01 - Repaso de análisis estructural matricial

Diego Andrés Alvarez Marín
Profesor Asociado
Universidad Nacional de Colombia
Sede Manizales

https://github.com/diegoandresalvarez/elementosfinitos

Contenido

- Solución al sistema de ecuaciones $\boldsymbol{q}=\boldsymbol{K}\boldsymbol{a}-\boldsymbol{f}$
- Conceptos básicos del análisis matricial de estructuras de barras
- Cerchas 2D y 3D
- Marcos 2D y 3D
- Apoyos inclinados
- Etapas básicas del análisis de un sistema de barras

El método de los elementos finitos

Es una técnica numérica para la solución **aproximada** de ecuaciones diferenciales e integrales.

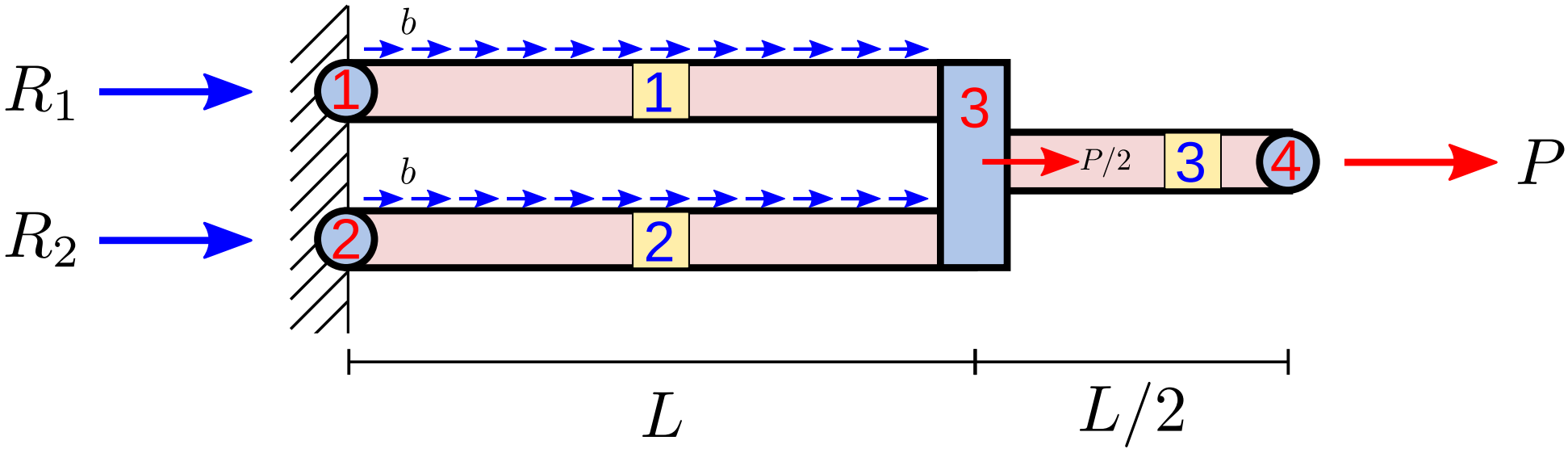
En el ámbito del análisis estructural (estructuras, geotecnia, pavimentos) es un poderoso método para la **estimación** de los esfuerzos, deformaciones y desplazamientos de una estructura bajo la acción de un conjunto de cargas.

El MEF se considera hoy en día como el procedimiento más potente que existe para el análisis de estructuras de carácter uni-, bi- o tridimensional sometidas a las acciones exteriores más diversas.

Historia del método de los EFs

- 1934-1938: Collar, Duncan: primer artículo sobre análisis matricial de estructuras
- 1943: Richard Courant (New York University): primer artículo matemático
- 1954: Argyris: desarrolla el concepto de ensamblaje matricial
- 1956: Turner, Clough, Martin, Topp (Boeing): primer artículo en ingeniería
- 1959: Turner: presenta un artículo sobre el método matricial de la rigidez, tal y como lo conocemos hoy
- 1960: Clough (Boeing, Berkeley) define el término “elemento finito”.
- 1965: La NASA invierte 4E6 USD (35E6 USD de hoy) para el desarrollo del software NASTRAN (**NAS**A **Str**uctural **An**alysis)
- 1965: Se empieza a aplicar a otros campos de la ingeniería
- 1967: Zienkiewicz (Swansea): primer libro sobre EFs
- 1970: Se empieza a desarrollar el software **ANSYS**
- 1973: Strang, Fix (M.I.T.): primer libro sobre EFs desde el punto de vista matemático

Análisis de tres barras trabajando a tracción



E y A iguales para cada barra

Ver solución en MATLAB y PYTHON en:
https://github.com/diegoandresalvarez/elementosfinitos/tree/master/codigo/repaso_matricial/barra_1d

Ensamblaje matricial

La expresión de equilibrio de una estructura compuesta de barras se obtiene a partir de la regla sencilla que expresa la suma de las fuerzas nodales de equilibrio en un nodo global j (debidas a las diferentes barras que concurren en el mismo) es igual a la sumatoria de las fuerzas exteriores (reacciones + fuerzas puntuales exteriores + fuerzas nodales equivalentes) que actúa en dicho nodo global j. Por lo tanto, para el nodo global j se tiene que:

$$\sum_{e \in E_j} q^{(e)}_{GaL(e,j)} = \sum_i f_i^{exterior}(j)$$

Aquí E_j representa el conjunto de barras que concurren en el nodo global j, $q_i^{(e)}$ son las fuerzas nodales de equilibrio para el nodo i local del elemento e y $\sum_i f_i^{exterior}(j)$ representa la suma de las fuerzas exteriores que concurren en el nodo global j .[6]

Coordenadas locales y globales
Matrices LaG y GaL

A **LaG** se le llama también la **lista de ensamblaje** o la **matriz de conectividades (connectivity matrix)**

LaG(e, nodo local)

barra (e)	# nodo local 1	# nodo local 2
1	1	3
2	2	3
3	3	4

GaL(e, nodo global)

barra (e)	# nodo global 1	# nodo global 2	# nodo global 3	# nodo global 4
1	1	x	2	x
2	x	1	2	x
3	x	x	1	2

$$E_1 = \{1\}$$
$$E_2 = \{2\}$$
$$E_3 = \{1,\ 2,\ 3\}$$
$$E_4 = \{3\}$$

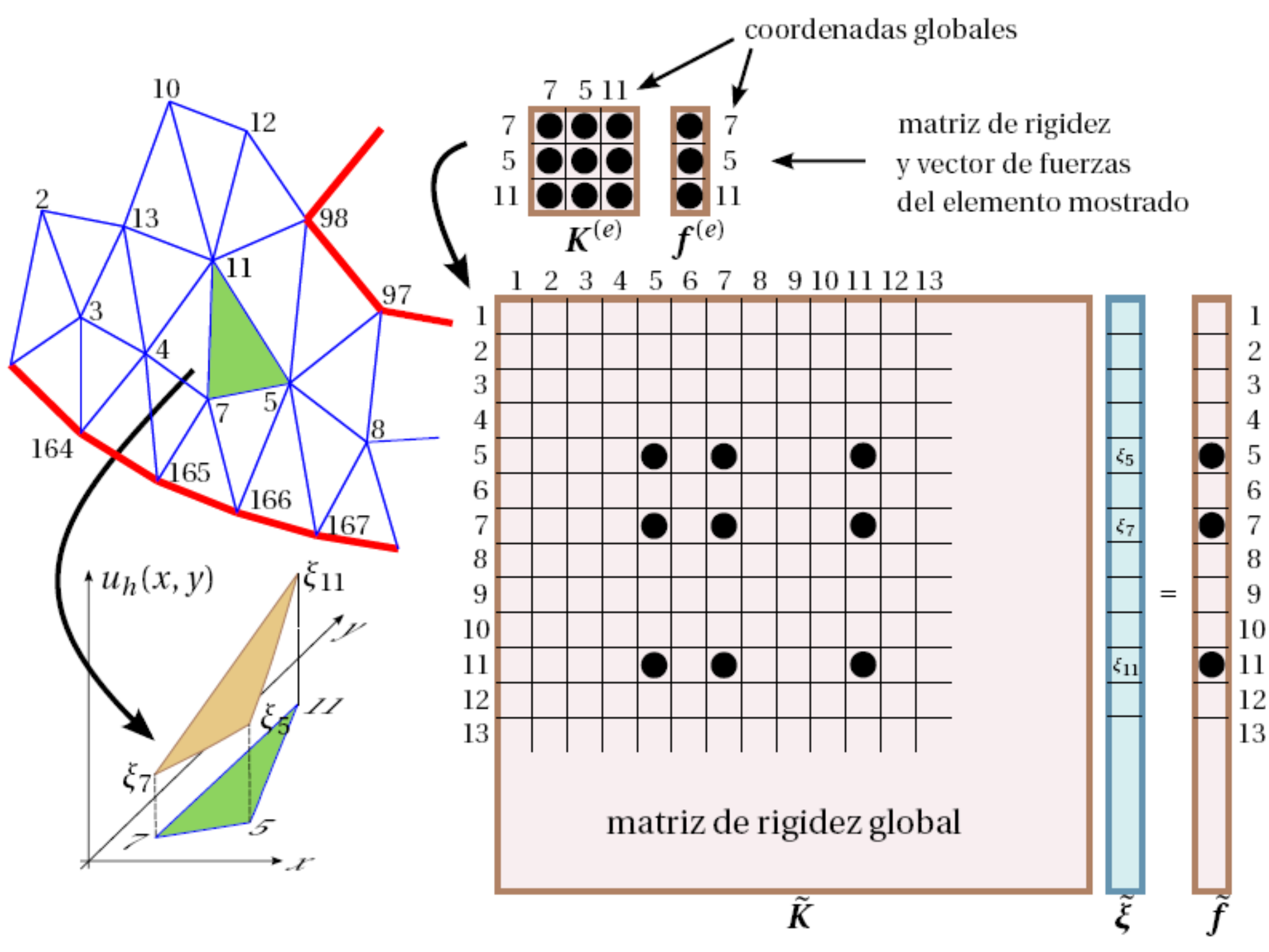
$$① \Rightarrow \quad q_1^{(1)} = R_1 + f_1^{(1)}$$

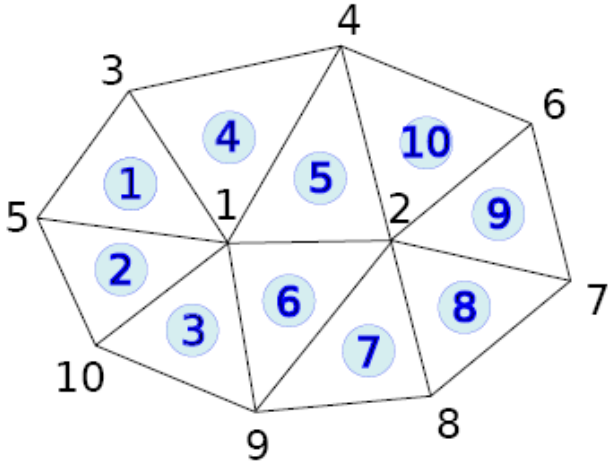
$$② \Rightarrow \quad q_1^{(2)} = R_2 + f_1^{(2)}$$

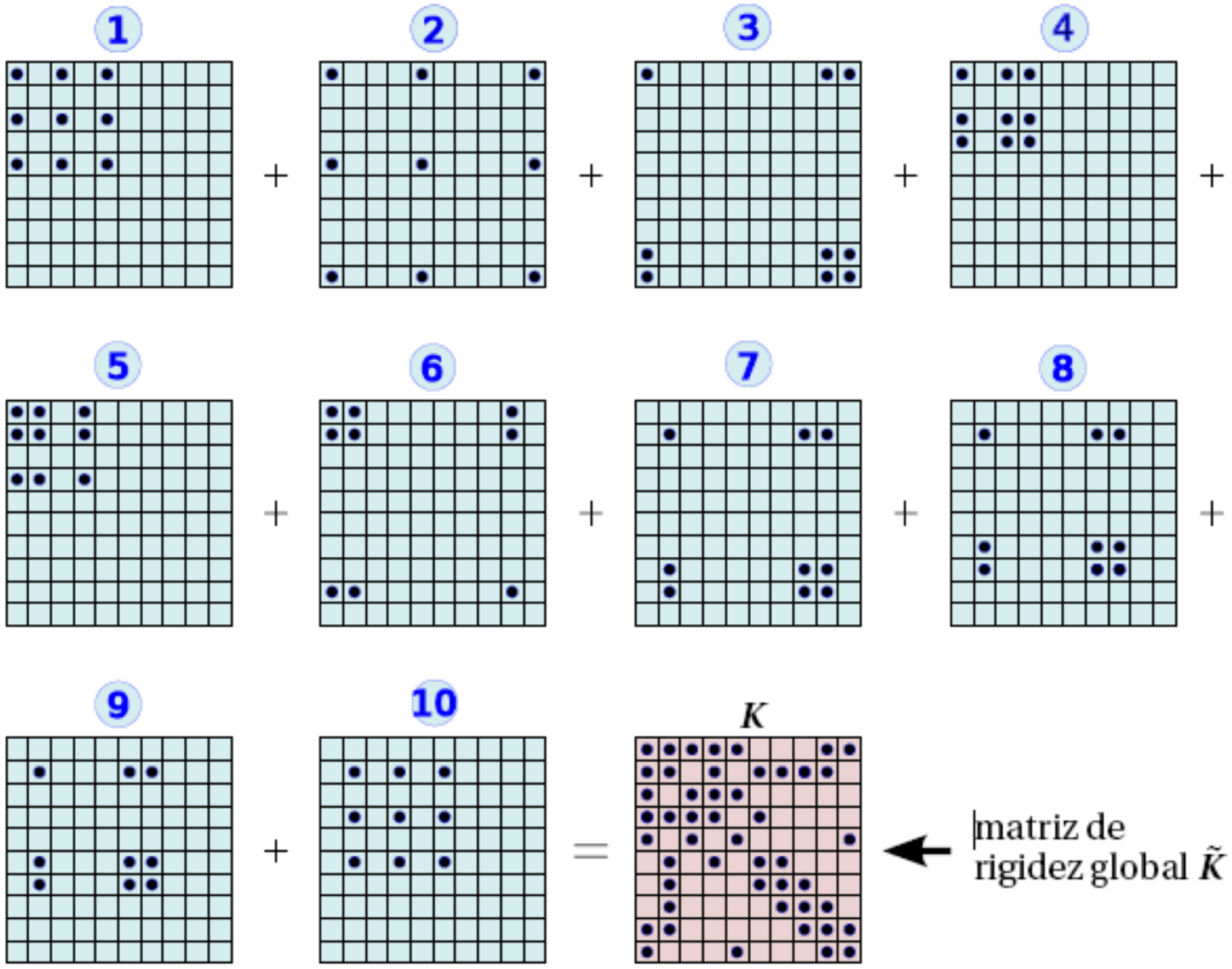
$$③ \Rightarrow q_2^{(1)} + q_2^{(2)} + q_1^{(3)} = \frac{P}{2} + f_2^{(1)} + f_2^{(2)}$$

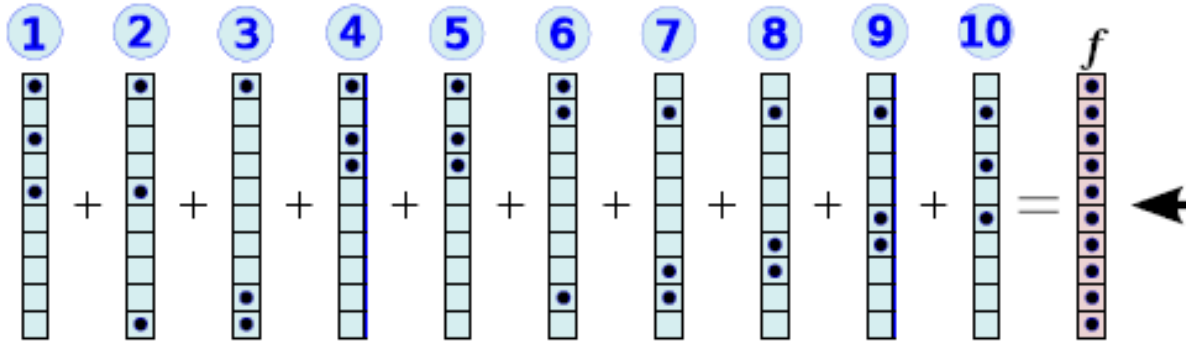
$$④ \Rightarrow \underbrace{q_2^{(3)}}_{\text{sumatoria de fuerzas nodales de equilibrio}} = \underbrace{P}_{\text{sumatoria de fuerzas exteriores}}$$

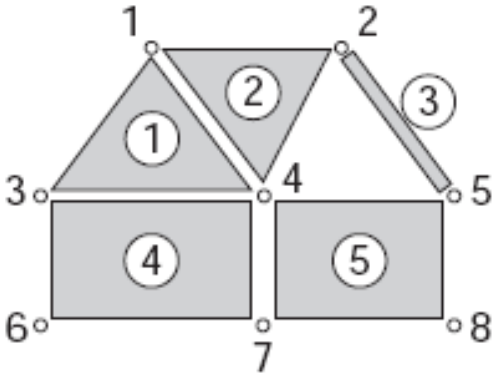
$$\sum_{e \in E_j} q^{(e)}_{GaL(e,j)} = \sum_i f_i^{exterior}(j)$$


coordenadas globales
matriz de rigidez
y vector de fuerzas
del elemento mostrado
$K^{(e)}$
$f^{(e)}$
matriz de rigidez global
$\tilde{K}$
$\tilde{\xi}$
$\tilde{f}$
=
$u_h(x,y)$
ξ_{11}
ξ_5
ξ_7
ξ_5
ξ_7

