Constrained Dynamics

Chandramouli P

Indian Institute of Technology Madras

September 21, 2023

Generalized Inertia

- If reference point O^i is the centre of mass ▸ RefCoord
  - $\mathbf{F}_i^i = m^i \begin{bmatrix} \ddot{R}_x^i & \ddot{R}_y^i \end{bmatrix}^{\mathrm{T}}; \; M^i = J^i \ddot{\theta}^i$
  - Inertia force and moment
- Mass moment of inertia J^i
  - $J^i = \int_{V^i} \rho^i \bar{\mathbf{u}}^{i\mathrm{T}} \bar{\mathbf{u}}^i \; dV^i$
  - $\bar{\mathbf{u}}^i$ position vector of any point on i from O^i
- The virtual work done by the inertia forces
  - $\delta W_i^i = \mathbf{F}^{i\mathrm{T}} \delta \mathbf{R}^i + M^i \delta \theta^i$
- We can replace these by an equipollent system of forces at any other point

Equipollent Forces

- Remember that the original and equipollent systems do the same virtual work
- Let us choose a point P^i for deriving the equipollent system

  RefCoord
- With respect to the reference point O^i
  - $\mathbf{r}_P^i = \mathbf{R}^i + \mathbf{A}^i \bar{\mathbf{u}}_P^i$ and $\delta \mathbf{r}_P^i = \delta \mathbf{R}^i + \mathbf{A}_\theta^i \bar{\mathbf{u}}_P^i \delta\theta^i$
  - One can rewrite second equation as $\delta \mathbf{R}^i = \delta \mathbf{r}_P^i - \mathbf{A}_\theta^i \bar{\mathbf{u}}_P^i \delta\theta^i$
- Then the virtual work of inertia forces becomes
  - $\delta W_i^i = \mathbf{F}^{i^{\mathrm{T}}} \delta \mathbf{r}_P^i + (M^i - \mathbf{F}^{i^{\mathrm{T}}} \mathbf{A}_\theta^i \bar{\mathbf{u}}_P^i) \delta\theta^i$
  - One can show that $\mathbf{F}^{i^{\mathrm{T}}} \mathbf{A}_\theta^i \bar{\mathbf{u}}_P^i = (\mathbf{u}_P^i \times \mathbf{F}^i) \cdot \hat{\mathbf{k}}$
  - $\mathbf{u}_P^i = \mathbf{A}^i \bar{\mathbf{u}}_P^i$

Parallel Axis Theorem

- The parallel axis theorem can be obtained as a special case
- Let us assume that P^i is a fixed point
  - This implies that $\delta \mathbf{r}_P^i = 0$
- Hence $\delta W_i^i = (M^i - \mathbf{F}^{i^{\mathrm{T}}} \mathbf{A}_\theta^i \bar{\mathbf{u}}_P^i) \delta\theta^i$
- As P^i is fixed $\mathbf{F}^i = m^i \ddot{\mathbf{R}}^i = m^i \Big\{ -\ddot{\theta}^i \mathbf{A}_\theta^i \bar{\mathbf{u}}_P^i + (\dot{\theta}^i)^2 \mathbf{A}^i \bar{\mathbf{u}}_P^i \Big\}$
  - Hence $-\mathbf{F}^{i^{\mathrm{T}}} \mathbf{A}_\theta^i \bar{\mathbf{u}}_P^i = m^i (l_P^i)^2 \ddot{\theta}^i$
- Recall that $M^i = J^i \ddot{\theta}^i$ which means
  - $\delta W_i^i = \big[J^i + m^i (l_P^i)^2 \big] \ddot{\theta}^i \delta\theta^i = J_P \ddot{\theta}^i \delta\theta^i$
  - Mass moment of inertia about an arbitrary point P^i on the rigid body is equal to mass moment of inertia about centre of mass plus product of mass and square of the distance between P^i and centre of mass

Mass Matrix

- The kinetic energy of the body T^i
  - $T^i = \frac{1}{2}\int_{V^i} \rho^i \dot{\mathbf{r}}^{i^{\mathrm{T}}} \dot{\mathbf{r}}^i \, dV^i$
  - Now we can write $\dot{\mathbf{r}}^i = \begin{bmatrix} \mathbf{I} & \mathbf{A}^i_\theta \bar{\mathbf{u}}^i_P \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{R}}^i \\ \dot{\theta}^i \end{Bmatrix}$
- Using this the kinetic energy becomes

$$
\begin{aligned}
T^i &= \frac{1}{2}\int_{V^i} \rho^i \begin{bmatrix} \dot{\mathbf{R}}^{i^T} & \dot{\theta}^{i^T} \end{bmatrix} \begin{Bmatrix} \mathbf{I}^T \\ \bar{\mathbf{u}}^{i^T} \mathbf{A}^{i^T}_\theta \end{Bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^i_\theta \bar{\mathbf{u}}^i \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{R}}^i \\ \dot{\theta}^i \end{Bmatrix} dV^i \\
&= \frac{1}{2}\begin{bmatrix} \dot{\mathbf{R}}^{i^T} & \dot{\theta}^{i^T} \end{bmatrix} \left\{ \int_{V^i} \rho^i \begin{bmatrix} \mathbf{I} & \mathbf{A}^i_\theta \bar{\mathbf{u}}^i \\ \bar{\mathbf{u}}^{i^T} \mathbf{A}^{i^T}_\theta & \bar{\mathbf{u}}^{i^T} \bar{\mathbf{u}}^i \end{bmatrix} dV^i \right\} \begin{Bmatrix} \dot{\mathbf{R}}^i \\ \dot{\theta}^i \end{Bmatrix} \\
&= \frac{1}{2}\dot{\mathbf{q}}^{i^T} \mathbf{M}^i \dot{\mathbf{q}}^i
\end{aligned}
$$

Mass Matrix ..2

- The mass matrix elements

$$\mathbf{M}^i = \begin{bmatrix} \mathbf{m}^i_{RR} & \mathbf{m}^i_{R\theta} \\ \mathbf{m}^{i\,\mathrm{T}}_{R\theta} & m^i_{\theta\theta} \end{bmatrix}$$

$$\mathbf{m}^i_{RR} = \int_{V^i} \rho^i \mathbf{I} \, dv^i = m^i \mathbf{I}$$

$$\mathbf{m}^i_{R\theta} = \mathbf{A}^i_\theta \int_{V^i} \rho^i \bar{\mathbf{u}}^i \, dV^i$$

$$m^i_{\theta\theta} = \int_{V^i} \rho^i \bar{\mathbf{u}}^{i\,\mathrm{T}} \bar{\mathbf{u}}^i \, dV^i$$

- Note that $m_{\theta\theta}$ is a scalar and represents the mass moment of inertia
- The $\mathbf{m}^i_{R\theta}$ matrix represents inertia coupling between translation and rotation of body
  - $\mathbf{m}^i_{R\theta} = \mathbf{0}$ if body co-ordinate system origin is at centre of mass

