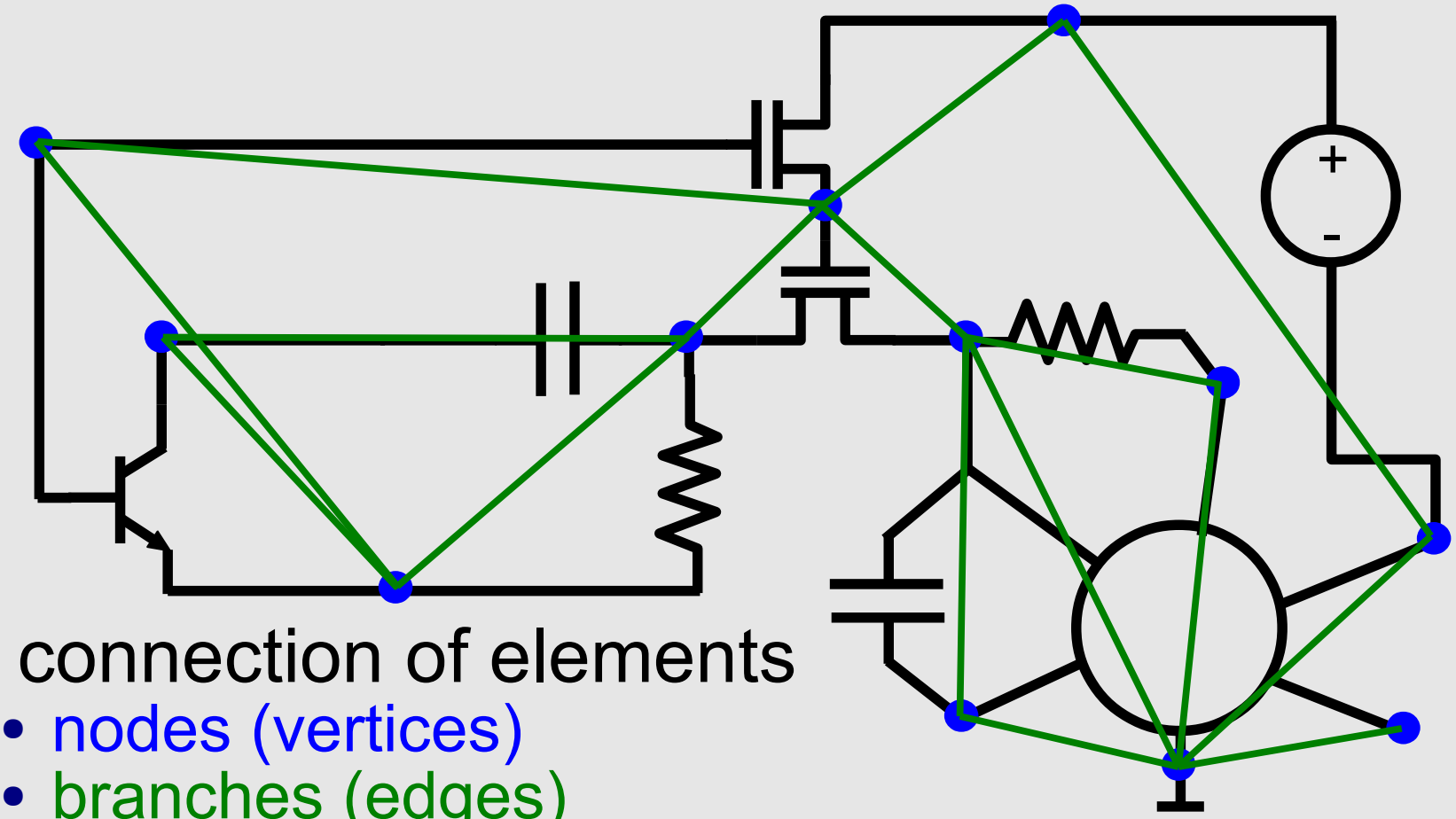
Fundamental Continuous-Time Algorithms

- Timing analysis: Boolean/delay abstractions
  - one level lower: continuous-time systems
- Continuous-time algorithms: topics
  - (a little modelling)
    - ckt, mechanical, biochemical examples
  - The Newton-Raphson algorithm
  - (glimpse of sparse matrix algorithms)
  - algorithms for ODE solution
  - frequency-domain concepts and algorithms for linear systems
    - linearizing nonlinear systems
  - freq.-domain algorithms for stochastic quantities
    - continuous-time randomness and stochastic concepts

Circuit Modelling (Equation Systems for Circuits)

- Why
  - need equations to solve!
- What
  - systems of differential equations
- How
  - put together (in various ways)
    - Branch Constitutive Relationships (BCRs):
      - “device models”: i.e., equations for all the elements
    - Kirchoff's Current Law (KCL):
      - branch currents sum to zero at every node
    - Kirchoff's Voltage Law (KVL):
      - branch voltages sum to zero around every loop

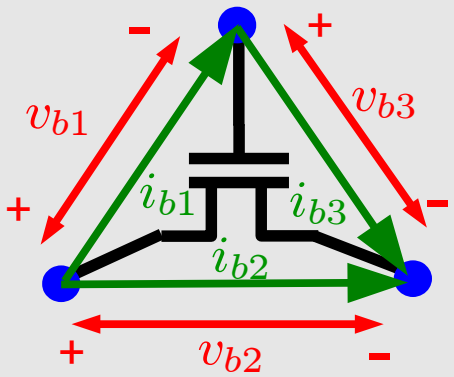
What is a Circuit?



- connection of elements
  - nodes (vertices)
  - branches (edges)
- element connections usually not arbitrary!
  - designed to achieve some desired function
    - eg: amplication, logic, oscillation

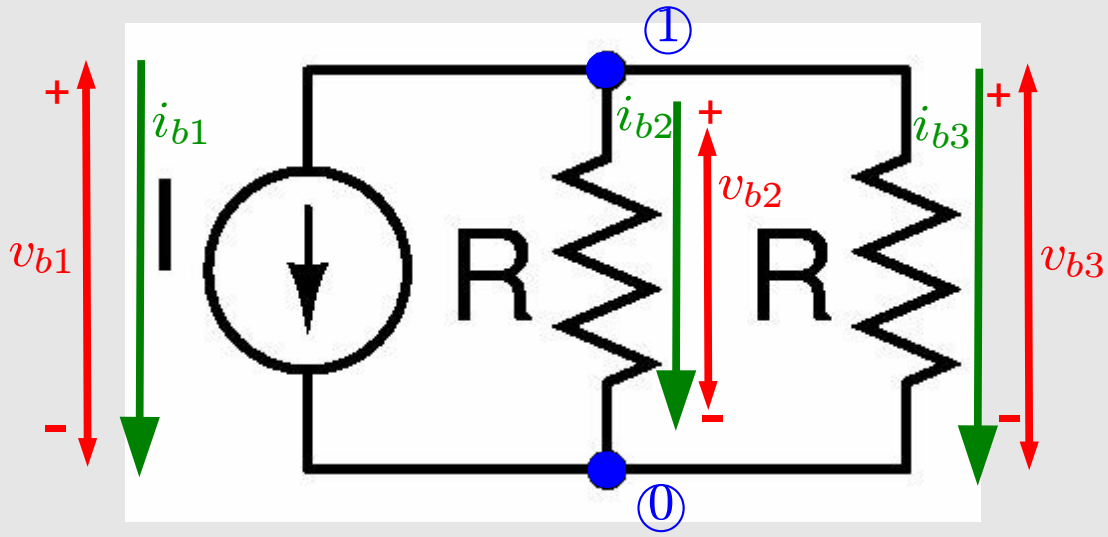
Writing Circuit Equations: the Flow

- Identify element branch variables (unknowns)
  - branch voltages, branch currents
  - (ensure v/i directions consistent)
- Write KCL equation at each node
  - KCL = Kirchoff's Current Law
    - *sum(all branch currents at node) = 0*
  - KCL relates all branch currents



