“A vote is like a rifle: its usefulness depends upon the character of the user.”

— Theodore Roosevelt
Before we get started...

- On Friday, we learned:
  - Review of isoparametric elements –what it means
  - Introduce the Absolute Nodal Coordinate Formulation
  - Define constraints with ANCF finite elements and rigid bodies

- This lecture...
  - Define constraints with ANCF finite elements and rigid bodies
  - ANCF cable element kinematics
  - ANCF cable element strains
  - ANCF cable element inertia forces
  - Introduce Chrono’s bilinear ANCF shell element
Example. Crankshaft mechanism with ANCF beam

Beam has the following coordinates

- Position vector (3D)
- $r_y$ cross section plane
- $r_z$ cross section plane
- $r_y$ perpendicular to $r_z$ in the undeformed configuration

$r_i = \frac{\partial r}{\partial i}$, where $i$ is local direction of the finite element
Constraints to Rigid Bodies

Beam cross section

Revolute

- Position

\[
C(R_b^1, \theta_b^1, r_y^1, r_z^1) = R_b^1 + A(\theta_b^1) \bar{u}^P - r^1 = 0
\]

- Local vector in flexible body is parallel to local vector in rigid body: Their global direction define the mechanical joint’s axis of revolution

\[
C(\theta_b^1, r_y^1) = \left( A^1 \left( \theta_b^1 \right) \bar{f}^1 \right)^T r_y^1 = 0
\]

\[
C(\theta_b^1, r_y^1) = \left( A^1 \left( \theta_b^1 \right) \bar{g}^1 \right)^T r_y^1 = 0
\]

- Where \( \bar{g}^1 \), \( \bar{f}^1 \) are local vectors of body 1 perpendicular to joint axis
Constraints to Rigid Bodies

Problem

- Obtain same equations for prismatic joint
- Obtain Jacobian of both revolute and prismatic constraints assuming only bodies in the system are those 3 of the crankshaft (be reminded body 2 is the flexible body)
Constraints to Rigid Bodies

- Constraint relationships and their derivatives can be obtained also for flexible body/flexible body constraints
- Key to defining these joints is
  - Identify what flexible body coordinates represent. E.g. Do they define a fiber orientation? Do they contain a vector perpendicular to the beam cross section?
- Constraints to ANCF bodies also define boundary conditions. These constraints are not straightforward to come up with because of the relations between them and the strains, which often define the boundary conditions.
ANCF Cable: Virtual Work of Elastic Forces

- Have one position vector gradient pointing along the beam centerline
- Account for axial and bending strains
- Nodal coordinates

\[ \mathbf{q}_j(t) = \begin{bmatrix} \mathbf{r}_j^T & \mathbf{r}_{j,x}^T \end{bmatrix}^T \]

- Shape functions: \( \xi = x/l \), \( \xi = 0, \ldots, 1 \)

\[
\begin{align*}
  s_1 &= 1 - 3\xi^2 + 2\xi^3 \\
  s_2 &= l \left( \xi - 2\xi^2 + \xi^3 \right) \\
  s_3 &= 3\xi^2 - 2\xi^3 \\
  s_4 &= l \left( -\xi^2 + \xi^3 \right)
\end{align*}
\]

Position field of an ANCF beam element

\[ \mathbf{r} = \begin{bmatrix} s_1 \mathbf{I}_{3 \times 3} & s_2 \mathbf{I}_{3 \times 3} & s_3 \mathbf{I}_{3 \times 3} & s_4 \mathbf{I}_{3 \times 3} \end{bmatrix} \begin{bmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \end{bmatrix} = \mathbf{S}(\xi)\mathbf{q} \]

Beam longitudinal position vector gradient

\[ \mathbf{r}_x = \mathbf{S}_x \left( \xi \right) \mathbf{q} \]
The virtual of the elastic forces for the gradient-deficient beam element may be defined as

\[
\delta W_e = \int_L \left[ EA \varepsilon_x \delta \varepsilon_x + EI \kappa \delta \kappa \right] dx
\]

where

\[
\varepsilon_x = \frac{1}{2} (\mathbf{r}_x^T \mathbf{r}_x - 1) \quad \text{and} \quad \kappa = \frac{\left| \mathbf{r}_x \times \mathbf{r}_{xx} \right|}{\left| \mathbf{r}_x \right|^3}
\]

are the axial Green-Lagrange strain and bending curvature.

