Irvine cable equations and neural networks

Jean-Pierre Merlet

Irvine cable equations and neural networks

Jean-Pierre Merlet

HEPHAISTOS project, INRIA Sophia-Antipolis, France
Jean-Pierre.Merlet@inria.fr

Abstract. A classical planar cable model has been presented in the Irvine textbook and is being used, for example, for the modeling of cable-driven parallel robot. It provides 2 equations involving parameters of the cable material, namely its Young modulus and its linear density, assumed here to be known, and 5 physical parameters namely the coordinates of one cable end-point B, the horizontal and vertical components of the force exerted at B and the length at rest of the cable. This model is extensively used in the modeling of devices where cables are involved (e.g the kinematics of cable-driven parallel robots) and the model analysis requires to be able to solve very efficiently this 2-equations system when it has 1 or 2 unknowns. We consider various cases where n physical parameters are known and we investigate how the Irvine equations may be exploited to compute the $5-n$ remaining parameters. In some cases where $n<3$it is possible to determine a closed-form solution for this $5-n$ parameters. But if $n=3$ it appears that only a numerical approach may allow to get the 2 remaining parameters. We present here a generic algorithm based on a mix of neural networks and deterministic algorithms allows one to get exact solution in that case.

Keywords: cable modeling, Irvine equations, cable-driven parallel robot, neural networks

1 Introduction

A cable model presented in Irvine textbook [4] (abbreviated here to Irvine model) is a planar model with a reference frame $A,\mathbf{x},\mathbf{z}$, where A is one of the end-point of the cable while $B(x_b,z_b)$ is the other end-point (figure 1).

The cable material is supposed to have E as Young modulus, μ as linear density and its cross-section area is A_0. The length at rest of the cable will be denoted L_0 while the force exerted on the cable at B has F_x/F_z horizontal and vertical components (note that the planar frame is defined so that $x_b>0, F_x>0$). The Irvine model provides the coordinates of any point of the cable according to its curvilinear abscissa s whose value is included in the range $[0,L_0]$.

$$x(s)=F_x\left(\frac{s}{EA_0}+\frac{\sinh^{-1}\left(\frac{F_z+\mu g(s-L0)}{F_x}\right)-\sinh^{-1}\left(\frac{-\mu gL0+F_z}{F_x}\right)}{\mu g}\right) \tag{1}$$

$$z(s)=\frac{F_z s}{EA_0}+\frac{\mu g\left(\frac{1}{2}s^2-L_0 s\right)}{EA_0}+\frac{\sqrt{{F_x}^2+\left(F_z+\mu g(s-L_0)\right)^2}-\sqrt{{F_x}^2+\left(-\mu gL_0+F_z\right)^2}}{\mu g} \tag{2}$$

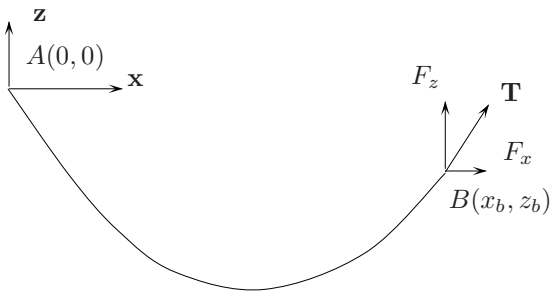

Fig. 1. Notation for a sagging cable

Consequently the coordinates of B are $x_b = x(L_0), z_b = z(L_0)$:

$$x_b = F_x \left(\frac{L0}{EA_0} + \frac{\sinh^{-1}\left(\frac{F_z}{F_x}\right) - \sinh^{-1}\left(\frac{-\mu g L0 + F_z}{F_x}\right)}{\mu g} \right) \tag{3}$$

$$z_b = \frac{F_z L_0}{EA_0} - \frac{\mu g {L_0}^2}{2EA_0} + \frac{\sqrt{{F_x}^2 + {F_z}^2} - \sqrt{{F_x}^2 + (-\mu g L_0 + F_z)^2}}{\mu g} \tag{4}$$

If we assume that E, μ, A_0 are known we have 5 physical parameters in these equations, namely x_b, z_b, F_x, F_z, L_0. These equations have been experimentally validated by measurements of the cables shape for cable-driven parallel robots (CDPR) [10, 11]. Other more sophisticated models have been proposed e.g. including torsion [2], fatigue [14], 3D [12], multi-toron [15] and lumped-mass [5] but the Irvine equations seems to be sufficient for dealing with a quasi-static cable mechanism.

