Maplesoft
Mathematics • Modeling • Simulation
A Cybernet Group Company

Engineers – Stop Doing Algebra by Hand!

Save Time and Reduce Risk with Computer Algebra Systems

Manual equation manipulation is labor intensive, time consuming, and notoriously prone to error. Simply put, doing algebra by hand is expensive.

When faced with expensive processes, engineers find ways to mechanize and cut costs. Math should be no different.

Computer algebra systems mechanize equation manipulation, reducing the need for human involvement and hence eliminate a source of risk. Pioneered originally by mathematicians and physicists, two trends have influenced their popularity with engineers.

- Human-centered design principles have vastly improved usability. Tasks such as equation manipulation, differentiation and ODE solving are now much easier to do, reducing the need for specialized training.
- Computer algebra systems now also offer tools for numerical math, plotting, connectivity, data analysis, documentation and deployment. This means that algebraic computations can be fully integrated into the entire engineering design process.

Better usability and broader capability have, in effect, democratized computer algebra systems. They are now a pragmatic tool for engineers of all skills and disciplines.

This white paper first describes the benefits of computer algebra systems. Then, many different engineering applications are described.

Benefits of Computer Algebra Systems

Fewer Errors, and Faster than Manual Equation Manipulation

Manual equation manipulation requires intense cognitive effort. If that level of concentration is not maintained, errors will invariably pollute the equations.

Computer algebra systems, however, eliminate the errors that invariably accompany manual equation manipulation, and are much faster.

Additionally, removing the cognitive overhead associated with manual equation derivation enables engineers to concentrate on higher-level, higher-value tasks.

Model More Sophisticated Systems

As the size of engineering systems increase linearly, the size of the equations that describe those engineering systems increases exponentially.

A key example is the modeling of multiple degree of freedom (DOF) robotic systems. As the number of joints increases, the transformation matrices required to describe joint motion exponentially increase in size. At some point, equation manipulation by hand is impractical; software support is hence needed.

A corollary is that computer algebra systems can be used to model more sophisticated engineering systems than is possible by hand.

Computationally Faster than Numeric Computation

Numerical computation refers to the iterative solution of equations using software; this is computationally time-consuming.

In many cases, computer algebra systems can be used to rearrange equations to an explicit formulation; this eliminates the need for time-consuming iterative approaches.

Preserve Information about Model Structure

By delaying numeric evaluation until only strictly necessary, computer algebra systems preserve information about model structure and parameter relationships. This information can be used for code generation, parameter-based optimization, model simplification and more.

How Do Engineers Use Computer Algebra Systems?

Introduction

We will now, through an exploratory application-based approach, illustrate how computer algebra systems are typically used by engineers.

Maple, a math tool with a hybrid symbolic-numeric math engine, is used to illustrate each example. Maple's broader calculation management features are used to document each example with natural math notation, images and text.

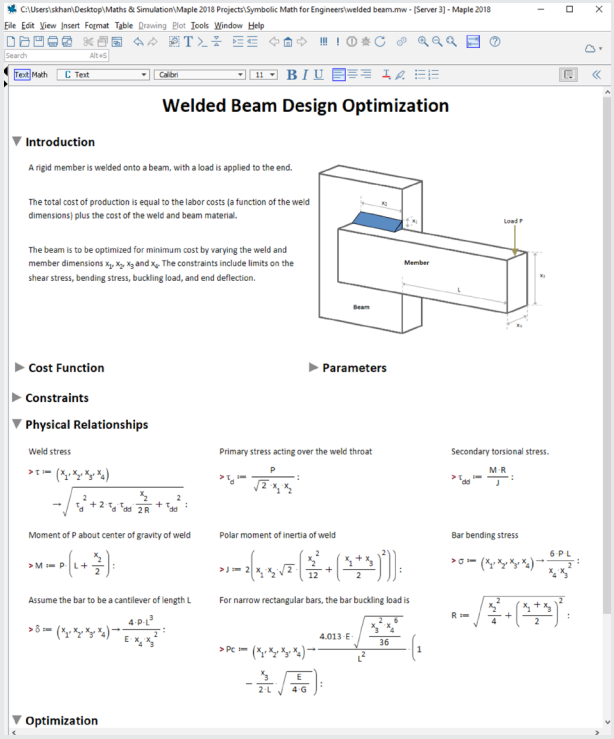


Figure 1. Maple provides a complete calculation management environment in which engineers can record and reuse the underlying reasoning, data and derivations

Transformation Matrices for a Multi-DOF Robot

The modeling of robotic arm manipulators involves the derivation of transformation matrices; these matrices grow in size as the degrees of freedom increase. Deriving these matrices by hand would take hours, with a high probability of introducing errors. However, computer algebra systems will derive transformation matrices for an arbitrarily complex robot in a matter of seconds, while eliminating the risk of error.

