Christopher Lum
lum@uw.edu

Lecture 04h
The Jacobian Matrix

Lecture is on YouTube

The YouTube video entitled 'The Jacobian Matrix' that covers this lecture is located at https://youtu.be/QexBVGVM690

Outline

-Introduction
-Derivative
-Gradient
-The Jacobian Matrix
    -History
-Examples
    -Mathematical Example
    -Nonlinear to Linear ODE

Introduction

The Jacobian matrix is simply a way to compute the sensitivity of a function to perturbations in the inputs variables.

Derivative

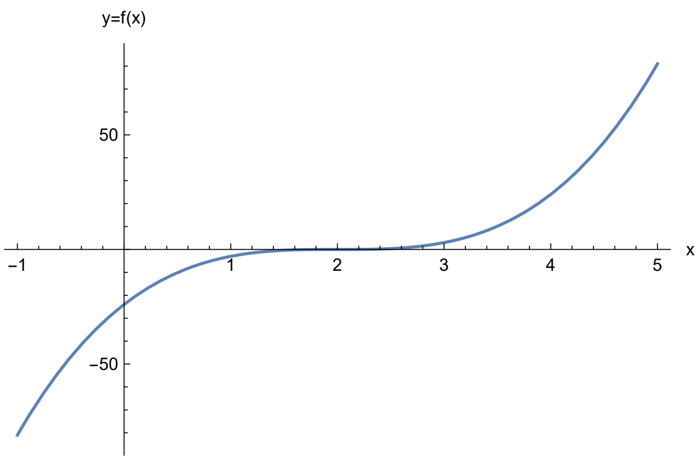
The derivative of a single input, single output function simply tells how sensitive the function's output is to perturbations in the function's input.

In[∘]:=
```
f[x_] = 3 (x - 2)^3;

Print["df (x)/dx"]
dfdx[x_] = D[f[x], x]

Plot[f[x], {x, -1, 5},
 AxesLabel → {"x", "y=f(x)"}]

Plot[dfdx[x], {x, -1, 5},
 AxesLabel → {"x", "df (x)/dx"}]
```

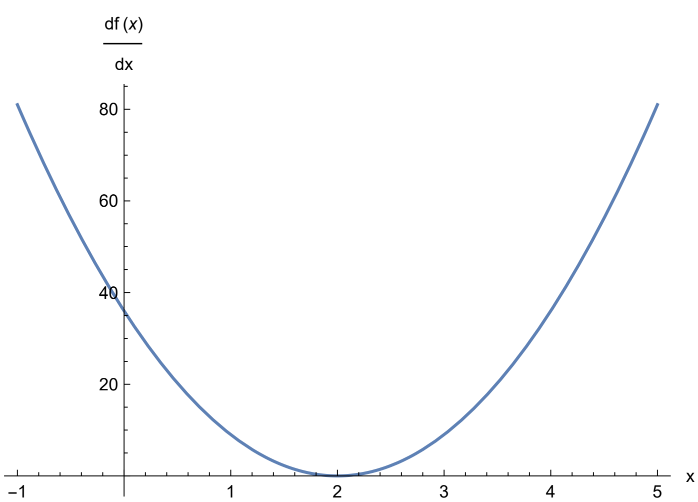
$\frac{df\ (x)}{dx}$

Out[∘]= $9\ (-2 + x)^2$

Out[∘]=



Out[∘]=



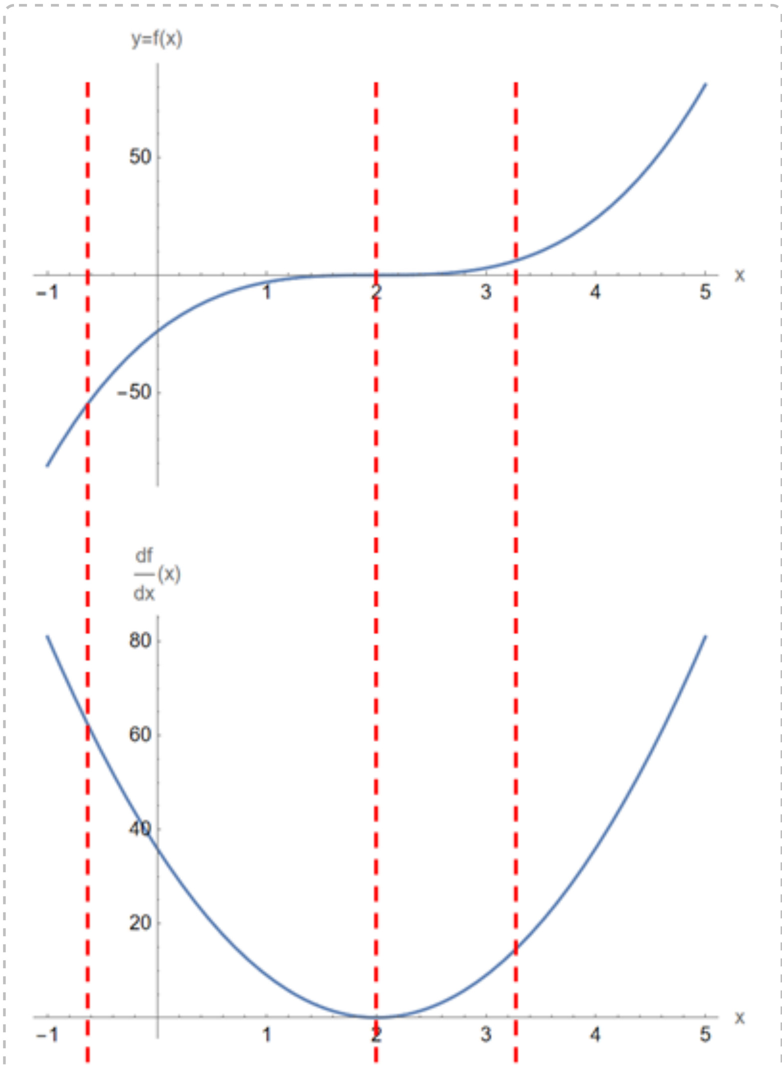
So we see that at $\frac{d\,f(x)}{d\,x}$ measures the sensitivity of the function and this sensitivity depends on the value of x (AKA the input).

