MMSE FOM Speed Up commit request

COM Commit Request 4p9_3
(Resubmit of Request 4p8_9)

Adam Gregory, Samtec

Background

- This commit request (4p9_3) is identical to the previous 4p8_9 request. That request was deferred so that people would have an opportunity to test the code changes and verify the output results.

Purpose

- ❑ Improve COM runtime in MMSE_FOM
- ❑ Since MMSE_FOM is the inner most piece of the optimization loops, any small improvement can have drastic impact on overall runtime
- ❑ Observed up to 25% runtime improvement with these changes
  - Actual improvement depends on many factors, but the 2 most impactful are:
    1. Length of pulse response
    2. RxFFE Floating taps

Commit request: 3 changes to speed up MMSE_FOM

- ❑ MMSE.m: Remove trailing zeros from H
  - The size reduction makes the Toeplitz and many operations in MMSE_FOM faster
- ❑ MMSE_FOM.m: Compute w transpose once instead of 3 times
- ❑ MMSE_FOM.m: Reduce runtime of wbl calculation by using linear algebra simplifications

MMSE.m: Remove trailing zeros from H

- If h has length less than num_ui, trailing zeros are added that have no effect on computations used for H
- MMSE_FOM.m contains a multiplication of H * transpose(H). Reducing the size of H significantly speeds up this operation

src/MMSE.m

```
@@ -90,7 +90,13 @@ end

  hc1=[ h  zeros(1,Nw-1) ];
  hr1=[ h(1) zeros(1,Nw-1)];
- H=toeplitz(hc1,hr1);
+ if length(samp_idx) < num_ui && length(h) > length(samp_idx)+Nw
+     % If length of h is less than num_ui, can save a lot of time in floating tap calcuation
+     % by trimming off the zeros at the end.
+     H = toeplitz(h(1:length(samp_idx)+Nw-1),hr1);
+ else
+     H = toeplitz(hc1,hr1);
+ end
```

MMSE_FOM.m: Compute w transpose 1 time

- w transpose is used 3 times in the calculation of sigma_e
- Save time by pre-computing once
- This seems trivial, but this can easily reduce the number of transposes called by millions.

```
- sigma_e=sqrt(sigma_X2*(w'*R*w+1+b'*b-2*w'*h0'-2*w'*Hb'*b)); % Commit request 4p4_5 from healey_3dj_COM_01_240416
+ w_tr = w';
+ sigma_e=sqrt(sigma_X2*(w_tr*R*w+1+b'*b-2*w_tr*h0'-2*w_tr*Hb'*b)); % Commit request 4p4_5 from healey_3dj_COM_01_240416
```

MMSE_FOM.m: Reduce wbl runtime

- The nature of the wbl calculation allows for some linear algebra manipulations that reduces the size of the matrix inversion needed
- See explanation on following slides
- It is important to keep the original form of wbl documented since the final simplification bears little resemblance to the original.

```diff
- wbl= [ R  -Hb' -h0';...
-     -Hb  ib  zb'; ...
-     h0  zb   0]\[h0'; zb' ;1];
+ speedup_wb_calculation = 1;
+ if speedup_wb_calculation
+     % This block uses some linear algebra cancellations to reduce the runtime of wb calculation
+     % This comes from the fact that the beginning of the last column and last row of the large matrix
+     % are the same as the beginning of the vector:  [h0 0]
+     % divide the large matrix into a block matrix:
+     % A = [R  -Hb'; -Hb ib]
+     % C = [h0 zb]
+     % B = [-h0'; zb] = -C'
+     % D = 0
+     % now the equation is:  wbl = inv([A B; C D]) * [C'; 1]
+     % use block matrix inversion formulas (where only A is invertible):
+     % S = inv(D-C*inv(A)*B) = inv(C*inv(A)*C')
+     % inv([A B; C D]) = [ inv(A) + inv(A)*B*S*C*inv(A)    -inv(A)*B*S ;   -S*C*inv(A)   S]
+     % replace B by -C': [ inv(A) - inv(A)*C'*S*C*inv(A)    inv(A)*C'*S ;   -S*C*inv(A)   S]
+     % multiply this by [C';1] and note that S is inherently scalar since it involves 1xN * NxN * Nx1
+     % Final form simplifies to:
+     % [inv(A)*C';  1-1/S] * S
+     A = [R  -Hb'; -Hb ib];
+     C = [h0  zb];
+     Ct = C';
+     Z = A\Ct;
+     S_inv = C*Z;
+     wbl = [Z; 1-S_inv]/S_inv;
+ else
+     wbl= [ R  -Hb' -h0';...
+         -Hb  ib  zb'; ...
+         h0  zb   0]\[h0'; zb' ;1];
+ end
```

