Format :

Title – Full title of the paper with hyper link to the pdf stored in box. This pdf is the annoted pdf with notes and comments for future me

Authors and affiliation – The authors their affiliation

Citations – Number of citations

Keywords – Some keywords from the paper

One liner summary – A summary of the paper in a single line describing the main idea (compulsory for every paper)

Summary – Summary of the paper in my own words (Optional)

Interesting things – A few pointers on the interesting things about this paper (Optional)

51)

Contact Models in Robotics: a Comparative Analysis

Lidec (the GOAT) - Inria

Year – Apr 2023

Journal – Only on Arvix for now

Citations – 1

One liner Summary – Survey of contact models used in recent robotics simulatrs.

Summary

- They expose limitations of existing simulators by inspecting both the physical approximations and the numerical methods that are at work
- They provide open source and generic implementation of the main robotic contact solver in C++
- Come up with a criteria to compare these models and then do a comparision between the physical and computational aspects of contact models and solvers
- Explore the impact of the choice of contact model in MPC for quadruped locomotion
- Did not understand a lot of this paper. Need to go through it once again

50)

A Modified Multiple Shooting Algorithm for Parameter Estimation in ODEs Using Adjoint Sensitivity Analysis

Ozgur Aydogmus – Turkey

Year – Aug 2020

Journal – Applied Mathematics and Computation

Citations – 4

One-liner summary – They modify the multiple shooting algorithm to use the adjoint method in calculating the objective and constraint function derivatives.

Summary

- The one liner summary pretty much covers this

49)

Scalable Differentiable Physics for Learning and Control

Yi-Ling Qiao – University of Maryland

Year – 2020

Journal – ICML

Citations – 88

One-liner summary - Provide a scalable framework for differentiable physics that can support a large number of objects and their interactions by localizing collision handling.

Summary

- Meshes are used as the general representation of objects and contacts are grouped into localized impact zones.
- Since they use meshes, they can handle both rigid and deformable bodies
- In the end their differentiable simulation method has linear complexity with respect to the number of objects and collisions
- They do some inverse problems and control problems with their differentiable simulator

48)

Understanding Automatic Differentiation Pitfalls

Jan Hückelheim - Argonne

Year – 12 May 2023

Journal – Under review

Citations – N.A.

One-liner summary – Discuss the problematic usage of AD in chaos, time-average oscillations, discretization's, fixed-point loops, lookup tables and linear solvers and show how the gradients provided are usually incorrect in these situations.

Summary

- Even when the functions are differentiable, if they are crazy, then the derivatives are not very useful
- The level of abstraction on which AD is applied is important and results may vary accordingly
- Branches will always cause problems -> Specially the way the branches are written
- Sometimes calculating the derivative of a function can incur more severe floating-point rounds errors.

47)

Rethinking Optimization with Differentiable Simulation from a Global Perspective - https://uwmadison.box.com/s/tnqew8pq1cygbn8hl5r8y7dm9ch218y4

Rika Antonova - Stanford

Year – 2022

Journal – CoRL

Citations – 9

Extensive summary of the paper is provided in the slides for the qualifying exam

46)

Do Differentiable Simulators Give Better Policy Gradients?

H.J. Terry Suh

Year – 2022

Journal – ICRA

Citations – 28

Extensive summary of the paper is provided in the slides for the qualifying exam

45)

Autonomous Drifting with 3 Minutes of Data via Learned Tire Models

Franck Djeumou – UT Austin

Year – May 2023

Journal – ICRA – Best paper runner up

Citations – N.A.

One-liner Summary – Use a Neural Network to represent a tire for a race car (toyota supra)

Summary -

- They use mainly two methods to develop these data driven tire models
    - A family of tire models using Neural ODEs – this one has slow inference
    - A tire model represented by a parametrization with curves generated from exp tanh functions. This one does fast inference and is real time.
- They do experimental evaluation on a modified toyota supra

- Basically the solution of the NODEs as well as the exp tanh functions represent the tire force curves. They basically use the NN to provide a bunch of paramerters that then go into an ODE which is solved to give the tire force curve.
- One critical point I have from the above is why not use an existing tire model with parameters and then learn the parameters with a NN? They basically do parameter learning but also don't give the prior information baked into these tire models when used.
- The features they use as input to the NN is the yaw rate, velocity, side slip angle and an estimation of the road friction force.
- They do optimization using tire forces directly, however the tire forces are not measured. The states are measured and using that the tire forces are estimated. These estimated tire forces are used as "ground truth"
- To me the results from the Fiala tire look good enough. This much effort is not necessary as the Fiala tire is fast to solve.

44)

Probabilistic Inference of Simulation Parameters via Parallel Differentiable Simulation

Eric Heiden – Nvidia

Year – May 2022

Journal – ICRA

Citations – 10 as of July 2023

One-liner Summary – Leverage parallel differentiable simulation to perform constrained stein variational gradient decent for parameter identifiability.

Summary -

- A really good paper with an amazing clear explanation of the approach.
- This approach is very much within reach
    - SVGD is implemented in pymc. However here a constrained approach is used. The constrains in this case come from the "multiple shooting" (trajectory is segmented) approach used and due to the uniform priors used. More details in the paper.
    - They only requirement is that we need parallel differentiable simultion. This is within reach for sure.
- The paper uses real data from a double pendulum, simulated data from a double pendulum and real data from a panda arm and infers upto a maximum of 11 parameters.

- They also compare to NUTS and beat it. Which is great because we can also do this comparision and see what happens for the vehicle models.
- Some missing info
    - How long did the calibration take?
    - Can this be done in an online setting. That will be a great paper as well since we can do this for the terrain slope estimation.

43)

Automatic Differentiation and Continuous Sensitivity Analysis of Rigid Body Dynamics

David Millard and Eric Heiden - USC

Year – Jan 2020

Journal – Can only find it on arvix

Citations – 7 as of July 2023

One-liner Summary – Compares in brief (and not very well) automatic differentiation of the entire time for the simulation and using continous sensitivity analysis. Does not explain in sensitivity analysis how the matrices are evaluated

Summary -

- Apart from the comparision in the ways of getting sensitivites (see one liner summary), they use the differentiable simualtor to then do model predictive control.
- They use Stan to do reverse mode AD on the entire simulation and compare it to the continuous sensitivity analysis method. No details about how the matrices are calculated in the continuous sensitivity method.
- They also compare the numerical difference method.
- They find that the reverse mode AD is extreamly slow and they actually don't use it for the controller experiment.

42)

CVODES: An ODE Solver with Sensitivity Analysis Capabilities

Radu Serban –

Year – Sept 2003

Journal – ACM

Citations – 223 as of July 03

One-liner Summary – Explains the different options available in CVODES which helps do continuous sensitivity analysis

41)

The Discrete Adjoint Method: Efficient Derivatives for Functions of Discrete Sequences

Michael Betancourt – Columbia

Year – Feb 2020

Journal – N.A. Just a technical note on arvix

Citations – 7 as of June 29

One-liner Summary – Really good derivations for Discrete and Continuous Adjoint Methods

40)

Discrete Adjoint Sensitivity Analysis of Hybrid Dynamical Systems

Hong Zhang – Argonne National Lab

Year – 2017

Journal – IEEE Circuits and Systems

Citations – 14 as of June 29

One-liner Summary – Mathematical formulation of Discrete Adjoint Sensitivity for DAE systems

Summary -

- Goal was to read this to better understand Discrete Adjoint Sensitivity Analysis. I think it helped a bit. Its similar to the Continous in the fact that you still need to solve so many more equations. However, these equations are all discretized equations. (Are the ODEs? Not clear yet.)
- Its from the perspective of power systems which made stuff more confusing.

39)

A Comparison of Automatic Differentiation and Continuous Sensitivity Analysis for Derivatives of Differential Equation Solutions

Yingbo Ma – Cambridge and a bunch of folks from Julia computing

Year – 2021

Journal – IEEE (thinks some conference though)

Citations – 4 as of June 28

One-liner Summary – Compares the continuous sensitivity analysis with discrete sensitivity analysis and in each they use the some elements of AD.

Summary-

- investigate the performance characteristics of Discrete Local Sensitivity Analysis implemented via Automatic Differentiation (DSAAD) against continuous adjoint sensitivity analysis (CASA). Within CASA AD is used in different ways.
- I understand completely CSA and CASA and I will probably write a technical report on this. However, I am unsure for DSAAD is done. They do not have an explanation of that in the paper. Ideally the report I write needs to be with this as well and then seeing mathematically how they compare.
- **They conclude that using forward DSAAD is faster for small system of ODE's (100 ODEs + parameters) compared to reverse mode DSAAD, CSA and CASA.**
- In terms of scalability, they say that CSA and CASA scale much better than forward DSAAD. The reasoning for this is interesting. They say that since DSAAD runs tape-based AD (which is 2 orders of magnitude slower on Nonlinear PDEs than source transformation), this is inefficient for differential equations because

"Notably, pure tape-based reverse-mode automatic differentiation did not perform or scale well in these benchmarks. The implementations have been generally optimized for the usage in machine learning models which make extensive use of large linear algebra like matrix multiplications, which decrease the size of the tape with respect to the amount of work performed. Since differential equations tend to be defined by nonlinear functions with scalar operations, the tape handling to work ratio thus decreases and is no longer competitive with other forms of derivative calculations"

- They say that using DSAAD with good AD tools that can do good source transformation might make it fast even for larger problems.

38)

FATODE: A LIBRARY FOR FORWARD, ADJOINT, AND TANGENT LINEAR INTEGRATION OF ODES

HONG ZHANG – Virginia Tech -> Argonne National Lab

Year – 2014

Journal – SIAM Journal of scientific computing

Citations – 29 as of June 28

One-liner Summary – ODE solver with discrete sensitivities written in FORTRAN

Summary -

- Read this paper partially to understand the differences between the continuous sensitivity approach and the discrete sensitivity approach -> CVODES uses the continuous sensitivity approach which is we differentiate first and then discretize whereas this package uses the discrete sensitivity appraoch where we first discretize and then differentiate.
- I was however unable to catch the difference between the 2. Both seem too similar to each other.

