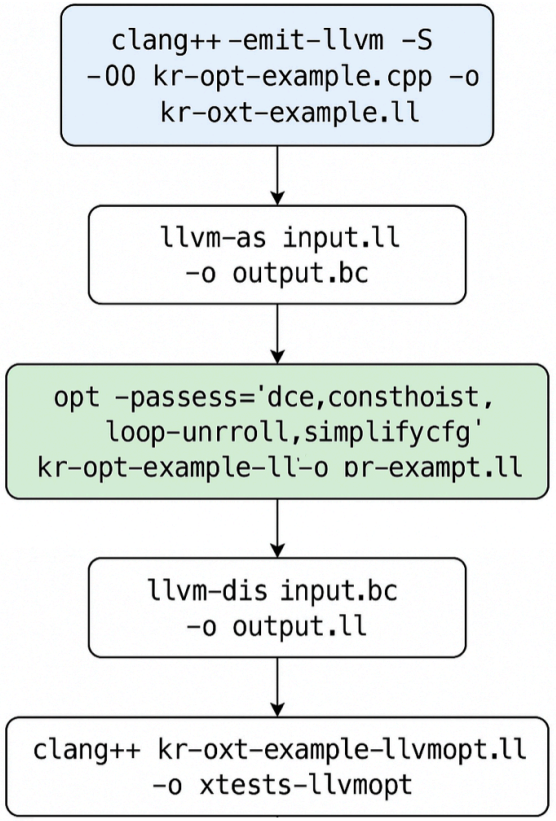
lecture 15

- *llvm - a powerful tool for code analysis*
  - *optimization example*
- *linear systems - Gauss elimination method*
  - *elementary matrices, LU decomposition*
- *orthogonal projection*
- *best fit (least squares) - the normal equations*
- *C++ thread -second intro*
  - *races and mutex variables (locks)*
  - *Riemann sum - quadrature example with speedup*

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

```cpp
#include <iostream>
#include <vector>
#include <chrono>

void matmul_ikj(const std::vector<double>& A,
                const std::vector<double>& B,
                std::vector<double>& C,
                int N) {
    for (int i = 0; i < N; ++i) {
        for (int k = 0; k < N; ++k) {
            double r = A[i * N + k];
            for (int j = 0; j < N; ++j) {
                C[i * N + j] += r * B[k * N + j];
            }
        }
    }
}

int main() {
    const int N = 512;
    std::vector<double> A(N * N, 1.0);
    std::vector<double> B(N * N, 2.0);
    std::vector<double> C(N * N, 0.0);

    auto start = std::chrono::high_resolution_clock::now();
    matmul_ikj(A, B, C, N);
    auto end = std::chrono::high_resolution_clock::now();

    std::chrono::duration<double> diff = end - start;
    std::cout << "Matrix multiplication took " << diff.count() << " seconds\n";
    std::cout << "Checksum: " << C[N * N - 1] << "\n"; // Ensure work not optimized out

    return 0;
}
```



Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

🧾 Step-by-step: Generate LLVM IR

```bash
clang++ -emit-llvm -S -O0 kr-opt-example.cpp -o kr-opt-example.ll
```

- Generates unoptimized human-readable LLVM IR (`.ll`).

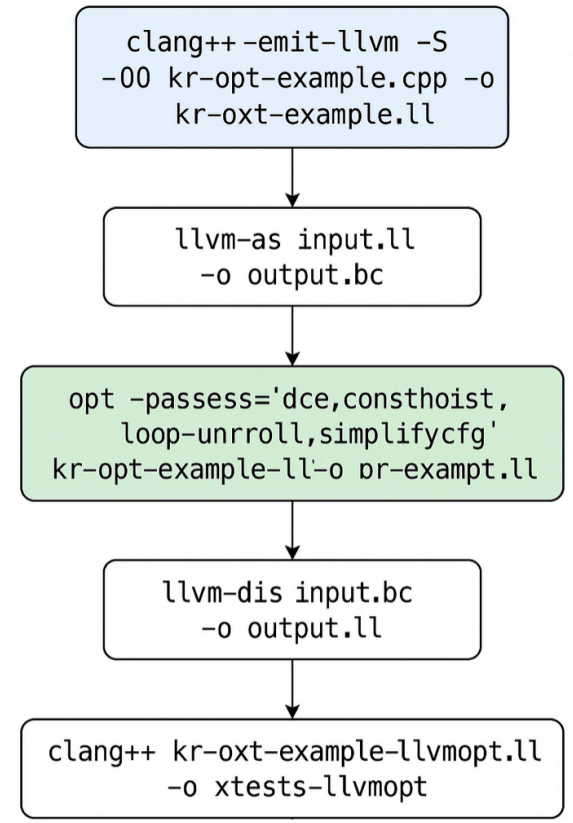
💾 Convert IR to Bitcode (Optional)

```bash
llvm-as kr-opt-example.ll -o kr-opt-example.bc
```

- `.bc` is a compact, binary version of the IR.
- Better for processing in pipelines.

🔍 View IR or Bitcode

```bash
less kr-opt-example.ll          # View source-level LLVM IR
llvm-dis kr-opt-example.bc      # Convert back to .ll for debugging
```



Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

📄 Step-by-step: Generate LLVM IR

```bash
clang++ -emit-llvm -S -O0 kr-opt-example.cpp -o kr-opt-example.ll
```

- Generates unoptimized human-readable LLVM IR (`.ll`).

💾 Convert IR to Bitcode (Optional)

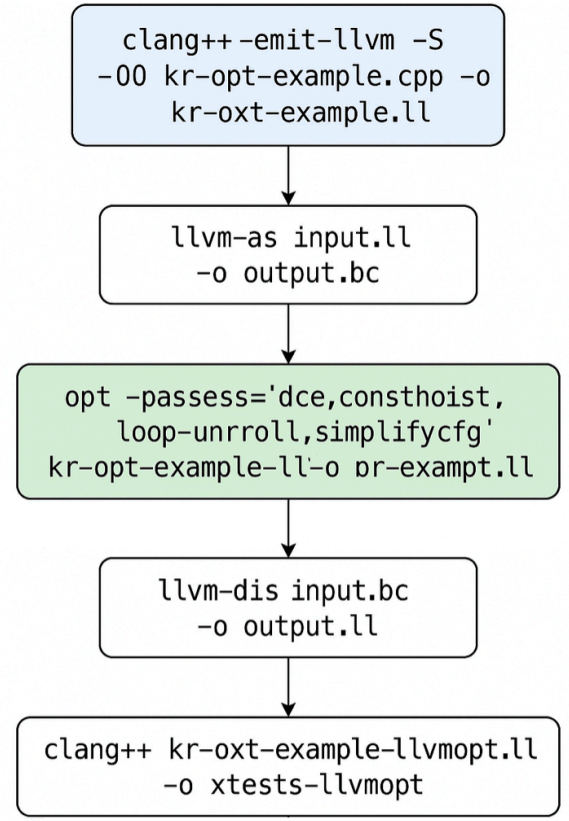
```bash
llvm-as kr-opt-example.ll -o kr-opt-example.bc
```

- `.bc` is a compact, binary version of the IR.
- Better for processing in pipelines.

