Symbolic Code Generation & Differentiation

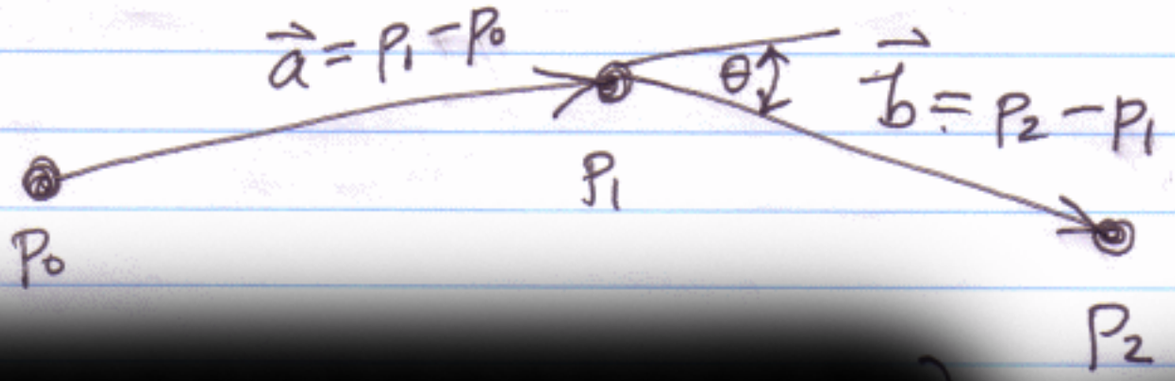
Doug James

Motivation: Bending Force

$$f_1 = -f_0 - f_2.$$

$$f_0 = +\frac{k}{2|a||b|}\left\{ -\vec{b} + \frac{(a\cdot b)}{ab}\frac{|b|}{|a|}\vec{a} \right\} = \frac{k}{2(a||b|}\left\{ -\vec{b} + \frac{(\vec{a}\cdot\vec{b})}{\|a\|^2}\vec{a} \right\}.$$

$$f_2 = +\frac{k}{2(a||b)}\left\{ \vec{a} - \frac{(a\cdot b)}{ab}\frac{|a|}{|b|}\vec{b} \right\} = \frac{k}{2(a||b|}\left\{ +\vec{a} - \frac{(a\cdot b)}{\|b\|^2}\vec{b} \right\}.$$



Now take a second derivative!

Symbolic math tools

- Useful for differentiation & optimized code gen.
- Examples:
  - Maple
  - Matlab
  - Mathematica
  - ...

Motivation 2: Newton-Schulz Iteration

- See whiteboard

Example 1

Evaluate the algebraic expression

$$\frac{3}{2}\mathbf{S} - \frac{1}{2}\mathbf{S}\mathbf{S}^{\mathrm{T}}\mathbf{S}$$

where

$$\mathbf{S} = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix}$$

Example 1

```
restart
with(linalg); with(codegen);
S    := array(1..3,1..3,[[m00,m01,m02],[m10,m11,m12],[m20,m21,m22]]);
StS  := multiply(transpose(S),S);
A    := multiply(S, matadd(array(1..3,1..3, identity), StS, 1.5, -0.5));
C(A, optimized);
```

Example 1

```
t1 = m00*m00;
t3 = m10*m10;
t5 = m20*m20;
t7 = 0.15E1-0.5*t1-0.5*t3-0.5*t5;
t15 = -0.5*m00*m01-0.5*m10*m11-0.5*m20*m21;
t23 = -0.5*m00*m02-0.5*m10*m12-0.5*m20*m22;
t27 = m01*m01;
t29 = m11*m11;
t31 = m21*m21;
t33 = 0.15E1-0.5*t27-0.5*t29-0.5*t31;
t41 = -0.5*m01*m02-0.5*m11*m12-0.5*m21*m22;
t46 = m02*m02;
t48 = m12*m12;
t50 = m22*m22;
t52 = 0.15E1-0.5*t46-0.5*t48-0.5*t50;
A[0][0] = m00*t7+m01*t15+m02*t23;
A[0][1] = m00*t15+m01*t33+m02*t41;
A[0][2] = m00*t23+m01*t41+m02*t52;
A[1][0] = m10*t7+m11*t15+m12*t23;
A[1][1] = m10*t15+m11*t33+m12*t41;
A[1][2] = m10*t23+m11*t41+m12*t52;
A[2][0] = m20*t7+m21*t15+m22*t23;
A[2][1] = m20*t15+m21*t33+m22*t41;
A[2][2] = m20*t23+m21*t41+m22*t52;
```

