Interconnect Modeling & Analysis
Time Moments of Impulse Response $h(t)$

- Definition of moments

  \[ h(t) \xrightarrow{L} H(s) \]

  \[
  H(s) = \int_0^\infty h(t)e^{-st} \, dt = \int_0^\infty h(t) \left( \sum_{i=0}^{\infty} \frac{1}{i!} (-st)^i \right) \, dt
  \]

  \[
  = \sum_{i=0}^{\infty} \frac{1}{i!} (-s)^i \int_0^\infty h(t) \cdot t^i \, dt
  \]

  \[ i\text{-th moment } m_i = \frac{1}{i!} (-1)^i \int_0^\infty h(t) \cdot t^i \, dt \]

- Note that $m_1$ = Elmore delay when $h(t)$ is monotone voltage response of impulse input
Derivatives of $h(t)$

- $H(s)$ can be expressed as a rational transfer function

$$H(s) = \sum_{j=1}^{N} \frac{k_j}{s - p_j}$$

- Expand $H(s)$ around $s \to \infty$

$$H(s) = h(0)s^{-1} + h^{(1)}(0)s^{-2} + \cdots + h^{(j)}(0)s^{-(j+1)} + \cdots$$

$$= \sum_{j=-\infty}^{\infty} -m_j s^j$$

Hint: \[ L\left(\frac{d^n h(t)}{dt^n}\right) = s^n H(s) - s^{n-1} h(0^-) - \cdots - s^{n-2} h^{(2)}(0^-) - h^{n-1}(0^-) \]
**q-th Pade Approximation**

- Pade approximation of type $(p/q)$:

\[
H_{p,q}(s) = \frac{b_p s^p + \cdots + b_1 s + b_0}{a_q s^q + \cdots + a_1 s + 1}
\]

\[= H(s) + O(s^{p+q+1})\]

- $q$-th Pade approximation ($q \ll N$):

\[
H_q(s) \equiv H_{q-1,q}(s) = \sum_{j=1}^{q} \frac{k_j}{s - p_j}
\]

- Formulate $2q$ constraints by matching $2q$ moments to compute $k_i$'s & $p_i$'s
General Moment Matching Technique

• Basic idea: match the moments $m_{-(2q-r)}$, $\ldots$, $m_{-1}$, $m_0$, $m_1$, $\ldots$, $m_{r-1}$

$$\hat{H}(s) = \frac{k_1}{s-p_1} + \frac{k_2}{s-p_2} + \cdots + \frac{k_q}{s-p_q}$$

$$= m_0 + m_1 s + \cdots + m_{r-1} s^{r-1} + O(s^r)$$

$$= -m_{-1} s^{-1} - \cdots - m_{-(2q-r)} s^{-(2q-r)} - O(s^{-(2q-r+1)})$$

• When $r = 2q-1$:
  (i) initial condition matches, i.e.
  $$\hat{h}(0^+) = h(0^+), \quad \text{or} \quad \lim_{s \to \infty} s \hat{H}(s) = \lim_{s \to \infty} s H(s)$$
  or $$(-\hat{m}_{-1} = -m_{-1})$$
  (ii) $\hat{m}_k = m_k$ for $k = 0, 1, \ldots, 2q - 2$
Compute Residues & Poles

\[ \frac{k_i}{s - p_i} = -\frac{k_i}{1 - s/p_i} = -\frac{k_i}{p_i} \sum_{j=0}^{\infty} \left( \frac{s}{p_i} \right)^j \]

\[ \begin{aligned}
&\begin{aligned}
&k_1 + k_2 + \cdots + k_q = h(0) = -m_{-1} \\
&\left( -\frac{k_1}{p_1} + \frac{k_2}{p_2} + \cdots + \frac{k_q}{p_q} \right) = m_0 \\
&\left( -\frac{k_1}{p_1^2} + \frac{k_2}{p_2^2} + \cdots + \frac{k_q}{p_q^2} \right) = m_1 \\
&\vdots \\
&\left( -\frac{k_1}{p_1^{2q-1}} + \frac{k_2}{p_2^{2q-1}} + \cdots + \frac{k_q}{p_q^{2q-1}} \right) = m_{2q-2}
\end{aligned}
\end{aligned} \]

\[ (= \lim_{s \to \infty} s \hat{H}(s)) \]

initial condition

match first 2q-1 moments

EQ1

condition initial
Basic Steps for Moment Matching

Step 1: Compute $2q$ moments $m_{-1}$, $m_0$, $m_1$, …, $m_{(2q-2)}$ of $H(s)$

Step 2: Solve $2q$ non-linear equations of EQ1 to get

$p_1, p_2, \cdots, p_q : \text{poles}$

$k_1, k_2, \cdots, k_q : \text{residues}$

Step 3: Get approximate waveform

$$\hat{h}(t) = k_1 e^{p_1 t} + k_2 e^{p_2 t} + \cdots + k_q e^{p_q t}$$

Step 4: Increase $q$ and repeat 1-4, if necessary, for better accuracy
Components of Moment Matching Model

• Moment computation
  – Iterative DC analysis on transformed equivalent DC circuit
  – Recursive computation based on tree traversal
  – Incremental moment computation

• Moment matching methods
  – Asymptotic Waveform Evaluation (AWE) [Pillage-Rohrer, TCAD’90]
  – 2-pole method [Horowitz, 1984] [Gao-Zhou, ISCAS’93]...
Basis of Moment Computation by DC Analysis

• Applicable to general RLC networks

• Used in Asymptotic Waveform Evaluation (AWE) [Pillage-Rohrer, TCAD’90]

• Represent a lumped, linear, time-invariant circuit by a system of first-order differential equations:

\[ C\dot{x} = -Gx + bu \]

\[ y = l^T x \]

where \( x \) represents circuit variables (currents and voltages)

\( G \) represents memoryless elements (resistors)

\( C \) represents memory elements (capacitors and inductors)

\( bu \) represents excitations from independent sources

\( y \) is the output of interest
Transfer Function

- Assume zero initial conditions and perform Laplace transform:
  \[sCX = -GX + bU\]
  \[Y = l^T X\]
  where \(X, U, Y\) denote Laplace transform of \(x, u, y\), respectively

- Transfer function:
  \[H(s) = Y(s)/U(s) = l^T (G + sC)^{-1}b\]

- Let \(s = s_0 + \sigma\), where \(s_0\) is an arbitrary, but fixed expansion point such that \(G+s_0C\) is non-singular
  \[H(s_0 + \sigma) = l^T (I - \sigma A)^{-1}r\]
  where \(A = -(G + s_0 C)^{-1}C\), \(r = (G + s_0 C)^{-1}b\)
Taylor Expansion and Moments

• Expansion of $H(s)$ about $\sigma = 0$:

$$H(s_0 + \sigma) = l^T (I + \sigma A + \sigma^2 A^2 + \cdots) r$$

$$= \sum_{k=0}^{\infty} m_k \sigma^k \quad \text{where} \quad m_k = l^T A^k r$$

• Recursive moment computation:

$$u_0 = r$$

$$u_1 = Ar$$

$$u_2 = A^2 r$$

$$\vdots$$

$$\implies (G + s_0 C) u_0 = b \iff u_0 = (G + s_0 C)^{-1} b$$

$$-(G + s_0 C) u_k = C u_{k-1} \quad k = 1, 2, \ldots$$
Taylor Expansion and Moments (Cont’d)