🔍 View IR or Bitcode

```bash
less kr-opt-example.ll         # View source-level LLVM IR
llvm-dis kr-opt-example.bc     # Convert back to .ll for debugging
```

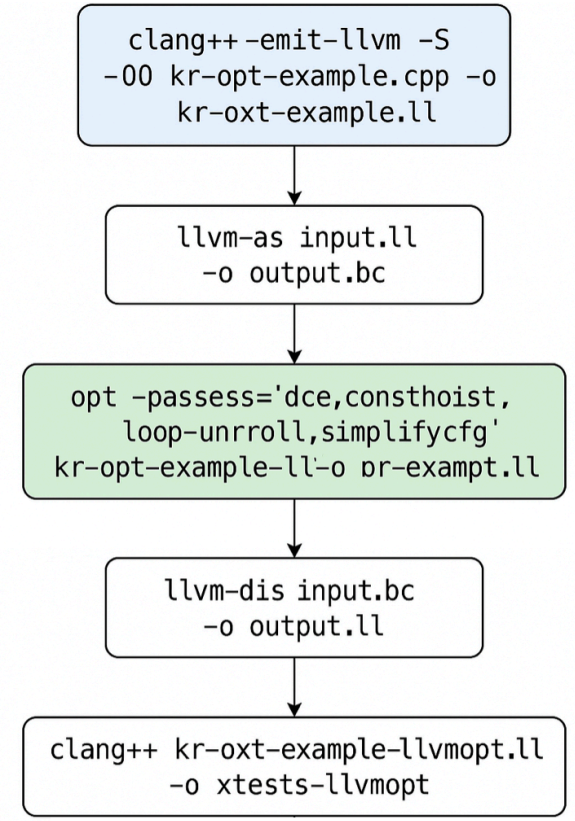
```
bash-3.2$ clang++ -emit-llvm -S -O0 kr-opt-example.cpp -o kr-opt-example.ll
bash-3.2$ llvm-as kr-opt-example.ll -o kr-opt-example.bc
bash-3.2$ llvm-dis kr-opt-example.bc
bash-3.2$ ls -lstr
total 496
  8 -rw-r--r--  1 kennethroche  staff     989 May  2 00:53 kr-opt-example.cpp
104 -rw-r--r--  1 kennethroche  staff   52752 May  2 00:57 kr-opt-example.bc
384 -rw-r--r--  1 kennethroche  staff  145977 May  2 00:58 kr-opt-example.ll
bash-3.2$
```



Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

Apply custom optimization pass:

- Applies selected LLVM passes:
  - `dce` : Dead Code Elimination
  - `consthoist` : Hoist constants out of loops
  - `loop-unroll` : Unroll loops to reduce overhead
  - `simplifycfg` : Simplify control flow graphs




```
bash-3.2$ opt -passes='dce,consthoist,loop-unroll,simplifycfg' kr-opt-example.ll -o kr-opt-example-llvmopt.
bash-3.2$ clang++ kr-opt-example-llvmopt.ll -o xtests-llvmopt
bash-3.2$ ls -lstr
total 704
  8 -rw-r--r--  1 kennethroche  staff     989 May  2 00:53 kr-opt-example.cpp
104 -rw-r--r--  1 kennethroche  staff   52752 May  2 00:57 kr-opt-example.bc
384 -rw-r--r--  1 kennethroche  staff  145977 May  2 00:58 kr-opt-example.ll
104 -rw-r--r--  1 kennethroche  staff   52752 May  2 01:06 kr-opt-example-llvmopt.ll
104 -rwxr-xr-x  1 kennethroche  staff   50656 May  2 01:06 xtests-llvmopt
bash-3.2$
```

```
bash-3.2$ ./xtests-llvmopt
Matrix multiplication took 0.405299 seconds
Checksum: 1024
```

- # linear systems, elementary matrices, Gauss elimination

given an n-vector $\boldsymbol{a}$, we can annihilate *all* of its entries below the kth position, provided that $a_k \neq 0$, by the following transformation:

$$\boldsymbol{M}_k \boldsymbol{a} = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & -m_{k+1} & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -m_n & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ a_{k+1} \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where $m_i = a_i/a_k$, $i = k+1, \ldots, n$. The divisor a_k is called the *pivot*. A matrix of this form is sometimes called an *elementary elimination matrix* or *Gauss transformation*, and its effect on a vector is to add a multiple of row k to each subsequent row, with the multipliers m_i chosen so that the result in each case is zero.

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

1. $\boldsymbol{M}_k$ is a lower triangular matrix with unit main diagonal, and hence it must be nonsingular.
2. $\boldsymbol{M}_k = \boldsymbol{I} - \boldsymbol{m}\boldsymbol{e}_k^T$, where $\boldsymbol{m} = [0, \ldots, 0, m_{k+1}, \ldots, m_n]^T$ and $\boldsymbol{e}_k$ is the kth column of the identity matrix.
3. $\boldsymbol{M}_k^{-1} = \boldsymbol{I} + \boldsymbol{m}\boldsymbol{e}_k^T$, which means that $\boldsymbol{M}_k^{-1}$, which we will denote by $\boldsymbol{L}_k$, is the same as $\boldsymbol{M}_k$ except that the signs of the multipliers are reversed.
4. If $\boldsymbol{M}_j$, $j > k$, is another elementary elimination matrix, with vector of multipliers $\boldsymbol{t}$, then

$$\boldsymbol{M}_k \boldsymbol{M}_j = \boldsymbol{I} - \boldsymbol{m}\boldsymbol{e}_k^T - \boldsymbol{t}\boldsymbol{e}_j^T + \boldsymbol{m}\boldsymbol{e}_k^T \boldsymbol{t}\boldsymbol{e}_j^T = \boldsymbol{I} - \boldsymbol{m}\boldsymbol{e}_k^T - \boldsymbol{t}\boldsymbol{e}_j^T,$$

since $\boldsymbol{e}_k^T \boldsymbol{t} = 0$. Thus, their product is essentially their "union." Because they have the same form, a similar result holds for the product of their inverses, $\boldsymbol{L}_k \boldsymbol{L}_j$. Note that the order of multiplication is significant; these results do not hold for the reverse product.

