Modeling and Simulation of Lagrangian Mechanics through Automatic Differentiation and High-Index DAE Solving

Ned Nedialkov
McMaster University, Hamilton, Canada

Joint work with John Pryce, Cardiff University, UK

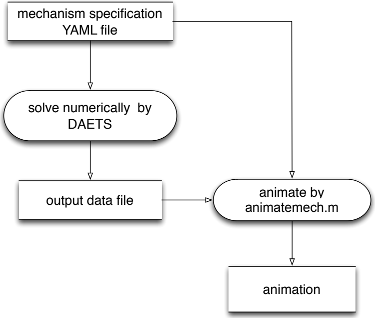
Go20 Conference on Scientific Computing and Software
Marsalforn, Gozo, Malta
May 22–26, 2023

Introduction

- A multi-body system or mechanism is made of rigid parts, springs etc., connected by joints, moving under various forces.
- John & I realized our high-index differential-algebraic equation (DAE) solver DAETS is well suited to simulating mechanisms.
- On top of DAETS we built a Lagrangian facility.
  - Using automatic differentiation (AD), from a Lagrangian function, constraints,..., it constructs at runtime Euler-Lagrange equations, an index-3 DAE, which DAETS integrates.
- We re-invented—differently—the ~ 30 years established Natural Coordinates (NCs) approach.
- Mechanism facility (in progress)
  - reads text-file spec of mechanism, initial values (IVs) etc.
  - assembles Lagrangian, constraints, ...

System "Data, Lagrangian, Action"

- Data: reads text-file spec of mechanism, IVs etc.
- Creates Lagrangian; calls DAETS to solve and write output file
- Action: visualizes by our MATLAB code animatemech



Text-file is in YAML, a human-readable data serialization language.

Examples

"Mechanism1": ▸ Animate

Andrews Mechanism: ▸ Animate

Outline

DAETS solver

Lagrangian mechanics and using DAETS on it

Rigid body tracking

Example: the RSCR mechanism

Conclusion

References

More examples

DAETS overview

- DAETS— Differential Algebraic Equations by Taylor Series
- Solves DAE initial value problems, for state variables $x_j(t),\ j = 1:n$, of the general form

$$f_i(\,t, \text{ the } x_j \text{ and derivatives of them}\,) = 0, \quad i = 1:n.$$

- Can be fully implicit.
- d/dt can appear anywhere in the expressions for f_i
  e.g. one of the equations could be

$$\frac{\left((x_1' \sin t)'\right)^2}{1 + (x_2')^2} + t^2 \cos x_2 = 0.$$

DAETS's numerical method

- For some DAE with n equations $f_i = 0$ in terms of n variables $x_j(t)$ & their derivatives, define signature matrix $\mathbf{\Sigma} = (\sigma_{ij})$

$$\sigma_{ij} = \begin{cases} \text{highest order of derivative of } x_j \text{ occurring in } f_i \\ \text{or } -\infty \text{ if doesn't occur.} \end{cases}$$

- Structural analysis (SA) derives non-negative integer offsets:
  $c_1, \ldots, c_n$ of the equations; $d_1, \ldots, d_n$ of the variables.
- Form system Jacobian $\mathbf{J} = (J_{ij})$ where

$$J_{ij} = \frac{\partial f_i}{\partial x_j^{(d_j - c_i)}}, \qquad \text{or 0 where this makes no sense.}$$

($x^{(p)} = p$th time derivative of x.)

- Mathematically (for smooth f_i)

  method succeeds at $\mathbf{x}$ $\iff$ $\mathbf{J}$ nonsingular at $\mathbf{x}$

  where "succeeds" = "generates Taylor series of solution at $\mathbf{x}$".

DAETS solves high-index problems

- Index ν measures how difficult is to solve a DAE compared to an ODE (index 0).
- For traditional methods, $\nu \geq 3$ considered hard.
- Based on structural analysis of DAE + Taylor series.
    - Builds on AD package FADBAD++.
- In principle unaffected by index.
- Finding consistent initial point is done through IPOPT, "software library for large scale nonlinear optimization".

