A Symbolic Approach to the Projection Method

Nam Pham Mark Giesbrecht

University of Waterloo, Canada

The Tenth Asian Symposium on Computer Mathematics
Beijing 2012

Outline

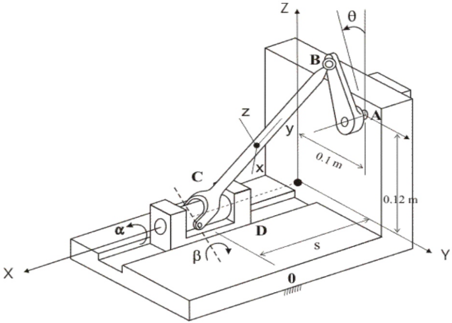
1. **Introduction**
   - Constrained mechanical system
   - The projection method
   - Problem definition

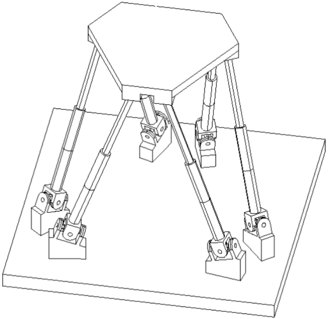
2. **Our symbolic-numeric code-generating algorithm**

3. **Experimental results**

How to do a simulation of a physical mechanical system?

1. Create a model of the system
2. Generate equations to describe the dynamic of the model
3. Solve the equations to determine the system response





Slider Crank Mechanism and Parallel Robot

Dynamics and kinematics of constrained mechanical system

- Kinematic constraint equations

$$C(x, t) = 0, \tag{1}$$

  with m nonlinear algebraic equations of n generalized coordinates $x_1, \cdots, x_n$ $(m < n)$.

- System dynamics

$$M\ddot{x} + C_J^T \lambda = F, \tag{2}$$

  where

  - C_J is the $m \times n$ Jacobian of the constraint matrix C
  - M is an $n \times n$ symmetric generalized mass matrix
  - λ is the $(m \times 1)$ Lagrange multiplier

- Solving these DAEs for $x(t)$ and $\lambda(t)$ is computationally expensive

Dynamics and kinematics of constrained mechanical system

- Kinematic constraint equations

$$C(x,t) = 0, \tag{1}$$

with m nonlinear algebraic equations of n generalized coordinates $x_1, \cdots, x_n$ $(m < n)$.
Symbolic computation: Allow parameters $z_1, \ldots, z_\ell \in \mathbb{R}$

- System dynamics

$$M\ddot{x} + C_J^T \lambda = F, \tag{2}$$

where

  - C_J is the $m \times n$ Jacobian of the constraint matrix C
  - M is an $n \times n$ symmetric generalized mass matrix
  - λ is the $(m \times 1)$ Lagrange multiplier

- Solving these DAEs for $x(t)$ and $\lambda(t)$ is computationally expensive

Solving these systems with parameters is extremely expensive!

The Projection Method

Blajer's (1992) projection method: hide algebraic equations from the dynamic equations:

- Find a null space basis D, an $n \times r$ matrix, such that

$$C_J D = 0 \text{ or } D^T C_J^T = 0, \tag{3}$$

- Multiply both sides of $M\ddot{x} + C_J^T \lambda = F$ by D^T

$$D^T M\ddot{x} = D^T F, \tag{4}$$

- Now we have ODEs in x and u, which can be easily solved to determine the coordinates x, velocity u, and constraint reaction λ during simulation

$$\dot{x} = Du, \tag{5}$$

$$D^T MD\dot{u} = D^T(F - M\dot{D}u), \tag{6}$$

$$\lambda = (CM^{-1}C^T)^{-1}C(M^{-1}F - \dot{D}u) \tag{7}$$

Numeric

- Numerical matrices are used to describe the system at a given instant in time.
- Values must be given for all parameters, *even if they aren't really known.*
- The model must be rebuilt at every time step during simulation.

Numeric vs. Symbolic Modelling and Simulation

Numeric

- Numerical matrices are used to describe the system at a given instant in time.
- Values must be given for all parameters, *even if they aren't really known.*
- The model must be rebuilt at every time step during simulation.

