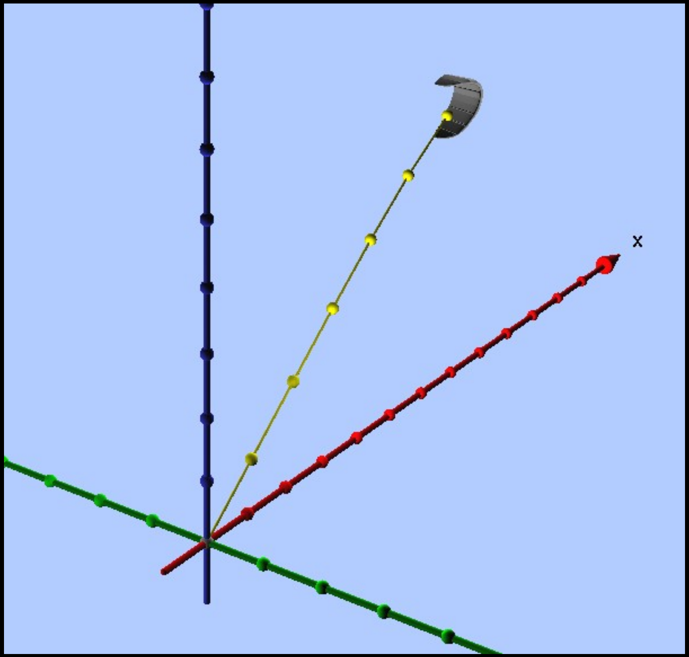
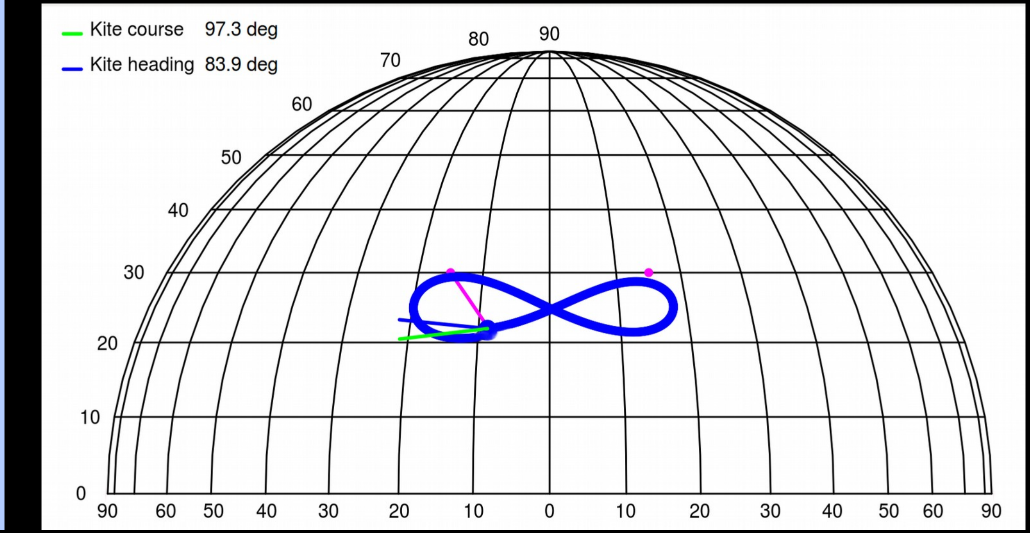




Dynamic Kite Power System Modelling

Airborne Wind Energy Conference 2013, September 10-11, Berlin, Germany

Uwe Fechner, MSc.

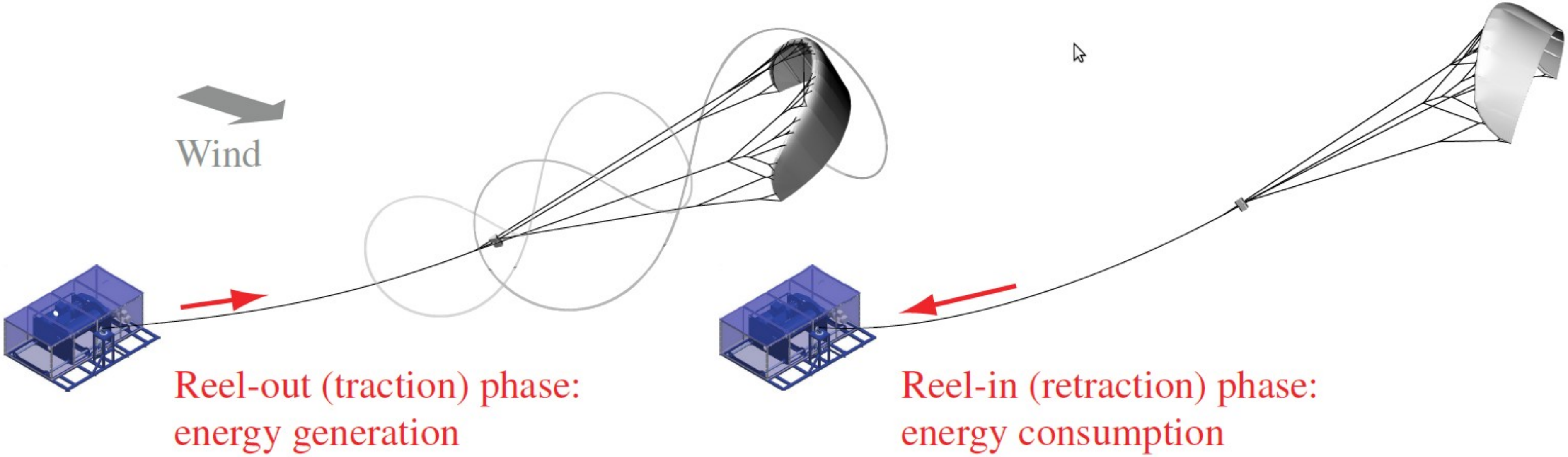
Outline

Dynamic Kite Power System Models

- *Introduction to Kite-Power Technology*
- *Applications of Dynamic Kite Power System Models*
- *Model components*
  - Kite
  - Tether
  - Winch
  - Kite Control Unit (KCU) and Sensors
- *Model Feature Overview*
- *Simulation Results*
  - Flight-path and winch controllers
  - 3D Trajectory
  - Power, forces, tether-length and height
- *Simulation Software*
- *Summary and Conclusion*