37)

ADD: Analytically Differentiable Dynamcs for Multi-Body Systems with Frictional Contact

Geilinger – ETH Zurich

Year – Dec 2020

Journal – ACM transaction on Graphics

Citations – 53 as of June 27

One-liner Summary – An analytically differentiable multi-body dynamics package that has some good explanation for how friction and contact is handled and also uses a fully implicit time integration scheme.

Summary:

- Their friction models has various setups that tradeoff between simulation accuracy and smoothness of objective function landscapes.
- Normal and tangential forces are handled through "**soft constraints**". I think by soft constraints they just mean the spring damper model of friction. They say that this is okay as long as you use a **fully implicit time stepping scheme.**
- They use their simulator in three settings
    - Real2Sim parameter identification
    - Control
    - Self-supervised learning
- Their simulator works for both rigid bodies and soft bodies – this is amazing and first of kind as far as I am aware.

- The friction model is "**smooth**" and balances the tradeoff between differentiability and accurate modelling well. This smooth frictional contact model in the limit must approach the discontinuous nature of the physical contacts.
- **They do analytical differentiation rather than doing num diff because they do not understand how to apply autodiff to implicit solvers.**
- **They investigate different friction models from multiple points of view: accuracy of forward simulation results, smoothness of optimization landscapes and suitability for gradient based optimization.**
- Section 4 contains of many studies comparing the different friction models for accuracy and speed.
- Some cool material parameter estimation and robot throwing examples are also done.
- They also find that graident based optimization absolutely destroy gradient free optimization techniques for these examples.
- **Penalty based contact models, especially in the normal component, are sufficiently accurate when combined with implicit time integration.**
- When persistent static friction is necessary, hard constraints for friction and soft constraints for the normal force are used in what is called the **hybrid method**.

36)

NeuralSim: Augmenting Differentiable Simulators with Neural Networks

Eric Heiden - USC

Year – May 2021

Journal – ICRA

Citations – 103 as of June 21

One-liner Summary – They augment NNs with a differentiable rigid body simulator they built to generate hybrid simulators where the Neural network augment the analytical simulator for complex stuff like friction, viscous forces and so on.

Summary -

- I think this is a really good paper and hits very close to home based on what we are trying to do in the lab.

- They a**ugment a differentiable rigid body physics engine via NNs** where the NNs learn nonlinear relationships between dynamic quantities and thus model effects that are usually not accounted for in traditional simulators.
- **These NNs don't learn just the residual that needs to be added to a state to make it close to reality, but rather are embedded inside the physics engine and kind-off augment the RHS of the physics engine.** This is very well explained by the image below:

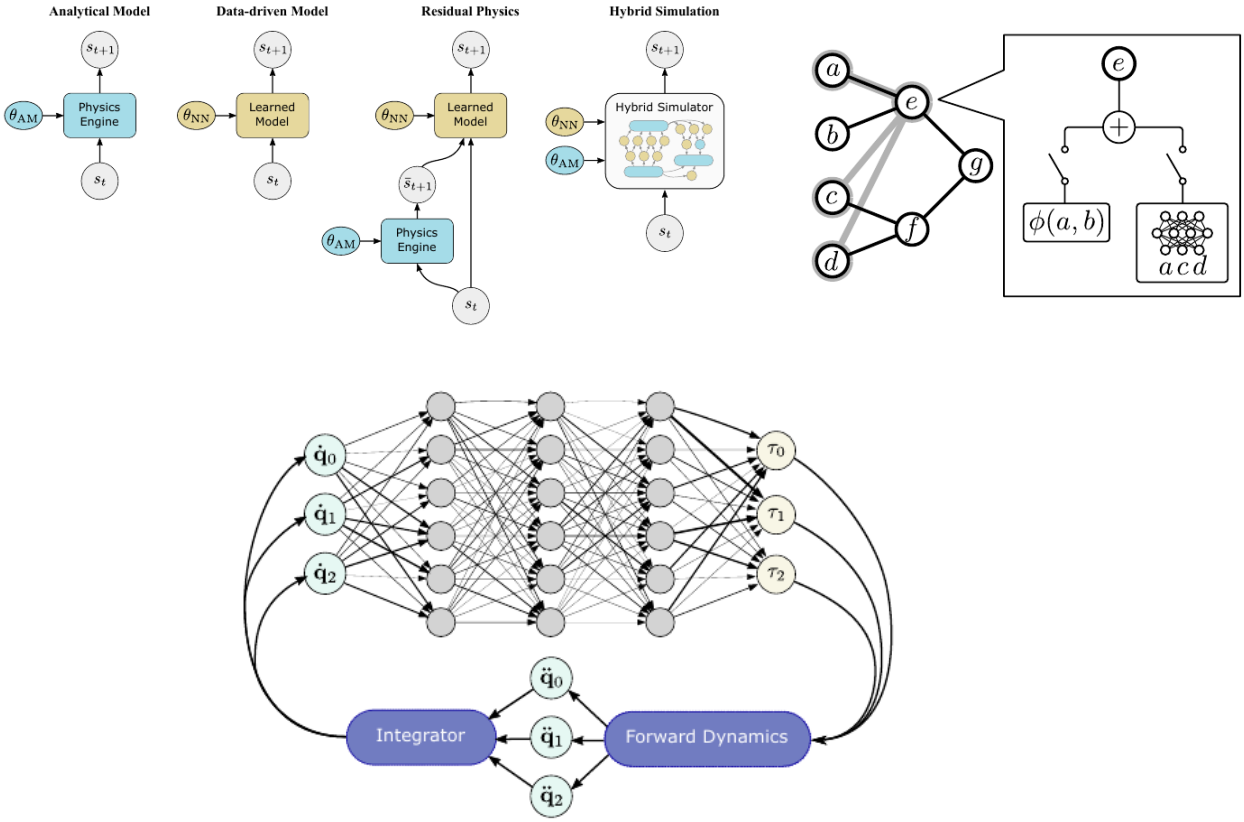


Fig. 3: Exemplar architecture of our hybrid simulator where a neural network learns passive forces τ given joint velocities $\dot{\mathbf{q}}$ (MLP biases and joint positions $\mathbf{q}$ are not shown).

- The use-case of hybrid simulation can be best summarized as - "**Even if the best-fitting analytical model parameters have been determined, rigid-body dynamics alone often does not exactly predict the motion of mechanisms in the real world. To address this, we propose a technique for hybrid simulation that leverages differentiable physics models and neural networks to allow the efficient reduction of the simulation-to-reality gap**"
- Some nice global optimizers are used. Pagmo is the library that does this in a massively parallel scale.
- Tiny Differentiable Simulator (TDS) is implemented in C++ and exposed to python via PyBind11. Autodiff available through **Ceres, Stan Math, CppAD with code gen. CppAD with code gen found to be the fastest with NumDiff actually not very far behind.**

- The most interesting part of the paper is what they do with this simulator and NN
    - **Experiment 1 – NN augments the friction force to provide a more realistic friction force matching real-world data.** They train a NN to augment the lateral 2D friction force with real-world data from planar pushing tests. The NN takes inputs of the objects current post and its velocity as well as the normal and the penetration depth at the point of contact between the tip (that's pushing the object) and the object. These weird quantities can be given as input here because they can be directly taken from the simulator being augmented. This is only possible if the simulator is differentiable.
    - **Experiment 2 – NN learns entirely new phenomena which was not accounted for in the analytical model.** In this case, they have 2 models of a water snake robot. One is in mujoco where the fluid is modelled and thus applies a viscous force on the robot. This is the data generator and represents the real world. The other model is using TDS, however, there is no fluid modelled and so the robot is as if it moves in vacuum. Then, they use a NN to apply to necessary viscous forces on the simplistic robot model without fluid and match with the mujoco model. The results are not amazing but decently good.
    - **Experiment 3 – NN learns which dynamical quantities actually influence the dynamics of the observed system**. They do this very very smartly using a sparse group lasso penatly for the cost function. Here, the weights in the first layer is forced to be sparse. Such cost function sparsifies the input weights so that the augmentation depends on only a small subset of physical quantities, which is desirable when simpler models are to be found that potentially overfit less to the training data while maintaining high accuracy. They also have an L2 loss on the weights form the upper network layers which penalizes the overall contribution of the neural augmentation to the predicted dynamics since we only want to augment the simulation where the analytical model is unable to match the real system. This is best explained by the below picture

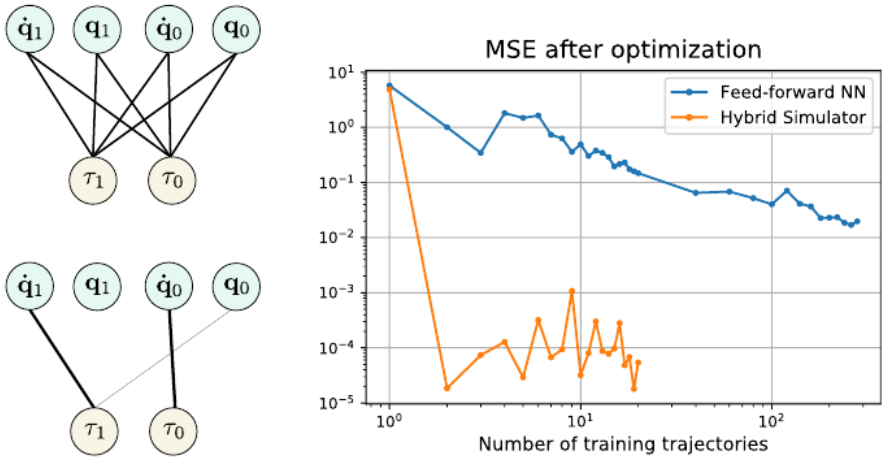


Fig. 10: *Left:* Neural augmentation for a double pendulum that learns joint damping. Given the fully connected neural network (top), after 15 optimization steps using the sparse group lasso cost function (Equation 2) (bottom), the input layer (light green) is sparsified to only the relevant quantities that influence the dynamics of the observed system. *Right:* Final mean squared error (MSE) on test data after training a feed-forward neural network and our hybrid physics engine.