Example 2

$$\|\mathbf{S}^T\mathbf{S} - \mathbf{I}\|_F^2$$

```
S    := array(1..3,1..3,[[Sm00,Sm01,Sm02],[Sm10,Sm11,Sm12],[Sm20,Sm21,Sm22]]);
StS  := multiply(transpose(S),S);
A    := matadd(StS, array(1..3,1..3, identity), 1, -1));
AF   := norm(A, 'frobenius') * norm(A, 'frobenius');
```

Example 2

$$\|\mathbf{S}^T\mathbf{S} - \mathbf{I}\|_F^2$$

```
t1 = Sm00*Sm00;
t2 = Sm10*Sm10;
t3 = Sm20*Sm20;
t5 = fabs(t1+t2+t3-1.0);
t6 = t5*t5;
t11 = fabs(Sm00*Sm01+Sm10*Sm11+Sm20*Sm21);
t12 = t11*t11;
t18 = fabs(Sm00*Sm02+Sm10*Sm12+Sm20*Sm22);
t19 = t18*t18;
t21 = Sm01*Sm01;
t22 = Sm11*Sm11;
t23 = Sm21*Sm21;
t25 = fabs(t21+t22+t23-1.0);
t26 = t25*t25;
t31 = fabs(Sm01*Sm02+Sm11*Sm12+Sm21*Sm22);
t32 = t31*t31;
t34 = Sm02*Sm02;
t35 = Sm12*Sm12;
t36 = Sm22*Sm22;
t38 = fabs(t34+t35+t36-1.0);
t39 = t38*t38;
t40 = t6+2.0*t12+2.0*t19+t26+2.0*t32+t39;
```

Example 2

$$\|\mathbf{S}^T\mathbf{S} - \mathbf{I}\|_F^2$$

```
t1 = Sm00*Sm00;
t2 = Sm10*Sm10;
t3 = Sm20*Sm20;
t5 = fabs(t1+t2+t3-1.0);                          ← Doh!
t6 = t5*t5;
t11 = fabs(Sm00*Sm01+Sm10*Sm11+Sm20*Sm21);        ← Doh!
t12 = t11*t11;
t18 = fabs(Sm00*Sm02+Sm10*Sm12+Sm20*Sm22);        ← Doh!
t19 = t18*t18;
t21 = Sm01*Sm01;
t22 = Sm11*Sm11;
t23 = Sm21*Sm21;
t25 = fabs(t21+t22+t23-1.0);                      ← Doh!
t26 = t25*t25;
t31 = fabs(Sm01*Sm02+Sm11*Sm12+Sm21*Sm22);        ← Doh!
t32 = t31*t31;
t34 = Sm02*Sm02;
t35 = Sm12*Sm12;
t36 = Sm22*Sm22;
t38 = fabs(t34+t35+t36-1.0);                      ← Doh!
t39 = t38*t38;
t40 = t6+2.0*t12+2.0*t19+t26+2.0*t32+t39;
```

Example 3: Newton-Schulz Iteration

- See code

Example 4: Neo-Hookean material

Input:

```
% declare symbols
syms F0 F1 F2 F3 F4 F5 F6 F7 F8;
F = [F0 F3 F6
     F1 F4 F7
     F2 F5 F8];
C = (F.')*F;
I_C = trace(C);
II_C = (1/2)*(I_C^2 - trace(C^2));
III_C = det(C);
E = 0.5 * (C - eye(3));
F = reshape(F, 9, 1);

% declare strain energy
syms mu J lambda;
strain_energy = mu/2 * (I_C - 3) - mu * log(J) + lambda / 2 * log(J)^2;

% differentiate
Rd_sym = sym('TEMP')*ones(1,9);
for i = 1:9
  Rd_sym(1,i) = -diff(strain_energy, F(i));
end
Kd_sym = sym('TEMP')*ones(9*9,1);
for i = 1:9
  for j = 1:9
    Kd_sym((i-1)*9+j) = diff(Rd_sym(1,i), F(j) );
  end
end

% generate C code
stiffnessDensity = maple('codegen[C]', Kd_sym, 'optimized');
```