The purpose of this paper is to determine how these equations may be used to calculate some of the physical parameters as soon as the other one are known (e.g. a trivial case is when F_x, F_z, L_0 are provided which allows one to calculate x_b, z_b). Note that F_x, F_z, L_0 appear in both equations while x_b, z_b appear only in one of them.

2 Dealing only with the z_b or x_b equation

Equation 4 has z_b, F_x, F_z, L_0 as parameters so that we have to fix 3 of the parameters in order to possibly calculate the 4th one. The possible combinations are $(F_x, F_z, L_0 \to z_b)_1$, $(z_b, F_z, L_0 \to F_x)_2$, $(z_b, F_x, L_0 \to F_z)_3$, $(z_b, F_x, F_z \to L_0)_4$. Combination 1 is trivial and the remaining combinations have already been treated in [7] where it has been shown that there is always a single solution for the unknown parameter. Equation 3 has x_b, F_x, F_z, L_0 as parameters but is more complex to manage because of the inverse hyperbolic function. The combination $(F_x, F_z, L_0 \to x_b)_1$ is trivial while for $(x_b, F_x, L_0 \to F_z)_2$ it has been shown in [7] that F_z may be obtained as the root of a 8th order polynomial. Regarding

$(x_b, F_x, F_z \to L_0)_3$ and $(x_b, F_z, L_0 \to F_x)_4$ it appears difficult to get a closed-form expression of the solution. However it has been shown in [7] that for both cases determining the unknown y (either L_0 or F_x) amounts to solve an implicit equation $H(y) = 0$ with $\partial H/\partial y > 0$ and $H(\beta) < 0$ where β is any strictly positive number. A consequence is that there is a single root of $H = 0$. Furthermore an upper bound for this root can be determined so that a dichotomy process may be used to determine efficiently the root.

3 Dealing with both the z_b, x_b equations

In that case we have 2 equations with 5 parameters so that we may consider two possibilities:

- fixing 4 parameters to get the last one that satisfy both equations. As obtaining x_b, z_b is trivial we have 3 combinations to consider that will lead to F_x or F_z or L_0. This problem will be called the *one parameter case*
- fixing 3 parameters to get the two last one. As $F_x, F_z, L_0 \to x_b, z_b$ is trivial we get 9 possible combinations that have to be considered. This problem will be called the *two parameters case*

3.1 Calculation for the one parameter case

The combination $x_b, z_b, F_x, L_0 \to F_z$ has been presented in [7] where we show that F_z is obtained as one root of a third order polynomial and that there is only one valid root. Regarding the other combinations we have shown in section 2 that the knowledge of z_b and of any pair of elements in the set $S = \{F_x, F_z, L_0\}$ is sufficient to obtain a single root for the remaining element of S.

3.2 Calculation for the two parameters case

Any problem in the two parameters case may be written generically as $y_1, y_2, y_3 \to y_4, y_5$ and amounts to solve 2 equations in 2 unknowns. This problem may be solve in two different ways:

1. use one equation to obtain an unknown as a function of the other one and reporting it in the other equation, thereby possibly obtaining a single solvable equation
2. solve numerically the whole system

Regarding approach 1 it appears that indeed one of the y_4, y_5 may be obtained from one of the equations (for example if $y_1 = x_b, y_2 = z_b, y_3 = L_0$ we have explicitly F_x as a function of F_z using only the z_b equation). However after this elimination the remaining equation is extremely complex, may be even difficult to solve numerically while it is unclear if its solving time will be lower than the one of approach 2.