• Expansion of $H(s)$ around $\sigma \to \infty$

$$H(s_0 + \sigma) = l^T (-\sigma^{-1} A^{-1})(I + \sigma^{-1} A^{-1} + \sigma^{-2} A^{-2} + \cdots)r$$

$$= \sum_{k=-1}^{\infty} -m_k \sigma^k \quad \text{where } m_k = l^T A^k r$$

• Recursive moment computation:

$$u_0 = r$$

$$u_{-1} = A^{-1} r$$

$$u_{-2} = A^{-2} r$$

$$\vdots$$

$$\implies (G + s_0 C)u_0 = b \iff u_0 = (G + s_0 C)^{-1} b$$

$$Cu_k = -(G - s_0 C)u_{k+1} \quad k = -1, -2, \ldots$$
**Interpretation of Moment Computation**

- Compute: \( u_0 = (G + s_0 C)^{-1} b \)
- When \( s_0 = 0 \), equivalent to DC analysis:
  - setting \( \dot{x} = 0 \), shorting inductors (0V) and opening capacitors (0A)
  - compute currents through inductors and voltages across capacitors as moments

- Convert: Inductor \( \rightarrow \) Voltage source
  Capacitor \( \rightarrow \) Current source

![Diagram showing the conversion of inductor to voltage source and capacitor to current source]
Interpretation of Moment Computation (Cont’d)

• Compute: \(- (G + s_0 C)u_1 = Cu_0\)

• When \(s_0 = 0\), equivalent to DC analysis:
  - setting \(x = u_0\), voltage sources of inductor \(L = Lm_L\), current sources of capacitor \(C = Cm_C\)
  - external excitations = 0
  - compute currents through inductors and voltages across capacitors as moments

• Convert: Inductor \(\rightarrow\) Voltage source
  Capacitor \(\rightarrow\) Current source

\[ V_{in} \quad L_2 \quad L_1 \quad C_2 \quad C_1 \]
\[ 0V \quad \pm \]

\[ L_2 m_{L_2} \quad L_1 m_{L_1} \quad C_2 m_{C_2} \quad C_1 m_{C_1} \]
Interpretation of Moment Computation (Cont’d)

- Compute: \( Cu_{-1} = -(G + s_0 C)u_0 \)
- When \( s_0 = 0 \), equivalent to DC analysis:
  - setting \( x = u_0 \), moments as currents through inductors and voltages across capacitors
  - external excitations = 0
  - compute voltage sources of inductors and current sources of capacitors
- Convert: Inductor \( \rightarrow \) Voltage source
  Capacitor \( \rightarrow \) Current source
Moment Computation by DC Analysis

- Perform DC analysis to compute the \((i+1)\)-th order moments

  voltage across \(C_j\) => the \((i+1)\)-th order moment of \(C_j\)

  current across \(L_j\) => the \((i+1)\)-th order moment of \(L_j\)

- DC analysis:
  - modified nodal analysis (used in original AWE)
  - sparse-tableau
  - .......

- Time complexity to compute moments up to the \(p\)-th order:
  \[ p \times \text{time complexity of DC analysis} \]
Advantage and Disadvantage of Moment Computation by DC Analysis

- Recursive computation of vectors $u_k$ is efficient since the matrix $(G+s_0C)$ is LU-factored exactly once.

- Computation of $u_k$ corresponds to vector iteration with matrix $A$:
  - Converges to an eigenvector corresponding to an eigenvalue of $A$ with largest absolute value.
Moment Computation for RLC Trees

- Most interconnects are RLC trees
- Exploit special structure of $G$ and $C$ matrices to compute moments
- Two basic approaches:
  - Recursive moment computation [Yu-Kuh, TVLSI’95]
  - Incremental moment computation [Cong-Koh, ICCAD’97]
- Both papers used a slightly different definition of moment
  \[ m_i = \frac{1}{i!} \int_0^\infty h(t) \cdot t^i \, dt \]
- We continue to use the same definition as before
  \[ m_i = \frac{(-1)^i}{i!} \int_0^\infty h(t) \cdot t^i \, dt \]
Basis of Moment Computation for RLC Trees

- Wire segment modeled as lumped RLC element
- Definition:
  \[ T_k = \text{subtree rooted at node } k \]
  \[ \bar{k} = \text{parent node of node } k \]
  \[ R_k = \text{resistance between nodes } \bar{k} \text{ and } k \]
  \[ L_k = \text{inductance between nodes } \bar{k} \text{ and } k \]
  \[ C_k = \text{capacitance at node } k \]
  \[ i_k = \text{current from } \bar{k} \text{ to } k \]
  \[ I_k(s) = \text{Laplace transform of } i_k \]
  \[ v_k = \text{voltage across } C_k \]
  \[ V_k(s) = \text{Laplace transform of } v_k \]
  \[ m^p_k = p\text{-th order moment of node } k \]
Transfer Function for RLC Trees

• Apply KCL at node $k$: $I_k(s) = \sum_{j \in T_k} C_j s V_j(s)$

• Let $P_k = \text{path from root to node } k$

  $P_{jk} = P_j \cap P_k$

  $R_{jk} = \text{total resistance on path } P_{jk}$

  $L_{jk} = \text{total inductance on path } P_{jk}$

  $Z_{ik} = R_{jk} + sL_{jk}$

• Voltage drop from root to node $i$: $V_{in}(s) - V_i(s) = \sum_{k \in P_i} (R_k + sL_k) \sum_{j \in T_k} C_j s V_j(s)$

  $= \sum_k Z_{ik} C_k s V_k(s)$

• Transfer function: $H_i(s) = 1 - \sum_k Z_{ik} C_k s H_k(s)$
Recursive Formula for Moments

• Transfer function: \( H_i(s) = 1 - \sum_{k} Z_{ik} C_k s H_k(s) \)

• Expanding \( H_i(s) \), \( H_k(s) \) into Taylor series:
  \[
  1 + \sum_{j=1}^{\infty} m_i^j s^j = 1 - \sum_{k} s R_{ik} C_k - \sum_{j=2}^{\infty} s^j \sum_{k} (R_{ik} C_k m_k^{j-1} + L_{ik} C_k m_k^{j-2})
  \]

• Define \( p \)-th order weighted capacitance of \( C_k \): \( C_k^p = m_k^p C_k \)

• Recursive formula for moments:
  \[
  m_i^p = \begin{cases} 
  0 & \text{if } p = -1 \\
  1 & \text{if } p = 0 \\
  -\sum_{k} (R_{ik} C_k^{p-1} + L_{ik} C_k^{p-2}) & \text{if } p \geq 1
  \end{cases}
  \]
Recursive Formula for Moments (Cont’d)

- Define $p$-th order weighted capacitance in subtree $T_k$
  \[ C_{T_k}^p = \sum_{j \in T_k} m_j^p C_j \]

- Can show that:
  \[ m_k^p = \begin{cases} 
  0 & \text{if } p = -1 \\
  1 & \text{if } p = 0 \\
  0 & \text{if } p \geq 1 \text{ and } k = \text{root} \\
  m_k^p - (R_k C_{T_k}^{p-1} + L_k C_{T_k}^{p-2}) & \text{if } p \geq 1 \text{ and } k \neq \text{root} 
\end{cases} \]

