The Moog VCF

S. Bilbao

Virtual Analog Modeling

Basic idea: the digital emulation of classic analog audio effects!

- amp modelling
- effects (pedals etc.)
- electromechanical reverbs (plate and spring)

Big business---there are many companies doing VA modelling: SoundToys, Eventide, Native Instruments, UHE, etc. Go browse the DAFx proceedings if you want to get an idea of current directions.

Reduces, almost always, to the simulation of ordinary differential equations! Starting point: a circuit schematic!

Big issues:

- strong nonlinear effects (numerical stability)
- aliasing (often requiring operation at oversampled rates)
- system ID, maybe requiring machine learning approaches

Moog VCF

The Moog voltage-controlled filter: a classic early analog filtering component…used to filter raw waveforms

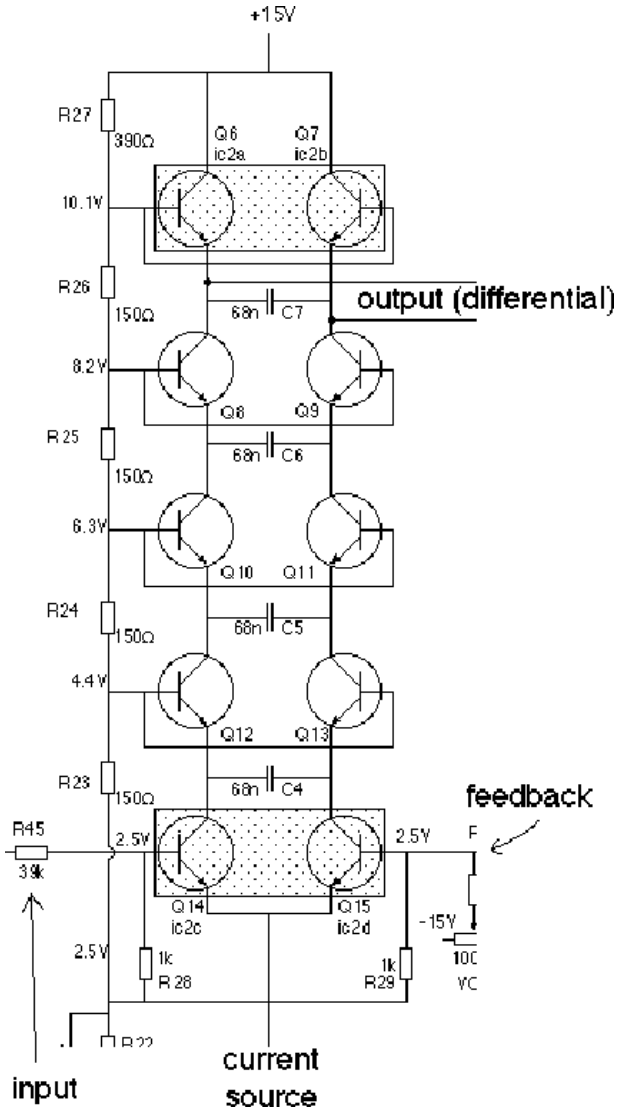
A ladder filter structure

Includes capacitors (meaning that the system has state, or memory)

Includes transistors, meaning that the filter as a whole is nonlinear…

Today: the linearised circuit

A great way to learn about numerical simulation methods…



System

Dynamics described by the following (dimensionless) system of equations:

$$\frac{1}{\omega_0}\frac{d}{dt}x_1 = -x_1 - 4rx_4 + u$$
$$\frac{1}{\omega_0}\frac{d}{dt}x_2 = -x_2 + x_1$$
$$\frac{1}{\omega_0}\frac{d}{dt}x_3 = -x_3 + x_2$$
$$\frac{1}{\omega_0}\frac{d}{dt}x_4 = -x_4 + x_3$$
$$y = x_4$$

$x_1(t), x_2(t), x_3(t), x_4(t)$: capacitance voltages
$u(t)$: input voltage
$y(t)$: output voltage

Parameters: $\omega_0 \geq 0$: angular frequency $0 \leq r \leq 1$: loss

NB: This is the linearised form---valid for small signal amplitudes

NB: Parameters in general will be time varying (they correspond to knobs!)