Cenrtifugal Inertia Force

- The generalized inertia forces can be obtained as
  - $\mathbf{Q}_i^i = \frac{d}{dt}\left(\frac{\partial T^i}{\partial \dot{\mathbf{q}}^i}\right)^{\mathrm{T}} - \frac{\partial T^i}{\partial \mathbf{q}^i}^{\mathrm{T}}$
- Another way will be to use the virtual work δW_i^i
  - $\delta W_i^i = \int_{V^i} \rho^i \ddot{\mathbf{r}}^{i^{\mathrm{T}}} \delta \mathbf{r}^i \, dV^i$; $\delta \mathbf{r}^i = \begin{bmatrix} \mathbf{I} & \mathbf{A}_\theta^i \bar{\mathbf{u}}^i \end{bmatrix} \begin{Bmatrix} \delta \mathbf{R}^i \\ \delta \theta^i \end{Bmatrix} = \mathbf{L}^i \delta \mathbf{q}^i$
- From this we can write
  - $\dot{\mathbf{r}}^i = \mathbf{L}^i \dot{\mathbf{q}}^i ; \ddot{\mathbf{r}}^i = \mathbf{L}^i \ddot{\mathbf{q}}^i + \dot{\mathbf{L}}^i \dot{\mathbf{q}}^i$; $\dot{\mathbf{L}}^i = \begin{bmatrix} \mathbf{0} & -\dot{\theta}^i \mathbf{A}^i \bar{\mathbf{u}}^i \end{bmatrix}$
- Substitution leads to
  - $\delta W_i^i = \int_{V^i} \rho^i \ddot{\mathbf{q}}^{i^{\mathrm{T}}} \mathbf{L}^{i^{\mathrm{T}}} \mathbf{L}^i \delta \mathbf{q}^i \, dV^i + \int_{V^i} \rho^i \dot{\mathbf{q}}^{i^{\mathrm{T}}} \dot{\mathbf{L}}^{i^{\mathrm{T}}} \mathbf{L}^i \delta \mathbf{q}^i \, dV^i$

Centrifugal Force

- The first integral leads to the mass matrix
  - $$\mathbf{M}^i = \int_{V^i} \rho^i \mathbf{L}^{i^\mathrm{T}} \mathbf{L}^i \, dV^i$$
- The second integral is the centrifugal inertia force
  - $$\mathbf{Q}_v^i = -\int_{V^i} \rho^i \mathbf{L}^{i^\mathrm{T}} \dot{\mathbf{L}}^i \dot{\mathbf{q}}^i \, dV^i$$
- Then we have $\delta W_i^i = [\mathbf{M}^i \ddot{\mathbf{q}}^i - \mathbf{Q}_v^i]^\mathrm{T} \delta \mathbf{q}^i$
- It can be shown that
  - $$\mathbf{Q}_v^i = (\dot{\theta}^i)^2 \mathbf{A}^i \begin{Bmatrix} \int_{V^i} \rho^i \bar{\mathbf{u}}^i \, dV^i \\ 0 \end{Bmatrix}$$
- For the body co-ordinate system origin at centre of mass
  - $\mathbf{Q}_v^i = \mathbf{0}$

Newton-Euler Equations of Motion

- Using the principle of virtual work in dynamics
  - $\delta W_i^i - \delta W_e^i - \delta W_c^i = 0$
- We have $\delta W_e^i = \mathbf{Q}_e^{i^T} \delta \mathbf{q}^i$ and $\delta W_c^i = \mathbf{Q}_c^{i^T} \delta \mathbf{q}^i$
- From the previous slide we then have
  - $[\mathbf{M}^i \ddot{\mathbf{q}}^i - \mathbf{Q}_v^i - \mathbf{Q}_e^i - \mathbf{Q}_c^i]^{\mathrm{T}} \delta \mathbf{q}^i = 0$
- Since we have isolated a single body we can write
  - $\mathbf{M}^i \ddot{\mathbf{q}}^i = \mathbf{Q}_v^i + \mathbf{Q}_e^i + \mathbf{Q}_c^i$
- For the origin at centre of mass this becomes

$$\boxed{\begin{bmatrix} m^i \mathbf{I} & \mathbf{0} \\ \mathbf{0} & J^i \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{R}}^i \\ \ddot{\theta}^i \end{Bmatrix} = \begin{Bmatrix} (\mathbf{Q}_e^i)_R \\ (\mathbf{Q}_e^i)_\theta \end{Bmatrix} + \begin{Bmatrix} (\mathbf{Q}_c^i)_R \\ (\mathbf{Q}_c^i)_\theta \end{Bmatrix}}$$

Multi-body System

- When we have n_b interconnected bodies
  - $\mathbf{M}^i\ddot{\mathbf{q}}^i = \mathbf{Q}_e^i + \mathbf{Q}_c^i;\ i = 1, 2, \ldots, n_b$
- We can combine all these and write it in a form
  - $\mathbf{M}\ddot{\mathbf{q}} = \mathbf{Q}_e + \mathbf{Q}_c$
- The structure is

$$\begin{bmatrix} \mathbf{M}^1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{M}^{n_b} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}}^1 \\ \ddot{\mathbf{q}}^2 \\ \vdots \\ \ddot{\mathbf{q}}^{n_b} \end{Bmatrix} = \begin{Bmatrix} \mathbf{Q}_e^1 \\ \mathbf{Q}_e^2 \\ \vdots \\ \mathbf{Q}_e^{n_b} \end{Bmatrix} + \begin{Bmatrix} \mathbf{Q}_c^1 \\ \mathbf{Q}_c^2 \\ \vdots \\ \mathbf{Q}_c^{n_b} \end{Bmatrix}$$

Constraint Force Elimination

- For a system of n_b connected rigid bodies
  - $$\sum_{i=1}^{n_b}\delta W_i^i = \sum_{i=1}^{n_b}\delta W_c^i + \sum_{i=1}^{n_b}\delta W_e^i$$
- As before for frictionless constraint forces we have
  - $$\sum_{i=1}^{n_b}\delta W_c^i = 0$$
- This then implies that
  - $$\sum_{i=1}^{n_b}\{\mathbf{M}^i\ddot{\mathbf{q}}^i - \mathbf{Q}_e^i\}^{\mathrm{T}}\delta\mathbf{q}^i = 0$$
- The above equation can be re-written in the form

$$\begin{Bmatrix} \delta\mathbf{q}^1 \\ \delta\mathbf{q}^2 \\ \vdots \\ \delta\mathbf{q}^{n_b} \end{Bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} \mathbf{M}^1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{M}^{n_b} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}}^1 \\ \ddot{\mathbf{q}}^2 \\ \vdots \\ \ddot{\mathbf{q}}^{n_b} \end{Bmatrix} - \begin{Bmatrix} \mathbf{Q}_e^1 \\ \mathbf{Q}_e^2 \\ \vdots \\ \mathbf{Q}_e^{n_b} \end{Bmatrix} \right) = 0$$

- Since the $\mathbf{q}$ are all not independent one cannot write
  - $\mathbf{M}\ddot{\mathbf{q}} - \mathbf{Q}_e = \mathbf{0}$
- We now try to partition $\mathbf{q}$ into dependent co-ordinates $\mathbf{q}_d$ and independent co-ordinates $\mathbf{q}_i$
- The constraint equations are $\mathbf{C}(\mathbf{q}, t) = \mathbf{0}$

Partition Co-ordinates

- For a virtual displacement $\delta\mathbf{q}$ we have
  - $\mathbf{C_q}\delta\mathbf{q} = \mathbf{0}$; $\delta\mathbf{q} = \begin{bmatrix}\delta\mathbf{q}_d & \delta\mathbf{q}_i\end{bmatrix}^{\mathrm{T}}$
- Now $\mathbf{q}_d$ is of dimension n_c with $\mathbf{q}_i$ of dimension $n - n_c$
- The virtual displacement equation above can be re-written as
  - $\mathbf{C}_{\mathbf{q}_d}\delta\mathbf{q}_d + \mathbf{C}_{\mathbf{q}_i}\delta\mathbf{q}_i = \mathbf{0}$ or $\delta\mathbf{q}_d = -\mathbf{C}_{\mathbf{q}_d}^{-1}\mathbf{C}_{\mathbf{q}_i}\delta\mathbf{q}_i$
- Using this we can now write
  - $\delta\mathbf{q} = \mathbf{B}_i\delta\mathbf{q}_i$ with $\mathbf{B}_i = \begin{Bmatrix}-\mathbf{C}_{\mathbf{q}_d}^{-1}\mathbf{C}_{\mathbf{q}_i} \\ \mathbf{I}\end{Bmatrix}$

