11th US National Congress on Computational Mechanics:

PolyTop: A Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes

Ivan Menezes[b], Cameron Talischi[a], Anderson Pereira[b], Glaucio H Paulino[a]

[a]University of Illinois at Urbana-Champaign, USA
[b]Tecgraf, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil

Minneapolis, Minnesota, July 26th, 2011

Outline

- ☐ Problem formulation: regularization maps and interpolation functions
- ☐ Spatial discretization and the discrete optimization problem
- ☐ Modular framework: separation of formulation and analysis
- ☐ Code structure: inputs and implementation
- ☐ Demo, results and performance

Topology optimization problem

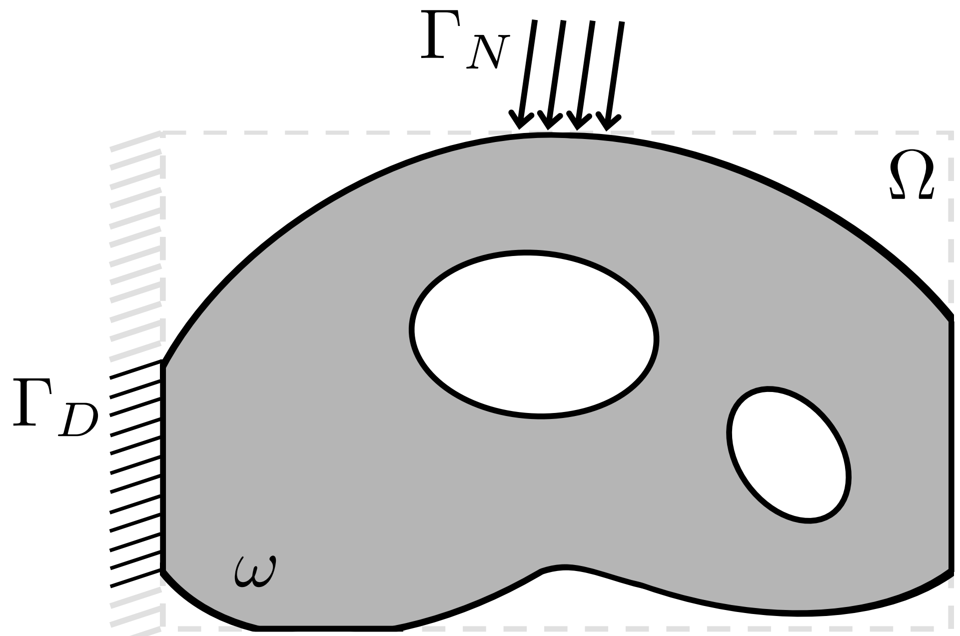
- The topology optimization problem is of the form:

$$\inf_{\omega \in \mathcal{O}} f(\omega, \mathbf{u}_\omega) \qquad \text{subject to} \qquad g_i(\omega, \mathbf{u}_\omega) \leq 0, \; i = 1, \ldots, K$$

where $\mathbf{u}_\omega$ solves the boundary value problem:

$$\int_\omega \mathbf{C}\nabla\mathbf{u}_\omega : \nabla\mathbf{v}\mathrm{d}\mathbf{x} = \int_{\tilde{\Gamma}_N} \mathbf{t} \cdot \mathbf{v}\mathrm{d}s, \quad \forall \mathbf{v} \in \mathcal{V}_\omega$$

with $\mathcal{V}_\omega = \left\{\mathbf{v} \in H^1(\omega; \mathbb{R}^d) : \mathbf{v}|_{\partial\omega \cap \Gamma_D} = \mathbf{0}\right\}$



