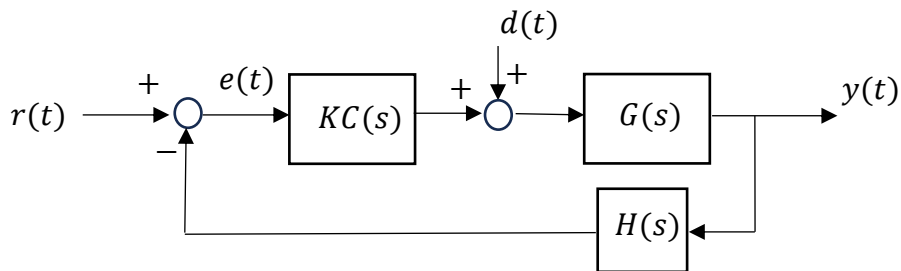


Week 11 Slides – Sensitivity and State-Space



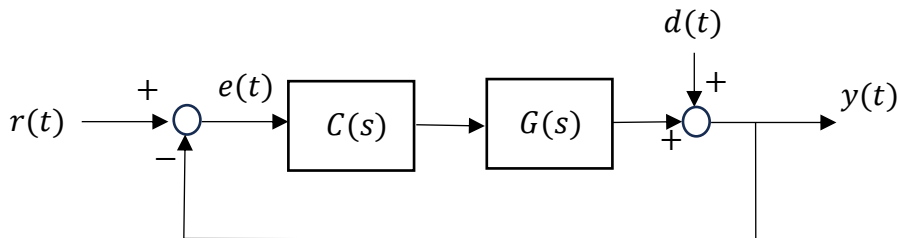
As we know, feedback control uses high gain to give good control ($T \approx 1$) and suppress disturbances. In the above

$$\frac{Y(s)}{D(s)} = \frac{G(s)}{1 + L(s)}$$

so the disturbance has little effect as long as $L \gg G$ (for frequencies within the designed operating range of the control system)

Sensitivity function

The disturbance is 'cleaner' if we add it after the plant – it still represents an external disturbance, but not necessarily due to something happening at the input of the plant. Examples of both types?



In this case (with K absorbed into C) the sensitivity to disturbance at the output is

$$\frac{Y(s)}{D(s)} = \frac{1}{1 + C(s)G(s)}$$

This is the sensitivity function $S(s)$

Note: $d(t)$ physically disturbs the plant we want to control – this is completely different from sensor noise, which only affects our measurement of the output.

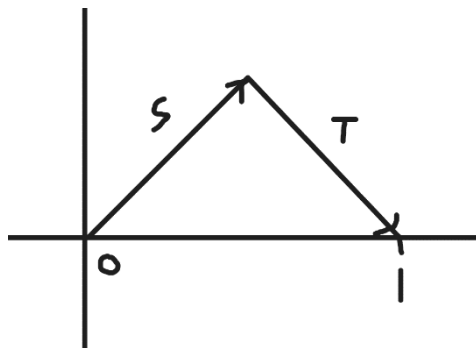
The combined effects of reference and disturbance is ‘linear superposition’ (addition!)

$$y = \frac{CG}{1 + CG} r + \frac{1}{1 + CG} d \equiv T(s)r + S(s)d$$

Also (check!)

$$S(s) + T(s) = 1$$

For any given frequency $s = j\omega$, S and T are complex numbers, they act like vecors.



- at low frequencies $T \approx 1$ and $S \approx 0$ – the control system is actively tracking the reference
- at high frequencies $T \approx 0$ and $S \approx 1$ – the control system is ignoring the reference and disturbances are no longer suppressed.

The amplitudes (gains) can be larger than one but not less:

$$|S| + |T| \geq 1$$

(this is the ‘triangle inequality’ which applies to any vector addition).

In control system design we want to avoid either frequency response gain being much greater than 1. (why is that?)

System Sensitivity

In general, for any engineering system we will be concerned if a disturbance or other typical change has a big effect on the function of the system.

If things work **ONLY** when the design assumptions are exact, ...



A baby on an elephant isn't likely to affect its operation (e.g. top speed). A small change in mass won't affect the performance variable of interest.