Example 4: Neo-Hookean material

Output:

```
void NEO_HOOKEAN::stiffnessDensity(const Real* F, Real* stiffness)
{
  const double mu = _mu;

  stiffness[0] = -mu;          stiffness[27] = 0.0;         stiffness[54] = 0.0;
  stiffness[1] = 0.0;          stiffness[28] = 0.0;         stiffness[55] = 0.0;
  stiffness[2] = 0.0;          stiffness[29] = 0.0;         stiffness[56] = 0.0;
  stiffness[3] = 0.0;          stiffness[30] = -mu;         stiffness[57] = 0.0;
  stiffness[4] = 0.0;          stiffness[31] = 0.0;         stiffness[58] = 0.0;
  stiffness[5] = 0.0;          stiffness[32] = 0.0;         stiffness[59] = 0.0;
  stiffness[6] = 0.0;          stiffness[33] = 0.0;         stiffness[60] = -mu;
  stiffness[7] = 0.0;          stiffness[34] = 0.0;         stiffness[61] = 0.0;
  stiffness[8] = 0.0;          stiffness[35] = 0.0;         stiffness[62] = 0.0;

  stiffness[9] = 0.0;          stiffness[36] = 0.0;         stiffness[63] = 0.0;
  stiffness[10] = -mu;         stiffness[37] = 0.0;         stiffness[64] = 0.0;
  stiffness[11] = 0.0;         stiffness[38] = 0.0;         stiffness[65] = 0.0;
  stiffness[12] = 0.0;         stiffness[39] = 0.0;         stiffness[66] = 0.0;
  stiffness[13] = 0.0;         stiffness[40] = -mu;         stiffness[67] = 0.0;
  stiffness[14] = 0.0;         stiffness[41] = 0.0;         stiffness[68] = 0.0;
  stiffness[15] = 0.0;         stiffness[42] = 0.0;         stiffness[69] = 0.0;
  stiffness[16] = 0.0;         stiffness[43] = 0.0;         stiffness[70] = -mu;
  stiffness[17] = 0.0;         stiffness[44] = 0.0;         stiffness[71] = 0.0;

  stiffness[18] = 0.0;         stiffness[45] = 0.0;         stiffness[72] = 0.0;
  stiffness[19] = 0.0;         stiffness[46] = 0.0;         stiffness[73] = 0.0;
  stiffness[20] = -mu;         stiffness[47] = 0.0;         stiffness[74] = 0.0;
  stiffness[21] = 0.0;         stiffness[48] = 0.0;         stiffness[75] = 0.0;
  stiffness[22] = 0.0;         stiffness[49] = 0.0;         stiffness[76] = 0.0;
  stiffness[23] = 0.0;         stiffness[50] = -mu;         stiffness[77] = 0.0;
  stiffness[24] = 0.0;         stiffness[51] = 0.0;         stiffness[78] = 0.0;
  stiffness[25] = 0.0;         stiffness[52] = 0.0;         stiffness[79] = 0.0;
  stiffness[26] = 0.0;         stiffness[53] = 0.0;         stiffness[80] = -mu;
}
```

Example 4: Ogden material

See code output(!)

Input:

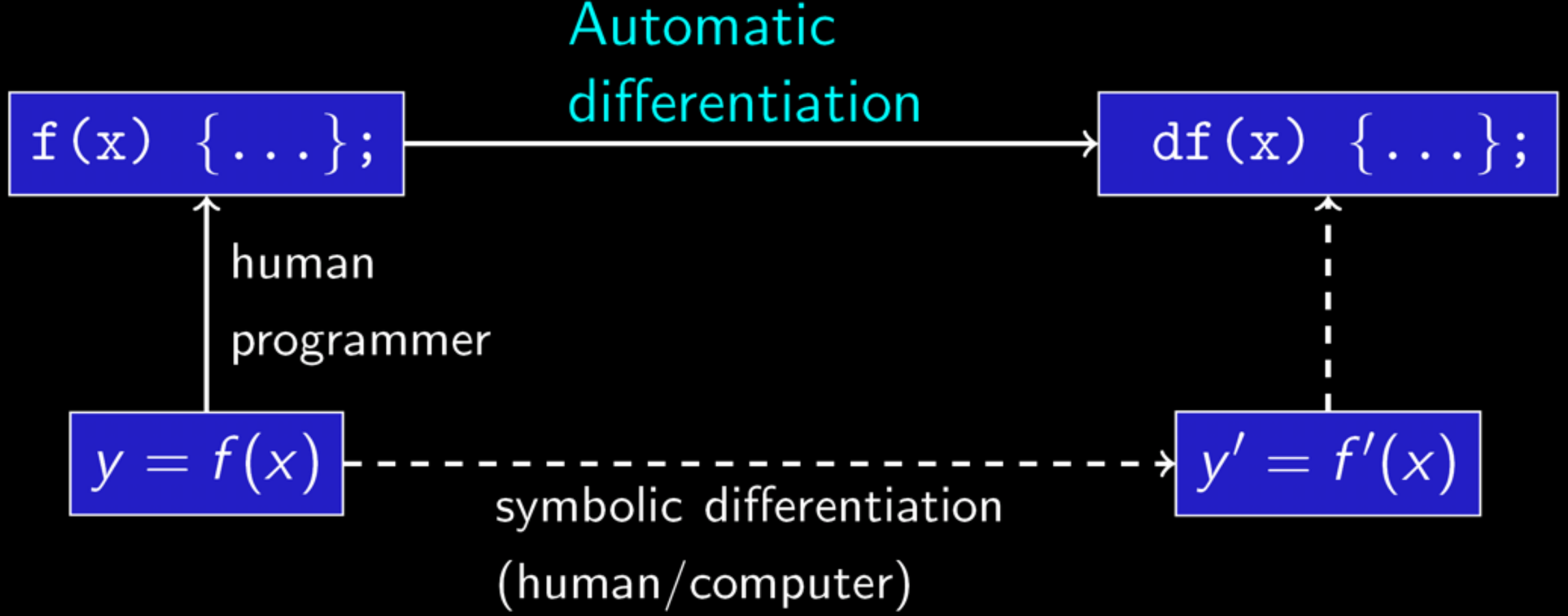
```
% declare symbols
syms F0 F1 F2 F3 F4 F5 F6 F7 F8;
F = [F0 F3 F6
     F1 F4 F7
     F2 F5 F8];
C = F * F.';
I_C = trace(C);
II_C = (1/2)*(I_C^2 - trace(C^2));
III_C = det(C);
E = 0.5 * (C - eye(3));
F = reshape(F, 9, 1);

% declare strain energy
syms alpha1 alpha2 C1 C2 C3;
p = I_C^2 - 3 * II_C
s = sqrt(p);
q = 2 * I_C^3 - 9 * I_C * II_C + 27 * III_C;
t = q / (2 * s^3);
theta = acos(t)/3;
r = (2/3) * s;
eig_1 = sqrt(I_C / 3 + r * cos(theta + (2 * pi) / 3 * (0)));
eig_2 = sqrt(I_C / 3 + r * cos(theta + (2 * pi) / 3 * (1)));
eig_3 = sqrt(I_C / 3 + r * cos(theta + (2 * pi) / 3 * (2)));
strain_energy = -C1 * log(1 - C2 * ((eig_1^alpha1 + eig_2^alpha1 + eig_3^alpha1 - 3)))
                + C3 * (eig_1^alpha2 + eig_2^alpha2 + eig_3^alpha2 - 3);

% differentiate
Rd_sym = sym('TEMP')*ones(1,9);
for i = 1:9
  Rd_sym(1,i) = -diff(strain_energy, F(i));
end
Kd_sym = sym('TEMP')*ones(9*9,1);
for i = 1:9
  for j = 1:9
    Kd_sym((i-1)*9+j) = diff(Rd_sym(1,i), F(j) );
  end
end

% generate C code
stiffnessDensity = maple('codegen[C]', Kd_sym, 'optimized');
```

Automatic Differentiation: Many compilers available



Automatic Differentiation: Many compilers available