Reformulation as sizing problem

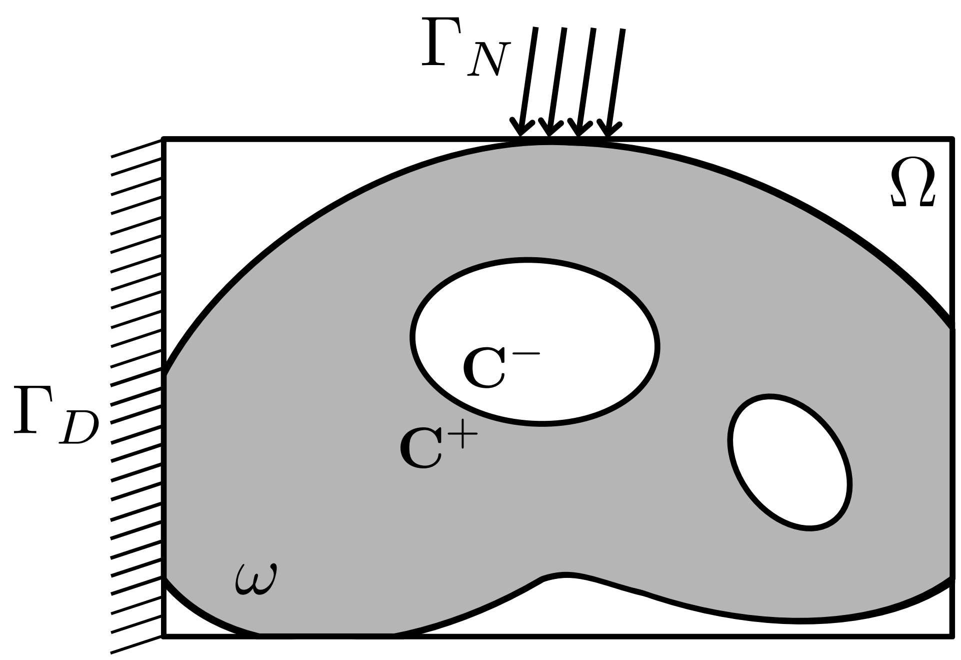
- The problem is reformulated as:

$$\inf_{\rho \in \mathcal{A}} f(\rho, \mathbf{u}_\rho) \qquad \text{subject to} \qquad g_i(\rho, \mathbf{u}_\rho) \leq 0, \; i = 1, \ldots, K$$

where $\mathcal{A} = \{\mathcal{P}(\eta) : \eta \in L^\infty(\Omega; [\underline{\rho}, \overline{\rho}])\}$ and the state equation is

$$\int_\Omega m_E(\rho) \mathbf{C} \nabla \mathbf{u}_\rho : \nabla \mathbf{v} \mathrm{d}\mathbf{x} = \int_{\Gamma_N} \mathbf{t} \cdot \mathbf{v} \mathrm{d}s, \quad \forall \mathbf{v} \in \mathcal{V}$$

with $\mathcal{V} = \left\{ \mathbf{v} \in H^1(\Omega; \mathbb{R}^d) : \mathbf{v}|_{\Gamma_D} = \mathbf{0} \right\}$



Mapping $\mathcal{P}$

- We impose regularity on the space of admissible sizing function $\mathcal{A}$ implicitly by means of “regularization” map $\mathcal{P}$, e.g.,

$$\mathcal{P}_F(\eta)(\mathbf{x}) := \int_\Omega F(\mathbf{x}, \overline{\mathbf{x}})\eta(\overline{\mathbf{x}})\mathrm{d}\overline{\mathbf{x}}$$

where F is a prescribed smooth kernel

Mapping $\mathcal{P}$

- We impose regularity on the space of admissible sizing function $\mathcal{A}$ implicitly by means of "regularization" map $\mathcal{P}$, e.g.,

$$\mathcal{P}_F(\eta)(\mathbf{x}) := \int_\Omega F(\mathbf{x}, \overline{\mathbf{x}})\eta(\overline{\mathbf{x}})\mathrm{d}\overline{\mathbf{x}}$$

  where F is a prescribed smooth kernel

  - Even if η is rough, $\rho = \mathcal{P}(\eta)$ is guaranteed to be smooth thereby eliminating the need to explicitly enforce regularity on ρ
  - It is the discretization of η that produces the set of design variables for the optimization problem