Apparently a mouse will scare an elephant – sensitivity is high and performance will be greatly affected!

Some people have this problem with spiders (irrational?) or snakes (rational?)

Suppose p is a parameter of an engineering system (mass, length, velocity etc.) and the performance output is found from an equation

$$y = f(u, p)$$

where u is an input.

If we change p by 1% and then y changes by 10% this would be a bit of a problem – the system is overly sensitive to the change in mass, length,



velocity etc. Unless p is tightly controlled we need this sensitivity to be small. Examples of systems with high and low sensitivity?

The sensitivity of y with respect to p is measured by the ratio of relative changes

$$S = \frac{\delta y/y}{\delta p/p}$$

Whatever the variables (mass, length, voltage, ...) S will be a dimensionless quantity.

More formally we can say the sensitivity of y with respect to p is

$$S_p^y = \frac{p}{y} \frac{\partial y}{\partial p}$$

the partial derivative indicating that other variables or parameters are held constant.

In the same way we can find the sensitivity of one transfer function as another is changed. For example the plant transfer function (or, more realistically, the frequency response) might change by a small amount, e.g., because a mass or electrical resistance change.

The sensitivity of the closed-loop transfer function with respect to changes in the plant TF is

$$S_G^T = \frac{G}{T} \frac{\partial T}{\partial G}$$

Using $T = CG/(1 + CG)$ we find $S_G^T = 1/(1 + CG) = S$

So the sensitivity function lives up to its name. And if G changes because of a parameter change, we can use the chain rule

$$S_p^T = \frac{p}{T} \frac{\partial T}{\partial p} = \frac{p}{T} \frac{\partial T}{\partial G} \frac{\partial G}{\partial p} = \frac{G}{T} \frac{\partial T}{\partial G} \cdot \frac{p}{G} \frac{\partial G}{\partial p} = S_G^T \cdot S_p^G = S \cdot S_p^G$$

The sensitivity with respect to a parameter depends on both S and sensitivity of the plant to the parameter in question.

Example – mass, spring damper with force input and displacement output:

$$G = \frac{1}{ms^2 + cs + k}$$

Then

$$S_m^G = \frac{m}{G} \frac{\partial G}{\partial m} = m(ms^2 + cs + k) \cdot (ms^2 + cs + k)^{-2} s^2 = ms^2 G$$

Hence

$$S_m^T = \frac{ms^2G}{1 + CG}$$

which is small provided $|C(j\omega)|$ is large.

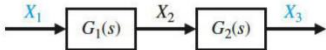
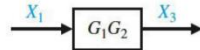
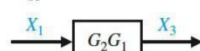
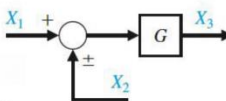
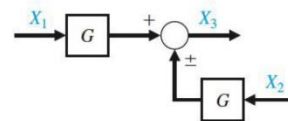
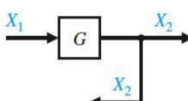
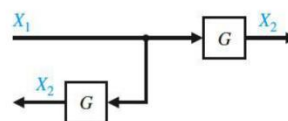
On the other hand, the sensitivity to sensor dynamics is (see tutorial problem)

$$S_H^T = \frac{H}{T} \frac{\partial T}{\partial H} = -HT$$

Note: the negative sign is largely irrelevant, it's the frequency response magnitude $|S_H^T| = |H||T|$ that's important. And in the operating frequency range of the control system $|S_H^T| \approx 1$, not big but not small either.

Block Diagrams – transformations and reduction

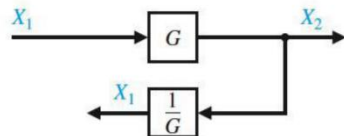
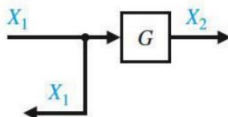
The following just use the simple property: $output = TF \times input$

Transformation	Original Diagram	Equivalent Diagram
1. Combining blocks in cascade		 or 
$X_3 = G_2 X_2 = G_2 (G_1 X_1) = G_2 G_1 X_1$		
2. Moving a summing point behind a block		
$X_3 = G(X_1 \pm X_2) = GX_1 \pm GX_2$		
3. Moving a pickoff point ahead of a block		
$X_2 = GX_1$		