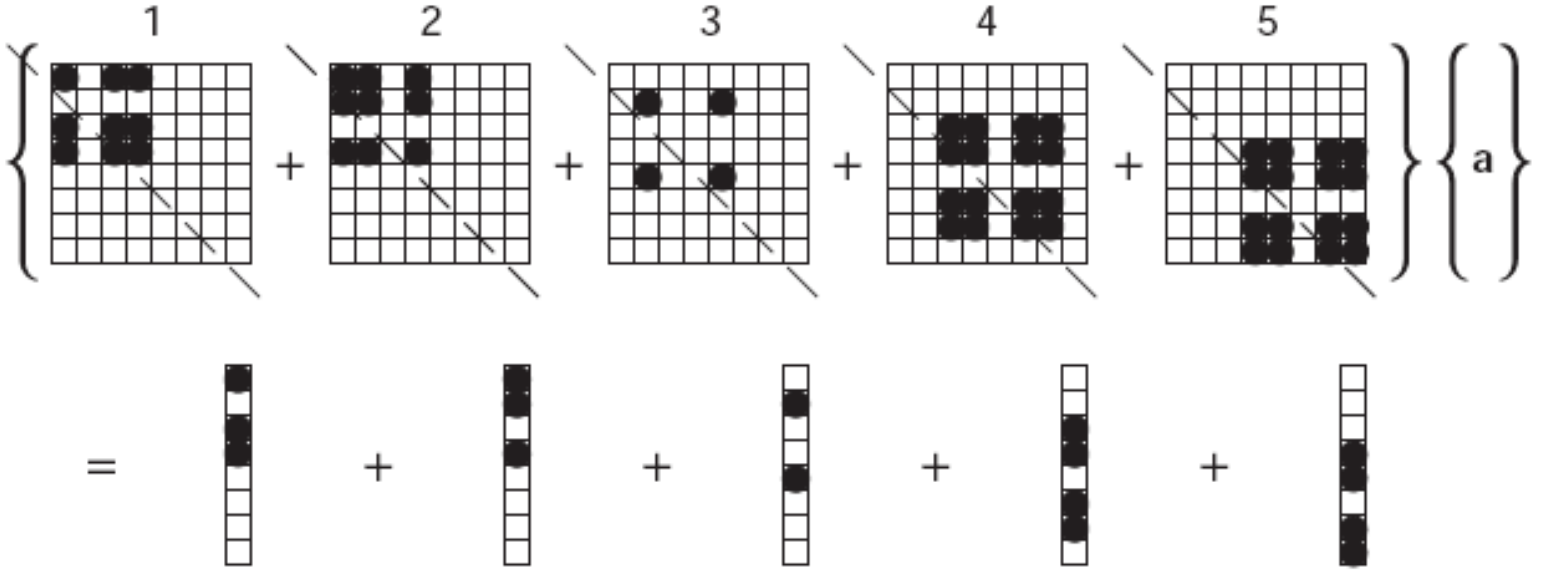

4
3
6
5
1
2
7
10
9
8
1
2
3
4
5
6
7
8
9
10


1
2
3
4
5
6
7
8
9
10
K
matriz de rigidez global $\tilde{K}$

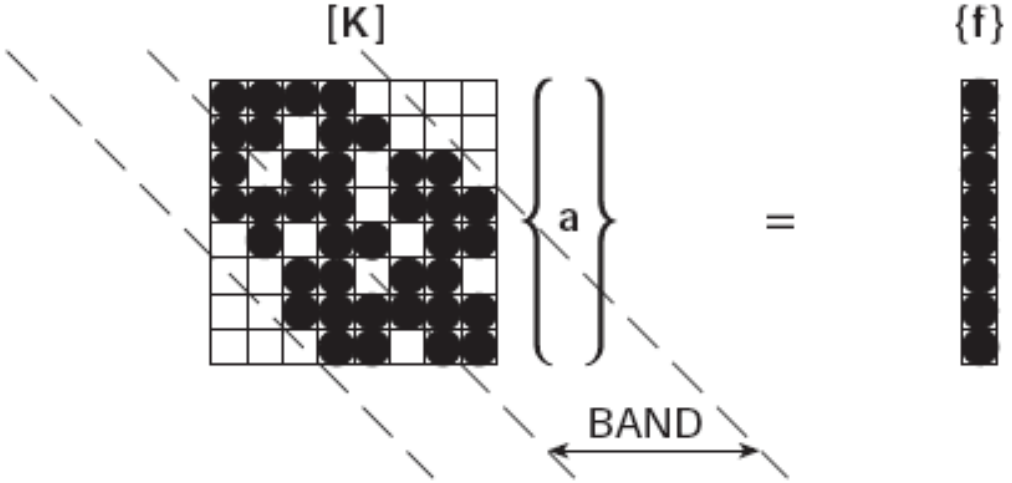

1
2
3
4
5
6
7
8
9
10
f
vector de fuerzas global $\tilde{f}$



(a)



(b)



(c)

Solución al sistema de ecuaciones

$$\begin{bmatrix} \boldsymbol{q}_d \\ \boldsymbol{q}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}_{cc} & \boldsymbol{K}_{cd} \\ \boldsymbol{K}_{dc} & \boldsymbol{K}_{dd} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_c \\ \boldsymbol{a}_d \end{bmatrix} - \begin{bmatrix} \boldsymbol{f}_d \\ \boldsymbol{f}_c \end{bmatrix}$$

$$\boldsymbol{a}_d = \boldsymbol{K}_{dd}^{-1} \left(\boldsymbol{q}_c + \boldsymbol{f}_c - \boldsymbol{K}_{dc} \boldsymbol{a}_c \right)$$

$$\boldsymbol{q}_d = \boldsymbol{K}_{cc} \boldsymbol{a}_c + \boldsymbol{K}_{cd} \boldsymbol{a}_d - \boldsymbol{f}_d$$

Siempre $\boldsymbol{q}_c = \boldsymbol{0}$ (ya que en estos grados de libertad no hay apoyos y por lo tanto tampoco hay reacciones) y usualmente $\boldsymbol{f}_d = \boldsymbol{0}$. Sin embargo si existen cargas puntuales que van directamente a los apoyos, $\boldsymbol{f}_d$ contendrá dichas cargas y $\boldsymbol{f}_d$ será diferente de $\boldsymbol{0}$.

Las cargas puntuales y las fuerzas nodales equivalentes siempre se ingresan en el vector $\boldsymbol{f}_c$. El vector $\boldsymbol{q}$ únicamente se utiliza para modelar las reacciones en los apoyos (específicamente $\boldsymbol{q}_d$, ya que $\boldsymbol{q}_c = \boldsymbol{0}$).

Solución al sistema de ecuaciones

$$\begin{bmatrix} \boldsymbol{q}_d \\ \boldsymbol{q}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}_{cc} & \boldsymbol{K}_{cd} \\ \boldsymbol{K}_{dc} & \boldsymbol{K}_{dd} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_c \\ \boldsymbol{a}_d \end{bmatrix} - \begin{bmatrix} \boldsymbol{f}_d \\ \boldsymbol{f}_c \end{bmatrix}$$

$$\boldsymbol{a}_d = \boldsymbol{K}_{dd}^{-1} \left(\cancel{\boldsymbol{q}_c} + \boldsymbol{f}_c - \boldsymbol{K}_{dc}\boldsymbol{a}_c \right)$$

$$\boldsymbol{q}_d = \boldsymbol{K}_{cc}\boldsymbol{a}_c + \boldsymbol{K}_{cd}\boldsymbol{a}_d - \boldsymbol{f}_d$$

```
% Se resuelve el sistema de ecuaciones: cálculo de los vectores ad y fd
ad = Kdd\(fc - Kdc*ac);    % desplazamientos desconocidos
qd = Kcc*ac + Kcd*ad - fd; % fuerzas nodales de equilibrio desconocidas
```

$c \rightarrow$ g.d.l. de desplazamientos conocidos

$d \rightarrow$ g.d.l. de desplazamientos desconocidos

$\boldsymbol{q} \rightarrow$ vector de fuerzas nodales de equilibrio (reacciones) (ngdl $\times$ 1)

$\boldsymbol{K} \rightarrow$ matriz de rigidez (ngdl $\times$ ngdl)

$\boldsymbol{a} \rightarrow$ vector de desplazamientos (ngdl $\times$ 1)

$\boldsymbol{f} \rightarrow$ vector de fuerzas nodales equivalentes (ngdl $\times$ 1)

Resolviendo

$$\boldsymbol{a}_d = \boldsymbol{K}_{dd}^{-1}\left(\boldsymbol{f}_c - \boldsymbol{K}_{dc}\boldsymbol{a}_c\right)$$

$$\boldsymbol{q}_d = \boldsymbol{K}_{cc}\boldsymbol{a}_c + \boldsymbol{K}_{cd}\boldsymbol{a}_d - \boldsymbol{f}_d$$

Depending on how the elements are connected together, the stiffness matrix ***K*** is generally very sparse; that is, many of the entries are zero. Since a value of zero does not contribute to the solution, most solution techniques are programmed in some fashion or another to ignore the zero terms. One method may use a small amount of memory to store the matrix but require optimization to achieve the least amount of memory, where as another method uses more variables to store the matrix (and hence require more memory) but be faster overall.

Verifique que su algoritmo no esté haciendo SWAPPING en su disco duro (lo mira en el TASK MANAGER). Si lo hace cambie de SOLVER o en caso extremo, cómprele RAM a su computador.

Resolviendo

$$\boldsymbol{a}_d = \boldsymbol{K}_{dd}^{-1}\left(\boldsymbol{f}_c - \boldsymbol{K}_{dc}\boldsymbol{a}_c\right)$$
$$\boldsymbol{q}_d = \boldsymbol{K}_{cc}\boldsymbol{a}_c + \boldsymbol{K}_{cd}\boldsymbol{a}_d - \boldsymbol{f}_d$$

El tiempo de cálculo $t_{\text{cálculo}}$ de este sistema de ecuaciones es:

$$t_{\text{cálculo}} \propto n_{\text{gdl}}^{\alpha}$$

donde α es un número entre 1.7 y 3.0 y n_{gdl} es el número de grados de libertad libres (filas de $\boldsymbol{K}_{dd}$). El valor de la constante α depende del algoritmo de solución escogido y de la estructura del la matriz $\boldsymbol{K}_{dd}$ (por ejemplo el ancho de banda).

$\alpha = 3$ para la eliminación gausiana.

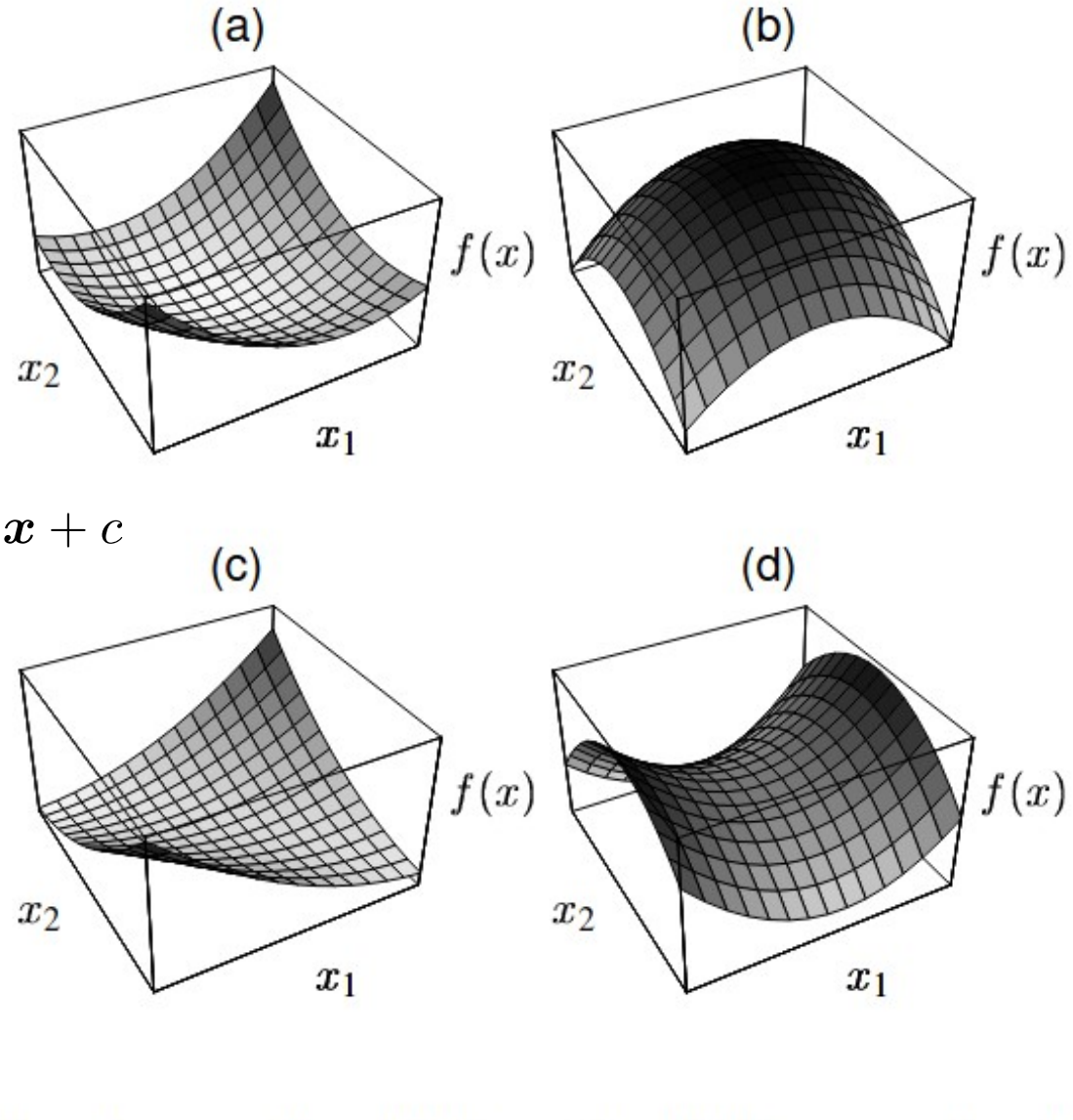
Matriz definida positiva

Se dice que una matriz simétrica real $\boldsymbol{K} \in \mathbb{R}^{d \times d}$ es *definida positiva* (semi-definida positiva) si y solo si $\boldsymbol{K}$ satisface alguna de las siguientes condiciones equivalentes:

- Todos sus valores propios λ_i son reales y $\lambda_i > 0$ ($\lambda_i \geq 0$).
- $\boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} > 0$ ($\boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} \geq 0$) para todo $\boldsymbol{x} \in \mathbb{R}^d \setminus \{\boldsymbol{0}\}$.

Propiedades:

- Si $\boldsymbol{A}$ es una matriz real invertible, entonces $\boldsymbol{A}^T \boldsymbol{A}$ es definida positiva.
- Si $\boldsymbol{K}$ es definida positiva, entonces $\boldsymbol{K}^{-1}$ existe y es definida positiva.
- Si $\boldsymbol{K}_1$ y $\boldsymbol{K}_2$ son definidas positivas, entonces $\boldsymbol{K}_1 + \boldsymbol{K}_2$ también es definida positiva.



$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} + c$$

Figure 5: (a) Quadratic form for a positive-definite matrix. (b) For a negative-definite matrix. (c) For a singular (and positive-indefinite) matrix. A line that runs through the bottom of the valley is the set of solutions. (d) For an indefinite matrix. Because the solution is a saddle point, Steepest Descent and CG will not work. In three dimensions or higher, a singular matrix can also have a saddle.

Jonathan Richard Shewchuk (1994) - An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Edition 1 1/4.
https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf

Solvers: direct vs iterative

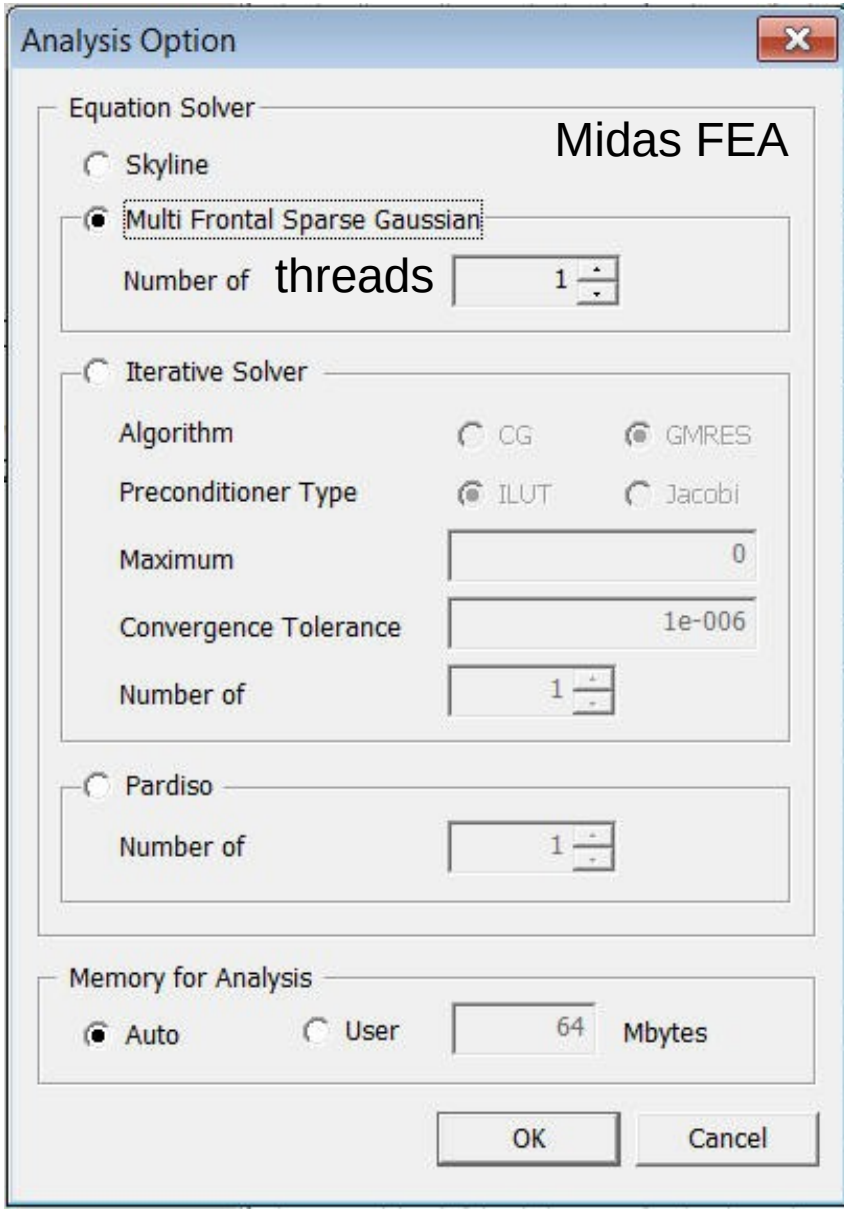
Existen dos tipos de métodos para resolver los grandes sistemas de ecuaciones:

- Métodos directos: son métodos que solucionan el sistema de ecuaciones de forma exacta asumiendo que el error de representación no existe.
- Métodos iterativos: son métodos que en teoría solo proveen la solución exacta después de un número infinito de iteraciones.

