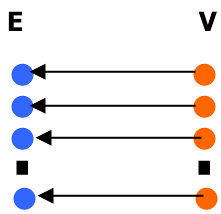
Case 1: ODE

$$0 = \dot{x} - f(t, x)$$



All variables $x(t)$ are of index one.

Case 2: DAE of index 1

$$0 = \dot{x} - f(t, x, y)$$
$$0 = g(t, x, y)$$

Assumptions: $\frac{\partial g}{\partial y}$ is invertible around the solution (x^*, y^*)
Conclusions: all variables (x, y) are of index one.

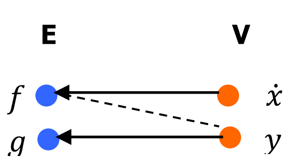
Proof:

$$0 = g(t, x, y) \Rightarrow 0 = g(t, x^*, y^*) + \frac{\partial g}{\partial y}\dot{y} + \frac{\partial g}{\partial x}\dot{x} + \frac{\partial g}{\partial t} = \frac{\partial g}{\partial y}\dot{y} + \frac{\partial g}{\partial x}\dot{x}$$
$$\Rightarrow \dot{y} = \left(\frac{\partial g}{\partial y}\right)^{-1}\frac{\partial g}{\partial x}\dot{x}$$

So, the original problem is equivalent (near the solution) to an ODE:

$$\dot{x} = f(t, x, y)$$
$$\dot{y} = \left(\frac{\partial g}{\partial y}\right)^{-1}\frac{\partial g}{\partial x} f(t, x, y)$$

Since the above ODE is obtained without differentiating any of the $f(t, x, y)$ equations then we have the same index as that of case 1.



Case 3: DAE of Index 2

$$
\begin{aligned}
0 &= \dot{x} - f(t,x,y) \\
0 &= g(t,x)
\end{aligned}
$$

Assumptions: $\frac{\partial g}{\partial y}$ is NOT invertible around the solution (x^*, y^*) and $\frac{\partial g}{\partial x}.f(t,x^*,y^*) = 0$.

Conclusions: all variables x are of index one; all variables y are of index two.

Proof:

$$
\begin{aligned}
0 &= \ddot{x} - \frac{\partial f}{\partial x}\dot{x} - \frac{\partial f}{\partial y}\dot{y} \\
0 &= \frac{\partial g}{\partial x}\dot{x} = \frac{\partial g}{\partial x} f(t,x,y)
\end{aligned}
$$

Pre-multiply the first equation by $\frac{\partial g}{\partial x}$ to get

$$0 = \frac{\partial g}{\partial x}\ddot{x} - \frac{\partial g}{\partial x}\frac{\partial f}{\partial x} f(t,x,y) - \frac{\partial g}{\partial x}\frac{\partial f}{\partial y}\dot{y}$$

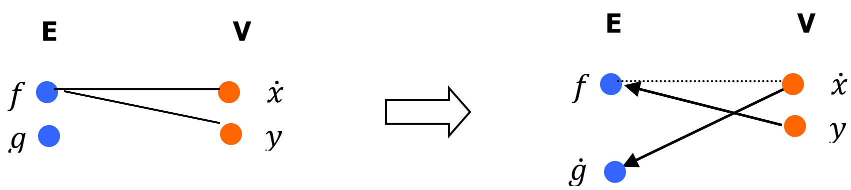
From the above assumption

$$0 = \frac{\partial g}{\partial x}\ddot{x} - \frac{\partial g}{\partial x}\frac{\partial f}{\partial y}\dot{y}$$

Hence,

$$0 = \dot{y} - \left(\frac{\partial g}{\partial x}\frac{\partial f}{\partial y}\right)^{-1}\frac{\partial g}{\partial x}\ddot{x}$$

Algorithm: The same logic can be obtained from MSS:



Therefore, variable x is of index one.
The index of variable y is of index two:

Index(y) = Index(x) + [# of Diff(x) - # of Diff(y)] = 1 + [1 – 0] = 2.