Further equalities use an inverse transfer function $1/G = G^{-1}$

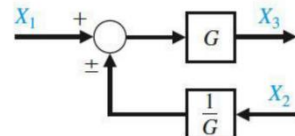
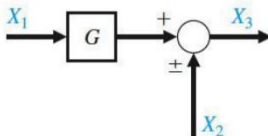
Moving a pickoff point behind a block

$$X_2 = GX_1$$



Moving a summing point ahead of a block

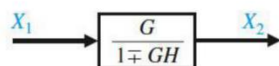
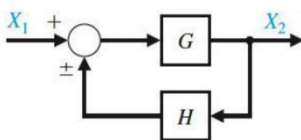
$$X_3 = GX_1 \pm X_2$$



Eliminating a feedback loop

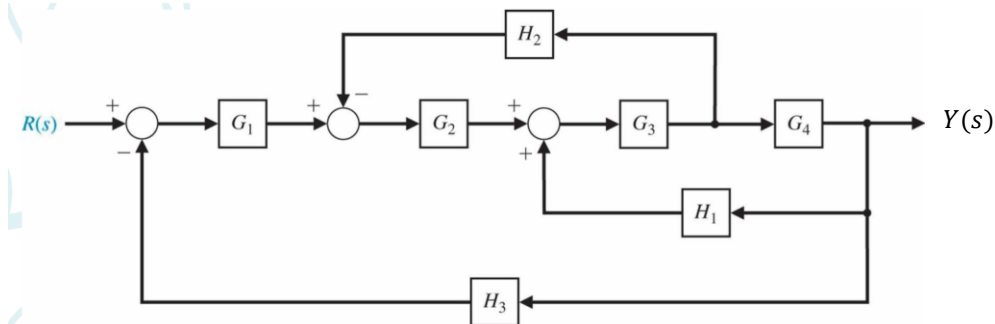
$$X_2 = G(X_1 \pm HX_2)$$

$$(1 \mp GH)X_2 = GX_1$$

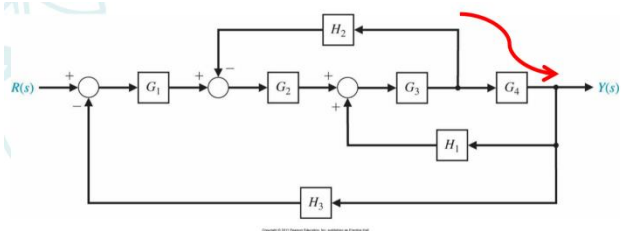


... and of course the familiar feedback loop

Here the feedback loops overlap, which complicates the diagram

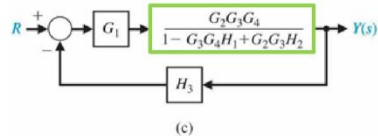
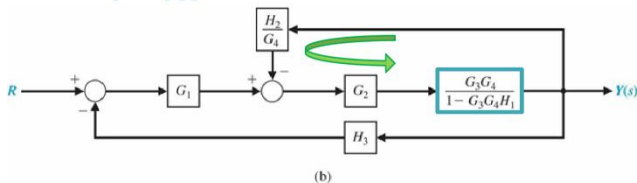
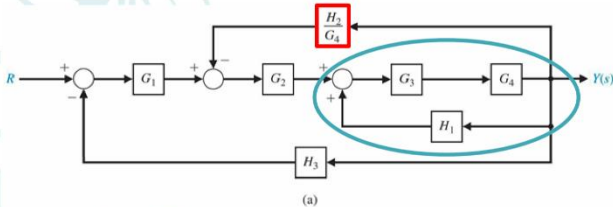


Below is one way to simplify and find the overall transfer function



$$R(s) \rightarrow \boxed{\frac{G_1 G_2 G_3 G_4}{1 - G_3 G_4 H_1 + G_2 G_3 H_2 + G_1 G_2 G_3 G_4 H_3}} Y(s)$$

(d)



Tutorial Questions

1. An LTI plant is represented by the transfer function

$$G(s) = \frac{s + 8}{s^3 + 7s^2 + 15s + 25}$$

The unity feedback system has a controller of the form

$$C(s) = K(1 + 0.1s)$$

Use Matlab to obtain the bode plot of the closed loop transfer function

$$T = \frac{CG}{1 + CG}$$

in the cases (i) $K=140$ (ii) $K=400$. Estimate the bandwidth in each case. Also find the closed-loop step responses and comment on the relationship between bandwidth and speed of response.