Wbl Explanation

- Represent wbl as inverse(X)*Y
- Divide X into 4 blocks which are themselves matrices
- A is NxN square matrix
- C is 1xN
- B is the negative transpose of C (Nx1)
- D = 0
- The zeros highlighted in red can be vectors of zeros. Doesn’t change any of the calculations.

$$w_{bl} = X^{-1}Y$$

$$X = \begin{bmatrix} R & -H_b^T & -h_0^T \\ -H_b & I & 0 \\ h_0 & 0 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} h_0^T \\ 0 \\ 1 \end{bmatrix}$$

Divide X into Blocks

$$A = \begin{bmatrix} R & -H_b^T \\ -H_b & I \end{bmatrix} \qquad B = \begin{bmatrix} -h_0^T \\ 0 \end{bmatrix} = -C^T$$

$$C = [h_0 \quad 0] \qquad D = 0$$

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A & -C^T \\ C & 0 \end{bmatrix}$$

Wbl Explanation

- Use the block matrix inversion formulas for the case where A is invertible
  - https://en.wikipedia.org/wiki/Block_matrix#Inversion
  - S is the inverse of the Schur complement
- The final form of the inverse of X is shown
- Note that S involves the multiplication of (1xN)(NxN)(Nx1), so it is scalar

Block Matrix Inversion Formulas (when A is invertible)

$$S = (D - CA^{-1}B)^{-1} = (CA^{-1}C^T)^{-1}$$

$$X^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BSCA^{-1} & -A^{-1}BS \\ -SCA^{-1} & S \end{bmatrix}$$

Replace with B with -C^T

$$X^{-1} = \begin{bmatrix} A^{-1} - A^{-1}C^TSCA^{-1} & A^{-1}C^TS \\ -SCA^{-1} & S \end{bmatrix}$$

Wbl Explanation

- ❑ Y can be represented in terms of transpose of C
- ❑ Simplify wbl from inverse(X)*Y
- ❑ The final form of wbl is significantly faster to calculate than the original form
  - Usually 10-20% faster

$$Y = \begin{bmatrix} h_0^T \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} C^T \\ 1 \end{bmatrix}$$

$$w_{bl} = X^{-1}Y = \begin{bmatrix} A^{-1} - A^{-1}C^TSCA^{-1} & A^{-1}C^TS \\ -SCA^{-1} & S \end{bmatrix} \begin{bmatrix} C^T \\ 1 \end{bmatrix}$$

$$w_{bl} = \begin{bmatrix} A^{-1}C^T - A^{-1}C^TSCA^{-1}C^T + A^{-1}C^TS \\ -SCA^{-1}C^T + S \end{bmatrix}$$

$$w_{bl} = \begin{bmatrix} A^{-1}C^T(I - SCA^{-1}C^T + S) \\ S(I - CA^{-1}C^T) \end{bmatrix}$$

Substitute S and replace Identity matrix with 1 since S is scalar

$$S = (CA^{-1}C^T)^{-1}$$

$$w_{bl} = \begin{bmatrix} A^{-1}C^T(1 - SS^{-1} + S) \\ S(1 - S^{-1}) \end{bmatrix} = \begin{bmatrix} A^{-1}C^TS \\ S - 1 \end{bmatrix}$$

$$\boxed{w_{bl} = \begin{bmatrix} A^{-1}C^T \\ 1 - \frac{1}{S} \end{bmatrix} \mathrm{S}}$$