Solvers: direct vs iterative

Los programas de EFs, tienen implementados internamente diferentes algoritmos para resolver sistemas de ecuaciones. Algunos de los más usados son:

- Direct solvers:
  - Skyline
  - Multi-Frontal Sparse Gaussian
  - PARDISO
  - MUMPS (excelente en clusters http://mumps.enseeiht.fr/)
  - SPOOLES (http://www.netlib.org/linalg/spooles/spooles.2.2.html)
  - LDLT
  - Multigrid methods
  - etc ...
- Iterative Solvers
  - CG
  - GMRES
  - PETSc
  - etc ...



https://www.comsol.com/blogs/solutions-linear-systems-equations-direct-iterative-solvers/

Solvers: direct vs iterative

- Direct solvers
  - Se usan para sistemas de hasta 50.000 ecuaciones.
- Iterative solvers:
  - Se usan para sistemas de más 50.000 ecuaciones y en sistemas de ecuaciones sparse.

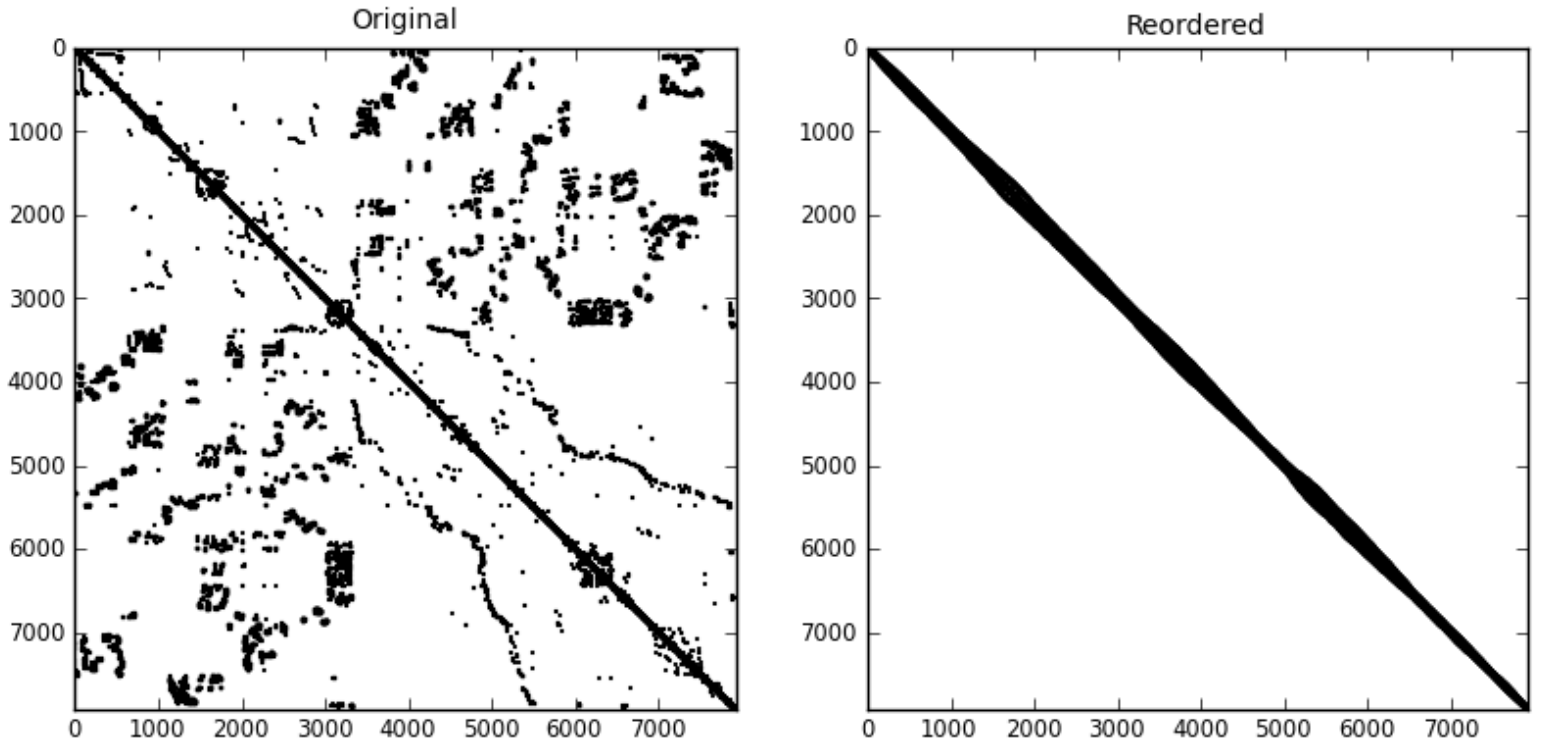
Dependiendo del algoritmo, algunos usan más o menos memoria, a cambio de velocidad en la solución.

Algunos algoritmos son adecuados para cálculo en paralelo.

Usualmente los algoritmos directos son más veloces.

Toca consultar el manual de referencia del software.

Ancho de banda



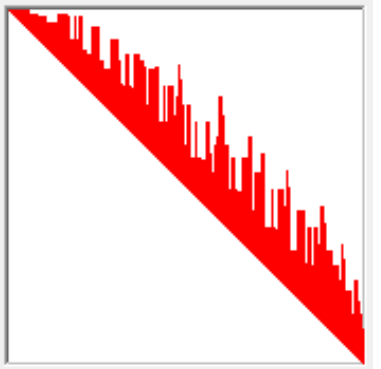
La numeración de los nodos debe hacerse de modo tal que el ancho de banda sea tan pequeño como sea posible. Existen algoritmos especializados que hacen esta labor (como el **algoritmo invertido de Cuthill-McKee**). Algunos programas de elementos finitos automáticamente numeran los nodos de modo que el ancho de banda sea el menor posible.

Mirar la documentación del SIMSCALE

- Allí hay una muy buena documentación con respecto a este tema.
- https://www.simscale.com/blog/2016/08/how-to-choose-solvers-for-fem/

Direct solvers: Skyline solver

https://en.wikipedia.org/wiki/Skyline_matrix



- Es el más popular en análisis estructural. Se usa para matrices de rigidez simétricas (solo se almacena la parte superior), definidas positivas (cuando no lo son, el programa usa el MFSPS (ver sgte diapositiva).
- Utiliza internamente algoritmos similares el reverse Cuthill–McKee algorithm + Cholesky para renumerar los nodos de modo que el ancho de banda se optimice. El MIDAS usa, por ejemplo, el wavefront reduction method (Sloan, 1987).
- It internally re-arranges the nodes to place more non-zero terms near the diagonal. That is, the bandwidth is optimized. This process can take significant time in a large model. Instead of storing the entire matrix, the minimum bandwidth for each row is stored. The minimum bandwidth is the number of entries starting with the diagonal term up to the last non-zero term. Since each row has a different bandwidth, a different number of terms are stored for each row. (This is where the terminology of "Skyline" solver comes from. Because each row has a different number of terms stored, the stored matrix looks like a city "skyline" if it were rotated 90 degrees.)
- Because fewer terms are stored, the skyline solver requires less memory than other methods, pero podría utilizar más espacio en disco.
- The skyline solver is efficient for small models < 50.000 gdl.

Multi Frontal Sparse Gaussian Solver (MFSPS)

- The MFSGS uses an optimum frontal division algorithm to minimize the number of calculations for simultaneous linear equations.
- Resuelve matrices simétricas y no simétricas.
- https://en.wikipedia.org/wiki/Frontal_solver
- Excelente si tiene varios núcleos.

PARDISO solver

- Existen dos versiones de este solver:
  - https://www.pardiso-project.org/
  - https://software.intel.com/en-us/mkl-developer-reference-fortran-intel-mkl-pardiso-parallel-direct-sparse-solver-interface
- The PARDISO solver shows both a high performance and memory efficient usage for solving large sparse symmetric and unsymmetric linear systems of equations by shared multiprocessors.
- Optimal operation on the Intel® processors, including parallel processing on multicore processors (Intel® Math Kernel Library (Intel® MKL))

Iterative solvers

(evítelos a menos que haya ajustado bien los parámetros)

Se utilizan en modelos de EFs con muchos gdl (más de 50.000 gdl). La solución se calcula con métodos iterativos como:

- **Conjugate Gradient Method (CG)**: se utiliza cuando la matriz de rigidez es simétrica y definida positiva (nuestro caso en análisis estructural). https://en.wikipedia.org/wiki/Conjugate_gradient_method
- **Generalized Minimum Residual Method (GMRES)**: se utiliza para matrices de rigidez no simétricas o no definidas positivas (suceden en análisis no lineales). https://en.wikipedia.org/wiki/Generalized_minimal_residual_method

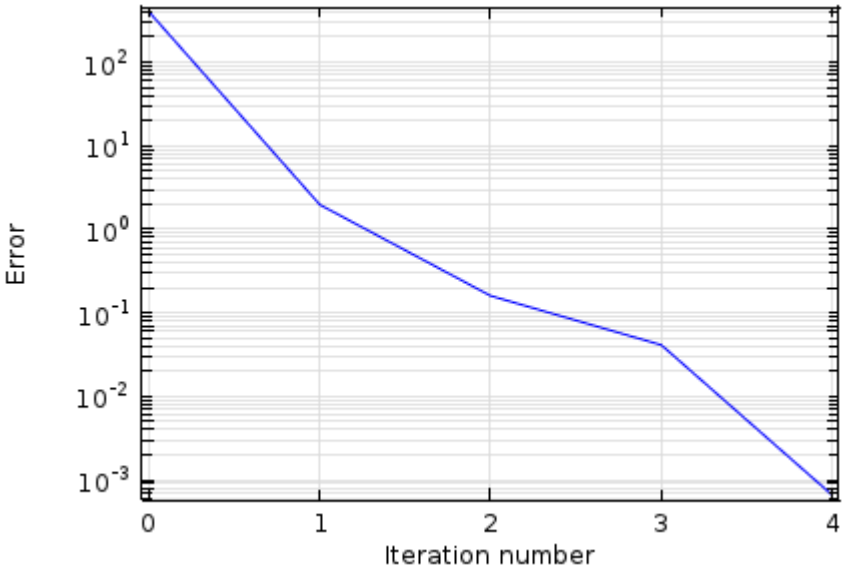
- Maximum iteration: colocar 0 hace que el MIDAS use max(ngdl/4, 1.000). Cuidado con este número. No hay garantía que el modelo converja en ese número de iteraciones.
- Convergence tolerance ε (valor por defecto 1e-6): especifica la tolerancia en

$$\|\boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}_i\| < \varepsilon\|\boldsymbol{f}\|$$

This tolerance can be made looser, for faster solutions, or tighter, for greater accuracy on the current mesh. The tolerance must always be greater than the machine precision (eps = 2.22e-16)

Iterative solvers

- Estos métodos no optimizan el ancho de banda.
- When able to converge, the iterative solver is the fastest solution for large models (sólidos > 150.000 gdl, fluídos > 50.000 gdl).
- **ADVERTENCIA!** The accuracy of the solution depends on the convergence tolerance; a smaller tolerance will result in a more accurate solution but may take more iterations. As with any iterative solution, the results should be checked to confirm that they meet the desired accuracy. In some cases, performing the analysis twice with a different convergence tolerance is the best way to confirm the accuracy.



Preconditioners

https://en.wikipedia.org/wiki/Preconditioner

El desempeño de los métodos iterativos depende del número de condición (condition number) de la matriz de rigidez $\boldsymbol{K}_{dd}$. El número de condición es la razón entre el valor propio más grande $\lambda_{máx}$ y el más pequeño $\lambda_{mín}$ de $\boldsymbol{K}_{dd}$.

$$\kappa(\boldsymbol{K}_{dd}) = \frac{|\lambda_{\max}(\boldsymbol{K}_{dd})|}{|\lambda_{\min}(\boldsymbol{K}_{dd})|}$$

Para cambiar el número de condición de $\boldsymbol{K}_{dd}$ y así mejorar el desempeño de los métodos iterativos se utilizan los precondicionadores.

El MIDAS tiene implementados:

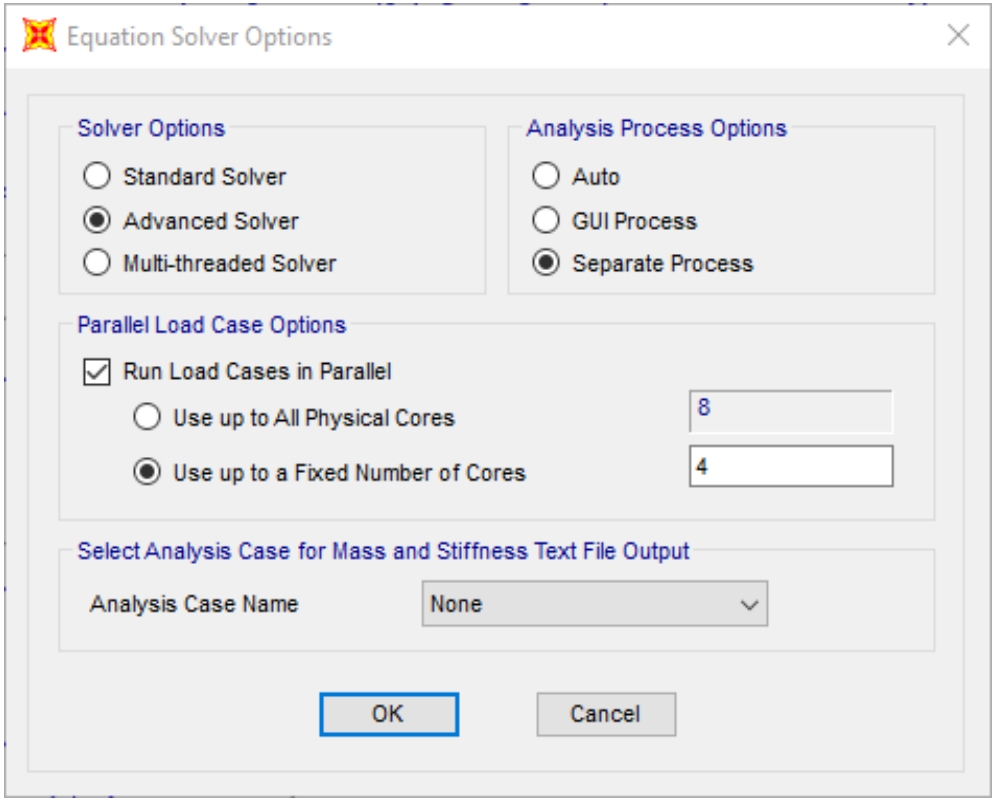
- Incomplete LU with drop-tolerance (ILUT): Meijerink & Van Der Vorst (1977)
- Jacobi Preconditioner

Configurando el solver en SAP2000

Al ser algoritmos distintos, los tres solvers podrían dar resultados ligeramente diferentes.