Embedding Technique

- So the original equation $\delta \mathbf{q}^{\mathrm{T}}\{\mathbf{M}\ddot{\mathbf{q}} - \mathbf{Q}_e\} = \mathbf{0}$ can be written as
  - $\delta \mathbf{q}_i^{\mathrm{T}} \mathbf{B}_i^{\mathrm{T}}\{\mathbf{M}\ddot{\mathbf{q}} - \mathbf{Q}_e\} = \mathbf{0}$ or $\mathbf{B}_i^{\mathrm{T}}\mathbf{M}\ddot{\mathbf{q}} - \mathbf{B}_i^{\mathrm{T}}\mathbf{Q}_e = \mathbf{0}$
- Remember that the acceleration form of the constraint equations
  - $\mathbf{C_q}\ddot{\mathbf{q}} = \mathbf{Q}_d = -(\mathbf{C_q}\dot{\mathbf{q}})_{\mathbf{q}}\dot{\mathbf{q}} - 2\mathbf{C}_{\mathbf{q}t}\dot{\mathbf{q}} - \mathbf{C}_{tt}$
- Partitioning into dependent and independent variables
  - $\ddot{\mathbf{q}}_d = -\mathbf{C}_{\mathbf{q}d}^{-1}\mathbf{C}_{\mathbf{q}i}\ddot{\mathbf{q}}_i + \mathbf{C}_{\mathbf{q}d}^{-1}\mathbf{Q}_d$
- From this we can write
  - $\ddot{\mathbf{q}} = \mathbf{B}_i\ddot{\mathbf{q}}_i + \boldsymbol{\gamma}$; $\boldsymbol{\gamma} = \begin{Bmatrix} \mathbf{C}_{\mathbf{q}d}^{-1}\mathbf{Q}_d \\ \mathbf{0} \end{Bmatrix}$

Final Form

- $\boxed{\mathbf{B}_i^{\mathrm{T}}\mathbf{M}\mathbf{B}_i\ddot{\mathbf{q}}_i = \mathbf{B}_i^{\mathrm{T}}\mathbf{Q}_e - \mathbf{B}_i^{\mathrm{T}}\mathbf{M}\boldsymbol{\gamma}}$
- How does one automatically identify the $\mathbf{q}_i$?
- The Jacobian matrix $\mathbf{C_q}$ is of size $n_c \times n$
- Gaussian Elimination with partial or full pivoting leads to a form

$$\begin{bmatrix} 1 & C_{12} & \cdots & C_{1n_c} & C_{1(n_c+1)} & \cdots & C_{1n} \\ 0 & 1 & \cdots & C_{2n_c} & C_{2(n_c+1)} & \cdots & C_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & C_{n_c(n_c+1)} & \cdots & C_{n_cn} \end{bmatrix} \begin{Bmatrix} \delta u_1 \\ \delta u_2 \\ \vdots \\ \delta u_{n_c} \\ \delta v_1 \\ \vdots \\ \delta v_{n-n_c} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{Bmatrix}$$

Automatic Determination

- The form of the previous equation is
  - $\mathbf{U}\delta\mathbf{u} + \mathbf{V}\delta\mathbf{v} = \mathbf{0}$
- The orginal co-ordinates have been re-ordered in $\delta\mathbf{u}$ and $\delta\mathbf{v}$
- Note that $\mathbf{U}$ is an upper triangular matrix with diagonal entries of 1
  - Will be non-singular as long as there are no redundant co-ordinates
- The vector $\delta\mathbf{u} = \delta\mathbf{q}_d$ and $\delta\mathbf{v} = \delta\mathbf{q}_i$

Implementation: Embedded Formulation

- Involves numerical integration of differential equations only
  - Independent co-ordinates only involved
  - Strongly coupled and smaller in size
- The dependent velocities and co-ordinates found from kinematic constraint equations
  - $\dot{\mathbf{q}}_d = -(\mathbf{C}_{\mathbf{q}d})^{-1}\left\{\mathbf{C}_t + \mathbf{C}_{\mathbf{q}i}\dot{\mathbf{q}}_i\right\}$
  - $\mathbf{q}_d = -(\mathbf{C}_{\mathbf{q}d})^{-1}\mathbf{C}_{\mathbf{q}i}\mathbf{q}_i$

Alternate Idea

- Estimate initial conditions for the initial configuration of the multi-body system
  - Both positions and velocities to be a good approximation
- Construct Jacobian matrix $\mathbf{C_q}$ using initial co-ordinates
  - $\mathbf{LU}$ factorization to identify independent co-ordinates
- Constraint equations are non-linear system of equations in dependent co-ordinates
  - Assuming the initial values of independent co-ordinates known
  - Newton-Raphson iteration used
- Above process done without need for decomposition of $\mathbf{C_q}$ into $\mathbf{C}_{\mathbf{q}d}$ and $\mathbf{C}_{\mathbf{q}i}$

Newton-Iteration

- The Newton iteration is written in the following form
  - $$\begin{Bmatrix} \mathbf{C_q} \\ \mathbf{I}_d \end{Bmatrix} \Delta \mathbf{q} = \begin{Bmatrix} -\mathbf{C} \\ \mathbf{0} \end{Bmatrix}$$
  - $\Delta \mathbf{q}$ are the Newton updates
  - $\mathbf{I}_d$ is a Boolean matrix of size $(N - N_c) \times N$
    - Entry of 1 in each row corresponding to independent co-ordinate
    - All other entries are 0 in that row
- We are basically driving $\Delta \mathbf{q}_i$ to be zero
- Above form a square matrix of size $N \times N$ and sparse
  - Sparse solvers can be used
  - No need to decompose $\mathbf{C_q}$

Next Steps

- From the solution of the non-linear algebraic equations all co-ordinates are obtained
- From this one can obtain all N velocities
  - $\begin{Bmatrix} \mathbf{C_q} \\ \mathbf{I}_d \end{Bmatrix} \dot{\mathbf{q}} = \begin{Bmatrix} -\mathbf{C}_t \\ \dot{\mathbf{q}}_i \end{Bmatrix}$
  - $\dot{\mathbf{q}}_i$ assumed to be known from numerical integration
- If the set of independent co-ordinates change during simulation one needs to simply change the location of the 1s and 0s

Numerical Integration

- Most numerical integration algorithms exist for first order ODEs of the form $\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}, t)$
- We need to re-cast the dynamic equations into state-space form
- Let us look at the simplest one Euler method
- Integrating from $t = t_0$ to $t = t_0 + h$ we have
  - $\mathbf{y}(t_0 + h) = \mathbf{y}(t_0) + \int_{t_0}^{t_0+h} \mathbf{f}(\mathbf{y}, t)\, dt$
- If h is small then one can approximate the above as
  - $\mathbf{y}(t_0 + h) = \mathbf{y}(t_0) + h\mathbf{f}(\mathbf{y}_0, t)$
- Can be generalized to a form
  - $\mathbf{y}(t_n + h) = \mathbf{y}(t_n) + h\mathbf{f}(\mathbf{y}_n, t_n)$