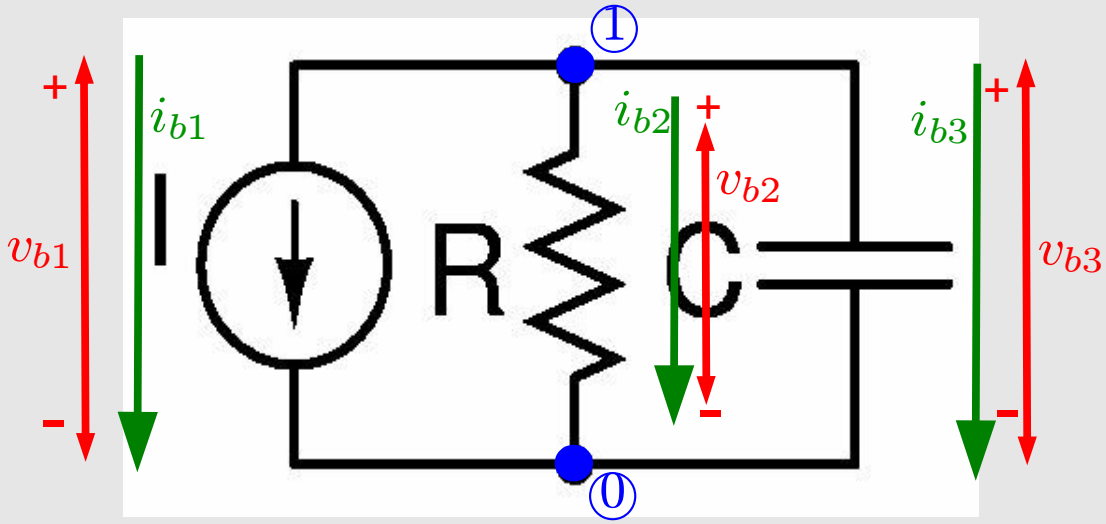
- Write KVL equations
  - KVL = Kirchoff's Voltage Law
    - *sum(branch voltages around any loop) = 0* (not used directly)
    - equivalent to saying: "node voltages" exist
  - KVL relates all branch voltages with "node voltages"
- Write BCR equations
  - i.e., the "device model" equations
  - BCRs relate branch currents and branch voltages
- Put it all together to get circuit equation system

Simple Memoryless Example



- Node 0: reference node ("ground")
  - (don't write KCL at this node)
- 1 KCL eqn. at node 1: $i_{b1} + i_{b2} + i_{b3} = 0$
- 3 KVL equations: $v_{b1} = v_{b2} = v_{b3} = e_1$
- 3 BCRs: $i_{b1} = I; \quad i_{b2} = \frac{v_{b2}}{R}; \quad i_{b3} = \frac{v_{b3}}{R}$
- 7 unknowns, 7 equations: "square" system
  - **Sparse Tableau Equations**
- solve by hand (eliminate): $I + 2\frac{e_1}{R} = 0 \Rightarrow e_1 = -\frac{IR}{2}$

Simple RC Example



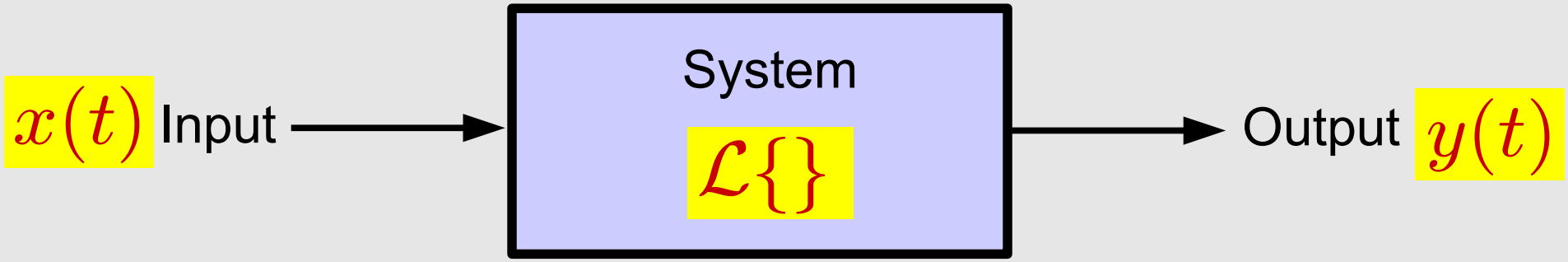
- 1 KCL eqn. at node 1: $i_{b1} + i_{b2} + i_{b3} = 0$
- 3 KVL equations: $v_{b1} = v_{b2} = v_{b3} = e_1$
- 3 BCRs: $i_{b1} = I(t); \quad i_{b2} = \frac{v_{b2}}{R}; \quad i_{b3} = C\frac{dv_{b3}}{dt}$
- 7 unknowns, 7 equations (incl. 1 differential eqn.)
  - (Sparse Tableau) **Differential-Algebraic Equations** (DAEs)
- eliminate by hand: obtain ODEs
  - **Ordinary Differential Equations**
- analytical solution available

$$I + \frac{e_1}{R} + C\frac{de_1}{dt} = 0$$

$$\Rightarrow \boxed{\frac{de_1}{dt} = -\frac{e_1}{RC} - \frac{I(t)}{C}}$$

What Does Linear Mean?

- System concept
  - important to identify input(s) and output(s)



- I/O relationship: $y(t) = \mathcal{L}\{x(t)\}$
  - $\mathcal{L}\{\cdot\}$: operator
    - ➔ uses the <u>whole function of time</u> $x(t)$ to determine $y(t)$

Linearity

- A system is linear iff:

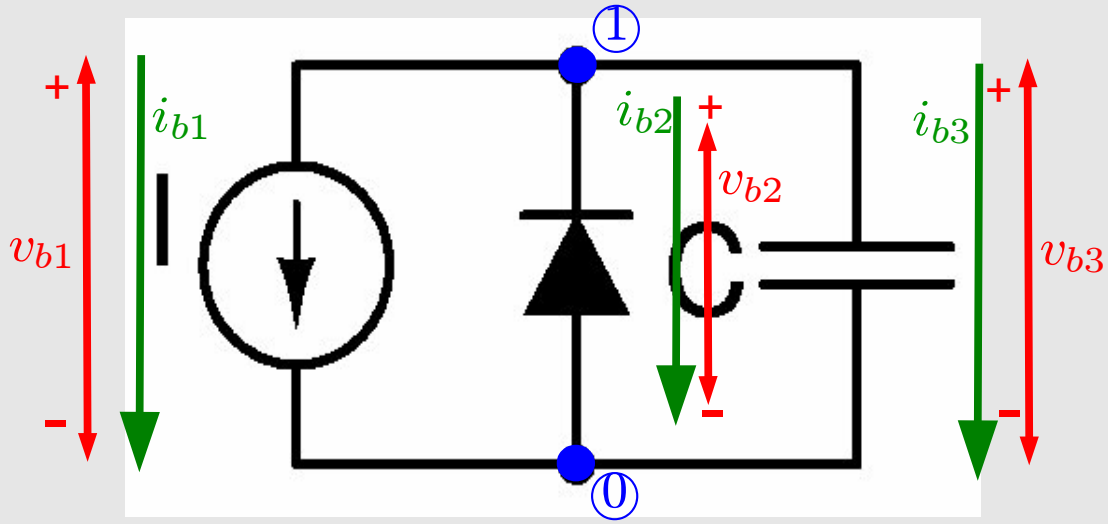
1) scaling the input by *any* constant α scales the output by α

$$\mathcal{L}\{\alpha x(t)\} = \alpha \mathcal{L}\{x(t)\}$$

3) adding *any* two inputs to each other results in adding the corresponding individual outputs

$$\mathcal{L}\{x_1(t) + x_2(t)\} = \mathcal{L}\{x_1(t)\} + \mathcal{L}\{x_2(t)\}$$

Diode-Capacitor (Nonlinear) Example



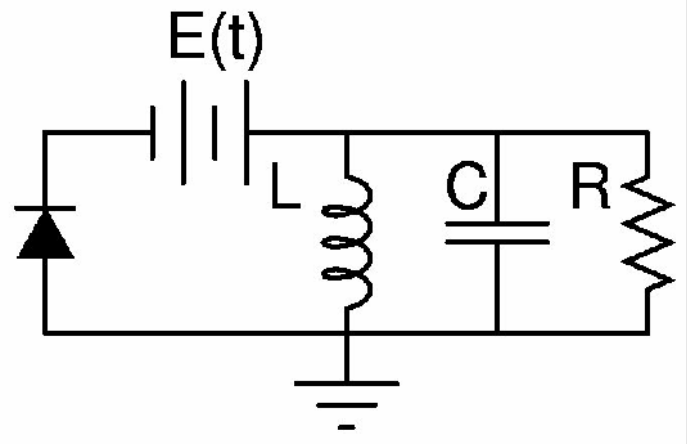
- 1 KCL eqn. at node 1: $i_{b1} + i_{b2} + i_{b3} = 0$
- 3 KVL equations: $v_{b1} = v_{b2} = v_{b3} = e_1$
- 3 BCRs: $i_{b1} = I(t); \quad i_{b2} = -\text{diode}(-v_{b2}; I_S, V_t); \quad i_{b3} = C\dfrac{dv_{b3}}{dt}$
- 7 unks, 7 eqns (incl. 1 differential, 1 nonlinear)
- eliminate by hand: ODEs
  - **(ODEs not always possible)**
  - **analytical soln not available!**
    - **numerical methods needed**

$$I(t) - \text{diode}(-e_1; I_S, V_t) + C\frac{de_1}{dt} = 0$$

$$\Rightarrow \boxed{\frac{de_1}{dt} = \frac{1}{C}\text{diode}(-e_1; I_S, V_t) - \frac{I(t)}{C}}$$

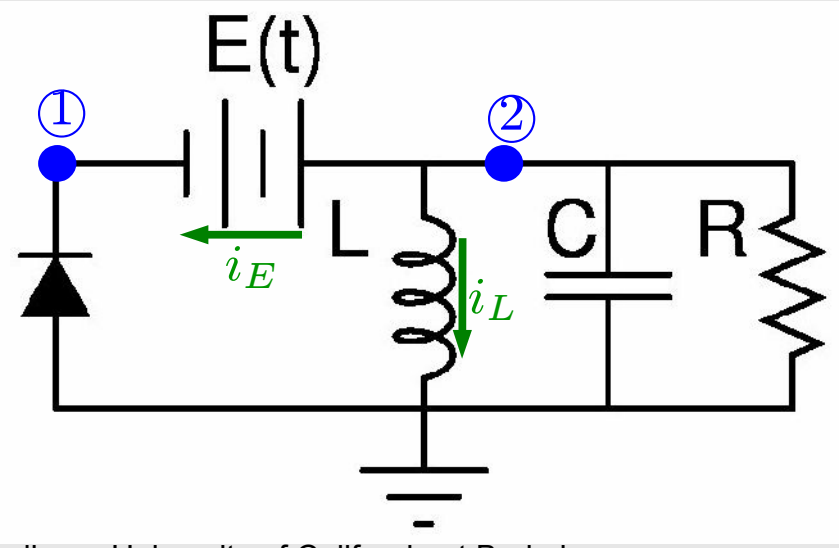
Nodal Analysis

- The idea: eliminate all branch variables
  - retain only node variables as unknowns
- Use KVL to eliminate branch voltages
  - replace branch voltages by node voltages in BCRs
- Use BCRs to eliminate branch currents
  - replace branch currents by BCR equations in KCL
- Gain over tableau: smaller equation system
  - #eqns/#unknowns = #nodes
- The problem: <u>BCRs must all be voltage-controlled</u>
  - voltage sources, inductors are not!
    - cannot eliminate some branch currents



Modified Nodal Analysis (MNA)

- Fix for nodal analysis problem
  - Use KVL to eliminate all branch voltages
    - replace branch voltages by node voltages in BCRs
  - Eliminate branch currents of all voltage-controlled BCRs
    - (hopefully, most BCRs voltage controlled)
  - Remaining BCRs: retain controlling branch currents
- Eliminates all branch voltages, most branch currents
  - smaller systems than tableau, handles all elements
  - is “lazy”: could eliminate more unknowns (with more effort)
    - e.g.: use BCR for E(t) to eliminate one of the node voltages



$$\mathrm{KCL}_1: \quad -\mathrm{diode}(-e_1; I_S, V_t) - i_E = 0$$

$$\mathrm{KCL}_2: \quad i_E + i_L + \frac{e_2}{R} + C\frac{de_2}{dt} = 0$$

$$\mathrm{BCR}_E: \quad e_2 - e_1 - E(t) = 0$$

$$\mathrm{BCR}_L: \quad e_2 - L\frac{di_L}{dt} = 0$$

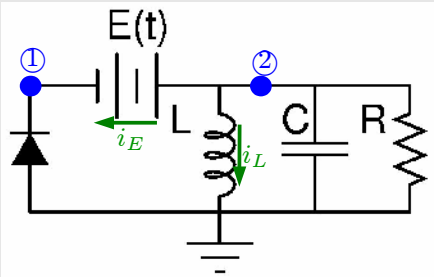
Standard DAE Form

- Tableau/NA/MNA equations in **standard DAE form**

$$\frac{d}{dt}\vec{q}(\vec{x}(t)) + \vec{f}(\vec{x}(t)) + \vec{b}(t) = \vec{0}$$

  - (vector) Differential-Algebraic Equation system
  - x: unknowns; q(): charges/fluxes; f(): resistive; b(t): inputs

- For our MNA example:



$$\begin{aligned}
\text{KCL}_1: &\quad -\text{diode}(-e_1; I_S, V_t) - i_E = 0 \\
\text{KCL}_2: &\quad i_E + i_L + \frac{e_2}{R} + C\frac{de_2}{dt} = 0 \\
\text{BCR}_E: &\quad e_2 - e_1 - E(t) = 0 \\
\text{BCR}_L: &\quad e_2 - L\frac{di_L}{dt} = 0
\end{aligned}$$