Technology

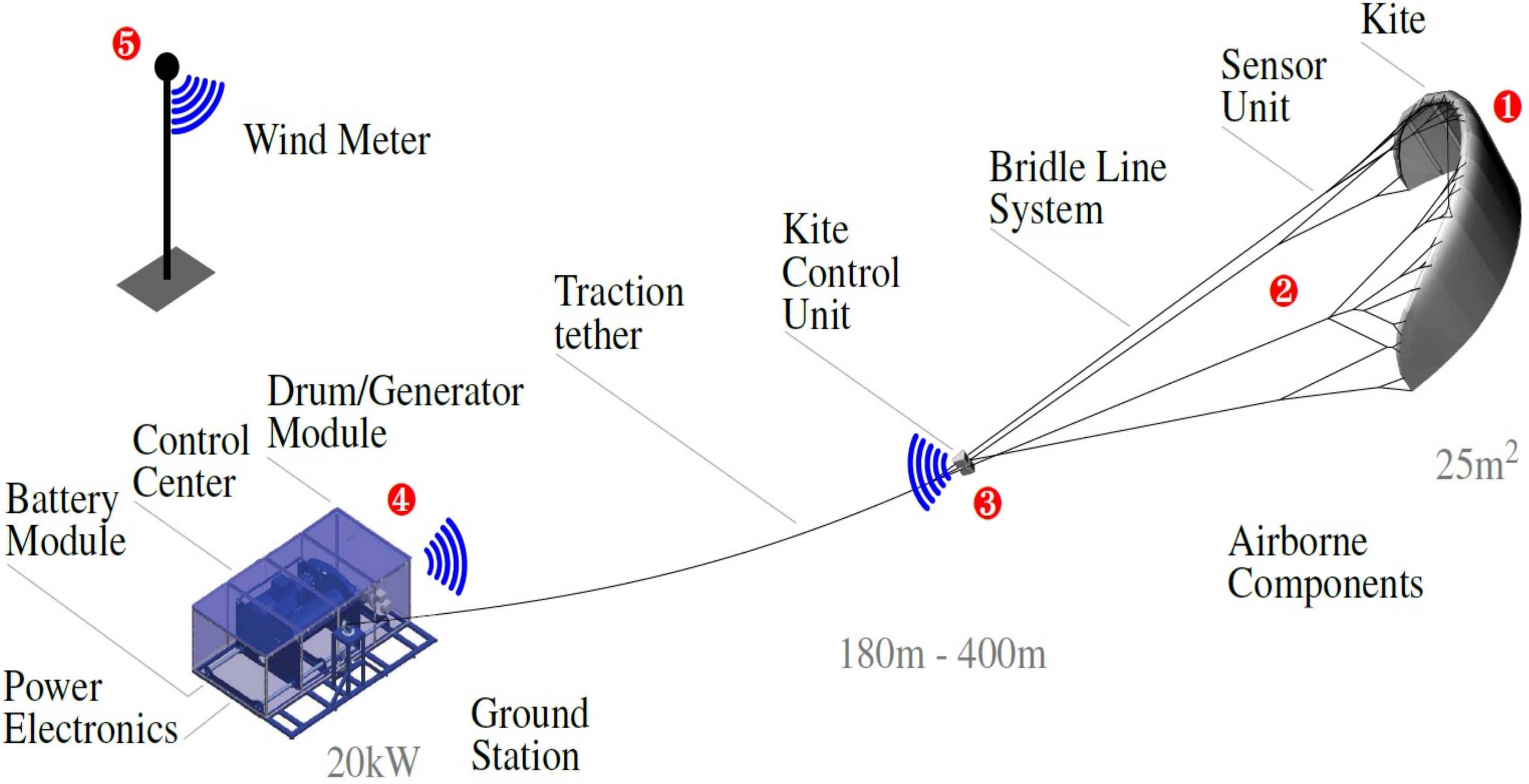
Kite Power System



Working principle of a a pumping kite power system [1].

Components of a Kite Power System

Kite Power System



System components, sensor locations (circles) and wireless connections [1]

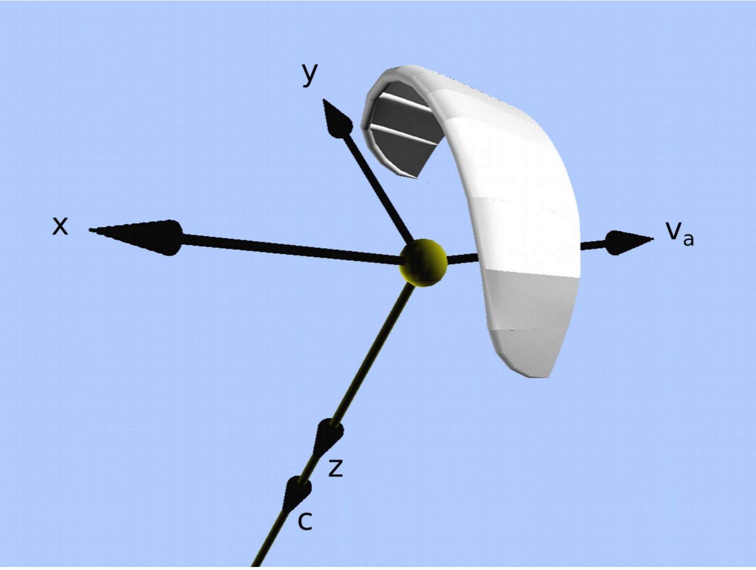
Applications of Dynamic Kite Power System Models

- *Flight path optimization;*
- *flight path controller design;*
- *winch controller design;*
- *software-in-the-loop (SIL) testing;*
- *develop strategies for automated launch and landing;*
- *pilot training;*
- *winch operator training.*

Kite Models

KPS Model Components

3 DOF point-mass model



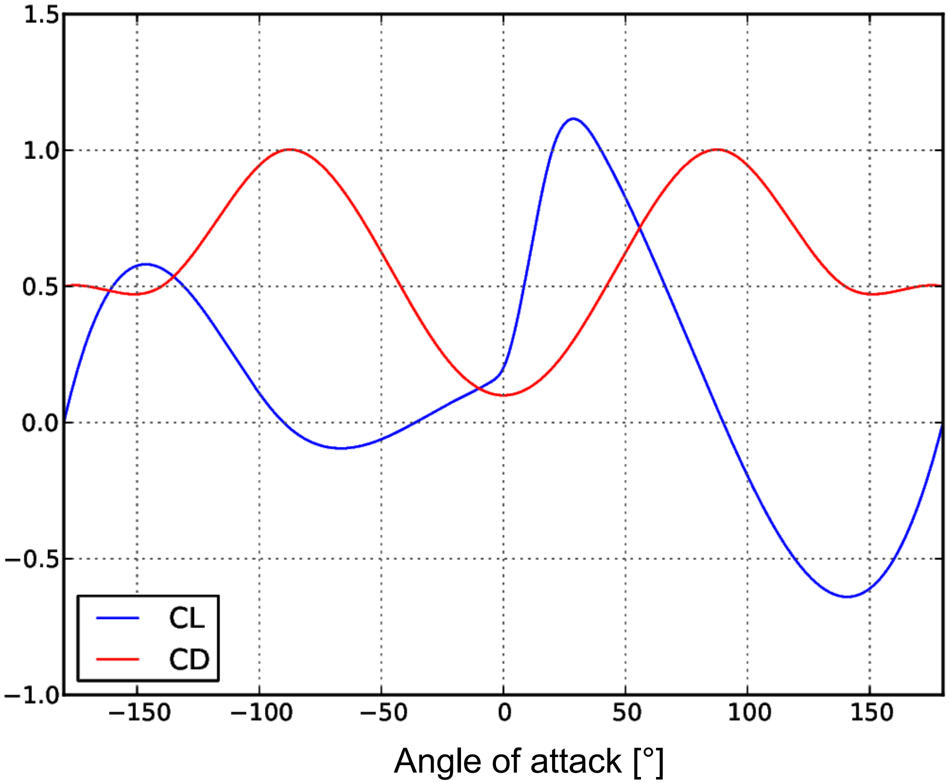
$$\mathbf{z} = \frac{\boldsymbol{c}}{\|\boldsymbol{c}\|}$$

$$\mathbf{y} = \frac{\boldsymbol{v_a} \times \boldsymbol{c}}{\| \boldsymbol{v_a} \times \boldsymbol{c} \|}$$

$$\mathbf{x} = \frac{\boldsymbol{y} \times \boldsymbol{c}}{\| \boldsymbol{y} \times \boldsymbol{c} \|}$$

Aerodynamic Forces (Lift and Drag)