Figure 2 demonstrates how Maple can be used to generate the Denevit & Hartenberg matrices, which assist in determining the coordinate transformations between joints in a robotic manipulator.

$$A1 := \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} :$$

$$A2 := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} :$$

$$B1 := A2.A1 :$$

$$A3 := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} :$$

$$A4 := \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} :$$

$$B2 := A3.A4 :$$

$$H := B1.B2;$$

$$H := \begin{bmatrix} \cos(\theta) & -\sin(\theta)\cos(\alpha) & \sin(\theta)\sin(\alpha) & \cos(\theta)\,a \\ \sin(\theta) & \cos(\theta)\cos(\alpha) & -\cos(\theta)\sin(\alpha) & \sin(\theta)\,a \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2. *Denavit and Hartenberg formulation for robotic manipulators*

Inverse Kinematics

Inverse kinematics involves finding the joint parameters to move a robot arm to a desired position. **Figure 3** demonstrates how the position constraints of a double pendulum can be rearranged to give the joint angles.

Code Translation of Joint Angles to C

This equation in **Figure 3** is now converted to C code.

Computer algebra systems can identify and factor out common subexpressions; this makes the code far more numerically efficient than hand-written code.

$$> \text{posCons} := \begin{bmatrix} \cos(\theta_1)\,Tx + \sin(\theta_2) + \sin(\theta_1)\,Ty \\ -\sin(\theta_1)\,Tx - 1 - \cos(\theta_2) + \cos(\theta_1)\,Ty \end{bmatrix} :$$

$$> \text{sol} := \text{solve}(\{\text{posCons}[1], \text{posCons}[2]\}, \{\theta_1, \theta_2\}, \text{explicit})$$

$$\text{sol} := \left\{ \theta_1 = \arctan\left(\frac{-Tx^2 - Ty^2 + \frac{\left(Tx^2 Ty + Ty^3 + \sqrt{-Tx^6 - 2Tx^4 Ty^2 - Tx^2 Ty^4 + 4Tx^4 + 4Tx^2 Ty^2}\right) Ty}{Tx^2 + Ty^2}}{Tx}, \right.\right.$$

$$\left. \frac{Tx^2 Ty + Ty^3 + \sqrt{-Tx^6 - 2Tx^4 Ty^2 - Tx^2 Ty^4 + 4Tx^4 + 4Tx^2 Ty^2}}{Tx^2 + Ty^2} \right), \theta_2$$

$$= \arctan\left(\frac{\left(-2Tx^2 - 2Ty^2\right)\left(Tx^2 Ty + Ty^3 + \sqrt{-Tx^6 - 2Tx^4 Ty^2 - Tx^2 Ty^4 + 4Tx^4 + 4Tx^2 Ty^2}\right)}{4Tx\left(Tx^2 + Ty^2\right)} + \frac{Tx^2 Ty + Ty^3}{2Tx}, \frac{Tx^2}{2} \right.$$

$$\left. + \frac{Ty^2}{2} - 1 \right) \Bigg\}, \left\{ \theta_1 = \arctan\left(\frac{-Tx^2 - Ty^2 - \frac{\left(-Tx^2 Ty - Ty^3 + \sqrt{-Tx^6 - 2Tx^4 Ty^2 - Tx^2 Ty^4 + 4Tx^4 + 4Tx^2 Ty^2}\right) Ty}{Tx^2 + Ty^2}}{2Tx}, \right.\right.$$

$$\left. - \frac{-Tx^2 Ty - Ty^3 + \sqrt{-Tx^6 - 2Tx^4 Ty^2 - Tx^2 Ty^4 + 4Tx^4 + 4Tx^2 Ty^2}}{2\left(Tx^2 + Ty^2\right)} \right), \theta_2 = \arctan\left(\right.$$