Impact on Output

- ❑ The changes to H and wbl both cause minor deviations in the output structure. Usually on the order of 1e-14
- ❑ The change to wbl will change numerical values since a different matrix inversion is used.
- ❑ The change in output due to removing trailing zeros from H is surprising. Matlab appears to have some special operations for runtime improvement when multiplying a matrix by its transpose. The output of the transpose multiplication is different even though removing zeros should not change anything
  - This is tiny numerical differences on the order of 1e-18

Impact on Output

- ❑ Screenshot of results comparison
  - Only showing a small part of the report

```
Field "FOM" is different
Difference = -2.78355e-12
Field "sigma_N" is different
Difference = 5.91974e-17
Field "channel_operating_margin_dB" is different
Difference = -4.44089e-15
Field "available_signal_after_eq_mV" is different
Difference = 1.18661e-12
Field "steady_state_voltage_weq_mV" is different
Difference = 9.10916e-12
Field "Peak_ISI_XTK_and_Noise_interference_at_BER_mV" is different
Difference = 7.56728e-13
Field "peak_ISI_XTK_interference_at_BER_mV" is different
Difference = 4.82281e-13
Field "peak_ISI_interference_at_BER_mV" is different
Difference = 4.82281e-13
Field "equivalent_ICI_sigma_assuming_PDF_is_Gaussian_mV" is different
Difference = 1.36113e-13
Field "SNR_ISI_XTK_normalized_1_sigma" is different
Difference = -3.55271e-15
Field "SNR_ISI_est" is different
Difference = -2.41585e-13
Field "DFE_taps" is different
Difference = 9.74776e-14
```

Runtime Comparison Example

- Showing runtime of lines that are relevant to the code update.
- Original on the left. Update on the Right.
- Total runtime for this example was 160 seconds vs. 120 seconds
- Note: this is for about 400K calls to MMSE_FOM. That number can grow much larger with more EQ loops.

Original (left)

0.11	397593	56	`if param.N_bg ~= 0`
49.70	397593	57	`    H=H( :,[1:Nfix idx+param.RxFFE_cmx+1 ]);`
0.04	397593	58	`end`
		59	`%% HH and R`
25.47	397593	60	`HH= H'*H;`

16.44	397593	73	`wb1= [ R  -Hb' -h0';...`
	397593	74	`    -Hb  ib  zb'; ...`
	397593	75	`    h0  zb   0]\[h0'; zb' ;1];`

4.62	397593	111	`sigma_e=sqrt(sigma_X2*(w'*R*w+1+b'*b-2*w'*h0'-2*w'*Hb'*b));`

11.03	3366	93	`H=toeplitz(hc1,hr1);`

Update (right)

0.11	397593	56	`if param.N_bg ~= 0`
28.67	397593	57	`    H=H( :,[1:Nfix idx+param.RxFFE_cmx+1 ]);`
0.02	397593	58	`end`
		59	`%% HH and R`
16.81	397593	60	`HH= H'*H;`

3.08	397593	91	`    A = [R  -Hb'; -Hb ib];`
0.62	397593	92	`    C = [h0  zb];`
0.19	397593	93	`    Ct = C';`
9.02	397593	94	`    Z = A\Ct;`

0.23	397593	139	`w_tr = w';`
3.50	397593	140	`sigma_e=sqrt(sigma_X2*(w_tr*R*w+1+b'*b-2*w_tr*h0'-2*w_tr*Hb'*b));`

< 0.01	3366	93	`if length(samp_idx) < num_ui && length(h) > length(samp_idx)+Nw`
		94	`    % If length of h is less than num_ui, can save a lot of time`
		95	`    % by trimming off the zeros at the end.`
6.64	3366	96	`    H = toeplitz(h(1:length(samp_idx)+Nw-1),hr1);`

Changes to config

- ❑ Changes to config
  - None
- ❑ Changes to output
  - Tiny numerical differences
- ❑ Download beta test code
  - [Beta Test: MMSE Speedup]

Thank You!