NB: assume initial conditions to be zero (this is a driven system!)

State Space Form

Can write this system in a canonical state space form:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{b}u \qquad y = \mathbf{c}^T\mathbf{x}$$

with:

$$\mathbf{x} = [x_1 \; x_2 \; x_3 \; x_4]^T$$

$$\mathbf{A} = \omega_0 \begin{bmatrix} -1 & 0 & 0 & -4r \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \qquad \mathbf{b} = \omega_0[1 \; 0 \; 0 \; 0]^T \qquad \mathbf{c} = [0 \; 0 \; 0 \; 1]^T$$

Laplace Transformation and Transfer Function

Laplace transformation of a function f(t)

$$\hat{f}(s) = \int_0^\infty f(t)e^{-st}dt \qquad\qquad s = \sigma + j\omega$$

System becomes (recalling we have zero initial conditions):

$$s\hat{\mathbf{x}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{b}\hat{u} \qquad \hat{y} = \mathbf{c}^T\hat{\mathbf{x}}$$

Transfer function, relating input to output:

$$\hat{y}(s) = H(s)\hat{u}(s) \qquad H(s) = \mathbf{c}^T\left(s\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{b}$$

Transfer function for real driving frequencies:

$$H(s = j\omega)$$

System Poles

Transfer function: $H(s) = \mathbf{c}^T (s\mathbf{I} - \mathbf{A})^{-1} \mathbf{b}$

System poles are determined entirely by the matrix A, or when

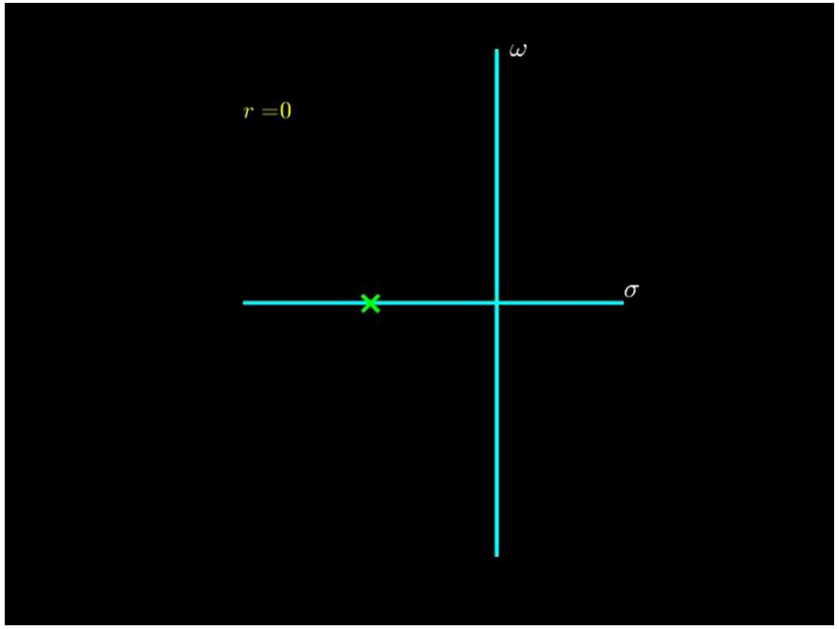
$$\det(s\mathbf{I} - \mathbf{A}) = 0 \qquad \rightarrow \qquad s \in \mathrm{eig}(\mathbf{A})$$

(This is where the transfer function becomes infinite!)

Eigenvalues of A:

$$\mathrm{eig}(\mathbf{A}) = \left\{ \begin{array}{l} \omega_0 \left(-1 + \sqrt{2} r^{1/4} e^{j\pi/4}\right) \\ \omega_0 \left(-1 + \sqrt{2} r^{1/4} e^{j3\pi/4}\right) \\ \omega_0 \left(-1 + \sqrt{2} r^{1/4} e^{j5\pi/4}\right) \\ \omega_0 \left(-1 + \sqrt{2} r^{1/4} e^{j7\pi/4}\right) \end{array} \right\}$$