Lagrangian mechanics

- Lagrangian method is popular because it simplifies modeling.
  - Many ways to choose coordinates; all describe same motion.
- Lagrangian function

  $$\mathcal{L} = T - V$$

  $T =$ total kinetic energy, in terms of velocities and possibly positions.
  $V =$ total potential energy, caused by conservative (energy preserving) forces depending only on system position.
- Usually have constraints on motion—assume holonomic (velocity-independent) for simplicity—and external forces.

Lagrangian cont.

- Describe configuration at time t by vector $\mathbf{q} = (q_1, \ldots, q_{n_q})$ of n_q generalized position coordinates.
  $\dot{\mathbf{q}}$ is generalized velocities.
- Assumptions from previous slide imply

$$\mathcal{L} = T - V \qquad \text{with } T = T(\mathbf{q}, \dot{\mathbf{q}}),\ V = V(\mathbf{q}),$$

  plus n_c constraints on motion:

$$0 = C_i(t, \mathbf{q}), \qquad i = 1 : n_c$$

- Assuming the C_j are independent, the system has

$$\text{DOF} = n_q - n_c \quad \text{positional degrees of freedom.}$$

  Fix DOF q_j's and DOF $\dot{q}_j$'s to specify an initial value problem.

Lagrangian cont.

- Whatever coordinates chosen, variational "stationary action" principle gives (n_q+n_c) Euler–Lagrange equations of motion:

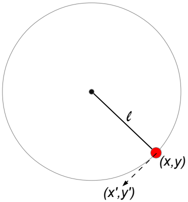
$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{q}_j} - \frac{\partial \mathcal{L}}{\partial q_j} + \sum_{i=1}^{n_c} \lambda_i \frac{\partial C_i}{\partial q_j} = Q_j(t), \qquad j = 1:n_q \qquad (1)$$

$$C_i(t, \mathbf{q}) = 0, \qquad i = 1:n_c \qquad (2)$$

- λ_i are Lagrange multipliers for the constraints. $Q_j(t)$ are generalized external force components, if any (whose definition also involves $\partial/\partial q_j$).
- If $n_c = 0$ the system is of second kind, reducible to an ODE. If $n_c > 0$ it's of first kind and is an index 3 DAE.

Example: free motion of simple pendulum

Taking $\mathbf{q} = (x, y) =$ Cartesian coordinates of pendulum bob (of mass m) with y downward, gives



$$T = \tfrac{1}{2}m(\dot{x}^2 + \dot{y}^2), \qquad V = -mgy$$

$$\mathcal{L} = \tfrac{1}{2}m(\dot{x}^2 + \dot{y}^2) + mgy,$$

with one constraint that we write

$$0 = C = \tfrac{1}{2}(x^2 + y^2 - \ell^2)$$

Euler–Lagrange, on dividing through by m, give pendulum index-3 DAE

$$\begin{aligned}
0 = A &= \ddot{x} + \lambda x && \text{from } 0 = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{x}} - \frac{\partial \mathcal{L}}{\partial x} + \lambda \frac{\partial C}{\partial x} \\
0 = B &= \ddot{y} + \lambda y - g && \text{from } 0 = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{y}} - \frac{\partial \mathcal{L}}{\partial y} + \lambda \frac{\partial C}{\partial y} \\
0 = 2C &= x^2 + y^2 - \ell^2
\end{aligned}$$

Example: how Lagrangian facility works

- User encodes

$$\begin{aligned} \mathcal{L} &= \tfrac{1}{2}m(\dot{x}^2 + \dot{y}^2) + mgy \\ C &= x^2 + y^2 - \ell^2 \end{aligned} \tag{3}$$

- Lagrangian facility converts (3) to

$$\begin{aligned} 0 &= m\ddot{x} + 2x\lambda \\ 0 &= m\ddot{y} + 2y\lambda - mg \\ 0 &= x^2 + y^2 - \ell^2 \end{aligned}$$

which is of the form DAETS accepts.

Pendulum cont.