Regarding approach 2 the literature is quite discrete about what method may be used to solve the Irvine equations, whatever the unknowns are, although gradient descent or Newton method have been mentioned. The drawbacks of these methods are that there is no guarantee that they will provide the solution and that their computation time may be

large. We have considered an original approach based on a Taylor expansion of degree n of both equations around some initial point F_x^0, F_z^0 that leads to two polynomials E_1, E_2 of degree n in F_x, F_z. Then calculating the resultant of these polynomials leads to a univariate polynomial of degree n^2 in F_x which is numerically solved. Only the positive roots are considered and are reported in E_1, E_2 for obtaining 2 polynomials in F_z. If these polynomials have a common root, then an approximate solution is found, that may be refined by using it as initial guess for the Newton scheme on the Irvine equations. An efficient implementation for $n = 3$ has shown that the solving time is low but the choice of the F_x^0, F_z^0 is crucial for obtaining the convergence of the Newton method.

As none of the proposed methods fully satisfy our constraints of obtaining (if possible exactly) the solution and of low computation time we have decided to investigate the use of neural networks but in an original manner.

4 Neural network for the two parameters case

The use of neural networks has been proposed for the kinematics of CDPR but few works integrate the Irvine equations in the model [1]. Furthermore kinematics of CDPR with the Irvine cable model has usually several solutions while we will see that neural networks are basically designed to provide a single solution although a scheme may be designed for the multiple solutions case (see for example the solving of the CDPR forward kinematics in [8]).

To illustrate the original approach we are proposing we will consider the combination $x_b, z_b, L_0 \rightarrow F_x, F_z$ but the method may be applied to any other combination. It must be noted that in that case the literature mentions that there is a single root for F_x, F_z also to the best of the author knowledge there is no formal proof of this uniqueness with a positive F_x. We will proceed with this assumption so that the use of neural networks may be considered.

The above combination is important for 3 dof CDPR that have 4 cables that output at known point A_i of the winches and are attached at the same point B on the platform while their L_0 lengths are measured. Indeed an external measurement system allows to measure the coordinates of B in the reference frame from which we can deduce the x_b, z_b of each cable. Obtaining the F_x, F_z will allow to obtain the shape of the cable, to determine the amount of slackness for each cable and the cable tension. But this calculation has to be done in real-time so that a very fast and accurate (ideally exact) solver of the Irvine equations is required. We will proceed here under the assumption that we are interested in solving the Irvine equations only over a set S of limited ranges for x_b, z_b and L_0.

4.1 Neural networks

There are several types of neural networks (NN) but one of the most commonly used is the *multi-layer perceptron (MLP)* [3]. A MLP has an input layer, an output layer and in-between hidden layers. A layer is constituted of neurons that receive as input a weighted sum of all the outputs of the neurons from the previous layer and map this input to the neuron output by using an *activation function*. Here we will use the

MLP as an universal approximation tool that receives as input x_b, z_b, L_0 and outputs an approximation of F_x, F_z. Each neuron of the input layer receives x_b, z_b, L_0 and the output layer has 2 neurons whose output will be respectively F_x and F_z.

A MLP is a *supervised* NN which requires a *learning set*, i.e. a large set of samples with x_b, z_b, L_0 and the corresponding F_x, F_z that are solutions of the Irvine equations. Note that such a learning set may be used for dealing with any combination just by picking 3 element inputs, the output being the 2 other elements.

In the *learning phase* of the MLP a stochastic optimizer try to find the neuron weights that allow to minimize a statistical index, called the *loss*, calculated from the errors between the MLP outputs and the training data over the whole learning set (e.g. the Mean Squared Error (MSE)). As the optimizer uses the gradient descent principle we have to define a *learning rate* i.e. a step size for the gradient estimation. A large learning rate allows for a fast descent but a too large one may lead to miss a local minimum while a low learning rate leads to a low decrease of the loss.