Elementary Elimination Matrices. If $\boldsymbol{a} = [2 \quad 4 \quad -2]^T$, then

$$\boldsymbol{M}_1 \boldsymbol{a} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{M}_2 \boldsymbol{a} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.5 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix}.$$

We also note that

$$\boldsymbol{L}_1 = \boldsymbol{M}_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{L}_2 = \boldsymbol{M}_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.5 & 1 \end{bmatrix},$$

and

$$\boldsymbol{M}_1 \boldsymbol{M}_2 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0.5 & 1 \end{bmatrix}, \quad \boldsymbol{L}_1 \boldsymbol{L}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -0.5 & 1 \end{bmatrix}.$$

Solve: linear system $\boldsymbol{Ax} = \boldsymbol{b}$

$$\boldsymbol{A} = \boldsymbol{LU} \qquad \boldsymbol{LUx} = \boldsymbol{b} \qquad \boldsymbol{Ly} = \boldsymbol{b} \qquad \boldsymbol{Ux} = \boldsymbol{y}$$

define the matrix $\boldsymbol{M} = \boldsymbol{M}_{n-1} \cdots \boldsymbol{M}_1$

$$\boldsymbol{MAx} = \boldsymbol{M}_{n-1} \cdots \boldsymbol{M}_1 \boldsymbol{Ax} = \boldsymbol{M}_{n-1} \cdots \boldsymbol{M}_1 \boldsymbol{b} = \boldsymbol{Mb}$$

$$\boldsymbol{U} = \boldsymbol{MA}$$

$$\boldsymbol{L} = \boldsymbol{M}^{-1} = (\boldsymbol{M}_{n-1} \cdots \boldsymbol{M}_1)^{-1} = \boldsymbol{M}_1^{-1} \cdots \boldsymbol{M}_{n-1}^{-1} = \boldsymbol{L}_1 \cdots \boldsymbol{L}_{n-1}$$

Note that $\boldsymbol{L}_k \boldsymbol{L}_j$ is unit lower triangular $k < j$

LU direct decomposition of A

```
for k = 1 to n - 1                          { loop over columns }
    if a_kk = 0 then stop                   { stop if pivot is zero }
    for i = k + 1 to n                      { compute multipliers
        m_ik = a_ik / a_kk                      for current column }
    end
    for j = k + 1 to n
        for i = k + 1 to n                  { apply transformation to
            a_ij = a_ij - m_ik a_kj             remaining submatrix }
        end
    end
end
```

$$
\begin{aligned}
x_1 + 2x_2 + 2x_3 &= 3, \\
4x_1 + 4x_2 + 2x_3 &= 6, \\
4x_1 + 6x_2 + 4x_3 &= 10,
\end{aligned}
\qquad
\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} 1 & 2 & 2 \\ 4 & 4 & 2 \\ 4 & 6 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 10 \end{bmatrix} = \boldsymbol{b}
$$

$$
\boldsymbol{M}_1\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 4 & 4 & 2 \\ 4 & 6 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 \\ 0 & -4 & -6 \\ 0 & -2 & -4 \end{bmatrix}
\qquad
\boldsymbol{M}_1\boldsymbol{b} = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 6 \\ 10 \end{bmatrix} = \begin{bmatrix} 3 \\ -6 \\ -2 \end{bmatrix}
$$

$$
\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & -4 & -6 \\ 0 & -2 & -4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 \\ 0 & -4 & -6 \\ 0 & 0 & -1 \end{bmatrix}
\qquad
\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.5 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ -6 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 \\ -6 \\ 1 \end{bmatrix}
$$

$$
\boldsymbol{U}\boldsymbol{x} = \begin{bmatrix} 1 & 2 & 2 \\ 0 & -4 & -6 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -6 \\ 1 \end{bmatrix} = \boldsymbol{M}\boldsymbol{b} = \boldsymbol{y}
$$

$$
\mathsf{x} = \begin{bmatrix} -1 & 3 & -1 \end{bmatrix}^T
$$

solve by back substitution

Kenneth J. Roche
University of Washington

$$\boldsymbol{L}_1\boldsymbol{L}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.5 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 4 & 0.5 & 1 \end{bmatrix} = \boldsymbol{L}$$

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 2 \\ 4 & 4 & 2 \\ 4 & 6 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 4 & 0.5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & -4 & -6 \\ 0 & 0 & -1 \end{bmatrix} = \boldsymbol{LU}$$

- generating a discrete orthonormal basis on a digital lattice -concepts

$\mathbb{X}$ inner product space. If $x \in \mathbb{X}$ and $x \in \text{span}\{e_1, e_2, \ldots, e_n\}$ n fixed, $x = \alpha_i e_i$.

$(x, e_j) = (\alpha_i e_i, e_j) = \sum_i \alpha_i (e_i, e_j) = \sum_i \alpha_i \delta_{ij} = \alpha_j \Rightarrow x = \sum_{i=1}^{n} (x, e_i) e_i$. Now,

$\forall\, x \in \mathbb{X}$ and x not necessarily in $Y_n = \text{span}\{e_1, e_2, \ldots, e_n\}$, we can write $x = y + z$

where $y \in Y_n$. Here $z = x - y$ and $z \perp y$. This will be true for a particular choice of y. let $y = \sum_i \alpha_i e_i \in Y_n$. and choose $\alpha_i = (x, e_i)$ — The projection of x onto Y_n.

then $\|y\|^2 = (y, y) = \left(\sum_i \alpha_i e_i, \sum_j \alpha_j e_j\right) = \sum_i \sum_j \alpha_i \bar{\alpha}_j (e_i, e_j) = \sum_i \sum_j \alpha_i \bar{\alpha}_j \delta_{ij}$ ...

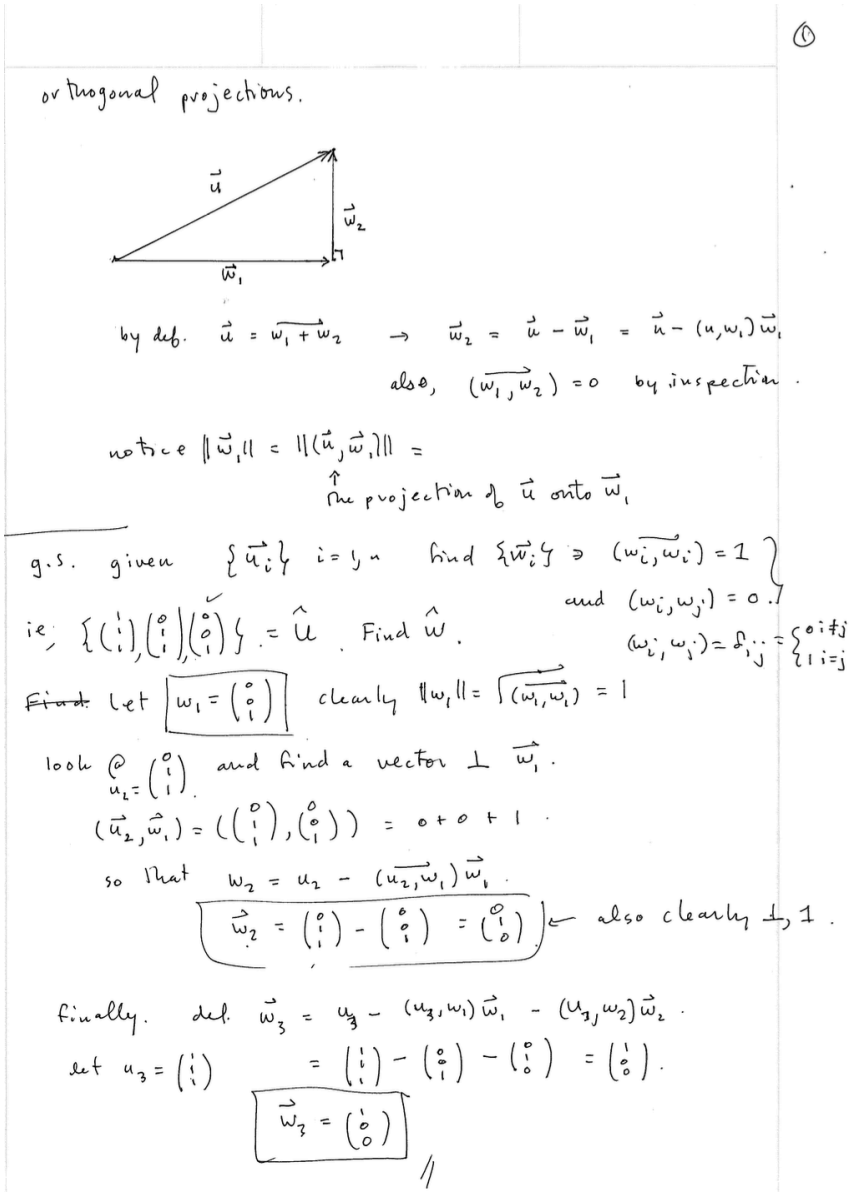
$= \sum_i |\alpha_i|^2 = \sum_i |(x, e_i)|^2$. AND if $z \perp y$ then $(z, y) = 0$. Check. $(x - y, y) = (x, y) - (y, y)$.