Alternatively, taking $\mathbf{q} = (\theta)$ = angle of pendulum from downward vertical, gives

$$T = \tfrac{1}{2}m(\ell\dot{\theta})^2, \qquad V = -mg\ell\cos\theta$$
$$\mathcal{L} = \tfrac{1}{2}m(\ell\dot{\theta})^2 + mg\ell\cos\theta$$

with no constraints. Then Euler–Lagrange lead to an ODE form

$$\ddot{\theta} = -\frac{g}{\ell}\sin\theta \qquad \text{from } 0 = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial\mathcal{L}}{\partial\dot{\theta}} - \frac{\partial\mathcal{L}}{\partial\theta}$$

which is equivalent to the DAE.

For one pendulum the angle model wins, but for $n > 1$ pendula (in a chain) the Cartesian model is **much** simpler . . .

Example: $n > 1$ pendula, in 3D Cartesians

- $\mathbf{r}_i = (x_i, y_i, z_i)$ position of ith bob (with z downward)
- Generalized coordinates
  $\mathbf{q} = (\mathbf{r}_1, \ldots, \mathbf{r}_n) = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)$
- 1st kind formulation is

$$\left.\begin{aligned} \mathcal{L} &= \tfrac{1}{2} m \sum_{i=1}^{n} |\dot{\mathbf{r}}_i|^2 + mg \sum_{i=1}^{n} z_i \\ 0 = C_j &= |\mathbf{r}_j - \mathbf{r}_{j-1}|^2 - \ell^2, \qquad j = 1 : n \end{aligned}\right\} \tag{4}$$

  where $\mathbf{r}_0 = \mathbf{0}$, and $|\cdot|^2$ is the squared length of a 3-vector.
  - Constraints say the rods have length ℓ
  - $3n$ coordinate variables, n Lagrange multipliers
    Hence second-order DAE of size $4n$ and index 3
- DAETS with our Lagrangian facility solves (4) as written.

Example: The same, in ODE form

- Use spherical polar coordinates (θ_i, ϕ_i) for rod i
  - θ_i is rod's angle with downward vertical
  - ϕ_i is angle of rotation from the xz plane
- With $\mathbf{q} = (\theta_1, \phi_1, \ldots, \theta_n, \phi_n)$ we can get rid of the constraints.
- $2n$ coordinates, so $4n$ ODEs when reduced to first-order.
- Formulation is way more complex. E.g. KE is

$$T = \frac{1}{2} m\ell^2 \sum_{k=1}^{n} \left| \sum_{i=1}^{k} \begin{pmatrix} \cos\theta_i \,\dot{\theta}_i \cos\phi_i - \sin\theta_i \sin\phi_i \,\dot{\phi}_i \\ \cos\theta_i \,\dot{\theta}_i \sin\phi_i + \sin\theta_i \cos\phi_i \,\dot{\phi}_i \\ -\sin\phi_i \,\dot{\phi}_i \end{pmatrix} \right|^2$$

  and you still have the $\partial/\partial q_i, \partial/\partial \dot{q}_i$ stuff to do.
- It seems any other way to remove the constraints will use angles in some form.

Choices

Independent coordinates $n_c{=}0$	**Dependent coordinates** $n_c{>}0$
• Often most of the coordinates are angles.	NCs use Cartesian coordinates of points fixed on bodies.
• Eqns reduce to ODE, but messy.	Eqns are index-3 DAE but simple.
• "ODEs are nice to solve."	"DAEs are nasty to solve."
• Has become a big industry with some elegant maths.	NCs are a smaller industry, also with some nice maths.

DAETS is well suited to DAEs of the NCs kind (assume smooth).

- Based on structural analysis SA of the DAE.
  Theorem: SA "succeeds" on any DAE of this kind.
- Uses Taylor series of high order—typically 15–20.
- Infrastructure: IPOPT and FADBAD++.