In[∘]:=
```
dfdx[-0.75]
dfdx[2]
dfdx[3.25]
```

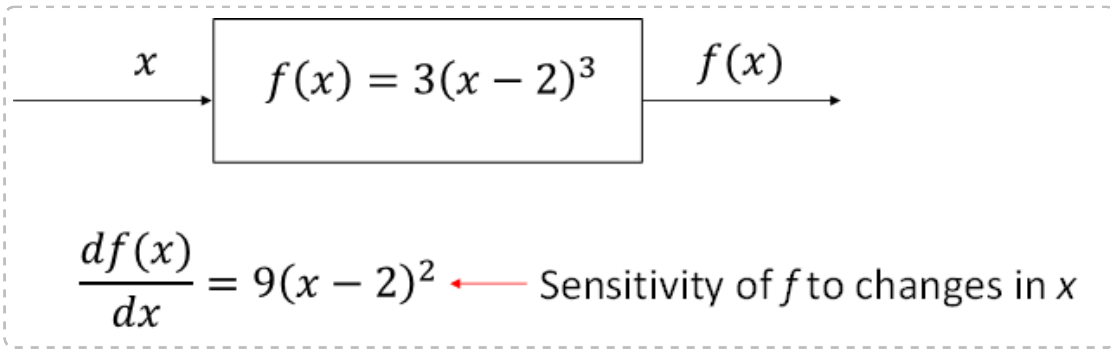
Out[∘]= 68.0625

Out[∘]= 0

Out[∘]= 14.0625



We can visualize this as



Gradient

If the function has multiple inputs

$$\overline{x} = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}$$

then instead of a single derivative, we instead compute the gradient of the function (see previous video entitled 'Gradient of a Function and the Directional Derivative' at https://y-outu.be/obeu4B8mXuw)

$$\nabla f(\overline{x}) = \frac{\partial f(\overline{x})}{\partial \overline{x}} = \begin{pmatrix} \frac{\partial f(\overline{x})}{\partial x_1} \\ \frac{\partial f(\overline{x})}{\partial x_2} \\ \ldots \\ \frac{\partial f(\overline{x})}{\partial x_n} \end{pmatrix}$$

We note that f is still a scalar function as it has only 1 output (see previous video entitled 'Scalar Functions, Vector Functions, and Vector Derivatives' at https://youtu.be/haJVEtLN6-k)

In[ ]:=
```
f[x1_, x2_] = 3 x1^2 + x2^3;
gradF[x1_, x2_] = (D[f[x1, x2], x1]
                   D[f[x1, x2], x2]);
gradF[x1, x2] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 6\ \text{x1} \\ 3\ \text{x2}^2 \end{pmatrix}$$

Again, we see that the gradient measures the sensitivity of the function w.r.t. perturbations in either x_1 or x_2 and this sensitivity depends on the value of x_1 and x_2 (AKA the input). In other words

$\frac{\partial f(\overline{x})}{\partial x_k}$ = how sensitive the output of the function f at the point $\overline{x}$ is to perturbations in x_k

In[ ]:=
```
x1A = -3;
x2A = 2;
gradF[x1A, x2A] // MatrixForm

x1B = 0;
x2B = 0;
gradF[x1B, x2B] // MatrixForm

x1C = 2;
x2C = 0;
gradF[x1C, x2C] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -18 \\ 12 \end{pmatrix}$$

