Rounding Error Analysis

Our discussion of floating point arithmetic lead to the fundamental result:

\[ x \otimes y = (x \times y) (1 + e) \quad |e| < 2^{-p} \]

We shall assume "round to nearest" so \(e = 2^{-p}\)

\(x \otimes y\) gives the correctly rounded result for any two operands \(x, y \in F\) (the set of floating point numbers).

For any real number \(z\), we define

\[ \hat{f}(z) \in F \]

to be the correctly rounded (in our case nearest) number in \(F\) to \(z\). We have

\[ z \in F \Rightarrow \hat{f}(z) = z \]

\[ |z| \leq N_{max} \Rightarrow \hat{f}(z) = z(1 + e) \quad |e| \leq 2^{-p} \]

(if subnormals included)

\[ |z| > N_{max} \Rightarrow \hat{f}(z) = \text{sign}(z) \cdot \infty \]
Thus \( x \circ y = \text{fl} \ (x \circ y) \).

If we have a sequence of operations such as
\[
s = x^T y, \quad x, y \in F^n
\]
\( (F^n \) stands for vectors of length \( n \) with floating point entries \( ) \)

We write
\[
s = \text{fl} (x^T y)
\]
to represent the floating point result of the algorithm for computing \( x^T y \)

\[
S = 0
\text{ for } j = 1 : n
\quad S = S + x(j) \cdot y(j)
\text{ end}
\]

**REMEMBER:** Order is important.
Computing the inner product in a different order may (probably will) yield a different value in the final \( S \).
If \( S_j \) denotes the value in \( s \) at the end of the \( j \)-th instance of the loop body then

\[
S_j = f_l( S_{j-1} + x(j) \cdot y(j) )
\]

\[
= S_{j-1} \cdot x(j) \cdot y(j) , \quad S_0 = 0.
\]

From the fundamental result

\[
S_j = \left[ S_{j-1} + x(j) \cdot y(j) \cdot (1+\varepsilon_j) \right](1+\varepsilon_j)
\]

where \( |\varepsilon_1| , |\varepsilon_j| \leq \varepsilon \)

Thus

\[
S_1 = x(1) \cdot y(1) \cdot (1+\varepsilon_1)
\]

\[
S_2 = \left[ S_1 + x(2) \cdot y(2) \cdot (1+\varepsilon_2) \right](1+\varepsilon_2)
\]

\[
= x(1) \cdot y(1) \cdot (1+\varepsilon_1)(1+\varepsilon_2) + x(2) \cdot y(2) \cdot (1+\varepsilon_2)(1+\varepsilon_2)
\]

\[
S_3 = \left[ S_2 + x(3) \cdot y(3) \cdot (1+\varepsilon_3) \right](1+\varepsilon_3)
\]

\[
= x(1) \cdot y(1) \cdot (1+\varepsilon_1)(1+\varepsilon_2)(1+\varepsilon_3) + x(2) \cdot y(2) \cdot (1+\varepsilon_2)(1+\varepsilon_3) \]

\[
+ x(3) \cdot y(3) \cdot (1+\varepsilon_3)(1+\varepsilon_3)
\]
Continuing this way, we find

\[ S_n = \sum_{j=1}^{n} x(j) y(j) (1+\varepsilon_j) \frac{\prod_{i=j}^{n} (1+\varepsilon_i)}{i=j} \]

where we have defined \( \varepsilon_j = 0 \), and \( 1 \leq j, k \leq n \).

Let

\[ M_j \equiv (1+\varepsilon_j) \frac{\prod_{i=j}^{n} (1+\varepsilon_i)}{i=j} - 1 \]

Then

\[ (1) \quad S_n = \sum_{j=1}^{n} x(j)y(j)y(j)(1+M_j) \quad \text{\{Backward Error\}} \]

\[ (2) \quad S_n = x^T y + z \quad \text{\{Forward Error\}} \]

where \( |z| = \left| \sum_{j=1}^{n} x(j)y(j) \right| \)

\[ \leq n \mu \sum_{j=1}^{n} |x(j)||y(j)| \]

and \( \mu \geq \max |y_j| \).