- Calculation of strain variations for virtual work

\[
\delta \varepsilon_x = \frac{\partial}{\partial \mathbf{e}} \left( \frac{1}{2} (\mathbf{r}_x^T \mathbf{r}_x - 1) \right) \delta \mathbf{e} = \mathbf{r}_x^T \frac{\partial \mathbf{r}_x}{\partial \mathbf{e}} \delta \mathbf{e}
\]

- For bending curvature:

\[
\delta \kappa = \frac{\partial \kappa}{\partial \mathbf{e}} \delta \mathbf{e} = \frac{1}{g^2} \left( g \frac{\partial f}{\partial \mathbf{e}} - f \frac{\partial g}{\partial \mathbf{e}} \right) \delta \mathbf{e}
\]

\[
\frac{\partial f}{\partial \mathbf{e}} = \frac{\partial}{\partial \mathbf{e}} \sqrt{\left( \mathbf{r}_x \times \mathbf{r}_{xx} \right)^T \left( \mathbf{r}_x \times \mathbf{r}_{xx} \right)} = \frac{\left( \mathbf{r}_x \times \mathbf{r}_{xx} \right)^T \left( \frac{\partial}{\partial \mathbf{e}} \mathbf{r}_x \times \mathbf{r}_{xx} + \mathbf{r}_x \times \frac{\partial}{\partial \mathbf{e}} \mathbf{r}_{xx} \right)}{\sqrt{\left( \mathbf{r}_x \times \mathbf{r}_{xx} \right)^T \left( \mathbf{r}_x \times \mathbf{r}_{xx} \right)}}
\]

\[
\frac{\partial g}{\partial \mathbf{e}} = \frac{\partial}{\partial \mathbf{e}} \left( \mathbf{r}_x^T \mathbf{r}_x \right)^{\frac{3}{2}} = 3 \left( \mathbf{r}_x^T \mathbf{r}_x \right)^{\frac{1}{2}} \left( \mathbf{r}_x^T \frac{\partial \mathbf{r}_x}{\partial \mathbf{e}} \right)
\]

- Quiz: How many bending curvatures does a 3D beam have?
The generalized internal force may be written as...

\[ Q^i_e \big|_{12 \times 1} = Q^i_{e,b} \big|_{12 \times 1} + Q^i_{e,a} \big|_{12 \times 1} = \]

\[ \int E I \frac{1}{g} \left( g \frac{\partial f}{\partial \epsilon} - f \frac{\partial g}{\partial \epsilon} \right) dx + \int E A 3 \left( \frac{r^T_x r_x}{2} \frac{\partial^2 r_x}{\partial e_x} \right) dx \]

Mass matrix easily obtained from kinetic energy

\[ T = \frac{1}{2} \int_V \rho \dot{r}^T \dot{r} \, dV = \frac{1}{2} \dot{e}^T M \ddot{e}, \]

where \( M_{12 \times 12} = \rho A \int_V \rho S^T S \, dx = \text{constant} \)

Gravity forces

\[ \delta W_g = F_g \cdot \delta r = g \int_V \rho \, dV \cdot S \delta e = g m \cdot S \delta e \]

\[ Q_g = S^T (mg) \]

Equations of motion after assembling finite elements

\[ M \ddot{e} = Q_e + Q_g \]
ANCF Fully Parameterized Beam

- Fully parameterized beam: The most straightforward beam element
- 2 nodes; one position vector and three position vector gradients: x, y, and z
- An elastic line approach has been used to alleviate Poisson locking (we’ll see more in next classes)
- It allows for 3D definition of elastic forces
  - Severe locking; i.e. bad convergence properties
    - Correct results not achieved even with fine discretization or
    - Need many elements to obtain correct results

Interpolation:

- Linear in y and z (cross section plane)
  - Poisson locking: Poisson effect
- Cubic in longitudinal direction
  - Shear locking
- FE locking: Excessive (unwanted) stiffness in some FE deformation modes

\[ r^P(\xi, \eta, \zeta) = S(\xi, \eta, \zeta) \begin{bmatrix} r^j_x & r^j_y & r^j_z & r^{j+1}_x & r^{j+1}_y & r^{j+1}_z \end{bmatrix}^T \]
Shear deformable ANCF beam

