Modelling and Inverting Complex-Valued Wiener Systems

Xia Hong[a], Sheng Chen[b,c], Chris J. Harris[b]

[a] School of Systems Engineering, University of Reading, Reading RG6 6AY, UK
x.hong@reading.ac.uk

[b] Electronics and Computer Science, Faculty of Physical and Applied Sciences,
University of Southampton, Southampton SO17 1BJ, UK
sqc@ecs.soton.ac.uk cjh@ecs.soton.ac.uk

[c] Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia

IEEE World Congress on Computational Intelligence
Brisbane Australia, June 10-15, 2012

Outline

Outline

Background

1. **Complex-valued** neural networks have been applied widely in nonlinear signal processing and data processing
   1. many good techniques for identifying CV nonlinear models
   2. very few good techniques for inverting CV nonlinear models
2. Communication applications often involve complex-valued signals propagating through CV **Wiener** systems, which require
   - modelling and inverting CV Wiener systems
3. Digital **predistorter** design for broadband systems employing power-efficient nonlinear **high power amplifier**, which needs
   1. Identifying CV Wiener system that represents nonlinear HPA with memory
   2. Pre inverting identified Wiener model to obtain predistorter for compensating nonlinear HPA

Our Approach

1. **B-spline** neural networks with **De Boor** algorithm offers effective means of modelling Wiener systems
   - Best numerical properties, and computational efficiency
2. Our previous work has developed **complex-valued** B-spline model for complex-valued Wiener systems
   - **Tensor product** between two sets of univariate B-spline basis functions
   - Gauss-Newton algorithm with effective initialisation exploits efficiency of De Boor recursion
3. In this work, we further develop efficient technique for **inverting** complex-valued Wiener system with B-spline model
   - Gauss-Newton algorithm with efficient De Boor inverse
4. Our approach is applied to digital **predistorter** design

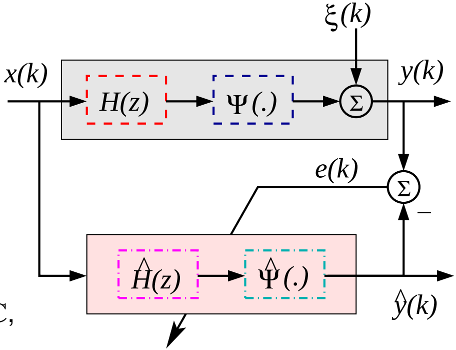
Outline

Wiener System

- CV Wiener system: cascade of **FIR filter** of order L

$$H(z) = \sum_{i=0}^{L} h_i z^{-i}, \ h_0 = 1$$

  followed by **nonlinear static** function $\Psi(\bullet) : \mathbb{C} \rightarrow \mathbb{C}$



- Specifically, given input $x(k) \in \mathbb{C}$,

$$w(k) = \sum_{i=0}^{L} h_i x(k-i) \ \text{ and } \ y(k) = \Psi\left(w(k)\right) + \xi(k)$$

  output $y(k) \in \mathbb{C}$, noise $\xi(k) \in \mathbb{C}$ with $E\left[|\xi_R(k)|^2\right] = E\left[|\xi_I(k)|^2\right] = \sigma_\xi^2$

- **Task**: given $\{x(k), y(k)\}_{k=1}^{K}$, identify $\Psi(\bullet)$ and $\boldsymbol{h} = \left[h_1 \cdots h_L\right]^{\mathrm{T}} \in \mathbb{C}^L$

Real B-Spline

Set of B-spline basis functions on $U_{\min} < w_R < U_{\max}$ is parametrised by piecewise polynomial of order $P_o - 1$, and knot vector of $(N_R + P_o + 1)$ knot values

1. $(N_R + P_o + 1)$ **knot values** break w_R-axis:

$$U_0 < \cdots < U_{P_o-1} = U_{\min} < U_{P_o} < \cdots < U_{N_R} < U_{N_R+1} = U_{\max} < \cdots < U_{N_R+P_o}$$