35)

Interactive Differentiable Simulation

Eric Heiden - USC

Year – May 2020

Journal – Only preprint in Arvix available

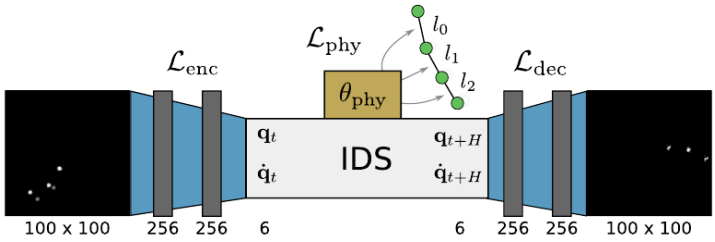
Citations – 45 as of June 21

One-liner Summary – They introduce a differentiable physics engine specifically for inference of physical properties of rigid body systems.

Summary -

- Introduced engine does not model contacts. Uses ABA for the dynamics and then a half-implicit integrator.
- Stan used to do autodiff – they say analytical diffs are not “general”
- No comparison for differentiation.

- Experiment 1 for showing its use is – They infer the length of links for image data. They learn an encoder to map the image to generalized coordinates. Then, they learn the parameters of the model in IDS to map the generalized coordinates in the current time step to the next time step and they finally learn a decorder to map the generalized coordinates in the next time step to the image of the next time step



- Second experiment – They learn the ideal robot arm (joint position and link lengths) to match as closely as possible the specified end-effector trajectory. They call this "Automatic Robot Design"
- Third experiment – They do MPC on a cart pole system while also learning the parameters of the model through optimization. This is called "Adaptive MPC"

34)

DiffTaichi: Differentiable Programming For Physical Simulation

Yuanming Hu - MIT (same author as below paper)

Year – 2020

Journal – ICLR

Citations – 265 as of June 19

One-liner Summary – A programming language for differentiable physics simulation

Summary-

- Uses a two scale Automatic differentiation framework.
    - Each kernel/function is made automatic differentiation using source transformation (they write their own complier – augment "taichi" langauge complier)
    - Then, a light weight tape records the order and arguments of these kernel calls and for the reverse mode AD just calls these kernels in the reverse order
    - This is done because if we use SCT on a simulation with 1000 of steps, it will result in code with "poor flexibility" and long compilation time.

- DiffTaichi is built off Taichi which is a programming language for physics simulation – advnatges of Taichi over traditional languages is listed in the paper.
- They show 10 examples of simulators written with this language -> ranging from rigid body dynamics to fluids, albeit very simple ones.

33)

ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics

Yuanming Hu - MIT

Year – 2019

Journal – ICRA

Citations – 63 as of June 19

One-liner Summary – A real-time differentiable simulator for soft robotics.

Summary -

- They built a real-time, differentiable hybrid Lagrangian-Eulerian physical simulator for deformable objects based on the Moving Least Squares Material Point Method (MLS-MPM).
- This can run fast on the GPU
- Most of the gradients for the MLS-MPM are derived analytically. Seems like for the analytical differentiation, they use symbolic gradients through a cached “memo” object.

32)

OptNet: Differentiable Optimization as a Layer in Neural Networks

Brandon Amos - CMU , now at Meta AI

Year – 2017

Journal – ICML

Citations – 323 as of June 19

One-liner Summary – Quadratic programs as a layer of the Neural network with capabilites to differentiate through them for backprop.

Summary -

- They use implicit differentiation and matrix differential calculus to derive the gradients of a quadratic optimization problem from the KKT matrix of the problem.
- They also in the process develop a GPU version of the quadratic optimization solver (GPU-based primal-dual interior point method (PDIPM))
- Did not understand much of the optimization math but now know that optimization problems can also be a layer of the Neural Network.

31)

End-to-End Differentiable Physics for Learning and Control

Filipe de A. Belbute-Peres - CMU and Co-Authors MIT

Year – 2018

Journal – NeurIPS

Citations – 191 as of June 19

One-liner Summary – They build a 2D differentiable simulator that simulates rigid body dynamics via a linear complementarity problem (LCP) where the differentiation is done analytically.

Summary -

- Why analytical gvradients? - this allows us to use general simulation methods for determining the non-differentiable parts of the dynamics (namely, the presence or absence of collisions between convex shapes), while still providing a simulation environment that is end-to-end differentiable (given the observed set of collisions).
- Why is finite differences not used? - (1) the high computational burden of finite differencing when computing the gradient with respect to a large number of model/policy parameters; and (2) **the instability of numerical gradients over long time horizons, especially if contacts change over the course of a rollout. -** Don't really know what this means or how it comes into play.
- Three use cases where differentiable simulation is useful is shown
    - System identification – Mass of a link is idetified
    - Prediction – 2D billiards ball movement is predicted.
    - Control – Cart-pole system is controlled using a iLQR controller

30)

A Differentiable Physics Engine for Deep Learning in Robotics

Jonas Degrave - DeepMind

Year – March 2019

Journal – Frontiers in Neurorobotics

Citations – 109 as of June 15

One-Liner Summary – Closed source differentiable Rigid Body dynamics code that they show is useful using 4 case studies.

Summary

- Seems like a closed source rigid body dynamics package that can find derivatives with respect to control parameters.
- They use theano to do the automatic differentiation.

The automatic differentiation makes the model 10 times slower

- 4 examples used where the differentiable simulator completely destroys learning using gradient free approaches.
- Briefly discusses the concept of Backpropogate Through Time (BPTT) -> Needed since you are differentiating with respect to the input conditions I think

29)

Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization

Yuval Tassa -UWash – Authors of Mujoco

Year – Oct 2012

Journal – IROS

Citations – 544 as of June 15

One-Liner Summary – Made some improvements to MPC

Summary

- They improved regularization and line search in MPC.
- They used Finite Differences in parallel to compute the sensitivities of the dynamics model.

28)

Whole-body Model-Predictive Control applied to the HRP-2 Humanoid

J. Koenemann - Universiry of Freiburg and LAAS-CNRS

Year – Oct 2015

Journal – IROS

Citations – 160 as of June 14

One-Liner Summary – They do MPC on a humanoid robot where the MPC runs on a workstation and comunicates inputs to the robot via wifi

Summary -

- MPC gives the sequence of control inputs and the optimal trajectory. Works with Differential Dynamic Programming. I did not understand all the controls aspects of the paper fully.
- The MPC runs on a workstation whereas the tracking control that converts the MPC controls to the actual motor controls runs on the robot
- The MPC uses a MujoCo dynamics model and uses finite-differences for the first order gradient information. This numerical differences is done with CPU parallelization on the workstation (not on the robot).
- Did not understand all the aspects of the controls so migth need to read more in order to understand.

There are around 4 papers here that reference the use of finite differences to calculate derivatives which can be used to make some points

27)

Automatic differentiation of rigid body dynamics for optimal control and estimation

Markus Giftthaler - ETH Zurich

Year – Nov 2017

Journal – Advanced Robotics

Citations – 31 as of June 14 2023

One-Liner Summary – Using Operator overloading Autodiff and then generating C code for expression graph for rigid body dynamics to dramaically speed up the use of autodiff.

Summary -

- They first do operator overloading using CppAD. Then using CppAdCodeGen to generated C code for the operator overloaded C++ code. This C code is then blazingly fast and can outperform inefficiently computed hand derivatives.
- They compare with numDiff and AutoDiff **but at run time**. NumDiff is only on a single core. They find that AutoDiff and NumDiff are pretty similar in computational speed. Errors of NumDiff is 10^-6 while Autodiff is in the range og 10^-13. However, the C generated sensitivity code is super fast and is faster than inefficiently calculated analytical diffs.
- Another interesting thing is that they found that forward mode diff was faster than reverse mode diff even when the input dimension was much larger than the output dimension (54:18).
- To demonstrate that this Autodiff approach to taking gradients helps speed up in some practical application, they used it in SLQ Optimal Control. The contact formulation is smoothed out by using sigmoid functions. Autodiff is used for linearizing the system (getting the state and control input jacobian)
- **Both NumDiff and Autodiff converge in the same number of iterations thus showing that the lack of precision does not impair convergence performance.** However, in their case since the cost of a single NumDiff is much higher than Autodiff, they see a huge boost in Wall clock time (500%)
- They also use this in dynamic trajectory replanning. Again same number of iterations but Autodiff is much faster by wall clock time.

Completely open source, all their code is available here!

Analytical Derivatives of Rigid Body Dynamics Algorithms

Justin Carpentier - LAAS – CRNS (some french national lab?)

Year – 2018

Journal – Robotics Science and Systems

Citations – 71 as of June 13 2023

One-Liner Summary – Analytically derived derivatives of robot dynamics with respect to generalized coordinates for both forward (ABA algorithm) and inverse dynamics (RNEA)

Summary -

- First derived the gradients for the forward pass and the backward pass of the RNEA algorithms using simple chain rule. RNEA algorithm is used for solving inverse dynamics.
- Then found a hack to use these gradients even in the forward dynamics without having to rederive them based on ABA. In essence, the found the relation between the gradient of inverse dynamics and the gradient of forward dynamics.
- Performance comparison done on 3 robots of 7, 18 and 36 DOF.
- Compared with finite differences but on single core – reason stated is
    - While it may be possible to parallelize finite differences, most of the current robots do not have enough computational resources to do it. We then decided to implement them on a single core.
- Running the analytical derivatives of inverse dynamics is 3 times slower than evaluating the inverse dynamics itself. Moreover, we have a ratio from 7 to 26 with respect to finite differences
- Performing analytical derivatives is at most 4 times slower than a call to the forward dynamics function. The ratio between analytic derivation and finite differences goes from 4 up to 10.
- In terms of error – Finite differences gave 10^-6 rounding errors.
- Compared with automatic differentiation from another library – RobCoGen.
- They were on average 50% faster than RoboCoGen.

ACCELERATED POLICY LEARNING WITH PARALLEL DIFFERENTIABLE S IMULATION

Jie Xu - Nvidia/MIT

Year – July 2022

Journal – ICLR

Citations – 27 as of June 7 2023

One-Liner Summary – Current SoTA in Model based RL for robotic tasks.

Summary -

- Uses parallel differentiable simulation with pytorch to train a Short Horizon Actor Critic network (a policy network and a value network) to perform very well (maximize reward).
- The differentation is done "analytically" using AD and mainly comes into play while calculating the losses with respect to the network parameters of which the dynamics is a layer.
- The "paralell" aspect comes into play to sample N trajectories from a particular state. Trajectory is a set of state action pairs.