KPS Model Components



$$\alpha = \frac{\pi}{2} - \frac{acos(\boldsymbol{v_a} \cdot -\boldsymbol{z})}{\| \boldsymbol{v_a} \|}$$

$$\boldsymbol{L} = \frac{1}{2}\, \rho\, C_L\, A_k\, |\boldsymbol{v_a}|^2$$

$$\boldsymbol{D} = \frac{1}{2}\, \rho\, C_D\, A_k\, |\boldsymbol{v_a}|^2 \cdot (1 + 0.6\, |s|)$$

$$-1 \leq s \leq 1$$

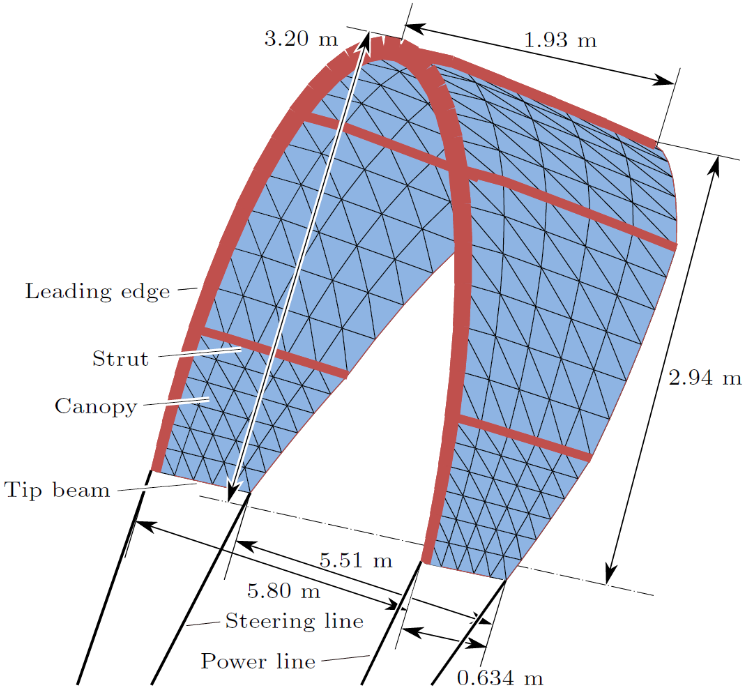
s: relative steering displacement

Kite Models II

KPS Model Components

Other Kite Models:

- *Semi-rigid 4 DOF point mass model [5];*
- *four point kite model (TU Delft)*
- *finite element kite model [4].*

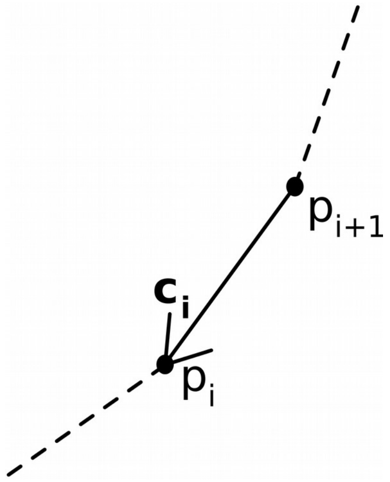


FE mesh of 16 m² kite [4]

Tether Models

KPS Model Components

Tether Models



- *Constraint;*
- *single spring damper;*
- *discretised point mass.*

Implemented:

Discretised point mass model with varying segment length (at zero force) and drag, but constant number of particles.

Simulating Reel-Out

KPS Model Components

The simulation is running in fixed time intervals of 50 ms.

The segment length at zero force is varied in the following way:

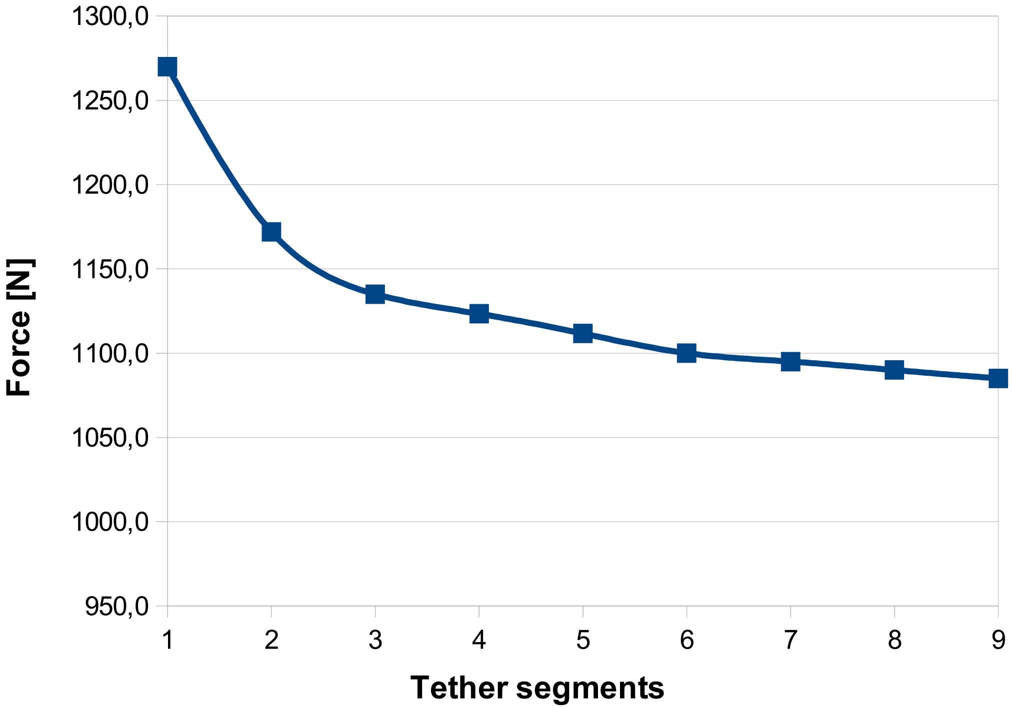
$$l_s = \frac{l_{t,i}}{n} + \frac{v_{t,o}\ (t - t_i)}{n}$$

Now the spring and damping 'constants' and the particle mass can be calculated:

$$k = \frac{k_0}{l_s} \qquad c = \frac{c_0}{l_s} \qquad m_p = m_0\ l_s$$

Tether Model accuracy

KPS Model Components



Steady-state tether force at 8 m/s wind speed with 150 m tether length.

- **Straight line tether: > 20% error**
- **8 tether segments: < 5 % error**

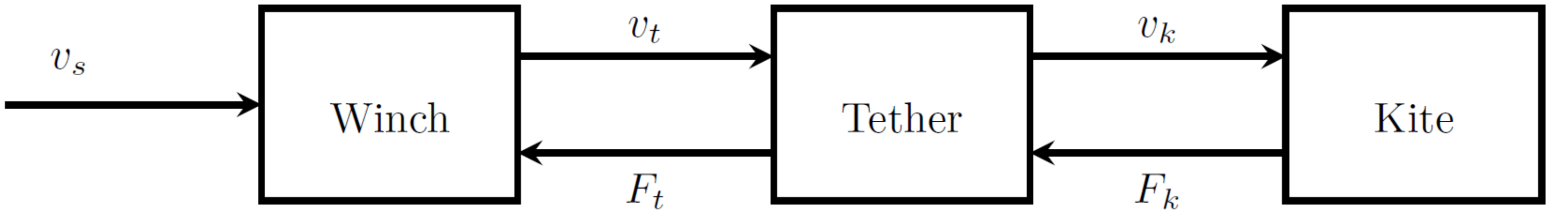
Winch (Generator) Models

KPS Model Components

What kind of models are needed:

- Efficiency models

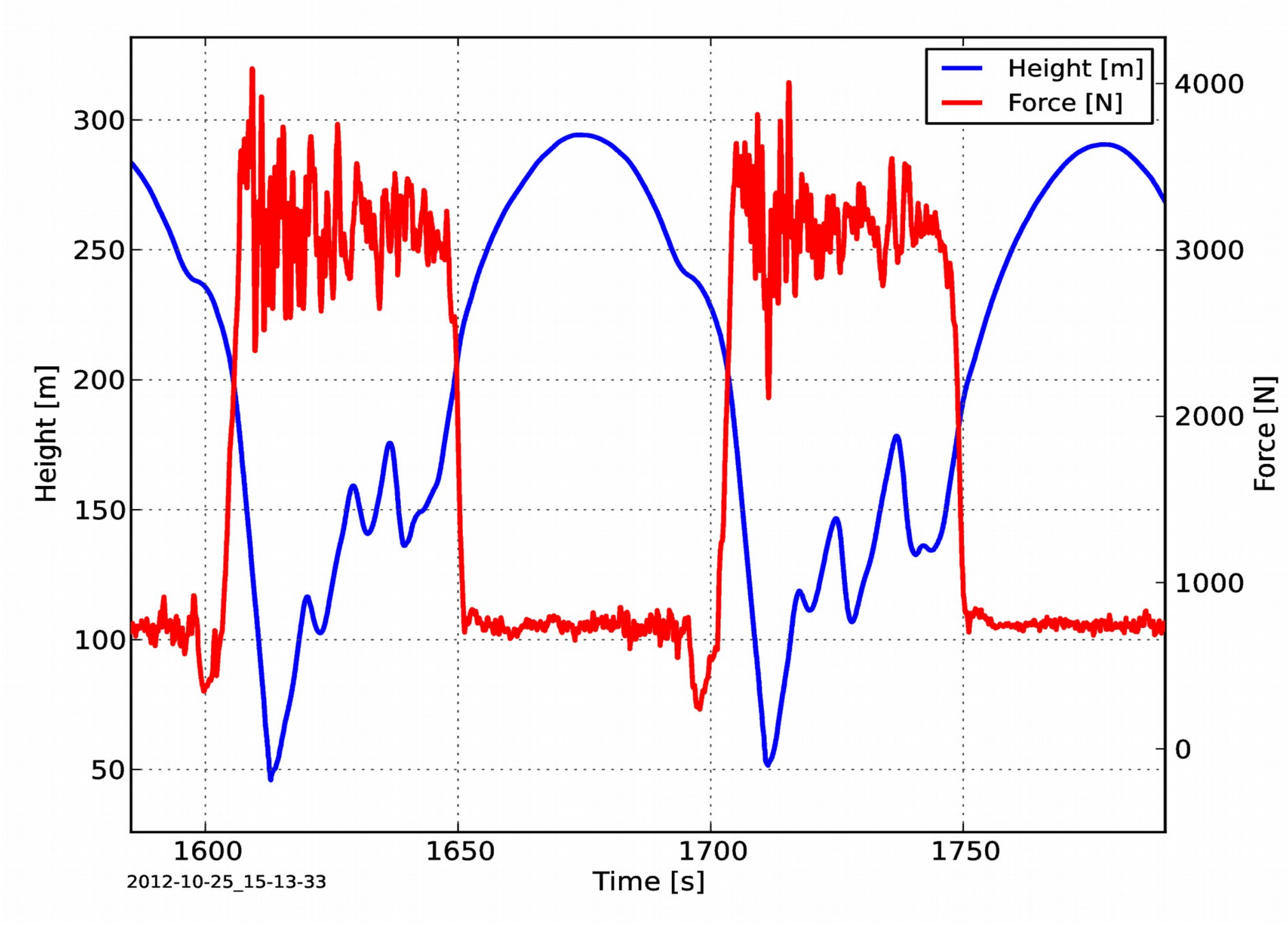
  → For the calculation of the system efficiency
- Dynamic models
  → For the design of the force control loop.



Dynamic System Model [2]

Why Force Control?

KPS Model Components



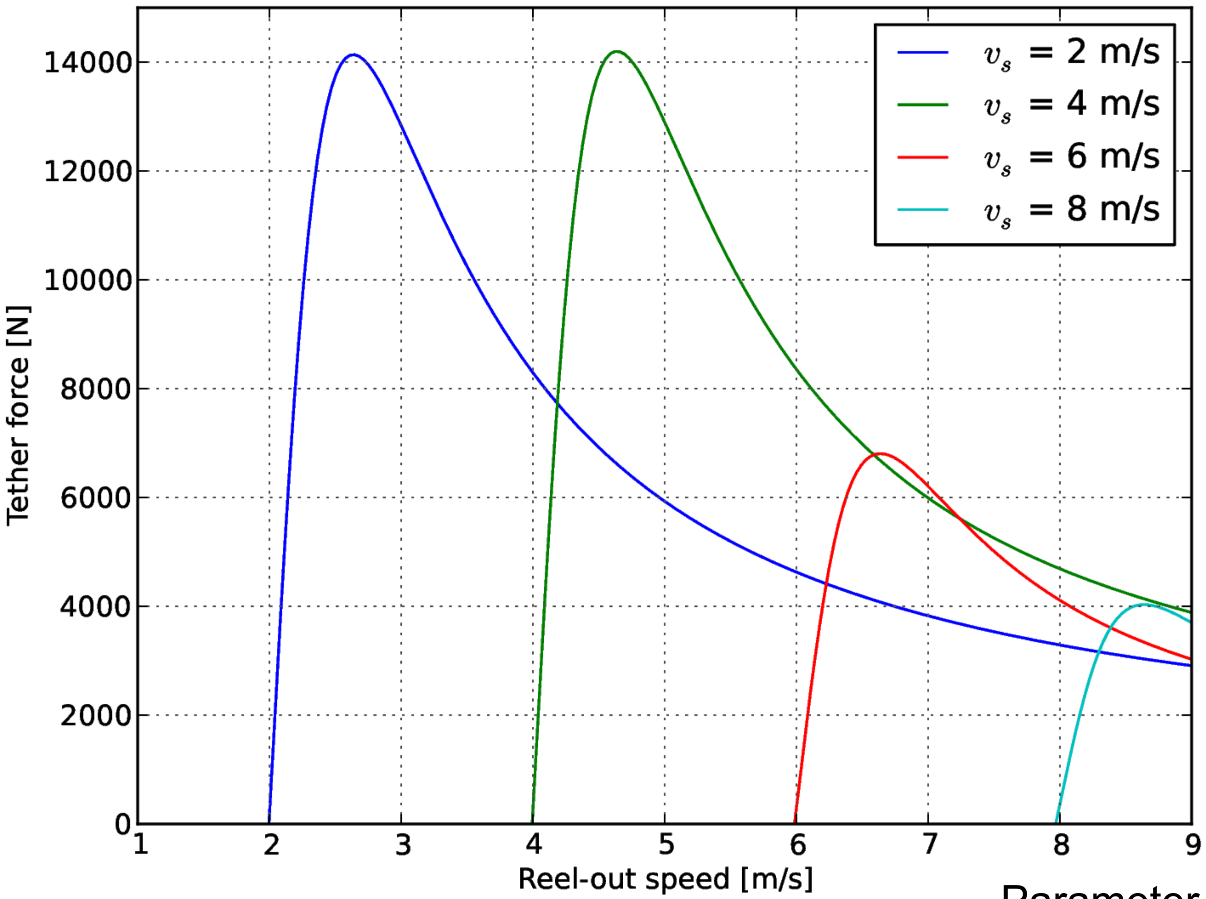
Currently:

A set value of 3200 N is needed to avoid force peaks above 4000 N.

Better force control → up to 25% more energy !