- Avoid locking issues
- Can use structural (Reissner) and continuum-based approaches
- Describe two bending strains, two shears, torsion, and stretch
Shear deformable ANCF beam

- Quadratic shape functions in longitudinal direction
- Linear interpolation over the cross section
- Reference of SF taken at the center of the element – node 3.

\[ r = \begin{bmatrix} s_1 I & s_2 I & \ldots & s_9 I \end{bmatrix} \]

\[ s_1 = -\frac{\xi}{2}(1 - \xi), \quad s_2 = \eta s_1, \quad s_3 = \zeta s_1 \text{ [First node]} \]

\[ s_4 = \frac{\xi}{2}(1 + \xi), \quad s_5 = \eta s_4, \quad s_6 = \zeta s_4 \text{ [Second node]} \]

\[ s_7 = -(\xi - 1)(\xi + 1), \quad s_8 = \eta s_7, \quad s_9 = \zeta s_7 \text{ [Third node]} \]
Shear deformable ANCF beam: Structural Mechanics

1) Create a coord. syst. at cross section

\[ e_1 = \frac{e_1}{|e_1|}, \quad e_1 = r_y \times r_z; \quad e_3 = \frac{e_3}{|e_3|}, \quad e_3 = r_z; \quad e_2 = \frac{e_2}{|e_2|}, \quad e_2 = r_y \times (r_y \times r_z); \]

\[ A_{cs} = [e_1 \quad e_2 \quad e_3] \]

2) Stretch and shear defined as

\[ \Gamma_1 = e_1^T r_x - 1, \quad \Gamma_2 = e_2^T r_x, \quad \Gamma_3 = e_3^T r_x, \quad r_x = \frac{\partial r}{\partial x} \]

3) Bending and torsion

\[ k = A^T A'_0 = \begin{bmatrix} 0 & -\kappa_3 & \kappa_2 \\ \kappa_3 & 0 & -\kappa_1 \\ -\kappa_2 & \kappa_1 & 0 \end{bmatrix}, \quad k = \text{axial}(k) \]

Using SM approach, the elastic energy of the element is

\[ U^{SM} = \frac{1}{2} \frac{L}{2} \int_{-\frac{L}{2}}^{\frac{L}{2}} \Gamma^T \text{diag}(EA, GAk_2, GAk_3) \Gamma + k^T \text{diag}(GJ, EI_2, EI_3) k \, d\xi \]

Stretch and shear

Bending and torsion
Shear deformable ANCF beam: Structural Mechanics

- Previous strain measures must be objective (they are!)
- Inertia matrix calculation is straightforward
- Generalized internal force must account for cross section area frame: Not as easy as CB approach

- This element has been validated for:
  - Small deformation (analytical solution)
  - Large deformation: Torsional moment (180 deg. twist)
  - Eigenfrequencies: Analytical solution of Timoshenko beam
- It has more coordinates. Additional cross section elastic energy must be introduced (not accounted for previously)
Shear deformable ANCF beam: Continuum-Based

The work of elastic forces can be derived from nonlinear continuum mechanics, using the relation between the nonlinear Green-Lagrange strain tensor and the second Piola-Kirchhoff stress tensor. The deformation gradient is defined as

\[ \mathbf{F} = \frac{\partial \mathbf{r}}{\partial \mathbf{r}_0} = \frac{\partial \mathbf{r}}{\partial \xi} \frac{\partial \xi}{\partial \mathbf{r}_0} = \begin{bmatrix} \frac{\partial r_1}{\partial \xi} & \frac{\partial r_1}{\partial \eta} & \frac{\partial r_1}{\partial \zeta} \\ \frac{\partial r_2}{\partial \xi} & \frac{\partial r_2}{\partial \eta} & \frac{\partial r_2}{\partial \zeta} \\ \frac{\partial r_3}{\partial \xi} & \frac{\partial r_3}{\partial \eta} & \frac{\partial r_3}{\partial \zeta} \end{bmatrix} \begin{bmatrix} \frac{\partial r_{01}}{\partial \xi} & \frac{\partial r_{01}}{\partial \eta} & \frac{\partial r_{01}}{\partial \zeta} \\ \frac{\partial r_{02}}{\partial \xi} & \frac{\partial r_{02}}{\partial \eta} & \frac{\partial r_{02}}{\partial \zeta} \\ \frac{\partial r_{03}}{\partial \xi} & \frac{\partial r_{03}}{\partial \eta} & \frac{\partial r_{03}}{\partial \zeta} \end{bmatrix}^{-1} \]
Shear deformable ANCF beam: Continuum-Based

Possibly distorted reference, captured by \( J = \frac{\partial r_0}{\partial \xi} \).