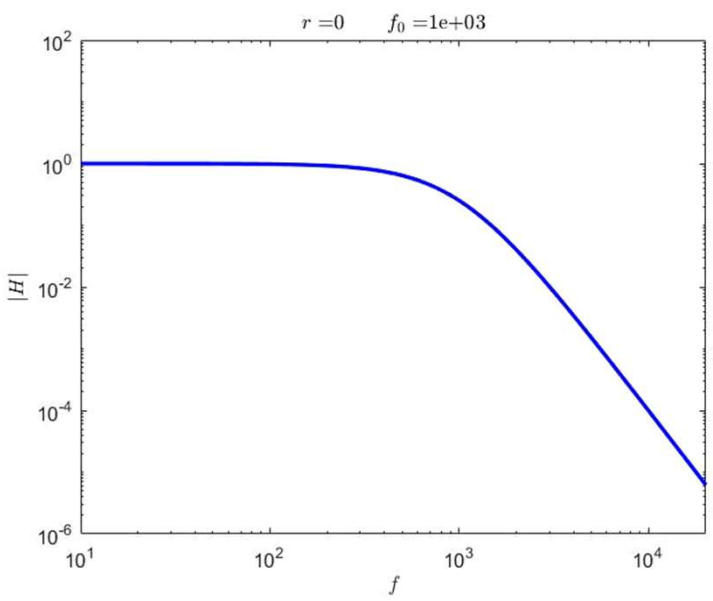
NB: all eigenvalues have nonpositive real part for r<=1---a lossy system!



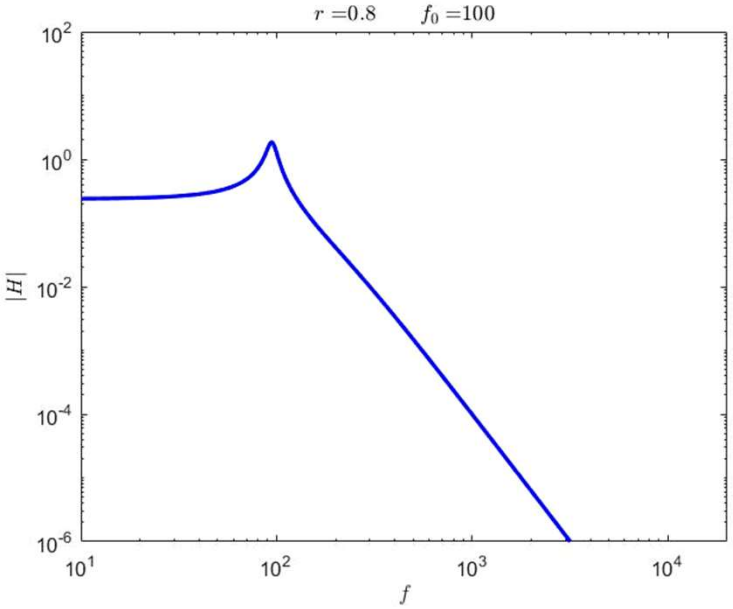
Sweeps

Useful to see the effect on the transfer function of sweeping the parameters:

Resonance parameter r



frequency



Behaves as a low pass for low r…and as a band pass for higher r

General form does not change (much!) as frequency is swept…

Numerical methods: Forward Euler

A first step: discretisation (time step k, sample rate 1/k): sample freq

$$\mathbf{x}(t), u(t), y(t) \qquad \rightarrow \qquad \mathbf{x}^n, u^n, y^n$$

Let's concentrate on just the state update part, so without input or output (these are affected little by the particular discretisation strategy)

The most basic approach: forward Euler:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} \qquad \rightarrow \qquad \delta_{t+}\mathbf{x} = \mathbf{A}\mathbf{x} \qquad \rightarrow \qquad \mathbf{x}^{n+1} = \mathbf{x}^n + k\mathbf{A}\mathbf{x}^n$$

An explicit numerical method (desirable!)…can update using only basic additions/multiplications

Numerical methods: Forward Euler

Under z transformation, or, alternatively, using exponential solutions:

$$\mathbf{x}^n = \hat{\mathbf{x}}^n z^n$$

we get

$$(z\mathbf{I} - \mathbf{I} - k\mathbf{A})\,\hat{\mathbf{x}} = 0$$

Nontrivial solutions: $z \in \mathrm{eig}\,(\mathbf{I} + k\mathbf{A})$

For numerical stability, require:

$$|z| \leq 1$$

This ensures that all natural frequencies of oscillation are damped…

For the Moog VCF, leads to a fairly complex condition (part of your assignment!):