Case 4: DAE of Index 3

$$
\begin{aligned}
0 &= \dot{x} - f(t,x,z) \\
0 &= \dot{z} - h(t,x,z,y) \\
0 &= g(t,x)
\end{aligned}
$$

[Example is the Pendulum]

Assumptions: $\frac{\partial g}{\partial y}$ is NOT invertible around the solution (x^*, y^*), $\frac{\partial g}{\partial x}.f(t,x^*,y^*) = 0$; and $\frac{\partial^2 g}{\partial x^2} f(t,x,z) \bullet f(t,x,z) + \frac{\partial g}{\partial x}\frac{\partial f}{\partial x} f(t,x,z) + \frac{\partial f}{\partial z} h(t,x,z) = 0$

Conclusions: all variables x are of index one, variables z are of index two; all variables y are of index three.

Algorithm:

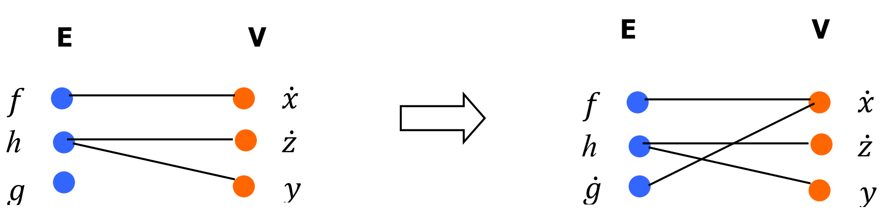
Therefore, variable x is of index one.

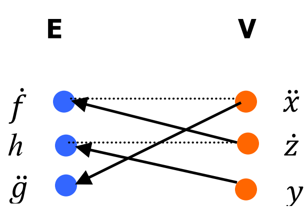
The index of variable z is of index two. Using Equation $\dot{f}$:

Index(z) = Index(x) + [# of Diff(x) - # of Diff(y)] = 1 + [2 – 1] = 2.

The index of variable y is of index three. Using Equation h:

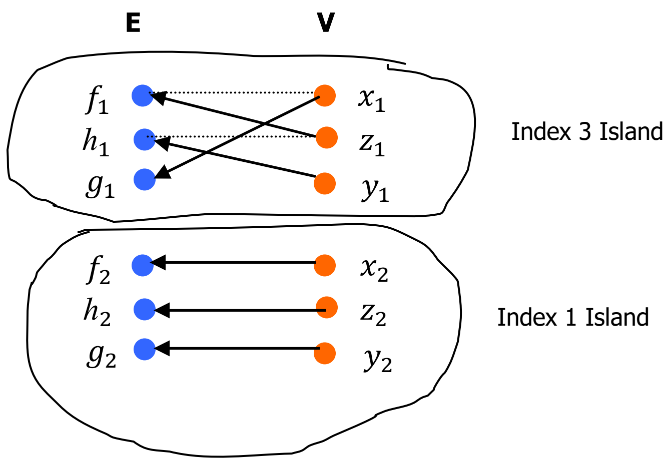
Index(y) = Index(z) + [# of Diff(z) - # of Diff(y)] = 2 + [1 – 0] = 3.





General Algorithm:

1. Using the final MSS bipartite graph run Tarjan to isolate its islands.



2. For each island identify highest-derivative variables (call them island leaders). Set the index of the leaders to 1.

3. Select a non-leader variable V and identify its assigned equation E.

4. From the graph, identify variables, other than V, that occur in the E, call them the *incidental* variables of E.

5. Find (recursively calculate) the index of each incidental variable in E.

6. Identify the incidental variable, V*, in E that has the highest index among all incidental variables in E.

7. Find the differentiation orders of both V and incidental variable V* within E (those orders are also the number of differentiation of V and V* in the final MSS graph).

8. Calculate the index of V as follows:

   Index(V) = Index(V*) + [Differentiation Order (V*) - Differentiation Order (V)]

9. Go to next island.

The index calculations should be used to also update the tolerances of high index variables and scale up the convergence factor for *Growing* flag within the solver.

Algorithm implementation:

Step 1: Mapping from final MSS VAL
NOTES: Position in VAL is the variable number
Value in VAL references more derived variable number

a. Count the most derived variables from VAL (-1 entries) → nMostDerived.
b. Build a map iVarRoot [nMostDerived] of most derived variables into indices of VAL.
c. iDiffOrder[nMostDerived] is the number of times each root variable has been differentiated. Calculated by traversing VAL.

Step 2: Building island graph for BLT from final MSS graph
a. For each most derived variable (rows in graph), find the assigned equation E.
b. Find the other variables referenced by E (incidental variables of E), filling columns in island graph for those in the most derived list.