Mapping $\mathcal{P}$

- We impose regularity on the space of admissible sizing function $\mathcal{A}$ implicitly by means of "regularization" map $\mathcal{P}$, e.g.,

$$\mathcal{P}_F(\eta)(\mathbf{x}) := \int_\Omega F(\mathbf{x}, \overline{\mathbf{x}})\eta(\overline{\mathbf{x}})\mathrm{d}\overline{\mathbf{x}}$$

where F is a prescribed smooth kernel

  - Even if η is rough, $\rho = \mathcal{P}(\eta)$ is guaranteed to be smooth thereby eliminating the need to explicitly enforce regularity on ρ
  - It is the discretization of η that produces the set of design variables for the optimization problem

- Other layout constraints such as symmetry can be achieved in the same way, for example,

$$\mathcal{P}_s(\eta)(\mathbf{x}) = \eta(x_1, |x_2|)$$

and these can be combined, $\mathcal{P} = \mathcal{P}_F \circ \mathcal{P}_s$

Interpolation functions

- Interpolation functions relate the sizing function to ρ to the material properties (e.g. stiffness and volume)

Interpolation functions

- Interpolation functions relate the sizing function to ρ to the material properties (e.g. stiffness and volume)
- Examples include

$$\text{SIMP:} \qquad m_E(\rho) = \varepsilon + (1-\varepsilon)\rho^p, \quad m_V(\rho) = m_P(\rho) = \rho$$

$$\text{RAMP:} \qquad m_E(\rho) = \varepsilon + (1-\varepsilon)\frac{\rho}{1+q(1-\rho)}, \quad m_V(\rho) = \rho$$

Interpolation functions

- Interpolation functions relate the sizing function to ρ to the material properties (e.g. stiffness and volume)
- Examples include

$$\text{SIMP:} \quad m_E(\rho) = \varepsilon + (1-\varepsilon)\rho^p, \quad m_V(\rho) = m_P(\rho) = \rho$$

$$\text{RAMP:} \quad m_E(\rho) = \varepsilon + (1-\varepsilon)\frac{\rho}{1+q(1-\rho)}, \quad m_V(\rho) = \rho$$

- "Nonlinear" filtering can also be cast in the same framework:

Interpolation functions

- Interpolation functions relate the sizing function to ρ to the material properties (e.g. stiffness and volume)
- Examples include

$$\text{SIMP:} \quad m_E(\rho) = \varepsilon + (1-\varepsilon)\rho^p, \quad m_V(\rho) = m_P(\rho) = \rho$$

$$\text{RAMP:} \quad m_E(\rho) = \varepsilon + (1-\varepsilon)\frac{\rho}{1+q(1-\rho)}, \quad m_V(\rho) = \rho$$

- "Nonlinear" filtering can also be cast in the same framework: for example, the approach Guest et a.l (2004) is equivalent to defining

$$m_E(\rho) = \varepsilon + (1-\varepsilon)\left[H(\rho)\right]^p, \quad m_V(\rho) = H(\rho)$$

where $H(x) = 1 - \exp(-\beta x) + x\exp(-\beta)$ since ρ is already of the form $\rho = \mathcal{P}_F(\eta)$

Interpolation functions

- Interpolation functions relate the sizing function to ρ to the material properties (e.g. stiffness and volume)

- Examples include

$$\text{SIMP:} \qquad m_E(\rho) = \varepsilon + (1-\varepsilon)\rho^p, \quad m_V(\rho) = m_P(\rho) = \rho$$

$$\text{RAMP:} \qquad m_E(\rho) = \varepsilon + (1-\varepsilon)\,\frac{\rho}{1+q(1-\rho)}, \quad m_V(\rho) = \rho$$

- "Nonlinear" filtering can also be cast in the same framework: for example, the approach Guest et a.l (2004) is equivalent to defining

$$m_E(\rho) = \varepsilon + (1-\varepsilon)\left[H(\rho)\right]^p, \quad m_V(\rho) = H(\rho)$$

where $H(x) = 1 - \exp(-\beta x) + x\exp(-\beta)$ since ρ is already of the form $\rho = \mathcal{P}_F(\eta)$