Euler's Method

- One can also arrive at this from a Taylor series
  - $\mathbf{y}(t_n + h) = \mathbf{y}(t_n) + h\frac{d\mathbf{y}}{dt}|_{t_n} + \frac{h^2}{2!}\frac{d^2\mathbf{y}}{dt^2}|_{t_n} + \frac{h^3}{3!}\frac{d^3\mathbf{y}}{dt^3}|_{t_n} + \cdots$
- This can be re-written in the form below
  - $\mathbf{y}(t_n + h) = \mathbf{y}(t_n) + h\mathbf{f}(\mathbf{y}_n, t_n) + \frac{h^2}{2!}(\frac{\partial \mathbf{f}}{\partial t} + \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\mathbf{f}) + \cdots$
- If we retain only the first term in the Taylor series we derive the Euler's method
  - This can also be looked as forward difference formula
  - The error will be of $\mathcal{O}(h)$ assuming the second derivative is bounded
  - It is called a first order method

Higher Order Methods

- Higher order methods will include more terms in the Taylor series expansion
- For example the mid-point method gives a second order scheme
  - $\mathbf{y}(t_n + h) = \mathbf{y}(t_n) + h\mathbf{f}\{y_n + \frac{1}{2}h\mathbf{f}(\mathbf{y}_n, t_n), t_n + \frac{1}{2}h\}$
- A Taylor series expansion of the function
  - $\mathbf{f}\{y_n + \frac{1}{2}h\mathbf{f}(\mathbf{y}_n, t_n), t_n + \frac{1}{2}h\} = \mathbf{f}(\mathbf{y}_n, t_n) + \frac{1}{2}h\mathbf{f}(\mathbf{y}_n, t_n)\frac{\partial \mathbf{f}}{\partial \mathbf{y}} + \frac{1}{2}h\frac{\partial \mathbf{f}}{\partial t} + \cdots$
- Substitution will show the expansion satisfies Taylor series to order h^2
  - Error will be of $\mathcal{O}(h^2)$

Generalized Second Order Schemes

- A general form for second order numerical integration schemes
  - $$\mathbf{y}(t_n + h) = \mathbf{y}(t_n) + h\Big[(1 - \tfrac{1}{2\alpha})\mathbf{f}(\mathbf{y}_n, t_n) + \tfrac{1}{2\alpha}\mathbf{f}\{\mathbf{y}_n + \alpha h \mathbf{f}(\mathbf{y}_n, t_n), t_n + \alpha h\}\Big]$$
- We have just used $\alpha = \frac{1}{2}$ to derive the mid-point method
- For $\alpha = 1$ we get the Heun's method
  - $$\mathbf{y}(t_n + h) = \mathbf{y}(t_n) + \tfrac{1}{2}h\Big[\mathbf{f}(\mathbf{y}_n, t_n) + \mathbf{f}\{\mathbf{y}_n + h\mathbf{f}(\mathbf{y}_n, t_n), t_n + h\}\Big]$$
- A choice of $\alpha = \frac{2}{3}$ gives the Ralston's method

Fourth-Order Runge-Kutta Method

$$
\begin{aligned}
\mathbf{y}(t_n + h) &= \mathbf{y}(t_n) + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \\
\mathbf{k}_1 &= \mathbf{f}(\mathbf{y}_n, t_n) \\
\mathbf{k}_2 &= \mathbf{f}(\mathbf{y}_n + \frac{h}{2}\mathbf{k}_1, t_n + \frac{h}{2}) \\
\mathbf{k}_3 &= \mathbf{f}(\mathbf{y}_n + \frac{h}{2}\mathbf{k}_2, t_n + \frac{h}{2}) \\
\mathbf{k}_4 &= \mathbf{f}(\mathbf{y}_n + h\mathbf{k}_3, t_n + h)
\end{aligned}
$$

- A Taylor series expansion will fit upto $\mathcal{O}(h^4)$
- Error for this method will be $\mathcal{O}(h^4)$
- At each time step four function evaluations of $(\mathbf{f}(\mathbf{y}, t))$ needs to be done

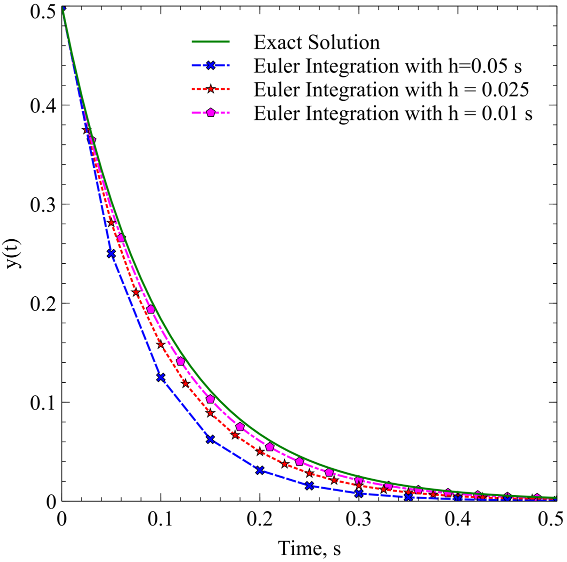
Stability

- All the methods discussed so far are explicit schemes
  - System state at next time step computed with information at current and previous time steps
  - If only current time step involved then called one step schemes
  - Else multi-step schemes
- Implicit schemes need current time step and next time step states
  - Involves iteration for solution
- Then why does one use implicit schemes at all?
  - They are numerically more stable
- We shall look at a simple example to demonstrate the idea

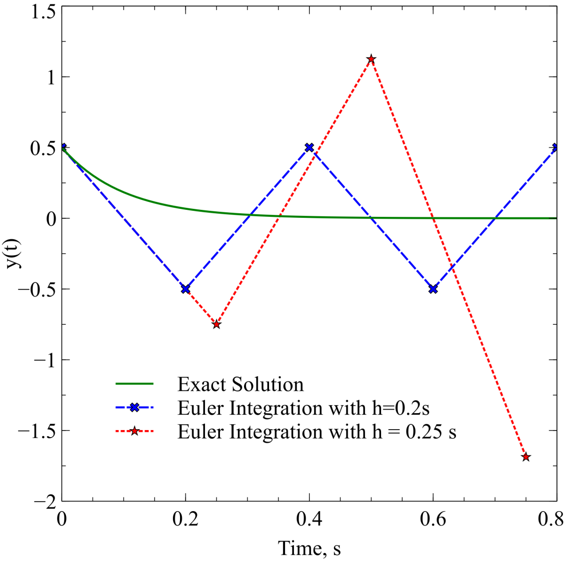
Stability for Euler Scheme

- Let us look at the numerical integration of the following equation
  - $\frac{dy}{dt} = -\alpha y;\ y(0) = y_0$ with exact solution $y(t) = y_0 e^{-\alpha t}$
- Our first-order explicit Euler method is
  - $y_{n+1} = y_n + hf(y_n, t_n) = (1 - \alpha h) y_n$
  - From the initial condition we have $y_{n+1} = (1 - \alpha h)^n y_0$
- Now we know the solution exponentially decays for positive α
- For the numerical solution to behave similarly we need
  - $|1 - \alpha h| < 1$ which leads to $h > 0$ and $h < \frac{2}{\alpha}$