2. N_R B-spline **basis functions** $B_q^{(\Re,P_o)}(w_R)$, $1 \le q \le N_R$, by **De Boor recursion**

$$B_q^{(\Re,0)}(w_R) = \begin{cases} 1, & \text{if } U_{q-1} \le w_R < U_q, \\ 0, & \text{otherwise,} \end{cases} \quad 1 \le q \le N_R + P_o$$

$$B_q^{(\Re,p)}(w_R) = \frac{w_R - U_{q-1}}{U_{p+q-1} - U_{q-1}} B_q^{(\Re,p-1)}(w_R) + \frac{U_{p+q} - w_R}{U_{p+q} - U_q} B_{q+1}^{(\Re,p-1)}(w_R)$$

$$\text{for } q = 1, \cdots, N_R + P_o - p \text{ and } p = 1, \cdots, P_o$$

3. **Derivatives** of $B_q^{(\Re,P_o)}(w_R)$, $1 \le q \le N_R$, also by **De Boor recursion**

$$\frac{dB_q^{(\Re,P_o)}(w_R)}{dw_R} = \frac{P_o}{U_{P_o+q-1} - U_{q-1}} B_q^{(\Re,P_o-1)}(w_R) - \frac{P_o}{U_{P_o+q} - U_q} B_{q+1}^{(\Re,P_o-1)}(w_R)$$

Imaginary B-Spline

Similarly, set of B-spline basis functions on $V_{\min} < w_I < V_{\max}$ is parametrised by piecewise polynomial of order $P_o - 1$, and knot vector of $(N_I + P_o + 1)$ knot values

1. $(N_I + P_o + 1)$ **knot values** break w_I-axis:

$$V_0 < \cdots < V_{P_o-1} = V_{\min} < V_{P_o} < \cdots < V_{N_I} < V_{N_I+1} = V_{\max} < \cdots < V_{N_I+P_o}$$

2. N_I B-spline **basis functions** $B_m^{(\Im,P_o)}(w_I)$, $1 \le m \le N_I$, by **De Boor recursion**

$$B_m^{(\Im,0)}(w_I) = \begin{cases} 1, & \text{if } V_{m-1} \le w_I < V_m, \\ 0, & \text{otherwise}, \end{cases} \quad 1 \le m \le N_I + P_o$$

$$B_m^{(\Im,p)}(w_I) = \frac{w_I - V_{m-1}}{V_{p+m-1} - V_{m-1}} B_m^{(\Im,p-1)}(w_I) + \frac{V_{p+m} - w_I}{V_{p+m} - V_m} B_{m+1}^{(\Im,p-1)}(w_I)$$

for $m = 1, \cdots, N_I + P_o - p$ and $p = 1, \cdots, P_o$

3. **Derivatives** of $B_m^{(\Im,P_o)}(w_I)$, $1 \le m \le N_I$, also by **De Boor recursion**

$$\frac{dB_m^{(\Im,P_o)}(w_I)}{dw_I} = \frac{P_o}{V_{P_o+m-1} - V_{m-1}} B_m^{(\Im,P_o-1)}(w_I) - \frac{P_o}{V_{P_o+m} - V_m} B_{m+1}^{(\Im,P_o-1)}(w_I)$$

Complex-Valued B-Spline

- Form **tensor product** between $B_q^{(\Re,P_o)}(w_R)$, $1 \leq q \leq N_R$, and $B_m^{(\Im,P_o)}(w_I)$, $1 \leq m \leq N_I$, yields new set of B-spline basis functions $B_{q,m}^{(P_o)}(w)$
- Give rise to **complex-valued** B-spline neural network

$$\widehat{y} = \widehat{\Psi}(w) = \sum_{q=1}^{N_R} \sum_{m=1}^{N_I} B_{q,m}^{(P_o)}(w)\omega_{l,m} = \sum_{q=1}^{N_R} \sum_{m=1}^{N_I} B_q^{(\Re,P_o)}(w_R) B_m^{(\Im,P_o)}(w_I)\omega_{q,m}$$

- $\omega_{q,m} = \omega_{R_{q,m}} + \mathrm{j}\omega_{I_{q,m}} \in \mathbb{C}$ are complex-valued **weights**
- Complex-valued B-spline model equals to two real-valued B-spline ones

$$\widehat{y}_R = \sum_{q=1}^{N_R} \sum_{m=1}^{N_I} B_q^{(\Re,P_o)}(w_R) B_m^{(\Im,P_o)}(w_I)\omega_{R_{q,m}}$$

$$\widehat{y}_I = \sum_{q=1}^{N_R} \sum_{m=1}^{N_I} B_q^{(\Re,P_o)}(w_R) B_m^{(\Im,P_o)}(w_I)\omega_{I_{q,m}}$$