  - Observe fact that SIMP penalization plays a crucial role since with $p = 1$, we have $m_E(\rho) \approx m_V(\rho)$ and so optimal solutions will consist mostly of "grey" no matter how large β is

Discretization

- Consider $\mathcal{T}_h = \{\Omega_\ell\}_{\ell=1}^N$ a partition of Ω such that $\Omega_\ell \cap \Omega_k = \emptyset$ for $\ell \neq k$ and $\cup_\ell \overline{\Omega_\ell} = \overline{\Omega}$

Discretization

- Consider $\mathcal{T}_h = \{\Omega_\ell\}_{\ell=1}^N$ a partition of Ω such that $\Omega_\ell \cap \Omega_k = \emptyset$ for $\ell \neq k$ and $\cup_\ell \overline{\Omega}_\ell = \overline{\Omega}$

- The discrete problem is the same optimization as before with $\mathcal{V}$ replaced by the finite element subspace $\mathcal{V}_h$ and $\mathcal{A}$ replaced by

$$\mathcal{A}_h = \left\{ \mathcal{P}_h(\eta_h) : \underline{\rho} \leq \eta_h \leq \overline{\rho}, \eta|_{\Omega_\ell} = \text{const } \forall \ell \right\}$$

  where $\mathcal{P}_h$ is an approximation to $\mathcal{P}$

Discretization

- Consider $\mathcal{T}_h = \{\Omega_\ell\}_{\ell=1}^N$ a partition of Ω such that $\Omega_\ell \cap \Omega_k = \emptyset$ for $\ell \neq k$ and $\cup_\ell \overline{\Omega}_\ell = \overline{\Omega}$

- The discrete problem is the same optimization as before with $\mathcal{V}$ replaced by the finite element subspace $\mathcal{V}_h$ and $\mathcal{A}$ replaced by

$$\mathcal{A}_h = \left\{ \mathcal{P}_h(\eta_h) : \underline{\rho} \leq \eta_h \leq \overline{\rho}, \eta|_{\Omega_\ell} = \text{const } \forall \ell \right\}$$

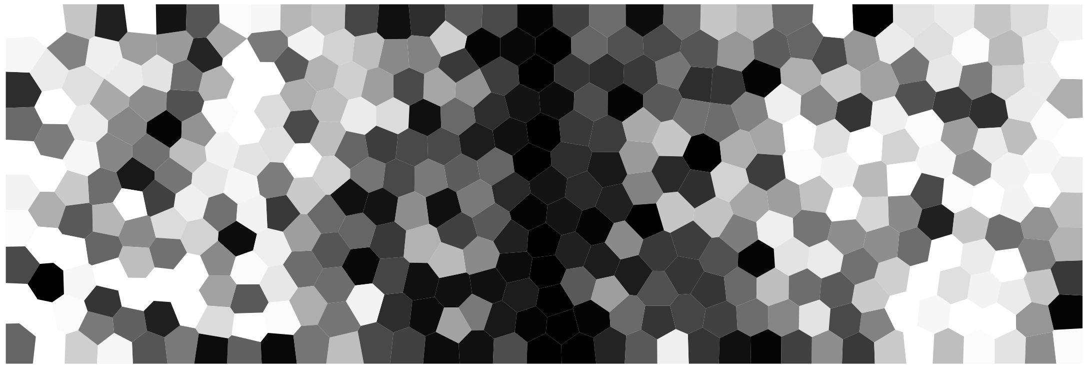
  where $\mathcal{P}_h$ is an approximation to $\mathcal{P}$

- Each piecewise constant η_h can be represented by vector $\mathbf{z} = [z_\ell]$ since $\eta_h(\mathbf{x}) = \sum_{\ell=1}^N z_\ell \chi_{\Omega_\ell}(\mathbf{x})$ and similarly each $\rho_h \in \mathcal{A}_h$ can be defined by elemental values $\mathbf{y} = \mathbf{P}\mathbf{z}$ where