Hence the parameters of a MLP are the number of hidden layers, the number of neurons in the layers, the activation function, the loss function and the learning rate. There is theoretically a MLP that can approximate accurately any function but the parameters of this MLP cannot be determined in advance. Furthermore even if the loss function goes to 0 it just implies that the MLP will provide the exact F_x, F_z for the samples in the learning set but this does not guarantee that we will get exactly the F_x, F_z for a sample outside the learning set (this is called the *over-fitting problem*) and how large the errors will be. In practice the loss function is never 0 so that we have prediction errors even for the learning set but this may be acceptable if the error is small in percentage.

4.2 Creating a learning set

We assume that some methods such as the Maple solver, Newton method or any other numerical method are able to provide a solution in F_x, F_z of the Irvine equations for n different triplets (x_b, z_b, L_0) distributed over the set of range S. The number n of such triplets may be low (in our implementation we use 20 triplets). We have thus a set $\mathscr{S}$ of n solutions S_j for the Irvine equations defined by $S_j = \{x_b^j, z_b^j, L_0^j, F_x^j, F_z^j\}$ with j in $[1,n]$. We define the 3-dimensional *input space* whose dimensions are associated respectively to x_b, z_b, L_0 and we will use a *continuation* process to establish a learning set for the MLP. For this purpose we will consider in turn each S_j, which is constituted of the coordinates of a point P_j in the input space together with its solution F_x^j, F_z^j. We choose a random unit vector $\mathbf{v}$ in the input space and define a point M in the input space as $M_k = P_j + \lambda_k \mathbf{v}$. For $k = 0$ we set $\lambda_k = 0$ so that $M_0 = S_j$ and M_0 is stored in the learning set. We then define an iteration rule $\lambda_{k+1} = \lambda_k + \varepsilon_k$ where ε_k has a small value that is initially set to ε. We use the solution in F_x, F_z obtained for M_k as initial guess for the Newton method applied on the Irvine equations obtained at M_{k+1}. If the Newton scheme does not converge we divide ε_k by 2 and repeat the process. If during this process ε_k becomes lower than a very small threshold we stop the process as the Newton non convergence indicates that we are close to a singularity of the Irvine equations. If this occurs we select a new random $\mathbf{v}$ and start again from M_0. If the Newton scheme converge and if the difference between the $x_b, z_b L_0, F_x, F_z$ at M_{k+1} differs from the data

previously stored in the learning set by more than given thresholds, then the data at M_{k+1} are stored in the learning set. As soon as the number of sample obtained for a given S_j is greater than a fixed threshold we move to the next S_j.

There is however a drawback in this approach. If the S_j is such that the cable is very slack a change of the L_0, x_b, z_b leads to very small changes in the F_x, F_z while if the cable is taught even a small change in the L_0, x_b, z_b) may lead to very large changes in the F_x, F_z. We will see later on how this problem has been managed.

Note that we may also design a verification set in the same way by picking randomly samples M_k in the learning set that will be used as starting points. To ensure that we have no common sample in the training and verification sets we store in $\mathbf{v_t^k}$ the $\mathbf{v}$ vectors that have been used for calculating the M_k of the learning set and we just ensure that the $\mathbf{v}$ vectors used for the verification set with M_k as starting point are such that the angle between $\mathbf{v_t^k}$ and $\mathbf{v}$ has an absolute value larger than a fixed threshold.

4.3 Initial MLP results

We have performed intensive and systematic tests with the learning set proposed in the previous section, designing MLP with different number of layers, neurons and activation function. These tests have shown that for all MLPs the final loss was still large so that none of them were satisfying the accuracy requirement, the best one having an error between 200 and 300% on F_x or F_z on samples of the learning set.

This may be partly explained by the high variation of the data in the learning set for a given change in the L_0 as explained in the previous section. Furthermore if the training stop criteria is just based on the value of the loss function or of the learning rate, then the training time is relatively high.

4.4 Redesigning the process

As mentioned in the previous sections we have several problems with the current MLP process: the very large variation in the training data (A), the MLP prediction error (B) and the training time (C).