Infrastructure benefits

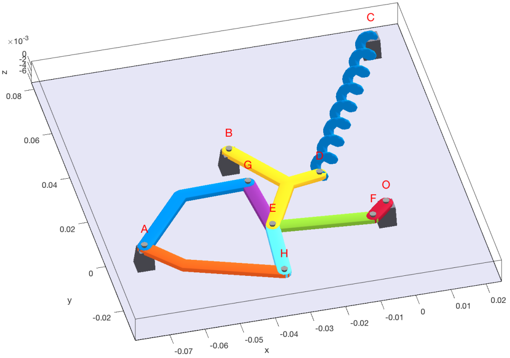
FADBAD++

- Inside DAETS forms Taylor series of solution, and system Jacobian.
- Outside DAETS, for a mechanism, forms Euler–Lagrange DAE from $\mathcal{L}$ and constraints by doing $\partial/\partial\mathbf{q}$, $\partial/\partial\,\dot{\mathbf{q}}$ and $\mathrm{d}/\mathrm{d}t$ at run time.

IPOPT

- Used by DAETS to find initial consistent point of DAE.
- For a mechanism this is initial position after fixing DOF IVs. IPOPT has proved very robust at finding this.
- Multiple solutions typical so need both fixed IVs & guesses.

E.g. in this Andrews Mechanism, G, H (independently) can lie either side of line AE.



How rigid motion is tracked

- Rigid motion of body $\mathcal{R}$ is described by "shift & rotate" map

$$\mathcal{R}_t = \mathsf{O}_t + \mathbf{Q}_t\mathcal{R}, \qquad \mathcal{R}_t \text{ is short for } \mathcal{R}(t)\text{, etc.} \qquad (*)$$

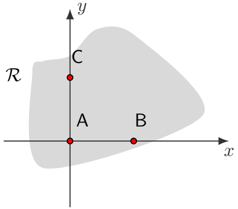
  $\mathcal{R}$: fixed in local frame
  $\mathcal{R}_t$: moving in world frame WF
  O_t: WF position of $\mathcal{R}$'s moving local origin
  $\mathbf{Q}_t$: rotation, whose columns $[\mathbf{i}_t, \mathbf{j}_t, \mathbf{k}_t]$ are $\mathcal{R}$'s moving orthonormal basis (ONB)

- (*) means that any point X and vector $\mathbf{u}$ fixed in $\mathcal{R}$ move in the WF according to

$$\mathsf{X}_t = \mathsf{O}_t + \mathbf{Q}_t\mathsf{X}, \qquad \mathbf{u}_t = \mathbf{Q}_t\mathbf{u}.$$

John–Ned tracking method

- NC methods express motion in terms of WF positions $A_t, B_t, \ldots$ of basic points or vectors (BPVs) $A, B, \ldots$ fixed on $\mathcal{R}$.
- In d dimensions, d BPVs "in general position" fix $\mathcal{R}$ uniquely.
- So in 3D we use 3 items A, B, C on $\mathcal{R}$.
  - A must be a point;
  - B, C can be either point or vector.



- "A = origin; B on +ve x axis; C in/toward upper xy plane."

- Adopt frame rule: $\mathcal{R}$'s local frame is unique coordinates s.t.

$$[\mathsf{A}, \mathsf{B}, \mathsf{C}] = \begin{bmatrix} 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \end{bmatrix}$$

and the upper triangular $\mathbf{R} = \begin{bmatrix} \times & \times \\ 0 & \times \end{bmatrix}$ has positive diagonal.

- From $\mathbf{R}$, and whether B, C is point or vector, we precompute constant matrix $\mathbf{U} = \begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \end{bmatrix}$ for each body such that

$$[\mathsf{A}_t, \mathsf{B}_t, \mathsf{C}_t]\ \mathbf{U} =: [\mathbf{i}_t, \mathbf{j}_t]$$

forms the first two vectors of $\mathcal{R}$'s moving ONB.

- Then cross product recovers the complete rotation matrix:

$$\mathbf{Q}_t = [\mathbf{i}_t, \mathbf{j}_t, \mathbf{i}_t \times \mathbf{j}_t].$$

To find $\mathcal{R}$'s KE, differentiate to get $\dot{\mathbf{Q}}_t$ and angular velocity vector $\boldsymbol{\omega}_t$, as function of $\dot{\mathsf{A}}_t, \dot{\mathsf{B}}_t, \dot{\mathsf{C}}_t$. (Omit, for time reasons.)