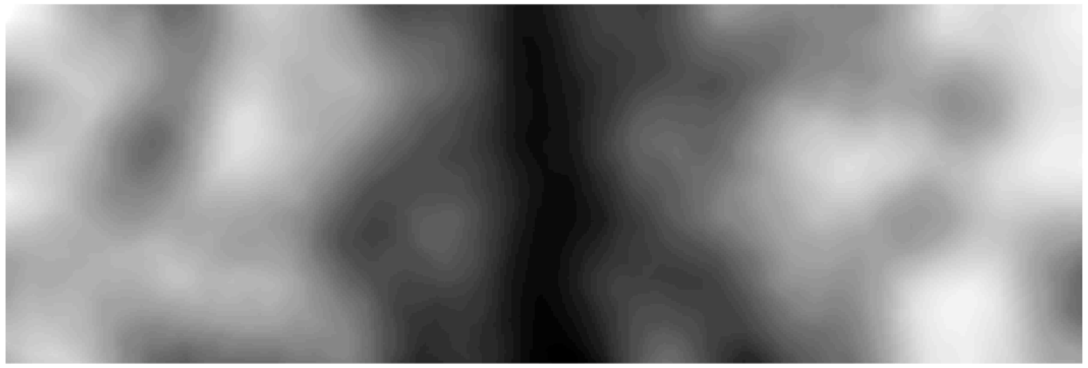
$$(\mathbf{P})_{\ell k} = \mathcal{P}(\chi_{\Omega_k})(\mathbf{x}_\ell^*)$$

Discretization

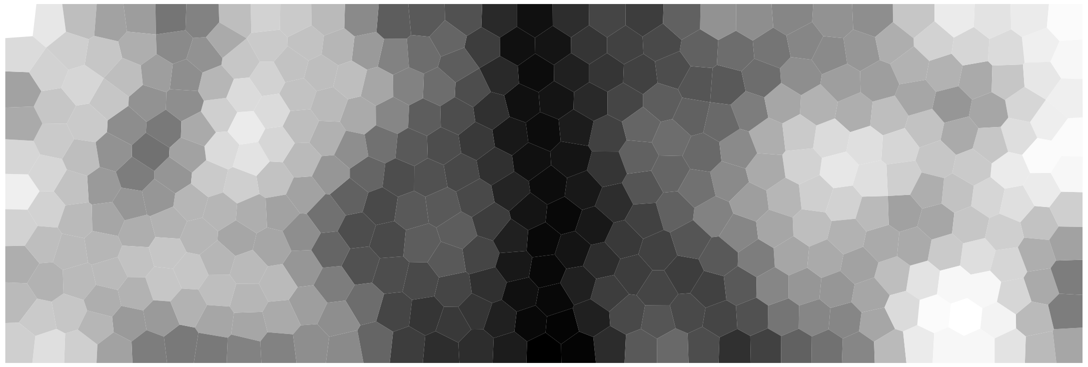
$$\eta_h = \sum_{\ell=1}^{N} z_\ell \chi_{\Omega_\ell}$$



$$\mathcal{P}(\eta_h)$$



$$\mathcal{P}_h(\eta_h) = \sum_{\ell=1}^{N} y_\ell \chi_{\Omega_\ell}$$



Modular framework

- For the minimum compliance problem, $\mathbf{E} := m_E(\mathbf{Pz})$ and $\mathbf{V} := m_V(\mathbf{Pz})$ are the only design related information that need to be provided to the analysis functions

Modular framework

- For the minimum compliance problem, $\mathbf{E} := m_E(\mathbf{Pz})$ and $\mathbf{V} := m_V(\mathbf{Pz})$ are the only design related information that need to be provided to the analysis functions
  - The analysis functions need not know about the choice of interpolations functions or the mapping $\mathcal{P}$
  - A clear advantage of this approach is that the analysis functions can be extended, developed and modified independently