- Complexity of De Boor recursion is $\mathcal{O}(P_o^2)$, and thus complexity of CV B-spline model is approximately $3 \cdot \mathcal{O}(P_o^2) \Rightarrow P_o$ **is very small**

Outline

Gauss-Newton Algorithm

1. With $N = N_R N_I$, $\widehat{\boldsymbol{h}} = \widehat{\boldsymbol{h}}_R + \mathrm{j}\widehat{\boldsymbol{h}}_I$ as estimate of $\boldsymbol{h} = \boldsymbol{h}_R + \mathrm{j}\boldsymbol{h}_I$, and $\boldsymbol{\omega} = \boldsymbol{\omega}_R + \mathrm{j}\boldsymbol{\omega}_I$, **parameter vector** of Wiener model is

$$\boldsymbol{\theta} = \left[\theta_1 \ \cdots \theta_{2(N+L)}\right]^{\mathrm{T}} = \left[\boldsymbol{\omega}_R^{\mathrm{T}} \ \boldsymbol{\omega}_I^{\mathrm{T}} \ \widehat{\boldsymbol{h}}_R^{\mathrm{T}} \ \widehat{\boldsymbol{h}}_I^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{2(N+L)}$$

2. Minimise **cost function** $J_{\mathrm{SSE}}(\boldsymbol{\theta}) = \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon}$, with $e(k) = y(k) - \widehat{y}(k)$,

$$\boldsymbol{\varepsilon} = [\varepsilon_1 \cdots \varepsilon_{2K}]^{\mathrm{T}} = [e_R(1) \cdots e_R(K) \ e_I(1) \cdots e_I(K)]^{\mathrm{T}} \in \mathbb{R}^{2K}$$

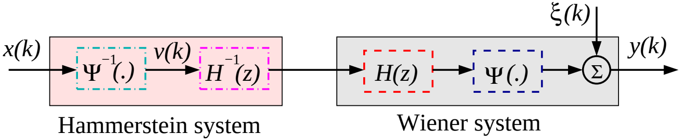
3. **Gauss-Newton** algorithm:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \mu\left(\left(\boldsymbol{J}^{(\tau)}\right)^{\mathrm{T}}\boldsymbol{J}^{(\tau)}\right)^{-1}\left(\boldsymbol{J}^{(\tau)}\right)^{\mathrm{T}}\boldsymbol{\varepsilon}\left(\boldsymbol{\theta}^{(\tau-1)}\right)$$

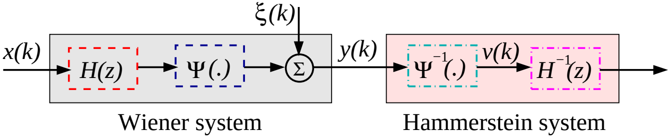
- **Jacobian** $\boldsymbol{J}$ of $\boldsymbol{\varepsilon}(\boldsymbol{\theta})$ can be evaluated efficiently with aid of De Boor recursions for B-spline functions and derivatives
- Biased LS estimates $\widehat{\boldsymbol{h}}^{(0)}$ and $\boldsymbol{\omega}^{(0)}$ can be quickly generated for parameter **initialisation** $\boldsymbol{\theta}^{(0)}$

Outline

Hammerstein Model



(a) Pre−Inverse



(b) Post−Inverse

Inverse of Wiener system is **Hammerstein** system, which consists of

- **Static nonlinearity** $\Psi^{-1}(\bullet)$ inverting static nonlinearity $\Psi(\bullet)$ in Wiener system,
- followed by **linear filter** $H^{-1}(z)$ inverting linear filter $H(z)$ in Wiener system