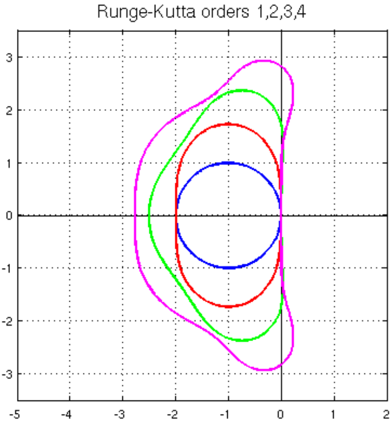
Convergence of Euler Method ($\alpha = 10$)



Instability of Euler Method ($\alpha = 10$)



λh plane RK Stability Boundaries



- Courtesy https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/23972/versions/22/previews/chebfun/examples/ode/html

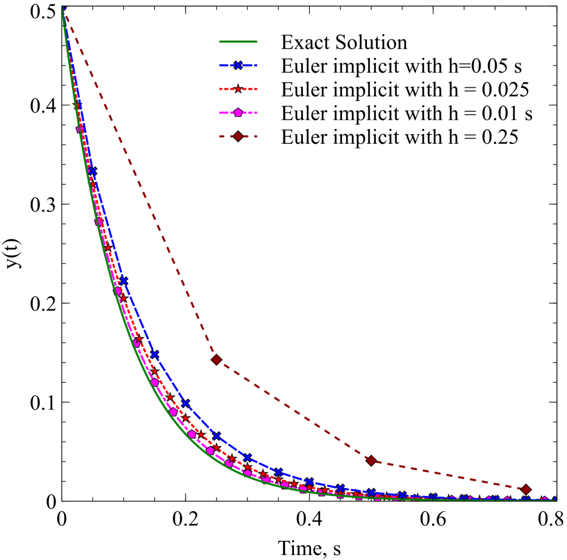
Implicit Euler Method

- $\mathbf{y}(t_n + h) = \mathbf{y}(t_n) + h\mathbf{f}(\mathbf{y}_{n+1}, t_n + h)$
- This can usually be solved by
  - Fixed-point iteration scheme
  - Newton-Raphson method
- Fixed-point iteration
  - $\mathbf{y}_{n+1}^0 = \mathbf{y}_n; \; \mathbf{y}_{n+1}^{k+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_{n+1}^k, t_{n+1}), k = 0, 1, 2, \cdots$
- Now let us look at the stability of this method through the same example
  - $\frac{dy}{dt} = -\alpha y$ with $y(0) = y_0$

Implicit Euler Stability

- The implicit Euler method for this example is
  - $y_{n+1} = y_n - \alpha h y_{n+1}$ or $y_{n+1} = \frac{1}{1+\alpha h} y_n = (\frac{1}{1+\alpha h})^n y_0$
- For $\alpha > 0$ the exact solution exponentially decays
- The numerical scheme will also decay provided
  - $|\frac{1}{1+\alpha h}| < 1$ which leads to $h > 0$ and $h < -\frac{2}{\alpha}$
  - Since $\alpha > 0$ the second condition would mean $h < 0$ which is not meaningful
  - This implies that for any $h > 0$ solution will decay
- The implicit method is called as A-stable

Implicit Euler Method ($\alpha = 10$)



Explicit Multi-Step Methods

- The Adams method is most popular and of the form
  - $$\mathbf{y}_{n+1} = \mathbf{y}_n + \sum_{i=1}^{s} \beta_i \mathbf{f}(\mathbf{y}_{n+1-i}, t_{n+1-i})$$ with $\sum_{i=1}^{s} \beta_i = 1$
  - s is the number of steps
- The solution can be written as follows
  - $$\mathbf{y}_{n+1} = \mathbf{y}_n + \int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{y}, s)\, ds$$
- In the Adams method $\mathbf{f}$ is replaced by a polynomial function
  - This is generated from knowledge of $\mathbf{f}$ at $t_{n+1-s}, t_{n+2-s}, \cdots, t_n$
  - $$\mathbf{P}_s(t) = \sum_{i=0}^{s-1} \mathbf{f}(\mathbf{y}_{n-i}, t_{n-i}) L_i(t) \; ; \quad L_i(t) = \prod_{\substack{k=0 \\ k \neq i}}^{s-1} \left(\frac{t - t_{n-k}}{t_{n-i} - t_{n-k}} \right)$$

Adams-Bashforth Method

- A 3 step Adams-Bashforth method will be
  - $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{12}\left[23\mathbf{f}(\mathbf{y}_n, t_n) - 16\mathbf{f}(\mathbf{y}_{n-1}, t_{n-1}) + 5\mathbf{f}(\mathbf{y}_{n-2}, t_{n-2})\right]$
- A s step Adams-Bashforth method is of order s
- One can see that the multi-step method is not self-starting
  - One can use a single step method for the first $s-1$ steps and then switch to the multi-step method
- At each time step only one function evaluation $(\mathbf{f}(\mathbf{y}, t))$ needs to be done

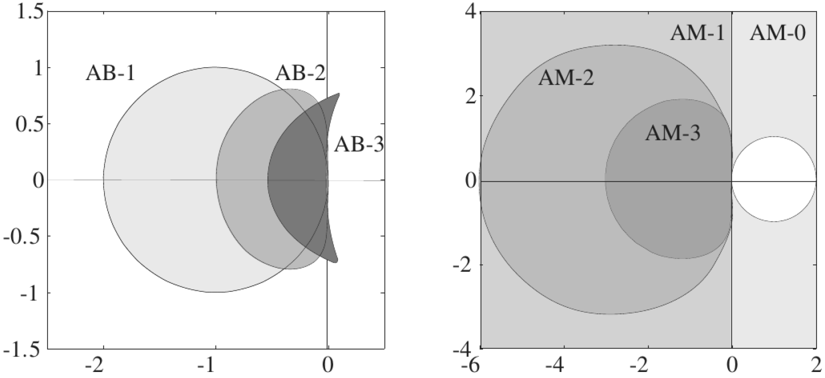
Stability of A-B methods

- Courtesy https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/23972/versions/22/previews/chebfun/examples/ode/html

Adams-Moulton Methods

- The idea of the scheme is exactly the same as the explicit scheme
  - Only difference is that it involves the next step
- For instance the two-step Adams-Moulton method
  - $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{12}\left[5\mathbf{f}(\mathbf{y}_{n+1}, t_{n+1}) + 8\mathbf{f}(\mathbf{y}_n, t_n) - \mathbf{f}(\mathbf{y}_{n-1}, t_{n-1})\right]$
- Similarly a four-step A-M method would be
  - $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{720}\left[251\mathbf{f}(\mathbf{y}_{n+1}, t_{n+1}) + 646\mathbf{f}(\mathbf{y}_n, t_n) - 264\mathbf{f}(\mathbf{y}_{n-1}, t_{n-1}) + 106\mathbf{f}(\mathbf{y}_{n-2}, t_{n-2}) - 19\mathbf{f}(\mathbf{y}_{n-3}, t_{n-3})\right]$
- Obviously one has to do a fixed-point iteration or Newton-Raphson solution
- A n step A-M method is of order $n+1$ whereas a n step A-B is of order n