(you will see why \( \mu \) is included in a moment)
In (1) we have expressed the computed result as the exact result obtained from perturbed data. Namely:

\[ S_n = \hat{x}^T \hat{y} \]

where \( \hat{x} = x \), and \( \hat{y} \) has components \( \hat{y}(j) = y(j) (1 + \epsilon(j)) \).

This is an example of Backward Error Analysis.

**Computed Result** = **Exact Result on Perturbed Data**

In (2) we express the computed result as the true result (on original data) plus a perturbation that resulted from round off:

\[ S_n = \hat{x}^T \hat{y} + \xi \]

This is **Forward Error Analysis**.
The computed $S_n$ and the floating point errors made along the way are the same in both cases. These are just two very different ways of interpreting the error.

If we can show that the errors $\epsilon_j$ are small (i.e. on the order of $\epsilon$) then our backward error analysis will show that the computed result $S_n$ is the exact inner product of two slightly perturbed vectors $x, y$

with $\|x - x\| = 0$, $\|y - y\| = \epsilon_{max}$

where $\epsilon_{max} = \max\{\|x\|, \|y\|\}$, 1, 2, or $\infty$ norm.

In fact we can show that if $n$ is not extremely large, then

$$M \leq \epsilon (1 + 2^{-k})$$

as long as $n \leq 2^{p - k + 1}$.

With this result, we say that the algorithm for computing $S = (x^T y)$
An algorithm \( f_A(x) \) for computing \( f(x) \) is **Backward Stable** if

\[
\frac{f_A(x) - f(x)}{\|x\|} = O(\epsilon)
\]

where \( |\epsilon| \leq \kappa \) for some modest constant \( \kappa \).

An algorithm \( f_A(x) \) for computing \( f(x) \) is **Forward Stable** if

\[
f_A(x) = f(x) + \epsilon
\]

where \( \|\epsilon\| = O(\epsilon)\|f(x)\| \).

Is our algorithm for computing \( f(x,y) \) forward stable?
The answer to this question depends upon \( x, y \).
In general, we must say no, since
\[
\sum_{j=1}^{n} x_j y_j \geq \left| \sum_{j=1}^{n} x_j y_j \right|
\]
is possible.

Example: \( e^t = 1 + t + \frac{t^2}{2} + \frac{t^3}{6} + \ldots + \frac{t^n}{n!} \)
\[
e^{-t} = 1 - t + \frac{t^2}{2} - \frac{t^3}{6} + \ldots + (-1)^n \frac{t^n}{n!},
\]
\[
e^t = x^T y, \quad x^T = (1, 1, \ldots, 1), \quad y^T = (1, 1, \frac{e^2}{2}, \ldots, \frac{e^n}{n!})
\]
\[
e^{-t} = w^T y, \quad w^T = (1, 1, \frac{1}{2}, \frac{1}{3!}, \ldots, \frac{1}{n!})
\]
Let us introduce the notation \( |x| = (|x_1|, |x_2|, \ldots, |x_n|) \)
(Replace components by their abs. value)
For \( t = 20 \), \( e^t = 4.9 \times 10^8 \), \( e^{-t} = 2 \times 10^{-9} \)
\[
e^t \cdot \mathbb{E}(x^T y) = x^T y + \varepsilon_1, \quad 1 \geq \mathbb{E} \left| x^T y \right| = \varepsilon_2 x^T y
\]
\[
2 \cdot x^T y (1 + \varepsilon_1)
\]
\[
e^{-t} \cdot \mathbb{E}(w^T y) = w^T y + \varepsilon_2, \quad 1 \geq \mathbb{E} \left| w^T y \right| = \varepsilon_2 w^T y
\]
Since \( \varepsilon_2 |w| = x. \)
Thus the forward error in computing $e^{-t}$ is about equal to $e e^{-t}$.

We have yet to show $m$ is small in (1).