We manage problem (A) by simply creating 3 different learning sets based on the value of the cable tension: one for slack cable, one for moderately taught cable and one for highly taught cable. Note that this partition is somewhat arbitrary and may be adapted as shown in section 4.5. We started addressing problem (B) by using the MLP prediction obtained for the lowest loss function as guess for the Newton method: if Newton converge, then an exact solution is obtained. We define the *success rate* as the ratio in percentage between the number of Newton convergences and the number of samples in the learning set and we get a success rate of about 30%. For further improvement we take into account the specificity of the Newton method as described by the Kantorovitch theorem [13]. Without going into the mathematical details we may summarize this theorem as follows. Let G_0 be a guess for the Newton method at a small distance from the solution G. For the Newton method we have *essential variable (s)* that play a crucial role in Newton convergence: a small error between G_0 and G on these variables leads to Newton divergence. On the other hand for the other *non-essential* variable(s) a relatively large error between G_0 and G does not prohibit Newton

to converge. Unfortunately the set of essential variable is a local property that change with G_0. Let assume now that during the training phase we create a MLP as soon as the loss has decreased and we check the success rate of this MLP. It may be thought that the success rate will increase with the decrease of the loss. This is simply not completely true: indeed a decrease of the loss may be due to a large decrease on the errors for the non-essential variables while the errors on the essential variables has somewhat increased: this may lead to more failure of the Newton scheme. This is illustrated in figure 2 that shows the success rate as a function of the loss. At the start of the training the loss is 84.35 and the success rate 99.966% while for lower loss the success rate oscillates but is lower than 99.966% until the success rate reaches 100% for a 0.67 loss which allows to end-up the training. We have noted in that case that if we let the training go on the loss will decrease but the success rate never again reaches 100%. This example shows that it may be interesting to stop the learning phase if the success

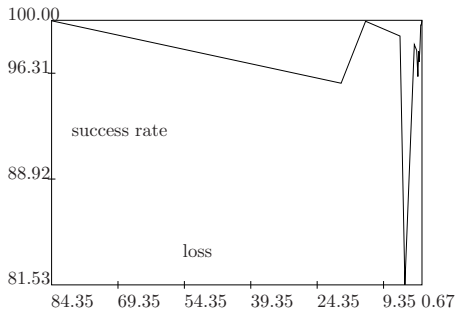

Fig. 2. Success rate as a function of the loss

rate reaches 100 % as this will allow to reduce drastically the learning time and solve problem (C). However this may be dangerous: as seen in the example a success rate close to 100% may be reached for the learning set even for a high loss and it is unclear if Newton may still converge for sample in the vicinity of the training sample. Hence we have adopted a prudent strategy: we store the model that has provided a 100% success rate for the learning set and we stop the training only as soon as we have found 3 times a 100% success rate model, the last one being our final model.

4.5 Experimental results

We have considered a cable with $E = 1e^{11}$ hPa, $\mu = 0.079$ kg/m, and a diameter of 4 mm. The allowed range for x_b is [0.01, 10]m, [-10,-0.1]m for z_b while L_0 is allowed to lie in [0.01,50]m. As mentioned earlier we have created 3 learning sets whose samples have been determined according to the difference d between the distance $||\mathbf{AB}||$ and L_0: very taught ($d \in [1e-4, 5e-4]m$), taught ($d \in [0.1, 0.5]m$) and slack ($d \in [0.5, 5]m$),

each set having about 3000 samples. MLP are built using Pytorch while the program that compute the success rate of a model during the training phase is written in C++.

Now we have to determine the number of layers and neurons that leads to the highest success rate. As the teaching time is relatively low (between 30 seconds and 5 minutes) we have conducted a systematic search regarding the MLP parameters. We have calculated the success rate for each learning set as a function of the number of layers from 2 to 6 and for an identical number of neurons in the hidden layers from 10 to 70 using a step size of 5 while 3 different classical activation functions were considered: ReLU, LeakyReLU and CELU. We use the Adam optimizer and the learning rate is adaptive: we start with a 0.03 rate and if after 50 iterations the loss has not decreased we divide the rate by 1.1. As soon as the learning rate is lower than $1e-6$ we stop the learning phase unless we have reached a 100% success rate. Note that Adam is a stochastic optimizer so that two runs with the same data may produce different results. Figure 3 presents part of the results for the ReLU activation function is ReLU for the very taught learning set (in this particular case we have deactivated the mechanism that stops the training if a 100% success rate is reached). We have 4 curves numbered from 2 to 5, one for each number of layer.