$= \left(x, \sum_i (x, e_i) e_i\right) - \|y\|^2 = \sum_i \overline{(x, e_i)} (x, e_i) - \sum_i |(x, e_i)|^2 = 0$. So, for this choice of y the statement holds, $z = x - y$ is $\perp$ y.

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

orthogonal projections and construction of orthonormal basis set

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025



① orthogonal projections.

by def. $\vec{u} = \vec{w_1} + \vec{w_2} \rightarrow \vec{w}_2 = \vec{u} - \vec{w}_1 = \vec{u} - (u, w_1)\vec{w}_1$

also, $(\vec{w_1}, \vec{w_2}) = 0$ by inspection.

notice $\|\vec{w}_1\| = \|(\vec{u}, \vec{w}_1)\| =$ ↑ the projection of $\vec{u}$ onto $\vec{w}_1$

g.s. given $\{\vec{u}_i\}$ $i = 1, n$ find $\{\vec{w}_i\} \ni (\vec{w_i}, \vec{w_i}) = 1$ and $(w_i, w_j) = 0$.

ie; $\{\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\} = \hat{u}$. Find $\hat{w}$.

$(w_i, w_j) = \delta_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$

~~Find~~ Let $\boxed{w_1 = \begin{pmatrix}0\\1\end{pmatrix}}$ clearly $\|w_1\| = \sqrt{(\vec{w_1}, \vec{w_1})} = 1$

look @ $u_2 = \begin{pmatrix}0\\1\end{pmatrix}$ and find a vector $\perp \vec{w}_1$.

$(\vec{u}_2, \vec{w}_1) = \left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right) = 0 + 0 + 1$

so that $w_2 = u_2 - (\vec{u_2}, \vec{w_1})\vec{w}_1$

$\boxed{\vec{w}_2 = \begin{pmatrix}0\\1\end{pmatrix} - \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix}}$ ← also clearly $\perp$, 1.

finally. def $\vec{w}_3 = u_3 - (u_3, w_1)\vec{w}_1 - (u_3, w_2)\vec{w}_2$.

let $u_3 = \begin{pmatrix}1\\1\end{pmatrix}$ $= \begin{pmatrix}1\\1\end{pmatrix} - \begin{pmatrix}0\\1\end{pmatrix} - \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\end{pmatrix}$.

$\boxed{\vec{w}_3 = \begin{pmatrix}0\\0\end{pmatrix}}$ //

one more example.

$$\{\vec{u_1}, \vec{u_2}, \vec{u_3}\} = \left\{\begin{pmatrix}1\\1\\1\end{pmatrix}, \begin{pmatrix}-1\\1\\0\end{pmatrix}, \begin{pmatrix}1\\2\\1\end{pmatrix}\right\}.$$

First. def. $\vec{w}_1 = \dfrac{u_1}{\|u_1\|}$ $\qquad \|u_1\| = (u_1, u_1)^{1/2} = \sqrt{1^2+1^2+1^2} = \sqrt{3}.$

so. $\boxed{\vec{w}_1 = \frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\end{pmatrix}}$

now. form $\vec{w}_2 = u_2 - (u_2, \vec{w}_1)\vec{w}_1$ $\qquad (\vec{u_2}, \vec{w_1}) = \underbrace{(-1\ 1\ 0)}\ \frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\end{pmatrix}.$

$= \vec{u}_2.$ $\qquad = \frac{-1}{\sqrt{3}} + \frac{1}{\sqrt{3}} + 0 = 0.$

$\|\vec{u}_2\| = (u_2, u_2)^{1/2} = \sqrt{(-1)^2 + (1)^2} = \sqrt{2}.$

$\Rightarrow \boxed{\vec{w}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\\0\end{pmatrix}}$

Finally. $\vec{w}_3 = u_3 - (u_3, w_2)\vec{w}_2 - (u_3, w_1)\vec{w}_1$

$(u_3, w_2) = \underbrace{(1\ 2\ 1)}\ \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}(-1+2+0) = \frac{1}{\sqrt{2}}.$

$(u_3, w_1) = \underbrace{(1\ 2\ 1)}\ \frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\end{pmatrix} = \frac{1}{\sqrt{3}}(1+2+1) = \frac{4}{\sqrt{3}}.$

$$= \begin{pmatrix}1\\2\\1\end{pmatrix} - \frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\\0\end{pmatrix} - \frac{4}{\sqrt{3}}\cdot\frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\end{pmatrix}$$

$$= \begin{pmatrix}1\\2\\1\end{pmatrix} - \begin{pmatrix}-\frac{1}{2}\\\frac{1}{2}\\0\end{pmatrix} - \begin{pmatrix}\frac{4}{3}\\\frac{4}{3}\\\frac{4}{3}\end{pmatrix}$$

$$\vec{w}_3 = \begin{pmatrix}1+\frac{1}{2}-\frac{4}{3}\\2-\frac{1}{2}-\frac{4}{3}\\1-0-\frac{4}{3}\end{pmatrix} = \begin{pmatrix}\frac{6}{6}+\frac{3}{6}-\frac{8}{6}\\\frac{12}{6}-\frac{3}{6}-\frac{8}{6}\\\frac{3}{3}-\frac{4}{3}\end{pmatrix} = \begin{matrix}\frac{1}{6}\\\frac{1}{6}\\-\frac{1}{3}=-\frac{2}{6}\end{matrix} = \frac{1}{6}\begin{pmatrix}1\\1\\-2\end{pmatrix}$$