$$\left. - \frac{\left(-2Tx^2 - 2Ty^2\right)\left(-Tx^2 Ty - Ty^3 + \sqrt{-Tx^6 - 2Tx^4 Ty^2 - Tx^2 Ty^4 + 4Tx^4 + 4Tx^2 Ty^2}\right)}{4Tx\left(Tx^2 + Ty^2\right)} + \frac{Tx^2 Ty + Ty^3}{2Tx}, \frac{Tx^2}{2} + \frac{Ty^2}{2} \right.$$

$$\left. - 1 \right) \Bigg\}$$

Figure 3. *Rearranging inverse kinematics equations*

```
> CodeGeneration[C](θ, optimize, deducetypes = false)
t1 = l2 * l2;
t3 = 0.2e1 * l2 * z;
t4 = x * x;
t5 = y * y;
t6 = z * z;
t10 = t1 - t3 + t4 + t5 + t6 + (l4 + l5 + l3) * (-l4 - l5 + l3);
t11 = t10 * t10;
t14 = sqrt((t4 + t5) * t11);
t16 = l4 + l5;
t26 = l3 * l3;
t28 = t16 * t16;
t32 = sqrt(-0.1e1 / t28 / t26 * (-t3 + t4 + t5 + t6 + (l5 + l2 + l4 + l3) * (-
l5 + l2 - l4 - l3)) * (-t3 + t4 + t5 + t6 + (-l5 + l2 - l4 + l3) * (l5 + l2 +
l4 - l3)));
t35 = l2 - z;
t40 = 0.1e1 / l3;
t42 = 0.1e1 / (t1 - t3 + t4 + t5 + t6);
t51 = atan2(t42 * t40 / t10 * (-t32 * t16 * t14 * l3 + t35 * t11), t40 * t42 *
(t32 * t35 * t16 * l3 + t14));
```

Figure 4. *Translating equations to C code*

Rearranging Photovoltaic Diode Equation

Figure 5 gives the equation that describes the behavior of a photovoltaic diode. Using standard mathematical functions, the equation cannot be rearranged to give an explicit value for the forward current, Id.

$$\text{diodeEq} := I_d = I_{ph} - I_0 \cdot e^{\frac{V_d + I_d \cdot R_s}{n \cdot V_t}} - \frac{V_d + I_d \cdot R_s}{R_{sh}} :$$

Figure 5. *Photovoltaic diode equation*

However, the equation can be rearranged using special functions. These functions are normally only encountered in advanced mathematical analysis; a few examples are LambertW, Fresnel, Bessel, and Appell.

Special functions are increasingly implemented in computer algebra tools, allowing engineers to use these functions without any specialized training. This is spurring the appearance of special functions in many advanced modelling applications.

Figure 6 demonstrates how Maple uses special functions to give an explicit equation for Id. The resulting equation uses the LambertW function.

$$\text{> res} := \text{solve}(\text{diodeEq}, I_d)$$

$$\text{res} := -\frac{\left(-W\left(-\frac{I_0 R_s R_{sh} e^{\frac{R_{sh}\left(I_{ph} R_s + V_d\right)}{n V_t \left(R_s + R_{sh}\right)}}}{-R_s V_t n - R_{sh} V_t n}\right) + \frac{R_{sh}\left(I_{ph} R_s + V_d\right)}{n V_t \left(R_s + R_{sh}\right)}\right) n V_t + V_d}{R_s}$$

Figure 6. *Rearranging the photovoltaic diode equation using the LambertW function*

Convolution

Convolving a square wave with itself is mathematically simple, as illustrated in Figure 7; the resulting equation is a triangle wave. This process uses the concept of symbolic integration.