The strain strain relation is given by \( \sigma = D\varepsilon \) where \( D \) is the elastic matrix and \( \varepsilon \) the Green-Lagrange strain vector.

The elasticity matrix is given (for isotropic materials), as:

\[
D = \frac{E\nu}{(1+\nu)(1-2\nu)} \begin{bmatrix}
\frac{1-\nu}{\nu} & 1 & 1 & 0 & 0 & 0 \\
1 & \frac{1-\nu}{\nu} & 1 & 0 & 0 & 0 \\
1 & 1 & \frac{1-\nu}{\nu} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1-2\nu}{2\nu} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1-2\nu}{2\nu} & k_2 \\
0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2\nu} k_3
\end{bmatrix}
\]

Become shear modulus \( G \), where theory of thick beams (Timoshenko) has been assumed \(-k_2\) and \( k_3 \) are Timoshenko shear correction factors, dependent on beam cross section shape and material properties.
Shear deformable ANCF beam: Continuum-Based

The elastic energy, to be used to obtain generalized internal forces, may then be written as:

$$U^{CB} = \frac{1}{2} \int_{-L/2}^{L/2} \int_{-H/2}^{H/2} \int_{-w/2}^{w/2} \varepsilon^T \sigma \det(J) \, d\xi \, d\eta \, d\zeta$$

Poisson ratio $\nu$ couples $\varepsilon_{xx}$ with $\varepsilon_{yy}$ and $\varepsilon_{zz}$. This coupling if integrated over the volume of this element causes unwanted stiffness of the bending mode—that is, locking. To avoid this, one type of selective integration is used:

$$U^{CB}_{SI} = \frac{1}{2} \int_{-L/2}^{L/2} \int_{-H/2}^{H/2} \int_{-w/2}^{w/2} \varepsilon^T \mathbf{D}^0 \varepsilon \det(J) \, d\xi \, d\eta \, d\zeta + \frac{1}{2} H W \int_{-L/2}^{L/2} \varepsilon^T \mathbf{D}^\nu \varepsilon \det(J) \, d\xi$$

In which:

$$\mathbf{D} = \underbrace{\mathbf{D}^0}_{\text{Matrix of elastic coeff. with no Poisson effect}} + \underbrace{\mathbf{D}^\nu(\nu)}_{\text{Matrix of elastic coeff. with Poisson effect}}$$

- Elastic forces are integrated using Gauss quadrature, as usual
- This element, in the continuum-based flavor is available in Chrono
- It captures many more modes than the “cable” element
Chrono implementation of ANCF cable element

In general, the equations of motion of an assembled FE system...

$$\mathbf{M}\ddot{\mathbf{e}} = \mathbf{Q}_e + \mathbf{Q}_g + \mathbf{Q}_a$$

In Chrono, these forces are obtained on a per-element basis. That is, each system’s finite element is interrogated for internal forces, inertia forces, gravity, Jacobian of internal forces, and possibly other forces (e.g. pressure)

- Co-rotational formulation
  - Bar element, Euler beam, Hexa8, Hexa20, Tetra4, Tetra10
- ANCF
  - Cable element, Beam element, Shell elements (isotropic, orthotropic, composite)
- Other
  - EAS brick element (isotropic and hyperelastic Mooney-Rivlin), Brick9 (plasticity)
Chrono implementation of ANCF cable element

- ANCF elements are in the module Chrono::FEA
  - Corotational: Bar element, Euler beam, Hexa8, Hexa20, Tetra4, Tetra10, ANCF cable, ANCF beam, ANCF shell, bricks, etc.
  - For best results, use together with MKL module (linear solver)
  - Visualization possible through Irrlicht (online) or postprocessing through GNUPlot, Paraview, Pov-Ray, etc.