Standard solver
- es el más lento y usa la mayor cantidad de disco.
- best for small problems
- utilizes only single core/CPU
- provides full instability information, which is useful for checking model stability before a long analysis

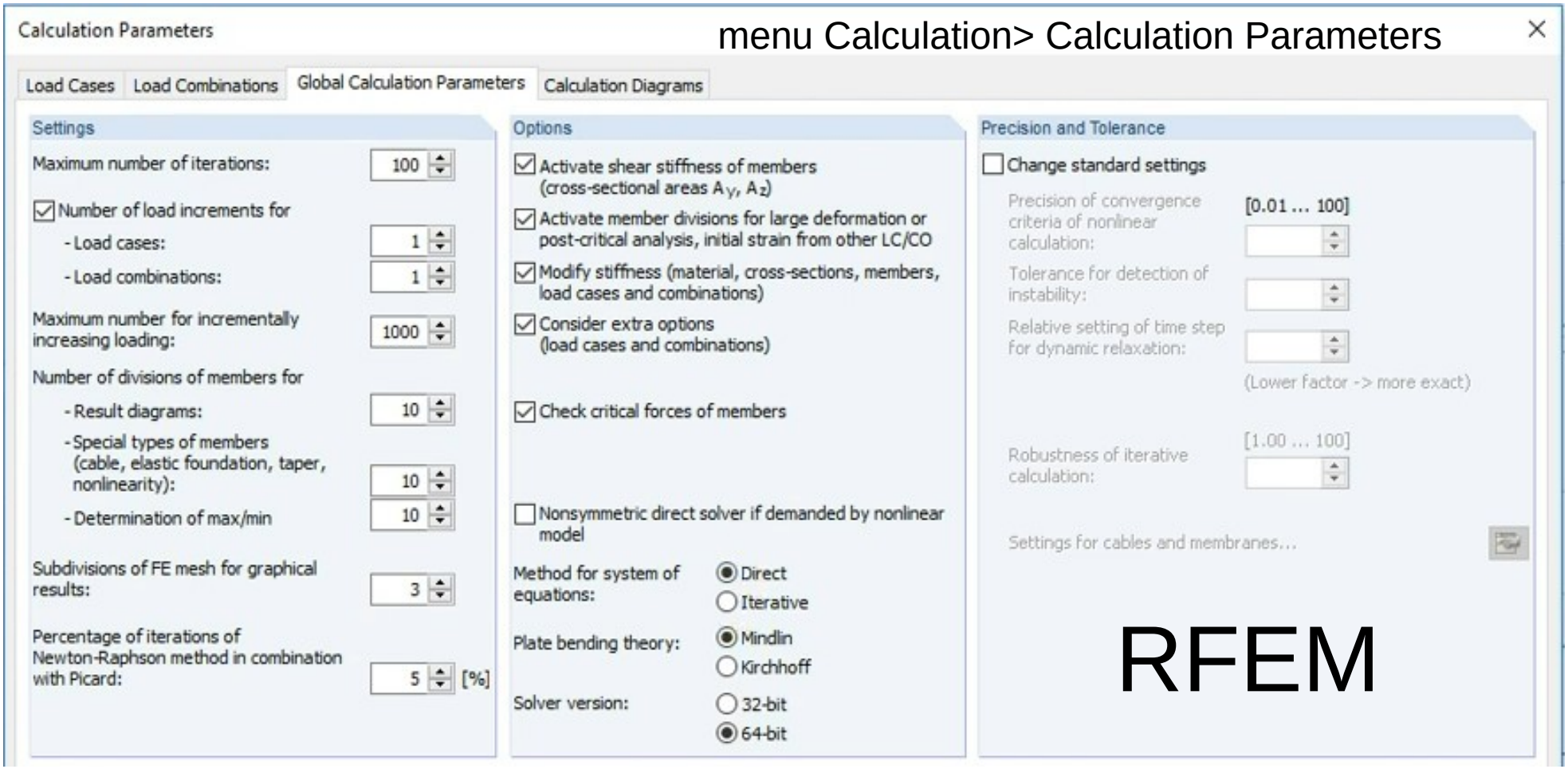
Advanced solver
- default setting
- x2 a x3 más rápido que el standard solver
- usa menos espacio de disco
- best for medium problems
- can utilize all cores/CPUs
- utilizes disk to handle very large models
- provides limited instability information



Multi-threaded solver
- es el más rápido de los solvers
- best for medium to large problems
- fully utilizes all cores/CPUs, no usa disco
- fully runs in RAM for speed
- provides no instability information

Recommendations:
- Use the Standard or Advanced (default) solver to check for stability early in the development of models.
- Switch to the Multi-threaded solver for speed when the model is well developed and stable.
- Use the Advanced solver instead if the model is too large and the Multi-threaded solver reports memory limitations.

Tomado de: https://wiki.csiamerica.com/display/kb/Explanation+of+solver+options



En resumen, por cuestiones de velocidad:

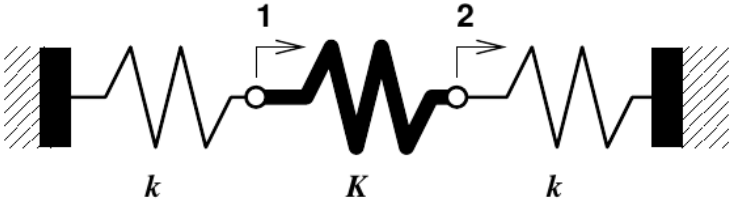
- Para problemas pequeños y medianos es preferible el uso de métodos directos.
- Para problemas grandes es preferible usar métodos iterativos.

Si el cálculo toma demasiado tiempo, verifique que no se esté usando la memoria SWAP del computador (el sistema operativo usa disco duro como si fuera memoria RAM). En este caso si se tiene un método directo, páselo a uno iterativo.

De otro lado, si el computador está tomando demasiado tiempo, pero la carga sobre el procesador y el uso de la memoria es bajo, páselo a uno directo.

La matriz de rigidez $\boldsymbol{K}$

- Los elementos de su diagonal son siempre positivos; si ellos son negativos o cero, es porque la estructura es inestable.
- Es una matriz singular, simétrica, semidefinida positiva y usualmente rala.
- Al eliminar los gdl restringidos, $\boldsymbol{K}$ se convierte en $\boldsymbol{K}_{dd}$; dicha matriz es simétrica y definida positiva y usualmente invertible.
- $\det(\boldsymbol{K}_{dd}) = 0$ (matriz singular) cuando:
  - La estructura no está bien restringida y por lo tanto ella tiene un movimiento rígido.
  - La estructura es inestable.
- $\det(\boldsymbol{K}_{dd}) \approx 0$ (matriz mal condicionada – *ill-conditioned matrix*) cuando:
  - Existen elementos demasiados rígidos en comparación a otros elementos.



$K = 10^6\, k$

Imagen tomada de: Henri P. Gavin (2014). Mathematical Properties of Stiffness Matrices. Duke University.

Cerchas

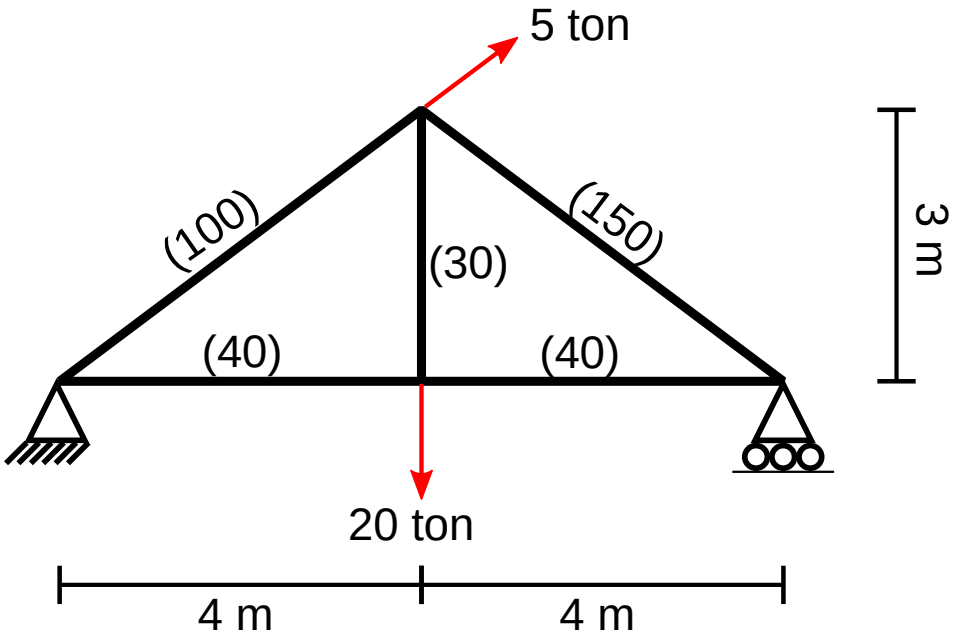
$$\boldsymbol{K}_{\text{loc}}^{(e)} = \begin{bmatrix} k^{(e)} & 0 & -k^{(e)} & 0 \\ 0 & 0 & 0 & 0 \\ -k^{(e)} & 0 & k^{(e)} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad k^{(e)} = \frac{A^{(e)} E^{(e)}}{L^{(e)}}$$

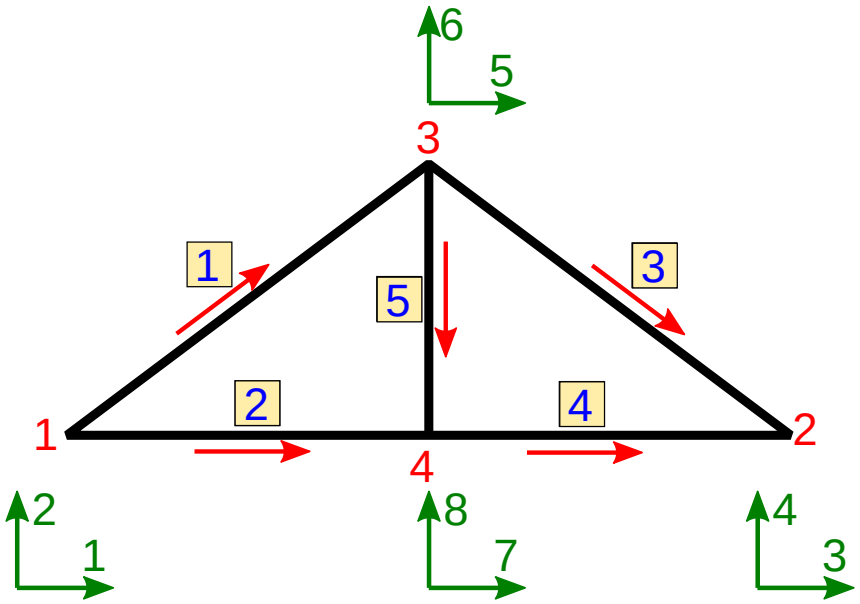
$$\boldsymbol{T}_{(e)} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

$$\boldsymbol{K}^{(e)} = \boldsymbol{T}_{(e)}^{T} \boldsymbol{K}_{\text{loc}}^{(e)} \boldsymbol{T}_{(e)}$$

Ejemplo 11.3 Uribe Escamilla

Resuelva completamente la estructura mostrada.
El material es acero: E = 2040 ton/cm^2
Las áreas están entre paréntesis en cm^2.

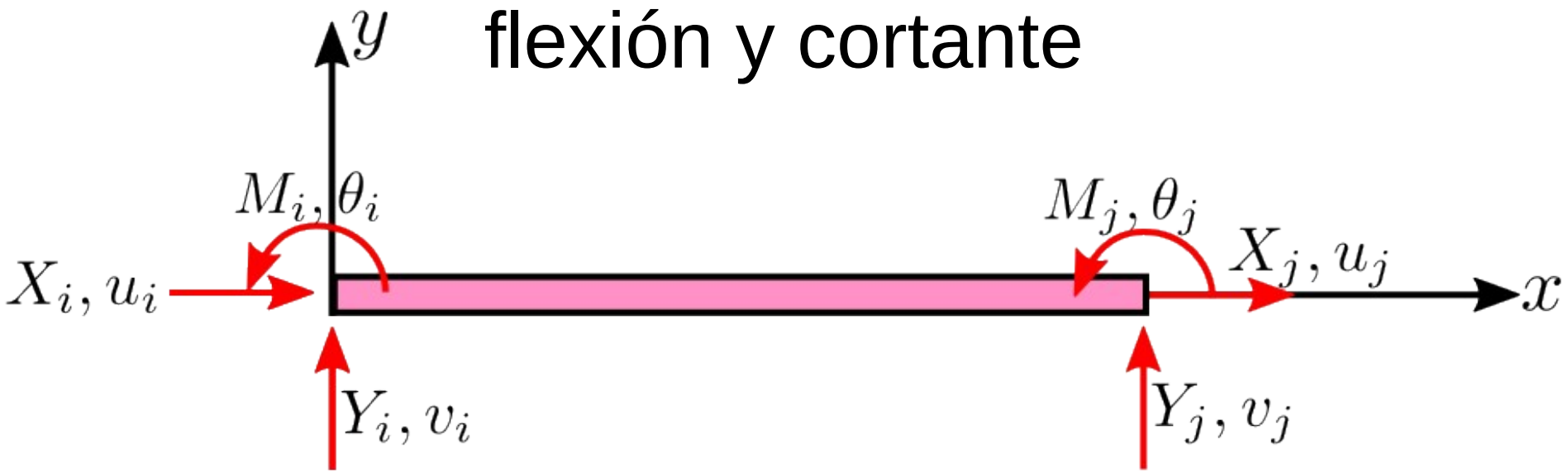




grados de libertad restringidos = 1, 2, 4

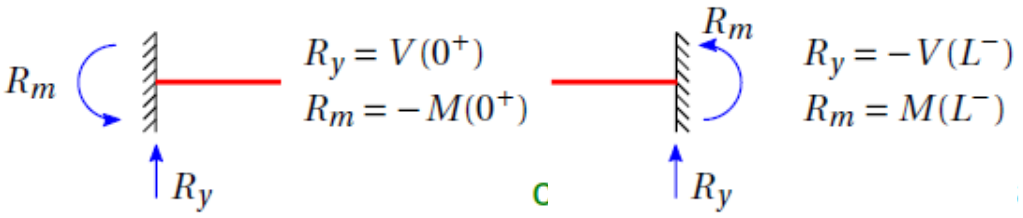
Códigos de MATLAB, PYTHON y OPENSEES en:
https://github.com/diegoandresalvarez/elementosfinitos/tree/master/codigo/repaso_matricial/cercha_2d

Matriz de rigidez de un elemento prismático sometido en sus extremos a carga axial, flexión y cortante



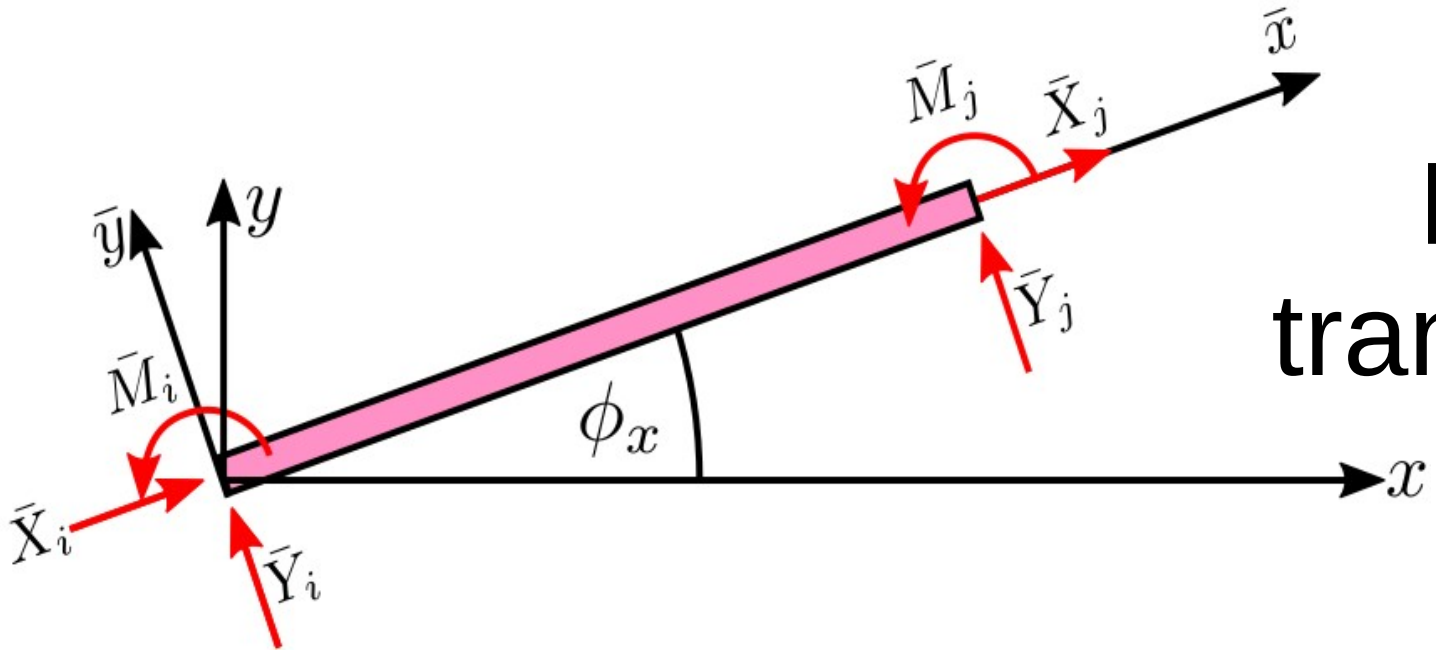
$$\begin{bmatrix} X_i \\ Y_i \\ M_i \\ X_j \\ Y_j \\ M_j \end{bmatrix} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ \theta_i \\ u_j \\ v_j \\ \theta_j \end{bmatrix}$$

```m
% Programa para deducir la matriz de rigidez de un elemento de portico 2D
clear, clc
syms w x L V(x) M(x) t(x) v(x) EI EA

Kflexion = sym(zeros(4));
for i = 1:4
    sol = dsolve(...
          diff(V,x) == 0,     ...
          diff(M,x) == V,     ...
          diff(t,x) == M/EI,  ...
          diff(v,x) == t,     ...
          v(0) == (i==1),     ... % con sus respectivas condiciones de
          t(0) == (i==2),     ... % frontera
          v(L) == (i==3),     ...
          t(L) == (i==4));

    Kflexion(:,i) = [ +subs(sol.V, x, 0)      % Yi  se evaluan las
                      -subs(sol.M, x, 0)      % Mi  reacciones verticales
                      -subs(sol.V, x, L)      % Yj  y los momentos en los
                      +subs(sol.M, x, L) ];   % Mj  apoyos
end
Kaxial = EA/L * [1 -1; -1 1];

K = sym(zeros(6));
K([1 4],[1 4])         = Kaxial;
K([2 3 5 6],[2 3 5 6]) = Kflexion;
K
```




```
K =
[  EA/L,             0,             0, -EA/L,             0,             0]
[     0,  (12*EI)/L^3,  (6*EI)/L^2,     0, -(12*EI)/L^3,  (6*EI)/L^2]
[     0,   (6*EI)/L^2,    (4*EI)/L,     0,  -(6*EI)/L^2,    (2*EI)/L]
[ -EA/L,             0,             0,  EA/L,             0,             0]
[     0, -(12*EI)/L^3, -(6*EI)/L^2,     0,  (12*EI)/L^3, -(6*EI)/L^2]
[     0,   (6*EI)/L^2,    (2*EI)/L,     0,  -(6*EI)/L^2,    (4*EI)/L]
```

https://github.com/diegoandresalvarez/elementosfinitos/blob/master/codigo/repaso_matricial/portico_2d/deduccion_K_y_fe_elemento_portico_2D/deduccion_Ke_portico2D.m