Well, $|1 + \xi_1| = 1/1 + \xi_2| \leq (1 + \xi)^n$.

We leave the proof of the following result as an exercise:

Lemma: If $n < 2^{p-(k+1)}$,

Then $(1 + \xi)^n = 1 + \xi n$, $|\xi n| \leq n \xi (1 + 2^{-k}) = n \mu n$

Hint: Use LM and Mean Value Theorem to show $(1 + \xi)^n < 1 + 2^{-k}$

Then use MVT on $(1 + \xi)^n$ to get the final result.

Note: If $k = 3$, $p = 24$, then $1 + 2^{-k} = 1 + \frac{1}{8} = \frac{9}{8}$

and $2^{p-(k+1)} = 2^{20} = 1,048,576$
Summary:

**Backward Error**

\[ f_b(x^Ty) = x^Ty, \quad \|x - \bar{x}\| = 0, \quad \frac{\|y - \bar{y}\|}{\|y\|} \leq \epsilon, \quad \mu = E(1 + 2^{-k}) \]

**Forward Error**

\[ f_c(x^Ty) = x^Ty + z, \quad 121 \leq \kappa_n(1 + \|x\|) \]

More on \( \| \) notation:

We say \( 1x1 \leq 1y1 \) if \( 1xy1 \leq 1y11 \), i.e., \( 1z1 \leq 1y1 \)

If \( A, B \in \mathbb{R}^{n \times m} \), then:

\[ |A| = (|a_{ij}|) \]

\[ |A| \leq |B| \Leftrightarrow |a_{ij}| \leq |b_{ij}| \]

Note \( |AB| \leq |A||B| \) (when \( AB \) defined)
It is now straightforward to extend our results for inner product to other basic operations

1) \( f_k(a \cdot x + y) = a \cdot x + y + z \)
\[ |z| \leq \mu(1a1|x| + 1y1) \]

Let \( a \in F \) \( x, y \in F^n \)

2) \( f_k(Ax) = Ax + z \)
\[ |z| \leq \mu(1A1|x|) \]

Let \( A \in \mathbb{R}^{m \times n} \) \( x \in F^n \)

3) \( f_k(AB) = AB + z \)
\[ |z| \leq \mu(1A1|B|) \]

Where \( A \in \mathbb{F}^{m \times n} \) \( B \in \mathbb{F}^{n \times n} \)

4) \( f_k(\|x\|_2) = \|x\|_2 (1 + \delta) \)
\[ |1 + \delta| \leq (\sqrt{1 + \mu}) (1 + \varepsilon) \]

(\text{Remember } \approx \text{ is correctly rounded in IEEE})
Why Backward Error Analysis?

We are ultimately interested in solving linear systems and thus we would like to assess the accuracy of the computed answer.

Thus we would like \( f \) such that

\[
Af = b
\]

where \( A \in \mathbb{F}^{n \times n} \), \( b \in \mathbb{F}^n \). (I'm using \( f \) for the moment to avoid confusion with \( x \) in the defn. of backward stability.)

Mathematically \( f = f(AB) = A^{-1}b \). Note that even if we could do this exactly, it is unlikely that \( f \in \mathbb{F}^n \). Let \( f^* \) be the floating point result of applying the algorithm \( f^*(A, b) = f \).

For HSL QR we would

1) Factor \( A = QR \)

2) Put \( c = Q^Tb \)

3) Solve \( Rf = c \)

This sequence of steps done in floating point gives \( f^* = f^*(A, b) \).
We shall soon show that
\[
\hat{f} = \hat{f}_*(A,b) = f(A+E,b)
\]
\[\text{ie } (A+E)\hat{f} = b \quad \text{with } ||E|| \leq \epsilon, ||A||
\]
This will show the HB Q-R method is backwards stable.

This does not imply anything about the accuracy of \( \hat{f} \) as an approximation to \( f \).