Step 3: Run BLT algorithm on the island graph

Step 4: Find the island leaders in each island (block)
a. Initialize Boolean map islandLeaders[nMostDerived] to false
b. Initialize diffIndex[nMostDerived] to -1
c. Find the most derived variable (from iDiffOrder) in each island. Set islandLeaders[iVar] = true, and diffIndex[iVar] = 1

Step 5: Calculate index for each variable
a. Loop through variables in each island. If diffIndex[iVar] has not be set (-1), calculate differential index as follows:
    a. Loop through other incidental variables in the island graph. If their index is unknown, calculate their index first (recursion).
    b. Determine the highest differential index of the other incidental vars in the island graph, and note which variable contributed this index.
    c. iDiffIndex[iVar] = iDiffIndex[iMaxDiffVar] + iDiffOrder[iMaxDiffVar] - iDiffOrder[iVar]
b. Write out the results diffIndex[nMostDerived] to differentialIndex[nAugmentedVars] using iVarRoot map.

??? The differential index is only being written out for the most derived variables, whereby, leaving the differential index of the less derived variables at 1. Is this correct? ie: if xdd is index 2, should xd and x be index 2?

Equation calculation:

Consider the following index 3 example (pendulum)
g, L are fixed parameters

f0 : 0 = -xd + w
f1 : 0 = yd + z
f2 : 0 = wd + t*x
f3 : 0 = zd + t*y - g
f4 : 0 = x^2 + y^2 - L^2

From MSS:
f5 : df4/dt = df4/dx * xd + df4/dy * yd
f6 : df0/dt = df0/dw * wd + df0/dxd * xdd
f7 : df1/dt = df1/dz * zd + df1/dyd * ydd
f8 : df5/dt = d2f4/dt = df5/dx * xd + df5/dy * yd + df5/dxd * xdd + df5/dyd * ydd

VAL - how variables and their derivatives are associated

```
        0, 1, 2, 3,  4,  5,  6,  7, 8,   9,  10
  VAL { x, y, w, z, xd, yd, wd, zd, t, xdd, ydd }
  ==> { 4, 5, 6, 7,  9, 10, -1, -1,-1,  -1,  -1 }
```

EAL - how equations and their derivatives are associated

```
           0,  1,  2,  3,  4,   5,   6,   7,   8
   eqns { f0, f1, f2, f3, f4, f4d, f1d, f2d, f5d }  (f5d == f4dd)
        {   6, 7, -1, -1,  5, 8,    -1,   -1, -1 }
```

Derivatives of functions are calculated as follows:
dfi/dt -> look in edge list for variable dependencies on equation i

chain rule: dfi/dt = sum(dfi/dj * dj/dt) for each equation to variable edge

get dfi/dj from jacobian if i < in_nEqns && j < in_nVars (i is not an augmented equation, and j is not an augmented variable),
otherwise calculate numerically by perturbing j

get dfi/d2j from mass if i < in_nEqns && j < in_nVars,
otherwise calculate numerically by perturbing dj

get dj/dt by getting derivative of j from VAL

EdgeList - how variables map to equations

	{ x,	y,	w,	z,	xd,	yd,	wd,	zd,	t,	x2,	y2 }
eqns	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10
0			1		1						
1				1		1					
2	1						1		1		
3		1						1	1		
4	1	1									
5	1	1			1	1					
6			1		1		1			1	
7				1		1		1			1
8	1	1			1	1				1	1

Equation calculations are therefore as follows:

f5 : df4/dt = jac(df4/dx) * xd + jac(df4/dy) * yd
f6 : df0/dt = jac(df0/dw) * wd + mass(df0/dxd) * xdd
f7 : df1/dt = jac(df1/dz) * zd + mass(df1/dyd) * ydd
f8 : df5/dt = d2f4/dt = num(df5/dx) * xd + num(df5/dy) * yd + num(df5/dxd) * xdd + num(df5/dyd) * ydd

f5 : 2x * xd + 2y * yd
f6 : 1 * wd + -1 * xdd
f7 : 1 * zd + 1 * ydd
f8: 2 * xd + 2 * yd + 0 * xdd + 0 * ydd

NOTES:

Equations f6, f7, and f8 rely on the values of xdd and ydd, which are MSS added variables. At present, these variables default to a value of 0, whereby, contributing nothing to the equations. It should be possible for the user to define initial values for xdd and ydd. We don’t have this facility at present because the user is not made aware of the additional equations or variables added by MSS.