Asynchronous Generator Model

KPS Model Components



For up to 4 m/s synchronous speed the force/speed characteristics is nearly linear.

For higher velocities a linear model is not sufficient.

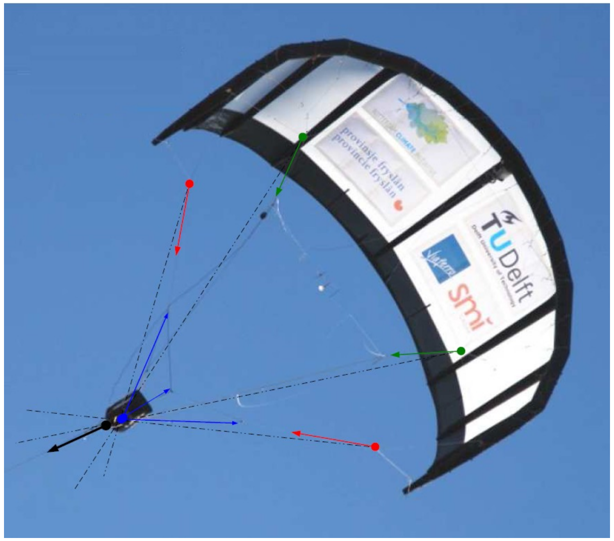
Parameter-fitted motor model [2]

Kite Control Unit and Sensor Models

KPS Model Components

The Kite Control Unit (KCU) model is taking the following effects into account:

- Limited actuator speed;
- actuator position controller;
- coupling between depower and steering;
- actuation controller delays.



Actuation vectors [3, p. 41]

The sensor models should take the sensor errors and sensor-delays into account.

Feature Overview

KPS Model Components

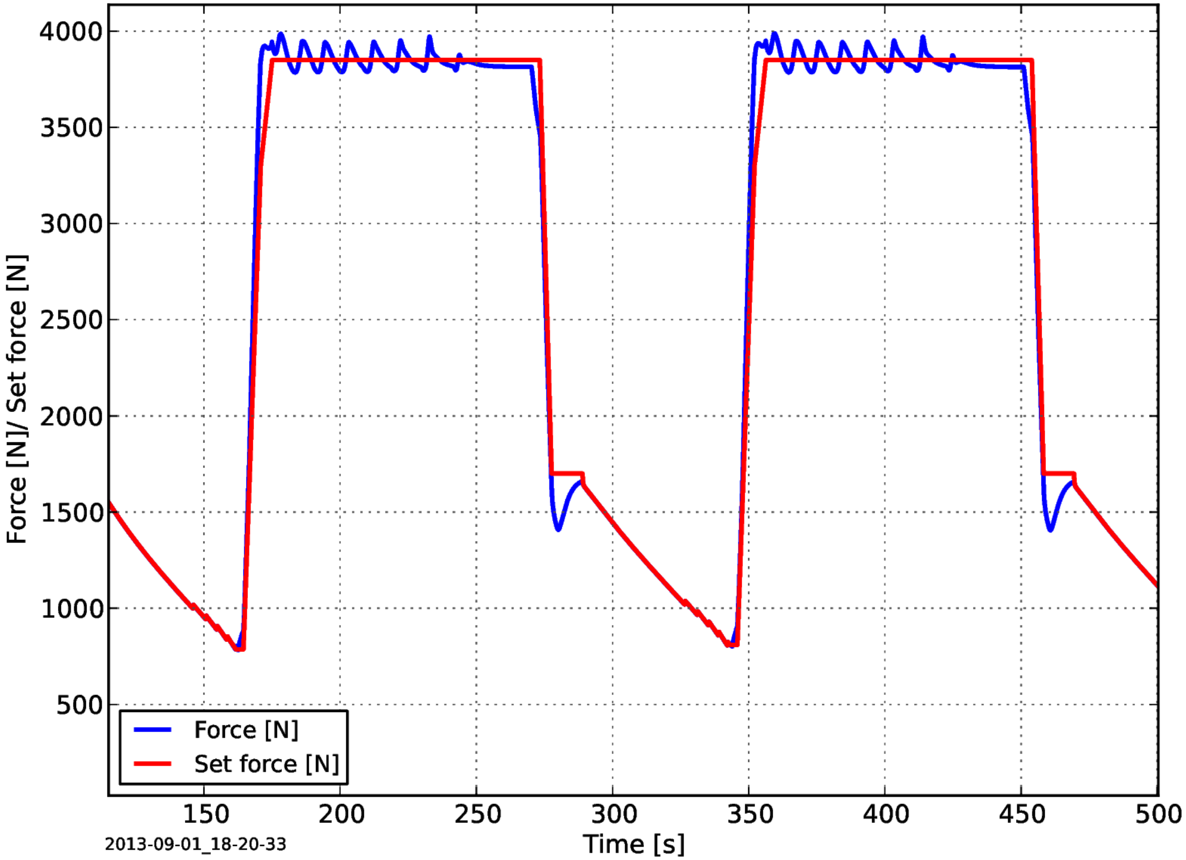
	3 DOF Model	4 DOF Model	4 Point Model	FEM Model	Spring/Damper	Segmented tether	Drag	Reel-out	Friction	Inertia	Async Generator	Sync Generator
Kite	x	F	F	F								
Tether					x	x	x	x				
Winch									x	x	x	F

	Actuator speed	Actuation pos controller	Coupling depower/ steering	Actuation delays	Sensor erros	Sensor delays	Wind speed	Wind direction	Turbulence	Gusts	Wind shear	Air density profile
KCU	x	x	F	x								
Sensors					F	F						
Environment							x	F	F	F	x	F