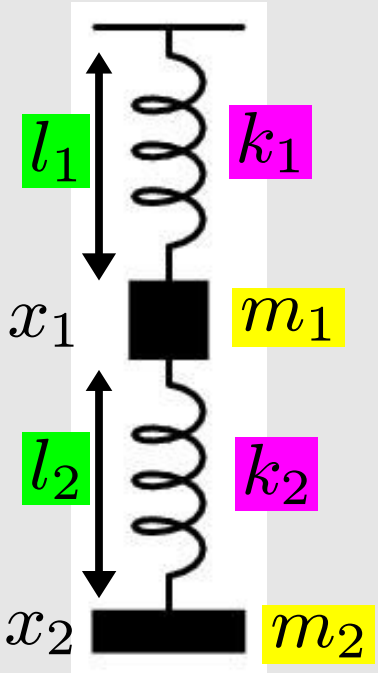
$$\vec{x}(t) = \begin{bmatrix} e_1(t) \\ e_2(t) \\ i_L(t) \\ i_E(t) \end{bmatrix} \quad \vec{q}(\vec{x}) = \begin{bmatrix} 0 \\ Ce_2 \\ 0 \\ -Li_L \end{bmatrix} \quad \vec{f}(\vec{x}) = \begin{bmatrix} -\text{diode}(-e_1; I_S, V_t) - i_E \\ i_E + i_L + \frac{e_2}{R} \\ e_2 - e_1 \\ e_2 \end{bmatrix} \quad \vec{b}(t) = \begin{bmatrix} 0 \\ 0 \\ -E(t) \\ 0 \end{bmatrix}$$

Mechanical Spring-Mass system

- position: measured down from the ceiling
- forces: down = positive
- Hooke's Law for the springs:
  - $F_1 = k_1(x_1 - l_1)$
  - $F_2 = k_2(x_2 - x_1 - l_2)$
- Newton's Law for the masses:
  - $m_1\ddot{x}_1 = -F_1 + F_2$
  - $m_2\ddot{x}_2 = -F_2$
- First-order differential equations

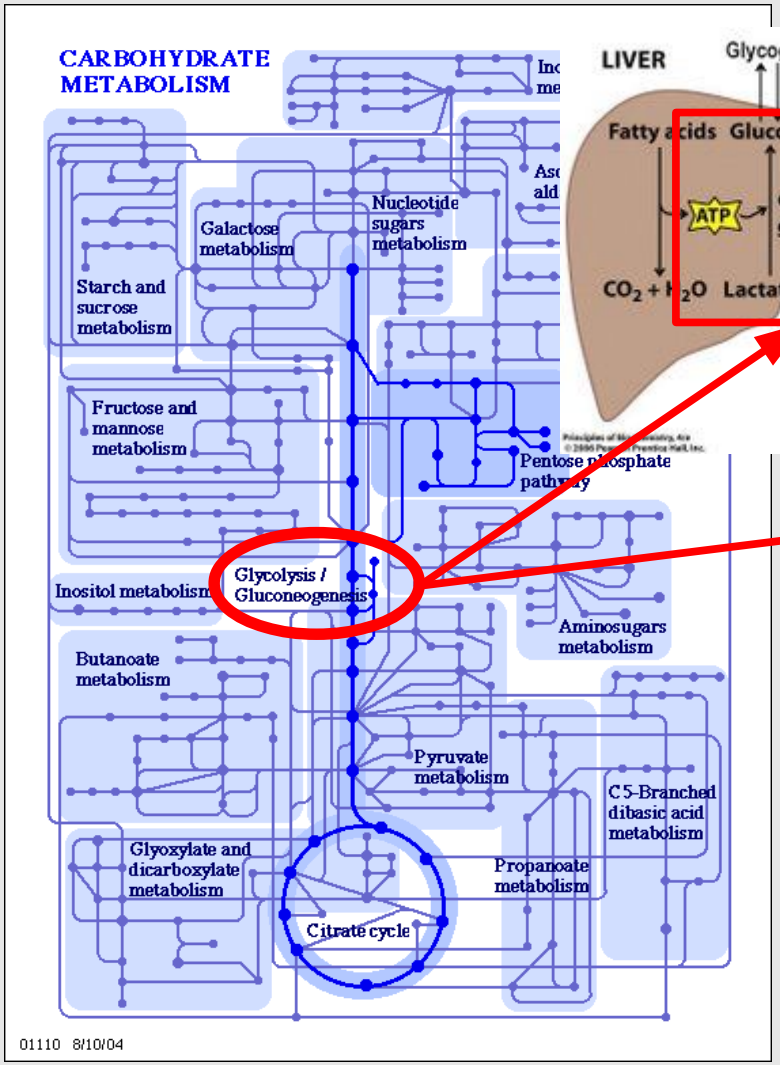
$$m_1\dot{y}_1 = -k_1(x_1 - l_1) + k_2(x_2 - x_1 - l_2)$$
$$m_2\dot{y}_2 = -k_2(x_2 - x_1 - l_2)$$
$$\dot{x}_1 = y_1$$
$$\dot{x}_2 = y_2$$

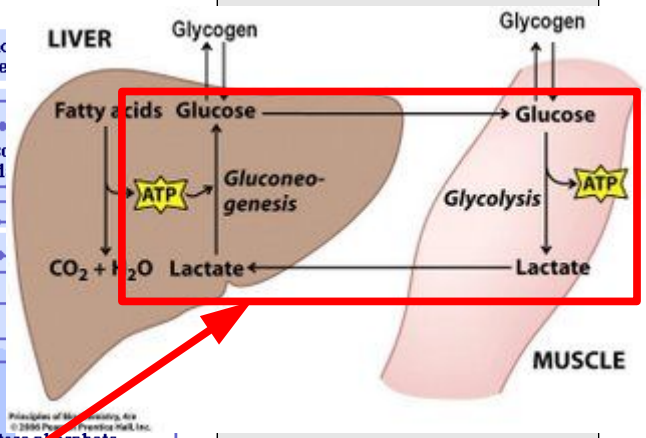
- Canonical form? Linearity?