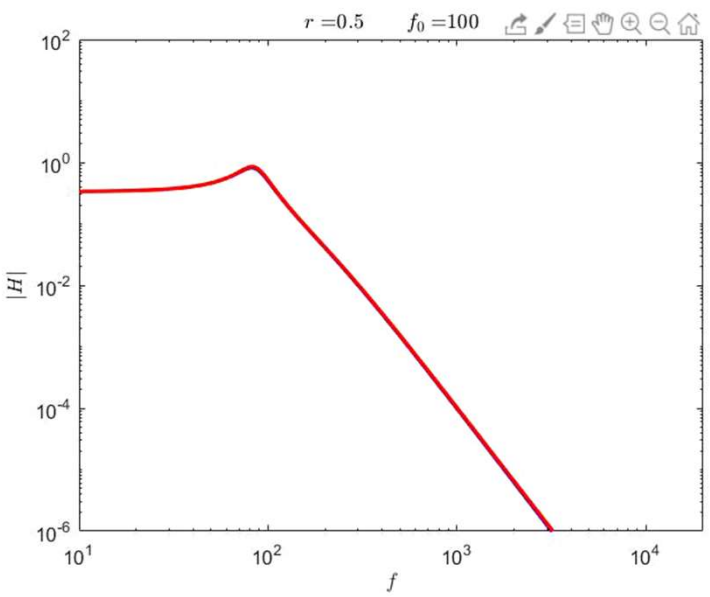
$$k \leq k_{max}\,(\omega_0, r)$$

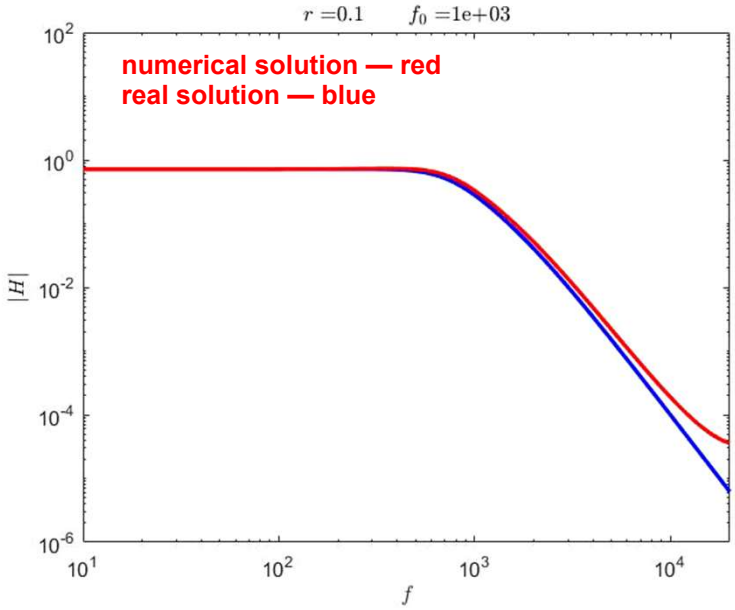
Complicated! Bound on time step depends on parameters…

Such bounds on the time step are typical of explicit methods

Forward Euler: Warping and Instability

Forward Euler is simple, but exhibits extreme frequency warping:





System is “underdamped”: resonances too sharp

Can improve this by oversampling (expensive!!!), which also helps with stability conditions

Not a good, or even usable design…but it is explicit, which is good.