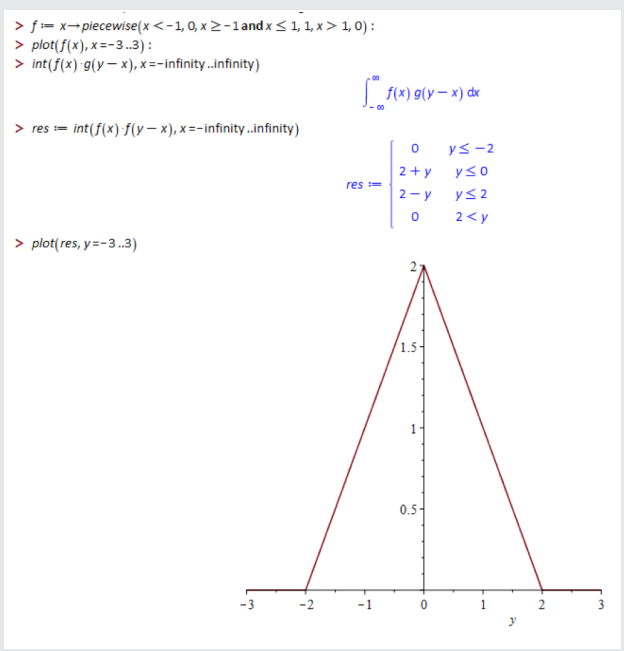


Figure 7. *Symbolic Convolution*

This is a simple example that could be easily processed by hand; larger systems, however, are not as amenable to manual derivation. **Figure 8**, for example, demonstrates the convolution of a cosine under a Hann function with itself; the result is very large and is only partially shown.

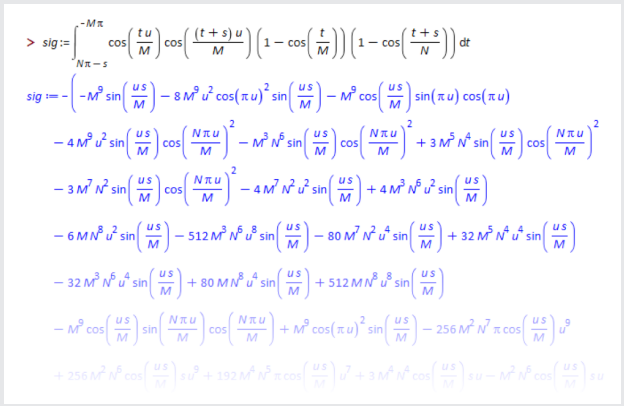


Figure 8. *Convolution of a cosine under a Hann Function with a cosine under a wider Hann function*

Translating a Netlist to a Transfer Function

A netlist is a textual description of an electric circuit, and describes component connections and their parameter values. These are typically used by SPICE-based circuit simulation tools, such as Saber®.

However, these dedicated tools do not provide tools for advanced analysis (such as extreme value analysis) or customized report generation. Hence the circuit must be analyzed in a math tool, but the equations would first need to be derived; an electrical engineer might do this by hand by applying the principles of nodal analysis and mesh analysis.

However, translating a netlist into as equation is a well-defined problem that can be implemented in a symbolic math tool. **Figure 9** demonstrates how Maple translates a netlist describing a one-pole filter into a transfer function (using a free add-on called Syrup).

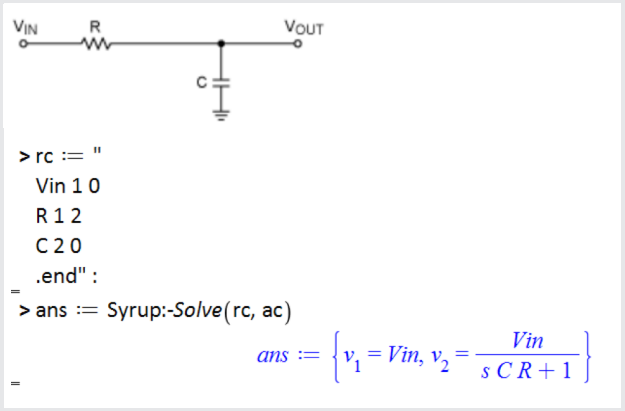


Figure 9. *One Pole Filter*

Translate the Transfer Function of a Biquad Filter to a Differential Equation

AC analyses of analog electrical circuits involve simulating time-domain algebraic or differential equations.

Figure 10 demonstrates how Maple translates the transfer function of a biquad filer to a time-domain equation.

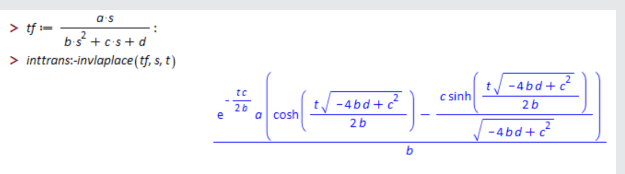


Figure 10. *Convert a transfer function to the time domain*

Rearranging Heat Balance Equations

The thermal efficiency of a thermodynamic cycle is a function of the heat and mass flows around a system. Deriving the heat and mass balances require a lgebraic manipulation.