A-M Stability

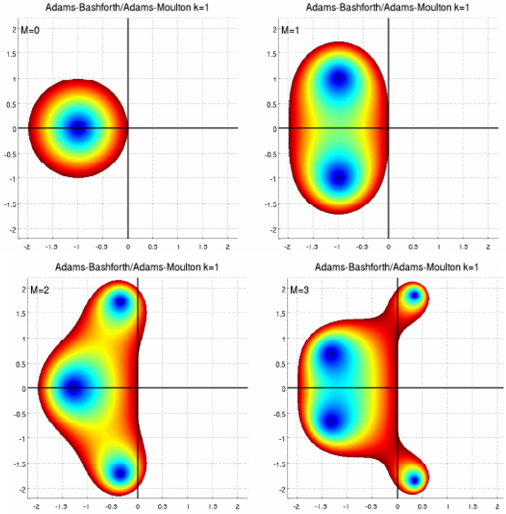


- From: http://www.maths.lth.se/na/courses/FMN081/FMN081-06/lecture23.pdf

Predictor-Corrector Methods

- Designed to reduce the iterative nature of implicit methods
- A predictor step is essentially an explicit method
  - $\hat{\mathbf{y}}_{n+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_n, t_n)$
- This estimate is now used in a corrector step based on an implicit scheme
  - $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}\left[\mathbf{f}(\mathbf{y}_n, t_n) + \mathbf{f}(\hat{\mathbf{y}}_{n+1}, t_{n+1})\right]$
- Now one can stop here or do additional corrector steps
  - $\tilde{\mathbf{y}}_{n+1} = \mathbf{y}_n + \frac{h}{2}\left[\mathbf{f}(\mathbf{y}_n, t_n) + \mathbf{f}(\hat{\mathbf{y}}_{n+1}, t_{n+1})\right]$
  - $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}\left[\mathbf{f}(\mathbf{y}_n, t_n) + \mathbf{f}(\tilde{\mathbf{y}}_{n+1}, t_{n+1})\right]$
- These are called $P(EC)^m$ schemes where P is predictor E is evaluate and C for corrector

Stability of $P(EC)^m$ schemes



- From: http://www3.math.tu-berlin.de/Vorlesungen/SS14/NumMath2/exp_StabRegionsPECE.pdf

Variable Time Step Scheme

- How does one determine the step size h?
- The computational effort is proportional to the number of time steps
  - We would want the step size to be as large as possible
  - Not too large that truncation error is big
- We want truncation error to be less than some tolerance ϵ
- Remember we are trying to solve $\dot{y} = f(y, t)$ with $y(t_0) = y_0$
  - Considering single variable for illustration purposes
  - Holds for $\mathbf{y}$

Estimate Error

- For a single step method of order p the approximate solution $\eta(t;h)$ can be written as
  - $\eta(t;h) = y(t) + h^p e_p(t) + h^{p+1} e_{p+1}(t) + \cdots$
- The error estimate is then given by
  - $e(t;h) = \eta(t;h) - y(t) = h^p e_p(t) + \mathcal{O}(h^{p+1})$
- If we take a step of size $\frac{h}{2}$ we have
  - $e(t;\frac{h}{2}) = \eta(t;\frac{h}{2}) - y(t) = (\frac{h}{2})^p e_p(t)$
- Hence one can now estimate $e_p(t)$ as simply
  - $\eta(t;h) - \eta(t;\frac{h}{2}) = e_p(t)(\frac{h}{2})^p(2^p - 1)$
  - $e_p(t)(\frac{h}{2})^p = \frac{\eta(t;h) - \eta(t;\frac{h}{2})}{(2^p - 1)}$
- For 4th order Runge-Kutta denominator is 15

Step Size Variation

- Now $e_p(t_0 + h) = \dot{e}_P(t_0)h^{p+1}$ assuming $y(t_0)$ is known
- We require $\epsilon = |\dot{e}(t_0)h^{p+1}|$
- Suppose we use a step size H to compute $\eta(t_0 + H; H)$ and $\eta(t_0 + H; \frac{H}{2})$
- From this we can derive the following
  - $\dot{e}_p(t_0) = \frac{2^p}{H^{p+1}(2^p-1)}\left[\eta(t_0 + H; H) - \eta(t_0 + H; \frac{H}{2})\right]$
- Hence we have
  - $\frac{H}{h} = \sqrt[p+1]{\frac{2^p}{2^p-1}\frac{|\eta(t_0+H;H)-\eta(t_0+H;\frac{H}{2})|}{\epsilon}}$

Step Size Control

- If $\frac{H}{h} \gg 2$ error high; replace H by $2h$ and proceed
-
- For error estimate we need to compute $\eta(t_0 + H; H)$ and $\eta(t_0 + H; \frac{H}{2})$
  - This involves 2 additional function evaluations per time step
- Instead one can use two different order methods with same h
  - $\hat{y}_{n+1} = y_n + h\mathbf{\Phi}_1(y_n, t_n; h);\ \tilde{y}_{n+1} = y_n + h\mathbf{\Phi}_2(y_n, t_n; h)$
  - $\mathbf{\Phi}_1 = \sum_{k=0}^{2} c_k f_k(y, t; h);\ \mathbf{\Phi}_2 = \sum_{k=0}^{3} \hat{c}_k f_k(y, t; h)$
  - $f_k = f(y + h\sum_{l=0}^{k-1} \beta_{kl} f_l, t + \alpha_k h)$
- $\mathbf{\Phi}_2$ order 3 while $\mathbf{\Phi}_1$ is order 2; $f_0,\ f_1$ and f_2 from $\mathbf{\Phi}_1$ used in $\mathbf{\Phi}_2$ with only one additional f_3 computed

Runge-Kutta-Fehlberg Method

k	α_k	β_{kl}			c_k	$\hat{c}_k$
		$l=0$	$l=1$	$l=2$		
0	0	0			$\frac{214}{891}$	$\frac{533}{2106}$
1	$\frac{1}{4}$	$\frac{1}{4}$			$\frac{1}{33}$	0
2	$\frac{27}{40}$	$-\frac{189}{800}$	$\frac{729}{800}$		$\frac{650}{891}$	$\frac{800}{1053}$
3	1	$\frac{214}{891}$	$\frac{1}{33}$	$\frac{650}{891}$		$-\frac{1}{78}$

Some General Comments

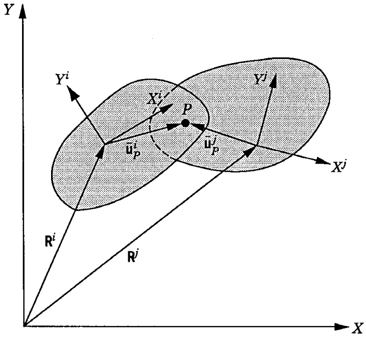
- Accuracy high for the numerical solution
  - Step-sizes are small and order of the method higher
- Stability to be good
  - Smaller step-sizes and lower order in case of multi-step methods
  - Choice of an implicit scheme over explicit scheme
- It is to be noted that Runge-Kutta methods are better
  - Higher order is more stable and hence step-sizes can be larger
  - Only negative is the number of computations compared to multi-step methods
- There are problems with extremely different time-scales
  - This can cause variable step-size methods to become computationally expensive
  - Step-size dictated by the lowest time scale or largest frequency component

Cash-Karp Method

- Developed for problems where sharp changes/discontinuites in the variables occur
- Cash and Karp in fact suggest that R-K methods are more suitable in such situations
  - The advantage is the flexibility in step-size of R-K method
- But when steep changes occur a number of steps are rejected as the error tolerance is not met
- So they have proposed a 5th order R-K but with orders 1-4 imbedded in that
  - Various coefficients required given here CashKarp