Backward Euler

Now try: backward Euler:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} \qquad \rightarrow \qquad \delta_{t-}\mathbf{x} = \mathbf{A}\mathbf{x} \qquad \rightarrow \qquad (\mathbf{I} - k\mathbf{A})\,\mathbf{x}^{n+1} = \mathbf{x}^n$$

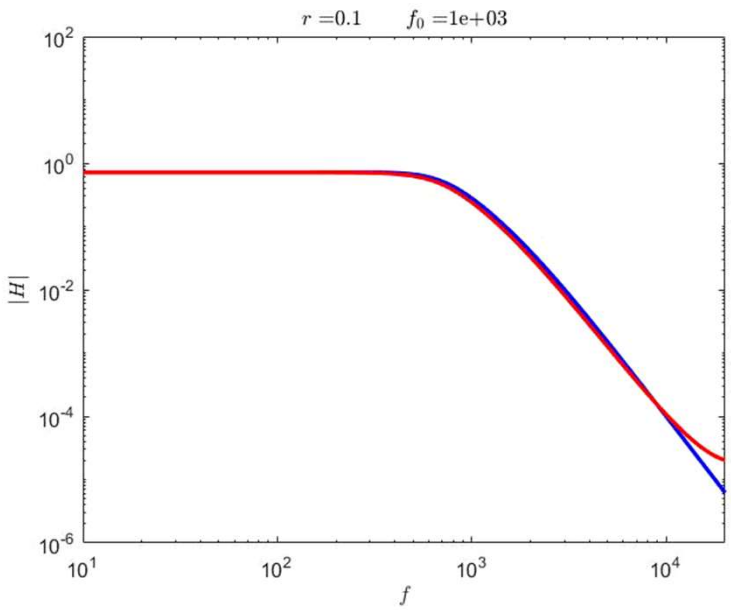
An implicit numerical method (less desirable!)…need to solve a linear system in order to update! But not too bad (it's only 4x4).

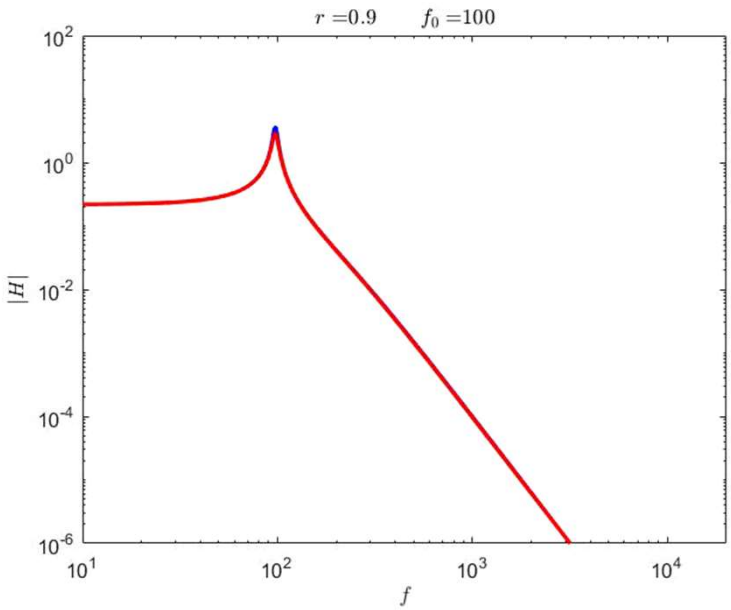
NB: You do not actually need to explicitly compute the inverse of I-kA. Rather, you need to solve a linear system. See the "backslash" operation in Matlab for this purpose.

Can show (again using z transform analysis): unconditionally stable, when Re(eig(A))<=0 (true for Moog VCF!)

Backward Euler: Warping

Backward Euler also exhibits extreme frequency warping:





System is “overdamped”: resonances too weak

Can again improve this by oversampling (expensive!!!)

Not a good, or even usable design…plus it is implicit, which is bad!