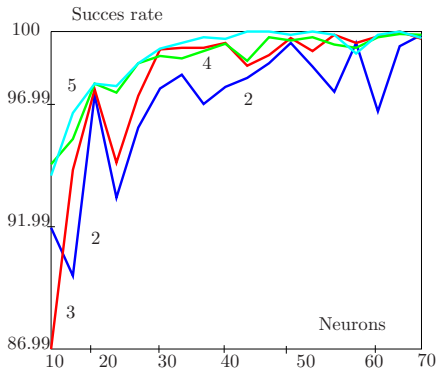

Fig. 3. Success rate as a function of the number of layers (the curve number) and the number of neurons in the hidden layer for the ReLU activation function

It can be seen that even with only 2 hidden layers we are able to reach a high success rate while for 4 and 5 layers we are able to reach a 100% success rate. It appears that in our case having six hidden layers and between 40 to 60 neurons per layer and using either the ReLU or CELU activation functions provides efficient models with 100% success rate over all the learning sets. We decided also to select two MLP for each learning sets: the one having the highest success rate (usually 100%) and the one having the lowest loss just to play it safe. We use therefore a total of 6 MLPs. The overall computation time for establishing the learning sets and training the MLPs was about 25 minutes. But this time may be drastically reduced by using specific parallel

processors devoted to IA such as the Jetson family: they have a massive array of GPU allowing the learning sets and training processing in a parallel way.

We have then established a verification set with 19652 samples and have run our solver on this set. The solver has been able to find the solution for all these samples (we say we have a *full coverage* of the verification set), which have been treated in a batch mode with 200 samples per batch, in an average computation time of 0.012 seconds per sample. However this time is not completely significant because the verification algorithm is written in C++ but has to use a Python program to exploit the Pytorch MLP model. Converting a Pytorch MLP model to C++ is possible but it is a difficult operation if one intend to get the best performances. However a significant reduction of computation time may be expected from this conversion.

A problem occurs if we have not full coverage of the verification sets i.e. a solution is not found for m samples $S_1,\ldots,S_m$. In that case the strategy is to create a learning set with S_1 as starting point and train a new MLP. We then check again the verification set as the new MLP will find the solution for S_1 but may find also solution for sample $S_k, k \neq 1$. If samples are still unsolved we repeat the process until we get full coverage of the verification set. We have also developed as trimming method that possibly allows to reduce the number of MLPs that are required to have full coverage.

5 Getting solutions bounds over a ball

If we are interested to get some performance data such as the maximal cable tension over a region in the input space just sampling the region is not completely satisfactory as we may miss the real maximum. But it is possible to obtain bounds for the performance in a ball centered at any point of the input space. More precisely we consider a point $\mathbf{X_0}$ of the input space and the solution $F_0 = (F_x^0, F_z^0)$ of the Irvine equations for $\mathbf{X_0}$. Let the ball $\mathscr{B}_i$ in the input space defined by the set of $\mathbf{X}$ such that $||\mathbf{X} - \mathbf{X_0}|| < \varepsilon$ where the norm $||$ is chosen as the infinity norm. It can be shown that if ε is small enough, then it is possible to calculate an α such that for any $\mathbf{X}$ in $\mathscr{B}_i$ the solution $F = (F_x, F_z)$ of the Irvine equations for $\mathbf{X}$ is included in ball centered at F_0 with radius α or in other words that $F_x \in [F_x^0 - \alpha, F_x^0 + \alpha]$ and $F_z \in [F_z^0 - \alpha, F_z^0 + \alpha]$. The determination of the ε, α cannot be explained here for lack of space but is based either the Kantorovitch theorem [13] or the Krawczyk operator [9] or the ε-inflation algorithm [6] and requires the use of interval analysis. As soon as the ranges for F_x, F_z have been obtained a simple interval calculation allows one to calculate an upper bound of the cable tension $\tau = \sqrt{F_x^2 + F_z^2}$ at B over the ball $\mathscr{B}_i$. Using this approach we estimate the tension not only at a single point of the input space but over a domain: this may allow for example to refine the maximal tension around a promising $\mathbf{X_0}$ point or possibly fully pave a region of the input space by balls so that we are guaranteed to obtain an upper bound of the cable tensions over the input space region.