Augmented form: Lagrange Multipliers

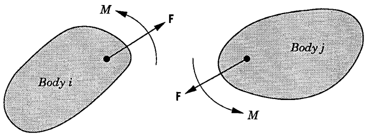
- Consider body i rigidly connected to body j at P
  - $\mathbf{R}^i + \mathbf{A}^i \bar{\mathbf{u}}_P^i - \mathbf{R}^j - \mathbf{A}^j \bar{\mathbf{u}}_P^j = \mathbf{0}$
  - $\theta^i - \theta^j = 0$
- The Jacobian matrix can be written as
  - $\mathbf{C_q} = \begin{bmatrix} \mathbf{C}_{\mathbf{q}^i} & \mathbf{C}_{\mathbf{q}^j} \end{bmatrix}$
  - $\mathbf{C}_{\mathbf{q}^i} = \begin{bmatrix} \mathbf{I} & \mathbf{A}_\theta^i \bar{\mathbf{u}}_P^i \\ \mathbf{0} & 1 \end{bmatrix}$
  - $\mathbf{C}_{\mathbf{q}^j} = -\begin{bmatrix} \mathbf{I} & \mathbf{A}_\theta^j \bar{\mathbf{u}}_P^j \\ \mathbf{0} & 1 \end{bmatrix}$



Courtesy: A. A. Shabana, 2010, *Computational Dynamics*, Third Edition, John Wiley & Sons.

Forces

- Let us define
  - $$\boldsymbol{\lambda} = -\begin{Bmatrix} \mathbf{F} \\ M \end{Bmatrix}$$
- Then the constraints force vector $\mathbf{F}^i = -\boldsymbol{\lambda}$ and $\mathbf{F}^j = \boldsymbol{\lambda}$
- The generalized constraint forces at the reference points
  - $$\mathbf{Q}_c^i = \begin{Bmatrix} \mathbf{F} \\ M + (\mathbf{A}^i \bar{\mathbf{u}}_P^i \times \mathbf{F}) \cdot \mathbf{k} \end{Bmatrix}$$
  - $$\mathbf{Q}_c^j = -\begin{Bmatrix} \mathbf{F} \\ M + (\mathbf{A}^j \bar{\mathbf{u}}_P^j \times \mathbf{F}) \cdot \mathbf{k} \end{Bmatrix}$$



Courtesy: A. A. Shabana, 2010, *Computational Dynamics*, Third Edition, John Wiley & Sons.

Final Form

- Recall that
  - $(\mathbf{A}^i \bar{\mathbf{u}}_P^i \times \mathbf{F}) \cdot \mathbf{k} = \bar{\mathbf{u}}_P^{i^{\mathrm{T}}} \mathbf{A}_\theta^{i^{\mathrm{T}}} \mathbf{F}$
  - $(\mathbf{A}^j \bar{\mathbf{u}}_P^j \times \mathbf{F}) \cdot \mathbf{k} = \bar{\mathbf{u}}_P^{j^{\mathrm{T}}} \mathbf{A}_\theta^{j^{\mathrm{T}}} \mathbf{F}$
- Hence we can write
  - $\mathbf{Q}_c^i = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \bar{\mathbf{u}}_P^{i^{\mathrm{T}}} \mathbf{A}_\theta^{i^{\mathrm{T}}} & 1 \end{bmatrix} \begin{Bmatrix} \mathbf{F} \\ M \end{Bmatrix}; \; \mathbf{Q}_c^j = - \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \bar{\mathbf{u}}_P^{j^{\mathrm{T}}} \mathbf{A}_\theta^{j^{\mathrm{T}}} & 1 \end{bmatrix} \begin{Bmatrix} \mathbf{F} \\ M \end{Bmatrix}$
- $\mathbf{Q}_c^i = -\mathbf{C}_{\mathbf{q}^i}^{\mathrm{T}} \boldsymbol{\lambda}$ and $\mathbf{Q}_c^j = -\mathbf{C}_{\mathbf{q}^j}^{\mathrm{T}} \boldsymbol{\lambda}$
- $\boldsymbol{\lambda}$ is called as Lagrange Multipliers

Revolute Joint

- Suppose body i and j connected by a revolute joint
- Then the constraint equation for this joint
  - $\mathbf{R}^i + \mathbf{A}^i \bar{\mathbf{u}}_P^i - \mathbf{R}^j - \mathbf{A}^j \bar{\mathbf{u}}_P^j = \mathbf{0}$
- The Jacobian is
  - $\mathbf{C_q} = \begin{bmatrix} \mathbf{I} & \mathbf{A}_\theta^i \bar{\mathbf{u}}_P^i & -\mathbf{I} & -\mathbf{A}_\theta^j \bar{\mathbf{u}}_P^j \end{bmatrix}$
- The generalized constraint force on body $\mathbf{Q}_c^i = -\mathbf{C}_{\mathbf{q}^i}^{\mathrm{T}} \boldsymbol{\lambda}$ and $\mathbf{Q}_c^j = -\mathbf{C}_{\mathbf{q}^j}^{\mathrm{T}} \boldsymbol{\lambda}$ are computed

Generalized Constraint Force: Body i

$$
\begin{aligned}
\mathbf{Q}_c^i &= -\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -\bar{x}_P^i \sin\theta^i - \bar{y}_P^i \cos\theta^i & \bar{x}_P^i \cos\theta^i - \bar{y}_P^i \sin\theta^i \end{bmatrix} \begin{Bmatrix} \lambda_1 \\ \lambda_2 \end{Bmatrix} \\
&= \begin{Bmatrix} -\lambda_1 \\ -\lambda_2 \\ \lambda_1(\bar{x}_P^i \sin\theta^i + \bar{y}_P^i \cos\theta^i) - \lambda_2(\bar{x}_P^i \cos\theta^i - \bar{y}_P^i \sin\theta^i) \end{Bmatrix}
\end{aligned}
$$

Generalized Constraint Force: Body j

$$\mathbf{Q}_c^j = -\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ \bar{x}_P^j \sin\theta^j + \bar{y}_P^j \cos\theta^j & -\bar{x}_P^j \cos\theta^j + \bar{y}_P^j \sin\theta^j \end{bmatrix} \begin{Bmatrix} \lambda_1 \\ \lambda_2 \end{Bmatrix}$$

$$= \begin{Bmatrix} \lambda_1 \\ \lambda_2 \\ -\lambda_1(\bar{x}_P^j \sin\theta^j + \bar{y}_P^j \cos\theta^j) + \lambda_2(\bar{x}_P^j \cos\theta^j - \bar{y}_P^j \sin\theta^j) \end{Bmatrix}$$

Multiple Joints

- Suppose body i is connected to other bodies by n_i joints
  - $\mathbf{C}_1(\mathbf{q},t) = \mathbf{0};\ \mathbf{C}_2(\mathbf{q},t) = \mathbf{0};\ \cdots;\ \mathbf{C}_{n_i}(\mathbf{q},t) = \mathbf{0}$
  - $\mathbf{q} = \begin{bmatrix} \mathbf{q}^{1^{\mathrm{T}}} & \mathbf{q}^{2^{\mathrm{T}}} & \cdots & \mathbf{q}^{n_b{}^{\mathrm{T}}} \end{bmatrix}^{\mathrm{T}}$
- The corresponding generalized constraint forces are
  - $\mathbf{Q}_1^i = -(\mathbf{C}_1)_{\mathbf{q}^i}^{\mathrm{T}} \boldsymbol{\lambda}_1;\ \mathbf{Q}_2^i = -(\mathbf{C}_2)_{\mathbf{q}^i}^{\mathrm{T}} \boldsymbol{\lambda}_2;\ \cdots;\ \mathbf{Q}_{n_i}^i = -(\mathbf{C}_{n_i})_{\mathbf{q}^i}^{\mathrm{T}} \boldsymbol{\lambda}_{n_i}$
- The overall generalized constraint force $\mathbf{Q}_c^i$
  - $$\mathbf{Q}_c^i = -\sum_{k=1}^{n_i} (\mathbf{C}_k)_{\mathbf{q}^i}^{\mathrm{T}} \boldsymbol{\lambda}_k$$
- One can expand to include all joints to write
  - $\mathbf{Q}_c^i = -\mathbf{C}_{\mathbf{q}^i}^{\mathrm{T}} \boldsymbol{\lambda}$
  - Joint reactions not associated with body i will have rows with zeros in $\mathbf{C}_{\mathbf{q}^i}$