Inverse of Static Nonlinearity $\Psi(\bullet)$

- **Inverse** of CV Wiener system's static nonlinearity, defined by $v(k) = \Psi^{-1}(x(k))$, is identical to find complex-valued **root** of $x(k) = \Psi(v(k))$, given $x(k)$
- Given identified $\widehat{\Psi}(\bullet)$, we have

$$\widehat{x}_R(k) = \sum_{q=1}^{N_R} \sum_{m=1}^{N_I} B_q^{(\Re,P_o)}(v_R(k)) B_m^{(\Im,P_o)}(v_I(k)) \omega_{R_{l,m}}$$

$$\widehat{x}_I(t) = \sum_{q=1}^{N_R} \sum_{m=1}^{N_I} B_q^{(\Re,P_o)}(v_R(k)) B_m^{(\Im,P_o)}(v_I(k)) \omega_{I_{l,m}}$$

- Define $\zeta(k) = x(k) - \widehat{x}(k)$ and **cost function** $S(k) = \zeta_R^2(k) + \zeta_I^2(k) \Rightarrow$ If $S(k) = 0$, then $v(k)$ is CV root of $x(k) = \widehat{\Psi}(v(k))$
- With $U_{\min} < v_R^{(0)}(k) < U_{\max}$, $V_{\min} < v_I^{(0)}(k) < V_{\max}$, **Gauss-Newton** algorithm:

$$\begin{bmatrix} v_R^{(\tau)}(k) \\ v_I^{(\tau)}(k) \end{bmatrix} = \begin{bmatrix} v_R^{(\tau-1)}(k) \\ v_I^{(\tau-1)}(k) \end{bmatrix} - \eta \left(\left(\boldsymbol{J}_v^{(\tau)}\right)^{\mathrm{T}} \boldsymbol{J}_v^{(\tau)} \right)^{-1} \left(\boldsymbol{J}_v^{(\tau)}\right)^{\mathrm{T}} \begin{bmatrix} \zeta_R^{(\tau-1)}(k) \\ \zeta_I^{(\tau-1)}(k) \end{bmatrix}$$

- 2×2 **Jacobian** of $\zeta(k)$, $\boldsymbol{J}_v$, can also be evaluated efficiently with aid of De Boor recursions for B-spline functions and derivatives

Inverse of Linear Filter

1. Given identified **Wiener** system's linear filter

$$\widehat{H}(z) = \sum_{i=0}^{L} \widehat{h}_i z^{-i}$$

2. **Hammerstein** model's linear filter

$$G(z) = z^{-\iota} \cdot \sum_{i=0}^{L_g} g_i z^{-i}$$

3. can readily be obtained by solving set of **linear** equations

$$G(z) \cdot \hat{H}(z) = z^{-\iota}$$

4. Delay $\iota = 0$ if $H(z)$ is minimum phase, and $g_0 = 1$ as $h_0 = 1$
5. To guarantee accurate inverse, length of $\boldsymbol{g} = [g_0 \ g_1 \cdots g_{L_g}]^{\mathrm{T}}$ should be three to four times of length of $\boldsymbol{h}$

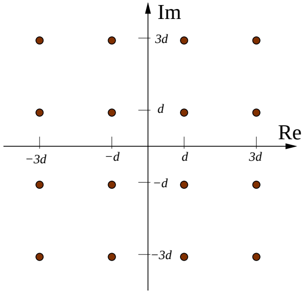
Outline

Wiener Model for HPA

- High power amplifier with **memory** is widely modelled as CV **Wiener** system
- CV input to HPA's **static nonlinearity** $\Psi(\bullet)$ is $w(k) = r(k) \cdot \exp(\mathrm{j}\psi(k))$
- Output of HPA is expressed as

$$y(k) = A(r(k)) \cdot \exp(\mathrm{j}(\psi(k) + \Phi(r(k))))$$

- ***M*-QAM** input $x(k)$ to HPA



16-QAM constellation

$$\mathbb{S} = \{d(2l - \sqrt{M} - 1) + \mathrm{j}d(2q - \sqrt{M} - 1), 1 \le l, q \le \sqrt{M}\}$$

- **Amplitude** and **phase** response of HPA's static nonlinearity are

$$A(r) = \begin{cases} \frac{\alpha_a r}{1+\beta_a r^2}, & 0 \le r \le r_{\mathrm{sat}}, \\ A_{\max}, & r > r_{\mathrm{sat}}, \end{cases} \quad \text{and} \quad \Phi(r) = \frac{\alpha_\phi r^2}{1 + \beta_\phi r^2}$$