Consider the Rankine cycle with two-stage regeneration in **Figure 11**.

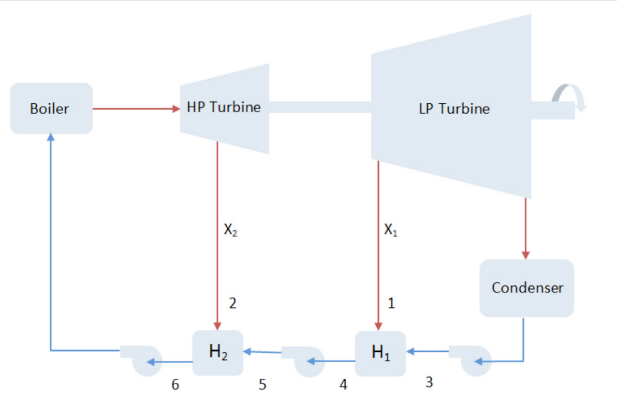


Figure 11. *Rankine Cycle with Two-Stage Regeneration*

A heat balance on the two pre-heaters H_1 and H_2 gives these equations,

$$h_3(1 - X_1 - X_2) + h_1 X_1 = h_4(1 - X_2)$$

$$h_5(1 - X_2) + X_2 h_2 = h_6$$

where X_1 and X_2 are the mass fractions of the working fluid extracted in the high and low pressure turbines, and h_n is the specific enthalpy at points $n = 1 .. 6$.

```
eq1 := h3 (1 − X1 − X2) + h1 X1 = h4 (1 − X2) :
eq2 := h5 (1 − X2) + X2 h2 = h6 :
solve({eq1, eq2}, {X1, X2})
```

$$\left\{X_1 = -\frac{h_2 h_3 - h_2 h_4 - h_3 h_6 + h_4 h_6}{h_1 h_2 - h_1 h_5 - h_2 h_3 + h_3 h_5}, X_2 = -\frac{h_5 - h_6}{h_2 - h_5}\right\}$$

Figure 12. *Heat Balance on Preheaters and Symbolic Manipulation of Equations*

Figure 12 illustrates how Maple is employed to rearrange these equations to give X_1 and X_2. If two states at points 1-6 are known (e.g. temperature and pressure) then

- enthalpy values can be determined,
- and X_1 and X_2 can then be calculated.

Integrating an Empirical Equation for Heat Capacity

The specific heat capacity of many chemicals is often described by empirical polynomials in temperature.

These polynomials can be integrated (as illustrated in **Figure 13**) to give an expression that can be used to calculate changes in enthalpy.

```
> Cp := 8.314510 · ( 3.730042760 10^6 / T^2 − 13835.01485 / T + 20.49107091 − 0.001961974759 T
    + 4.727313040 10^−7 T^2 − 3.728814690 10^−11 T^3 + 1.623737207 10^−15 T^4 ) :
> int(Cp, T)
170.3732140 T − 0.008156429375 T^2 − 3.101347783 10^7 / T − 115031.3693 ln(T) + 1.310176385 10^−6 T^3
    − 7.750816758 10^−11 T^4 + 2.700115848 10^−15 T^5
```

Figure 13. *Integrating Empirical Equation for Specific Heat Capacity*

Beam Deflection with Distributed and Point Load

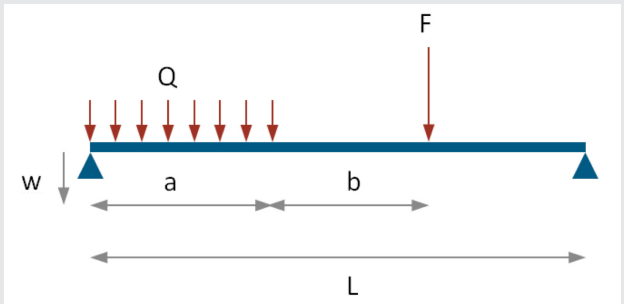


Figure 14. *Beam with distributed and point load*

Students of structural and civil engineering encounter the Euler-Bernoulli beam deflection equation early in their education:

$$EI\frac{d^4}{dx^y}w(x) = q(x)$$

This equation connects the deflection of the beam *w(x)* to the applied load *q(x)*. With the appropriate initial and boundary conditions, the equation can be solved to give explicit expressions describing the deflection.