Multi-body System

- The total vector for n_b bodies becomes
  - $\mathbf{Q}_c = \begin{bmatrix} \mathbf{Q}_c^1 & \mathbf{Q}_c^2 & \cdots & \mathbf{Q}_c^{n_b} \end{bmatrix}^{\mathrm{T}}$

$$\mathbf{Q}_c = -\begin{Bmatrix} \mathbf{C}_{\mathbf{q}^1}^{\mathrm{T}}\boldsymbol{\lambda} \\ \mathbf{C}_{\mathbf{q}^2}^{\mathrm{T}}\boldsymbol{\lambda} \\ \vdots \\ \mathbf{C}_{\mathbf{q}^{n_b}}^{\mathrm{T}}\boldsymbol{\lambda} \end{Bmatrix} = -\mathbf{C}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\lambda}$$

$$\mathbf{C}_{\mathbf{q}} = \begin{bmatrix} \mathbf{C}_{\mathbf{q}^1} & \mathbf{C}_{\mathbf{q}^2} & \cdots & \mathbf{C}_{\mathbf{q}^{n_b}} \end{bmatrix}$$

- Recall earlier we had $\mathbf{M}\ddot{\mathbf{q}} = \mathbf{Q}_e + \mathbf{Q}_c$
- This now becomes $\boxed{\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{Q}_e}$

Augmented Formulation

- Assuming that $\mathbf{Q}_e$ is known then we have n equations with $n + n_c$ unknowns
  - n accelerations $\ddot{\mathbf{q}}$ and n_c Lagrange multipliers $\boldsymbol{\lambda}$
- So we introduce the following set of equations by differentiating the constraint equations
  - $\mathbf{C_q}\ddot{\mathbf{q}} = \mathbf{Q}_d$
- This leads to the augmented form

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_\mathbf{q}^{\mathrm{T}} \\ \mathbf{C_q} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{Q}_e \\ \mathbf{Q}_d \end{Bmatrix}$$

Numerical Implementation

- Now that all the co-ordinates and velocities are available one can calculate the accelerations and Lagrange Multipliers

$$\boxed{\begin{bmatrix} \mathbf{M} & \mathbf{C}_{\mathbf{q}}^{\mathrm{T}} \\ \mathbf{C}_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{Q}_e \\ \mathbf{Q}_d \end{Bmatrix}}$$

- The Lagrange multipliers provide the generalized reaction forces
- Now we integrate forward in time
  - Using the state-space form
- This will yield the velocities and co-ordinates for the next time step
- While doing this we hold the Lagrange multiplier values fixed

Constraint Stabilization Method

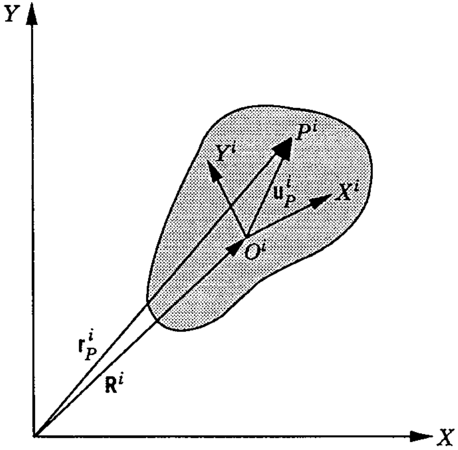
- One of the issues is the growth of the equation below with time
  - $\ddot{\mathbf{C}} = \mathbf{C}_{\mathbf{q}}\ddot{\mathbf{q}} - \mathbf{Q}_d = \mathbf{0}$
  - While at $t = 0$ this is satisfied exactly with increasing t it is not
  - In fact the form of the solution is $\mathbf{q} = \mathbf{a}_1 t + \mathbf{a}_2$
- Baumgarte proposed a stabilization scheme
  - $\ddot{\mathbf{C}} + 2\alpha\dot{\mathbf{C}} + \beta^2\mathbf{C} = \mathbf{0}$
  - Here $\alpha > 0$ and $\beta \neq 0$
  - Solution form is $\mathbf{q} = \mathbf{a}_1 e^{s_1 t} + \mathbf{a}_2 e^{s_2 t}$
- This leads to the form
  - $\mathbf{C}_{\mathbf{q}}\ddot{\mathbf{q}} = \mathbf{Q}_d - 2\alpha(\mathbf{C}_{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}_t) - \beta^2\mathbf{C}$
- This form is used in the augmented form to yield

Stabilization Scheme

$$\boxed{\begin{bmatrix} \mathbf{M} & \mathbf{C}_{\mathbf{q}}^{\mathrm{T}} \\ \mathbf{C}_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{Q}_e \\ \mathbf{Q}_d - 2\alpha(\mathbf{C}_{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}_t) - \beta^2\mathbf{C} \end{Bmatrix}}$$

- All accelerations are integrated without partitioning into dependent and independent co-ordinates
- Gives good results but
- No reliable method to choose α and β

Reference Co-ordinates



Courtesy: A. A. Shabana, 2010, *Computational Dynamics*, Third Edition, John Wiley & Sons.

GoBack GoBack2

Cash-Karp R-K Formulas

$$
\begin{array}{c|cccccc l}
0 & 0 & & & & & & \\
\frac{1}{5} & \frac{1}{5} & 0 & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & 0 & & & & \\
\frac{3}{5} & \frac{3}{10} & -\frac{9}{10} & \frac{6}{5} & 0 & & & \\
1 & -\frac{11}{54} & \frac{5}{2} & -\frac{70}{27} & \frac{35}{27} & 0 & & \\
\frac{7}{8} & \frac{1631}{55296} & \frac{175}{512} & \frac{575}{13824} & \frac{44275}{110592} & \frac{253}{4096} & 0 & \\
\hline
 & \frac{37}{378} & 0 & \frac{250}{621} & \frac{125}{594} & 0 & \frac{512}{1771} & \textbf{Order 5} \\
 & \frac{2825}{27648} & 0 & \frac{18575}{48384} & \frac{13525}{55296} & \frac{277}{14336} & \frac{1}{4} & \textbf{Order 4} \\
 & \frac{19}{54} & 0 & -\frac{10}{27} & \frac{55}{54} & 0 & 0 & \textbf{Order 3} \\
 & -\frac{3}{2} & \frac{5}{2} & 0 & 0 & 0 & 0 & \textbf{Order 2} \\
 & 1 & 0 & 0 & 0 & 0 & 0 & \textbf{Order 1}
\end{array}
\qquad (5)
$$

Courtesy: J. R. Cash and A. H. Karp, 1990, A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides, ACM Transactions on Mathematical Software, pp. 201-222 Return