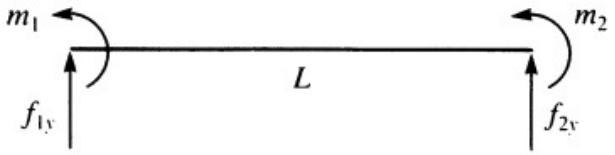


Matriz de transformación

$$
\underbrace{\begin{bmatrix} \bar{X}_i \\ \bar{Y}_i \\ \bar{M}_i \\ \bar{X}_j \\ \bar{Y}_j \\ \bar{M}_j \end{bmatrix}}_{\boldsymbol{f}^{(e)}_{\text{loc}}}
=
\underbrace{\begin{bmatrix}
\cos\phi_x & \sin\phi_x & 0 & 0 & 0 & 0 \\
-\sin\phi_x & \cos\phi_x & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & \cos\phi_x & \sin\phi_x & 0 \\
0 & 0 & 0 & -\sin\phi_x & \cos\phi_x & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}}_{\boldsymbol{T}_{(e)}}
\underbrace{\begin{bmatrix} X_i \\ Y_i \\ M_i \\ X_j \\ Y_j \\ M_j \end{bmatrix}}_{\boldsymbol{f}^{(e)}}
$$

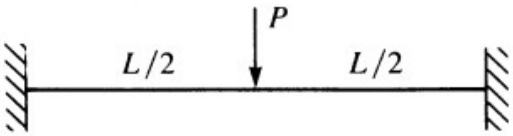
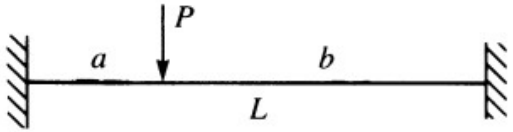
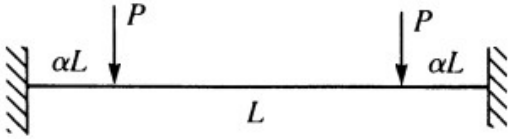
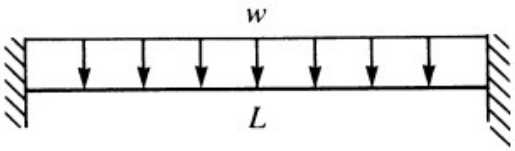
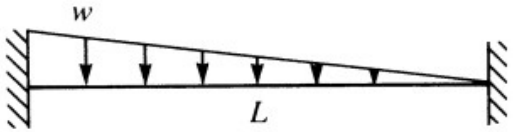
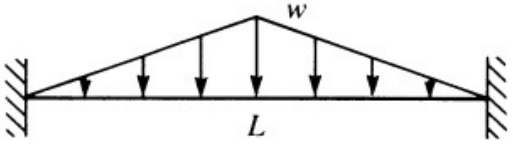
Algunas fuerzas nodales equivalentes

Tabla tomada de:
Daryl L. Logan (2012) - A First Course in the Finite Element Method, 5 ed.

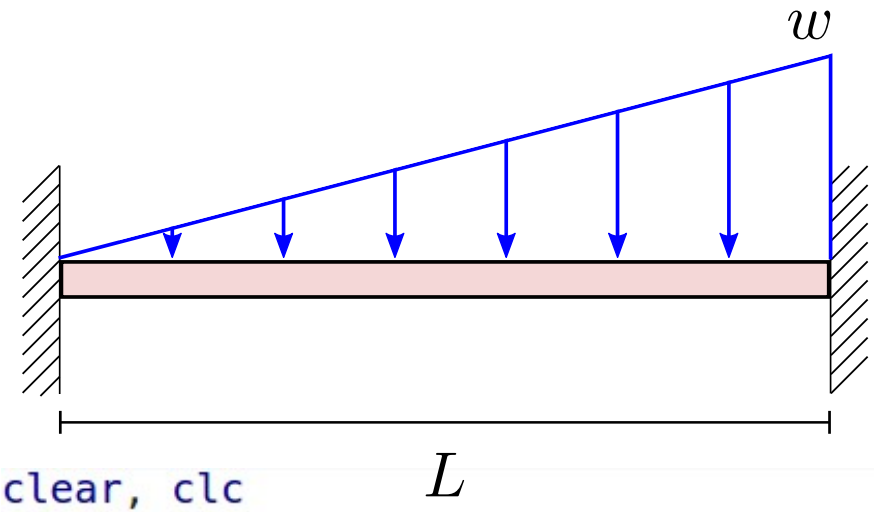


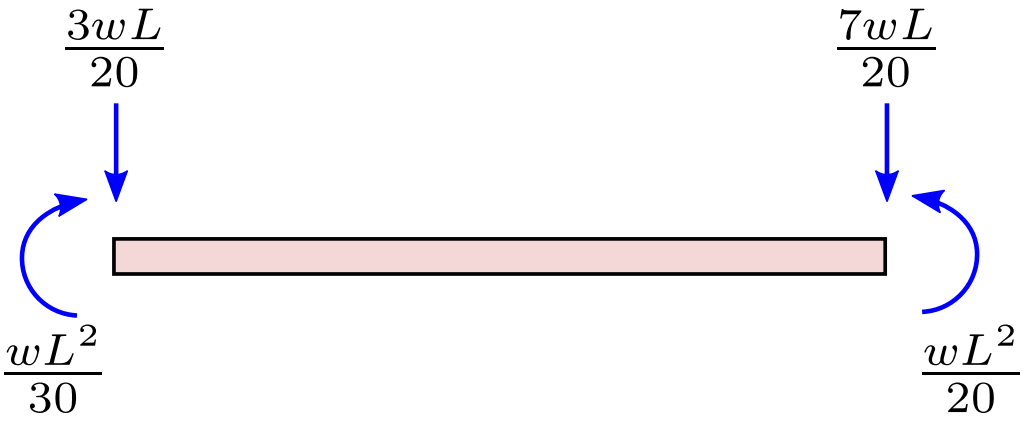
Positive nodal force conventions

Table D–1 Single element equivalent joint forces f_0 for different types of loads

	f_{1y}	m_1	Loading case	f_{2y}	m_2
1.	$\frac{-P}{2}$	$\frac{-PL}{8}$		$\frac{-P}{2}$	$\frac{PL}{8}$
2.	$\frac{-Pb^2(L+2a)}{L^3}$	$\frac{-Pab^2}{L^2}$		$\frac{-Pa^2(L+2b)}{L^3}$	$\frac{Pa^2b}{L^2}$
3.	$-P$	$-\alpha(1-\alpha)PL$		$-P$	$\alpha(1-\alpha)PL$
4.	$\frac{-wL}{2}$	$\frac{-wL^2}{12}$		$\frac{-wL}{2}$	$\frac{wL^2}{12}$
5.	$\frac{-7wL}{20}$	$\frac{-wL^2}{20}$		$\frac{-3wL}{20}$	$\frac{wL^2}{30}$
6.	$\frac{-wL}{4}$	$\frac{-5wL^2}{96}$		$\frac{-wL}{4}$	$\frac{5wL^2}{96}$

Cálculo de fuerzas nodales equivalentes

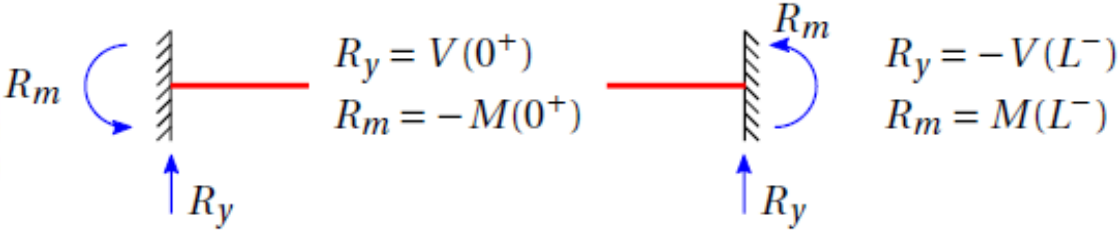





```
clear, clc

syms syms V(x) M(x) t(x) v(x) EI w x L   % se definen las variables y constantes
q = -w*x/L;                              % carga triangular

sol = dsolve(...
      diff(V) == q,    diff(M) == V, ... % se definen las ecuaciones diferenciales
      diff(t) == M/EI, diff(v) == t, ... %
      v(0) == 0, v(L) == 0,          ... % con sus respectivas condiciones
      t(0) == 0, t(L) == 0);             % de frontera.

% Se evaluan las cargas nodales equivalentes
disp('Y1 = '); disp(-subs(sol.V, x, 0));
disp('Y2 = '); disp(+subs(sol.V, x, L));

% Y los momentos nodales equivalentes
disp('M1 = '); disp(+subs(sol.M, x, 0));
disp('M2 = '); disp(-subs(sol.M, x, L));
```

Obteniendo:

```
Y1 = -(3*L*w)/20
Y2 = -(7*L*w)/20
M1 = -(L^2*w)/30
M2 = (L^2*w)/20
```

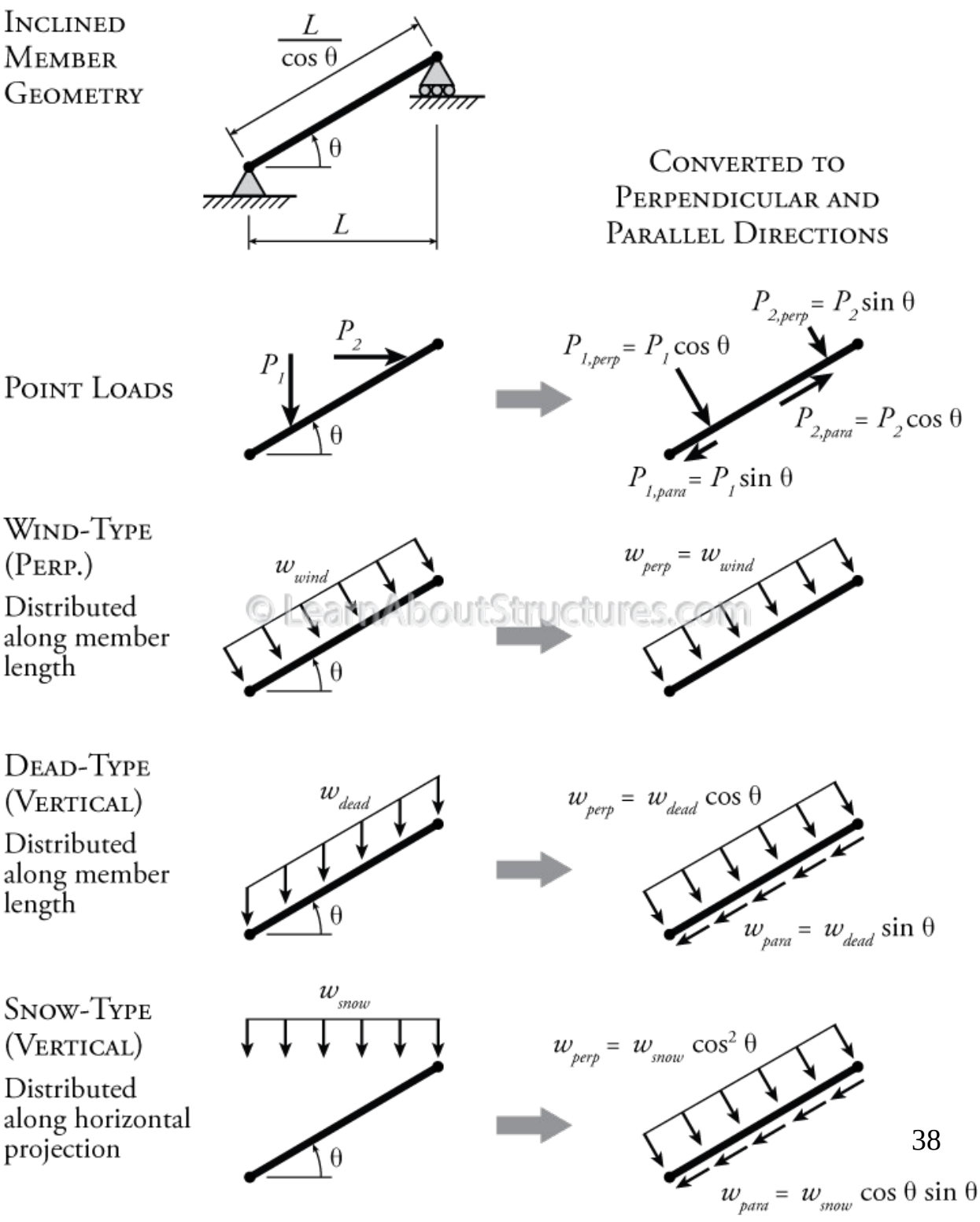


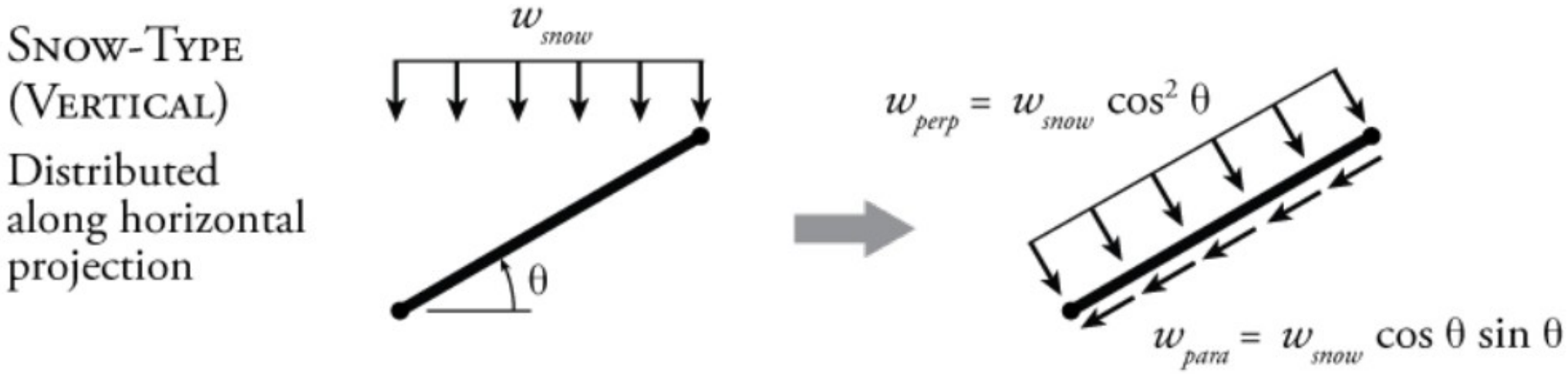
https://github.com/diegoandresalvarez/elementosfinitos/blob/master/codigo/repaso_matricial/portico_2d/deduccion_K_y_fe_elemento_portico_2D/deduccion_fe_carga_triangular.m

Cargas distribuídas inclinadas

http://www.learnaboutstructures.com/Determinate-Frame-Analysis

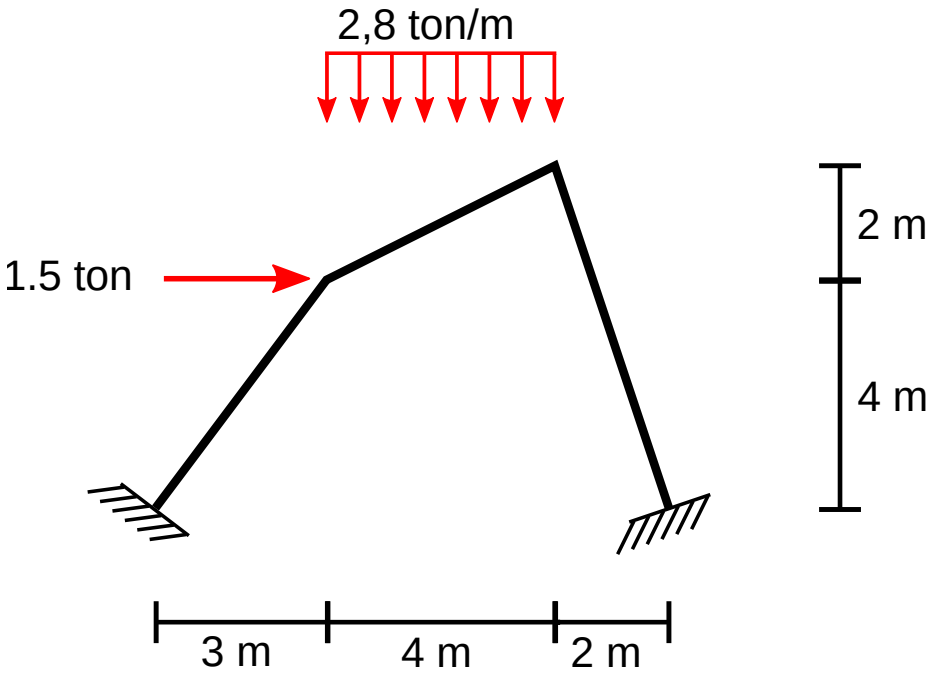
For members that are inclined on an angle, it is often most convenient to analyse them by first transforming all the loads on the member into the local member axis direction (perpendicular and parallel to the inclined member).