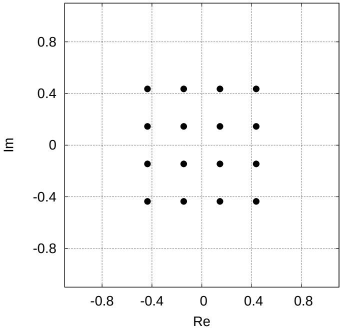
- r_{sat}: **saturation** input amplitude, $A_{\max}$: **saturation** output amplitude

A HPA Example

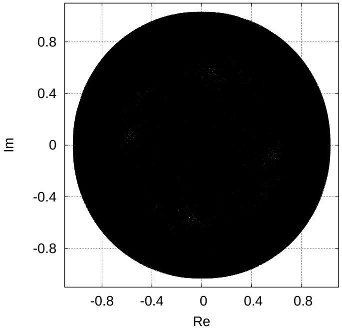
- Operating status of HPA is specified by **input back-off** (IBO),

$$\text{IBO} = 10 \cdot \log_{10} \frac{P_{\text{sat}}}{P_{\text{avg}}}$$

- **Parameters** of Wiener HPA: $\boldsymbol{h} = [0.75 + j0.2 \; 0.15 + j0.1 \; 0.08 + j0.01]^{\mathrm{T}}$ and $\boldsymbol{t} = [\alpha_a \; \beta_a \; \alpha_\phi \; \beta_\phi]^{\mathrm{T}} = [2.1587 \; 1.15 \; 4.0 \; 2.1]^{\mathrm{T}}$
- (a) HPA's input $x(k)$, and (b) HPA's output $y(k)$, given IBO$=$ 4 dB



(a)



(b)

Outline

Wiener HPA Identification

- B-spline model setting: piecewise **cubic** polynomial ($P_o = 4$), $N_R = N_I = 8$ with empirically determined **knot** sequence

$$\{-12.0, -6.0, -2.0, \ -\mathbf{1.2}, -0.6, -0.3, 0.0, 0.3, 0.6, \mathbf{1.2}, 2.0, 6.0, 12.0\}$$

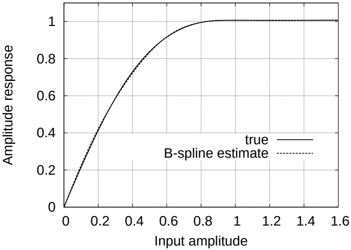
- Identification results for HPA's **linear** filter part $\boldsymbol{h}$

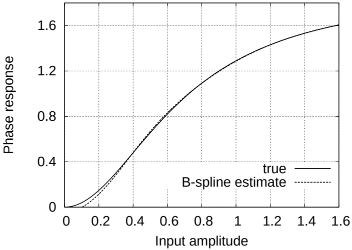
true parameter vector: $\boldsymbol{h}^{\mathrm{T}} = [0.7500 + \mathrm{j}0.2000 \ 0.1500 + \mathrm{j}0.1000 \ 0.0800 + \mathrm{j}0.0010]$
estimate under **IBO**$=$ 0 **dB**: $\widehat{\boldsymbol{h}}^{\mathrm{T}} = [0.7502 + \mathrm{j}0.1996 \ 0.1499 + \mathrm{j}0.0999 \ 0.0800 + \mathrm{j}0.0008]$
estimate under **IBO**$=$ 4 **dB**: $\widehat{\boldsymbol{h}}^{\mathrm{T}} = [0.7502 + \mathrm{j}0.2001 \ 0.1501 + \mathrm{j}0.1001 \ 0.0800 + \mathrm{j}0.0011]$