24)

Mastering Atari Games with Limited Data

Weirui Ye - Tsinghua University

Year – 2021

Journal – NuerIPS

Citations – 74 as of June 6 2023

One-Liner Summary – Current SoTA in Model based RL for Atari games with limited data.

Summary

- Did not understand many of the technical details
- Main takeaway was that the dynamics of the model and the Reward function estimation are very important when using the Monte Carlo Search Tree algorithm to do RL.
- They learn this "environment" model in a very straight forward way.

23)

A Brief Survey of Deep Reinforcement Learning

Kai Arulkumaran - Imperial College London

Year – Sep 2017

Journal – IEEE Signal processing

Citations – 1037 as of June 5 2023

One-Liner Summary – Overview of Reinforcement learning techniques in which the policy is a neural network.

Summary

- A good review paper. Main takeaway was the basic understanding of Reinforcement Learning and the division into Value based methods and Policy based methods.
- Did not read into the detail of the different kinds of methods, just did a high level overview.

1)

A Review of Automatic Differentiation and its Efficient Implementation

Charles C. Margossian – Columbia

*Published 12 Nov 2018 – **146 citations** as of May 8 2023*

Keywords – Automatic Differentiation

One liner summary: AD implementation details discussed and compared along with an overview of how a few packages do things.

Summary -

The paper discusses first an overview of how AD works with a log gaussian likelihood example. Explains both Forward mode and Reverse mode and shows that Reverse Mode is better when inputs are much larger in number compared to inputs. Then the two methods commonly used for AD are discussed: Source Transformation and Operator Overloading. Some optimizations in operator overloading is also discussed. There is a great table in page 17 that describes the advantages and disadvantages of these 2 methods and the packges that use them.

Interesting things -

- Casadi has good support for Python but it works on source code transformation so will need to rewrite the entire model.
- ADEPT, CodiPack and SACADO are some of the fastest tools available.
- Table on page 17 super useful for quick summary

2)

Differentiable Simulation

Stelian Coros – ETH, Miles Macklin – Nvidia, Bernhard Thomaszewski – ETH, Nils – TUM,

*Presented on Dec 14 2021 at SIGGRAPH – **2 citations** as of May 9 2023*

Keywords – Differentiable Simulation

One liner summary: It's a course that provides an introduction to the theory and practice of differentiable simulation of dynamic mechanical systems with applications in robotic manipulation, parameter estimation and fluid capture.

Summary -

First discusses the theoritical framework of Automatic Differentiation and some important math behind it. Then different people come and talk about its applications in various different fields. It's a good quick read.

Interesting things -

- *85 is a great slide about where automatic differentiation would for into the current simulation framework. There is also an additional frictional contact model that is differentiable that is shown in the slide*
- *106 is a great slide about the use cases of differentiable simulation. The column is the optimization criteria and the row labels are the decision variables that are optimized for. Model parameters, control parameters and policy parameters can be optimized for motion goals and model-data mismatch using differentiable models.*

3)

AI-IMU Dead-Reckoning

Martin Brossard, Axel Barrau and Silvere Bonnabel – PSL Research institute in Paris

*Published December 2020 – **87 citations** at May 13th*

Keywords – State Estimation, Inertial Estimation

One liner Summary: Uses only IMU with Invariant EKF and a Neural Network to do state estimation. The Neural Network learns the Sensor noise covariance matrix from just raw IMU signals

Interesting things:

- *Could be implemented in our state estimation stack.*
- *Introduced to IEKF which could be the way to use IMU measurements to do state estimation.*
- *An IMU noise model also discussed which could be implemented in Chron*

4)

Towards Differentiable Agent-Based Simulation

Phillipp Andelfinger , University of Rostock Germany,

Published January 2023 - 1 Citation as of May 17th

Keywords : Agent-based simulation, optimization, Backpropagation

One liner Summary : Answers majorly the question: Is gradient based optimization using differentiable simulations of multi-agent systems more efficient than gradient free optimization?

Summary:

Really really good paper. A lot of questions discussed!

- The paper generally studies differentiable simulation in the context of multi-agent-based models. It mainly explores x questions
    - What needs to be done to make a non-differentiable model into a differentiable one? - All discrete or step functions (such as if statements) need to be made smooth and such. Page 3,4 is about all of this.
    - Then, once we have this completely differentiable model (using Adept!), how different is it from the non-differentiable one in terms of
        - **Output.** How close is the output? - The output is pretty close. Some differences but usually negligible if the smoothing approximations are good.
        - **Performance** – how much slower is the differentiable model? - Its much slower. The fully differentiable model is about 200 times slower. However, they then selectively induce smooth approximations in certain regions and find that a smart differentiable model will be 16 times slower for up to 1000 agents (example used is vehicles).
    - Then the differentiable model, with different gradient based optimizers are compared with a non-differentiable model with gradient free optimizers to see if the overhead caused by the differentiable model is cancelled out by the efficiency of optimizer using gradient information. The optimization is with respect to a lot of parameters (2500 in one case and 10000 in another)
        - Turns out that even though the differentiable simulations are 16 times slower, **gradient based optimizers using these models converge quicker for all cases except 1. The one failure case is due to the vanishing gradients problem which I buy (page 16).**
    - There is then a small study on combining gradient based and gradient free optimizers and exploring the sparsity in differentiable simulators (see page 19)
    - Some challenges discussed are
        - Identifying the model aspects for which gradient information is required.
        - Quantifying uncertainty in system output given the uncertainty in the parameters with differentiable simulators discussed – Differential error analysis. Another thing for uncertainty quantification in model response is – Smooth Interpretation
        - IPA (Infinitesimal Perturbation Analysis) discussed as another form of gradient computation. Needs to be looked at.

5)

On Correctness of Automatic Differentiation for Non-Differentiable Functions

Lead : Wonyeoil Lee – KAIST

Journal – NIPS

Published on Dec 6 2020 – Cited 28 times as of May 17th

Keywords – Automatic Differentiation

One Liner Summary – Automatic Differentiation systems give the correct derivative if the function is *piecewise analytic under analytic partition (PAP)*

Summary:

Paper is very math heavy but conveys the following point

- **If the function is a PAP function** (mathematical definition in paper) - from what I understand PAP functions are almost the same as piecewise continuous function except the "piecewise" happens according to "analytic partitions" and the "continuous" is replaced by "analytic" (which means infinitely differentiable". This means that PAP are functions that are differentiable "almost everywhere".
- **Then you can take intensional derivatives. –** for all purposes, we don't care what this means because of the next statement
- **These intensional derivatives are proven to always exist and coincide with standard derivatives for almost all inputs.** (proposition 8 on page 6)
- **These intensional derivaties are what "most" autodiff systems compute.**
- **Additionally, they claim that most functions used with autodiff systems are PAP –** I highly doubt this claim since its probably "most functions used in ML are PAP"

So, in conclusion, **if we have a PAP function,** that is essentially a certain category of not always differentiable function, **then we can rely on autodiff to give us the correct derivatives**. The main thing is understanding when a function is PAP and when its not. Additionally, this is only *a* correctness condition for autodiff systems. There are many more and this is no way the gold standard. **Additionally, *correctness* of autodiff systems do not necessarily imply the correctness of gradient decent/HMC/variational inference built upon those systems. For that there is something else called Clarke subdifferential (need to study optimization for this).**

6)

Real-time Model Predictive Control and System Identification Using Differentiable Simulation

Sirui Chen – Hong Kong University

PI- Karen Liu – Stanford University

Journal – IEEE Robotics and Automation

Published on Nov 2022 – Citations 2 as of May 18 2023

Keywords – Differentiable simulators, Online system Identification, Real time MPC

One Liner Summary – Develop a fast differentiable physics simulation framework that simultaneously performs system identification and optimal control using observations in real-time.

Summary:

Best paper I have read to date. Exactly what I am interested in. The main work done here is:

- They show that when you have a differentiable physics simulator, you can do fancy things:
  - You can use it to optimize system parameters online while the robot moves and parameters change -> This happens on the modelling thread
  - You can parallelly run a control thread that uses the most recent system parameters to control the robot to do a particular task -> This happens on the planning thread.
  - They show that to do both of these tasks efficiently in parallel, you need to have a *fast* and *differentiable* physics simulator and I agree!
- There is a lot of innovation on the system identification side that is enabled due to the differentiable model
  - The main challenge is that some observations may not help identify parameters due to
    - Non identifiability
    - Observations being out dated
  - For this they **assess the confidence in the estimated parameters using numerical analysis of the dynamic equations**. What they basically do is that once the parameters have been identified, they put them into the matrix M, C and G (Newton-Euler form) and assess the rank of these matrices. If its low rank, they see which columns contribute to this low rank and p**ut a low weight on the parameters** that are associated with that column. They can do all this quickly because they can evaluate Jacobian's quickly because the model is differentiable through AD. -> ***I think we can do better here... we can do this based on information theory which will rely lesser on the model and will be more general.***
  - One more AMAZING thing is that, they incorporate actually "**actively controlled system identification**" - T**his means that if they find the weight on the identified parameters**

to be consistently low, I.e., the parameters are consistently non-identifiable, then they switch the planning thread to find state trajectories that maximizes the parameter excitation for the system parameters of interest! This basically means that the robot then explores in order to identify system parameters and abandons the task at hand. Once the weights are high again, the robot goes back to the original task.

- Results shown that this method outperforms
    - Methods that use stochastic optimization
    - Methods that do not do the weighing of the identified states
- Over
    - 4 simulation experiments – cart pole, inverted pendulum, robot arm, moving an elastic rod
    - 1 real experiment – robot arm
- They say that optimizing complex trajectories with contact remains challenging still. A more effective controller is also needed.