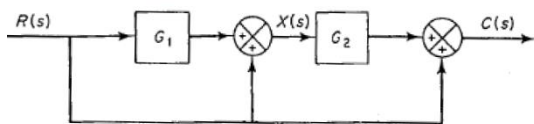
- Plot the sensitivity function $S = 1 - T$ for the two closed-loop systems of Problem 1. Check the bandwidth of S in each case and comment.
- Show that for

$$T = \frac{CG}{1 + CGH}$$

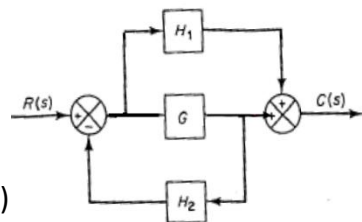
the sensitivity function with respect to changes in sensor performance is given by

$$S_H^T = \frac{H}{T} \frac{\partial T}{\partial H} = -HT$$

- Determine the overall transfer function $C(s)/R(s)$ in each of the following cases [note, here $c(t)$ is an output signal]



(a)



(b)

State-Space

The module so far has focused almost entirely on representing control systems via transfer functions, block diagrams, transfer functions and mostly on SISO systems (single-input, single-output).

“State space” simply means modelling a system using first-order differential equations. This is achieved by introducing a suitable number of **state variables** x_1, x_2, \dots

For a mass-spring damper (second order) we might choose $x_1 = y, x_2 = \dot{y}$. In an electrical system we might use current or voltage as state variables.

Transfer function and state-space models are interchangeable, but there are differences:

- state-space models the ‘internal workings’ of a system, while transfer functions only consider inputs and outputs – the ‘box is transparent’

- state-space deals with MIMO systems more easily than transfer functions.
- state-space offers a wider range of standard controller design methods (such as pole placement)
- state-space provides a variety of new tools to help the designer (such as 'state observers')

A (nonlinear) state-space SISO model looks like this:

$$\frac{dx_1}{dt} = f_1(x_1, x_2, u)$$

$$\frac{dx_2}{dt} = f_2(x_1, x_2, u)$$

$$y = g(x_1, x_2, u)$$

As with the TF approach we have input u and output y .

There are also the internal variables (x_1, x_2). This is a general second-order state-space system. The plant dynamics are fully defined by the functions f_1, f_2, g .

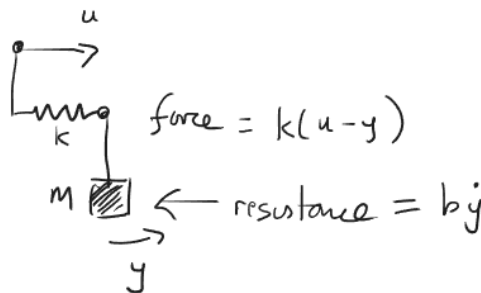
For state-space, design and analysis are performed in the familiar time domain, we hardly need to talk about the 's-domain' or the 'frequency domain' except when it's useful. So, in many ways state-space is easier to understand

Example Find the state-space equations for the following system.