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

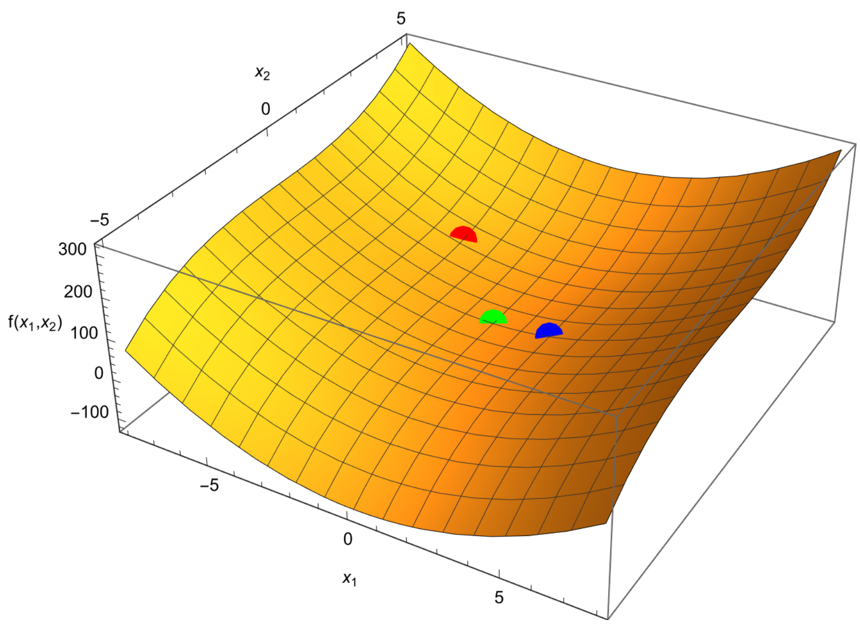
Out[ ]//MatrixForm=

$$\begin{pmatrix} 12 \\ 0 \end{pmatrix}$$

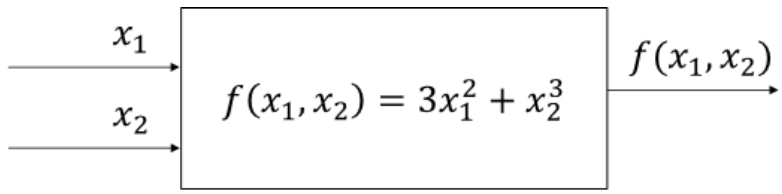
In[ ]:=

```
PA = ( x1A  x2A  f[x1A, x2A] );
ptA = Point[PA];

PB = ( x1B  x2B  f[x1B, x2B] );
ptB = Point[PB];

PC = ( x1C  x2C  f[x1C, x2C] );
ptC = Point[PC];

p1 = Show[
  Plot3D[f[x1, x2], {x1, -8, 8}, {x2, -5, 5},
   AxesLabel → {"x1", "x2", "f(x1,x2)"}],
  Graphics3D[{AbsolutePointSize[15], Red, ptA}],
  Graphics3D[{AbsolutePointSize[15], Green, ptB}],
  Graphics3D[{AbsolutePointSize[15], Blue, ptC}]
 ]
```