John–Ned versus classical NCs

- John–Ned NCs track $\mathcal{R}$'s motion by 3 BPVs, storing $3 \times 3 = 9$ scalars/body, but using nonlinear operation $\mathbf{i}_t \times \mathbf{j}_t$.
- Classical NCs track by BPVs: $4 \times 3 = 12$ scalars/body, but avoiding nonlinear operation.
  (In both cases fewer on average due to sharing BPVs between bodies.)
- Difference: nonlinearity gives us a varying mass matrix $\mathbf{M}(\mathbf{q})$, while classical NCs produce a constant $\mathbf{M}$.

  Larger but constant versus smaller but varying.
  - We have more compact models.
  - DAETS handles without difficulty time varying mass matrices.

Assembly example: the RSCR

Revolute-Spherical-Cylindrical-Revolute

A well studied 3D mechanism with 1 DOF.

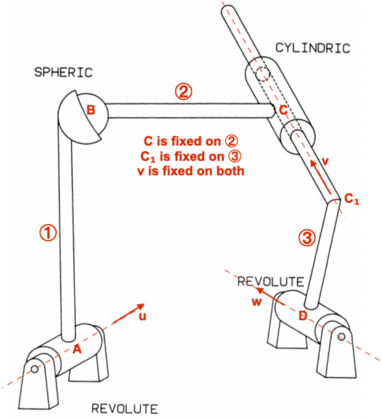
Basic points/vectors of parts:

① AB$\mathbf{u}$

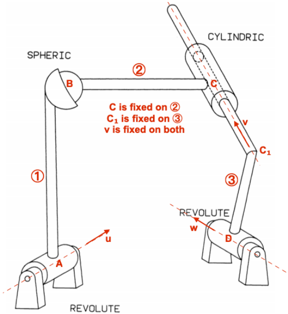
② CB$\mathbf{v}$

③ $\text{DC}_1\mathbf{w}\mathbf{v}$

The cylindric joint is made by ② and ③ sharing $\mathbf{v}$.



Without loss, assume
– angles at A, C, C_1, D are 90^o
and in WF,
– D=origin, A on –ve x axis;
– $\mathbf{w}$ in xy plane at angle δ to x axis;
– $\mathbf{u}$ at angle $\langle \alpha, \beta \rangle$ in spherical polars.
and in ③,
– $\mathbf{v}$ makes angle γ with $\mathbf{w}$.



(Design params: 3 lengths, 4 angles.)
Take $\mathbf{q} = (\mathrm{C}_1, \mathrm{B}, \mu)$, where μ is a scalar used in cylindric joint.
Total 7 scalars.

RSCR kinematics & degrees of freedom

By assignments we'll define vector $\mathbf{v}$ in ③ as function of C_1 only, then share it with ② (underscore means fixed in WF):

From q	Assign	Equate	DOF
Define ③'s frame vectors, hence $\mathbf{v}$, and ③'s rigidity			
C_1	$\mathbf{i}_3 := \mathsf{C}_1/L_3$	$0 = \mathsf{C}_1^2 - L_3^2$	$+3\ -2$
	$\mathbf{k}_3 := \mathbf{i}_3 \times \underline{\mathbf{w}}$	$0 = \mathsf{C}_1 \cdot \underline{\mathbf{w}}$	
	$\mathbf{v} := \cos(\gamma)\underline{\mathbf{w}} + \sin(\gamma)\mathbf{k}_3$		
Define cylindric joint, and ②'s rigidity			
B, μ	$\mathsf{C} := \mathsf{C}_1 + \mu\mathbf{v}$	$0 = (\mathsf{B} - \mathsf{C})^2 - L_2^2$	$+3{+}1\ -2$
		$0 = (\mathsf{B} - \mathsf{C}) \cdot \mathbf{v}$	
Define ①'s rigidity			
		$0 = (\mathsf{B} - \underline{\mathsf{A}})^2 - L_1^2$	-2
		$0 = (\mathsf{B} - \underline{\mathsf{A}}) \cdot \underline{\mathbf{u}}$	
		Net DOF:	$1 = +7\ -6$

RSCR, cont.