The equation of motion is

$$m\ddot{y} = k(u - y) - b\dot{y}$$

Which has second-order derivatives



Define state variables

$$x_1 = y, x_2 = \dot{y}$$

The equations of motion are then

$$\dot{x}_1 = x_2$$

$$m\dot{x}_2 = k(u - x_1) - bx_2$$

The standard format is then

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{k}{m}x_1 - \frac{b}{m}x_2 + \frac{k}{m}u$$

All second-order SISO systems have the same general form, defined by the coefficients of input and state variables.

$$\dot{x}_1 = a_{11}x_1 + a_{12}x_2 + b_1u$$

$$a_{11} = 0, a_{12} = 1, b_1 = 0$$

$$\dot{x}_2 = a_{21}x_1 + a_{22}x_2 + b_2u$$

$$a_{21} = -\frac{k}{m}, a_{22} = -\frac{b}{m}, b_2 = \frac{k}{m}$$

These are the 'state equations'. We also need 'output equations' to represent the measurement.

$$y(t) = x_1$$

In general

$$y = c_1x_1 + c_2x_2 + du$$

$$c_1 = 1, c_2 = 0, d = 0$$

The equations are conveniently put in matrix format:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} u$$

$$y = (c_1 \quad c_2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (d)u$$

A second-order MIMO system with two inputs and two outputs has the same general structure

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

System modelling will determine the coefficients in the matrices.

Defining the input, state, and output variables as **vectors**

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

we can write the state and output equations in the compact form

$$\dot{\mathbf{x}} = \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{u}$$

$$\mathbf{y} = \mathbf{C} \mathbf{x} + \mathbf{D} \mathbf{u}$$

The format stays the same even if we have a complex system with a large number of states and inputs (e.g., an FE model of a bridge with 100,000 degrees of freedom!)

There are tools in Matlab to convert between SS and TF forms as well as special tools for state-space controller design, e.g. PLACE which puts closed-loop poles wherever the designer wants them!

Linearization

Finally, if a system is nonlinear it's often useful to design a controller using a linear approximation.

Linearization is when a linear approximation is applied to the nonlinear plant equations. For example, a pendulum model uses the following function

$$f(x) = \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots \approx x$$

where x is the angle of the pendulum from vertical. Assuming x is small the higher powers are very very small and are ignored. Linearization does not require a state-space form, but it's easier to write in a general way:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \approx \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)_0 (\mathbf{x} - \mathbf{x}_0) + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right)_0 (\mathbf{u} - \mathbf{u}_0)$$

The state variable is then redefined as $\mathbf{x} - \mathbf{x}_0$ and the state-space coefficients are found from

$$A = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)_0 \quad B = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right)_0$$

Example of linearization - pendulum

Example – nonlinear oscillator

Linear: $m\ddot{x} + c\dot{x} + kx = 0$

Nonlinear: $m\ddot{x} + c(x^2 - 1)\dot{x} + kx = 0$

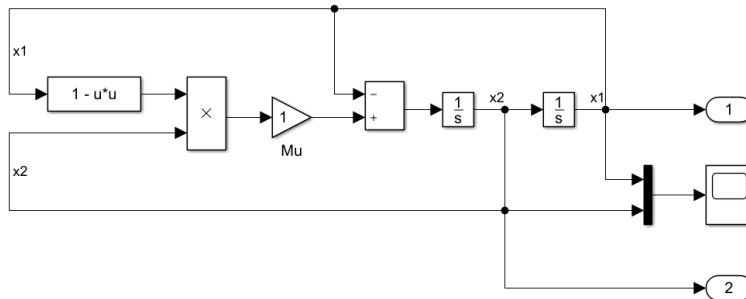
In each case the state-space form $x_1 = x, x_2 = \dot{x}$ is convenient.

The above nonlinear oscillator has a damping term which becomes negative near the origin but positive far from the origin. This simple system is known as the *Van der Pol oscillator*.

Matlab has a built-in Simulink example for this system (with $k = m = c = 1$)
>> vdp



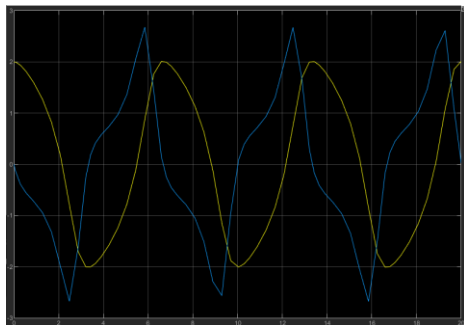
van der Pol Equation



Copyright 2004-2013 The MathWorks, Inc.