Symbolic

- All equations of motion are formulated once instead of every step during simulation
- Engineers can view the governing equations in a meaningful form
- Arbitrary substitutions for unknown quantities are not needed.

Numeric

- Numerical matrices are used to describe the system at a given instant in time.
- Values must be given for all parameters, *even if they aren't really known.*
- The model must be rebuilt at every time step during simulation.

Symbolic

- All equations of motion are formulated once instead of every step during simulation
- Engineers can view the governing equations in a meaningful form
- Arbitrary substitutions for unknown quantities are not needed.

Computer Algebra in Industrial Simulation

- MapleSim – symbolic physical modelling and simulation tool
- Talk tomorrow: *Symbolic Computation Techniques for Advanced Mathematical Modelling* by Junlin Xu

Our problem: Code generation for symbolic null spaces

Formal definition

Input: $A \in \mathbb{R}(z_1, z_2, \cdots, z_\ell)^{m \times n}$, with $m \leq n$ and rank r,

Output: straight-line code which takes parameters $\alpha_1, \ldots, \alpha_\ell \in \mathbb{R}$ and evaluates a specific (consistent) basis of the null space of A:

$$w_1(\alpha_1, \ldots, \alpha_\ell), w_2(\alpha_1, \ldots, \alpha_\ell), \ldots, w_{n-r}(\alpha_1, \ldots, \alpha_\ell) \in \mathbb{R}^n$$

Formal definition

Input: $A \in \mathbb{R}(z_1, z_2, \cdots, z_\ell)^{m \times n}$, with $m \leq n$ and rank r,

Output: straight-line code which takes parameters $\alpha_1, \ldots, \alpha_\ell \in \mathbb{R}$ and evaluates a specific (consistent) basis of the null space of A:

$$w_1(\alpha_1, \ldots, \alpha_\ell), w_2(\alpha_1, \ldots, \alpha_\ell), \ldots, w_{n-r}(\alpha_1, \ldots, \alpha_\ell) \in \mathbb{R}^n$$

Difficulties

- A is condensed with complex multivariate function
- Symbolic manipulation can lead to massive expression swell

Formal definition

Input: $A \in \mathbb{R}(z_1, z_2, \cdots, z_\ell)^{m \times n}$, with $m \leq n$ and rank r,

Output: straight-line code which takes parameters $\alpha_1, \ldots, \alpha_\ell \in \mathbb{R}$ and evaluates a specific (consistent) basis of the null space of A:

$$w_1(\alpha_1, \ldots, \alpha_\ell), w_2(\alpha_1, \ldots, \alpha_\ell), \ldots, w_{n-r}(\alpha_1, \ldots, \alpha_\ell) \in \mathbb{R}^n$$

Difficulties

- A is condensed with complex multivariate function
- Symbolic manipulation can lead to massive expression swell

Previous proposed solutions

- Apply linear graph theory to reduce the number of equations (McPhee 2004)
- Fraction-free factoring to control the generation of large expression (Zhou, 2004)

Our problem: Code generation for symbolic null spaces

Formal definition

Input: $A \in \mathbb{R}(z_1, z_2, \cdots, z_\ell)^{m \times n}$, with $m \leq n$ and rank r,

Output: straight-line code which takes parameters $\alpha_1, \ldots, \alpha_\ell \in \mathbb{R}$ and evaluates a specific (consistent) basis of the null space of A:

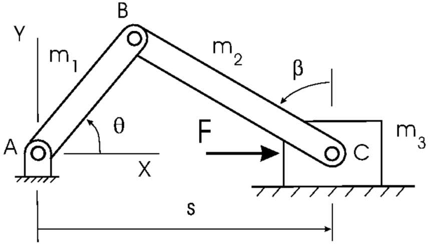
$$w_1(\alpha_1, \ldots, \alpha_\ell), w_2(\alpha_1, \ldots, \alpha_\ell), \ldots, w_{n-r}(\alpha_1, \ldots, \alpha_\ell) \in \mathbb{R}^n$$

Difficulties

- *A* is condensed with complex multivariate function
- Symbolic manipulation can lead to massive expression swell