For simple load cases, such as a simply supported beam with a distributed load, the Euler-Bernoulli equation can be solved manually.

However, more complex load cases would take significant amounts of time to solve by hand. For example, consider the simply supported beam in **Figure 14**; the beam has a uniform load (across part of the beam) and a point load.

Mathematically, the distributed load can be described by a Heaviside step function, and the point load described by a Dirac function. *q(x)* hence becomes

$$q(x) = Q(1 - Heaviside(x - a)) + F\ Dirac(x - (a + b))$$

Figure 15 describes how Maple is used to solve the Euler-Bernouilli equation with this loading.

Euler-Bernoulli equation

$$> de := EI \cdot \frac{d^4}{dx^4} w(x) = q(x) :$$

Initial and boundary conditions

> ibc := w(0) = 0, w(L) = 0, (D@@2)(w)(0) = 0, w(L) = 0, (D@@2)(w)(L) = 0 :

Distributed load

> q := x→Q·(1 − Heaviside(x − a)) + F·Dirac(x − (a + b)) :

> deSol := dsolve({de, ibc}, w(x)) :

deflection := simplify(rhs(deSol), symbolic) assuming positive

$$deflection := \frac{1}{144EIL}\left(4Fx(L-x)(L+x)(L-a-b)^3\delta'(L-a-b) + 48F\left(-\frac{x^2}{2} + \left(-\frac{a}{2} + L - \frac{b}{2}\right)(a+b)\right)x(L-a-b)\,\theta(L-a-b) + 24Fx(L-x)(L+x)(L-a-b)^2\delta(L-a-b) - 24FL(-x+a+b)^3\theta(x-a-b) - 6\left(\frac{x(L-x)(L+x)(L-a)^4\delta'(L-a)}{6} + \frac{4x(L-x)(L+x)(L-a)^3\delta(L-a)}{3} + x(L-a)^2\left(L^2 + 2La - a^2 - 2x^2\right)\theta(L-a) + \left((-x+a)^4\theta(x-a) - xL^3 + 2Lx^3 - x^4\right)L\right)Q\right)$$

Figure 15. *Solving the Euler-Bernoulli Beam Bending Equation with a Distributed and Point Load*

Closed Loop Transfer Function

Consider the closed loop control system in **Figure 16**. Typically, an engineer may want to calculate the closed loop transfer function for such a system.

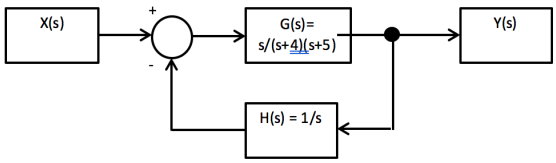


Figure 16. *Closed Loop Control System*

Figure 17 illustrates how the closed loop transfer function is derived using Maple. The steps, while mathematically straightforward, are tedious to do by hand for non-trivial systems.