Out[ ]=



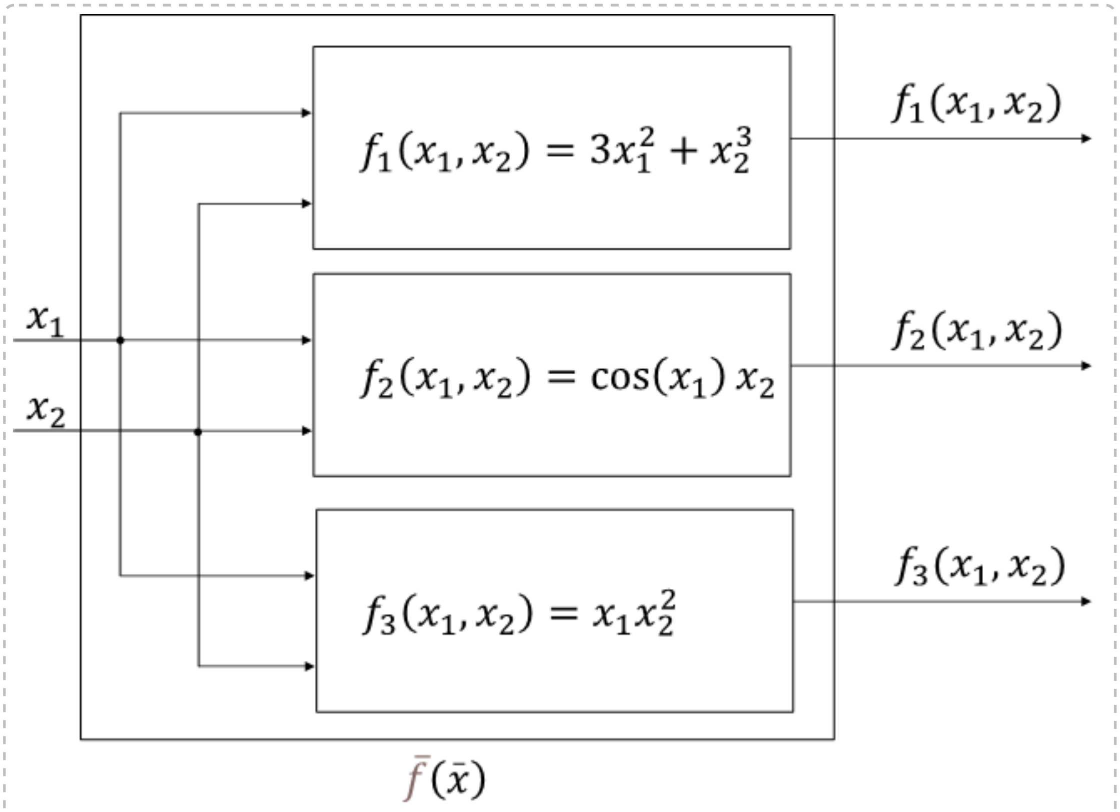
Again, we can visualize this as



$$\nabla f(\bar{x}) = \begin{pmatrix} \frac{\partial f(\bar{x})}{\partial x_1} \\ \frac{\partial f(\bar{x})}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 6x_1 \\ 3\,x_2^2 \end{pmatrix} \begin{array}{l} \longleftarrow \text{Sensitivity of } f \text{ to changes in } x_1 \\ \longleftarrow \text{Sensitivity of } f \text{ to changes in } x_2 \end{array}$$

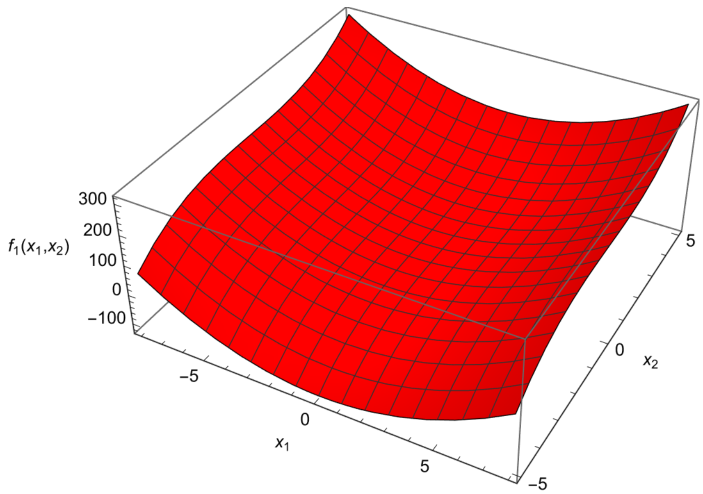
The Jacobian Matrix

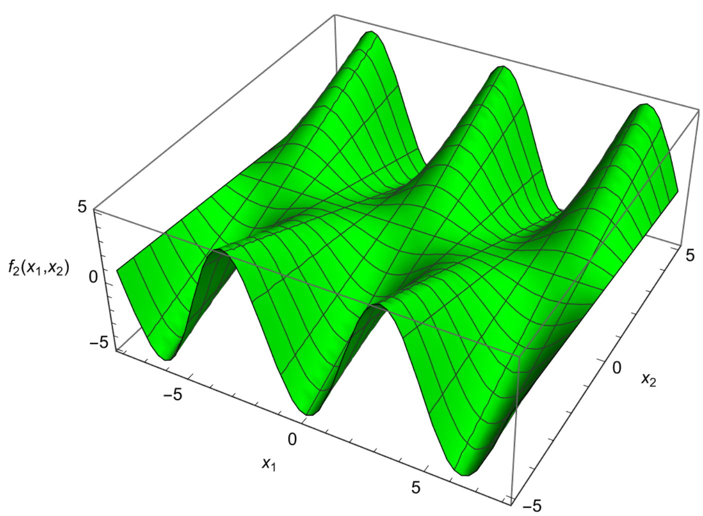
The Jacobian Matrix is simply an extension of the gradient of a scalar function to a vector valued valued function. Recall that a vector valued function can simply be thought of as several scalar valued functions stacked on top of one another.

