{lT} H g^j + \sum_{i=0}^{j-1} \beta_i d^{lT} H d^i$$

$$0 = -d^{lT} H g^j + \beta_l d^{lT} H d^l$$

$$\Rightarrow \beta_l = \frac{g^{jT} H d^l}{d^{lT} H d^l}$$

$$\therefore \underline{\underline{d^j = -g^j}} + \underline{\sum_{i=0}^{j-1} \left(\frac{g^{jT} H d^i}{d^{iT} H d^i} \right) d^i}$$

$$x^{i+1} = x^i + \alpha^i d^i \Rightarrow Hx^{i+1} + b = Hx^i + b + \alpha^i H d^i$$

$$\Rightarrow g^{i+1} = g^i + \alpha^i H d^i$$

$$\Rightarrow H d^i = \frac{1}{\alpha^i} (g^{i+1} - g^i)$$

$$\therefore d^j = -g^j + \sum_{i=0}^{j-1} \left(\frac{g^{jT}(g^{i+1} - g^i)}{d^{iT}(g^{i+1} - g^i)} \right) d^i$$

$$x^k = x^j + \sum_{i=j}^{k-1} \alpha^i d^i \Rightarrow Hx^k + b = Hx^j + b + \sum_{i=j}^{k-1} \alpha^i H d^i$$

$$\Rightarrow g^k = g^j + \sum_{i=j}^{k-1} \alpha^i H d^i$$

$$\Rightarrow g^{kT} d^l = g^{jT} d^l + \sum_{i=j}^{k-1} \alpha^i \underline{d^{lT} H d^i}$$

$$\Rightarrow g^{kT} d^j = g^{jT} d^j + \alpha^j d^{jT} H d^j = g^{jT} d^j - \left(\frac{g^{jT} d^j}{d^{jT} H d^j} \right) d^{jT} H d^j = 0$$

Furthermore, the mutual orthogonality of gradients ($g^{kT} g^j = 0$ for $k \neq j$) causes most terms in the original summation for d^j to vanish. Specifically, for $i < j-1$, $g^{jT} (g^{i+1} - g^i) = 0$, reducing the summation to the single term $i = j-1$:

$d^j = -g^j + [g^{jT} (g^j - g^{j-1}) / d^{j-1T} (g^j - g^{j-1})] d^{j-1}$

Using the orthogonality condition $d^{j-1T} g^j = 0$ and the relation $d^{j-1T} g^{j-1} = -g^{j-1T} g^{j-1}$, the denominator simplifies:

$d^{j-1T} (g^j - g^{j-1}) = g^{j-1T} g^{j-1}$

Thus, the coefficient simplifies to:

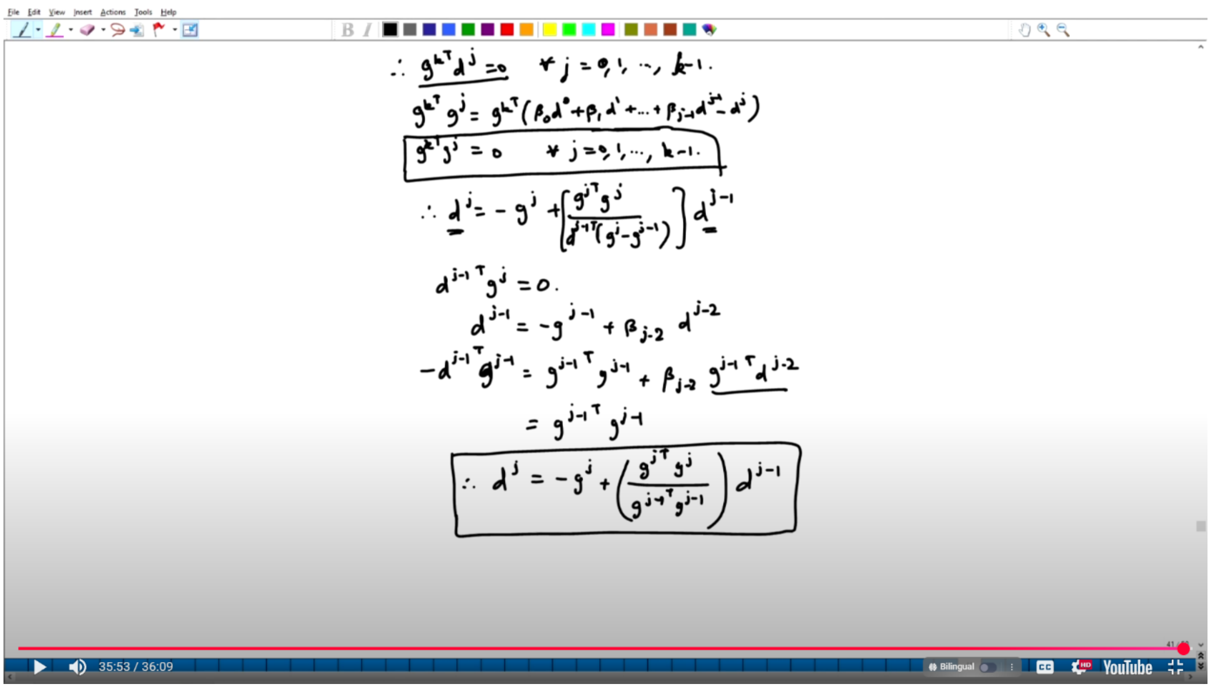
$\beta^{j-1} = (g^{jT} g^j) / (g^{j-1T} g^{j-1})$

Resulting in the efficient recurrence:

$d^j = -g^j + [(g^{jT} g^j) / (g^{j-1T} g^{j-1})] d^{j-1}$

This formulation ensures conjugacy while depending only on the immediate previous direction and the current and previous gradients, making it computationally efficient for solving large-scale optimization problems.

(Refer Slide Time 35:54)



$$\therefore \underline{g^{k^T} d^j = 0} \quad \forall j = 0, 1, \dots, k-1.$$

$$g^{k^T} g^j = g^{k^T}(\beta_0 d^0 + \beta_1 d^1 + \dots + \beta_{j-1} d^{j-1} - d^j)$$

$$\boxed{g^{k^T} g^j = 0 \quad \forall j = 0, 1, \dots, k-1.}$$

$$\therefore \underline{d}^j = -g^j + \left[\frac{g^{j^T} g^j}{d^{j-1^T}(g^j - g^{j-1})}\right] \underline{d}^{j-1}$$

$$d^{j-1^T} g^j = 0.$$

$$d^{j-1} = -g^{j-1} + \beta_{j-2} d^{j-2}$$

$$-d^{j-1^T} g^{j-1} = g^{j-1^T} g^{j-1} + \beta_{j-2} \underline{g^{j-1^T} d^{j-2}}$$

$$= g^{j-1^T} g^{j-1}$$

$$\boxed{\therefore d^j = -g^j + \left(\frac{g^{j^T} g^j}{g^{j-1^T} g^{j-1}}\right) d^{j-1}}$$

We will continue in the next lecture. Thank you.