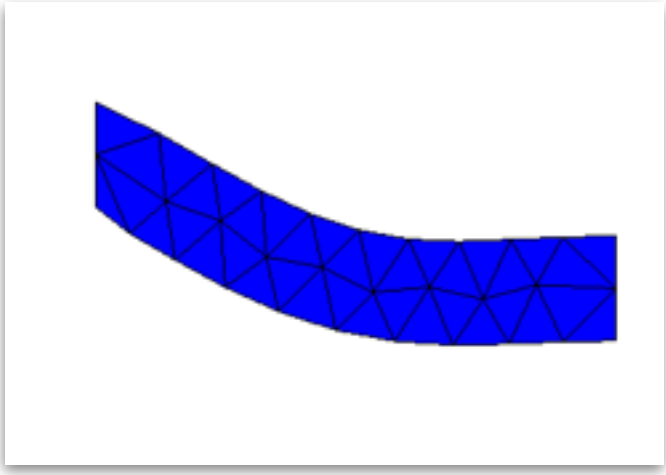
## TP MASTER IVI

# TP1 Modèle Masse-Ressort - CORRIGE



Le but du projet est de coder un modèle masse ressort en Scilab, avec une intégration temporelle en explicite.

+ Modifier le programme pour simuler la chute libre des points sous l'effet de la gravité (orientée selon y !!). Il faudra définir des variables pour l'acceleration et la vitesse des points et utiliser le schéma d'intégration suivant (backward Euler)

```
// Initialisation
F_t = zeros(2*numNoeuds,1);
A = zeros(2*numNoeuds,1);
V_t = zeros(2*numNoeuds,1);
X_t = zeros(2*numNoeuds,1);

// Dans la boucle de simulation..
// application du principe fondamental de la dynamique
    // mise à zero des forces
    F_t = zeros(2*numNoeuds,1);
    // gravite  et calcul de l'acceleration
    for i=1:numNoeuds
        F_t([2*i]) = F_t([2*i]) - m*g;


        // fixation du noeud
        P = noeuds(:,i);
        if (P(1)==5),
            F_t([2*i-1]) = 0;
            F_t([2*i]) =0;
        end

        A([2*i-1]) = F_t([2*i-1]) / m;
        A([2*i]) = F_t([2*i]) / m;

    end
```

+ Trouver comment calculer la force que crée les ressorts sur les noeuds (la fonction findSegment permettant de trouver les segments du maillage). La valeur nominale de la force est la suivante (mais attention, elle est orientée dans la direction du segment, il va donc falloir modifier la formule !!).

```
force = k * (longeur - longeurInit)
```

+ Modifier votre programme pour intégrer le calcul des ressorts du maillage.

```
// Initialisation: calcul des longueurs au repos des ressorts:
L0=[];
for s=1:numSegments,
    i1= segments(1,s);
    i2= segments(2,s);
    L0(s) = norm( X_t([2*i1-1 2*i1]) - X_t([2*i2-1 2*i2]) )
end
```

```
// dans la boucle de simulation (bloc à placer entre mise à zero des forces et gravite et calcul de l'acceleration)
 // mise à zero des forces
    F_t = zeros(2*numNoeuds,1);

    L=[];
    for s=1:numSegments,
        i1= segments(1,s);
        i2= segments(2,s);
        // longueur actuelle
        L(s) = norm( X_t([2*i1-1 2*i1]) - X_t([2*i2-1 2*i2]) )
        // normale actuelle
        n = (X_t([2*i1-1 2*i1]) - X_t([2*i2-1 2*i2]))*(1.0/L(s));

        // force du ressort
        F = n*(k*(L(s)-L0(s)))// + d*V)

        // ajout de la force dans le vecteur
        F_t([2*i1-1 2*i1]) = F_t([2*i1-1 2*i1])  - F;
        F_t([2*i2-1 2*i2]) = F_t([2*i2-1 2*i2])  + F;

    end
```