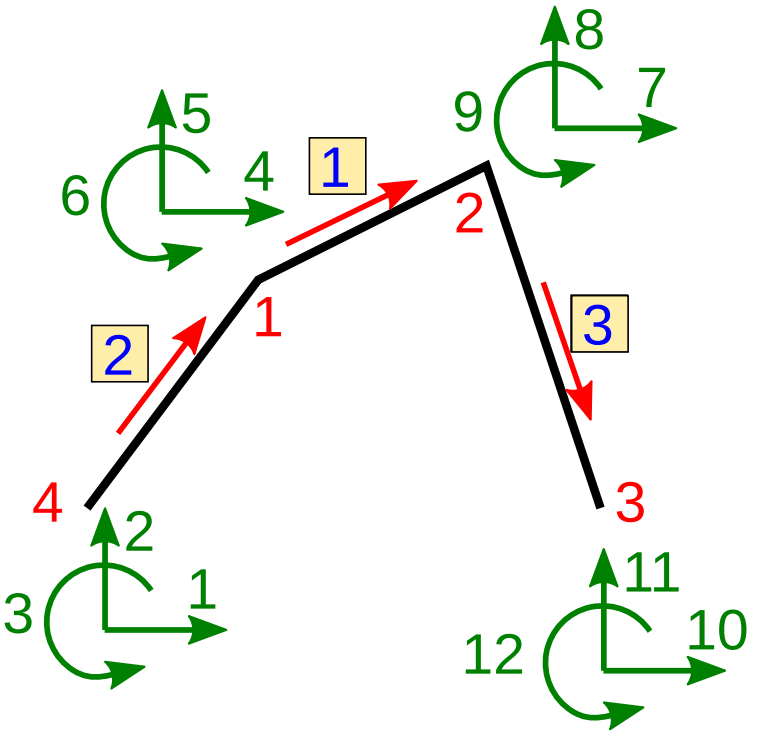
The fourth and final type ('snow-type') is a distributed load which is not perpendicular to the member and is also not distributed along the member length, but along the horizontal projection of the member (in this case, the distance L). For the case of a snow load, only a certain amount of snow can fall in a certain horizontal area, so the greater the inclination of the member, the longer the length that the snow will be spread out over. The total vertical load here will be equal to $w_{snow}L$, which is less than the corresponding total vertical load from the dead load case, which was equal to $w_{dead}L/\cos\theta$. Before the snow-type distributed load can be split into perpendicular and parallel components, it must be spread evenly over the diagonal length of the member (instead of the horizontal projection). If we split the total vertical load $w_{snow}L$ over the total diagonal length $L/\cos\theta$ then we get a new vertical distributed load equal to $w_{snow}\cos\theta$ which is now distributed along the diagonal length of the member. From here, the load may be distributed into perpendicular and parallel components as was done for the dead-type load, which results in the components shown in Figure 4.7.

Ejemplo 11.23 Uribe Escamilla



Dimensiones:
Vigas: 30cm x 35cm
Columnas: 30cm x 30cm

E = 190 ton/cm^2



grados de libertad
restringidos = 1, 2, 3, 10, 11, 12

Ver solución con MATLAB y PYTHON en:
https://github.com/diegoandresalvarez/elementosfinitos/tree/master/codigo/repaso_matricial/portico_2d

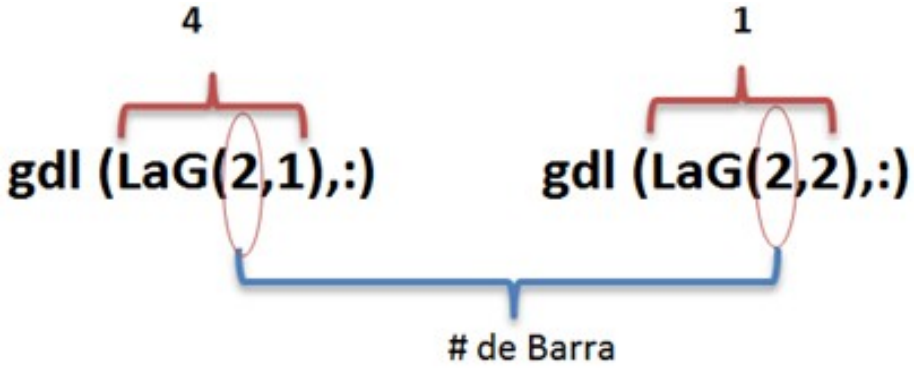
LaG

# Barra	Nodo 1 Loc	Nodo 2 Loc
1	1	2
2	4	1
3	2	3

gdl

nodo	x	y	rot
1	4	5	6
2	7	8	9
3	10	11	12
4	1	2	3

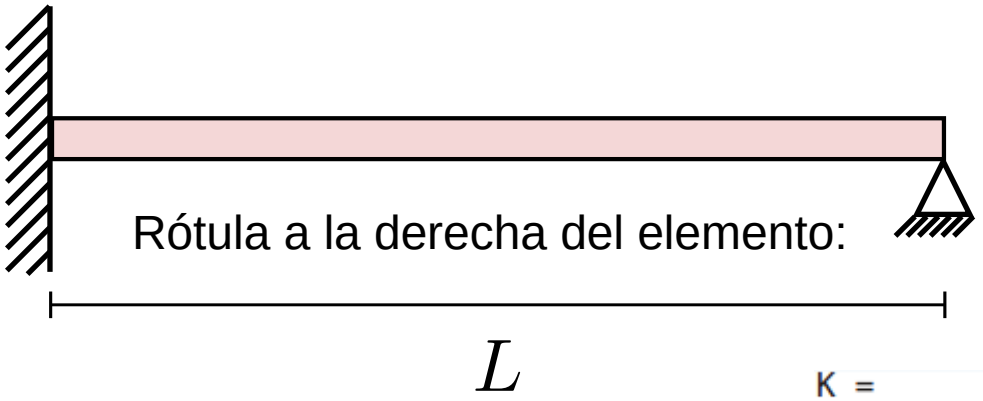
Ejemplo: idx para la barra 2:



idx

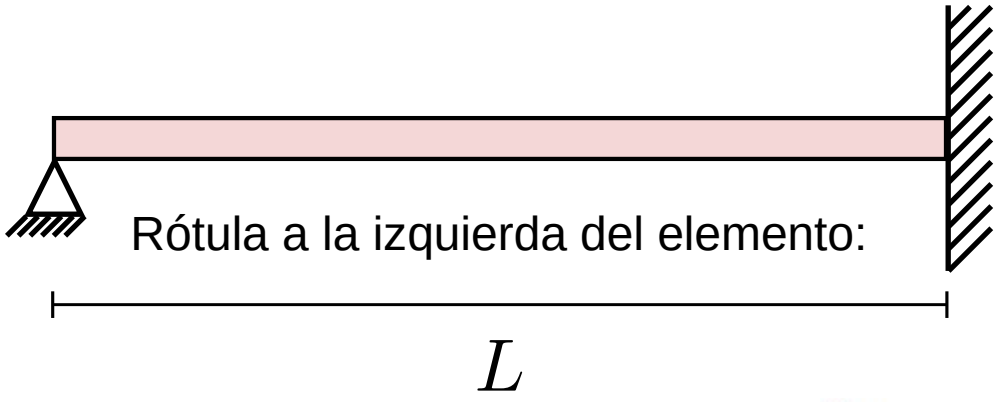
Barra	gdl 1	gdl2	gdl3	gdl4	gdl5	gdl6
1	4	5	6	7	8	9
2	1	2	3	4	5	6
3	7	8	9	10	11	12

Rótulas interiores a un pórtico

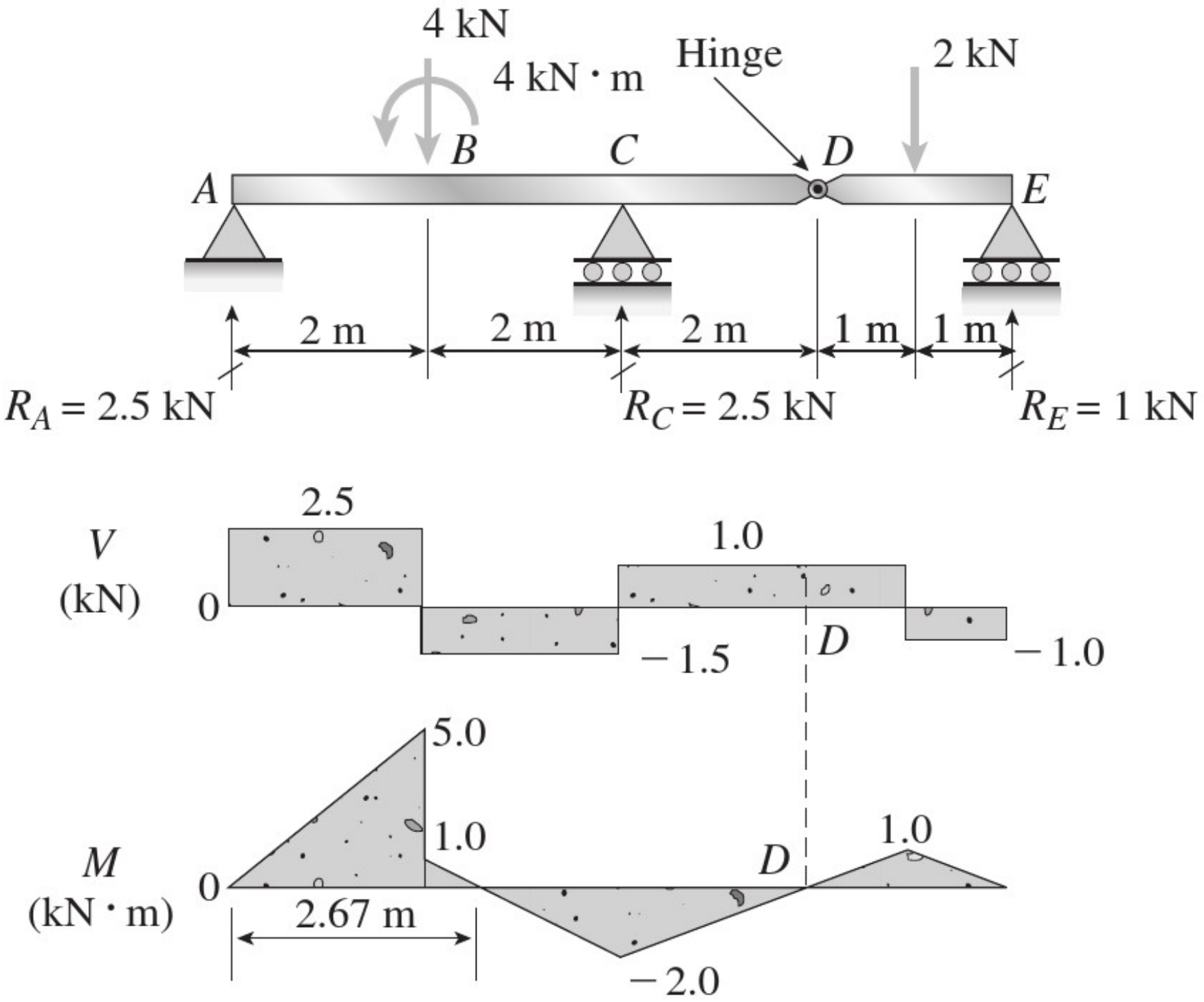



```
K =

[  AE/L,           0,           0, -AE/L,           0]  ->  u1
[     0,  (3*EI)/L^3,  (3*EI)/L^2,     0, -(3*EI)/L^3]  ->  v1
[     0,  (3*EI)/L^2,    (3*EI)/L,     0, -(3*EI)/L^2]  ->  t1
[ -AE/L,           0,           0,  AE/L,           0]  ->  u2
[     0, -(3*EI)/L^3, -(3*EI)/L^2,     0,  (3*EI)/L^3]  ->  v2
```

Ver programas en:
https://github.com/diegoandresalvarez/elementosfinitos/tree/master/codigo/repaso_matricial/empotrado_viga_rotula




```
K =

[  AE/L,           0, -AE/L,           0,           0]  ->  u1
[     0,  (3*EI)/L^3,     0, -(3*EI)/L^3,  (3*EI)/L^2]  ->  v1
[ -AE/L,           0,  AE/L,           0,           0]  ->  u2
[     0, -(3*EI)/L^3,     0,  (3*EI)/L^3, -(3*EI)/L^2]  ->  v2
[     0,  (3*EI)/L^2,     0, -(3*EI)/L^2,    (3*EI)/L]  ->  t2
```



Ver solución en:
https://github.com/diegoandresalvarez/elementosfinitos/blob/master/codigo/repaso_matricial/empotrado_viga_rotula/ejemplo_empotrado_viga_rotula1.m

Rotación de los grados de libertad

Un nodo i puede tener dos sistemas de coordenadas un global $\boldsymbol{x}$ y otro local $\boldsymbol{x}'$. Los desplazamientos en esos sistemas de coordenadas se relacionan como:

$$\underbrace{\begin{bmatrix} u_i' \\ v_i' \\ \phi_i' \end{bmatrix}}_{\boldsymbol{u}_i'} = \underbrace{\begin{bmatrix} \cos\theta_i & \sin\theta_i & 0 \\ -\sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{T}_i} \underbrace{\begin{bmatrix} u_i \\ v_i \\ \phi_i \end{bmatrix}}_{\boldsymbol{u}_i}$$

donde $\boldsymbol{T}_i$ es la *matriz de rotación asociada al nodo* i.

Como cada nodo puede tener un sistema de coordenadas diferentes, los vectores globales de desplazamiento $\boldsymbol{a}$ y $\boldsymbol{a}'$ (en coordenadas globales y locales respectivamente) se relacionan por:

$$\boldsymbol{a}' = \boldsymbol{T}\boldsymbol{a}$$

donde

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}_1 & & & \boldsymbol{0} \\ & \boldsymbol{T}_2 & & \\ & & \ddots & \\ \boldsymbol{0} & & & \boldsymbol{T}_n \end{bmatrix}$$

siendo n el número de nodos de la estructura.

Finalmente, la ecuación de equilibrio global se convierte en:

$$\boldsymbol{q}' = \boldsymbol{T}\boldsymbol{K}\boldsymbol{T}^T\boldsymbol{a}' - \boldsymbol{f}'.$$

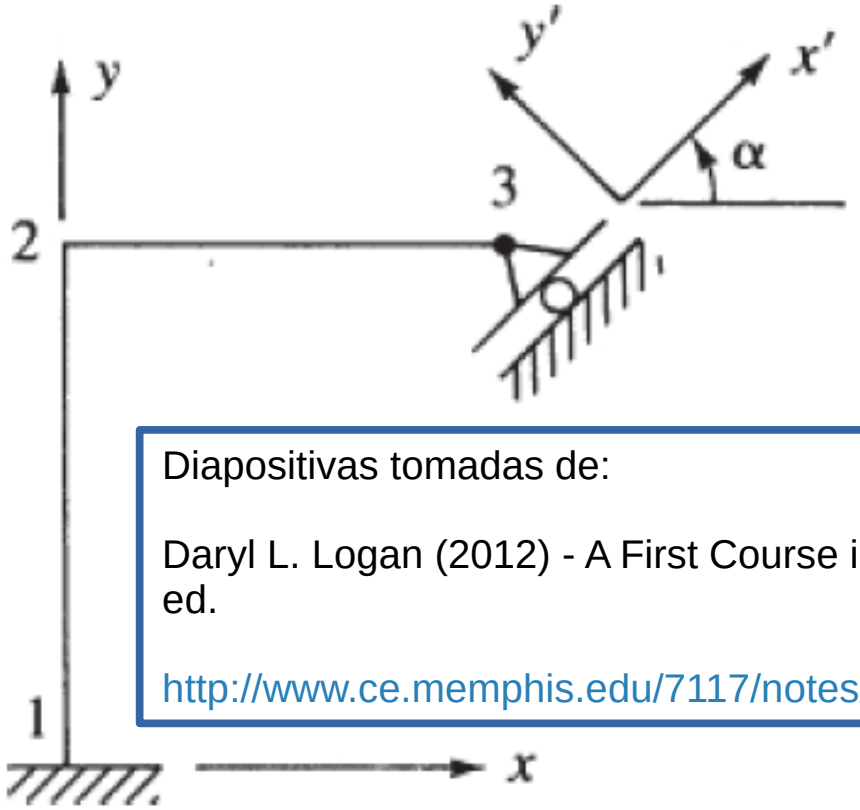
donde

$$\boldsymbol{q}' = \boldsymbol{T}\boldsymbol{q} \qquad \boldsymbol{a}' = \boldsymbol{T}\boldsymbol{a} \qquad \boldsymbol{f}' = \boldsymbol{T}\boldsymbol{f}.$$

Inclined or Skewed Supports

If a support is inclined, or skewed, at some angle α for the global x axis, as shown below.

The boundary conditions on the displacements are not in the global x-y directions but in the x'-y' directions.



Diapositivas tomadas de:

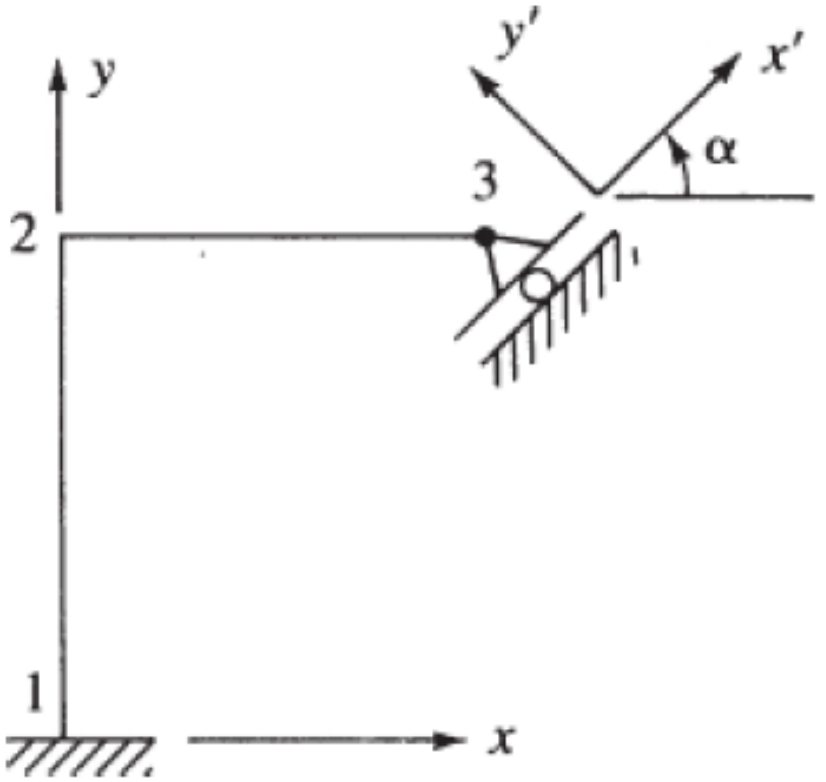
Daryl L. Logan (2012) - A First Course in the Finite Element Method, 5 ed.

http://www.ce.memphis.edu/7117/notes/presentations/chapter_05a.pdf

Plane Frame and Grid Equations

Inclined or Skewed Supports

We must transform the local boundary condition of $v'_3 = 0$ (in local coordinates) into the global x-y system.