7)

Dual Online Stein Variational Inference for Control and Dynamics

Lucas Barcelos - University of Sydney et al

Fabio Ramos – NVIDIA

Conference - Proceedings of Robotics: Science and Systems (RSS) - Citations 12 as of May 18

YouTube video - RSS 2021, Spotlight Talk 85: Dual Online Stein Variational Inference for Control and Dynamics

Keywords – Online system identification, Variational Inference, Stochastic control

One-liner summary – Develop a variational inference algorithm to estimate distribution over model parameters and control inputs online

Summary:

Very interesting paper that has a different approach to the above. Opposite approach to the above problem where here no differentiable model (this is actually not clear) is needed and distributions over parameters and control inputs is obtained instead of point estimates. Main Ideas are this:

- **Distribution over control inputs is learnt and then a control input is sampled from it. The distribution learn minimizes the KL Divergence between it and the target distribution which minimizes the expected cost of a task**. This is done to accommodate stochastic dynamics and their robustness in modelling uncertainty. **The MPC problem is thus formulated as a Bayesian**

Inference task whose goal is to estimate a distribution over control policies given the state and the observed costs.

- This distribution over control policies is parameterized by theta. Then they learn a probability distribution that tells the probability that theta generates an optimal set of trajectories. The control input is then sampled` from this distribution and used.
- A major advantage of this is that it makes full use of stochastic dynamics so we don't need to take the mean of the posterior of the parameters defining our dynamics and then feed it into the model for the controller. We can use the entire distribution itself. **So, the optimality of a given trajectory now depends on its *expected* cost over the entire distribution of p(model_parameters) and not just model_parameters!**

- **Simultaneously, the distribution of model parameters is also learnt online using variational inference.** The method used here is pretty straight forward:
  - We start with a prior distribution over the parameters and move from this distribution to the posterior distribution every time an observation is made. This update is made over multiple steps L. The prior distribution is represented just with a set of particles.
  - The posterior distribution is just a set of points as well. However, to move to the next distribution, with variational inference, we need the gradient of the likelihood as well as the gradient of the prior distribution with respect to model parameters**. The prior distribution, which is just a set of points is then approximated to a closed form distribution using Gaussian Mixture Models with a fixed diagonal covariance matrix. -** Major doubt I have here is whether we need a model differentiable with respect to model parameters over here or not. It does seem like we need one because we need to evaluate the gradient of the likelihood in the variational inference step.
- Only interesting result is with a real vehicle they try to estimate the position of the center of mass for a circular trajectory. This is a very simple inference with just a single parameter so its not much. They actually don't do that well, out simple MPC appraoch with the 4 DOF model does much much better. The approach is interesting but I don't think is the most efficient. Would be interesting to try it out though.

8)

Efficient computation of derivatives for solving optimization problems in R and Python using SWIG-generated interfaces to ADOL-C

K. Kulshreshtha - University of Paderborn, Germany

S.H.K Narayanan – Argonne National Lab

Journal – Optimization methods and Software

Citations – 2 as of May 19 2023

Keywords – Automatic differentiation

One-Liner Summary – They wrapped a C++ operator overloading package Adol-C into Python and R

Interesting things -

- Shows that wrapping the entire library itself is possible. However, we will want to only wrap only our module not the entire library.

9)

A Benchmark of Selected Algorithmic Differentiation Tools on Some Problems in Computer Vision and Machine Learning

Filip Srajer – ETH Zurich (work done while he interned at Microsoft)

Andrew Fitzgibbon – Microsoft

Journal – Optimization methods and Software

Citations – 32 as of May 19

One-Liner Summary – Compares AD tools (both Source Transformation and Operator overloading) from different languages and benches on 3 problems.

Table 1.: List of tools. OO: operator overloading, ST: source transformation: F: forward, R: reverse.

Language	Tool	Approach	Mode
C++		Manual (by hand)	
C++		Finite differences	
C++	Adept [15]	OO	F, R
C++	ADIC2 [17]	ST	F, R
C++	ADOL-C [28]	OO	F, R
C++	Ceres Solver [1]	OO	F
C++	clad [27]	ST via compiler	F
C/Fortran	Tapenade [14]	ST	F, R
F#	DiffSharp [5]	OO	F, R
MATLAB	ADiGator [18]	ST via OO	F
MATLAB	ADiMat [7]	OO via ST	F, R
MATLAB	MuPAD [25]	Symbolic	
Julia	ForwardDiff.jl [19]	OO	F
Python	Autograd [16]	OO	F
Python	Theano [4]	Symbolic	

Interesting things:

- Results on Gaussian Mixture Model test

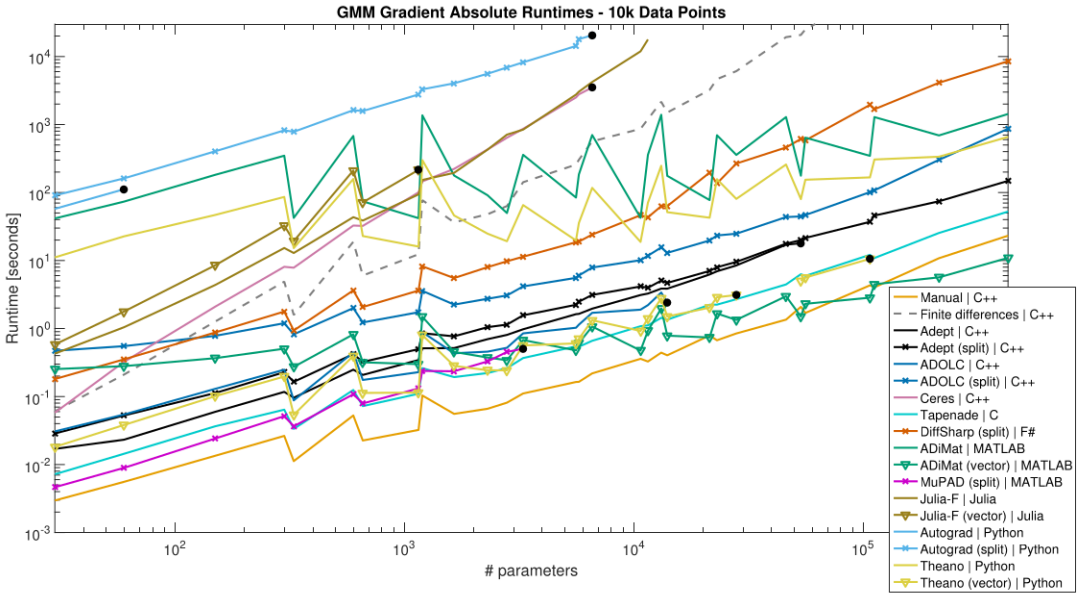


Figure 3.: Absolute runtimes for GMM with 10k data points. Some of the tools were run with (split) or (vector) implementations (see Sec. 5). The curve endings emphasized by the black dots symbolize that the tools crashed on bigger instances and those not emphasized did not finish in our time limit. Note that both axes are log-scaled. Best viewed in color.

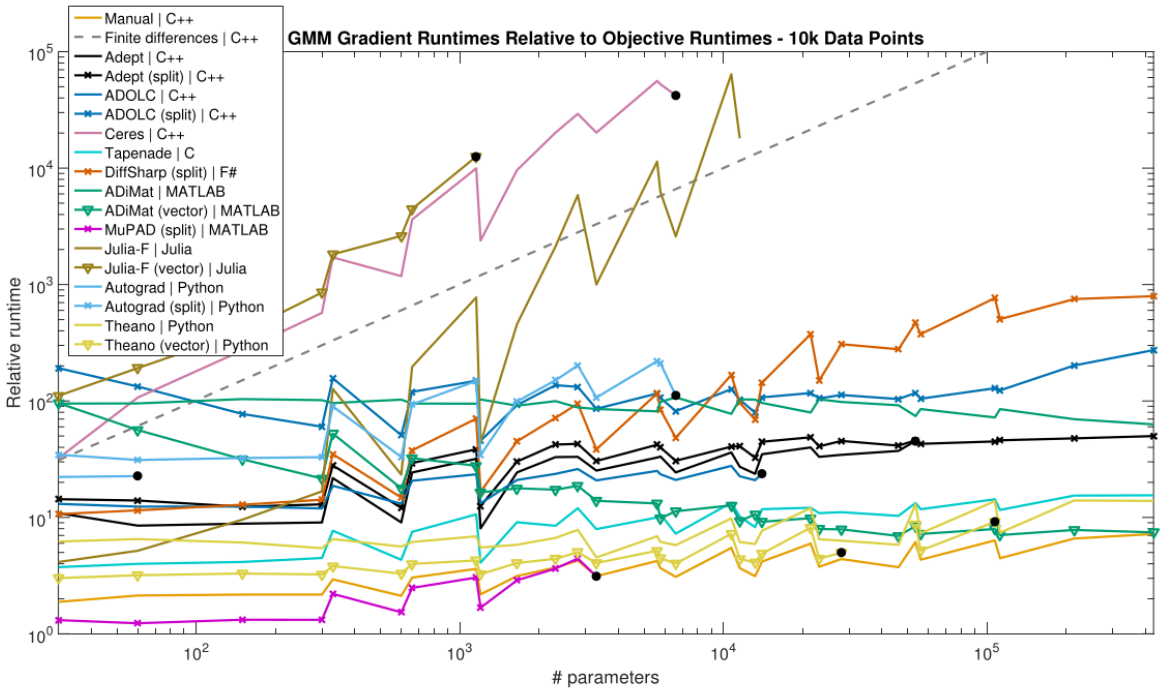


Figure 4.: Relative runtimes for GMM with 10k data points. Some of the tools were run with (split) or (vector) implementations (see Sec. 5). The curve endings emphasized by the black dots symbolize that the tools crashed on bigger instances and those not emphasized did not finish in our time limit. Note that both axes are log-scaled. Best viewed in color.

Adept does well. In source transformation Tapende is good. However, it can only work on pure C code. Theono from Python and MuPAD from MATLAB do very very well.