$$> G := \frac{s}{(s+4)\cdot(s+5)} :$$

$$H := \frac{1}{s} :$$

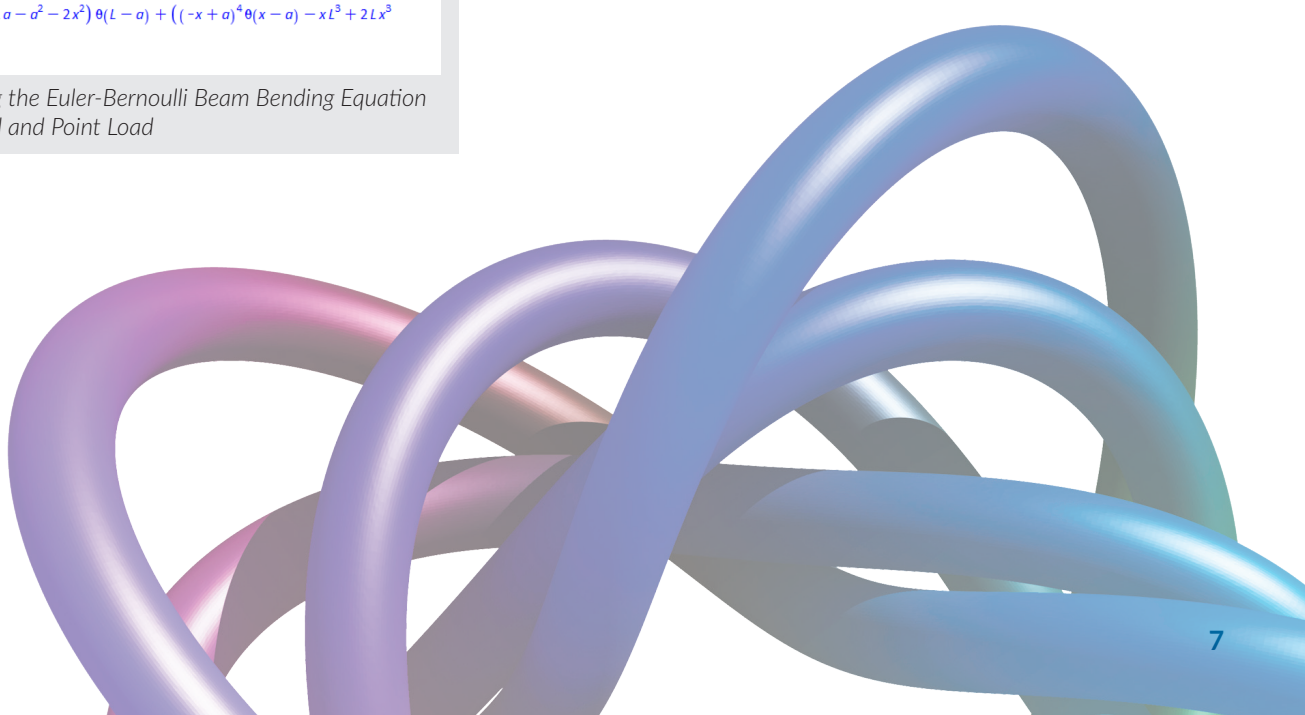
$$> simplify\left(\frac{G}{1 + G \cdot H}\right)$$

$$\frac{s}{s^2 + 9s + 21}$$

Figure 17. *Closed Loop Transfer Function for a Control Loop*

Controllability Matrix of a DC Motor

In **Figure 18**, we extract the symbolic controllability matrix of a DC motor described by differential equations.



```
> eq_sym :=
  L·i'(t) + R·i(t) = v(t) − K·θ'(t),
  J·θ''(t) + b·θ'(t) + Ks·θ(t) = K·i(t) :
> sys_sym := DynamicSystems:-StateSpace([eq_sym], inputvariable = [v(t)], outputvariable = [θ(t), i(t)]) :
> DynamicSystems:-ControllabilityMatrix(sys_sym)
```

$$\begin{bmatrix} \frac{1}{L} & -\frac{R}{L^2} & \frac{R^2}{L^3} - \frac{K^2}{L^2 J} \\ 0 & 0 & \frac{K}{J L} \\ 0 & \frac{K}{J L} & -\frac{K R}{J L^2} - \frac{b K}{J^2 L} \end{bmatrix}$$

Figure 18. Symbolic Controllability Matrix

Terminal Settling Velocity

Consider a spherical particle falling in a fluid. **Figure 19** gives the equations for the drag force and the buoyancy force; the terminal settling velocity is reached when both are equal.

The equations are rearranged to give an explicit expression for the settling velocity.

Drag force

$$> Fd := \frac{1}{2}\cdot\pi\cdot\frac{Dia^2}{4}\cdot CD\cdot\rho_f\cdot v^2 :$$

Buoyancy force

$$> Fb := \left(\rho_p - \rho_f\right)\cdot g\cdot\pi\cdot\frac{Dia^3}{6} :$$

The terminal settling velocity is reached when the drag force equals the buoyancy force.

$$> res := v_{terminal} = solve(Fb = Fd, v)[1]$$

$$res := v_{terminal} = \frac{2\sqrt{-3\,CD\,\rho_f\,g\,Dia\left(\rho_f - \rho_p\right)}}{3\,CD\,\rho_f}$$

Figure 19. Terminal Settling Velocity of a Settling Particle

Balancing Chemical Equations

Monomethylhydrazine (CH_6N_2) and Dinitrogen Tetroxide (N_2O_4) are typically used in rocket propulsion as a fuel and oxidizer. In determining the theoretical rocket performance, the adiabatic flame temperature of the combustion products needs to be calculated; this partly involves balancing the carbon, hydrogen, oxygen and nitrogen atoms in the feed and combustion products.