- Equivalently:
  \[ m_k^p = -\sum_{j \in P_k} (R_j C_{T_j}^{p-1} + L_j C_{T_j}^{p-2}) \]

- Similarity with definition of Elmore delay
Recursive Moment Computation for RLC Trees

[Yu-Kuh, TVLSI’95]

- Initialize (-1)-th and 0-th order moments
- Compute moments from order 1 to order \( p \) successively
- Bottom-up tree traversal phase: \( O(n) \) for \( n \) nodes
  - Compute all \( p \)-th order weighted capacitance \( C_k m_k^p \)
  - Compute the \( p \)-th order weighted capacitance in subtree \( T_k \)

\[
C_{T_k}^p = \sum_{j \in T_k} C_k m_k^p
\]

- Top-down tree traversal phase; \( O(n) \) for \( n \) nodes
  - Compute the moment at each capacitor node \( k \)

\[
m_k^{p+1} = m_k^{p+1} - (R_k C_{T_k}^p + L_k C_{T_k}^{p-1})
\]

- Time complexity to compute moments up to the \( p \)-th order = \( O(np) \)
Incremental Moment Computation
[Cong-Koh, ICCAD’97]

- Iterative tree traversal approaches such as [Kuh-Yu, TVLSI’95] assume a static tree topology
  - More suitable for interconnect analysis
  - Not suitable for tree topology optimization
  - After each modification of tree topology, need to recompute all $p$-th order moments

- Incremental updates of sink moments
  - More suitable for tree optimization algorithm that construct topology in a bottom-up fashion
  - As we modify the tree topology, update the transfer function $H_i(s)$ for sink $s_i$ incrementally
Basis for Bottom-Up Moment Computation

- Consider the computation of moments for sink $w$
- Assume sink $w$ is originally in subtree $T_v$
- Merge subtree $T_v$ with another subtree and the new tree is rooted at node $u$
- Definition:
  
  $m_k^p = p$ - th order moment of node $k$ in original subtree $T_v$
  $C_{T_k}^p = p$ - th order weighted capacitance of subtree $T_k$ in original subtree $T_v$
  
  $H_{v-k}(s) = \text{transfer function of node } k \text{ in original subtree } T_v$
  $ar{m}_k^p = p$ - th order moment of node $k$ in new subtree $T_u$
  $\bar{C}_{T_k}^p = p$ - th order weighted capacitance of subtree $T_k$
  in new subtree $T_u$
  $\bar{H}_{u-k}(s) = \text{transfer function of node } k \text{ in new subtree } T_u$
Bottom-Up Moment Computation

- Maintain transfer function $H_{v-w}(s)$ for sink $w$ in subtree $T_v$, and moment-weighted capacitance of subtree:
  
  \[ m^j_w \text{ for } j = 0 \ldots p, \quad C^j_{T_v} \text{ for } j = 0 \ldots p - 1 \]

- As we merge subtrees, compute new transfer function $\overline{H}_{u-v}(s)$ and weighted capacitance recursively:
  
  \[
  \overline{C}^{j-1}_{T_v} = \overline{m}^{j-1}_v C^j_v + \sum_{q=0}^{j-1} \overline{m}^{j-1-q}_v C^q_{T_v},
  \]

  \[
  \overline{m}^j_v = -(R_v \overline{C}^{j-1}_{T_v} + L_v \overline{C}^{j-2}_{T_v}) \text{ for } j = 1 \ldots p
  \]

- New transfer function for node $w$

  \[
  \overline{H}_{u-w}(s) = \overline{H}_{u-v}(s) \times H_{v-w}(s)
  \]

  \[
  \overline{m}^j_w = \sum_{q=0}^{j} \overline{m}^{j-q}_v m^q_w \text{ for } j = 0 \ldots p
  \]

- New moment-weighted capacitance of $T_u$:

  \[
  \overline{C}^j_{T_u} = \sum_{v \in \text{child}(u)} \overline{C}^j_{T_v} \text{ for } j = 0 \ldots p - 1
  \]
Complexity Analysis of Incremental Moment Computation

- Assume topology has $n$ nodes
- Complexity due to each edge: $O(p^2)$
- Total time complexity: $O(np^2)$
- If there are $k$ nodes of interest, time complexity = $O(knp^2)$
Moment Matching by AWE
[Pillage-Rohrer, TCAD’90]

• Recall the transfer function obtained from a linear circuit
  - Let $s = s_0 + \sigma$, where $s_0$ is an arbitrary, but fixed expansion point such that $G + s_0 C$ is non-singular
    \[
    H(s_0 + \sigma) = \mathbf{l}^T (I - \sigma \mathbf{A})^{-1} \mathbf{r}
    \]
    where $\mathbf{A} = -(G + s_0 C)^{-1} C$, \quad $\mathbf{r} = (G + s_0 C)^{-1} \mathbf{b}$

• When matrix $\mathbf{A}$ is diagonalizable
  \[
  \mathbf{A} = \mathbf{S} \Lambda \mathbf{S}^{-1} \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)
  \]
  \[
  H(s_0 + \sigma) = \mathbf{l}^T \mathbf{S} (I - \sigma \Lambda)^{-1} \mathbf{S}^{-1} \mathbf{r}
  \]
  \[
  = \mathbf{f}^T \mathbf{g}
  \]

\[
\therefore H(s_0 + \sigma) = \sum_{j=1}^{N} \frac{f_j g_j}{1 - \sigma \lambda_j} \quad \lambda_j: \text{reciprocal of pole}
\]
**q-th Pade Approximation**

- Pade approximation of type \((p/q)\):
  \[
  H_{p,q}(s_0 + \sigma) = \frac{b_p \sigma^p + \cdots + b_1 \sigma + b_0}{a_q \sigma^q + \cdots + a_1 \sigma + 1}
  = H(s_0 + \sigma) + O(\sigma^{p+q+1})
  \]

- \(q\)-th Pade approximation \((q \ll N)\):
  \[
  H_q(s_0 + \sigma) \equiv H_{q-1,q} = \sum_{j=1}^{q} \frac{k_j}{\sigma - p_j}
  \]

- Equivalent to finding a reduced-order matrix \(A_R\) such that eigenvalues \(\lambda_j\) of \(A_R\) are reciprocals of the approximating poles \(p_j\) for the original system
Asymptotic Waveform Evaluation

• Recall EQ1: 
  $-(k_1 + k_2 + \cdots + k_q) = -h(0) = m_{-1}$

  $-(\frac{k_1}{p_1} + \frac{k_2}{p_2} + \cdots + \frac{k_q}{p_q}) = m_0$

  $-(\frac{k_1}{p_1^2} + \frac{k_2}{p_2^2} + \cdots + \frac{k_q}{p_q^2}) = m_1$

  $\vdots$

  $-(\frac{k_1}{p_1^{2q-1}} + \frac{k_2}{p_2^{2q-1}} + \cdots + \frac{k_q}{p_q^{2q-1}}) = m_{2q-2}$

• Let $\lambda_j = 1/p_j$

  $V_j = [1 \quad \lambda_j \quad \lambda_j^2 \quad \cdots \quad \lambda_j^{q-1}]^T$