Trapezoid Rule

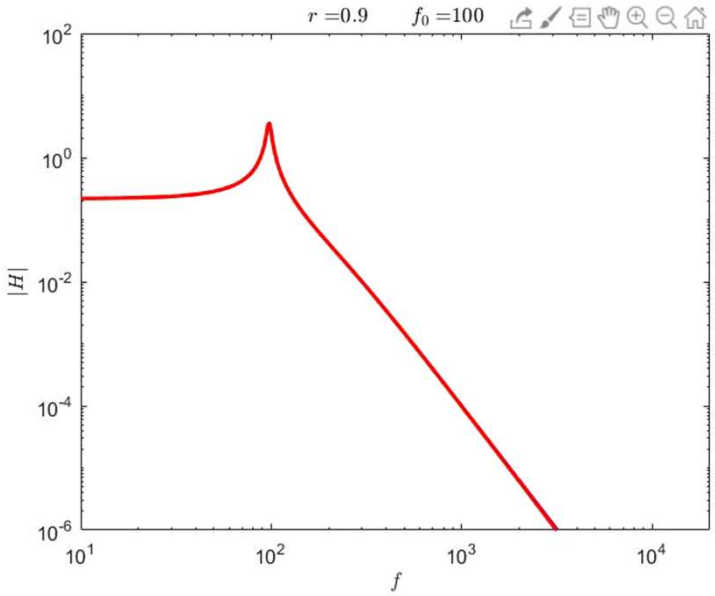
Now try: trapezoid rule:

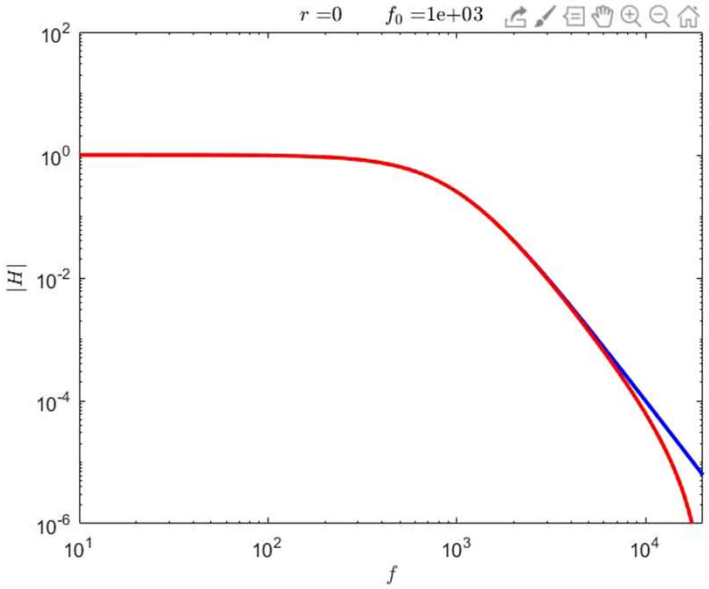
$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} \qquad \rightarrow \qquad \delta_{t+}\mathbf{x} = \mathbf{A}\mu_{t+}\mathbf{x} \qquad \rightarrow \qquad \left(\mathbf{I} - \frac{k}{2}\mathbf{A}\right)\mathbf{x}^{n+1} = \left(\mathbf{I} + \frac{k}{2}\mathbf{A}\right)\mathbf{x}^{n}$$

Still an implicit numerical method (like backward Euler)…need to solve a linear system in order to update! And it is even more computationally costly (see RHS).

Can show (again using z transform analysis): unconditionally stable, when Re(eig(A))<=0 (true for Moog VCF!)

Trapezoid Rule





Response is a lot better!

Still some warping in high frequency range---but 2x oversampling often used to combat this.

An excellent design---heavily used in industry!

Methods: in General

Just have showed three methods here:

FE: explicit (good), warping (bad), complex stability conditions (bad)

BE: implicit (bad), warping (bad), no stability conditions (good)

Trap: implicit (bad), warping (not bad), no stability conditions (good)

Are there others? Oh yes…try experimenting to find your own.

For example:

Parameterised: $\delta_{t+}\mathbf{x} = \mathbf{A}\left(\alpha\mu_{t+} + (1-\alpha)\right)\mathbf{x}$

Split: $\delta_{t+}\mathbf{x} = \left(\mathbf{A}_1\mu_{t+} + \mathbf{A}_2\right)\mathbf{x} \qquad \mathbf{A}_1 + \mathbf{A}_2 = \mathbf{A}$

Exact: $\mathbf{x}^{n+1} = e^{\mathbf{A}k}\mathbf{x}^n$ only use for no inputs

RK methods, etc.:

Time varying Setting

For a real Moog VCF emulation, we want to allow time variation of parameters:

$$\omega_0 = \omega_0^n \qquad\qquad r = r^n$$

and thus

$$\mathbf{A}^n = \omega_0^n \begin{bmatrix} -1 & 0 & 0 & -4r^n \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

This changes the updates, especially for implicit methods. For the trapezoid rule, for example,

$$\left(\mathbf{I} - \frac{k}{2}\mathbf{A}^{n+1}\right)\mathbf{x}^{n+1} = \left(\mathbf{I} + \frac{k}{2}\mathbf{A}^{n}\right)\mathbf{x}^{n}$$

This requires a linear system solution in the update---as the matrix on the left hand side must be recomputed at each time step!