Assuming the combustion products contain CO, HNO, H_2O, NO_2, O, CO_2, HO_2, H_2O_2, N_2, OH, H, H_2, NO, N_2O and O_2, the overall balance equation for the combustion of CH_6N_2 and N_2O_4 can be written thus.

$$a\,CH_6N_2 + b\,N_2O_4 = n_1\,CO + n_2\,HNO + n_3\,H_2O + n_4\,NO_2 + n_5\,O + n_6\,CO_2 + n_7\,HO_2 + n_8\,H_2O_2 + n_9\,N_2 + n_{10}\,OH + n_{11}\,H + n_{12}\,H_2 + n_{13}\,NO + n_{14}\,N_{20}\,n_{15}\,O_2$$

Generating the individual atom balances from the overall balance is painstaking because of the sheer number of chemical species. As illustrated in **Figure 20**, the process can be mechanized with computer algebra.

```
> eq :=
  a (C + 6 H + 2 N) + b (2 N + 4 O) =
   n1 (C + O) + n2 (H + N + O) + n3 (2 H + O) + n4 (N + 2 O) + n5 O + n6 (C + 2 O)
      + n7 (H + 2 O) + n8 (2 H + 2 O) + n9 ·2 N + n10 (O + H) + n11 H + n12 ·2 H
      + n13 (N + O) + n14 (2 N + O) + n15 ·2 O :
> elems := [ C, H, O, N] :
  eqr := collect(expand(rhs(eq)), elems) :
  eql := collect(expand(lhs(eq)), elems) :
  eqs := zip(`=`, map2(coeff, eql, elems), map2(coeff, eqr, elems)) :
Balance on C atoms
> eqs1
                                  a = n1 + n6
Balance on H atoms
> eqs2
                        6 a = n10 + n11 + 2 n12 + n2 + 2 n3 + n7 + 2 n8
Balance on O atoms
> eqs3
     4 b = n1 + n10 + n13 + n14 + 2 n15 + n2 + n3 + 2 n4 + n5 + 2 n6 + 2 n7 + 2 n8
Balance on N atoms
> eqs4
                          2 a + 2 b = n13 + 2 n14 + n2 + n4 + 2 n9
```

Figure 20. *Balancing the Combustion Reaction of Monomethylhydrazine (CH_6N_2) and Dinitrogen Tetroxide (N_2O_4)*

Combustion can produce many more species than used in this example; generating the atom balance equations by hand would be laborious, but is easy with computer algebra.

Battery Modeling and Model Reduction

Electrochemical battery models derived from porous electrode theory are described by partial differential equations. While being physically accurate, these models are computationally intensive to simulation.

Several techniques are used to simplify these models while retaining physical accuracy. These techniques include collocation and Galerkin's method; both transform non-linear PDEs into a set of ODEs and are described elsewhere (Dao et al, 2012).

Conclusion

The 1950s and 1960s saw the birth of the first computer algebra systems. Due to the skills and training of their creators, these innovative tools were first designed for the needs of mathematicians and physicists.

Initially, a few forward-thinking engineers exploited symbolic math for advanced research applications. The benefits, however, remained out of reach for the vast majority of engineers.

This started to change in the early eighties with the advent of cheap computing power. The next 30 years also saw the evolution of the human-centered design principles that radically improved the usability of computer algebra systems.

Moreover, a maturing feature set, including tools for managing calculations as well as doing calculations, made integrating mechanized algebra into the engineering design process much simpler

Computer algebra systems thus gradually entered the mainstream consciousness of engineers, and have grown in popularity year-on-year. Moreover, the applications discussed in this white paper clearly demonstrate the benefits of mechanized algebra across the entire breadth of engineering.

References

[1] Simplification and order reduction of lithium-ion battery model based on porous-electrode theory, Thanh-Son Dao et al., Journal of Power Sources, Volume 198, 15 January 2012, Pages 329-337

Maplesoft
Mathematics • Modeling • Simulation
A Cybernet Group Company

www.maplesoft.com | info@maplesoft.com
Toll-free: (US & Canada) 1-800-267-6583 | Direct:1-519-747-2373