Plane Frame and Grid Equations

Inclined or Skewed Supports

Therefore, the relationship between of the components of the displacement in the local and the global coordinate systems at node 3 is:

$$\begin{Bmatrix} u'_3 \\ v'_3 \\ \phi'_3 \end{Bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} u_3 \\ v_3 \\ \phi_3 \end{Bmatrix}$$

We can rewrite the above expression as:

$$\{d'_3\} = [t_3]\{d_3\} \qquad [t_3] = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Plane Frame and Grid Equations

Inclined or Skewed Supports

We can apply this sort of transformation to the entire displacement vector as:

$$\{d'\} = [T_i]\{d\} \qquad \text{or} \qquad \{d\} = [T_i]^T \{d'\}$$

where the matrix $[T_i]$ is:

$$[T_i] = \begin{bmatrix} [I] & [0] & [0] \\ [0] & [I] & [0] \\ [0] & [0] & [t_3] \end{bmatrix}$$

Both the identity matrix $[I]$ and the matrix $[t_3]$ are 3 x 3 matrices.

Plane Frame and Grid Equations

Inclined or Skewed Supports

The force vector can be transformed by using the same transformation.

$$\{f'\} = [T_i]\{f\}$$

In global coordinates, the force-displacement equations are:

$$\{f\} = [K]\{d\}$$

Applying the skewed support transformation to both sides of the force-displacement equation gives:

$$[T_i]\{f\} = [T_i][K]\{d\}$$

By using the relationship between the local and the global displacements, the force-displacement equations become:

$$[T_i]\{f\} = [T_i][K][T_i]^T\{d'\} \quad \Rightarrow \quad \{f'\} = [T_i][K][T_i]^T\{d'\}$$

Plane Frame and Grid Equations

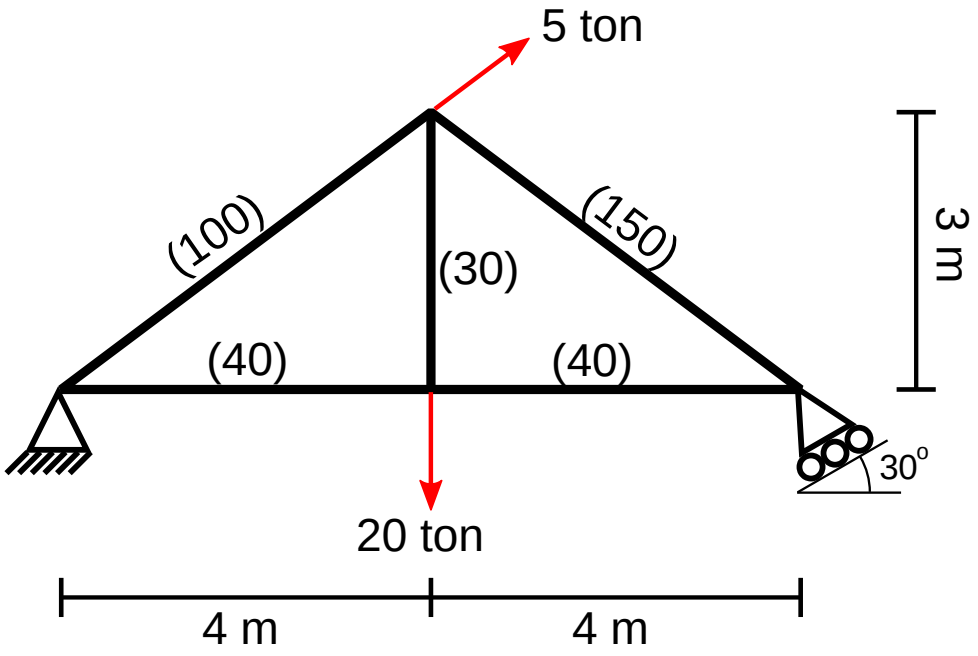
Inclined or Skewed Supports

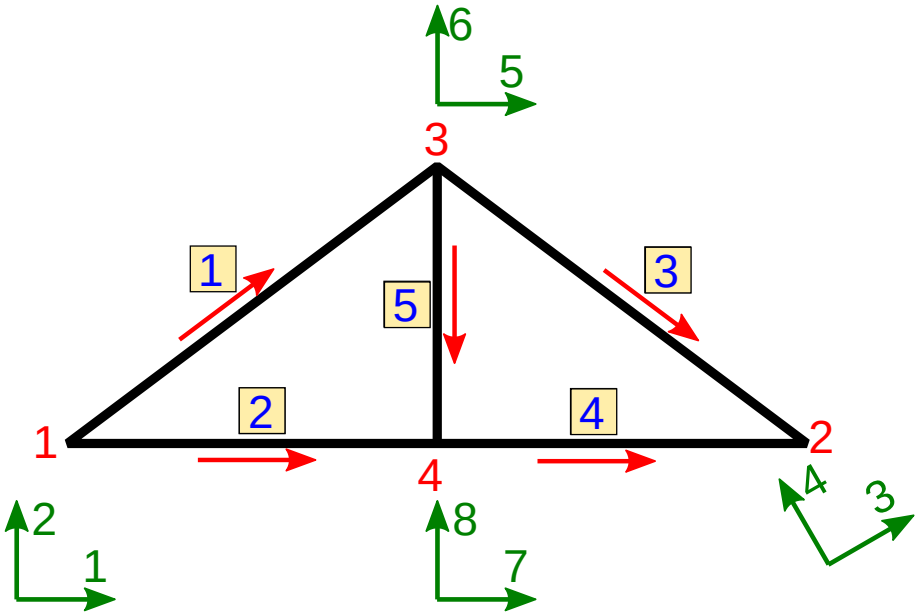
Therefore the global equations become:

$$\begin{Bmatrix} F_{1x} \\ F_{1y} \\ M_1 \\ F_{2x} \\ F_{2y} \\ M_2 \\ F'_{3x} \\ F'_{3y} \\ M_3 \end{Bmatrix} = [T_i][K][T_i]^T \begin{Bmatrix} u_1 \\ v_1 \\ \phi_1 \\ u_2 \\ v_2 \\ \phi_2 \\ u'_3 \\ v'_3 \\ \phi_3 \end{Bmatrix}$$

EJEMPLO

Resuelva completamente la estructura mostrada.
El material es acero: E = 2040 ton/cm^2
Las áreas están entre paréntesis en cm^2.
El apoyo se encuentra inclinado 30 grados.

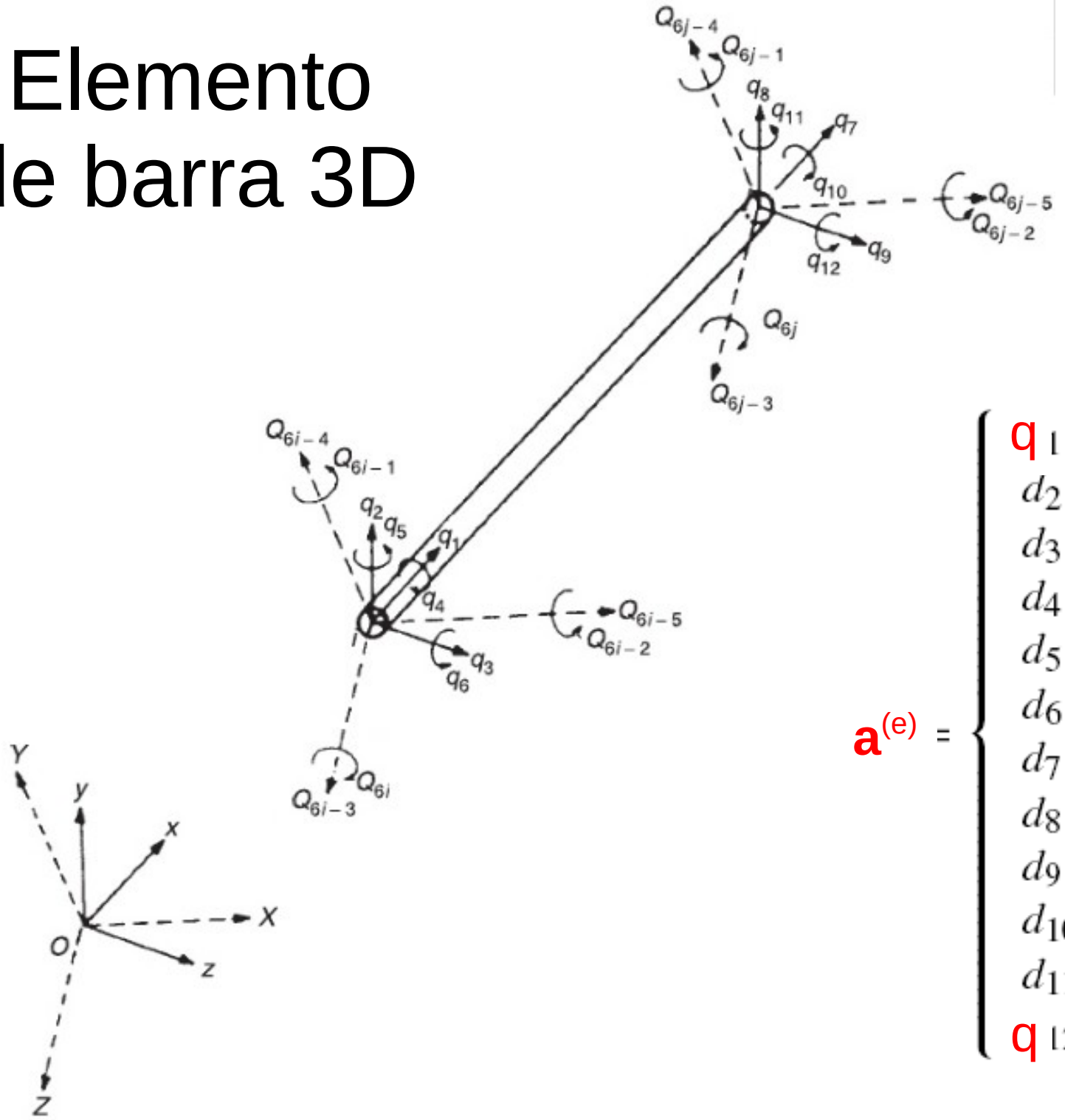




grados de libertad restringidos = 1, 2, 4

https://github.com/diegoandresalvarez/elementosfinitos/blob/master/codigo/repaso_matricial/cercha_2d/ejemplo_cercha_apoyo_inclinado.m

Elemento de barra 3D



$$\mathbf{a}^{(e)} = \begin{Bmatrix} q_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \\ d_8 \\ d_9 \\ d_{10} \\ d_{11} \\ q_{12} \end{Bmatrix} = \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ \theta_{x1} \\ \theta_{y1} \\ \theta_{z1} \\ u_2 \\ v_2 \\ w_2 \\ \theta_{x2} \\ \theta_{y2} \\ \theta_{z2} \end{Bmatrix}$$

Elemento de barra 3D

$[k^{(e)}]$
12×12

$$
\begin{array}{c}
\begin{matrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 & q_9 & q_{10} & q_{11} & q_{12} \end{matrix} \\
= \left\{ \begin{array}{cccccccccccc}
\frac{EA}{l} & & & & & & & & & & & \\
0 & \frac{12EI_{zz}}{l^3} & & & & & & & & & & \\
0 & 0 & \frac{12EI_{yy}}{l^3} & & & & & & \text{Symmetric} & & & \\
0 & 0 & 0 & \frac{GJ}{l} & & & & & & & & \\
0 & 0 & \frac{-6EI_{yy}}{l^2} & 0 & \frac{4EI_{yy}}{l} & & & & & & & \\
0 & \frac{6EI_{zz}}{l^2} & 0 & 0 & 0 & \frac{4EI_{zz}}{l} & & & & & & \\
-\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{l} & & & & & \\
0 & \frac{-12EI_{zz}}{l^3} & 0 & 0 & 0 & \frac{-6EI_{zz}}{l^2} & 0 & \frac{12EI_{zz}}{l^3} & & & & \\
0 & 0 & \frac{-12EI_{yy}}{l^3} & 0 & \frac{6EI_{yy}}{l^2} & 0 & 0 & 0 & \frac{12EI_{yy}}{l^3} & & & \\
0 & 0 & 0 & \frac{-GJ}{l} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{l} & & \\
0 & 0 & \frac{-6EI_{yy}}{l^2} & 0 & \frac{2EI_{yy}}{l} & 0 & 0 & 0 & \frac{6EI_{yy}}{l^2} & 0 & \frac{4EI_{yy}}{l} & \\
0 & \frac{6EI_{zz}}{l^2} & 0 & 0 & 0 & \frac{2EI_{zz}}{l} & 0 & \frac{-6EI_{zz}}{l^2} & 0 & 0 & 0 & \frac{4EI_{zz}}{l}
\end{array} \right\}
\begin{matrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \\ q_9 \\ q_{10} \\ q_{11} \\ q_{12} \end{matrix}
\end{array}
$$

Matriz de transformación de coordenadas $\boldsymbol{T}^{(e)}$

$$\begin{Bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \\ q_9 \\ q_{10} \\ q_{11} \\ q_{12} \end{Bmatrix} = \begin{bmatrix} l_{ox} & m_{ox} & n_{ox} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l_{oy} & m_{oy} & n_{oy} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l_{oz} & m_{oz} & n_{oz} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & l_{ox} & m_{ox} & n_{ox} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & l_{oy} & m_{oy} & n_{oy} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & l_{oz} & m_{oz} & n_{oz} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & l_{ox} & m_{ox} & n_{ox} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & l_{oy} & m_{oy} & n_{oy} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & l_{oz} & m_{oz} & n_{oz} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & l_{ox} & m_{ox} & n_{ox} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & l_{oy} & m_{oy} & n_{oy} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & l_{oz} & m_{oz} & n_{oz} \end{bmatrix} \begin{Bmatrix} Q_{6i-5} \\ Q_{6i-4} \\ Q_{6i-3} \\ Q_{6i-2} \\ Q_{6i-1} \\ Q_{6i} \\ Q_{6j-5} \\ Q_{6j-4} \\ Q_{6j-3} \\ Q_{6j-2} \\ Q_{6j-1} \\ Q_{6j} \end{Bmatrix}$$

$$\boldsymbol{f}_{\text{loc}}^{(e)} = \boldsymbol{T}^{(e)} \boldsymbol{f}^{(e)}$$

$$\underset{12 \times 12}{\boldsymbol{T}^{(e)}} = \begin{bmatrix} [\lambda] & [0] & [0] & [0] \\ [0] & [\lambda] & [0] & [0] \\ [0] & [0] & [\lambda] & [0] \\ [0] & [0] & [0] & [\lambda] \end{bmatrix} \qquad \underset{3 \times 3}{[\lambda]} = \begin{bmatrix} l_{ox} & m_{ox} & n_{ox} \\ l_{oy} & m_{oy} & n_{oy} \\ l_{oz} & m_{oz} & n_{oz} \end{bmatrix} \qquad \underset{3 \times 3}{[0]} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Here, l_{ox}, m_{ox}, and n_{ox} denote the direction cosines of the x axis (line ij in Figure 9.8); l_{oy}, m_{oy}, and n_{oy} represent the direction cosines of the y axis; and l_{oz}, m_{oz}, and n_{oz} indicate the direction cosines of the z axis with respect to the global X, Y, Z axes. It can be seen that finding the direction cosines of the x axis is a straightforward computation since

$$l_{ox} = \frac{X_j - X_i}{l}, \qquad m_{ox} = \frac{Y_j - Y_i}{l}, \qquad n_{ox} = \frac{Z_j - Z_i}{l} \tag{9.44}$$

where X_k, Y_k, Z_k indicate the coordinates of node k $(k = i, j)$ in the global system. However, the computation of the direction cosines of the y and z axes requires some special effort. Finally, the stiffness matrix of the element with reference to the global coordinate system can be obtained as

$$[K^{(e)}] = [\lambda]^T [k^{(e)}][\lambda] \tag{9.45}$$

$$\boldsymbol{K}^{(e)} = \boldsymbol{T}_{(e)}^T \boldsymbol{K}_{\text{loc}}^{(e)} \boldsymbol{T}_{(e)}$$