Nonlinear Setting

The Moog VCF model presented here has been linear; but the actual model exhibits some weak nonlinear effects

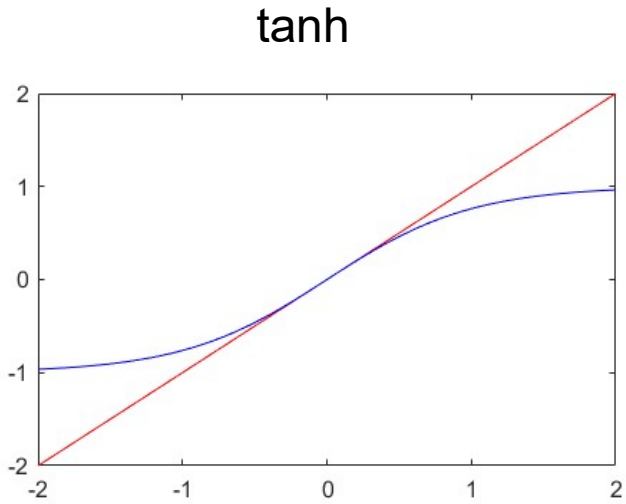
$$\frac{1}{\omega_0}\frac{d}{dt}x_1 = -\tanh(x_1) + \tanh(u - 4rx_4)$$

$$\frac{1}{\omega_0}\frac{d}{dt}x_2 = -\tanh(x_2) + \tanh(x_1)$$

$$\frac{1}{\omega_0}\frac{d}{dt}x_3 = -\tanh(x_3) + \tanh(x_2)$$

$$\frac{1}{\omega_0}\frac{d}{dt}x_4 = -\tanh(x_4) + \tanh(x_3)$$

$$y = x_4$$

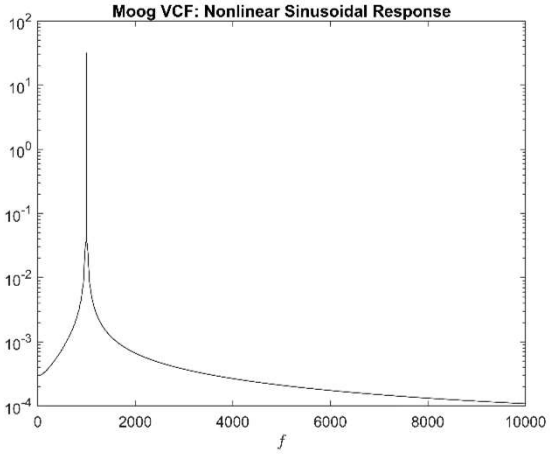


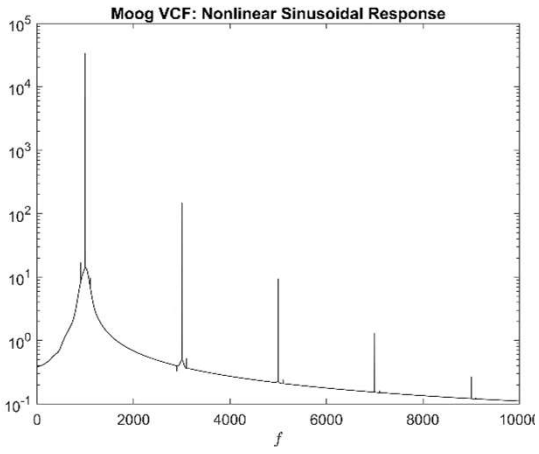
A soft clipping nonlinearity…reduces to the linear model at low amplitudes!