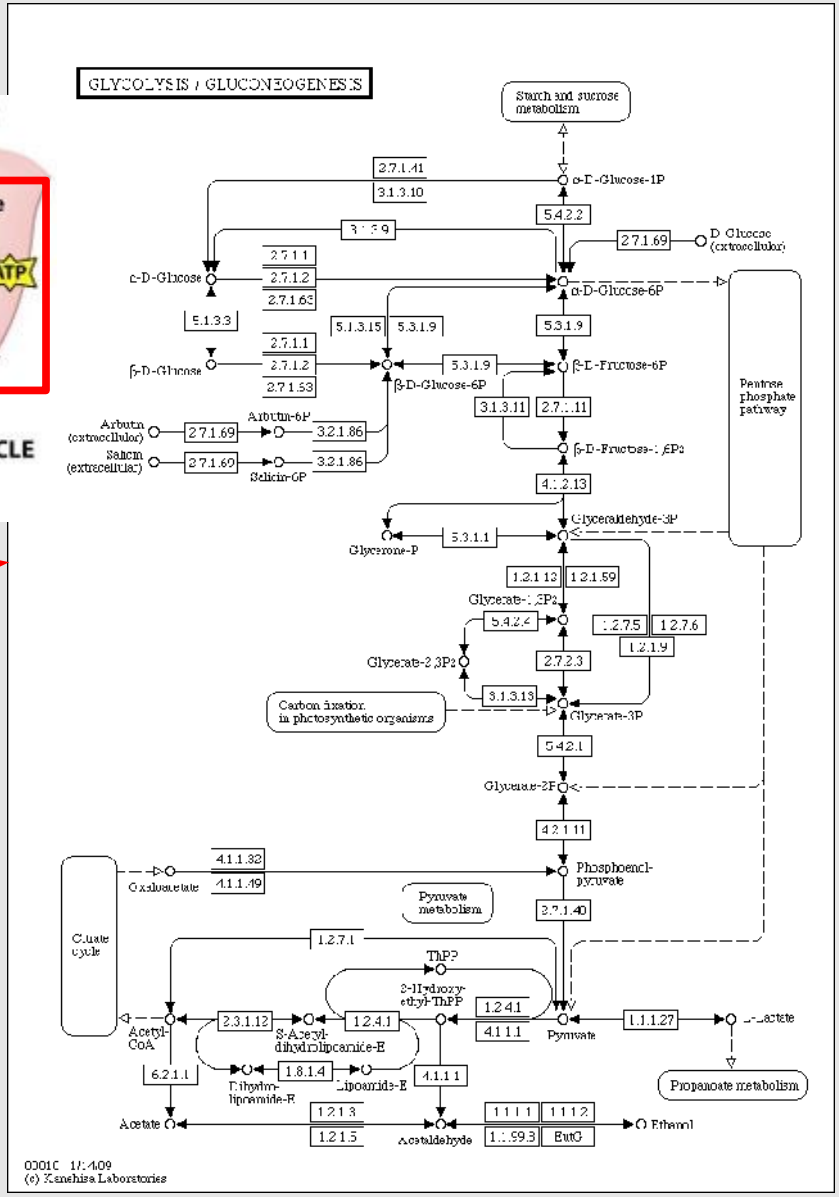


Biochemical Systems

- Reactions governing conversion of glucose to ATP (energy)