Tomado de Liu&Quek, p118

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{T}_3 \end{bmatrix}$$

in which

$$\mathbf{T}_3 = \begin{bmatrix} l_x & m_x & n_x \\ l_y & m_y & n_y \\ l_z & m_z & n_z \end{bmatrix}$$

where l_k, m_k and n_k $(k = x, y, z)$ are direction cosines defined by

$$l_x = \cos(x, X), \quad m_x = \cos(x, Y), \quad n_x = \cos(x, Z)$$
$$l_y = \cos(y, X), \quad m_y = \cos(y, Y), \quad n_y = \cos(y, Z)$$
$$l_z = \cos(z, X), \quad m_z = \cos(z, Y), \quad n_z = \cos(z, Z)$$

x,y,z = ejes locales X,Y,Z = ejes globales

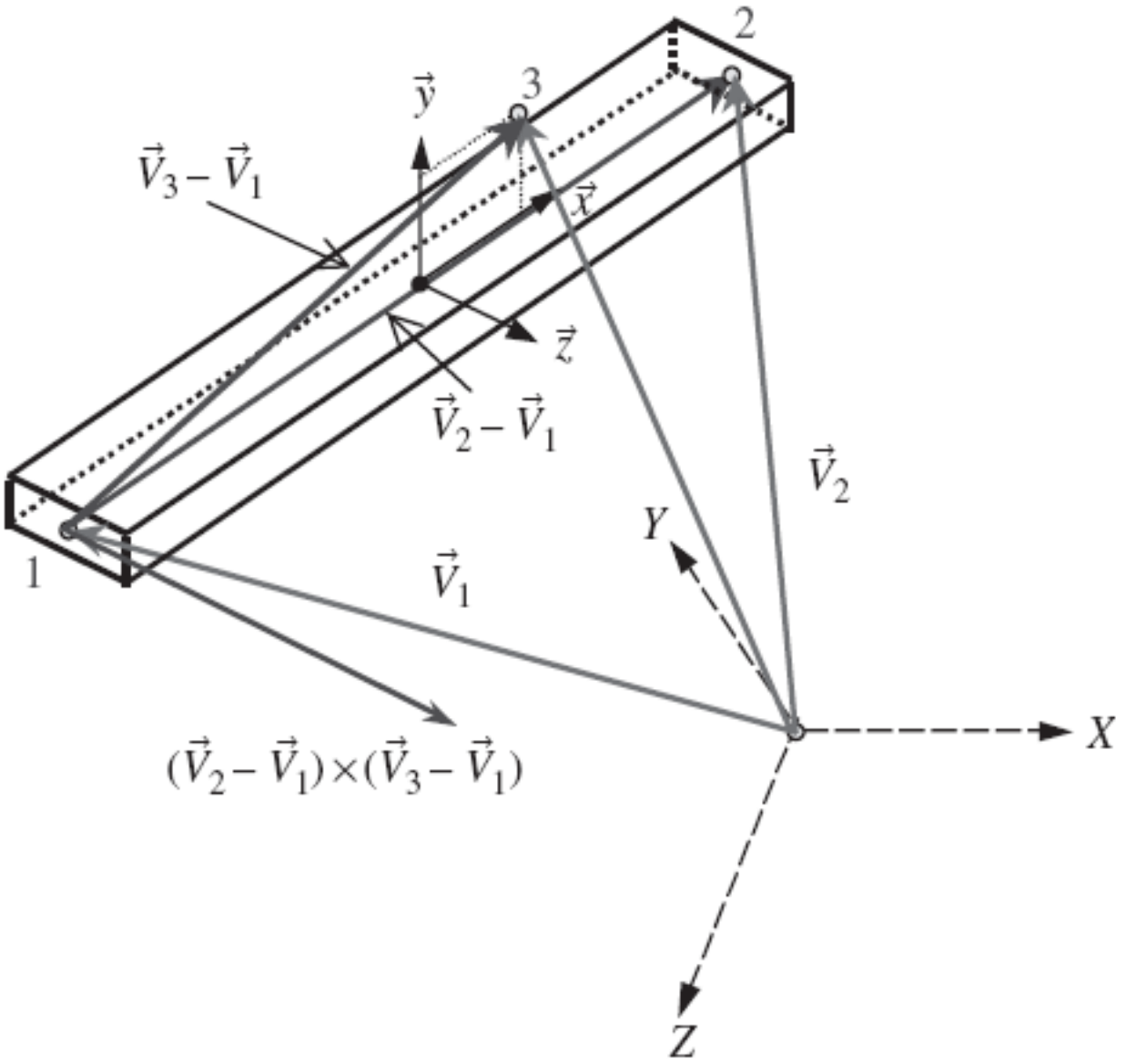
To define these direction cosines, the position and the three-dimensional orientation of the frame element have to be defined first. With nodes 1 and 2, the location of the element is fixed on the local coordinate frame, and the orientation of the element has also been fixed in the x direction. However, the local coordinate frame can still rotate about the axis of the beam. One more additional point in the local coordinate has to be defined. This point can be chosen anywhere in the local x–y plane, but not on the x-axis. Therefore, node 3 is chosen, as shown in Figure 6.6.

The position vectors $\vec{V}_1$, $\vec{V}_2$ and $\vec{V}_3$ can be expressed as

$$\vec{V}_1 = X_1\vec{X} + Y_1\vec{Y} + Z_1\vec{Z} \tag{6.25}$$

$$\vec{V}_2 = X_2\vec{X} + Y_2\vec{Y} + Z_2\vec{Z} \tag{6.26}$$

$$\vec{V}_3 = X_3\vec{X} + Y_3\vec{Y} + Z_3\vec{Z} \tag{6.27}$$


2
3
1
$\vec{y}$
$\vec{x}$
$\vec{z}$
$\vec{V}_3 - \vec{V}_1$
$\vec{V}_2 - \vec{V}_1$
$\vec{V}_1$
$\vec{V}_2$
$(\vec{V}_2 - \vec{V}_1) \times (\vec{V}_3 - \vec{V}_1)$
X
Y
Z

where X_k, Y_k and Z_k ($k = 1, 2, 3$) are the coordinates for node k, and $\vec{X}$, $\vec{Y}$ and $\vec{Z}$ are unit vectors along the X, Y and Z axes. We now define

$$\left.\begin{aligned} X_{kl} &= X_k - X_l \\ Y_{kl} &= Y_k - Y_l \\ Z_{kl} &= Z_k - Z_l \end{aligned}\right\} \quad k, l = 1, 2, 3 \tag{6.28}$$

Vectors $(\vec{V}_2 - \vec{V}_1)$ and $(\vec{V}_3 - \vec{V}_1)$ can thus be obtained using Eqs. (6.25) to (6.28) as follows:

$$\vec{V}_2 - \vec{V}_1 = X_{21}\vec{X} + Y_{21}\vec{Y} + Z_{21}\vec{Z} \tag{6.29}$$

$$\vec{V}_3 - \vec{V}_1 = X_{31}\vec{X} + Y_{31}\vec{Y} + Z_{31}\vec{Z} \tag{6.30}$$

The length of the frame element can be obtained by

$$l_e = 2a = \left|\vec{V}_2 - \vec{V}_1\right| = \sqrt{X_{21}^2 + Y_{21}^2 + Z_{21}^2} \tag{6.31}$$

The unit vector along the x-axis can thus be expressed as

$$\vec{x} = \frac{(\vec{V}_2 - \vec{V}_1)}{\left|\vec{V}_2 - \vec{V}_1\right|} = \frac{X_{21}}{2a}\vec{X} + \frac{Y_{21}}{2a}\vec{Y} + \frac{Z_{21}}{2a}\vec{Z} \tag{6.32}$$

Therefore, the direction cosines in the x direction are given as

$$\begin{aligned} l_x &= \cos(x, X) = \vec{x} \cdot \vec{X} = \frac{X_{21}}{2a} \\ m_x &= \cos(x, Y) = \vec{x} \cdot \vec{Y} = \frac{Y_{21}}{2a} \\ n_x &= \cos(x, Z) = \vec{x} \cdot \vec{Z} = \frac{Z_{21}}{2a} \end{aligned} \tag{6.33}$$

From Figure 6.6, it can be seen that the direction of the z-axis can be defined by the cross product of vectors $(\vec{V}_2 - \vec{V}_1)$ and $(\vec{V}_3 - \vec{V}_1)$. Hence, the unit vector along the z-axis can be expressed as

$$\begin{aligned} \vec{z} &= \frac{(\vec{V}_2 - \vec{V}_1) \times (\vec{V}_3 - \vec{V}_1)}{\left|(\vec{V}_2 - \vec{V}_1) \times (\vec{V}_3 - \vec{V}_1)\right|} \\ &= (l_z, m_z, n_z) \end{aligned} \tag{6.34}$$

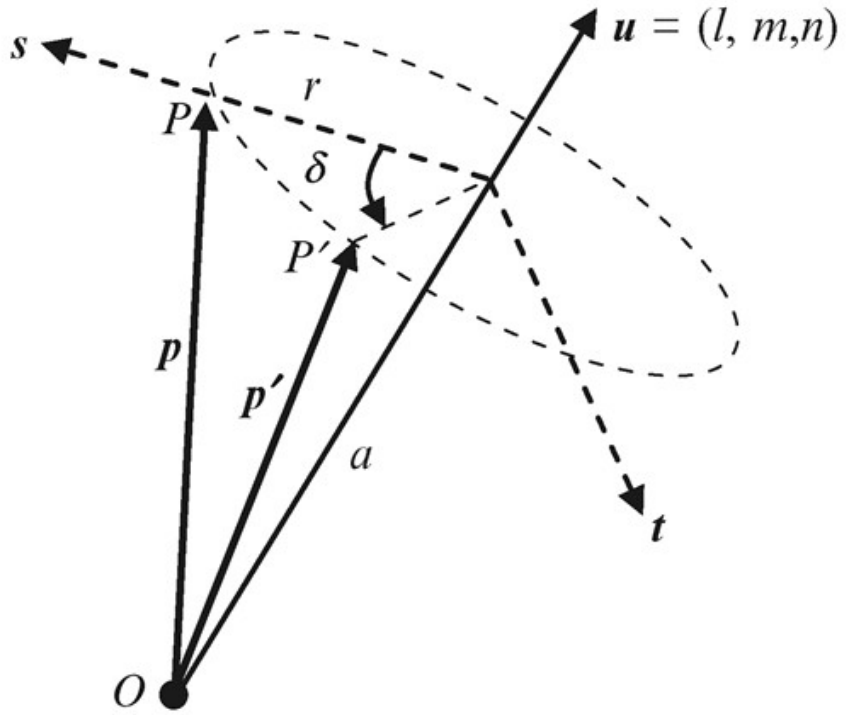
Since the y-axis is perpendicular to both the x-axis and the z-axis, the unit vector along the y-axis can be obtained by cross product,

$$\vec{y} = \vec{z} \times \vec{x} \ = (l_y, m_y, n_y) \tag{6.38}$$

Fórmula de rotación de Rodrigues

Sea $\boldsymbol{p}$ un vector en $\mathbb{R}^3$ y $\hat{\boldsymbol{u}}$ un vector unitario que describe el eje de rotación sobre el cual se quiere rotar $\boldsymbol{p}$ un ángulo δ (en el sentido de la mano derecha). La fórmula de Rodrigues sirve para calcular la posición del vector rotado $\boldsymbol{p}'$:

$$\boldsymbol{p}' = \boldsymbol{p}\cos\delta + (\hat{\boldsymbol{u}} \times \boldsymbol{p})\sin\delta + \hat{\boldsymbol{u}}(\hat{\boldsymbol{u}} \cdot \boldsymbol{p})(1 - \cos\delta).$$



Condensación nodal
(Guyan reduction)

Se utiliza para eliminar GDL innecesarios en nuestros cálculos, ya que no nos interesa los desplazamientos en dichos GDL.

$r \rightarrow$ lo asociaremos a los GDL a retener

$e \rightarrow$ lo asociaremos a los GDL a eliminar

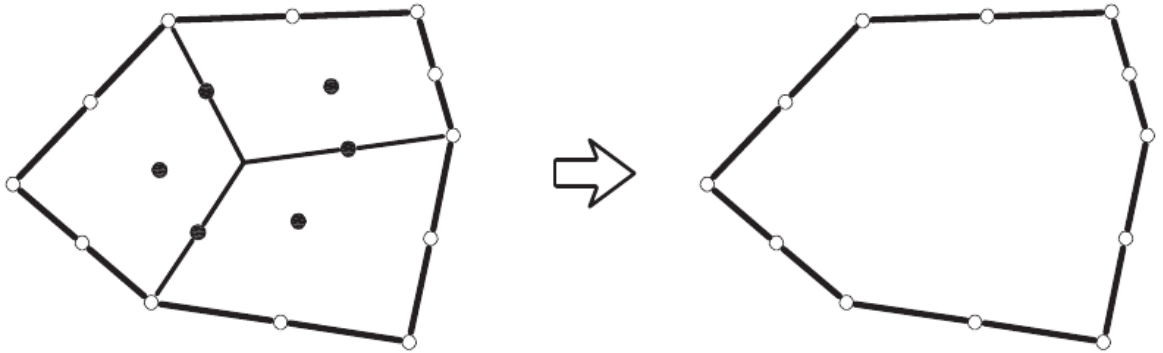


Fig. 9.7 Condensation of internal nodes in a quadrilateral mesh

$$\begin{bmatrix} \boldsymbol{q}_r \\ \boldsymbol{q}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}_{rr} & \boldsymbol{K}_{re} \\ \boldsymbol{K}_{er} & \boldsymbol{K}_{ee} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_r \\ \boldsymbol{a}_e \end{bmatrix} - \begin{bmatrix} \boldsymbol{f}_r \\ \boldsymbol{f}_e \end{bmatrix}$$

Condensación nodal

$$\begin{bmatrix} \boldsymbol{q}_r \\ \boldsymbol{q}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}_{rr} & \boldsymbol{K}_{re} \\ \boldsymbol{K}_{er} & \boldsymbol{K}_{ee} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_r \\ \boldsymbol{a}_e \end{bmatrix} - \begin{bmatrix} \boldsymbol{f}_r \\ \boldsymbol{f}_e \end{bmatrix}$$

$$\boldsymbol{a}_e = \boldsymbol{K}_{ee}^{-1} \left(\boldsymbol{q}_e + \boldsymbol{f}_e - \boldsymbol{K}_{er} \boldsymbol{a}_r \right)$$

Recuerde que $\boldsymbol{K}_{er} = \boldsymbol{K}_{re}^T$

$$\begin{aligned} \boldsymbol{q}_r &= \boldsymbol{K}_{rr}\boldsymbol{a}_r + \boldsymbol{K}_{re}\boldsymbol{a}_e - \boldsymbol{f}_r \\ &= \boldsymbol{K}_{rr}\boldsymbol{a}_r + \boldsymbol{K}_{re}\boldsymbol{K}_{ee}^{-1} \left(\boldsymbol{q}_e + \boldsymbol{f}_e - \boldsymbol{K}_{er}\boldsymbol{a}_r \right) - \boldsymbol{f}_r \\ &= \underbrace{\left(\boldsymbol{K}_{rr} - \boldsymbol{K}_{re}\boldsymbol{K}_{ee}^{-1}\boldsymbol{K}_{er} \right)}_{\overline{\boldsymbol{K}}_r} \boldsymbol{a}_r - \underbrace{\left(\boldsymbol{f}_r - \boldsymbol{K}_{re}\boldsymbol{K}_{ee}^{-1} \left(\boldsymbol{q}_e + \boldsymbol{f}_e \right) \right)}_{\overline{\boldsymbol{f}}_r} \end{aligned}$$

GUYAN, R. J. (1965). *Reduction of stiffness and mass matrices*. AIAA Journal, 3(2), 380-380. https://doi.org/10.2514/3.2874

Condensación nodal

El EF condensado se trata como un EF cualquiera y $\overline{\boldsymbol{K}}_r$ y $\overline{\boldsymbol{f}}_r$ se ensamblan en el sistema global siguiendo los procedimientos usuales.

Los desplazamientos y fuerzas en los GDL eliminados se pueden recuperar usando las ecuaciones:

$$\begin{bmatrix} \boldsymbol{q}_r \\ \boldsymbol{q}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}_{rr} & \boldsymbol{K}_{re} \\ \boldsymbol{K}_{er} & \boldsymbol{K}_{ee} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_r \\ \boldsymbol{a}_e \end{bmatrix} - \begin{bmatrix} \boldsymbol{f}_r \\ \boldsymbol{f}_e \end{bmatrix}$$

$$\boldsymbol{a}_e = \boldsymbol{K}_{ee}^{-1} \left(\boldsymbol{q}_e + \boldsymbol{f}_e - \boldsymbol{K}_{er} \boldsymbol{a}_r \right)$$

$$\boldsymbol{q}_e = \boldsymbol{K}_{er} \boldsymbol{a}_r + \boldsymbol{K}_{ee} \boldsymbol{a}_e - \boldsymbol{f}_e$$

Etapas básicas del análisis de un sistema de barras

- **Etapa de preproceso**:
  - definición de geometría del problema
  - definición de las propiedades del material
  - definición de las condiciones de carga
  - definición de la malla de elementos finitos (numeración local y global de los grados de libertad, de los nodos y de los elementos)

Etapas básicas del análisis de un sistema de barras

- **<u>Etapa de cálculo</u>**
  - Cálculo de las matrices de rigidez $\boldsymbol{K}^{(e)}$ y los vectores de fuerza nodales $\boldsymbol{f}^{(e)}$ de cada elemento del sistema
  - Ensamblaje de la matriz de rigidez global $\boldsymbol{K}$ y vector de fuerzas global $\boldsymbol{f}$

Etapas básicas del análisis de un sistema de barras

- **<u>Etapa de cálculo</u>**
  - Solución del sistema de ecuaciones

$$\begin{bmatrix} \boldsymbol{q}_d \\ \boldsymbol{q}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}_{cc} & \boldsymbol{K}_{cd} \\ \boldsymbol{K}_{dc} & \boldsymbol{K}_{dd} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_c \\ \boldsymbol{a}_d \end{bmatrix} - \begin{bmatrix} \boldsymbol{f}_d \\ \boldsymbol{f}_c \end{bmatrix}$$

  - para calcular los desplazamientos en los nudo ***a*** (Cholesky, métodos especiales que manejan matrices ralas)

$$\boldsymbol{a}_d = \boldsymbol{K}_{dd}^{-1} \left(\boldsymbol{f}_c - \boldsymbol{K}_{dc} \boldsymbol{a}_c \right)$$

$$\boldsymbol{q}_d = \boldsymbol{K}_{cc} \boldsymbol{a}_c + \boldsymbol{K}_{cd} \boldsymbol{a}_d - \boldsymbol{f}_d$$

Etapas básicas del análisis de un sistema de barras

- **Etapa de cálculo**
  - A partir de los valores de las incógnitas en los nudos obtener información sobre los esfuerzos y deformaciones en las barras
- **Etapa de postproceso**
  - Presentación gráfica de la información

Bibliografía

- Henri P. Gavin (2014). Mathematical Properties of Stiffness Matrices. Duke University.
- Daryl L. Logan (2012) - *A First Course in the Finite Element Method*, CL. Engineering. 5 ed.
- Oñate (2009). *Structural Analysis with the Finite Element Method - Linear Statics*, Volume 1. Springer. https://doi.org/10.1007/978-1-4020-8733-2