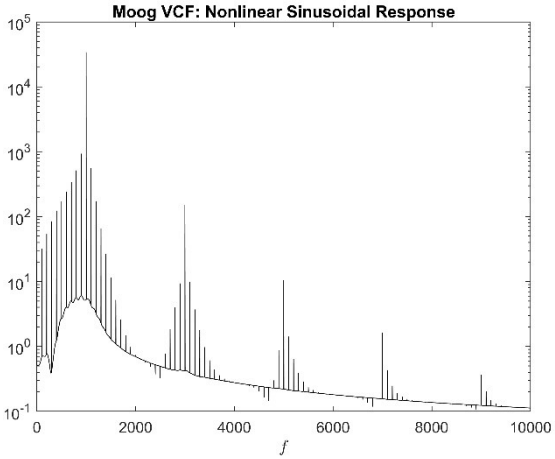
Numerical design is a lot harder now…no frequency domain tools!

Nonlinear Responses

Consider responses of the Moog VCF when the input is sinusoidal







Very low amplitude
(linear response)

Moderate amplitude
(harmonic distortion)

High amplitude
(aliasing!)

Aliasing---a huge problem! Lots of recent work on reducing this artefact
(see DAFx proceedings)

Solution Strategies

Can write Moog VCF system in vector form as

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, u\right) \qquad \mathbf{f}\left(\mathbf{x}, u\right) = \begin{bmatrix} \omega_0 \left(-\tanh(x_1) + \tanh(u - 4rx_4)\right) \\ \omega_0 \left(-\tanh(x_2) + \tanh(x_1)\right) \\ \omega_0 \left(-\tanh(x_3) + \tanh(x_2)\right) \\ \omega_0 \left(-\tanh(x_4) + \tanh(x_3)\right) \end{bmatrix}$$

All numerical methods presented earlier (FE, BE, trapezoid) extend directly.

Discrete time representation: $\mathbf{x}^n \qquad \mathbf{f}^n = \mathbf{f}\left(\mathbf{x}^n, u^n\right)$

Trapezoid rule is used heavily, due to its nice stability properties:

$$\delta_{t+}\mathbf{x} = \mu_{t+}\mathbf{f}\left(\mathbf{x}, u\right) \qquad \rightarrow \qquad \mathbf{x}^{n+1} - \mathbf{x}^n - \frac{k}{2}\left(\mathbf{f}\left(\mathbf{x}^{n+1}, u^{n+1}\right) + \mathbf{f}\left(\mathbf{x}^n, u^n\right)\right) = 0$$

Newton-Raphson: Scalar Case

Iterative approaches (e.g. Newton-Raphson) are a popular choice:

Suppose we want to solve:

$$G(w) = 0$$

First, choose a starting value (a guess) for x:

$$w = w^{(0)}$$

Now, iterate, using values of the function and its derivative at x^{(0)}:

$$w^{(m+1)} = w^{(m)} - G(w^{(m)})/G'(w^{(m)}$$

Can stop iterating when an appropriate tolerance level is reached, e.g.,

$$|w^{(m+1)} - w^{(m)}| \leq \epsilon$$

NB: Newton Raphson is surprisingly good…but, depending on the function, there may not be a unique solution! There are many much more advanced methods relying on special properties of the function (e.g., convex optimisation methods).

Newton-Raphson: Vector Case

For the VCF: given at time step n, x^{n} is known, and u^{n} is assumed known for all n, we have a nonlinear equation to solve:

$$\underbrace{\mathbf{x}^{n+1} - \mathbf{x}^{n} - \frac{k}{2}\left(\mathbf{f}\left(\mathbf{x}^{n+1}, u^{n+1}\right) + \mathbf{f}\left(\mathbf{x}^{n}, u^{n}\right)\right)}_{\mathbf{G}(\mathbf{x}^{n+1})=\mathbf{G}(\mathbf{w})} = 0$$

Now, G is a vector (4x1) function of a vector unknown w (4x1):

Newton Raphson looks like:

$$\mathbf{w}^{(m+1)} = \mathbf{w}^{(m)} - \mathbf{J}^{-1}\mathbf{G}(\mathbf{w}^{(m)})$$

J is the Jacobian, or vector derivative (4x4) of G at w.

Caution: do not confuse the time step (n) with the iteration number (m)!