  $V = [V_1 \quad V_2 \quad \cdots \quad V_q]$

  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_q)$
Asymptotic Waveform Evaluation (Cont’d)

• Rewrite EQ1: \(-Vk = m_l\)
  \(-V\Lambda^q k = m_h\)

where \(k = [k_1 \ k_2 \ \ldots \ k_q]^T\)

\[m_l = [m_{-1} \ m_0 \ \ldots \ m_{q-2}]^T\]

\[m_h = [m_{q-1} \ k_q \ \ldots \ k_{2q-2}]^T\]

• Solving for \(k\): \(k = -V^{-1} m_l\)

\(V\Lambda^q V^{-1} m_l = m_h\)

• Let \(A_R = V\Lambda V^{-1}\)

\(\therefore A_R^q m_l = m_h\)

→ Need to compute all the poles first
Structure of Matrix $A_R$

- Matrix: 
  $$\begin{bmatrix}
    0 & 1 & 0 & \cdots & 0 \\
    0 & 0 & 1 & \cdots & 0 \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & 0 & \cdots & 1 \\
    -a_q & -a_{q-1} & -a_{q-2} & \cdots & -a_1
  \end{bmatrix}$$

  has characteristic equation: 
  $$\lambda^q + a_1\lambda^{q-1} + \cdots + a_{q-1}\lambda + a_q = 0$$

  Eigenvalue $\lambda_j$ has eigenvector $[1 \ \lambda_j \ \lambda_j^2 \ \cdots \ \lambda_j^{q-1}]^T$

- Therefore, $A_R$ could be a matrix of the above structure

- Note that: $\lambda = 1/\sigma$

  Characteristic equation becomes the denominator of $H_q(s)$:
  $$1 + a_1\sigma + \cdots + a_{q-1}\sigma^{q-1} + a_q\sigma^q = 0$$
Solving for Matrix $A_R$

- Consider multiplications of $A_R$ on $m_l$

$$
\begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
-a_q & -a_{q-1} & -a_{q-2} & \cdots & -a_1
\end{bmatrix}
\begin{bmatrix}
m_{-1} \\
m_0 \\
\vdots \\
m_{q-3} \\
m_{q-2}
\end{bmatrix}
=
\begin{bmatrix}
m_0 \\
m_1 \\
\vdots \\
m_{q-3} \\
m_{q-2}
\end{bmatrix}
$$

produces

$$m'_{q-1} = -a_q m_{-1} - a_{q-1} m_0 - a_{q-2} m_1 - \cdots - a_1 m_{q-2}$$

$$A_R^2 m_l = 
\begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
-a_q & -a_{q-1} & -a_{q-2} & \cdots & -a_1
\end{bmatrix}
\begin{bmatrix}
m_0 \\
m_1 \\
\vdots \\
m_{q-2} \\
m_{q-1}
\end{bmatrix}
=
\begin{bmatrix}
m_0 \\
m_1 \\
\vdots \\
m_{q-2} \\
m_{q-1}
\end{bmatrix}
$$

produces

$$m'_q = -a_q m_0 - a_{q-1} m_1 - a_{q-2} m_2 - \cdots - a_1 m'_{q-1}$$
Solving for Matrix $A_R$ (Cont’d)

- After $q$ multiplications of $A_R$ on $m_l$

$$A_R^q m_l = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_q & -a_{q-1} & -a_{q-2} & \cdots & -a_1 \end{bmatrix} \begin{bmatrix} m_{q-2} \\ m'_{q-1} \\ m_{q-1} \\ m_{2q-4} \\ m'_{2q-3} \end{bmatrix} = \begin{bmatrix} m'_{q-1} \\ m'_q \\ m'_{2q-3} \\ m'_{2q-2} \end{bmatrix} = m_h$$

produces $m'_{q-2} = -a_q m_{q-2} - a_{q-1} m'_{q-1} - a_{q-2} m'_q - \cdots - a_1 m'_{2q-3}$

- Equating $m'$ with $m$:

$$\begin{bmatrix} m_{-1} & m_0 & m_1 & \cdots & m_{q-2} \\ m_0 & m_1 & m_2 & \cdots & m_{q-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{q-3} & m_{q-2} & m_{q-1} & \cdots & m_{2q-4} \\ m_{q-2} & m_{q-1} & m_q & \cdots & m_{2q-3} \end{bmatrix} \begin{bmatrix} -a_q \\ -a_{q-1} \\ \vdots \\ -a_2 \\ -a_1 \end{bmatrix} = \begin{bmatrix} m'_{q-1} \\ m'_q \\ \vdots \\ m'_{2q-3} \\ m'_{2q-2} \end{bmatrix}$$
Summary of AWE

Step 1: Compute \(2q\) moments, choice of \(q\) depends on accuracy requirement; in practice, \(q \leq 5\) is frequently used

Step 2: Solve a system of linear equations by Gaussian elimination to get \(a_j\)

\[
\begin{bmatrix}
m_{-1} & m_0 & m_1 & \cdots & m_{q-2} \\
m_0 & m_1 & m_2 & \cdots & m_{q-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
m_{q-3} & m_{q-2} & m_{q-1} & \cdots & m_{2q-4} \\
m_{q-2} & m_{q-1} & m_q & \cdots & m_{2q-3}
\end{bmatrix}
\begin{bmatrix}
-a_q \\
-a_{q-1} \\
\vdots \\
-a_2 \\
-a_1
\end{bmatrix}
= \begin{bmatrix}
m_{q-1} \\
m_q \\
\vdots \\
m_{2q-3} \\
m_{2q-2}
\end{bmatrix}
\]

Step 3: Solve the characteristic equation of \(A_R\) to determine the approximate poles \(p_j\)

\[
\lambda^q + a_1 \lambda^{q-1} + \cdots + a_{q-1} \lambda + a_q = 0
\]

Step 4: Solve for residues \(k_j\)

\[
k = -V^{-1}m_l
\]
Numerical Limitations of AWE

- Due to recursive computation of moments
  - Converges to an eigenvector corresponding to an eigenvalue of matrix $A$ with largest absolute value
  - Moment matrix used in AWE becomes rapidly ill-conditioned
  - Increasing number of poles does not improve accuracy
  - Unable to estimate the accuracy of the approximating model
- Remedial techniques are sometimes heuristic, hard to apply automatically, and may be computationally expensive
Pade Approximation via Lanczos Method (PVL)  
[Feldmann-Freund, TCAD’95]

• Basic idea of non-symmetric Lanczos method:
  – Consider the right and left Krylov subspaces
    Right Krylov subspace = \( K_q(v_1, A) = \text{span}\{v_1, Av_1, \ldots, A^{q-1}v_1\} \)
    Left Krylov subspace = \( K_q(w_1, A^T) = \text{span}\{w_1, A^Tw_1, \ldots, (A^T)^{q-1}w_1\} \)
  – Apply Lanczos recursion to generate for \( j = 2 \ldots q \):
    • Two sequences of bi-orthogonal basis vectors for left and right
      Krylov subspaces \( K_j(v_1, A) \) and \( K_j(w_1, A^T) \), respectively
    • A sequence of non-symmetric tridiagonal Lanczos matrices \( T_j \)
      which are matrix representation of bi-orthogonal projection of \( A \)
      onto \( K_j(v_1, A) \) and \( K_j(w_1, A^T) \)
      • Eigenvalues of \( T_j \) approximate those of the given matrix \( A \)