- To simplify dynamics let the parts be thin uniform rigid rods AB, BC, C_1D of masses m_1, m_2, m_3 (let joints be massless).

  Such a rod of mass m, moving with ends at $Y(t), Z(t)$ in WF, has KE $\frac{m}{6}(|\dot{Y}|^2 + \dot{Y}\cdot\dot{Z} + |\dot{Z}|^2)$. Then

$$\begin{aligned} T &= T_1 + T_2 + T_3 \quad (\text{Note A, D are fixed so } \dot{A} = \dot{D} = \mathbf{0}) \\ &= \tfrac{m_1}{6}(|\dot{B}|^2) + \tfrac{m_2}{6}(|\dot{B}|^2 + \dot{B}\cdot\dot{C} + |\dot{C}|^2) + \tfrac{m_3}{6}(|\dot{C}_1|^2) \\ &= \tfrac{1}{2}\,\dot{\mathbf{q}}^T \big(\underbrace{\mathbf{M}_1}_{\text{const}} + \mathbf{M}_2(\mathbf{q}) + \underbrace{\mathbf{M}_3}_{\text{const}}\big)\,\dot{\mathbf{q}}\,. \end{aligned}$$

- Dependence of $\mathbf{M}_2$ on $\mathbf{q}$ is because

$$\dot{C} = \dot{C}_1 + \mu\dot{\mathbf{v}} + \dot{\mu}\mathbf{v}$$

  is nonlinear in $\mathbf{q}$.

Compare sizes of system

- This shows length n_q of $\mathbf{q}$, and number n_c of constraints, in Lagrangian formulation of two examples, by classical "Jalon–Bayo" Natural Coordinates (JBNCs), and the Nedialkov–Pryce kind (NPNCs).
- Resulting DAE size is $n_q + n_c$, but $2n_q + n_c$ (in parentheses) in 1st order form for solvers such as DASSL.

	1 DOF RSCR		6 DOF robot	
	JBNCs	NPNCs	JBNCs	NPNCs
#coords n_q	13	7	30	18
#constraints n_c	12	6	24	12
DAE size	25 (38)	13	54 (84)	30

- Our savings are mainly from more economical frames, and use of assignments instead of equations where possible.

Conclusion

- Modelling: write Lagrangian, constraints, etc. in Cartesian coordinates.
  Much simpler than using angle coordinates and converting to an ODE.
- Since high-index DAEs are now as easy to solve as ODEs, a Lagrangian formulation needn't avoid constraints.
- So rigid-body mechanical systems can be modeled in Cartesian coordinates, which is simpler.
- This makes the concept so easy that Lagrangian stuff can be taught at undergraduate level.

- Teaching. John & I are late to this party, but:
  - Classical NC-ers have argued for years: Cartesian coordinates make MBS ideas so easy that Lagrangian stuff can be taught at undergraduate level. We agree.
  - We can add: with DAETS the DAEs are now as easy as ODEs.
- Our infrastructure should be good at important tasks such as:
  - Finding stationary equilibrium configuration.
  - Inverse dynamics: torques etc. that give a desired motion.
  - Kinematic design, e.g. make RSCR move optimally near a desired curve.

References

- JP, NN. (2020) Multibody Dynamics in Natural Coordinates through Automatic Differentiation and High-Index DAE Solving
  Acta Cybernetica 24, 315–341
- JP, NN, G. Tan, X. Li (2018) How AD can help solve differential-algebraic equations
  Optimization Methods and Software, 33:4-6, 729-749
- B. Derakhshan. Multibody Dynamics Problems in Natural Coordinates: Theory, Implementation and Simulation
  M.Sc. thesis, Computational Science and Engineering, McMaster University, 2022

- YouTube Multi-body Lagrangian Simulations
    - Outer planets
    - Gravitating masses in 2D
    - Rods, plate, springs, particles, collinears
    - RSCR Chebyshev linkage version