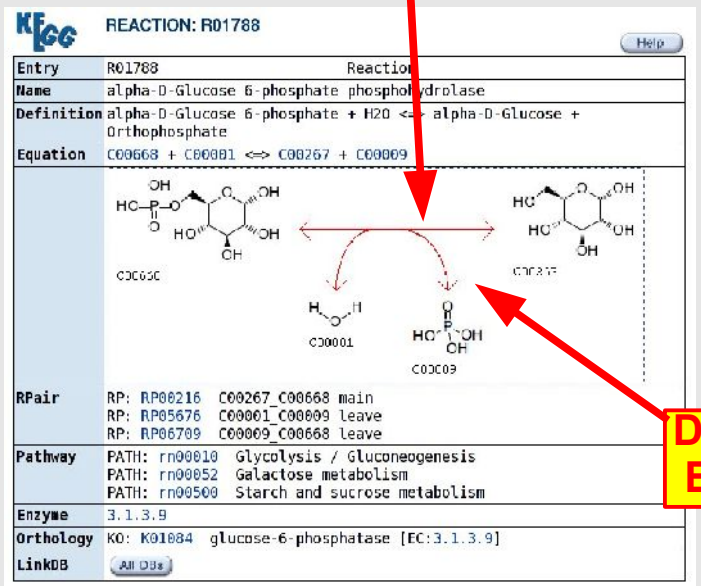


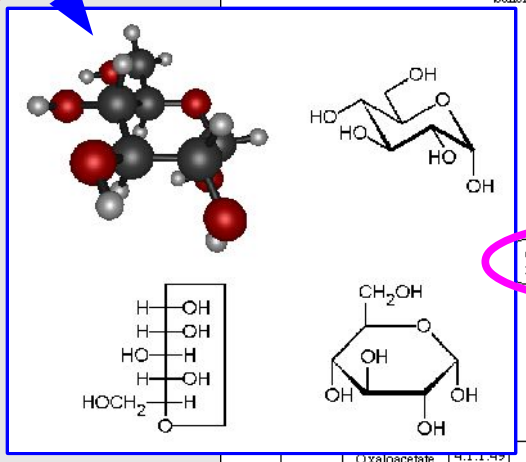


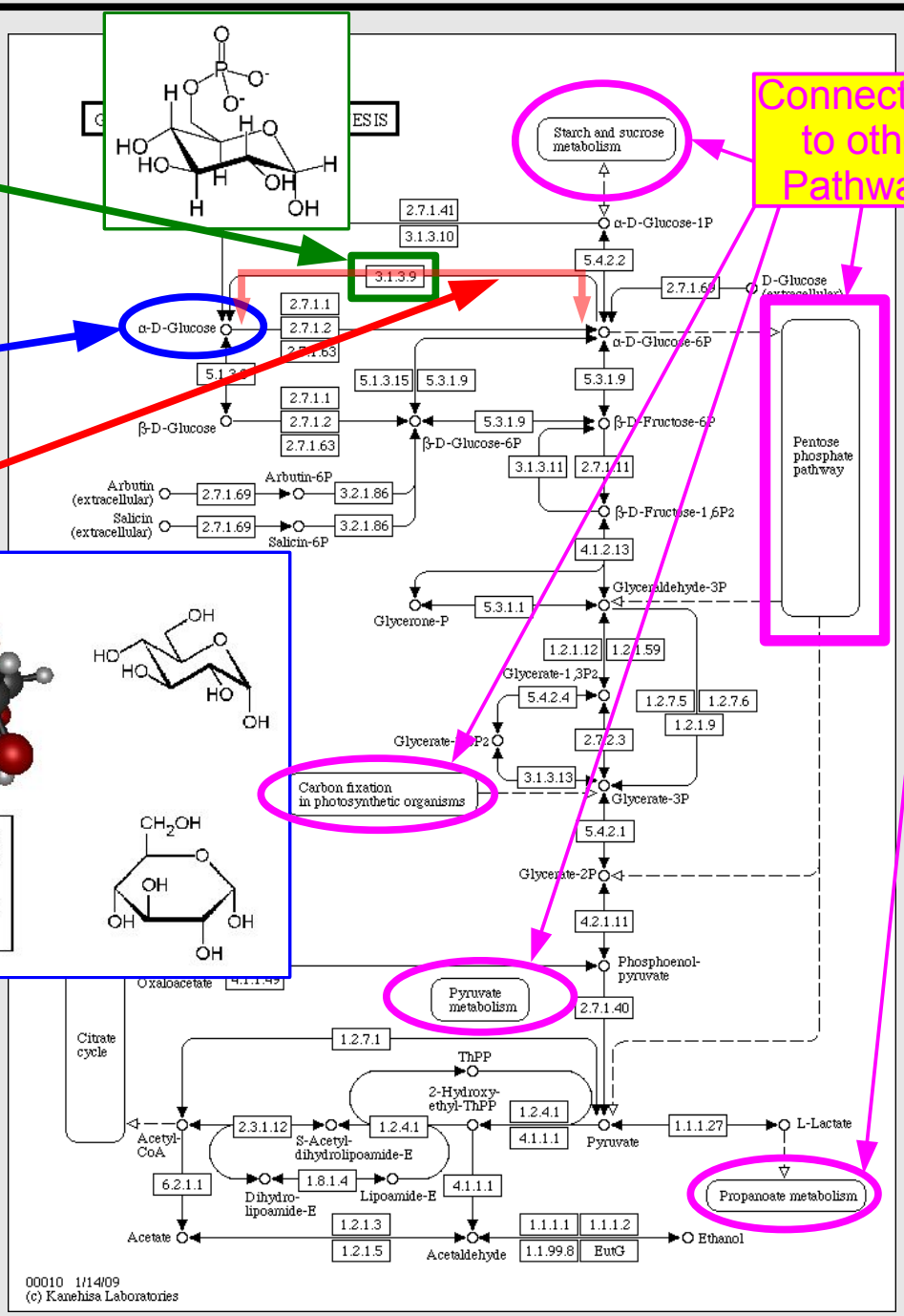


Pathway: Chain of Reactions

- **Enzyme** (protein)
  - glucose-6-phosphatase
    - catalyzes reaction
- **Compound**
  - alpha-D-Glucose
- **Reaction**
  - alpha-D-glucose 6-phosphate phosphohydrolase

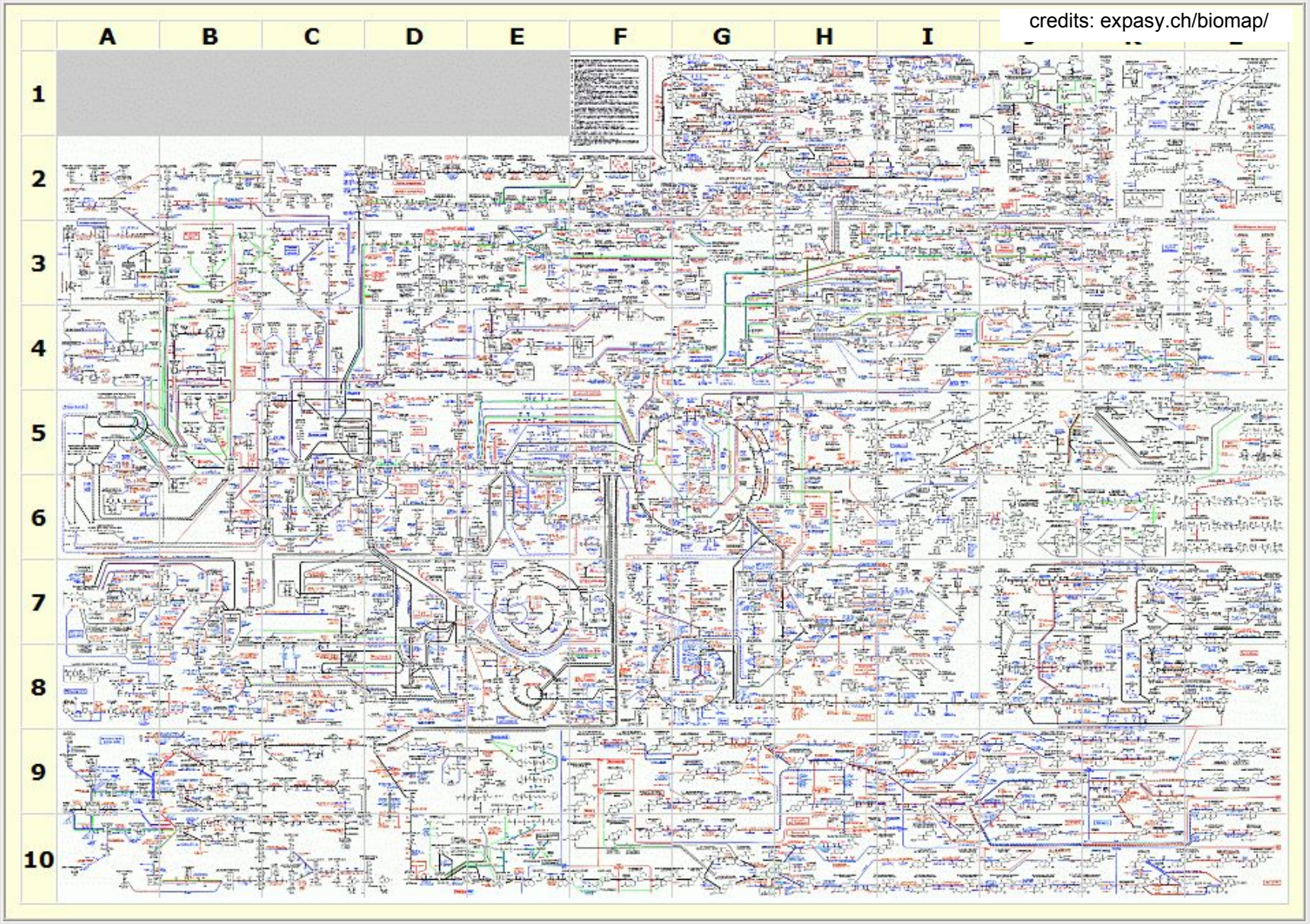






Differential Equations

Metabolic Pathway Maps



Rate Equations for Simple Reactions

$$A \underset{k_2}{\overset{k_1}{\rightleftharpoons}} B$$

- Molecule/reactant/"species" A changing to B
  - more precisely: 1 molecule of A converts to 1 molecule of B
    - and vice-versa: reversible reaction
  - k_1, k_2: forward and reverse ***rate constants***
- **Law of mass action**
  - **rate of concentration change is proportional to concentration**
    - **rate constants: the proportionality constants**

$$\frac{d[A]}{dt} = -k_1[A] + k_2[B]$$

$$\frac{d[B]}{dt} = -k_2[B] + k_1[A]$$

concentration of B

differential equation system

concentration of A

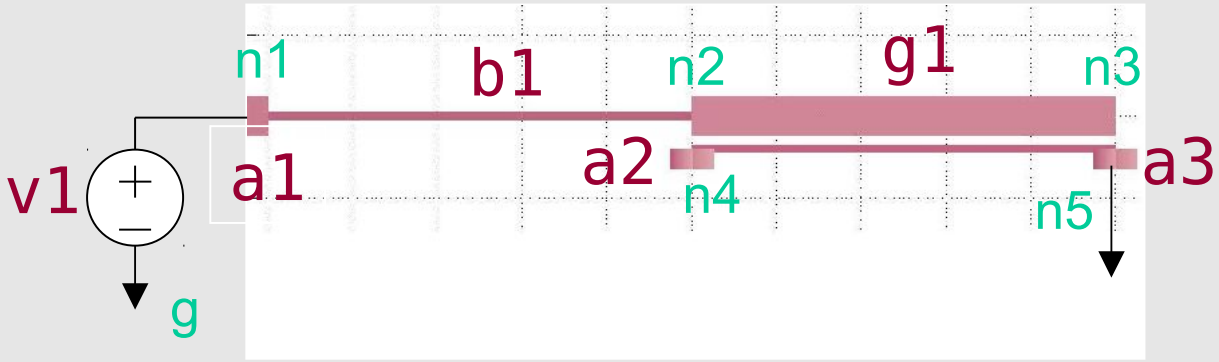
Circuit Netlists and Parsing

- Circuits typically described in a simple (plain text) format: "**netlist**"

```
VCC 7 0 12
VEE 8 0 -12
VIN 1 0 AC 1
RS1 1 2 1K
RS2 6 0 1K
Q1 3 2 4 MOD1
Q2 5 6 4 MOD1
RC1 7 3 10K
RC2 7 5 10K
RE 4 8 10K
.MODEL MOD1 NPN BF=50 VAF=50 IS=1.E-12 RB=100 CJC=.5PF TF=.6NS
.END
```