For this, we need the perturbation theory. We showed that
\[
\frac{||\hat{f} - f||}{||f||} \leq \frac{K(A) ||E||}{1 - K(A) ||E||} \quad \text{(any } b \text{)}
\]
\[
= \left[ \frac{K(A)}{1 - K(A) ||E||} \right] \mathcal{O}(E) \approx K(A) \mathcal{O}(E)
\]
where \( K(A) = ||A|| \cdot ||A^{-1}|| \).

Backward Stable \Rightarrow \text{Well Conditioned} \Rightarrow \text{Accurate Answer}

If \( K(A) \) is large, accuracy is lost.
When we do rounding error analysis, as we did with inner product, we generally encounter terms of the form

\[ Z = (1 + \varepsilon_1)(1 + \varepsilon_2) \cdots (1 + \varepsilon_n), \quad \varepsilon_j = O(\varepsilon) \]

We have seen that so long as \( n \) is not extremely large, it is safe to say

\[ Z = 1 + \xi, \quad \xi = O(\varepsilon) \]

Occasionally, we see terms such as

\[ r = \frac{1}{1 + \delta}, \quad \delta = O(\varepsilon). \]

Again, it is safe to say

\[ r = 1 + \rho, \quad |\rho| = O(\varepsilon). \]

For, if we put

\[ 1 + \rho = \frac{1}{1 + \delta} \]

and solve for \( r \), we get

\[ \rho = \frac{-\delta}{1 + \delta} = O(\varepsilon) \]
If we wish, we can keep track of every detail, as we did with inner product. If we were publishing a new result, we probably would do this. But, as you have seen, this gets tedious even in the simplest of cases, e.g., inner product.

To avoid clouding the main issues with this kind of detail, we shall use $O(\varepsilon)$ notation. We need to exercise some care to avoid inadvertently introducing a large constant and still calling the term $O(\varepsilon)$.

We shall need to use facts about manipulation such as

$$O(\varepsilon)(1 + O(\varepsilon)) = O(\varepsilon)$$

$$\left(1 + O(\varepsilon)\right)\left(1 + O(\varepsilon)\right) = 1 + O(\varepsilon)$$

$$\frac{1 + O(\varepsilon)}{1 + O(\varepsilon)} = 1 + O(\varepsilon)$$

Remember: If $\varepsilon = 1 + \varepsilon$, $\varepsilon > 0$.

All we know is $|\varepsilon| < \varepsilon$

So $\frac{1 + O(\varepsilon)}{1 + O(\varepsilon)} = \frac{1 + \varepsilon}{1 + \varepsilon} = 1 + O(\varepsilon)$
Stability of Householder's Method:

1) Represent \( Q = I - 2 \frac{vv^T}{vv^T} \)

For a given vector \( v \in F^n \), we can represent the H.H. transformation with the vector \( v \) together with

\[
T = \frac{2}{vv^T}(v) = \frac{2(1+\eta)}{vv^T(1+\delta)}, \quad \eta = 0(\varepsilon), \quad \delta = 0(\varepsilon)
\]

So \( T = \frac{2}{vv^T}(1 + O(\varepsilon)) \)

2) Apply \( Q = I - 2 \frac{vv^T}{vv^T} \) to \( a \in F^n \)

Now that we have \( v \) and \( T \) computed and stored, we can apply \( Q \) to an arbitrary vector \( a \in F^n \) with the following steps

1) \( \theta = \frac{2}{vv^T}(va) = va + \varepsilon_1, \quad |\varepsilon_1| \leq \eta \mu \| v \|_2 \| a \|_2 \)

2) \( \theta = \frac{2}{vv^T}(\theta) = \theta(1+\varepsilon_2), \quad |\varepsilon_2| \leq \delta \)

3) \( w = \frac{2}{vv^T}(a-v\theta) = a - v\theta + \varepsilon \\
\| \varepsilon \| \leq 2\mu (|a| + \|v\|_1 \|a\|_1) \)
After completion of step 2

\[ \Theta = \frac{2}{V^2} (1 + O(\varepsilon)) (v^T a + \xi) \]

\[ = \frac{2v^T a}{V^2} (1 + O(\varepsilon)) + \frac{2}{V^2} (1 + O(\varepsilon)) \]