x: Implemented; F: Shall be implemented in the near future

Force Controller Optimization

Simulation Results

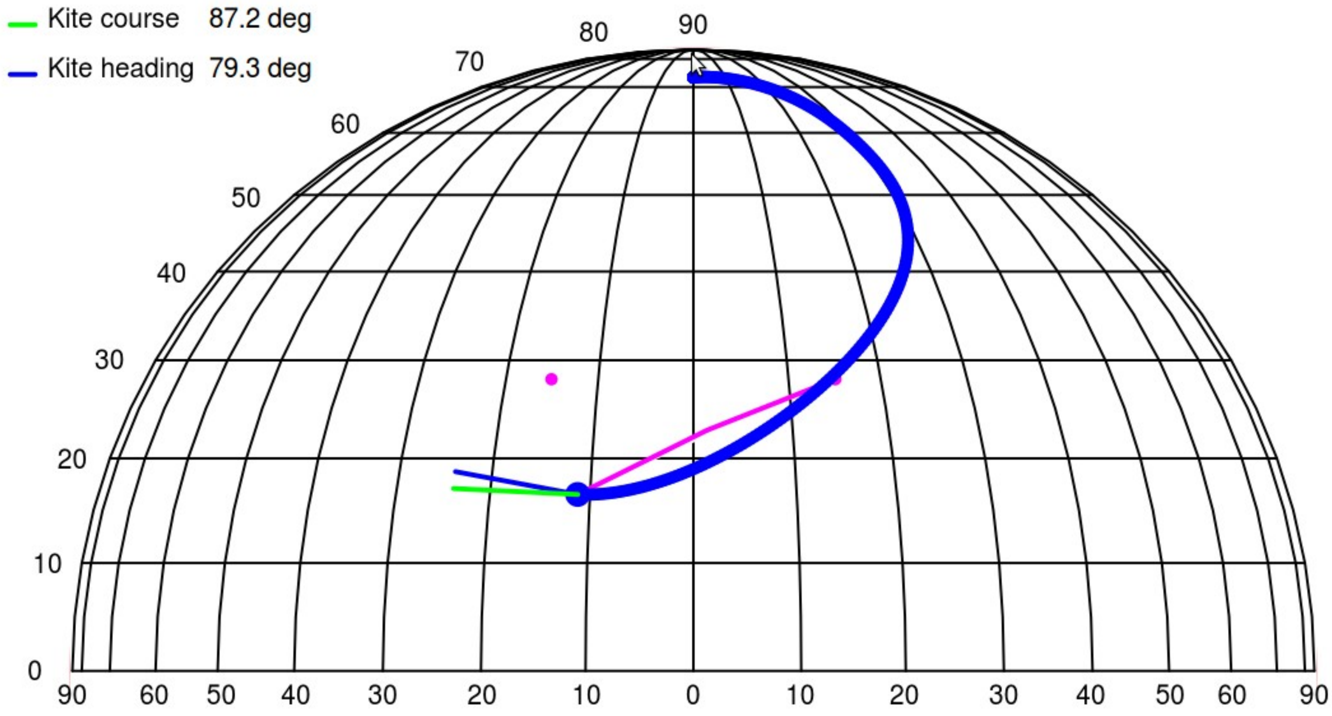


Improved force control by shaping the input signal of the force controller:

3800 N already possible.

Flight Path Planner and Controller

Simulation Results

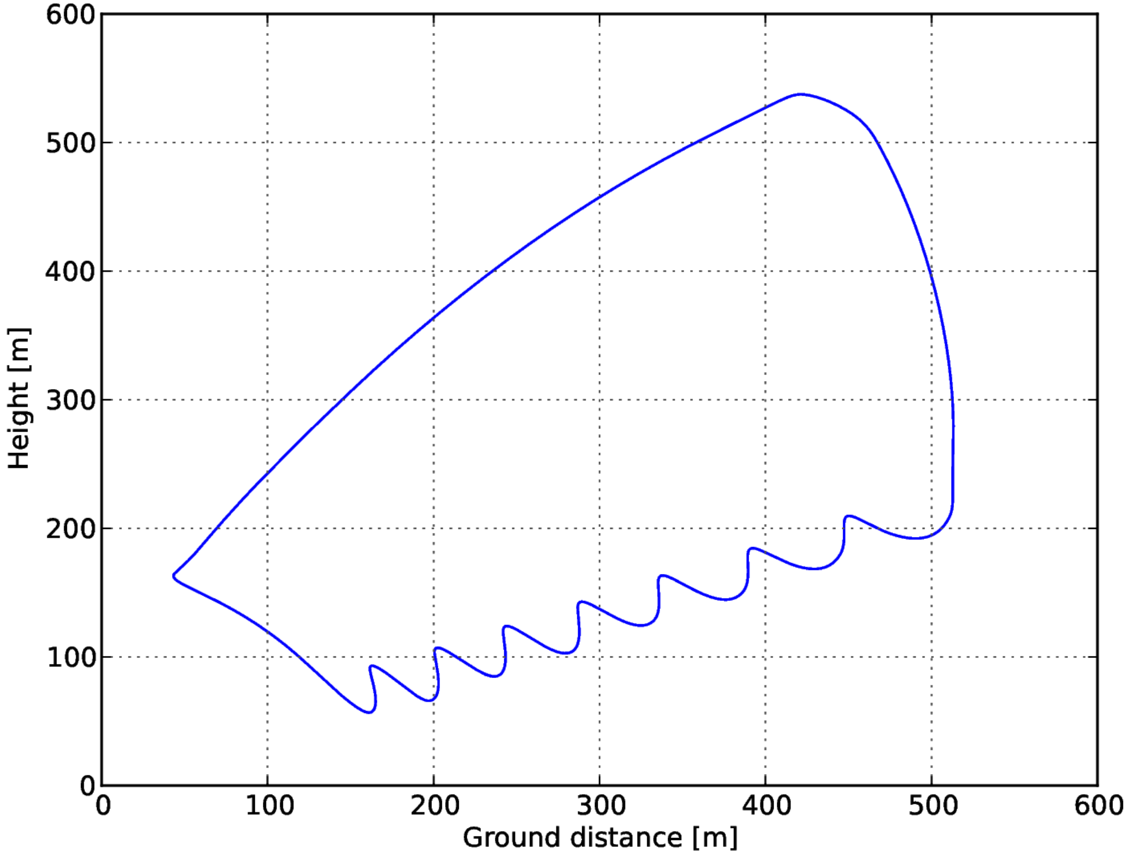


Different flight paths and their impact can now be tested easily.

Here one additional point was added for a smooth transition phase.

Flight Trajectory

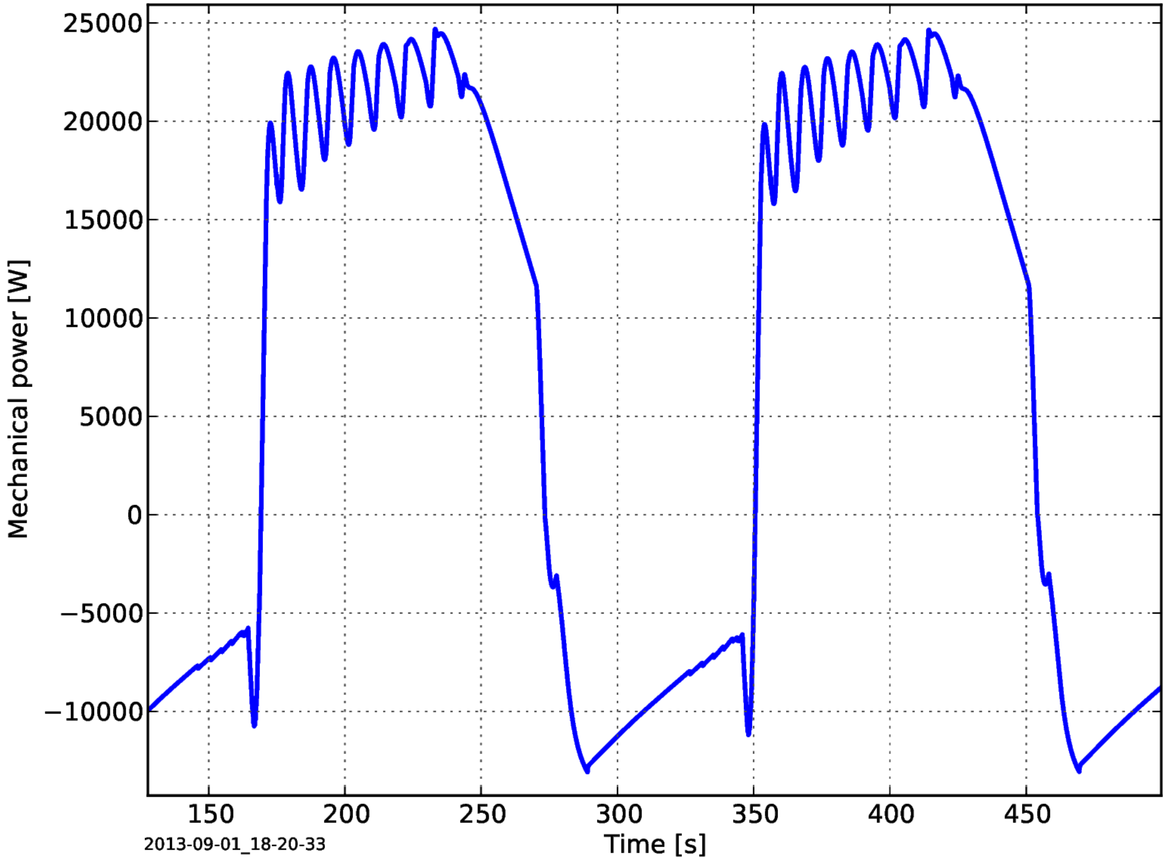
Simulation Results



A look at the trajectory from the side is usefull to optimize it for different operation conditions, e.g. for different maximal heights.