Advantages of our approach

- Very fast
- Partial and incremental symbolic evaluation

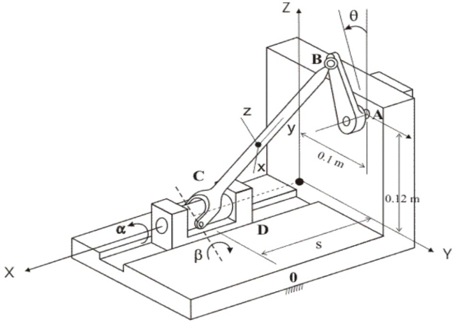


Example: Planar (2D) Slider Crank Mechanism

Planar Slider Crank Mechanism with 1 degree of freedom

$$C = \begin{pmatrix} L_1 cos\theta + L_2 sin\beta - s \\ L_1 sin\theta - L_2 cos\beta - s \\ \theta - f(t) \end{pmatrix} = 0$$

$$C_J = \frac{\delta(C)}{\delta(\theta, \beta)} = \begin{bmatrix} -L_1 sin\theta & L_2 cos\beta & -1 \\ L_1 cos\theta & L_2 sin\beta & 0 \\ 1 & 0 & 0 \end{bmatrix}$$



Example: Spatial (3D) Slider Crank Mechanism

In a slightly more complicated Spatial (3D) Slider Crank Mechanism, the second column is:

$$C_J[*,2] = \begin{bmatrix} -L_2\cos(\beta) \\ -L_2\sin(\beta)\cos(\alpha)\cos(\theta) - L_2\sin(\beta)\sin(\alpha)\sin(\theta) \\ L_2\sin(\beta)\cos(\alpha)\sin(\theta) - L_2\sin(\beta)\sin(\alpha)\cos(\theta) \end{bmatrix}$$

Example: Spatial (3D) Slider Crank Mechanism

Substitute $\sin(\alpha) = \frac{2x}{1+x^2}$, $\cos(\alpha) = \frac{1-x^2}{1+x^2}$ where $x = \tan(\frac{\alpha}{2})$:

$$J[*;2] = \begin{bmatrix} -L_2 \cdot \frac{1-x_3{}^2}{1+x_3{}^2} \\ -2L_2 \cdot \frac{(1-x_2{}^2)x_3(1-x_1{}^2)}{(1+x_2{}^2)(1+x_3{}^2)(1+x_1{}^2)} - 8L_2 \cdot \frac{x_2 x_1 x_3}{(1+x_2{}^2)(1+x_3{}^2)(1+x_1{}^2)} \\ 4L_2 \cdot \frac{x_2 x_3(1-x_1{}^2)}{(1+x_2{}^2)(1+x_3{}^2)(1+x_1{}^2)} - 4L_2 \cdot \frac{(1-x_2{}^2)x_3 x_1}{(1+x_2{}^2)(1+x_3{}^2)(1+x_1{}^2)} \end{bmatrix}$$

Our algorithm

Sketch of our approach

Computing the null space using LU decomposition in a hybrid symbolic-numeric fashion

1. Choose the ordering of row and column interchanges using “indicative” numerical values
2. Perform a symbolic LU decomposition of the “permuted” A without pivoting
3. Generate straight-line code to evaluate a null space basis at any setting of the parameters

Strategy for pivot selection

1. Choose “random” values $\alpha_1, \ldots, \alpha_\ell$ of parameters $z_1, \ldots, z_\ell$ from a finite subset $\mathcal{S} \subseteq \mathbb{C}$;
2. Return P, Q such that $P \cdot A(\alpha_1, \ldots, \alpha_\ell) \cdot Q$ has an LU-decomposition (without pivoting), using Gaussian Elimination with complete row/column pivoting.
   I.e., just record the row/column pivot selection.

- Good news: the probability of success is high (Schwarz-Zippel Lemma)
- Bad news: Choosing random points might be be numerically unstable...