Modular framework

- For the minimum compliance problem, $\mathbf{E} := m_E(\mathbf{Pz})$ and $\mathbf{V} := m_V(\mathbf{Pz})$ are the only design related information that need to be provided to the analysis functions
  - The analysis functions need not know about the choice of interpolations functions or the mapping $\mathcal{P}$
  - A clear advantage of this approach is that the analysis functions can be extended, developed and modified independently

- The sensitivity analysis can be "separated" along the same lines:

$$\frac{\partial g_i}{\partial \mathbf{z}} = \frac{\partial \mathbf{E}}{\partial \mathbf{z}} \frac{\partial g_i}{\partial \mathbf{E}} + \frac{\partial \mathbf{V}}{\partial \mathbf{z}} \frac{\partial g_i}{\partial \mathbf{V}}$$

where $\partial g_i / \partial \mathbf{E}$ and $\partial g_i / \partial \mathbf{V}$ are sensitivities with respect to analysis parameters and

$$\frac{\partial \mathbf{E}}{\partial \mathbf{z}} = \mathbf{P}^T J_{m_E}(\mathbf{Pz}), \qquad \frac{\partial \mathbf{V}}{\partial \mathbf{z}} = \mathbf{P}^T J_{m_V}(\mathbf{Pz})$$

Modular framework

- For the minimum compliance problem, $\mathbf{E} := m_E(\mathbf{Pz})$ and $\mathbf{V} := m_V(\mathbf{Pz})$ are the only design related information that need to be provided to the analysis functions
  - The analysis functions need not know about the choice of interpolations functions or the mapping $\mathcal{P}$
  - A clear advantage of this approach is that the analysis functions can be extended, developed and modified independently

- The sensitivity analysis can be “separated” along the same lines:

$$\frac{\partial g_i}{\partial \mathbf{z}} = \frac{\partial \mathbf{E}}{\partial \mathbf{z}} \frac{\partial g_i}{\partial \mathbf{E}} + \frac{\partial \mathbf{V}}{\partial \mathbf{z}} \frac{\partial g_i}{\partial \mathbf{V}}$$

  where $\partial g_i / \partial \mathbf{E}$ and $\partial g_i / \partial \mathbf{V}$ are sensitivities with respect to analysis parameters and

$$\frac{\partial \mathbf{E}}{\partial \mathbf{z}} = \mathbf{P}^T J_{m_E}(\mathbf{Pz}), \qquad \frac{\partial \mathbf{V}}{\partial \mathbf{z}} = \mathbf{P}^T J_{m_V}(\mathbf{Pz})$$

- The optimizer should also be kept separate but this is more common

Inputs to PolyTop

- The input to the main kernel `PolyTop` consists of two Matlab structure arrays containing the optimization and analysis fields:

fem	opt
fem.NNode	opt.zMin
fem.NElem	opt.zMax
fem.Node	opt.zIni
fem.Element	opt.MatIntFnc
fem.Supp	opt.P
fem.Load	opt.MaxIter
fem.ShapeFnc	opt.Tol
...	...

Inputs to PolyTop

- The input to the main kernel `PolyTop` consists of two Matlab structure arrays containing the optimization and analysis fields:

fem	opt
fem.NNode	opt.zMin
fem.NElem	opt.zMax
fem.Node	opt.zIni
fem.Element	opt.MatIntFnc
fem.Supp	opt.P
fem.Load	opt.MaxIter
fem.ShapeFnc	opt.Tol
...	...

- Given an input vector $\mathbf{y}$, `MatIntFnc` function returns arrays $\mathbf{E} = m_E(\mathbf{y})$ and $\mathbf{V} = m_V(\mathbf{y})$ and the sensitivity vectors $\partial\mathbf{E}/\partial\mathbf{y} := m'_E(\mathbf{y})$ and $\partial\mathbf{V}/\partial\mathbf{y} := m'_V(\mathbf{y})$

PolyTop

- `PolyTop` possesses fewer than 190 lines, of which 116 lines pertain to the finite element analysis including 81 lines for the element stiffness calculations for polygonal elements