6 Conclusion

In that paper we have shown that neural networks may be able to solve efficiently and exactly the Irvine equations although we cannot guarantee that this will be case what-

ever are the input elements. However the neural networks approach must not be considered as a black box and the solver is grounded on a mathematical analysis of the problem at hand and mixes neural networks and deterministic methods. We have also mentioned the possibility of obtaining lower and upper bounds for the solution over a domain of the input space. The computation time of the current solver is about 0.012s and the training time of about 25 minutes. But we believe that the use of highly parallel processor devoted to IA will allow to decrease the solving time to less than 1 ms and the training time to less than 5 minutes and we are currently working on the validation of these numbers. Having a very fast Irvine solver may allow to drastically decrease the simulation time of devices involving several cables.

References

1. Chawla I. and others . Neural network-based inverse kineto-static analysis of cable-driven parallel robot considering cable mass and elasticity. In *5th Int. Conf. on cable-driven parallel robots (CableCon)*, virtual, July, 7-9, 2021.
2. Ghoreishi S.R. and others . Analytical modeling of synthetic fiber rope. part II: a linear elastic model for 1+6 fibrous structure. *Int. J. of Solids and Structures*, pages 2943–2966, 2007.
3. Haykin Simon. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.
4. Irvine H. M. *Cable Structures*. MIT Press, 1981.
5. Kamman J.W. and Huston R.L. Multibody dynamics modeling of variable length cable systems. *Multibody System Dynamics*, 5(3):211–221, 2001.
6. Mayer G. Epsilon-inflation in verification algorithms. *Journal of Computational and Applied Mathematics*, 60:147–169, 1995.
7. Merlet J-P. Some properties of the Irvine cable model and their use for the kinematic analysis of cable-driven parallel robots. *Mechanism and Machine Theory*, 135:271–280, https://hal.inria.fr/hal-02426393v1, 2019.
8. Merlet J-P. Advances in the use of neural network for solving the direct kinematics of CDPR with sagging cable. In *6th Int. Conf. on cable-driven parallel robots (CableCon)*, Nantes, https://hal.science/hal-04017643/file/cablecon_2023_v2.pdf, June, 25-28, 2023.
9. Neumaier A. and Zuhe S. The Krawczyk operator and Kantorovich's theorem. *Journal of mathematical analysis and applications*, 149:437–443, 1990.
10. Riehl N. and others . Effects of non-negligible cable mass on the static behavior of large workspace cable-driven parallel mechanisms. In *IEEE Int. Conf. on Robotics and Automation*, pages 2193–2198, Kobe, May, 14-16, 2009.
11. Riehl N. and others . On the determination of cable characteristics for large dimension cable-driven parallel mechanisms. In *IEEE Int. Conf. on Robotics and Automation*, pages 4709–4714, Anchorage, May, 3-8, 2010.
12. Samuel W. and others . Synthetic mooring rope for marine renewable energy applications. *Renewable energy*, 83:1268–1278, November 2015.
13. Tapia R.A. The Kantorovitch theorem for Newton's method. *American Mathematic Monthly*, 78(1.ea):389–392, 1971.
14. Weher M., Pott A., and Wehking K-H. Bending fatigue strength and lifetime of fiber ropes. In *3rd Int. Conf. on cable-driven parallel robots (CableCon)*, Québec, 2017.
15. Wu Weiguo and Cao Xin. Mechanics model and its equation of wire rope based on elastic thin rod theory. *International Journal of Solids and Structures*, 102-103:21 – 29, 2016.