Numerical static pivot selection

Remember: Gaussian elimination is relatively stable with complete pivoting, where we always choose the largest pivot

Strategy: Choose the "largest" pivot via random evaluations

We offer two heuristic approaches given for choosing pivot:

1. Evaluation at real values to assess the degree of the pivot function
2. Evaluations at random points off the unit circle to get an idea of coefficient size

Overall heuristic:

- Choose 4 random evaluations (2 real, 2 on unit circle)
- Perform 4 simultaneous Gaussian Eliminations, same pivoting choices
  - Choose a pivot which makes all evaluations large (or start over)

Choosing pivots in the spatial slider crank example

We perform Gaussian elimination with complete row-column pivoting simultaneously on 4 random evaluations of $A(z_1, z_2, z_3)$:

$$A(\omega_1^2, \omega_2^2, \omega_3^2) = \begin{bmatrix} 0.0 & 7.7405\text{e-}12 - 1.4447\text{e-}1i & 0.0 & 1.0 \\ \text{-}5.1923\text{e-}1 + 3.7140\text{e-}10i & 1.2421\text{-}8.6191\text{e -}10i & 3.9562\text{e-}1 - 8.7185\text{e-}2 & 0.0 \\ 3.5456\text{e-}10 + 5.3896\text{e-}1i & -8.5540\text{e-}10 - 1.19671i & -1.4832\text{e-}1 - 4.6630\text{e-}1i & 0.0 \end{bmatrix}$$

$$A(\omega_1^1, \omega_2^3, \omega_3^6) = \begin{bmatrix} 0.0 & 4.8246\text{e-}11 - 1.3143i & 0.0 & 1.0 \\ 4.7239\text{+ }1.7945\text{e-}9i & 5.0294\text{+}2.4527\text{e-}9,\ i & -4.8475 + 8.7185\text{e-}2i & 0.0 \\ -1.7148e - 9 + 4.9033i & -2.9437 + 4.8454i & -1.4832\text{e-}1 - 4.9760i & 0.0 \end{bmatrix}$$

$$A(2.0, 3.0, 4.0) = \begin{bmatrix} 0.0 & 0.2647058824 & 0.0 & 1.0 \\ -0.07411764706 & -0.1355294118 & 0.2301176471 & 0 \\ -0.2541176470 & 0.03952941175 & 0.2461176470 & 0.0 \end{bmatrix}$$

$$A(4.0, 3.0, 5.0) = \begin{bmatrix} 0.0 & 0.2769230769 & 0.0 & 1.0 \\ 0.0423529411 & -0.1140271494 & 0.1136470589 & 0 \\ -0.2736651585 & -0.01764705884 & 0.2656651585 & 0 \end{bmatrix}$$

Get the following two permutation matrices from the pivots

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

So PAQ has a strict LU decomposition, and it is numerically robust (at least at these 4 points...but heuristically most of the time)

Step 2: Generate straight-line code for the null-space

We have quickly determined permutation matrices P, Q such that

$$PAQ = LU \quad \text{where} \quad \begin{array}{l} L \in \mathbb{R}(z_1, \ldots, z_\ell)^{m\times m} \text{ lower triangular, } L_{ii} = 1 \\ U \in \mathbb{R}(z_1, \ldots, z_\ell)^{m\times n} \text{ upper triangular} \end{array}$$

- A specific null-space basis determined by last $n - r$ columns of the computed U
- Evaluated U at $\alpha_1, \ldots, \alpha_\ell$ to instantiate null-space basis
- Completely straight-line code — no decisions to make
- Procedure works with high probability: essentially when $U_{ii}(\alpha_1, \ldots, \alpha_\ell) \neq 0$, which is “almost all the time”
    - use Schwarz-Zippel Lemma to be more precise

Heuristic numerical performance

We have quickly determined permutation matrices P, Q such that

$$PAQ = LU \quad \text{where} \quad \begin{array}{l} L \in \mathbb{R}(z_1, \ldots, z_\ell)^{m\times m} \text{ lower triangular, } L_{ii} = 1 \\ U \in \mathbb{R}(z_1, \ldots, z_\ell)^{m\times n} \text{ upper triangular} \end{array}$$

- Numerically good when $U_{ii}(\alpha_1, \ldots, \alpha_\ell)$ “large enough”; these are the pivots
- When choosing the pivots, want the rational functions U_{ii} to be “large enough”
- Idea: the size of random values reflects the size of the rational function (coefficients and degree) with high probability
- Support:
  - Numerical Schwartz-Zippel – similar to Kaltofen, Yang, Zhi (2007)
  - Real evaluation in floating point – estimate degree
  - Gaussian elimination with static pivoting: Li & Demmel (1998)