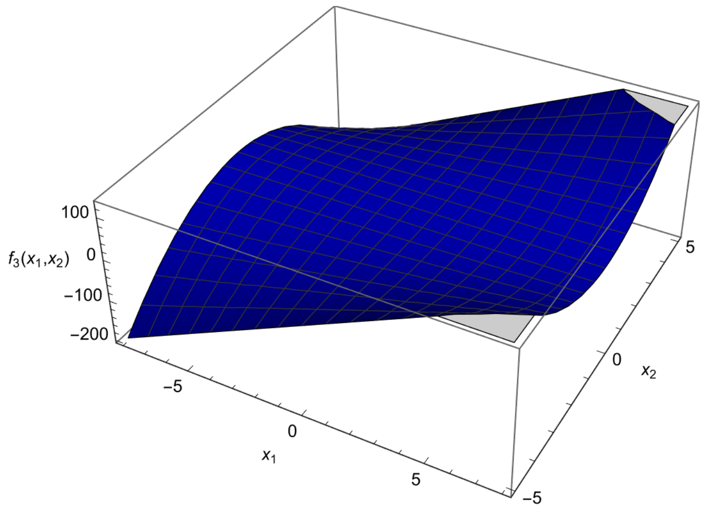


In[ ]:=
```
f1[x1_, x2_] = 3 x1^2 + x2^3;
f2[x1_, x2_] = Cos[x1] x2;
f3[x1_, x2_] = x1 x2^2;
```

In[ ]:=
```
x1Min = -8;
x1Max = 8;
x2Min = -5;
x2Max = 5;
Plot3D[f1[x1, x2], {x1, x1Min, x1Max}, {x2, x2Min, x2Max},
 AxesLabel → {"x1", "x2", "f1(x1,x2)"}, PlotStyle → Red]

Plot3D[f2[x1, x2], {x1, x1Min, x1Max}, {x2, x2Min, x2Max},
 AxesLabel → {"x2", "x2", "f2(x1,x2)"}, PlotStyle → Green]

Plot3D[f3[x1, x2], {x1, x1Min, x1Max}, {x2, x2Min, x2Max},
 AxesLabel → {"x1", "x2", "f3(x1,x2)"}, PlotStyle → Blue]
```

Out[ ]=


Out[ ]=


Out[ ]=


We can write

$$\bar{f}(\bar{x}) = \begin{pmatrix} f_1(\bar{x}) \\ f_2(\bar{x}) \\ \ldots \\ f_m(\bar{x}) \end{pmatrix}$$

The Jacobian is simply a matrix of the gradient vectors. By convention, we typically transpose the gradient of each function so the k^{th} row of the Jacobian matrix is the gradient of the k^{th} scalar function

$$J(\bar{x}) = \frac{\partial \bar{f}(\bar{x})}{\partial \bar{x}} = \begin{pmatrix} \frac{\partial f_1(\bar{x})}{\partial x_1} & \frac{\partial f_1(\bar{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\bar{x})}{\partial x_n} \\ \frac{\partial f_2(\bar{x})}{\partial x_1} & \frac{\partial f_2(\bar{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\bar{x})}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_m(\bar{x})}{\partial x_1} & \frac{\partial f_m(\bar{x})}{\partial x_2} & \cdots & \frac{\partial f_m(\bar{x})}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \nabla f_1(\bar{x})^T \\ \nabla f_2(\bar{x})^T \\ \vdots \\ \nabla f_m(\bar{x})^T \end{pmatrix} \qquad (m \times n \text{ matrix})$$

Each entry in the matrix gives

$J_{\text{ij}}(\bar{x})$ = sensitivity of $\bar{f}(\bar{x})$ output i to changes/perturbations in input j (AKA x_j)

We see that the Jacobian matrix completely characterizes the sensitivity of all the function’s outputs in response to perturbations in all the function’s inputs.

History

The Jacobian Matrix is named after the mathematician Carl Gustav Jacob Jacobi (1804-1851).

-German mathematician who made fundamental contributions to elliptic functions, dynamics, differential equations, determinants, and number theory.
-Hamilton-Jacobi equation: alternate equations of motion formulation for classical mechanics.
-Jacobi crater on the near side of the moon is named after him (68 km diameter, 3.3 km deep)

Carl Gustav Jacob Jacobi
(1804 – 1851)

The Jacobi Crater
on the Moon

Examples

Mathematical Example

Consider the function $\bar{f}(\bar{x})$ defined earlier

$$\bar{f}(\bar{x}) = \begin{pmatrix} f_1(x_1, x_2) \\ f_3\,(x_1, x_2) \\ f_2\,(x_1, x_2) \end{pmatrix}$$

$$J(\bar{x}) = \frac{\partial \bar{f}(\bar{x})}{\partial \bar{x}} = \begin{pmatrix} \frac{\partial f_1(\bar{x})}{\partial x_1} & \frac{\partial f_1(\bar{x})}{\partial x_2} \\ \frac{\partial f_2(\bar{x})}{\partial x_1} & \frac{\partial f_2(\bar{x})}{\partial x_2} \\ \frac{\partial f_3(\bar{x})}{\partial x_1} & \frac{\partial f_3(\bar{x})}{\partial x_2} \end{pmatrix} = \begin{pmatrix} \nabla f_1(\bar{x})^T \\ \nabla f_2(\bar{x})^T \\ \nabla f_3(\bar{x})^T \end{pmatrix}$$

