# Mechano-adaptive generation of space-frames (MAGOS) GH plug-in

**Introduction**

1. License: CC-BY. This software is distributed as is without any warranty. You’re allowed to use it freely, as long as you credit the authors. If you use this software in your research, we kindly ask you to cite this paper: [LINK TO PAPER]. The authors are grateful to Foster+Partners and The Royal Veterinary College for funding the research project that led to this work.
2. What is MAGOS?  
   MAGOS is a GH plug-in that takes two principal inputs to generate a space frame:
   1. The volume the space frame should take up
   2. A grid of stress-tensors within the volume that represent the loads the space frame should withstand.

The result is a set of node locations and strut connections that we expect perform better than traditional space frames for their specific load case. Additionally, MAGOS takes a set of parameters with which the user can tune various properties of the result. MAGOS is written in C# and is inspired by the fact that the network of struts inside long bones (the ‘trabeculae’) tend to align in directions of stress. For more in-depth information, please read the paper or consult the documentation.

**Quick-start guide for F+P internal users**

1. Adding the MAGOS plug-in: Make sure you are using the latest versions of Kangaroo and Kangaroo2 in the usual GH components folder (For some time now, Kangaroo gets re-installed in this location:   
     
   C:\Program Files\Common Files\McNeel\Rhinoceros\5.0\Plug-ins\Grasshopper0.9.76.0\Components

every time an F+P machine is restarted, and as a consequence the MAGOS plug-in is not recognised. If this is the case, please delete Kangaroo.gha and Kangaroo.dll in said location to avoid this problem after each restart of your machine. This will be hopefully fixed soon).

1. Setting up a geometry and a load-case. There are example geometries and corresponding load-cases ready to test. [explain here, refer to example COMSOL/other thing set-up files]
2. Run MAGOS
   1. With the compiled component:
   2. Within a C# script

Example in testCompiledEqualsScriptable.gh can give some hints to this, this will be expanded in the future.

1. Post-processing and interpreting the output [MORE NEEDS TO BE DONE HERE!!!]
2. Contact: Please get in touch with Alessandro ([afelder@fosterandpartners.com](mailto:afelder@fosterandpartners.com)) or Daniel ([dpiker@fosterandpartners.com](mailto:dpiker@fosterandpartners.com)) with any questions or feedback, which will be warmly welcomed.