Algorithm: Matching

dbro@kth.se

```
MATCH(G)
1  assign ← ∅
2  for each f ∈ G.F
3      do C ← ∅
4          if not MATCH-EQUATION(G, f, C, assign, ∅)
5              then return (FALSE, assign)
6  return (TRUE, assign)
```

Color visited vertices

$$C \subseteq G.F \cup G.V$$

Assigns variables to equations

$$assign[v] = \begin{cases} f & \text{if } f \text{ matches } v \\ \text{NIL} & \text{otherwise} \end{cases}$$

```
MATCH-EQUATION(G, f, C, assign, vmap)
 1  C ← C ∪ {f}
 2  if there exits a v ∈ G.V such that (f, v) ∈ G.E
 3      and assign[v] = NIL and vmap[v] = NIL
 4      then assign[v] ← f
 5           return TRUE
 6      else for each v where (f, v) ∈ G.E and v ∉ C
 7           and vmap[v] = NIL
 8               do C ← C ∪ {v}
 9                  if MATCH-EQUATION(G, assign[v], C, assign, vmap)
10                      then assign[v] ← f
11                           return TRUE
12  return FALSE
```

Algorithm: BLT Sort

dbro@kth.se

Input: a bipartite graph G

```
BLT(G)
 1  (match, assign) ← MATCH(G)
 2  if not match
 3     then  return error "Singular"
 4
 5  D.V ← G.F
 6  D.E ← ∅
 7  for each (f, v) ∈ G.E where f ∈ G.F and assign[v] ≠ f
 8      do D.E ← D.E ∪ {(assign[v], f)}
 9
10  MAKEEMPTY(O)
11  MAKEEMPTY(S)
12  i ← 0
13  lowlink ← ∅
14  number ← ∅
15  for each v ∈ D.V
16      do if number[v] = NIL
17            then STRONGCONNECT(v, D, S, i, lowlink, number, O)
18  return O
```

Output: a stack of sets of equation vertices, where each set represents an equation block in the BLT matrix.

Algorithm: StrongConnect (Tarjan)

dbro@kth.se

```
StrongConnect(v, D, S, i, lowlink, number, O)
 1  i ← i + 1
 2  lowlink[v] ← i
 3  number[v] ← i
 4  PUSH(S, v)
 5  for each w ∈ D.V where (v, w) ∈ D.E
 6      do if number[w] = NIL
 7            then StrongConnect(w, D, S, i, lowlink, number, O)
 8                 lowlink[v] ← MIN(lowlink[v], lowlink[w])
 9            else  if w ∈ S and number[w] < number[v]
10                    then lowlink[v] ← MIN(lowlink[v], number[w])
11  if lowlink[v] = number[v]
12     then eqset ← ∅
13          while not ISEMPTY(S) and number[TOP(S)] ≥ number[v]
14              do eqset ← eqset ∪ {POP(S)}
15          PUSH(O, eqset)
16  return
```

Algorithm: Pantelides

dbro@kth.se

```
PANTELIDES(G, vmap, eqmap)
 1  assign ← ∅
 2  for each e ∈ G.F
 3      do f ← e
 4         repeat
 5                C ← ∅
 6                match ← MATCH-EQUATION(G, f, C, assign, vmap)
 7                if not match
 8                   then for each v ∈ C where v ∈ G.V
 9                            do let v′ be a vertex, such that v′ ∉ G.V
10                               vmap[v] ← v′
11                               G.V ← G.V ∪ {v′}
12                         for each f ∈ C where f ∈ G.F
13                            do let f′ be a vertex, such that f′ ∉ G.F
14                               eqmap[f] ← f′
15                               G.F ← G.F ∪ {f′}
16                               for each v ∈ G.V where (f, v) ∈ G.E
17                                  do G.E ← G.E ∪ {(f′, v), (f′, vmap[v])}
18                         for each v ∈ C where v ∈ G.V
19                            do assign[vmap[v]] ← eqmap[assign[v]]
20                         f ← eqmap[f]
21            until match
22  return assign
```

Mapping variables to differentiated variables

$$vmap[v] = \begin{cases} v' & \text{if } \frac{dv}{dt} = v' \\ \text{NIL} & \text{otherwise} \end{cases}$$

Mapping equations to their differentiated version

$$eqmap[f] = \begin{cases} f' & \text{if } \frac{df}{dt} = f' \\ \text{NIL} & \text{otherwise} \end{cases}$$

Assigns variables to equations

$$assign[v] = \begin{cases} f & \text{if } f \text{ matches } v \\ \text{NIL} & \text{otherwise} \end{cases}$$