Power Output

Simulation Results



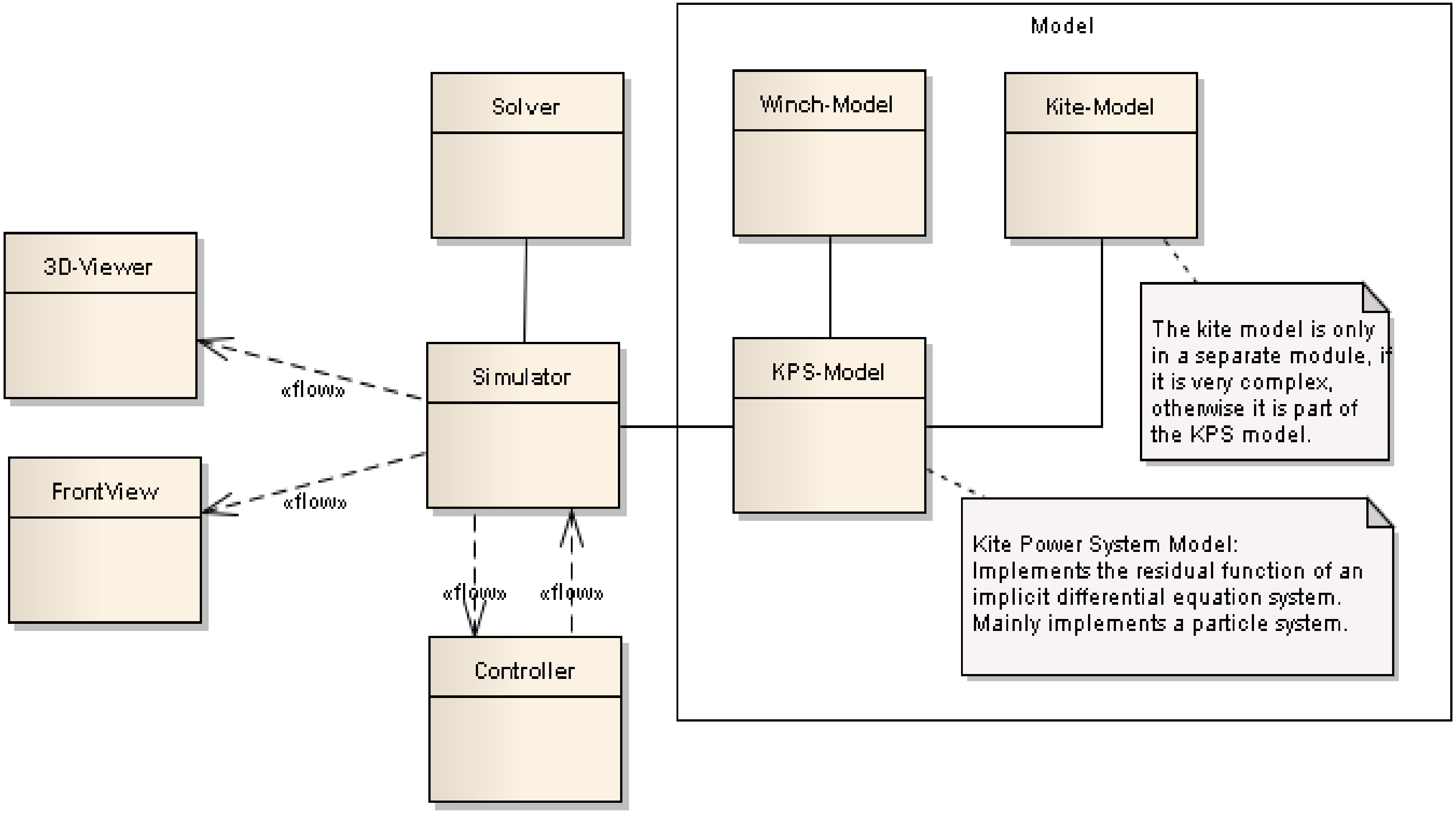
The reel-out power is already quite constant (still room for improvement) and nearly reaches the maximal power that the generator can handle.

In average:

$$p_{mech} = 7400W$$

Software structure

Simulation Software

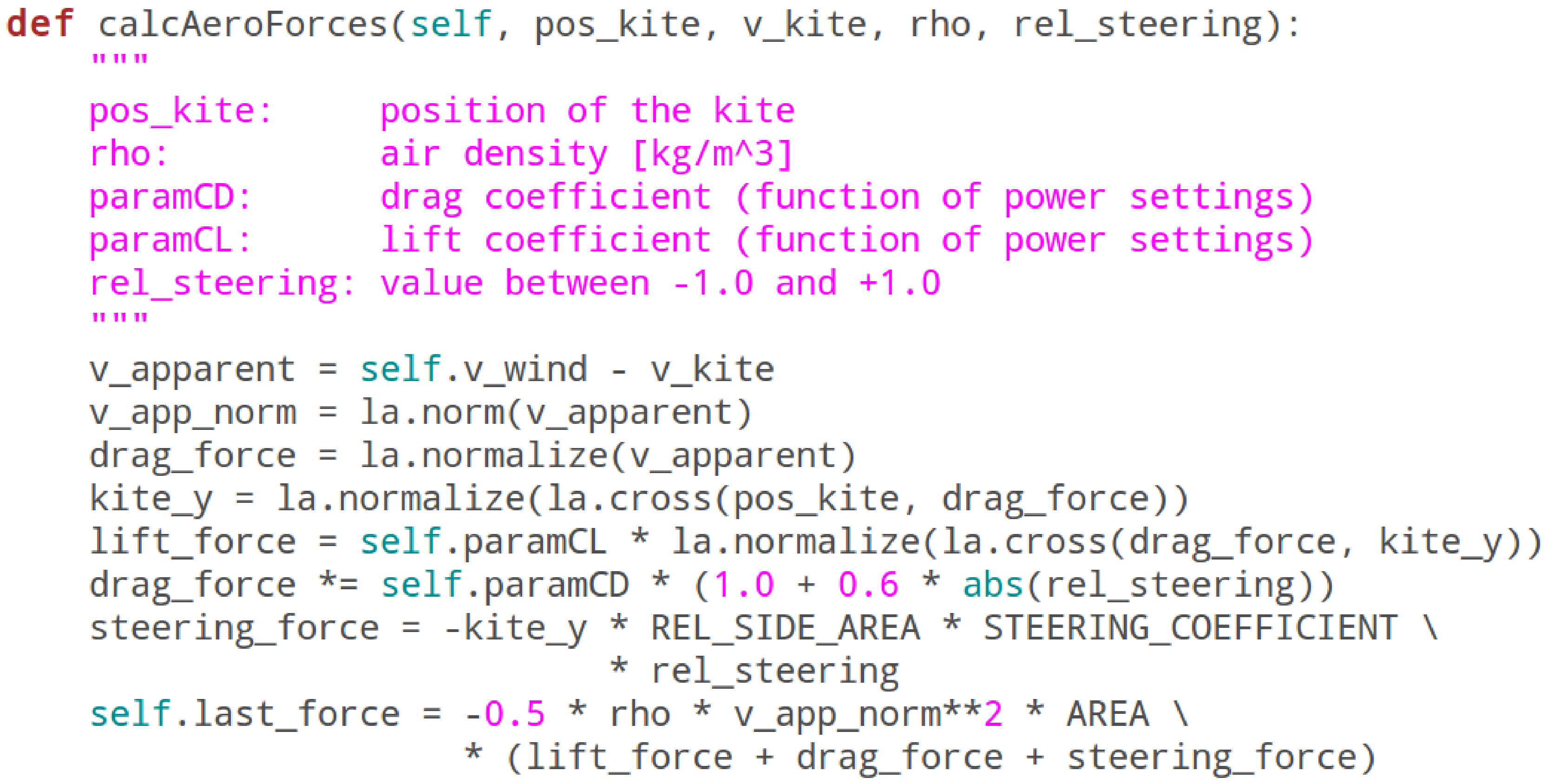


The kite model is only in a separate module, if it is very complex, otherwise it is part of the KPS model.