After step 3

\[ W = a - v \xi + \varepsilon \]

\[ = a - 2\frac{v^T a}{V^2} - 2v \frac{v^T a}{V^2} O(\varepsilon) \]

\[ - 2v \frac{v^T a}{V^2} O(\varepsilon) \]

\[ + \varepsilon \]

Now,

\[ ||W - a||_2 \leq 2\left( \frac{||v^T a||_2}{||v||_2^2} + \frac{||v^T a||_2}{||v||_2^2} \right) O(\varepsilon) \]

\[ + ||\varepsilon||_1 \]

But, \[ ||\varepsilon||_1 \leq (||a||_1 + ||v||_1 \xi) 2\varepsilon \]

and \[ ||\varepsilon||_1 \leq 2 \frac{||v^T a||_2}{||v||_2^2} (1 + O(\varepsilon)) \]

So \[ ||\varepsilon||_1 \leq ||a||_1 O(\varepsilon) \]

Thus \[ ||W - a||_2 \leq ||a||_2 O(\varepsilon) \]

computed \[ \frac{||W - a||_2}{||a||_2} = \frac{||W - a||_2}{||a||_2} = O(\varepsilon) \]
This is the best we could hope for.

In practice we must scale \( v \) so that \( v^T a \) does not overflow unnecessarily. For most purposes, simply replacing \( v \rightarrow \frac{v}{\|v\|} \)

would be fine. However, to be extra careful you might want to scale \( v \) by a power of the base \( z \): \( v \rightarrow \frac{v}{z^m} \)

where \( m \) is an integer s.t. \( \|v, z^m\| = 1 \)

and then use the previous algorithm. Since this scaling only shifts the exponents there is no rounding (unless a normal component is sent to a subnormal).

**Orthogonality of \( Q \)**

Let \( Q = I - z v v^T \)

\[
Q^T Q = (I - z v v^T)(I - z v v^T)
\]

\[
= I - (z v v^T - 2) + z v v^T
\]

Since this scaling only shifts the exponents there is no rounding (unless a normal component is sent to a subnormal).

\[
\|z v v^T\| = (1 + o(\epsilon))\frac{2}{z^T v v^T} = 2(1 + o(\epsilon))
\]

Thus \( \|E\| = O(\epsilon) \).
This shows the representation of \( Q \) is very nearly orthogonal.

When we apply \( Q^T = Q \) to a matrix \( A \), we apply it column by column and see that

\[
fe(Q^T A) = Q^T A + F, \quad \|F\| = \|A\| \Omega(\varepsilon)
\]

Thus the floating point computation of \( A^T A \) factorization is

\[
\begin{aligned}
&\text{for } j = 1 : n \\
&\quad 1) \text{ Compute } Q_j \\
&\quad 2) A \leftarrow Q_j^T A = Q_j^T A + F_j \\
&\end{aligned}
\]

At the end of this \( A \) contains an upper triangular \( R \in \mathbb{R}^{n \times n} \) such that

\[
Q^T (A + F) = R
\]

\[
Q = Q_1, Q_2, \ldots, Q_n
\]

Note: Each \( Q_j \) is the representation of \( I - 2 \frac{u_j v_j^T}{\|u_j\|^2} \).
Let $A_j$ denote the contents of $A$ after the $j$-th instance of the loop body. Then

$$A_j = Q_j^T A_{j-1} + F_j, \quad A_0 = A$$

So,

$$A_1 = Q_1^T A + F_1$$

$$A_2 = Q_2^T Q_1^T A + Q_2^T F_1 + F_2$$

$$A_3 = Q_3^T Q_2^T Q_1^T A + Q_3^T Q_2^T F_1 + Q_3^T F_2 + F_3$$

$$R = A_n = Q^n A + \sum_{j=1}^{n} \left( \prod_{i=j}^{n} Q_i^T \right) F_j$$

Now, using the property $Q_i^T Q_i = I + E_i$, we can easily see that $\| \left( \prod_{i=j}^{n} Q_i^T \right) F_j \| = \| F_j \| (1 + o(\epsilon))$

and since $\| F_j \| = \| A \| O(\epsilon)$, we have

$$R = Q^n A + \hat{E}, \quad \| \hat{E} \| = \| A \| O(\epsilon)$$

where $\hat{E} = \sum_{j=1}^{n} \left( \prod_{i=j}^{n} Q_i^T \right) F_j$. Put $F = Q \hat{E}$.