$\vec{w}_3 = \frac{1}{6}\begin{pmatrix}1\\1\\-2\end{pmatrix}.$ $\qquad \|\vec{w}_3\| = (w_3, w_3)^{1/2} = \sqrt{\frac{1}{36}\cdot\sqrt{1^2+1^2+2^2}}$

$= \frac{1}{6}\cdot\sqrt{6} = \frac{\sqrt{6}}{6}.$ ↓

$\vec{w}_3 \to \dfrac{1}{\|\vec{w}_3\|}\cdot\vec{w}_3 = \dfrac{1}{\left(\frac{\sqrt{6}}{6}\right)}\cdot\frac{1}{6}\begin{pmatrix}1\\1\\-2\end{pmatrix} = \frac{6}{\sqrt{6}}\cdot\frac{1}{6}\begin{pmatrix}1\\1\\-2\end{pmatrix} = \frac{1}{\sqrt{6}}\begin{pmatrix}1\\1\\-2\end{pmatrix}$ //

check $\|\vec{w}_3\| = \left[\underbrace{\frac{1}{\sqrt{6}}(1\ 1\ -2)}\cdot\frac{1}{\sqrt{6}}\begin{pmatrix}1\\1\\-2\end{pmatrix}\right]^{1/2} = \sqrt{\frac{1}{6}(1^2+1^2+(-2)^2)}$

$= 1$ //

check $(\vec{w_3}, \vec{w_1}) = \underbrace{\frac{1}{\sqrt{6}}(1\ 1\ -2)}\ \frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\end{pmatrix} = \frac{1}{\sqrt{18}}\cdot(\underbrace{1\cdot1 + 1\cdot1 + (-2)\cdot1}_{2-2=0}) = 0.$ //

$(\vec{w_3}, \vec{w_2}) = \frac{1}{\sqrt{6}}\underbrace{(1\ 1\ -2)}\ \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\\0\end{pmatrix} = \frac{1}{\sqrt{12}}(\underbrace{1\cdot(-1)}_{-1} + \underbrace{1\cdot1}_{+1} + (-2)\cdot0) = 0.$ //

$(\vec{w_1}, \vec{w_2}) = \frac{1}{\sqrt{3}}\underbrace{(1\ 1\ 1)}\ \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\\0\end{pmatrix} = \frac{1}{\sqrt{6}}(-1+1+0) = 0.$ //

check now, $\vec{u}_1 \overset{?}{=} (\vec{u_1}, \vec{w}_1)\vec{w}_1 + (\vec{u_1}, \vec{w_2})\vec{w}_2 + (\vec{u_1}, \vec{w_3})\vec{w}_3$

$$\begin{pmatrix}1\\1\\1\end{pmatrix} = \left[\underbrace{(1\ 1\ 1)}\ \frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\end{pmatrix}\right]\frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\end{pmatrix} + \left[\underbrace{(1\ 1\ 1)}\ \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\\0\end{pmatrix}\right]\frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\\0\end{pmatrix}$$

$$+ \left[\underbrace{(1\ 1\ 1)}\ \frac{1}{\sqrt{6}}\begin{pmatrix}1\\1\\-2\end{pmatrix}\right]\frac{1}{\sqrt{6}}\begin{pmatrix}1\\1\\-2\end{pmatrix}$$

$$= \frac{1}{\sqrt{3}}\cdot3\cdot\frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\end{pmatrix} + \frac{1}{\sqrt{2}}\cdot0\cdot\frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\\0\end{pmatrix} + \frac{1}{\sqrt{6}}\cdot0\cdot\frac{1}{\sqrt{6}}\begin{pmatrix}1\\1\\-2\end{pmatrix}$$

$= \frac{3}{3}\begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}1\\1\\1\end{pmatrix}$ // ✓

$\vec{u}_2$? $\quad \vec{u}_2 = (u_2, w_1)w_1 + (u_2, w_2)w_2 + (u_2, w_3)w_3.$

$= \underbrace{(-1\ 1\ 0)}\ \frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\end{pmatrix} + \underbrace{(-1\ 1\ 0)}\ \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\\0\end{pmatrix}\cdot\frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\\0\end{pmatrix} + \underbrace{(-1\ 1\ 0)}\ \frac{1}{\sqrt{6}}\begin{pmatrix}1\\1\\-2\end{pmatrix}$

$\begin{pmatrix}-1\\1\\0\end{pmatrix} = \frac{2}{2}\begin{pmatrix}-1\\1\\0\end{pmatrix} = \begin{pmatrix}-1\\1\\0\end{pmatrix}$ ✓ // etc.

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

- Fourier sequences - useful orthonormal sets ??

Continuous Fnc. Example. $[0, 2\pi] \leftrightarrow [-\pi, \pi]$ and $(x,y) = \int_{-\pi}^{\pi} x(t)\, y(t)\, dt$ (Real Space)

The set $\{1, \cos nt, \sin nt\}$ is orthogonal. $\{u_n(t)\}$ is orthogonal sequence in X w/ $u_n(t) = \cos nt$, $n \geq 0$

$\{v_n(t)\}$ is orthogonal sequence in X with $v_n(t) = \sin nt$, $n \geq 1$. Seek (u_n, u_m), (v_n, v_m), (u_n, v_m).

$\rightarrow \int_{-\pi}^{\pi} \cos mt \cos nt\, dt = \begin{cases} 0 & m \neq n \\ \pi & m = n \neq 0 \\ 2\pi & m = n = 0 \end{cases}$. $\int_{-\pi}^{\pi} \sin mt \sin nt\, dt = \begin{cases} 0 & m \neq n \\ \pi & m = n \neq 0 \end{cases}$. $\int_{-\pi}^{\pi} \cos mt \sin nt\, dt = 0$.

normalize. we want that $(u_m, u_m) = 1$ but $(u_m, u_m) = \begin{cases} \pi & m \neq 0 \\ 2\pi & m = 0 \end{cases}$ so define $e_m = \frac{u_m}{\sqrt{(u_m, u_m)}}$.

We get $\{u_m\} \rightarrow \left\{\frac{1}{\sqrt{2\pi}}, \frac{1}{\sqrt{\pi}} \cos nt\right\}$ and $\{v_n\} \rightarrow \left\{\frac{1}{\sqrt{\pi}} \sin nt\right\}$. Fourier Sequences ...

- data analysis - best fit

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

k. j. roche.

given set $\{(x_i, y_i)\}$ of experimental / observed values.

wish to find f ∋ $f(x) = y$ is a good fit to the data.

- data contains errors / isn't perfect.

plan: find a curve that "best" fits the data.

simple. try using a line so that

$$y = mx + b \quad \rightarrow \quad y_i = mx_i + b.$$

$$\left.\begin{aligned} y_1 &= mx_1 + b \\ y_2 &= mx_2 + b \\ &\vdots \\ y_n &= mx_n + b. \end{aligned}\right\} \quad \vec{y} = m\vec{x} + b = \begin{pmatrix} \vec{1} & \vec{x} \end{pmatrix} \begin{pmatrix} b \\ m \end{pmatrix}$$

$$\text{so, } \vec{y} = \begin{bmatrix} \vec{1} & \vec{x} \end{bmatrix} \begin{pmatrix} b \\ m \end{pmatrix}$$

$$\{\vec{y} = M\vec{v}\}$$

where $\vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$, $M = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$, $\vec{v} = \begin{pmatrix} b \\ m \end{pmatrix}$

(linear dependence / independence / subspaces)

There are more equations than unknowns – overdetermined. ✱

add page 1b

Now, unless (x_i, y_i) are all colinear, then possible to satisfy the equality. It means $\|\overrightarrow{y - M\vec{v}}\| \neq 0$.