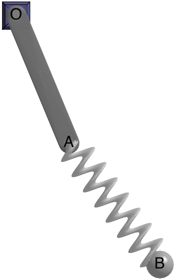
    ⋮

Example: Mechanism0

- Rod $\mathbf{OA}$ pivots at fixed point O.
- Spring $\mathbf{AB}$ attached at $\mathbf{A}$.
- Particle attached at B.
- System moves under gravity.
- Animation

Mechanism0: specification in YAML

```
Title:  Mechanism0
Dimension: 2
PhysicalParams:
  { L: 1,  M: 1,  k:  500,  l :  1,  mu: .5,  m: 2 }
PartData:
  Fixed: { O: [0,  0]  }
  Rigids:
    OA:
      Geom: [[L, 0]]  # local  frame geometry
      #      centroid ,  mass, moment of inertia
      Dyna: [[L/2, 0],  M, M*L**2/12]
  Springs:
    AB: [k,  l ,  mu] # stiffness ,  length ,  mass
  Particles :
    B:  m # mass
AppliedForces:
  Gravity:  #turns it  on with default  value
```

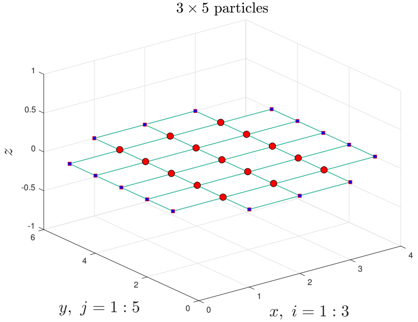


Mechanism0: IVs; solver & animation settings

```
ProblemData: # integration interval, etc.
  t0: 0
  tend: 20
  positions : { A: [L, 0], B: [(L+1), 0] } # guesses for A, B
  fixedpositions : { Ax, B } # don't change values for A.x, B
  velocities : { Ay: 10, B: [0, 0] }
SolverParams: # to guide DAETS
  Integration :
    tol : 1e-12
    order: 20
Animation: # to guide animation
  Scene:
    view: [0, 90] # camera azimuth, elevation
```

Larger example: Particle-spring system

- Rectangular grid of $m \times n$ particles connected by damped springs.
- A test for cloth simulation in movies.



- Particle (i, j) is attached to

$$(i \pm 1, j) \text{ and } (i, j \pm 1) \text{ for } i = 1:m,\ j = 1:n$$

- Index $i = 0$ or $m + 1$, resp. $j = 0$ or $n + 1$, means a fixed position.

Particle-spring cont

- Each particle (i, j)
  - coordinates $\boldsymbol{r}_{ij} = (x_{ij}, y_{ij}, z_{ij})$ full 3D motion
  - mass M
- Each spring
  - stiffness k
  - length at rest l
  - damping $k_d\times$ stretch-rate (except the boundary ones)
- Spacing Δx and Δy between particles in x and y directions.
- Initially all particles at rest in xy plane, we push the middle particle upwards.
- Animation 90×90 particles, 24,300 second-order ODEs.

Particle-spring cont

- Energy contributions

	KE	PE
Particle (i,j)	$\frac{1}{2}M\lvert\dot{\boldsymbol{r}}_{ij}\rvert^2$	Mgz_{ij}
Spring (i,j)–$(i,j+1)$		$\frac{1}{2}(\lvert\boldsymbol{r}_{i,j+1}-\boldsymbol{r}_{ij}\rvert-l)^2$

  Add up over whole mesh to get T, V and then $\mathcal{L}=T-V$.

- Spring (i,j)–$(i,j+1)$ dissipation

$$\frac{1}{2}k_d\lvert\dot{\boldsymbol{r}}_{i,j+1}-\dot{\boldsymbol{r}}_{ij}\rvert^2$$

  Add up to obtain Rayleigh dissipation function R.

- Lagrangian facility takes $\mathcal{L}$ and R and produces 2nd-order ODE of size $3mn$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial\mathcal{L}}{\partial\dot{\boldsymbol{r}}_{ij}}-\frac{\partial\mathcal{L}}{\partial\boldsymbol{r}_{ij}}+\frac{\partial R}{\partial\dot{\boldsymbol{r}}_{ij}}=0, \qquad i=1:m,\ j=1:n$$

  which DAETS integrates.