PolyTop

- `PolyTop` possesses fewer than 190 lines, of which 116 lines pertain to the finite element analysis including 81 lines for the element stiffness calculations for polygonal elements

```matlab
function [z,V,fem] = PolyTop(fem,opt)
Iter=0; Tol=opt.Tol*(opt.zMax-opt.zMin); Change=2*Tol; z=opt.zIni; P=opt.P;
[E,dEdy,V,dVdy] = opt.MatIntFnc(P*z);
[FigHandle,FigData] = InitialPlot(fem,z);
while (Iter<opt.MaxIter) && (Change>Tol)
  Iter = Iter + 1;
  %Compute cost functionals and analysis sensitivities
  [f,dfdE,dfdV,fem] = ObjectiveFnc(fem,E,V);
  [g,dgdE,dgdV,fem] = ConstraintFnc(fem,E,V,opt.VolFrac);
  %Compute design sensitivities
  dfdz = P'*(dEdy.*dfdE + dVdy.*dfdV);
  dgdz = P'*(dEdy.*dgdE + dVdy.*dgdV);
  %Update design variable and analysis parameters
  [z,Change] = UpdateScheme(dfdz,g,dgdz,z,opt);
  [E,dEdy,V,dVdy] = opt.MatIntFnc(P*z);
  %Output results
  fprintf('It: %i \t Objective: %1.3f\tChange: %1.3f\n',Iter,f,Change);
  set(FigHandle,'FaceColor','flat','CData',1-V(FigData)); drawnow
end
```

Comment about FEM implementation

- Certain quantities used in the analysis functions such as element stiffness matrices as well as the connectivity of the global stiffness matrix $\mathbf{K}$ need to be computed only once

Comment about FEM implementation

- Certain quantities used in the analysis functions such as element stiffness matrices as well as the connectivity of the global stiffness matrix $\mathbf{K}$ need to be computed only once
- This is accomplished in PolyTop by computing vector `fem.i`, `fem.j`, `fem.k` and `fem.e` with assembly computed by command:

```
K = sparse(fem.i,fem.j,E(fem.e).*fem.k);
```

Comment about FEM implementation

- Certain quantities used in the analysis functions such as element stiffness matrices as well as the connectivity of the global stiffness matrix $\mathbf{K}$ need to be computed only once

- This is accomplished in PolyTop by computing vector `fem.i`, `fem.j`, `fem.k` and `fem.e` with assembly computed by command:

```
K = sparse(fem.i,fem.j,E(fem.e).*fem.k);
```

- The isoparametric polygonal elements can be viewed as extension of the common linear triangles and bilinear quads to all convex n-gons
  - This element stiffness calculations add very little overhead

Demo, results and efficiency

- Matlab demo

Demo, results and efficiency

- Matlab demo
- Comparison of efficiency with 88 line:

mesh size	90×30	150×50	300×100	600×200
total time of `PolyTop`	15.5	**40.7**	**187**	**1016**
total time of 88 line	**14.8**	44.4	360	4463

Demo, results and efficiency

- Matlab demo
- Comparison of efficiency with 88 line:

mesh size	90×30	150×50	300×100	600×200
total time of `PolyTop`	15.5	**40.7**	**187**	**1016**
total time of 88 line	**14.8**	44.4	360	4463

- Source of discrepancy is the computation of volume constraint inside the OC optimizer:

$$V(\mathbf{z}) = \sum_{\ell=1}^{N} (\mathbf{Pz})_\ell = \mathbf{1}^T (\mathbf{Pz}) = \left(\mathbf{1}^T \mathbf{P}\right) \mathbf{z} = \left(\mathbf{P}^T \mathbf{1}\right)^T \mathbf{z}$$

  - Though the above expression is not explicitly used in `PolyTop`, the decoupling of the OC scheme from the analysis routine naturally leads to the more efficient calculation

Concluding remarks

- We have a general framework for topology optimization using unstructured meshes in arbitrary domains
- The analysis routine and optimization algorithm are separated from the specific choice of topology optimization formulation
- The FE and sensitivity analysis routines can be extended, maintained, developed, and/or modified independently

QUESTIONS?