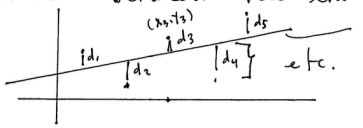
We seek $\hat{\vec{v}}$ ∋ $\|\overrightarrow{y - M\hat{\vec{v}}}\|$ minimizes the error, is as close to zero possible. $\sqrt{(y - Mw,\ y - Mw)} \equiv$ inner product.

$$\|y - Mw\|^2 = (y_1 - (b + mx_1))^2 + (y_2 - (b - mx_2))^2 + \cdots + (y_n - (b - mx_n))^2.$$

let $d_1 = |y_1 - b - mx_1|$, then $d_i^2 = (y_i - (b - mx_i))^2$ ∀ i.

and $\|y - Mw\|^2 = \sum_i d_i^2$. let d_i represent the

distance between the line and point (x_i, y_i).



look @ d_3 ⇒ $y_3 - d_3$ is on the line.

look @ d_4 ⇒ $y_4 + d_4$ is on the line.

that is $y_3 - d_3 = mx_3 + b$ ↳ etc.

and $y_4 + d_4 = mx_4 + b$.

✱ Thm. $\vec{w}$ minimizes $\|\vec{y} - M\vec{w}\|$ when $M\vec{w}$ is the ⊥ projection of $\vec{y}$ on the column space of M.

It means that $(y - M\vec{w}) \cdot M\vec{v} = 0 \quad \forall\ \vec{v} \in \mathbb{R}^2$

we have. $(y - Mw) \cdot M\vec{v} = (M\vec{v})^T \cdot (y - Mw)$

$= \vec{v}^T M^T (y - Mw)$

$= \vec{v}^T (M^T y - M^T M w)$

$= (M^T y - M^T M w) \cdot \vec{v} = 0$

since $\vec{v}$ arbitrary, $M^T y - M^T M w = 0$

$\rightarrow$ $M^T y = M^T M w$

here 2 eqns in 2 unknowns.

These are the normal eqns.

clearly $M^T y = M^T M \omega$ and we seek $\vec{\omega}$.

if $(M^T M)^{-1}$ exists, we're good.

$\vec{\omega} = (M^T M)^{-1} M^T y$ done. ☆ Normal eqns are solved!

Ex Find least squares fit to S.

$\{(0,1),(1,3),(2,4),(3,4)\} = S$ ← given.

$$\begin{aligned} 1 &= m\cdot 0 + b \\ 3 &= m\cdot 1 + b \\ 4 &= m\cdot 2 + b \\ 4 &= m\cdot 3 + b \end{aligned} \quad \Bigg\} \quad \begin{pmatrix} 1 \\ 3 \\ 4 \\ 4 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix}.$$

so, $M = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix}$ $M^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{pmatrix}$.

$$M^T M = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} (0+1+2+3) & (1+1+1+1) \\ (0+1+4+9) & (0+1+2+3) \end{pmatrix}$$

$$N = \begin{pmatrix} 6 & 4 \\ 14 & 6 \end{pmatrix}$$

recall $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

Then $A^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

so long as $\det A \neq 0$.

$\det N = 36 - 56 = -20 \neq 0$.

$N^{-1} = \frac{1}{(-20)}\begin{pmatrix} 6 & -4 \\ -14 & 6 \end{pmatrix}$ ★

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

so that we have

$$\vec{\omega} = \underbrace{\frac{1}{(-20)}\begin{pmatrix} 6 & -4 \\ -14 & 6 \end{pmatrix}}_{\text{★}} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 4 \\ 4 \end{pmatrix}$$

$$\text{★} = -\frac{1}{10}\begin{pmatrix} 3 & -2 \\ -7 & 3 \end{pmatrix} \underbrace{\begin{pmatrix} (1+3+4+4) \\ (0+3+8+12) \end{pmatrix}}_{} = \begin{pmatrix} 12 \\ 23 \end{pmatrix}$$

$$= \left(-\frac{1}{10}\right)\begin{pmatrix} 3 & -2 \\ -7 & 3 \end{pmatrix}\begin{pmatrix} 12 \\ 23 \end{pmatrix}$$

$$= \left(-\frac{1}{10}\right)\begin{pmatrix} 3\cdot 12 - 2\cdot 23 \\ -7\cdot 12 + 3\cdot 23 \end{pmatrix} = -\frac{1}{10}\begin{pmatrix} 36-46 \\ -84+69 \end{pmatrix} = -\frac{1}{10}\begin{pmatrix} -10 \\ -15 \end{pmatrix} = \begin{pmatrix} \frac{-10}{-10} \\ \frac{-15}{-10} \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{2} \end{pmatrix}$$

$-84 + 69$: $7 - 4 = 3$; $80 - 60 = 20$; $20 - 9 = 11$; so -15.

Thus, $\vec{\omega} = \begin{pmatrix} 1 \\ \frac{1}{2} \end{pmatrix} = \begin{pmatrix} m \\ b \end{pmatrix}$

and the best fit (in the least square sense) is $\boxed{y = x + \frac{1}{2}}$ //

☆ Now, let's fit a polynomial of degree n to the data.

ie $y = a_0 + a_1 x^1 + \cdots + a_m x^m$

2

$S = \{ (x_i, y_i) \}$ given.

so, we have.

$$y_1 = a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_m x_1^m$$
$$y_2 = a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_m x_2^m$$
$$\vdots$$
$$y_n = a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_m x_n^m$$

Clearly this reduces to.

$$\vec{y} = M \vec{w}$$

where $M = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{pmatrix}$ $\vec{w} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix}$

here $M = n \times m$ matrix.

Apply same method and find. $\vec{w} = (M^T M)^{-1} M^T \vec{y}$ done.

$E_0 = E_f$. conservative force $f(x)$ (func. of position)

$E_0 = T_0 + U_0 = mgh$

$E_f = \frac{1}{2} m v^2$

$\Big\}$ $mgh = \frac{1}{2} m v^2$. $\rightarrow v = \sqrt{2gh}$ at the ground.

also. recall.

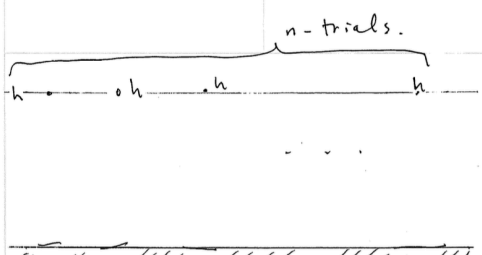
g constant. $\quad \ddot{g}(t) = \frac{dv_y}{dt} \rightarrow v_y = gt + c$ but $v_y = \frac{dy}{dt}$.