- For some reason, implemented in its entirety in a header file: ChElementBeamANCF.h
a. Mass matrix

class ChElementBeamANCF : public ChElementBeam, public ChLoadableU, public ChLoadableUVW {
protected:
    std::vector<std::shared_ptr<ChNodeFEAxyzD> > nodes;

    std::shared_ptr<ChBeamSectionCable> section;
    ChMatrixNM<double, 12, 1> m_GenForceVec0;
    ChMatrixNM<double, 12, 12> m_JacobianMatrix;  ///< Jacobian matrix (Kfactor*[K] + Rfactor*[R])
    ChMatrixNM<double, 12, 12> m_MassMatrix;      ///< mass matrix

public:
    ChElementBeamANCF() {
        nodes.resize(2);
        m_use_damping = false;  ///< flag to add internal damping and its Jacobian
        m_alpha = 0.0;          ///< scaling factor for internal damping

        // this->StiffnessMatrix.Resize(this->GetNdofs(), this->GetNdofs());
        // this->MassMatrix.Resize(this->GetNdofs(), this->GetNdofs());
    }

    bool m_use_damping;  ///< Boolean indicating whether internal damping is added
    double m_alpha;      ///< Scaling factor for internal damping

    virtual ~ChElementBeamANCF() {}

    virtual int GetNnodes() override { return 2; }
    virtual int GetNdofs() override { return 2 * 6; }
    virtual int GetNodeNdofs(int n) override { return 6; }

    virtual std::shared_ptr<ChNodeFEAbase> GetNodeN(int n) { return nodes[n]; }
}
a. Mass matrix

/// This material can be shared between multiple beams.
class ChApiFea ChBeamSectionCable : public ChBeamSection {
    public:
        double Area;
        double I;
        double E;
        double density;
        double rdamping;

    ChBeamSectionCable() {
        E = 0.01e9;  // default E stiffness: (almost rubber)
        SetDiameter(0.01);  // defaults Area, I
        density = 1000;  // default density: water
        rdamping = 0.01;  // default raleygh damping.
    }

    virtual ~ChBeamSectionCable() {}
virtual void ShapeFunctions(ChMatrix<> & N, double xi) {
    double l = this->GetRestLength();

    N(0) = 1 - 3 * pow(xi, 2) + 2 * pow(xi, 3);
    N(1) = l * (xi - 2 * pow(xi, 2) + pow(xi, 3));
    N(2) = 3 * pow(xi, 2) - 2 * pow(xi, 3);
    N(3) = l * (-pow(xi, 2) + pow(xi, 3));
}

/// Fills the N shape function derivative matrix with the
/// values of shape function derivatives at abscysa 'xi'.
/// Note, xi=0 at node1, xi=+1 at node2.
/// NOTE! to avoid wasting zero and repeated elements, here
/// it stores only the four values in a 1 row, 4 columns matrix!
virtual void ShapeFunctionsDerivatives(ChMatrix<> & Nd, double xi) {
    double l = this->GetRestLength();

    Nd(0) = (6.0 * pow(xi, 2.0) - 6.0 * xi) / l;
    Nd(1) = 1.0 - 4.0 * xi + 3.0 * pow(xi, 2.0);
    Nd(2) = -(6.0 * pow(xi, 2.0) - 6.0 * xi) / l;
    Nd(3) = -2.0 * xi + 3.0 * pow(xi, 2.0);
}

virtual void ShapeFunctionsDerivatives2(ChMatrix<> & Ndd, double xi) {
    double l = this->GetRestLength();
    Ndd(0) = (12 * xi - 6) / pow(l, 2);
    Ndd(1) = (-4 + 6 * xi) / l;
    Ndd(2) = (6 - 12 * xi) / pow(l, 2);
    Ndd(3) = (-2 + 6 * xi) / l;
}
a. Mass matrix

virtual void ComputeMassMatrix() {
    assert(section);
    double Area = section->Area;
    double rho = section->density;

class MyMass : public ChIntegrable1D<ChMatrixNM<double, 12, 12> > {
    public:
        ChElementBeamANCF* element;
        ChMatrixNM<double, 3, 12> S;
        ChMatrixNM<double, 1, 4> N;