Kite Power System Model: Implements the residual function of an implicit differential equation system. Mainly implements a particle system.

Python as Modelling Language

Simulation Software: 3 DOF Kite Model

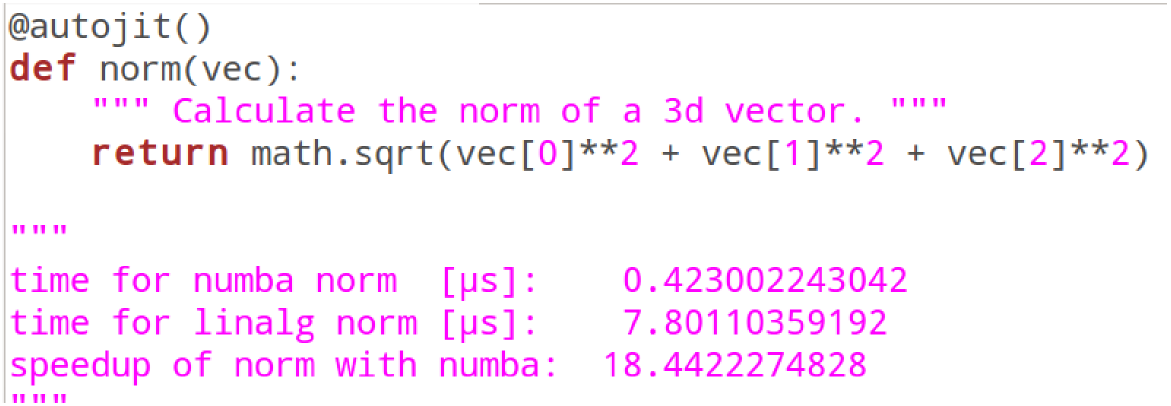


```python
def calcAeroForces(self, pos_kite, v_kite, rho, rel_steering):
    """
    pos_kite:     position of the kite
    rho:          air density [kg/m^3]
    paramCD:      drag coefficient (function of power settings)
    paramCL:      lift coefficient (function of power settings)
    rel_steering: value between -1.0 and +1.0
    """
    v_apparent = self.v_wind - v_kite
    v_app_norm = la.norm(v_apparent)
    drag_force = la.normalize(v_apparent)
    kite_y = la.normalize(la.cross(pos_kite, drag_force))
    lift_force = self.paramCL * la.normalize(la.cross(drag_force, kite_y))
    drag_force *= self.paramCD * (1.0 + 0.6 * abs(rel_steering))
    steering_force = -kite_y * REL_SIDE_AREA * STEERING_COEFFICIENT \
                             * rel_steering
    self.last_force = -0.5 * rho * v_app_norm**2 * AREA \
                           * (lift_force + drag_force + steering_force)
```

Numba as Python Compiler

Simulation Software

Numba is a just-in-time compiler for Python, that can increase the speed of Python code by a factor of up to 1000. It uses the Low Level Virtual Machine (LLVM) infrastructure. To compile a Python function it is sufficient to add one line of code: “@autojit”. Example:



```
@autojit()
def norm(vec):
    """ Calculate the norm of a 3d vector. """
    return math.sqrt(vec[0]**2 + vec[1]**2 + vec[2]**2)

"""
time for numba norm  [µs]:    0.423002243042
time for linalg norm [µs]:    7.80110359192
speedup of norm with numba:  18.4422274828
"""
```

For kite-power simulations a speed increase by a factor of 10 is realistic.

Selecting a Solver

Simulation Software

Because we have a very stiff equation system, formulated as an implicit equation system in the following form:

$$\boldsymbol{r} \;=\; F(t, \boldsymbol{Y}, \dot{\boldsymbol{Y}})$$

the choice of solvers is limited:

Different solvers from the Assimulo [6] software suite can be choosen:

The IDA solver delivers good results, but the RADAU5DAE solver delivers even better results: It has a parameter to choose how often the Jacoians shall be recalculated, and by reducing the number of recalculations the speed can be improved significantly compared to the IDA solver.

Summary and Conclusion

A modular frame-work for kite power system simulations was presented. Most of the components, that are needed for accurate and fast simulations are in place.

Different controllers can now be tested agains this model.

The model calculates the tether force with a high accuracy and a good time resolution (200 Hz easily possible). This makes it possible to simulate even kites on a short tether, as it is needed for launch and landing.

The implementation of the main model components in Python makes it easy for engineers with limited programming knowledge to contribute model components.

Future work

- *Integrate more kite models into the framework;*
- *implement a good environmental model;*
- *add scripting for automatic execution of different model scenarios;*
- *model validation.*

Interested in Cooperation?

- *Extending the model to other air-born wind-energy system types?*
- *Exchange of model components?*
- *Exchange of controllers?*

Literature

Phd research Uwe Fechner

[1] R. van der Vlugt, J. Peschel, R. Schmehl: "Design and Experimental Characterization of a Pumping Kite Power System". Chapter submitted to Airborne Wind Energy, Springer, 2013.

[2] E. Schreuder: “Analytical winch representation using a parameter-fitted asynchronous motor model.” Intermediate Report, TU Delft, 2013

[3] J. Ramirez: “Airborn Wind Energy – Data-driven LPV Modelling for Flight Control of a Kite Power Airborne Wind Energy Generator.” MSc Thesis, TU Delft, 2013

[4] A. Bosch, R. Schmehl, P. Tiso, D. Rixen: “Dynamic nonlinear aeroelastic model of a kite for power generation“. Submitted to AIAA Journal of Guidance, Control and Dynamics, 2012.

[5] Knappskog, H. (2011). Nonlinear control of Tethered Airfoils. MSc Thesis, Norwegian University of Science and Technology.

[6] C. Andersson, C. Führer, J. Åkesson and M. Gäfvert: Assimulo: A simulation package for solving ordinary differential equations. http://www.jmodelica.org/assimulo

[7] T. Oliphant, J. Riehl, S. K. Lam, H. Grecco, and M. Florisson. Numba: A Dynamic Python Compiler for Science. http://numba.pydata.org

Optimizer for Kite Power Systems

Phd research Uwe Fechner

[8] U. Fechner, R. Schmehl: "Model-Based Efficiency Analysis of Wind Power Conversion by a Pumping Kite Power System". DOI: 10.1007/978-3-642-39965-7_14, Airborne Wind Energy Book, Chapter 14, Springer, 2013.

[9] Dmitrey Kroshko,OpenOpt: Free scientific-engineering software for mathematical modeling and optimization, 2007, http://www.openopt.org/

Open Source Python Software

- Python, OpenOpt
- Ubuntu Linux

https://bitbucket.org/ufechner

Uwe Fechner
Faculty of Aerospace Engineering
Delft University of Technology
Tel : +31 15 278 8902
Email : u.fechner@tudelft.nl
Web : www.kitepower.eu