• Advantages:
  – Avoid direct computation of moments
  – Numerically robust for higher-order Pade approximation
Non-symmetric Lanczos Method

Step 0: Set $\rho_1 = \| r \|_2$, $\eta_1 = \| l \|_2$, $v_1 = r / \rho_1$, and $w_1 = 1 / \eta_1$

Set $v_0 = w_0 = 0$ and $\delta_0 = 0$

For $n = 1, 2, \ldots, q$ do:

Step 1: Compute $\delta_n = w_n^T v_n$

Step 2: Set $\alpha_n = \frac{w_n^T A v_n}{\delta_n}$, $\beta_n = \frac{\delta_n}{\delta_{n-1}} \eta_n$, $\gamma_n = \frac{\delta_n}{\delta_{n-1}} \rho_n$

Step 3: Set $\hat{v}_{n+1} = A v_n - v_n \alpha_n - v_{n-1} \beta_n$

$\hat{w}_{n+1} = A^T w_n - w_n \alpha_n - w_{n-1} \gamma_n$

Step 4: Set $\rho_{n+1} = \| \hat{v}_{n+1} \|_2$, $\eta_{n+1} = \| \hat{w}_{n+1} \|_2$, and

$v_{n+1} = \frac{\hat{v}_{n+1}}{\rho_{n+1}}$, $w_{n+1} = \frac{\hat{w}_{n+1}}{\eta_{n+1}}$
Remarks on Lanczos Method

• Breakdown will occur if one encounters $\delta_n = 0$
• Division by nonzero yet small number $\delta_n$ may result in numerical instability
• Problems can be remedied by using a so-called look-ahead variant of the Lanczos recursion
• If matrix $A$ is symmetric and the starting vectors $v_1$ and $w_1$ are identical, it is known as symmetric Lanczos method
• However, symmetric Lanczos method cannot match the maximal number of moments
  – Only half of the moments can be matched
Properties of Lanczos Vectors

- **Lanczos vectors:** \( \{v_n\}_{n=1}^{q+1} \) and \( \{w_n\}_{n=1}^{q+1} \)
- **Bi-orthogonality:**
  \[
  w_j^T v_k = v_j^T w_k = \begin{cases} 
  \delta_j & \text{if } j = k \\
  0 & \text{if } j \neq k 
  \end{cases} \quad j, k = 1, 2, \ldots, q + 1
  \]

Proof by induction:

\[
 v_{k+1} \rho_{k+1} = Av_k - v_k \alpha_k - v_{k-1} \beta_k
\]

\[
 \Rightarrow w_j^T v_{k+1} \rho_{k+1} = w_j^T Av_k - w_j^T v_k \alpha_k - w_j^T v_{k-1} \beta_k
\]

If \( j = k \):
\[
 w_k^T v_{k+1} \rho_{k+1} = w_k^T Av_k - w_k^T v_k \alpha_k - w_k^T v_{k-1} \beta_k = 0
\]

If \( j < k \):
\[
 w_{j+1} \eta_{j+1} = A^T w_j - w_j \alpha_j - w_{j-1} \gamma_j
\]

\[
 \Rightarrow w_j^T A = w_{j+1}^T \eta_{j+1} + w_j^T \alpha_j + w_{j-1}^T \gamma_j
\]

\[
 \Rightarrow w_j^T v_{k+1} \rho_{k+1} = w_{j+1}^T v_k \eta_{k+1} + w_j^T v_k \alpha_j + w_{j-1}^T v_k \gamma_j - w_j^T v_k \alpha_k - w_j^T v_{k-1} \beta_k = 0
\]
Properties of Lanczos Vectors (Cont’d)

• Let \( V_k = [v_1 \ v_2 \ \cdots \ v_k] \) \( k = 1, 2, \ldots, q \)
  \[ W_k = [w_1 \ w_2 \ \cdots \ w_k] \]

Then \( D_k = W_k^T V_k = \text{diag} (\delta_1, \delta_2, \ldots, \delta_k) \)

• Bases of Krylov subspaces:
  \[ \{v_n\}_{n=1}^k \text{ span } K_k (v_1, A) \]
  \[ \{w_n\}_{n=1}^k \text{ span } K_k (w_1, A^T) \]
Properties of Lanczos Matrices

- Lanczos matrices: for \( k = 1, 2, \ldots, q \)

\[
T_k = \begin{bmatrix}
\alpha_1 & \beta_2 & 0 & \cdots & 0 \\
\rho_2 & \alpha_2 & \beta_3 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \rho_3 & \cdots & \cdots & \beta_k \\
0 & \cdots & \cdots & \cdots & \beta_k \\
0 & \cdots & 0 & \rho_k & \alpha_k
\end{bmatrix}
\]

and

\[
\tilde{T}_k = \begin{bmatrix}
\alpha_1 & \gamma_2 & 0 & \cdots & 0 \\
\eta_2 & \alpha_2 & \gamma_3 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \eta_3 & \cdots & \cdots & \gamma_k \\
0 & \cdots & \cdots & \cdots & \gamma_k \\
0 & \cdots & 0 & \eta_k & \alpha_k
\end{bmatrix}
\]

- Relation between Lanczos matrices:

\[
\tilde{T}_k^T = D_k T_k D_k^{-1}
\]

- Relation to matrix \( A \):

\[
A V_k = V_k T_k + \rho_{k+1} v_{k+1} e_k^T \\
A^T W_k = W_k \tilde{T}_k + \eta_{k+1} w_{k+1} e_k^T
\]

\( e_k^T \) is the coordinate vector whose \( j \)-th component = 1 and whose other components are 0.
Connection of Lanczos to Pade

- $T_q$ is the matrix representation of bi-orthogonal of $A$ onto $T_q(v_1, A)$ and $T_q(w_1, A^T)$

- It is the best approximation to $A$ in the sense of matching the maximal number of moments
  - Recall that $m_k = l^T A^k r = (l^T A^{k'}) (A^{k''} r)$ where $k = k' + k''$
  - From relations of Lanczos matrices and $A$

$$
(l^T A^j)^T = \eta_1 (A^T)^j w_1
= \eta_1 (A^T)^j W_q e_1

= \eta_1 W_q \tilde{T}_q^j e_1

\Rightarrow l^T A^j = \eta_1 e_1^T (\tilde{T}_q^T)^j W_q^T

A^j r = \rho_1 A^j v_1
= \rho_1 V_q T_q^j e_1
\begin{cases}
  m_k = (\eta_1 e_1^T (\tilde{T}_q^T)^k W_q^T) (\rho_1 V_q T_q^{k''} e_1) \\
  = (\eta_1 \delta_1 e_1^T T_q^k D_q^{-1} W_q^T) (\rho_1 V_q T_q^{k''} e_1) \\
  = \eta_1 \rho_1 \delta_1 e_1^T T_q^k e_1 \\
  = (l^T r) (e_1^T T_q^k e_1) \quad \forall k = 0, \cdots, 2q - 1
\end{cases}
$$
Connection of Lanczos to Pade (Cont’d)