- Parser: reads netlists, sets up data structures
  - info about nodes, connectivities, elements/models, parameters, …
- Equation Engine: provides q(), f(), b(t) functions
  - using info in data structures, device model equations

Mechanical Netlist (SUGAR)




```
uses mumps.net
v1 Vsrc     *  [n1 g]   [V=10]
e1 eground  *  [g]      []
a1 anchor   p1 [n1]     [l=5e-6 w=10e-6 oz=180 R=100]
b1 beam2de  p1 [n1 n2]  [l=1e-4 w=2e-6 oz=0 R=1000]
g1 gap2de   p1 [n2 n3 n4 n5] [l=1e-4 w1=1e-5 w2=2e-6 …
                    gap=2e-6 R1=100 R2=100 oz=0]
a2 anchor   p1 [n4]     [l=5e-6 w=1e-5 oz=-90 R=100]
e2 eground  *  [n4]     []
a3 anchor   p1 [n5]     [l=5e-6 w=1e-5 oz=-90 R=100]
e3 eground  *  [n5]     []
```

Systems Biology Markup Language (SBML)

$$E + S \underset{k_{\mathrm{off}}}{\overset{k_{\mathrm{on}}}{\rightleftharpoons}} ES \xrightarrow{k_{\mathrm{cat}}} E + P$$

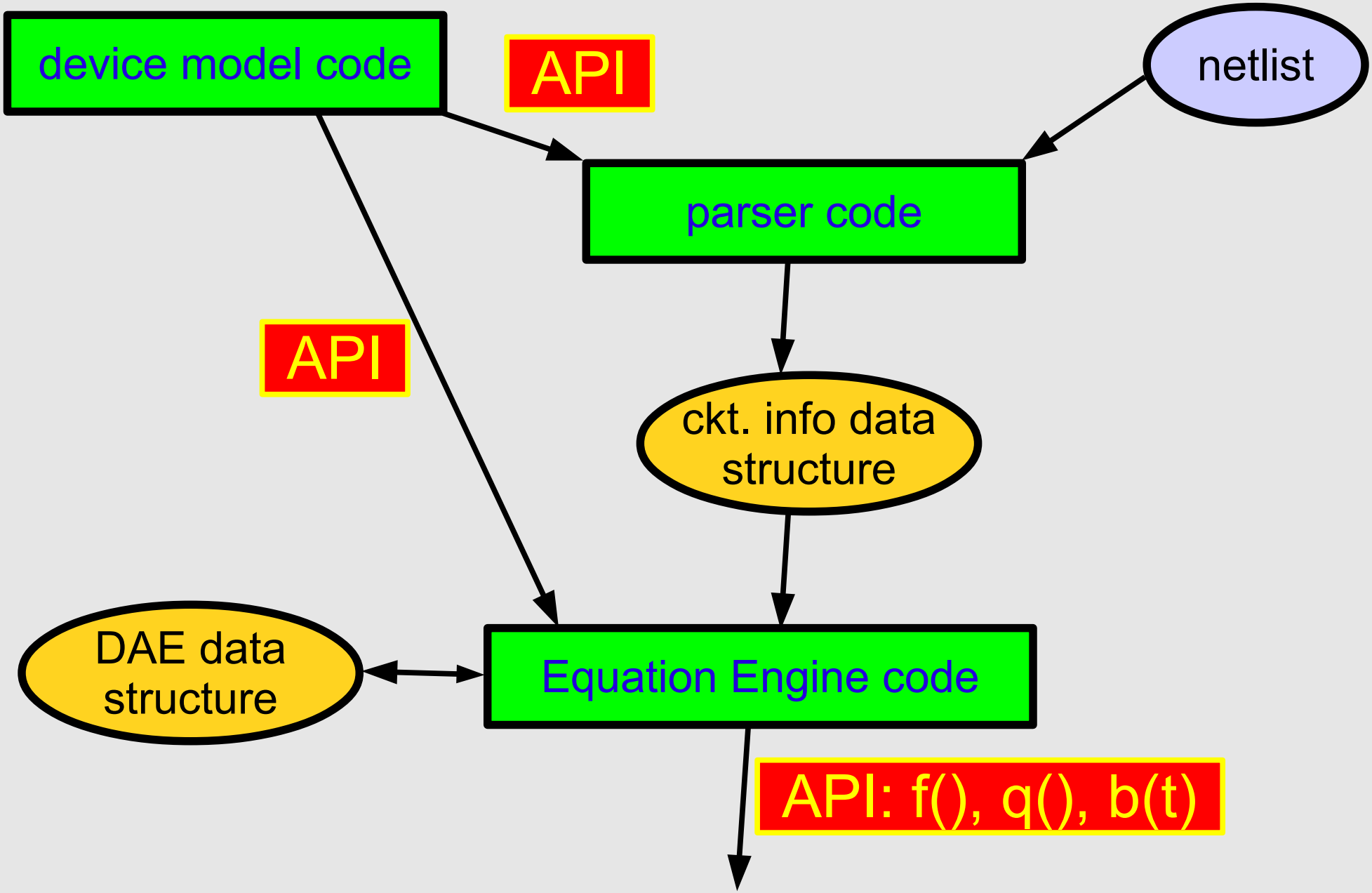
reaction

SBML description

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml level="3" version="1" xmlns="http://www.sbml.org/sbml/level3/version1/core">
  <model extentUnits="mole" timeUnits="second">
    <listOfUnitDefinitions>
      <unitDefinition id="per_second">
        <listOfUnits>
          <unit kind="second" exponent="-1" scale="0" multiplier="1"/>
        </listOfUnits>
      </unitDefinition>
      <unitDefinition id="litre_per_mole_second">
        <listOfUnits>
          <unit kind="mole"   exponent="-1" scale="0" multiplier="1"/>
          <unit kind="litre"  exponent="1"  scale="0" multiplier="1"/>
          <unit kind="second" exponent="-1" scale="0" multiplier="1"/>
        </listOfUnits>
      </unitDefinition>
    </listOfUnitDefinitions>
    <listOfCompartments>
      <compartment id="comp" size="1e-14" spatialDimensions="3" units="litre" constant="true"/>
    </listOfCompartments>
    <listOfSpecies>
      <species compartment="comp" id="E"  initialAmount="5e-21" boundaryCondition="false"
               hasOnlySubstanceUnits="false" substanceUnits="mole" constant="false"/>
      <species compartment="comp" id="S"  initialAmount="1e-20" boundaryCondition="false"
               hasOnlySubstanceUnits="false" substanceUnits="mole" constant="false"/>
      <species compartment="comp" id="P"  initialAmount="0"     boundaryCondition="false"
               hasOnlySubstanceUnits="false" substanceUnits="mole" constant="false"/>
      <species compartment="comp" id="ES" initialAmount="0"     boundaryCondition="false"
               hasOnlySubstanceUnits="false" substanceUnits="mole" constant="false"/>
    </listOfSpecies>
    <listOfReactions>
      <reaction id="veq" reversible="true" fast="false">
        <listOfReactants>
          <speciesReference species="E"  stoichiometry="1" constant="true"/>
          <speciesReference species="S"  stoichiometry="1" constant="true"/>
        </listOfReactants>
```