In[∘]:=
```
f1[x1_, x2_] = 3 x1^2 + x2^3;
f2[x1_, x2_] = Cos[x1] x2;
f3[x1_, x2_] = x1 x2^2;
```

In[∘]:=
```
J[x1_, x2_] = ( D[f1[x1, x2], x1]  D[f1[x1, x2], x2]
                D[f2[x1, x2], x1]  D[f2[x1, x2], x2]
                D[f3[x1, x2], x1]  D[f3[x1, x2], x2] );

J[x1, x2] // MatrixForm
```

Out[∘]//MatrixForm=

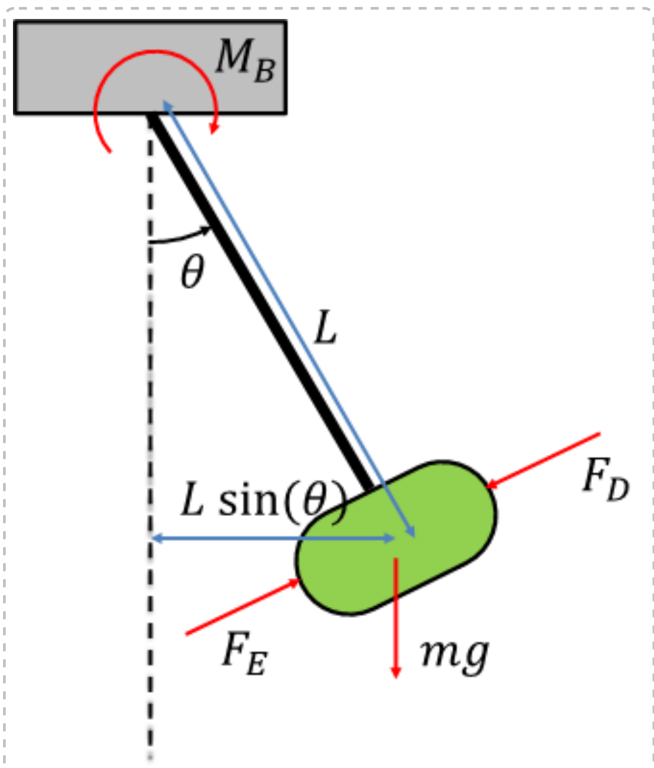
$$\begin{pmatrix} 6\text{ x1} & 3\text{ x2}^2 \\ -\text{x2 Sin[x1]} & \text{Cos[x1]} \\ \text{x2}^2 & 2\text{ x1 x2} \end{pmatrix}$$

$$\mathrm{J}(\bar{x}) = \frac{\partial \bar{f}(\bar{x})}{\partial \bar{x}} = \begin{pmatrix} \nabla f_1(\bar{x})^T \\ \nabla f_2(\bar{x})^T \\ \nabla f_3(\bar{x})^T \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1(\bar{x})}{\partial x_1} & \frac{\partial f_1(\bar{x})}{\partial x_2} \\ \frac{\partial f_2(\bar{x})}{\partial x_1} & \frac{\partial f_2(\bar{x})}{\partial x_2} \\ \frac{\partial f_3(\bar{x})}{\partial x_1} & \frac{\partial f_3(\bar{x})}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 6\,x_1 & 3\,x_2^2 \\ -x_2\sin(x_1) & \cos(x_1) \\ x_2^2 & 2\,x_1 x_2 \end{pmatrix}$$

Sensitivity of f_1 to changes in x_1
Sensitivity of f_1 to changes in x_2
Sensitivity of f_2 to changes in x_1
Sensitivity of f_2 to changes in x_2
Sensitivity of f_3 to changes in x_1
Sensitivity of f_3 to changes in x_2

Nonlinear to Linear ODE

Consider a rocket engine on the end of a test stand shown below. This is effectively a standard pendulum with a somewhat novel system for imparting moments on the system.

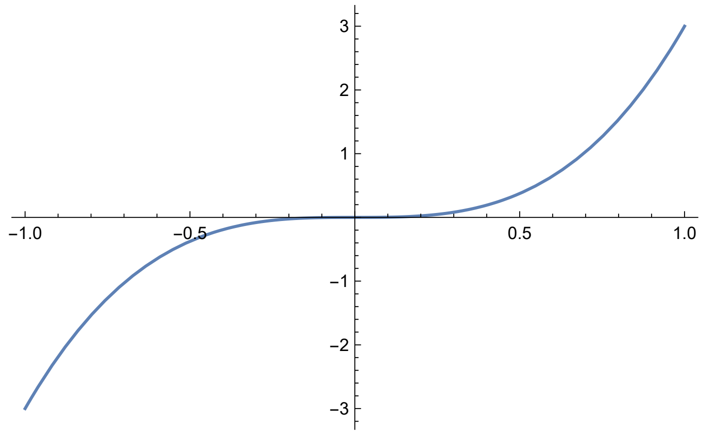


Suppose the force from the engine is related to the throttle as through

$$F_E = \alpha\, u_1^{\,3}$$

In[◦]:=
```
Plot[α u1^3 /. {α → 3}, {u1, -1, 1}]
```

Out[◦]=



Suppose the drag force is modeled as

$$F_D = \beta V = \beta L \dot{\theta}$$

Suppose the braking moment is given as

$$M_B = \gamma u_2 \dot{\theta} \qquad u_2 \in [0, 1] \qquad \text{(braking moment is related to speed of rotation)}$$