• $q$-th order Pade approximation

\[ m_k = (l^T r)(e_1^T T_q^k e_1) \quad \forall k = 0, \cdots, 2q - 1 \]

\[
(l^T r) \cdot e_1^T (I - \sigma T_q)^{-1} e_1 = (l^T r) \cdot \sum_{k=0}^{\infty} e_1^T T_q^k e_1 \sigma^k
\]

\[ = \sum_{k=0}^{2q-1} m_k \sigma^k + O(\sigma^{2q}) \]

\[ \therefore H_q (s_0 + \sigma) = (l^T r) \cdot e_1^T (I - \sigma T_q)^{-1} e_1 \]
Eigendecomposition of $T_q$

$$T_q = S_q \Lambda_q S_q^{-1} \quad \Lambda_q = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_q)$$

$$H(s_0 + \sigma) = l^T r \cdot e_1^T S_q (I - \sigma \Lambda_q)^{-1} S_q^{-1} e_1$$

$$= \mu^T \quad = \nu$$

$$\therefore H(s_0 + \sigma) = \sum_{j=1}^{q} \frac{l^T r \cdot \mu_j \nu_j}{1 - \sigma \lambda_j} \quad \lambda_j: \text{reciprocal of pole}$$

$$H(s_0 + \sigma) = k_\infty + \sum_{\lambda_j \neq 0} \frac{-l^T r \cdot \mu_j \nu_j / \lambda_j}{\sigma - 1 / \lambda_j}$$

$$k_\infty = \sum_{\lambda_j = 0}^{q} l^T r \cdot \mu_j \nu_j \text{ may result if one of the eigenvalues is zero}$$
PVL Algorithm

Step 1: Run $q$ steps of the Lanczos process to obtain the tridiagonal matrix $T_q$

Step 2: Compute an eigendecomposition

$$T_q = S_q \Lambda_q S_q^{-1} \quad \Lambda_q = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_q)$$

Set $\mu = S_q^T e_1$ and $\nu = S_q^{-1} e_1$

Step 3: Compute the poles and residues of $H_q$

Set $p_j = 1/\lambda_j$ and $k_j = \frac{1^T r \cdot \mu_j \nu_j}{\lambda_j}$

for all $j = 1, 2, \ldots, q$ with $\lambda_j \neq 0$

$$\kappa_\infty = \sum_{j=1}^{q} \frac{1^T r \cdot \mu_j \nu_j}{\lambda_j} \quad \text{if there is at least one } \lambda_j = 0$$
Remarks on PVL Algorithm

- Require roughly the same amount of computational work as AWE
  - Dominating cost is the LU factorization of \((G+s_0C)\) which needs to be computed only once
  - Needs total of \(2q\) computations of \(A v_n\) and \(A^T w_n\), same amount computation as AWE

- Numerically more robust than AWE

- PVL algorithm can be used to obtain bounds for the pole-approximation error

- Expansion point \(s_0\) is chosen such that \(s_0 = 2\pi f_{max}\) where the frequency range of interest is in \([0, f_{max}]\)
  - PVL is fairly insensitive to the choice of \(s_0\)
Arnoldi Algorithm for Krylov Subspace

[Silveira et al., ICCAD’96]

• Considers only the (right) Krylov subspace:

\[ K_q(v_1, A) = \text{span}\{v_1, Av_1, \ldots, A^{q-1}v_1\} \]

• Apply the Arnoldi process (applied to \( A \) and \( r \)) to produce for \( j = 2 \ldots q \):
  – One sequence of \( j \) orthonormal Arnoldi vectors that span the Krylov subspace \( K_j(v_1,A) \)
  – A sequence of \( j \)-by-\( j \) upper Hessenberg (tridiagonal + upper triangular) matrices \( T_j^{(A)} \) which are matrix representations of orthogonal projections of \( A \) onto the Krylov subspaces

• Arnoldi process achieves orthonormality by using recurrences that involve all previous vectors
  – numerically optimal but time consuming

• The approximate transfer function matches only \( q \) moments
Arnoldi Process

Step 0: Set $\hat{v}_1 = r$

For $n = 1, 2, \ldots, q$ do:

Step 1: Compute $t_{n,n-1} = \|\hat{v}_n\|_2$

If $t_{n,n-1} = 0$, then stop

(Krylov subspace $K_n(A, r)$ is exhausted)

Step 2: Set $v_n = \frac{\hat{v}_n}{t_{n,n-1}}$

Step 3: Set $\hat{v}_{n+1} = Av_n$

Step 4: For $i = 1, 2, \ldots, n$ do:

Set $t_{i,n} = v_i^H \hat{v}_{n+1}$ and $\hat{v}_{n+1} = \hat{v}_{n+1} - v_i t_{i,n}$
**Hessenberg Matrix**

- **Upper Hessenberg Matrix**

\[
T^{(A)}_n = \begin{bmatrix}
t_{11} & t_{12} & \cdots & \cdots & t_{1n} \\
t_{21} & t_{22} & t_{23} & \ddots & \vdots \\
0 & t_{32} & t_{33} & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & t_{n-1,n} \\
0 & \cdots & 0 & t_{n,n-1} & t_{n,n}
\end{bmatrix}
\]

- **Relation to matrix \( A \):**

For \( k = 1, 2, \cdots, q \)

\[
V_k = \begin{bmatrix}v_1 & v_2 & \cdots & v_k\end{bmatrix}
\]

\[
AV_k = V_k T^{(A)}_k + t_{k+1,k} v_{k+1} e_k^T
\]
Arnoldi-Based Reduced-Order Modeling

• The Arnoldi approach matches only $q$ moments

$$A^k r = \|r\|_2 A^k v_1$$

$$= \|r\|_2 A^k V_q e_1$$

$$= \|r\|_2 V_q (T_q^{(A)})^k e_1$$

$$m_k = l^T A^k r = \|r\|_2 \cdot l^T V_q (T_q^{(A)})^k e_1 \quad \text{for } k < q$$

$$\therefore H_q^{(A)}(s_0 + \sigma) = \|r\|_2 \cdot l^T V_q (I - \sigma T_q^{(A)})^{-1} e_1 = H(s_0 + \sigma) + O(\sigma^q)$$

• Arnoldi-based reduced-order modeling is not Padé-approximation, it is only a Padé-type approximation

• Possible to perform two-sided Arnoldi process to obtain Padé-approximation, but too expensive
Matrix-Pade via Lanczos (MPVL)  
[Feldmann-Freund, DAC’95]

- Interface between nonlinear and linear subnetworks is more complicated than one-port
  - AWE, PVL, and Arnoldi-based approach assume scalar-valued transfer function $H(s)$ for a single port

- An $m$-input $p$-output linear network can be analyzed by a $p \times m$ matrix of Laplace-domain transfer functions
  - Interface can be modeled by obtaining a Pade approximation for each pair of inputs and outputs using AWE, PVL, or Arnoldi-based approach
  - Computational cost increases rapidly with the size of interface
  - Individual approximate transfer functions are computed separately without sharing information

- MPVL used a block Lanczos algorithm to compute a matrix Pade approximation to the entire matrix-valued transfer function simultaneously
Matrix Transfer Function