$\Big\}$ $dy = (gt + c)\,dt$.

$$y(t) = y_0 + v_0 t + \frac{1}{2} g t^2.$$

conduct experiment to estimate the acceleration due to gravity.

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High
Performance Scientific Computing
Spring Quarter 2025



object of mass m is dropped n-times. The time & distance traveled are estimated. The following data is collected: $\{ (.1, .18), (.2, .31)$ by measure.
$(.3, 1.03), (.4, 2.48), (.5, 3.73) \}$.

estimate g, the initial displacement y_0, and initial velocity v_0.

$$\begin{pmatrix} .18 \\ .31 \\ 1.03 \\ 2.48 \\ 3.73 \end{pmatrix} = \begin{pmatrix} y_0 + v_0(.1) + \frac{1}{2} g (.1)^2 \\ y_0 + v_0(.31) + \frac{1}{2} g (.31)^2 \\ y_0 + v_0(1.03) + \frac{1}{2} g (1.03)^2 \\ y_0 + v_0(2.48) + \frac{1}{2} g (2.48)^2 \\ y_0 + v_0(3.73) + \frac{1}{2} g (3.73)^2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & (.1) & \frac{(.1)^2}{2} \\ 1 & (.31) & \frac{(.31)^2}{2} \\ 1 & (1.03) & \frac{(1.03)^2}{2} \\ 1 & (2.48) & \frac{(2.48)^2}{2} \\ 1 & (3.73) & \frac{(3.73)^2}{2} \end{pmatrix} \begin{pmatrix} y_0 \\ v_0 \\ g \end{pmatrix}$$

$\vec{y} = M \vec{w}$

find. $\vec{w} = (M^T M)^{-1} M^T \vec{y}$ // done.

how good is this estimate?

- *C++ threads introduction*
  - *processing in shared memory and lightweight processes (threads)*

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

```cpp
#include <iostream>
#include <thread>

void threadFunc(int id)
{
    std::cout << "Thread " << id << " started." << std::endl;
    // Do some work...
    std::cout << "Thread " << id << " finished." << std::endl;
}

int main()
{
    const int numThreads = 4;
    std::thread threads[numThreads];

    // Spawn threads
    for (int i = 0; i < numThreads; ++i)
    {
        threads[i] = std::thread(threadFunc, i);
    }

    // Wait for threads to finish
    for (int i = 0; i < numThreads; ++i)
    {
        threads[i].join();
    }

    std::cout << "All threads finished." << std::endl;

    return 0;
}
```

- C++ threads
  - OS allocates a new stack for each thread used to store local variables and function call frames
  - threads share the same heap and code segment as main()
  - dynamically allocated memory is visible to all threads
  - when threads finish, the stack is deallocated by OS, and frees memory it allocated on the heap

```cpp
#include <iostream>
#include <thread>

void threadFunc(int id)
{
    std::cout << "Thread " << id << " started." << std::endl;
    // Do some work...
    std::cout << "Thread " << id << " finished." << std::endl;
}

int main()
{
    const int numThreads = 4;
    std::thread threads[numThreads];

    // Spawn threads
    for (int i = 0; i < numThreads; ++i)
    {
        threads[i] = std::thread(threadFunc, i);
    }

    // Wait for threads to finish
    for (int i = 0; i < numThreads; ++i)
    {
        threads[i].join();
    }

    std::cout << "All threads finished." << std::endl;

    return 0;
}
```

```
bash-3.2$ g++ -o xcpp-thread-into -std=c++14 cpp-thread-intro.cpp
bash-3.2$ ./xcpp-thread-into
Thread 0 started.Thread
Thread 1Thread 0Thread 2 started.
3 started.Thread 2 finished.

Thread 1 finished.
 finished.
 started.
Thread 3 finished.
All threads finished.
bash-3.2$
```

Kenneth J. Roche
University of Washington

```cpp
#include <iostream>
#include <thread>

int count = 0; // global variable

void increment()
{
    for (int i = 0; i < 100000; ++i)
    {
        count++;
    }
}

int main()
{
    std::thread t1(increment);
    std::thread t2(increment);

    t1.join();
    t2.join();

    std::cout << "expected: " << 2 * 100000 << " Count: " << count << std::endl;

    return 0;
}
```

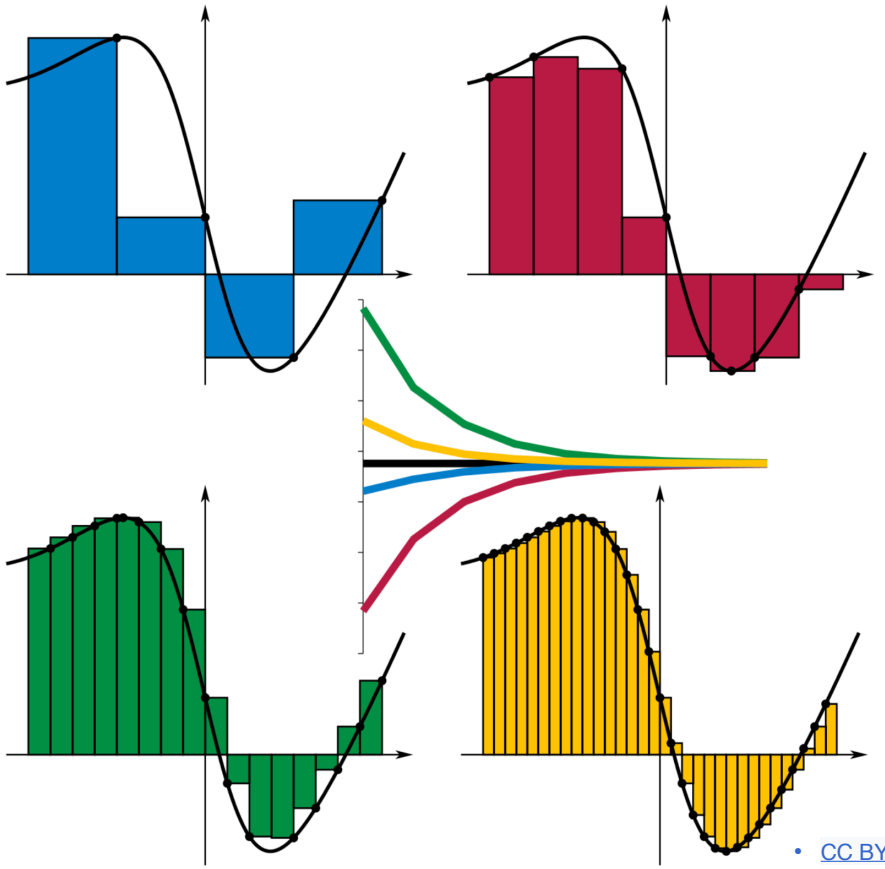
```
bash-3.2$ g++ -o xcpp-thread-race -std=c++14 cpp-thread-race.cpp
bash-3.2$ ./xcpp-thread-race
expected: 200000 Count: 100000
bash-3.2$ ./xcpp-thread-race
expected: 200000 Count: 100730
bash-3.2$ ./xcpp-thread-race
expected: 200000 Count: 105343
bash-3.2$ ./xcpp-thread-race
expected: 200000 Count: 112946
bash-3.2$ ./xcpp-thread-race
expected: 200000 Count: 108084
bash-3.2$ ./xcpp-thread-race
expected: 200000 Count: 106771
bash-3.2$ ./xcpp-thread-race
expected: 200000 Count: 107119
bash-3.2$
```

- C++ threads
  - race conditions
  - unpredictable behavior??