We can list the moments as

$$\sum M = F_E L - F_D L - M_B - m g L \sin(\theta)$$

$$= \alpha u_1^3 L - \beta L \dot{\theta} L - \gamma u_2 \dot{\theta} - m g L \sin(\theta)$$

$$\sum M = \alpha L u_1^3 - \beta L^2 \dot{\theta} - \gamma u_2 \dot{\theta} - m g L \sin(\theta)$$

Consider the nonlinear dynamic equations of motion of the form

$$I_z \ddot{\theta} = \sum M$$

If we consider all the mass to be centered in the rocket, then $I_z = m L^2$

$$m L^2 \ddot{\theta} = \alpha L u_1^3 - \beta L^2 \dot{\theta} - \gamma u_2 \dot{\theta} - m g L \sin(\theta)$$

$$\ddot{\theta} = \frac{\alpha L}{m L^2} u_1^3 - \frac{\beta L^2}{m L^2} \dot{\theta} - \frac{\gamma u_2}{m L^2} \dot{\theta} - \frac{m g L}{m L^2} \sin(\theta)$$

$$\ddot{\theta} = \frac{\alpha}{m L} u_1^3 - \frac{\beta}{m} \dot{\theta} - \frac{\gamma}{m L^2} u_2 \dot{\theta} - \frac{g}{L} \sin(\theta)$$

We can write a state space representation (see ‘State Space Representation of Differential Equations’ at https://youtu.be/pXvAh1IOO4U) using the following state vector and control vector

$$\overline{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} \qquad \overline{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

So we have

$$\dot{\overline{x}} = \begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \end{pmatrix}$$

$$= \begin{pmatrix} \dot{\theta} \\ \frac{\alpha}{m L} u_1{}^3 - \frac{\beta}{m} \dot{\theta} - \frac{\gamma}{m L^2} u_2 \dot{\theta} - \frac{g}{L} \sin(\theta) \end{pmatrix}$$

$$= \begin{pmatrix} x_2 \\ \frac{\alpha}{m L} u_1{}^3 - \frac{\beta}{m} x_2 - \frac{\gamma}{m L^2} u_2 x_2 - \frac{g}{L} \sin(x_1) \end{pmatrix}$$

If we consider the states and controls to be independent variables, we can write

$$\overline{z} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

So we have

$$\dot{z}_1 = z_2$$
$$\dot{z}_2 = \frac{\alpha}{m L} z_3{}^3 - \frac{\beta}{m} z_2 - \frac{\gamma}{m L^2} z_4 z_2 - \frac{g}{L} \sin(z_1)$$

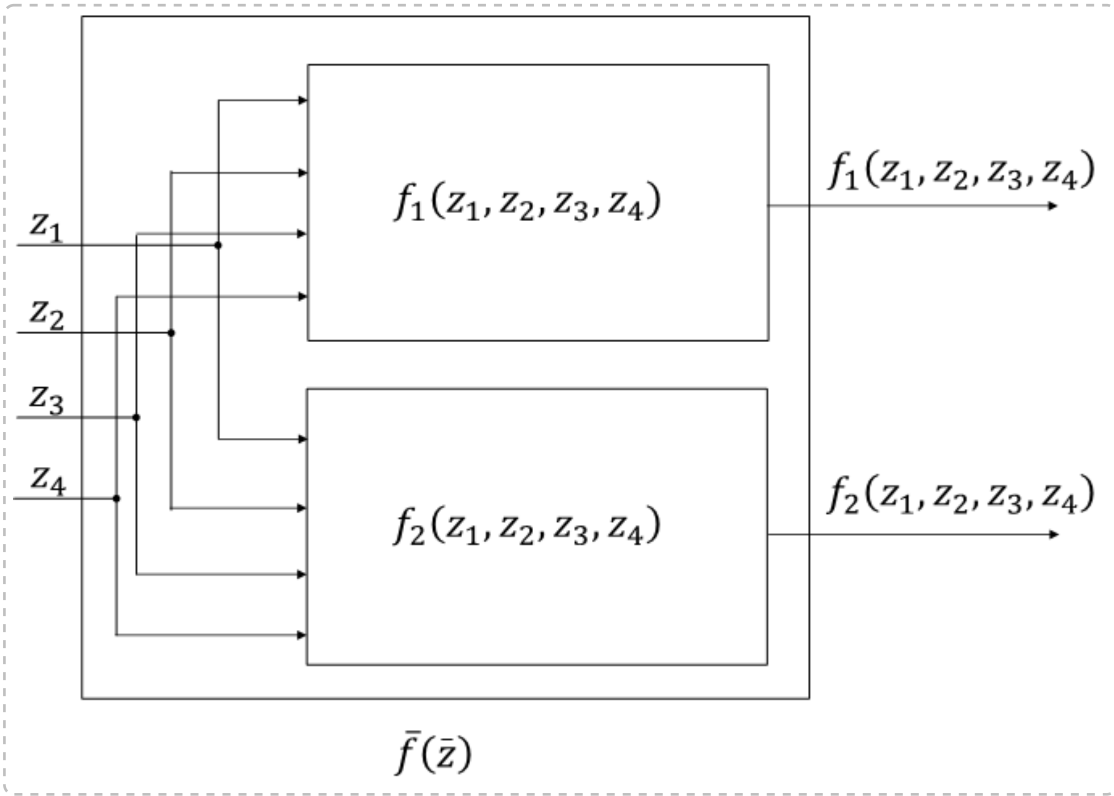
$$\dot{z}_1 = f_1(\overline{z})$$
$$\dot{z}_2 = f_2(\overline{z})$$