    /// Evaluate the S'*S  at point x
    virtual void Evaluate(ChMatrixNM<double, 12, 12>& result, const double x) {
        element->ShapeFunctions(N, x);
        // S=[N1*eye(3) N2*eye(3) N3*eye(3) N4*eye(3)]
        ChMatrix33<> Si;
        Si.FillDiag(N(0));
        S.PasteMatrix(&Si, 0, 0);
        Si.FillDiag(N(1));
        S.PasteMatrix(&Si, 0, 3);
        Si.FillDiag(N(2));
        S.PasteMatrix(&Si, 0, 6);
        Si.FillDiag(N(3));
        S.PasteMatrix(&Si, 0, 9);
        // perform  r = S"S
        result.MatrTMultiply(S, S);
    }

    MyMass myformula;
    myformula.element = this;

    ChQuadrature::Integrate1D<ChMatrixNM<double, 12, 12> >(this->m_MassMatrix, // result of integration will go there
        myformula, // formula to integrate
        0,          // start of x
        1,          // end of x
        4           // order of integration
    );
    this->m_MassMatrix *= (rho * Area * this->length);
}
b. Generalized internal forces

Axial

class MyForcesAxial : public ChIntegrable1D<ChMatrixNM<double, 12, 1> > {
public:
    ChElementBeamANCF* element;
    ChMatrixNM<double, 4, 3>* d;  // this is an external matrix, use pointer
    ChMatrixNM<double, 12, 1>* d_dt;  // this is an external matrix, use pointer
    ChMatrixNM<double, 3, 12> Sd;
    ChMatrixNM<double, 1, 4> N;
    ChMatrixNM<double, 1, 4> Nd;
    ChMatrixNM<double, 1, 12> strainD;
    ChMatrixNM<double, 1, 1> strain;
    ChMatrixNM<double, 1, 3> Nd_d;
    ChMatrixNM<double, 12, 12> temp;

    /// Evaluate \( (\text{strainD}^T \cdot \text{strain}) \) at point \( x \)
    virtual void Evaluate(ChMatrixNM<double, 12, 1>& result, const double x) {
        element->ShapeFunctionsDerivatives(Nd, x);

        ChMatrix33<> Sdi; Sdi.FillDiag(Nd(0)); Sd.PasteMatrix(&Sdi, 0, 0);
        ChMatrix33<> Sdi; Sdi.FillDiag(Nd(1)); Sd.PasteMatrix(&Sdi, 0, 3);
        ChMatrix33<> Sdi; Sdi.FillDiag(Nd(2)); Sd.PasteMatrix(&Sdi, 0, 6);
        ChMatrix33<> Sdi; Sdi.FillDiag(Nd(3)); Sd.PasteMatrix(&Sdi, 0, 9);

        Nd_d = Nd * (*d);
        strainD = Nd_d * Sd;

        // Sd=[Nd1*\text{eye}(3) \text{Nd2*eye}(3) \text{Nd3*eye}(3) \text{Nd4*eye}(3)]
        ChMatrix33<> Sdi;
        Sdi.FillDiag(Nd(0));
        Sd.PasteMatrix(&Sdi, 0, 0);
        Sdi.FillDiag(Nd(1));
        Sd.PasteMatrix(&Sdi, 0, 3);
        Sdi.FillDiag(Nd(2));
        Sd.PasteMatrix(&Sdi, 0, 6);
        Sdi.FillDiag(Nd(3));
        Sd.PasteMatrix(&Sdi, 0, 9);

    }
};
b. Generalized internal forces

Axial

```cpp
// strain = (Nd*(d*d')*Nd' - 1)*0.5;
strain.MatrMultiplyT(Nd_d, Nd_d);
strain(0, 0) += -1;
strain(0, 0) *= 0.5;

// Add damping forces if selected
if (element->m_use_damping)
  strain(0, 0) += (element->m_alpha) * (strainD * (*d_dt))(0, 0);

result.MatrTMultiply(strainD, strain);
// result: strainD'*strain
```

Calculation of strain

Optional addition of damping

Result dependent on space parameter

```cpp
MyForcesAxial myformulaAx;
myformulaAx.d = &d;
myformulaAx.d_dt = &vel_vector;
myformulaAx.element = this;

ChMatrixNM<double, 12, 1> Faxial;
ChQuadrature::Integrate1D<ChMatrixNM<double, 12, 1>>(Faxial,   // result of integration will go there
  myformulaAx,   // formula to integrate
  0,   // start of x
  1,   // end of x
  5,   // order of integration
  );
Faxial *= -E * Area * length;
Fi = Faxial;
```