- quadrature
  - Riemann sum
  - error??

$$\int_a^b f(x)\,dx = \lim_{\|\Delta x\|\to 0} \sum_{i=1}^{n} f(x_i^*)\,\Delta x_i$$

$$[a,b] \qquad \Delta x = \frac{b-a}{n}$$



$$a,\ a+\Delta x,\ a+2\Delta x,\ \ldots,\ a+(n-2)\Delta x,\ a+(n-1)\Delta x,\ b$$

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

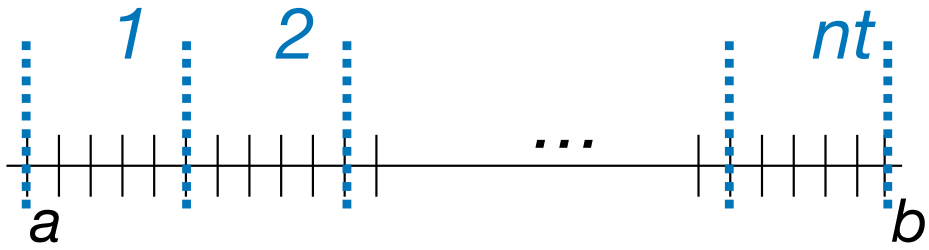
```cpp
double f(double x)
{
    return std::sin(x); // function to integrate
}

double riemann_sum(double a, double b, int n)
{
    double h = (b - a) / n;
    double sum = 0.0;
    for (int i = 0; i < n; ++i)
    {
        double x = a + i * h;
        sum += f(x);
    }
    sum *= h;
    return sum;
}
```

sequential discretization: regular mesh

Kenneth J. Roche
University of Washington

```cpp
int main(int argc, char *argv[])
{
    const double a = 0.0;
    const double b = 1.0;
    const int n = 100000000;

    // timer foo
    auto start = std::chrono::high_resolution_clock::now();
    auto stop = std::chrono::high_resolution_clock::now();
    auto duration = std::chrono::duration_cast<std::chrono::nanoseconds>(stop - start);

    if (argc != 2)
    {
        std::cerr << "Usage: " << argv[0] << " nthreads" << std::endl;
        return 1;
    }
    const int num_threads = std::atoi(argv[1]);

    // Compute Riemann sum sequentially
    start = std::chrono::high_resolution_clock::now();
    double seq_sum = riemann_sum(a, b, n);
    stop = std::chrono::high_resolution_clock::now();
    duration = std::chrono::duration_cast<std::chrono::nanoseconds>(stop - start);
    long double elapsed_time = (duration.count() * 1.e-9);
    std::cout << "Sequential:\t" << seq_sum << "\ttime: " << elapsed_time << std::endl;

    // Compute Riemann sum in parallel
    start = std::chrono::high_resolution_clock::now();
    double par_sum = parallel_riemann_sum(a, b, n, num_threads);
    stop = std::chrono::high_resolution_clock::now();
    duration = std::chrono::duration_cast<std::chrono::nanoseconds>(stop - start);
    elapsed_time = (duration.count() * 1.e-9);
    std::cout << "Parallel:\t" << par_sum << "\ttime: " << elapsed_time << std::endl;

    return 0;
}
```

// plan: partition the domain for evaluation

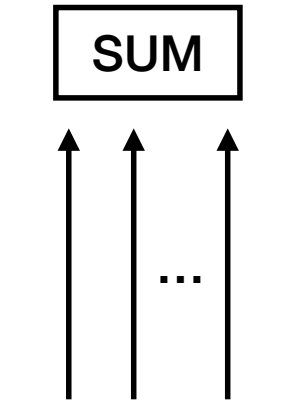


Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

```
double parallel_riemann_sum(double a, double b, int n, int num_threads)
{
    double h = (b - a) / n;
    double sum = 0.0;
    std::vector<std::thread> threads(num_threads);

    // Mutex for protecting the sum variable
    std::mutex sum_mutex;

    // Spawn threads to compute partial sums
    for (int i = 0; i < num_threads; ++i)
    {
        threads[i] = std::thread(compute_partial_sum, std::ref(sum), std::ref(sum_mutex), a, h, n, num_threads, i);
    }

    // Wait for threads to finish
    for (int i = 0; i < num_threads; ++i)
    {
        threads[i].join();
    }

    return sum;
}
```

- function to the *std::thread* constructor (can use a *lambda* expression)
- *std::ref* is used to pass *sum* and *sum_mutex* by reference to the function

```cpp
void compute_partial_sum(double &sum, std::mutex &sum_mutex, double a, double h, int n, int num_threads, int i)
{
    double partial_sum = 0.0;
    int start = (n / num_threads) * i;
    int end = (i == num_threads - 1) ? n : (n / num_threads) * (i + 1);
    for (int j = start; j < end; ++j)
    {
        double x = a + j * h;
        partial_sum += f(x);
    }
    partial_sum *= h;

    // Lock the mutex and update the sum variable
    sum_mutex.lock();
    sum += partial_sum;
    sum_mutex.unlock();
}
```



mutex to lock a variable in memory

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025

- concurrency and speedup!

```
bash-3.2$ g++ -o xcpp-thread-quadrature -std=c++14 cpp-thread-quadrature.cpp
bash-3.2$ ./xcpp-thread-quadrature
Usage: ./xcpp-thread-quadrature nthreads
bash-3.2$ ./xcpp-thread-quadrature 1
Sequential:     0.459698        time: 1.48345
Parallel:       0.459698        time: 1.53519
bash-3.2$ ./xcpp-thread-quadrature 2
Sequential:     0.459698        time: 1.55284
Parallel:       0.459698        time: 0.804145
bash-3.2$ ./xcpp-thread-quadrature 4
Sequential:     0.459698        time: 1.52221
Parallel:       0.459698        time: 0.460617
bash-3.2$ ./xcpp-thread-quadrature 6
Sequential:     0.459698        time: 1.53839
Parallel:       0.459698        time: 0.318315
bash-3.2$ ./xcpp-thread-quadrature 8
Sequential:     0.459698        time: 1.36719
Parallel:       0.459698        time: 0.22421
bash-3.2$
```

End Lecture 15

Kenneth J. Roche
University of Washington
AMATH 483 / 583 High Performance Scientific Computing
Spring Quarter 2025