To see the final result:
The H.H. Q-R factorization produces a computed \( Q \) (kept in product form) and an upper triangular \( R \) such that
\[
Q^T(A+F) = R
\]

i.e.
\[
A+F = QR
\]

and with
\[
Q^TQ = I + E_q \quad \|E_q\| = O(\varepsilon)
\]

Note: Everything here is also valid for \( A \in \mathbb{F}^{m \times n} \) with \( m \geq n \).

For \( m = n \), the steps are

1) Compute \( QR = A+F \)

2) Form \( c = (Q^Tb) = QTb + \tilde{z} \quad \|\tilde{z}\| = O(\varepsilon)\)

3) Solve \( Rx = c \)

If we can show that the last step gives a computed solution \( \hat{x} \) such that
\[
(R+G) \hat{x} = c \quad \text{with} \quad \|G\| = O(\varepsilon) \quad \|A\| = O(\varepsilon)
\]
then

\[(R+G)\hat{x} = \hat{Q}^T b + \hat{z}\]

\[(QR + AG)\hat{x} = b + \hat{Q}^T\hat{z}\]

\[(A + F + AG)\hat{x} = b + \hat{Q}^T\hat{z}\]

\[(A + E)\hat{x} = b + \hat{z}\]

where \[\|E\|_2 = \|A\|_2 O(\varepsilon)\] and \[\|E\|_2 = \|A\|_2 O(\varepsilon)\]

From our perturbation theory,

\[\frac{\|\hat{x} - x\|_2}{\|x\|_2} = \kappa(A) O(\varepsilon)\]

The constant \(\kappa(A) O(\varepsilon)\) is a slow order function of \(n\).

In the above expressions,

\[E = F + AG\]

and \[\hat{z} = \hat{Q}^T\hat{z}\]

\[\|E\|_2 = \|F\|_2 + \|AG\|_2 \leq \|A\|_2\|\hat{Q}\|_2 + \|G\|_2\]

\[\|E\|_2 = \|A\|_2 O(\varepsilon)\] and \(\|E\|_2 = \|A\|_2 O(\varepsilon)\)

\[\|\hat{z}\|_2 = \|Q\|_2 \|\hat{z}\|_2 \|\hat{Q}\|_2 = \|Q\|_2 O(\varepsilon)\].
Error Analysis of Triangular Systems

Recall the backsubstitution algorithm for solving upper triangular systems of equations

\[ Rx = c. \]

We want to analyze the rounding error. To do this, we shall express the algorithm in a slightly different form

\[
\begin{align*}
x &= 0 \\
\text{for } j &= n:1:1 \\
1) \quad x(j) &= (c(j) - x(j))/R(j,j) \\
2) \quad x(1:j-1) &= x(1:j-1) + R(1:j-1,j)^*x(j) \\
\end{align*}
\]

end

Note that

\[ x(1) = [c(1) - R(1,2:n) * x(2:n)] / R(1,1) \]

where \( x(2:n) \) has been computed through steps \( j = n, n-1, \ldots, 2 \).