- Results from Bundle Assignment (BA) - A computer graphics problem

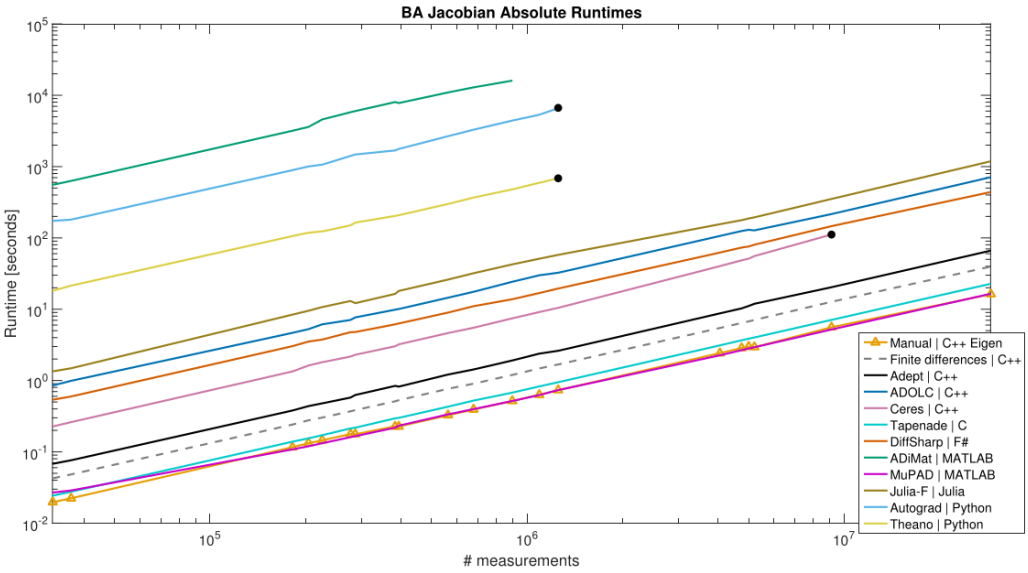


Figure 6.: Absolute runtimes for BA. Note that Eigen matrix library [12] was utilized for implementing hand-derived derivatives. The curve endings emphasized by the black dots symbolize that the tools crashed on bigger instances and those not emphasized did not finish in our time limit. Note that both axes are log-scaled. Best viewed in color.

One thing that suprises me ALOT is that Finite Difference is faster here. Need to understand the problem better here probably.

- Results from Hand Tracking (HT) - A computer graphics problem

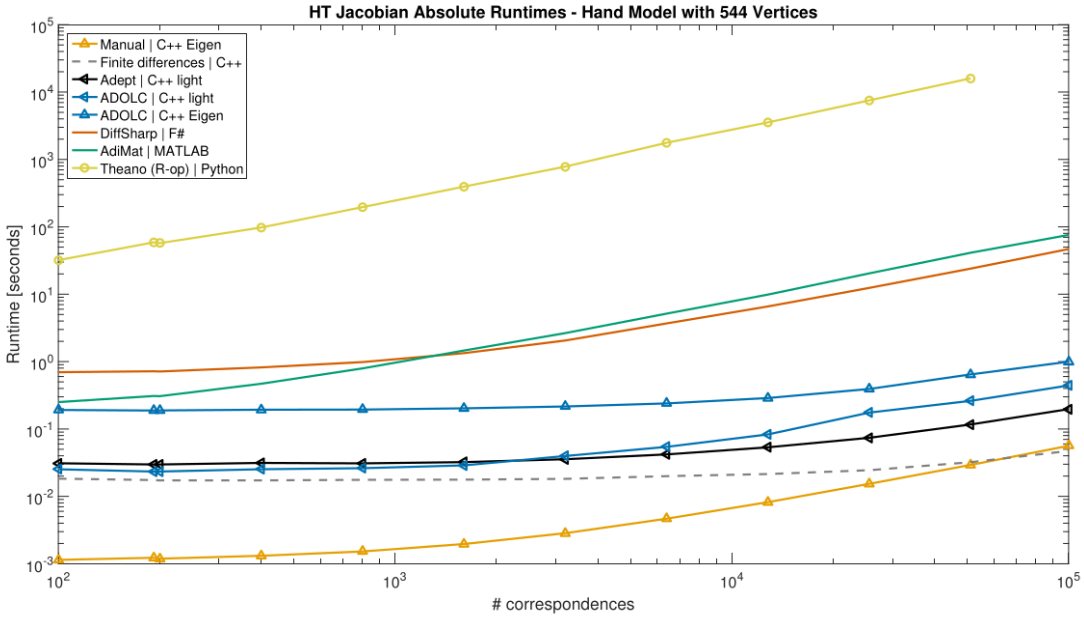


Figure 8.: Absolute runtimes for HT with the smaller hand model. Only some tools were benchmarked (see Sec. 5). Theano did not finish in our time limit for the largest number of correspondences. Note that both axes are log-scaled. Best viewed in color.

Again, Finite differences looks really fast for some reason.

10)

Kinematic state estimation for a Mars rover

J. (Bob) Balaram – JPL

Year - 2000

Journal – Robotica – 23 citations as of May 22

One-Liner Summary - Introduces a process kinematic model for a rover and then uses a EKF to do state estimation based on a gyro and a sun-sensor

Interesting things:

- The kinematic model is super interesting. Theey also have a cool kinematic slip equation that is implemented as a constraint because it is treated as a measurement in the filter.
- The EKF stuff is basically what we do – they did it in 2000.
- Results are decent -> similar to ours.

11)

Online BayesSim for Combined Simulator Parameter Inference and Policy Improvement

Rafael Possas , Fabio Ramos - Nvidia

Year – 2020

Conference – IROS – 11 citations as of May 22

One-Liner Summary – Integration of simulator parameter inference with both RL and MPC

Summary:

- The simulator parameter inference is Bayesian where the estimate the posterior as a Gaussian Mixture model where the posterior is proportional to a conditional density that is a mixture of K Gaussians. The posterior distribution refinement is iterative and is done online.
- They also propose using a RNN (Recurrent Neural Network) to automate the computation of a low-dimensional representation of state-action trajectories to get rid of the scaling problem as the state-action pair increases in dimension. This latent output from the RNN is what is used to estimate the posterior. This removes the need to manually define meaningful summary statistics, which sometimes, can be a quite difficult and complex task.

Algorithm 1 Online BayesSim

//observed and real trajectories: $\mathbf{S}^s, \mathbf{A}^s$ *and* $\mathbf{S}^r, \mathbf{A}^r$
//RNN Mixture of Gaussians (MoG) estimator: $q_\phi(\boldsymbol{\theta}|\mathbf{z})$
//RL Policy: $\pi_\beta(\boldsymbol{a_t}|\boldsymbol{s_t})$
Inputs: *total_steps*, *policy_train_steps*, *mog_train_steps*, *num_sampled_params*, $p(\boldsymbol{\theta}_0)$
Outputs: $q_\phi(\boldsymbol{\theta}|\mathbf{z}), \pi_\beta(\boldsymbol{a_t}|\boldsymbol{s_t})$

Initialize weights $\boldsymbol{\beta}$ and $\boldsymbol{\phi}$ randomly

$t \leftarrow 1$
repeat
    $\boldsymbol{\theta}_t \sim p(\boldsymbol{\theta}_{t-1})$
    $\mathbf{S}^s, \mathbf{A}^s \leftarrow$ Run $\pi_{\beta_t}(\boldsymbol{a_t}|\boldsymbol{s_t})$ in sim with $\boldsymbol{\theta}_t$.
    $\boldsymbol{\beta}_t \leftarrow \boldsymbol{\beta}_{t-1} + \lambda \nabla \pi_{\beta_t}(\boldsymbol{a_t}|\boldsymbol{s_t})$
    $\phi_t \leftarrow \phi_{t-1} + \lambda \nabla q_\phi(\boldsymbol{\theta}_t|\psi_{\gamma_t}(\mathbf{S}^s, \mathbf{A}^s))$
    $\mathbf{S}^r, \mathbf{A}^r \leftarrow$ Run $\pi_{\beta_t}(\boldsymbol{a_t}|\boldsymbol{s_t})$ on real env.
    $p(\boldsymbol{\theta}_t|\mathbf{S}, \mathbf{A}) \leftarrow q_\phi(\boldsymbol{\theta}_t|\psi_{\gamma_t}(\mathbf{S}^r, \mathbf{A}^r))$
    $p(\boldsymbol{\theta}_t) \leftarrow p(\boldsymbol{\theta}_t|\mathbf{S}, \mathbf{A})$
    $t \leftarrow t + 1$
until $t <$ *total_steps* or convergence reached

- A lot of mentions about why using models with parameters as a distribution is so useful.

12)

Kinematic Modeling and Analysis of Skid-Steered Mobile Robots with Applications to Low-Cost Inertial-Measurement-Unit-Based Motion Estimation

Jingang Yi – Rutgers University

Year – 2009

Journal – IEEE Transactions on Robotics

Citations – 147 as of May 23

One-Liner Summary – Kinematic model of a skid steered mobile robot and how it can be used with an IMU and wheel encoder to do state estimation.

Summary --

- The kinematic model is simple, but it includes slip. This slip mechanism largely depends on correctly estimating the instantaneous center of rotation (ICR) of the left and right side wheels. This requires a parameter S which they evaluate emperically through experimentation.
- Then , the IMU motion is defined by kinematic equations -> The accelerations are provided by the IMU measurements.

acceleration and angular rate measurements in $\mathcal{B}$ as $\mathbf{A}_B = [a_{Bx} \;\; a_{By} \;\; a_{Bz}]^T \in \mathbb{R}^3$ and $\boldsymbol{\omega}_B = [\omega_{Bx} \;\; \omega_{By} \;\; \omega_{Bz}]^T \in \mathbb{R}^3$, respectively. We obtain the following kinematic motion equations for the IMU [21], [22]:

$$\dot{\mathbf{P}}_I = \mathbf{V}_I \tag{10a}$$

$$\dot{\mathbf{V}}_I = C_B^I \mathbf{A}_B + \mathbf{G} \tag{10b}$$

$$\dot{\phi} = \omega_{Bx} + s_\phi \tan\theta \omega_{By} + c_\phi \tan\theta \omega_{Bz} \tag{10c}$$

$$\dot{\theta} = c_\phi \omega_{By} - s_\phi \omega_{Bz} \tag{10d}$$

$$\dot{\psi} = \frac{s_\phi}{c_\theta}\omega_{By} + \frac{c_\phi}{c_\theta}\omega_{Bz} \tag{10e}$$

- The “measurements” are actually “virtual velocity” measurements that are obtained from the wheel encoder, a noise model for the ground topology and the yaw rate from the IMU.
- The estimation of the parameter “S” is very elaborate. This can be directly replaced by Bayesian methods.
- EKF is used. The results look very impressive!