$$\begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \overline{f}(\overline{z})$$

where $\overline{f}(\overline{z}) = \begin{pmatrix} f_1(\overline{z}) \\ f_2(\overline{z}) \end{pmatrix} = \begin{pmatrix} z_2 \\ \frac{\alpha}{m L} z_3{}^3 - \frac{\beta}{m} z_2 - \frac{\gamma}{m L^2} z_4 z_2 - \frac{g}{L} \sin(z_1) \end{pmatrix}$

```
In[ ]:= f1[z1_, z2_, z3_, z4_] = z2;
f2[z1_, z2_, z3_, z4_] = α/(m L) z3^3 - β/m z2 - γ/(m L^2) z4 z2 - g/L Sin[z1];
```

We can visualize this as shown below



We can compute the Jacobian of the function $\overline{f}(\overline{z})$

In[∘]:=
```
J[z1_, z2_, z3_, z4_] =
  (D[f1[z1, z2, z3, z4], z1]  D[f1[z1, z2, z3, z4], z2]  D[f1[z1, z2, z3, z4], z3]  D[f1[z1, z2,
   D[f2[z1, z2, z3, z4], z1]  D[f2[z1, z2, z3, z4], z2]  D[f2[z1, z2, z3, z4], z3]  D[f2[z1, z2,
  );

J[z1, z2, z3, z4] // MatrixForm
```

Out[∘]//MatrixForm=

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{g\,\mathrm{Cos}[z1]}{L} & -\frac{\beta}{m} - \frac{z4\,\gamma}{L^2\,m} & \frac{3\,z3^2\,\alpha}{L\,m} & -\frac{z2\,\gamma}{L^2\,m} \end{pmatrix}$$

In order to assess the sensitivity of the function at the point $\overline{z}_o$ we have can write

$$\Delta\,\overline{f}(\overline{z}_o) = J(\overline{z}_o)\,\Delta\,\overline{z}$$

$$= \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{g}{L}\cos(z_{1,o}) & \frac{-\beta}{m} - \frac{\gamma}{m\,L^2}\,z_{4,o} & \frac{3\,\alpha}{m\,L}\,z_{3.\,o}{}^2 & -\frac{\gamma}{m\,L^2}\,z_{2,o} \end{pmatrix} \begin{pmatrix} \Delta\,z_1 \\ \Delta\,z_2 \\ \Delta\,z_3 \\ \Delta\,z_4 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{g}{L}\cos(z_{1,o}) & \frac{-\beta}{m} - \frac{\gamma}{m L^2} z_{4,o} & \frac{3\alpha}{m L} z_{3.o}{}^2 & -\frac{\gamma}{m L^2} z_{2,o} \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta u_1 \\ \Delta u_2 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 1 \\ -\frac{g}{L}\cos(z_{1,o}) & \frac{-\beta}{m} - \frac{\gamma}{m L^2} z_{4,o} \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \frac{3\alpha}{m L} z_{3.o}{}^2 & -\frac{\gamma}{m L^2} z_{2,o} \end{pmatrix} \begin{pmatrix} \Delta u_1 \\ \Delta u_2 \end{pmatrix}$$

$$= A\,\Delta\,\overline{x} + B\,\Delta\,\overline{u}$$

Note that this is the foundation of how one linearizes a dynamic system. We need to be somewhat careful with equilibrium points and perform a formal Taylor series expansion (see ‘The Taylor Series’ at https://youtu.be/kbV9LdQXVtg) but these are details best left for another video.

Next Steps

The Jacobian matrix will play a key role in the discussion of

-The Chain Rule
-Linearizing a Dynamic System
-Backpropagation (Neural Networks)