Time efficiency with typical multibody models

Models	C_J imensions	No. of parameters
Planar Slider Crank	4×3	3
Planar Seven Body Mechanism	7×6	7
Quadski Turning	19×11	16
Hydraulic Stewart Platform	24×18	41

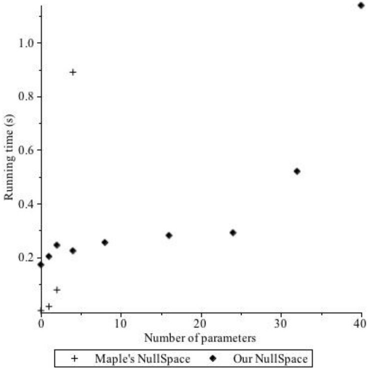
Multibody models from MapleSim

Models	Maple	Hybrid
Planar Slider Crank	0.046s	0.016s
Planar Seven Body Mechanism	0.078s	0.031s
Quadski Turning	timeout ($>$200s)	0.56s
Hydraulic Stewart Platform	timeout ($>$200s)	1.64s

Running time (in seconds)

Remember: we are only evaluating at one point (with C code)

Running time with different numbers of parameters



Running time on Hydraulic Stewart Platform with different numbers of parameters

Important advantage: we can easily instantiate more or fewer parameters, and evaluate the same nullspace.

Memory usage

Models	C_J dimensions	Size of straight-line code
Planar Slider Crank	3×4	5671
Planar Seven Body	6×7	75045
Quadski Turning	11×19	41706824
Hydraulic Stewart Platform	18×24	11849101

The final straight-line code can be greatly simplified by

- Common expression identification
- Trigonometric simplification

Example of the straight-line code for Slider-Crank Mechanism

Straight-line code for Spatial Slider-Crank Mechanism

```
t1 = pow(x[1], 0.2e1);
t2 = 0.1e1 - t1;
t1 = 0.1e1 + t1;
t3 = pow(x[2], 0.2e1);
t4 = 0.1e1 + t3;
t5 = pow(x[0], 0.2e1);
t6 = 0.1e1 - t5;
t5 = 0.1e1 + t5;
t5 = 0.1e1 / t5;
t4 = 0.1e1 / t4;
t1 = 0.1e1 / t1;
t7 = t1 * x[2] * t4 * t5;
t8 = t7 * (-0.3e1 / 0.5e1 * t2 * t6 - 0.12e2 / 0.5e1 * x[1] * x[0]);
t3 = 0.1e1 - t3;
t9 = -t2 * x[0] + x[1] * t6;
t10 = 0.3e1 / 0.5e1 * t1 * t3 * t4 * t5 * t9;
t6 = 0.6e1 / 0.5e1 * x[1] * x[0] + 0.3e1 / 0.10e2 * t2 * t6;
t11 = t1 * (-0.3e1 / 0.5e1 * t9 * t4 * t5 * t3 + x[1] / 0.5e1 - 0.3e1 / 0.25e2 * t2);
t10 = 0.1e1 / t10;
t12 = t1 * t3 * t4 * t5 * t6 * t10;
t7 = 0.6e1 / 0.5e1 * t7 * t9 - t12 * t8;
t7 = 0.1e1 / t7;
t1 = (t1 * (-t4 * t6 * t5 * t3 + t2 / 0.10e2 + 0.6e1 / 0.25e2 * x[1]) - t12 * t11) * t7;
cg2[0][0] = t10 * (t1 * t8 - t11);
cg2[1][0] = -t1;
cg2[2][0] = 1;
cg2[3][0] = -0.3e1 / 0.10e2 * t1 * t3 * t4;
```

Optimized straight-line code using Maple's CodeGeneration

Summary

- We have proposed a hybrid symbolic-numeric algorithm to compute the null space basis of a multivariate matrix.
- Our approach is significantly faster than computing null space symbolically, making it applicable to use in symbolic modelling and simulation.
- By using static pivot selection, our straight-line code for generating the null space is numerically robust at almost all parameters settings.

Future Challenges

- More robust numerical methods
  - Iterative refinement (from Li & Demmel 1998)
  - Wiser pivot selection
- Better code generation
- ...

The ultimate goal of this research