The inner product \( R(1,2:n) * x(2:n) \) has been computed in reverse order and this is subtracted from \( c(1) \) just prior to division by \( R(1,1) \).
To analyze this, let us partition the system

\[ R = \begin{pmatrix} P & r^T \\ 0 & R \end{pmatrix}, \quad X = \begin{pmatrix} \hat{x} \\ x \end{pmatrix}, \quad C = \begin{pmatrix} \hat{c} \\ c \end{pmatrix} \]

Here, \( p = R(1,1) \), \( r^T = R(1,2:n) \)
\( \hat{x} = x(1) \), \( \hat{X} = x(2:n) \)
\( y = C(1) \), \( c = C(2:n) \)

\( \hat{X} \) represents the computed solution just after completion of loop instance \( j=2 \).

Then

\[ \hat{x} = \text{fl} \left( \frac{y - r^T \hat{X}}{\rho} \right) \]

\[ = \text{fl} \left( \frac{y - r^T (I + D) \hat{X}}{\rho(1 + \delta_1)} \right) \]

\[ = \frac{1}{\rho(1 + \delta_1)(1 + \delta_1)} \left[ y - r^T (I + D) \hat{X} \right] \]

Here, we have expressed the error of subtraction \( \hat{X} = \text{fl} (y - r^T (I + D) \hat{X}) \) as \( \frac{1}{1 + \delta_1} \)
and the error of division

\[ \text{fl} \left( \frac{\hat{X}}{\rho} \right) = \frac{\hat{X}}{\rho(1 + \delta_1)} \]

and it is easily seen that

\[ (1 + \delta_1)(1 + \delta_1) = 1 + \mu, \quad \mu, 1 < 2 \mu. \]
Note \( r^T (I+D) \hat{x} = \sum_{j=1}^{n} r(j) \hat{x}(j)(1+\delta_j) \)

where \( D = \text{diag}(5, \ldots, 5_n) \) and \( 1 \leq j \leq n \).

Moreover, \( \tilde{x} \) is the computed solution to the \((n-1)\) by \((n-1)\) upper triangular system
\[
\tilde{R} \tilde{x} = \tilde{c}.
\]

Suppose that \( \tilde{x} \) exactly solves
\[
(\ast) \quad (\tilde{R} + \tilde{G}) \tilde{x} = \tilde{c}
\]

with \( |e_j^T \tilde{G}| \leq \{(n-1)j + 1\} \leq e_j^T \tilde{R} \), \( 1 \leq j \leq n-1 \).

(Note that this implies \( \tilde{G} \) is upper triangular.)

Combining \((\ast)\) and \((\ast\ast)\) and \((1+\delta)(1+\delta)2/\lambda_N\)
gives
\[
\begin{bmatrix}
\rho(1+\delta) & r^T (I+D) \\
0 & \tilde{R} + \tilde{G}
\end{bmatrix}
\begin{bmatrix}
\hat{X} \\
\tilde{X}
\end{bmatrix}
= 
\begin{bmatrix}
\hat{C} \\
\tilde{C}
\end{bmatrix}
\]

or
\[
[R + G] \hat{x} = \tilde{c},
\]

where
\[
G = \begin{bmatrix}
\rho a_1 & r^T D \\
0 & \tilde{G}
\end{bmatrix}.
\]
with \( |e_j^T G| = [0, |e_j^T \hat{G}|] \)

\[
\leq [0, (n-1)-(j-1)+1) \mu |e_j^T \hat{R}| \\
= (n-j+1) \mu |e_j^T R| 
\]

for \( 2 \leq j \leq n \), and with

\[
|e_j^T G| = [1 \mu |e_j^T R|] \\
\leq n \mu |e_j^T R| .
\]

This establishes (with the trivial case \( n=1 \)) an inductive proof of the following result.

**Lemma:** Let \( \hat{x} \) be the computed solution to

\[ R \hat{x} = c \]

where \( R \in \mathbb{F}^{n \times n} \) is upper triangular and nonsingular, and where \( c \in \mathbb{F}^n \). Then there is an upper triangular \( G \in \mathbb{F}^{n \times n} \) such that

\[
(R + G) \hat{x} = c
\]

with \( |e_j^T G| \leq (n-j+1) \mu |e_j^T R| \)

Or, more crudely stated \( |G| \leq n \mu |R| \). \( \square \)