13)

An Accurate Simple Model for Vehicle Handling using Reduced-order Model Techniques

Chul Kim – General Motors

SAE Technical Paper series

Year – 2001

Citations – 4 as of May 23

One-Liner Summary – Reduced Order Model obtained from an ADAMs model using singular perturbation method for vehicle handling simulation.

Summary

- Results look good but for a very specific case of vehicle handling
- They then design a 4 wheel steering system using the Reduced Order Model and show that their model is good

14)

Toward predictive digital twins via component-based reduced-order models and interpretable machine learning

Michael G. Kapteyn – MIT

Karen E. Willcox – UT Austin

Year - 2020

Journal – American Institute of Aeronautics and Astronautics

Citations – 75 as of May 23 2023

One-Liner Summary – They build a digital twin of a UAV by first building a library of reduced order models for each of the components of the UAV and then using optimal classification trees to pick model (which is a combination of the components) that best represents the observed data.

Summary -

- The paper introduces a methodology for building digital twins of complex systems. There are 2 players in this methodology
    - The library of components that make up the big model. Each of these components are FEA models but with reduced order.
    - A Optimal Classification Tree (OCT) which acts as a model selector to pick the appropriate model based on the observed data.
- Player 1 – The component models
    - These are reduced order models built using ***Static-Condensation Reduced-Basis-Element.*** Not much detail provided about the method, but has a lot to do with FEA and stuff so not paying much attention.
    - Each of these ROMs have a bunch of parameters. This is what makes up an ***archetype component***.
    - A combination of archetype components, connected via ***ports,*** make up a full scale model.
    - A library of full scale models are constructed using different parameters for each of these archetype components.
- Player 2 – Model selector OCT
    - The model selector is trained on sensor data and the model itself – this is weird – they actually built an UAV but still used simulation to train the model selector.
    - How they train is – they use one model, say Mj. This Mj is simulated for a given test. They use priveldged information to gather say the strain across mutiple components of Mj. They then add noise to this previledged information to produce “sensor readings”.

Then, they learn a mapping from this “sensor readings” to a parameter set, say mu_j that define the component.

- This mapping set is learnt by an OCT -
  - Because it is interpretable
  - And simple

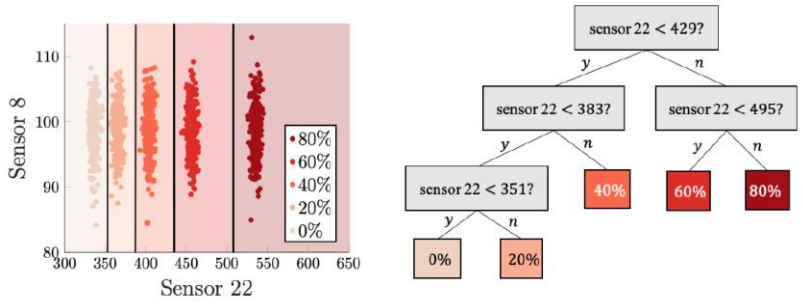


Fig. 7 An OCT for computing the damage parameter μ_1. Left: Partitioning of the feature space (the space of strain measurements) using axis-aligned splits. Right: The decision tree for classifying the value of μ_1.

- In conclusion I feel that the idea is great but it is shown on a very simple examle, in simulation where it is expected to work.
- I also question how different is this from a parameter estimation problem? In a probabilistic parameter estimation problem, when we provide a prior for the parameters and then sample for it, aren’t we sampling from a infinite “library of models”?

15)

Data-driven physics-based digital twins via a library of component-based reduced-order models

Kapteyn – MIT

Willcox – UT Austin

Journal – Journal for numerical methods in Engineering

Year – 2020

Citations – 39 as of May 23 2023

One-Liner Summary – Same as above paper but instead of using OCT as a model selector, they use a Bayesian approach to assign beliefs to each model and then take an expectation to get the estimate.

Summary -

- They now have a distribution over the models – each model is assigned a belief given by the bayes rule. This belief depends on a transition probability (first term) that is defined using domain knowledge and a likelihood of that model explaining the observed data.
- One thing that is unclear here is their likelihood function definition – probably gaussian

$$b_t(M_j) \propto \left[\sum_{k=1}^{|\mathcal{M}|} p(d_t = M_j | d_{t-1} = M_k, u_t) b_{t-1}(M_k) \right] p(o_t | d_t = M_j, u_t).$$

- They also maintain a "error in library of digital twin estimates" which is given by

$$o_t = Z(M_j, u_t) + v_t, \quad v_t \sim N(0, \sigma^2),$$

$$\epsilon_t(b_t, o_t; \mathcal{M}) = |\mathbb{E}_{b_t}[Z(M_j, u_t)] - o_t|.$$

- This time, for testing, they use the actual UAV and collect strain sensor data from it.

16)

A Differentiable Newton-Euler Algorithm for Real-World Robotics

Michael Lutter – TU Darmstadt

Year – Oct 2021

Journal – Under-review at IEEE robotics and automation

One-Liner Summary: Show that differentiable models help in system identification and solve the under-fitting problem of white box models.

Summary

- Not much going on really. They show that the Newton Euler eqution is differentiable.
- They augment this with different “actuator” models – some data driven and some white box and show that since the model itself is differentiable, the parameters for these models can also be learnt.

17)

Differentiable Simulation for Physical System Identification

Quentin Le Lidec - Inria, Ecole normale suprieure, CNRS, PSL

Year – April 2021

Journal – IEEE Robotics and Automation

Citations – 6 as of May 25

One-Liner Summary – They use a an extension of the staggered projections algorithm to model friction and contact and differentiate this analytically and use it to do parameter identification.

Summary -

- The contact and friction is modelled using 1 Quadratic Programming optimization problem and 1 Quadratic programming with Quadratic constraints optimization problem. These 2 problems are solved alternatingly.
- They also differentiable through the optimization procedure analytically, making use of the Karush-Kuhn-Tucker optimality conditions. The derivative then involves solving a linear system.
- This shows that even if the model contains an optimization loop, we can differentiate it and make the model differentiable.

18)

Vehicle Dynamic State Estimation: State of the Art Schemes and Perspectives

Hongyan Guo – Chinese University

Year – 2018

Journal – IEEE/CAA JOURNAL OF AUTOMATICA

Citations – 64 as of May 31 2023

One-Liner Summary – A good survey of sensors, Vehicle Dynamics models and state estimation methods used to perform state estimation for autonomous vehicles.

Summary -

- Sensor configurations were not very relevant to us. However, GPS seems to rarely be used.
- Vehicle models were very well explained. However, there are no vehicle models used in state estimation that have good tires, a powertrain and stuff. The maximum they go is to 14 DOF that is with suspension.
- State estimation majorly discussed is the EKF and then "observer-based models". Need to read up a bit more on these observer-based models as they say that these are more efficient and better. **NO MENTION OF PARTICLE FILTERS**
- Another thing discussed is the issue of vehicle dynamic state estimation with unknown inputs. The unknown inputs here mean the tire road interaction (road banking, road slope) etc. A few papers cited that I will read but this is stated as a major challenge in state estimation.

19)

Advanced Estimation Techniques for Vehicle System Dynamic State: A Survey

Xianjian Jin - Shanghai University, Shanghai 200072, China

Year – Oct 2019

Journal – Sensors

Citations – 31 as of May 31 2023

One-Liner Summary – A better survey of sensors, Vehicle Dynamics models and state estimation methods used to perform state estimation for autonomous vehicles.

Summary -

- This is better than the above paper, much more extensive
- Vehicle models with tire models discussed – No one is using TMEasy tires
- Dual Extended Kalman Filter which does both parmameter and state estimation discussed.
- 3 papers with particle filters for state estimation is discussed.
- Data driven state estimators also discussed.

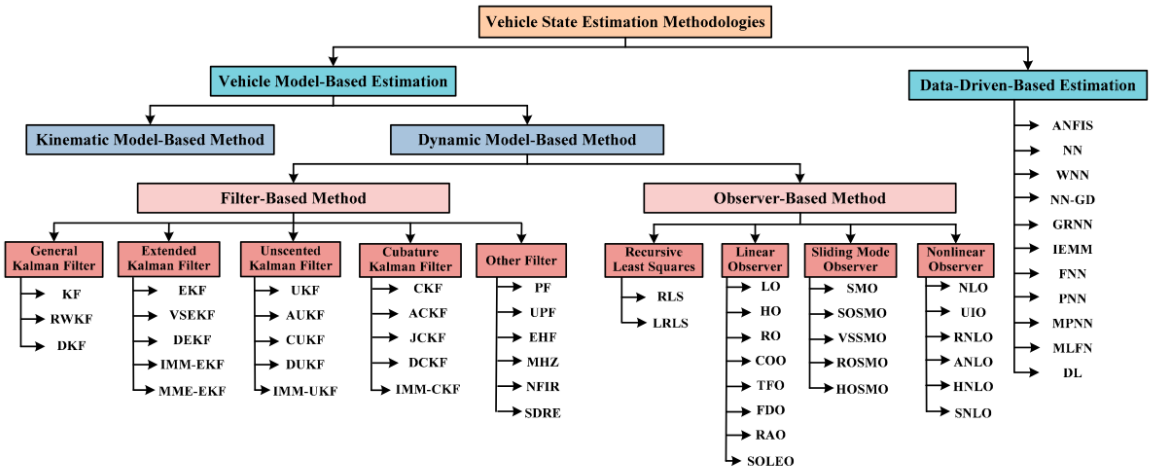


Figure 5. Categorization of vehicle dynamic state estimation methodologies.

Table 1. Methodologies, models, estimations and sensor configurations for filter-based vehicle state estimation.