```
        <listOfProducts>
          <speciesReference species="ES" stoichiometry="1" constant="true"/>
        </listOfProducts>
        <kineticLaw>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <times/>
              <ci>comp</ci>
              <apply>
                <minus/>
                <apply>
                  <times/>
                  <ci>kon</ci>
                  <ci>E</ci>
                  <ci>S</ci>
                </apply>
                <apply>
                  <times/>
                  <ci>koff</ci>
                  <ci>ES</ci>
                </apply>
              </apply>
            </apply>
          </math>
          <listOfLocalParameters>
            <localParameter id="kon"  value="1000000" units="litre_per_mole_second"/>
            <localParameter id="koff" value="0.2"     units="per_second"/>
          </listOfLocalParameters>
        </kineticLaw>
      </reaction>
      <reaction id="vcat" reversible="false" fast="false">
        <listOfReactants>
          <speciesReference species="ES" stoichiometry="1" constant="true"/>
        </listOfReactants>
        <listOfProducts>
          <speciesReference species="E"  stoichiometry="1" constant="true"/>
          <speciesReference species="P"  stoichiometry="1" constant="true"/>
        </listOfProducts>
        <kineticLaw>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <times/>
              <ci>comp</ci>
              <ci>kcat</ci>
              <ci>ES</ci>
            </apply>
          </math>
          <listOfLocalParameters>
            <localParameter id="kcat" value="0.1" units="per_second"/>
          </listOfLocalParameters>
        </kineticLaw>
      </reaction>
    </listOfReactions>
  </model>
</sbml>
```

Parsing and Equation Setup Flow



DAEs with inputs and outputs

- MNA equations in **standard DAE form**

$$\frac{d}{dt}\vec{q}(\vec{x}(t)) + \vec{f}(\vec{x}(t)) + \vec{b}(t) = \vec{0}$$

  - (vector) Differential-Algebraic Equation system
  - x: unknowns; q(): charges/fluxes; f(): resistive; b(t): inputs

- Incorporating inputs and outputs into the DAE:

state (unknown) vector
size n

input coupling matrix
size n x n_{inp}

inputs
size n_{inp}

DAE
size n

$$\frac{d}{dt}\vec{q}(\vec{x}(t)) + \vec{f}(\vec{x}(t)) + B(\vec{x})\vec{u}(t) = \vec{0}$$

$$\vec{y}(t) = C(\vec{x})\vec{x}(t) + D(\vec{x})\vec{u}(t)$$

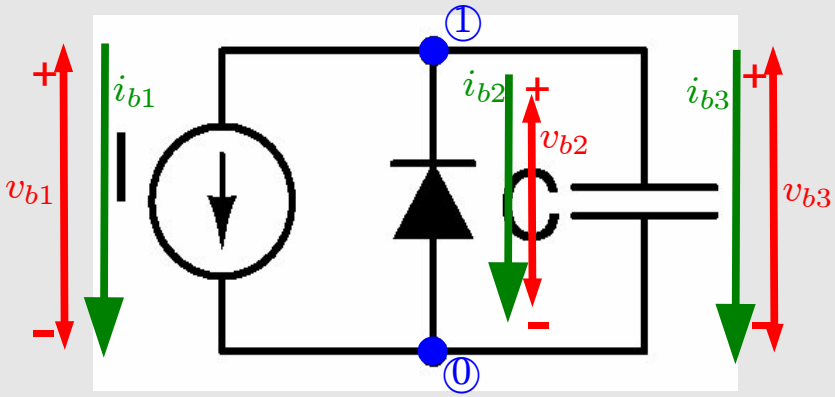
outputs
size n_{out}

state->output coupling matrix
size n_{out} x n

input->output feedthrough matrix
size n_{out} x n_{inp}

diode-capacitor-ISRC example (NF)

- The circuit and its equations:



$$I(t) - \text{diode}(-e_1; I_S, V_t) + C\frac{de_1}{dt} = 0$$

input: $I(t)$

output: $-e_1(t)$

- DAE form with I/O:

$$\frac{d}{dt}\vec{q}(\vec{x}(t)) + \vec{f}(\vec{x}(t)) + B(\vec{x})\vec{u}(t) = \vec{0}$$

$$\vec{y}(t) = C(\vec{x})\vec{x}(t) + D(\vec{x})\vec{u}(t)$$

$\vec{x}(t) = [e_1(t)]$

$\vec{q}(\vec{x}) = [Ce_1]$

$\vec{f}(\vec{x}) = [-\text{diode}(-e_1; I_S, V_t)]$

$\vec{u}(t) = [I(t)]$ $B(\vec{x}) = [1]$

$C(\vec{x}) = [-1]$ $D(\vec{x}) = [0]$

- How does this change if another current source is added in parallel?

DAE API v6 (MATLAB)

```
%%%%%%%%%%%%%%%%%%%%%%%% DAEAPIv6 FUNCTIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%DAE.version (string). Use: versionstring = DAE.version;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% SIZES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%DAE.nunks (function handle). Use: nunks = feval(DAE.nunks,DAE);
%       - returns the number of unknowns (size of the state vector x).
%DAE.neqns (function handle). Use: neqns = feval(DAE.neqns,DAE);
%       - returns the number of state equations (ie, size of q(x) and f(x)).
%DAE.ninputs (function handle). Use: ninps = feval(DAE.ninputs,DAE);
%       - returns the number of inputs (ie, size of u(t)).
%DAE.noutputs (function handle). Use: nouts = feval(DAE.noutputs,DAE);
%       - returns the number of outputs (ie, size of y(t)).
%DAE.nparms (function handle). Use: nparms = feval(DAE.nparms,DAE);
%       - returns the number of parameters (ie, size of DAE.parms).
%DAE.nNoiseSources (function handle). Use: m = feval(DAE.nNoiseSources,DAE);
%       - returns the number of noise inputs (ie, size of n(t)).
%
%%%%%%%%%%%%% NAMES of DAE, UNKS, I/O, EQNS, PARMS, NOISE SOURCES %%%%%%%%%%%%%%
%
%DAE.daename (function handle). Use: name = feval(DAE.daename, DAE);
%               name is a string - just a name for the DAE.
%DAE.unknames (function handle). Use: unames = feval(DAE.unknames,DAE);
%               unames is a cell array containing names of unknowns (strings)
%DAE.eqnnames (function handle). Use: eqnames = feval(DAE.eqnnames,DAE);
%               eqnames is a cell array containing names of equations (strings)
%DAE.inputnames (function handle). Use: inpnames = feval(DAE.inputnames,DAE);
%               inpnames is a cell array containing names of the inputs (strings)
%DAE.outputnames (function handle). Use: opnames = feval(DAE.outputnames,DAE);
%               opnames is a cell array containing names of the outputs (strings)
%DAE.parmnames (function handle). Use: pnames = feval(DAE.parmnames,DAE);
%               pnames is a cell array containing DAE parameter names (strings)
%DAE.NoiseSourceNames (function handle):
                                                        44,30-36      13%
```