• Represent a $m$-input $p$-output lumped, linear, time-invariant circuit by a system of first-order differential equations:

$$C \dot{x} = -Gx + Bu$$

$$y = L^T x$$

where $B$ is $N \times m$, $L$ is $N \times p$, $u$ is a vector of input functions, and $y$ is the vector of unknown output functions

• Transfer function matrix:

$$H(s) = Y(s)(U(s))^{-1} = L^T (G + sC)^{-1} B$$

• Let $s = s_0 + \sigma$, where $s_0$ is an arbitrary, but fixed expansion point such that $G + s_0 C$ is non-singular

$$H(s_0 + \sigma) = L^T (I - \sigma A)^{-1} R$$

where $A = -(G + s_0 C)^{-1} C$, $r = (G + s_0 C)^{-1} B$
Matrix Pade Approximation

• Expansion of matrix transfer function $H(s_0 + \sigma)$:

$$H(s_0 + \sigma) = \sum_{i=0}^{\infty} M_i \cdot \sigma^i$$

where $M_i$ is $p \times m$, and it is called the moments or Markov parameters of $H$

• $q$-th Pade approximation ($q << N$):

$$H_q(s_0 + \sigma) = H(s_0 + \sigma) + O(\sigma^{2q})$$

$$= \sum_{i=0}^{2q-1} M_i \sigma^i + O(\sigma^{2q})$$

• Need a block Lanczos method to implicitly compute

$$M_i = L^T A^i R$$
Block Krylov Subspace

- Consider not just a single starting vector but multiple starting vectors: right and left starting vectors in matrices \( R \) and \( L \)

- Definition of right and left block Krylov Matrices

\[
K(R, A) = [R \ A R \ A^2 R \ \cdots \ A^{N-1} R]
\]

\[
K(L, A^T) = [L \ A^T L \ (A^T)^2 L \ \cdots \ (A^T)^{N-1} L]
\]

- \( K_n(R, A) \) \((K_n(L, A^T))\): \( n \)-th dimensional right (left) block Krylov subspace is the subspace spanned by the first \( n \) linearly independent columns of the matrix \( K(R, A) \) \((K(L, A^T))\)

- Block Lanczos construct two sequences of \( j \) bi-orthogonal basis vectors \( V_j = \{v_1, v_2, \ldots, v_j\} \) and \( W_j = \{w_1, w_2, \ldots, w_j\} \) for left and right block Krylov subspaces \( K_j(R, A) \) and \( K_j(L, A^T) \)
Sketch of Block Lanczos Method

Step 0: \( R = [r_1 \ r_2 \ \cdots \ r_m] \) and \( L = [l_1 \ l_2 \ \cdots \ l_p] \)

Set auxiliary vectors \( \hat{v}_i = r_i \) for \( i = 1 \ldots m \)

\( \hat{w}_i = l_i \) for \( i = 1 \ldots p \)

For \( n = 1, 2, \ldots, q \) do:

Step 1: Compute \( v_n = \frac{\hat{v}_n}{\|\hat{v}_n\|_2} \) and \( w_n = \frac{\hat{w}_n}{\|\hat{w}_n\|_2} \)

Step 2: Compute auxiliary vectors \( \hat{v}_{n+m} = Av_n \)

\( \hat{w}_{n+p} = Aw_n \)

Step 3: Biorthogonalize

\( \hat{v}_{n+m} \) against previous \( p \) Lanczos vectors \( w_i \)

\( \hat{w}_{n+p} \) against previous \( m \) Lanczos vectors \( v_i \)

Step 4: Biorthogonalize

next \( m \) Lanczos vectors \( \hat{v}_{n+i} \) against \( w_n \)

next \( p \) Lanczos vectors \( \hat{w}_{n+i} \) against \( v_n \)
Lanczos Matrix and Reduced-Order Model

- Block Lanczos method also produces two $q \times q$ Lanczos matrices $T_q$ and $\tilde{T}_q$
  - $T_q$ and $\tilde{T}_q$ are banded matrices, $T_q$ has $m$ sub-diagonals and $p$ super-diagonals, $\tilde{T}_q$ has $p$ sub-diagonals and $m$ super-diagonals
  - Almost identical relations with $A$ and Lanczos vectors as in the case of single-vector Lanczos method

- $T_q$ is used in a reduced-order model

  $$\begin{align*}
  C\dot{x} &= -Gx + Bu, \quad y = L^T x \\
  \Rightarrow \quad -A\dot{x} + (I + s_0A)x &= Ru, \quad y = L^T x
  \end{align*}$$

  Use a $q$-size vector $d$ to approximate, i.e., restrict $x$ in the right Krylov subspace $V_q$

  $$\begin{align*}
  x &= V_q d \Rightarrow -AV_q \dot{d} + (I + s_0A)V_q d = Ru, \quad y = L^T V_q d \\
  \Rightarrow \quad -T_q \dot{d} + (I + s_0T_q)d &= W^T Ru, \quad y = L^T V_q d
  \end{align*}$$

  - Approximate the original system with a smaller system, combined with the rest of the system for simulation
Stable and Passive Reduced-Order Models

- Stable and passive networks can never generate more energy than it absorbs.
- Desirable to have guaranteed stable reduced-order models for general RLC circuits to ensure success of simulation.
- Mathematically, a real symmetric matrix $A$ is said to be (strictly) stable if all its eigenvalues have (negative) non-positive real parts.
- A real symmetric matrix $A$ is said to be negative semi-definite (definite) if for any non-zero vector $x$
  \[ x^T A x \leq 0 \]
- Can show that if the real symmetric matrix $A$ is negative semi-definite, then $A$ is stable.
  - Consider eigenvalue $\lambda$ and corresponding eigenvector $x$
    \[ x^T A x = x^T \lambda x = \lambda x^T x \leq 0 \implies \lambda \leq 0 \]
Stable and Passive Reduced-Order Models

- Reduced-order models based on Arnoldi and symmetric Lanczos methods are stable if the real matrix $A$ is negative semi-definite

\[ x^T T_q x = x^T V_q^T A V_q x \]

\[ = (V_q x)^T A (V_q x) \leq 0 \]

- Matrix $A = - (G + s_0 C)^{-1} C$ is not always negative semi-definite even if $G$ and $C$ are in general positive semi-definite

- Reduced-order models based on non-symmetric Lanczos methods can still be unstable even if the Pade approximates are generated from stable circuits

Make use of congruence transformation on semi-definite matrices

\[ B = Q^T A Q \]

$B$ is a congruent to $A$
Coordinate-Transformed Arnoldi Method
[Silveira et al., ICCAD’96]

• Basic idea: if A is negative semi-definite, matrix B is symmetric, then BAB is negative semi-definite

\[ x^T BABx = x^T B^T ABx \]

\[ = (Bx)^T A(Bx) \leq 0 \]

• Matrix \( A = -(G + s_0 C)^{-1}C \) is not always negative semi-definite

• Consider a change of coordinates: \( \tilde{x} = C^{\frac{1}{2}} x \)

\[ C\dot{x} = -Gx + Bu = -(G + s_0 C)x + s_0 Cx + Bu \]

\[ \Rightarrow (G + s_0 C)^{-1} C\dot{x} = -Ix + s_0 (G + s_0 C)^{-1} Cx + (G + s_0 C)^{-1} Bu \]

\[ \Rightarrow C^{\frac{1}{2}} (G + s_0 C)^{-1} C^{\frac{1}{2}} \tilde{x} = -I\tilde{x} + s_0 C^{\frac{1}{2}} (G + s_0 C)^{-1} C^{\frac{1}{2}} \tilde{x} + \]

\[ C^{\frac{1}{2}} (G + s_0 C)^{-1} Bu \]