The model can be 'decoded' by focusing on the inputs to the integrator blocks, so $\dot{x}_1 = x_2$, $\dot{x}_2 = \text{output of the } \pm \text{ block}$.

While the **vdP** equations are simple, the response can be quite complicated. The Simulink model is set up to simulate an initial condition response, from $x_1(0) = 0, x_2(0) = 2$



x_1 : yellow, x_2 : blue

Phase Portraits

For 2D systems it is common to draw a set of paths in state-space, with one diagram showing the results of many different initial conditions.

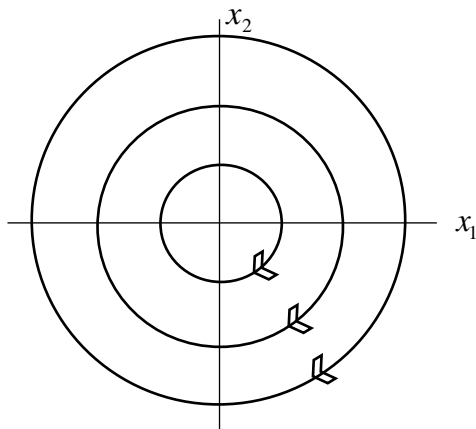
For the linear system

$$m\ddot{x} + c\dot{x} + kx = 0$$

($m = k = 1, c = 0$) this is a set of nested circles.

Note

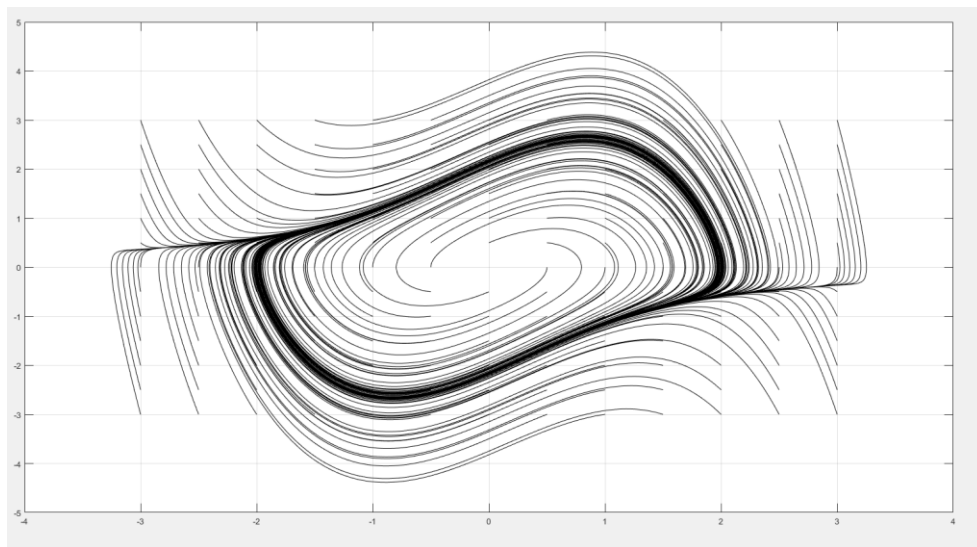
- We know the direction of motion, even if the arrows are not shown. (How?)



For vdp

$$m\ddot{x} + c(x^2 - 1)\dot{x} + kx = 0 \quad (m = c = k = 1)$$

the results are as follows



This was carried out by repeat simulation of the vdp Simulink model. Note the effect of negative damping near the origin, positive damping for larger distances.

Tutorial Questions (continued)

5. For the nonlinear system

$$m\ddot{x} + c \sin \dot{x} + kx = 0$$

linearize the equations for small displacements and velocities and decide whether the linear system is stable.

6. Write down the (linear and nonlinear) state equations for the system of Problem 5. (set $x_1 = x$, $x_2 = \dot{x}$ and re-arrange the equations so they are first-order differential equations)
7. Referring to the given Matlab code (vpd_portrait.m) reproduce the phase plane plot shown above.
8. [*slightly tricky, can skip] Determine the phase portrait of the nonlinear system

$$\ddot{x} + 0.6\dot{x} + 3x + x^2 = 0$$