- At IBO$=$ 0 dB, HPA is heavily saturated

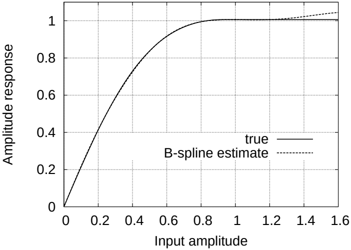
Results for HPA's Static Nonlinearity

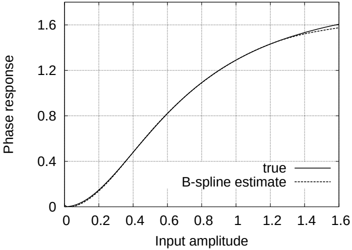
(a) IBO= 0 dB





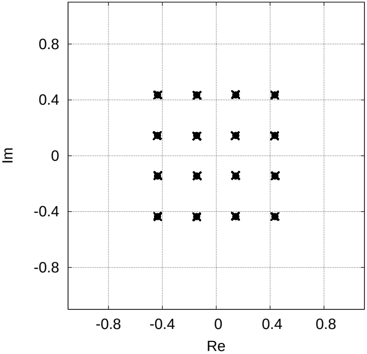
(b) IBO= 4 dB



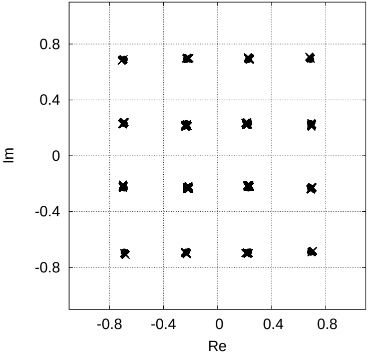


Predistorter Design

- Length of predistorter's **inverse filter** is set to $L_g = 12$.
- Output of **combined** predistorter and HPA $y(k)$, marked by $\times$, for 16-QAM input signal $x(k)$, marked by $\bullet$
- (a) IBO of 4 dB, and (b) IBO of 0 dB



(a)



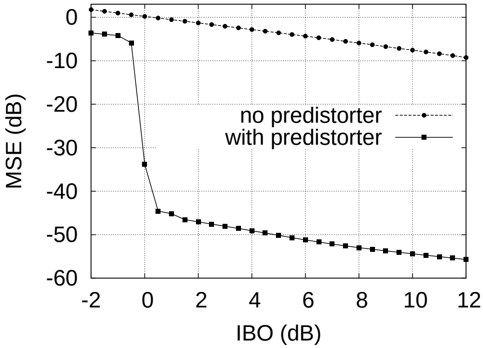
(b)

Mean Square Error

- **Mean square error** metric

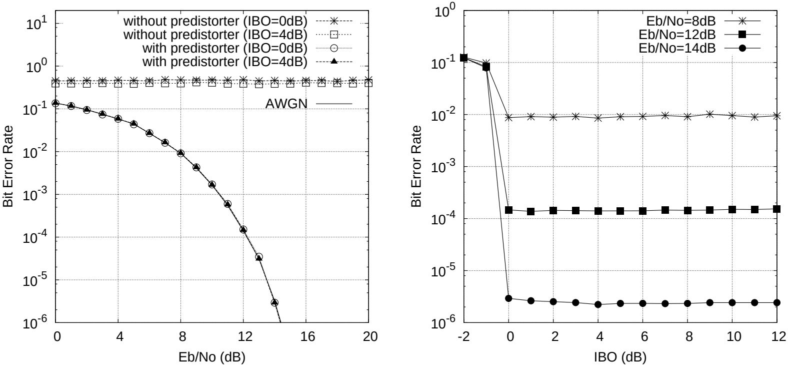
$$\text{MSE} = 10\log_{10}\left(\frac{1}{K_{\text{test}}}\sum_{k=1}^{K_{\text{test}}}|x(k)-y(k)|^2\right)$$

with $K_{\text{test}} = 10^5$ test samples



Bit Error Rate

- Output signal after HPA is transmitted over **additive white Gaussian noise** channel to determine **bit error rate** at receiver
- Channel **signal to noise ratio**: $\text{SNR} = 10\log_{10}\left(\text{E}_\text{b}/\text{N}_\text{o}\right)$, where E_b is energy per bit, and N_o power of channel's AWGN
- (a) BER versus SNR, and (b) BER versus IBO for different SNR



(a) (b)

Outline

Summary

1. Identification of complex-valued Wiener systems
   - Tensor product of two univariate B-spline neural networks to model Wiener system's static nonlinearity
   - Efficient Gauss-Newton algorithm for parameter estimate
   - Naturally incorporate De Boor recursions for both B-spline function values and derivatives
2. Accurate inverse of complex-valued Wiener systems
   - Inverse of complex-valued static nonlinearity is directly calculated from estimated B-spline model
   - Efficient Gauss-Newton algorithm for this inverting
   - Naturally utilise De Boor recursions for both B-spline function values and derivatives
3. Application to digital predistorter design for high power amplifiers with memory has been demonstrated