• Perform Arnoldi recursion based on transformed system matrix:

\[ \tilde{A} = C^{\frac{1}{2}} (G + s_0 C)^{-1} C^{\frac{1}{2}} \]
Pole Analysis by Congruence Transformation (PACT) [Kerns-Yang, TCAD’97]

- Pade approximation assumes that matching lower-order moments will produce dominant poles
- Cannot guarantee the “right” spectrum of poles are approximated
- PACT makes use of the basic idea of congruence transformation of semi-definite matrices
  - Transformation based on Cholesky factorization
  - Transformation based on Eigendecomposition
- Direct pole analysis allows preservation of poles of the RC network between dc and a user-specified maximum frequency
- RC Network reduction is equivalent to dropping of high-frequency poles
- Size of reduced RC network depends only on the number of low-frequency poles found
Circuit Description

• Laplace transform of a lumped, linear, time-invariant circuit:

\[(G + sC)x = Bu = b\]

• Partition and order the entries of matrices such that

\[
\begin{pmatrix}
G_P & G_C^T \\
G_C & G_I
\end{pmatrix} + s
\begin{pmatrix}
C_P & C_C^T \\
C_C & C_I
\end{pmatrix}
\begin{bmatrix}
x_P \\
x_I
\end{bmatrix} =
\begin{bmatrix}
b_P \\
0
\end{bmatrix}
\]

\(G_P, C_P\) : Port matrices describe connection among ports

\(G_I, C_I\) : Internal matrices describe connection among internal nodes

\(G_C, C_C\) : Connection matrices describe branches between ports and internal nodes
Matrix Transfer Function

- Transfer functions at the port (driving point impedances + admittances):
  \[ T(s)x_P = b_P \]
  \[ T(s) = (G_P + sC_P) - (G_C + sC_C)^T (G_I + sC_I)^{-1} (G_C + sC_C) \]

- Properties of sub-matrices \( G_P, C_P, G_I, C_I \):
  - If the values of the resistors and capacitors are positive, then each diagonal entry of \( G \) and \( C \) is positive and greater than or equal to the sum of the absolute value of the off-diagonal elements in the corresponding row.
  - Conditions are sufficient, but not necessary, to ensure that the matrices are non-negative definite (positive semi-definite).
  - Cannot consider inductance since matrix \( C \) would not satisfy these condition (although \( C \) may still be non-negative definite, and the method can still be applied).
  - If each internal node has a dc path to a port node, \( G_I \) is positive definite.
Poles and Eigenvalues

- Poles of $\mathbf{T}(s)$ occur where $(\mathbf{G}_I + s\mathbf{C}_I)$ is singular
- Equal to $-\lambda^{-1}$ where $\lambda$ is the solution to the generalized eigenvalue problem:
  \[ |\mathbf{C}_I - \lambda \mathbf{G}_I| = 0 \]
- Since $\mathbf{G}_I$ is symmetric positive definite, $\mathbf{C}_I$ is symmetric positive semi-definite, the eigenvalues are real and positive
- Therefore, the poles are real and less than zero
- To eliminate high-frequency poles implies elimination of small eigenvalues
- Two transformations are introduced to isolate and eliminate unwanted poles to reduce the size of the network
Transformation by Cholesky Factorization

• Convert $G_I$ into identity matrix by Cholesky factorization (which exists since $G_I$ is positive definite)

$$LL^T = G_I$$

• Let

$$X = \begin{bmatrix} I & 0 \\ -G_I^{-1}G_C & (L^T)^{-1} \end{bmatrix}$$

$$G' = X^T GX = \begin{bmatrix} G'_P & 0 \\ 0 & I \end{bmatrix}$$

$$C' = X^T CX = \begin{bmatrix} C'_P & C''_C \\ C'_C & C'_I \end{bmatrix}$$

$$T(s) = G'_P + sC'_P - s^2C''_C (I + sC'_I)^{-1}C'_C$$

• Needs to perform eigenanalysis on internal capacitance matrix $C'_I$ to drop unwanted poles

• Will not affect dc behavior since $C'_I$ makes no contribution to the first two moments
Transformation by Eigendecomposition

\[ C'_I = U \Lambda U^T \]

\( \Lambda \) : diagonal matrix of eigenvalues

\( U \) : orthogonal square matrix of eigenvectors

Let \( X = \begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix} \), \( G'' = X^T G' X = \begin{bmatrix} G'_P & 0 \\ 0 & I \end{bmatrix} \)

\( C'' = X^T C' X = \begin{bmatrix} C'_P & C''_T \\ C''_C & C''_I = \Lambda \end{bmatrix} \)

\[ T(s) = G'_P + sC'_P - \frac{s^2 r_1^T r_1}{1 + s \lambda_1} - \ldots - \frac{s^2 r_N^T r_N}{1 + s \lambda_N} \]

where \( r_i \) is the \( i \)-th row of \( C''_C \) and

\( \lambda_i \) is the \( i \)-th diagonal of \( C''_I \)
Elimination of Unwanted Poles

\[ G'' = \begin{bmatrix} G'_P & 0 \\ 0 & I \end{bmatrix} \quad C'' = \begin{bmatrix} C'_P & C''^T_C \\ C''_C & C''_I = \Lambda \end{bmatrix} \]

- Each pole is associated with a single internal node
- Unwanted poles are dropped by cutting the corresponding internal nodes
- Equivalent to removing the corresponding rows and columns in \( G'' \) and \( C'' \)
- Equivalent to removing the corresponding eigenvectors in \( U \) before the transformation is performed

- Which poles should we drop?
  - Given maximum frequency \( \omega_c \) and maximum error \( \varepsilon_c \)
  - If cut-off eigenvalue \( \lambda_c \) is defined by \( \varepsilon_c = \omega_c \lambda_c + \omega_c^3 \lambda_c^3 \)
  - Then, the error \( \varepsilon \) of dropping \( \lambda_i < \lambda_c \) is bounded by \( \varepsilon_c \)

\[
\varepsilon = \max_{k,l=[1\ldots m], \omega=[-\omega_c \ldots \omega_c]} \left[ \frac{1}{2} \left| T_{kk}(j\omega) + T_{ll}(j\omega) \right| \right]
\]
Computation of Poles

• Use symmetric Lanczos method to compute a sequence of Lanczos vectors $V_k$ and Lanczos matrix $T_k$

• Eigenvalues of $T_k$, known as Ritz values, approximates eigenvalues of $C'_I$

• Columns of $U$, known as Ritz vectors, are obtained by $V_k Z_k$ where $Z_k$ is the eigenvectors of $T_k$

• Use the Lanczos with selective orthogonalization
  – Ritz values usually converged first to extreme eigenvalues of $C'_I$
  – Additional vectors are calculated until those (large) Ritz values within a specified range have converged
  – Avoids loss of orthogonality which slows convergence of the less dominant eigenvalues and creates “ghosts” of dominant eigenvalues
  – Allows closely spaced or multiple occurrences of eigenvalues to be found