Methodologies	Models	Estimations	Sensor Configurations	References
KF	Single-track + Linear tire model	V_y	r_z, δ_f	[21]
KF + RLS	Single-track + Roll + Linear tire model	β, φ	r_z, δ_f, a_x, a_y, ω_{ij}, T_{ij}, F_{yij}, MSU	[22]
KF	Single-track + Linear tire model	β	r_z, ψ, GPS	[23–25]
DKF	Single-track + Linear tire model	C_α	V_x, V_y, ψ, GPS	[26]
RWKF	Double-track + Linear tire model	F_x, F_y	r_z, a_x, a_y, ω_{ij}	[27]
EKF	Single-track + Pacejka tire model	β	r_z, a_y	[28]
EKF	Single-track + Linear tire model	β, C_α	r_z, F_{yij}, MSU	[29]
EKF	Single-track + Pacejka tire model	β	$T_{\delta f}$, r_z, a_x, a_y	[30]
EKF + SMC	Single-track+Roll+Dugoff tire model	β, φ	V_x, r_z, a_x, a_y	[31]
EKF	Single-track + Pacejka tire model	β	r_z, a_y	[32]
EKF	Double-track + Roll + Pacejka tire model	V_x, V_y, φ, F_y, DBE	r_z, a_x, a_y, p	[33,34]
EKF	Double-track + Dugoff tire model	β, F_y	r_z, δ_f, a_x, a_y, ω_{ij}	[35]
VSEKF	Double-track + Dugoff tire model	β	r_z, δ_f, a_y	[36]
EKF	Double-track + Roll + Dugoff tire model	β, F_y, F_z	r_z, a_x, a_y, p	[37]
EKF	Double-track + Pacejka tire model	V_x, V_y	r_z, δ_f, a_x, a_y, ω_{ij}	[38]
EKF + MME	Double-track + Pacejka tire model	V_x, V_y, F_z, F_y	r_z, δ_f, a_y	[39]
EKF	Single-track + Burchhardt tire model	β, F_x, F_y, μ	r_z, δ_f, a_x, a_y, ω_{ij}	[40]
EKF	Single-track + Pacejka tire model	β	r_z, a_y	[41]
EKF	Longitudinal model + Pacejka tire model	μ	a_y, ω_{ij}	[42,43]
DEKF	Double-track + Roll + Pacejka tire model	μ, F_x, F_y, F_z	r_z, a_x, a_y, p	[44]
EKF	Single-track + Roll + Linear tire model	β, φ	r_z, δ_f, a_x, a_y, ψ, GPS	[45,46]
EKF	Single-track + Brush tire model	β, μ	V_x, V_y, ψ, r_z, a_x, a_y, p, GPS	[47]
EKF	Single-track + Pacejka tire model	V_x, β, θ, C_α,μ	r_z, δ_f, a_x, a_y, ω_{ij}	[48]
DEKF	Double-track + Pacejka tire model	β, m, I_{zz}	r_z, a_y, V_x	[49]
DEKF	Single-track + Roll + Linear tire model	β, φ, C_α, I_{zz}	r_z, δ_f, a_x, a_y, p	[50]
DEKF	Double-track +Dugoff tire model	β, μ	r_z, a_x, a_y	[51]
IMM-EKF	Single-track + Other nolinear tire model	β, μ	r_z, a_y	[52]
IMM-UKF	Double-track + Roll + Dugoff tire model	β,φ	r_z, a_x, a_y, p, ω_{ij}	[53]
IMM-EKF	Single-track + Other nolinear tire model	β, F_x, F_y	r_z, δ_f, V_x, a_y	[54]
UKF	Double - track+ UniTire tire model	V_x, V_y	r_z, a_x, a_y, ω_{ij}	[55]
UKF	Single-track + Linear tire model	β	a_x, a_y	[56]
AUKF	Double-track + Pacejka tire model	β	a_y, r_z	[57,58]
CUKF	Single-track + Random Walk model	β, F_y	r_z, a_y	[59]
UKF/EKF	Double-track + Dugoff tire model	β, F_y	r_z, δ_f, a_x, a_y, ω_{ij}	[60]
UKF	Double-track + Dugoff tire model	μ	a_x, a_y	[61,62]
UKF	Double-track + Pacejka tire model	V_x, V_y,μ	r_z, a_y, ω_{ij}	[63]
DUKF	Double-track + Dugoff tire model	β, m	r_z, a_y, V_y	[64]
DUKF	Double-track + Roll + Pacejka tire model	β, φ, F_y, F_z, DBE	r_z, a_x, a_y, p, ω_{ij}	[65]
CKF	Single-track + Linear tire model	β	δ_f, a_y	[66]
CKF	Double-track + Roll + Dugoff tire model	β,φ, F_x, F_y	r_z, a_x, a_y, p, ω_{ij}	[67]
ACKF	Double-track + Pacejka tire model	V_x, V_y	r_z, δ_f, a_x, a_y, ω_{ij}	[68,69]
JCKF, DCKF	Double-track + Pacejka tire model	V_x, V_y,μ	r_z, a_x, a_y	[70]
IMM+CKF	Double-track + Pacejka tire model	V_x, V_y	r_z, a_x, a_y	[71]
PF	Double-track + Dugoff tire model	β, F_x, F_y	r_z, a_x, a_y	[72,73]
UPF	Double-track + Pacejka tire model	β, F_y	r_z, a_x, a_y	[74]
MHE	Single-track + Pacejka tire model	β	r_z	[75,76]
SDRE + EKF	Single-track + Random Walk model	β	r_z, a_y	[77]
EHF	Single-track + Linear tire model	β, C_α	r_z, ψ, GPS	[78]
MHE	Single-track + Pacejka tire model	β, P_y	r_z, P_z, GNSS	[79]

EFFECT OF VEHICLE MODEL ON THE ESTIMATION OF LATERAL VEHICLE DYNAMICS

J. Kim - Department of Vehicle Dynamics Research Team, Hankook Tire Co., LTD., R&D Center, Korea

Year – Oct 2010

Journal – International Journal of Automotive Technology

Citations –26 as of May 31 2023

One-Liner Summary – Estimate tire parameters using an EKF first, then use the estimated parameters with combinations of a 2 DOF, 3 DOF, 4 DOF vehicle model with either Linear and Nonlinear tires modelling Relaxation Length and not modeling relaxation length.

Summary -

- Interesting way of estimating tire parameters using EKF.
- Conclusions are that Relaxation length is very important for sharp turns. Nonlinear tires are also very important for sharp turns.
- Vehicle model does effect state estimation by quite a bit for extreme maneuvers.
- IMU and GPS is used for the data

21)

Real-Time Estimation of the Road Bank and Grade Angles with Unknown Input Observers

E. Hashemi - University of Waterloo

Year – Jan 2017

Journal – Taylor & Francis in Vehicle System Dynamics

Citations – 40 as of June 1 2023

One-Liner Summary – Estimating Road bank and grade angles using an Unknown Input observer and a kinematic model which relates the suspension height measurements to the roll and pitch angle.

Summary -

- Did not get a whole lot from it because of my lack of observer knowledge.
- Need to have the suspension measurements to really use this
- Highlighted a few references for further reading if we are interested in estimating this road bank and road grade angle.

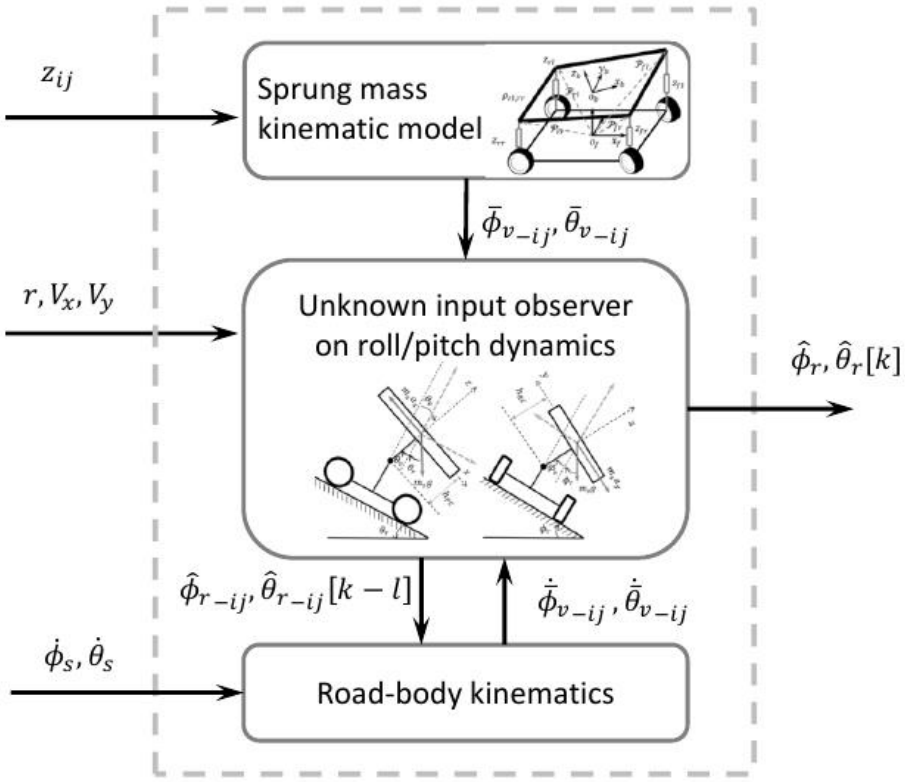


Fig. 1: The proposed structure for the road angle estimation

22)

Dual extended Kalman filter for vehicle state and parameter estimation

T. A. WENZEL - Jaguar and Land Rover Research, Whitley, UK

Year – Feb 2006

Journal – Taylor & Francis in Vehicle System Dynamics

Citations – 278 as of June 2 2023

One-Liner Summary – Estimating the vehicle states and 3 parameters using simple IMU data from a real vehicle and a simulation (yaw rate, IMU lateral acceleration and longitudinal velocity)

Summary:

- Mass, yaw inertia and wheel base estimated along with the states in an approach that runs two EKFs in parallel, one for parmaeter estimation and one for state estimation.
- Parameter estimation uses exactly the same structure. The only innovation is in the way the parmaeter process noise is given. See page 9 for this.
- They also have the double track vehicle model with the TM easy tire and the Magic tire and they compare both of them. Both perform decently well.
- The inputs to the DEKF and hence the model is the steering angle and the wheel angular velocities which are also measured. Thus the estimation of parmaeter and states is done offline.
- Parameter estimation results are horrible however the estimated state is great!

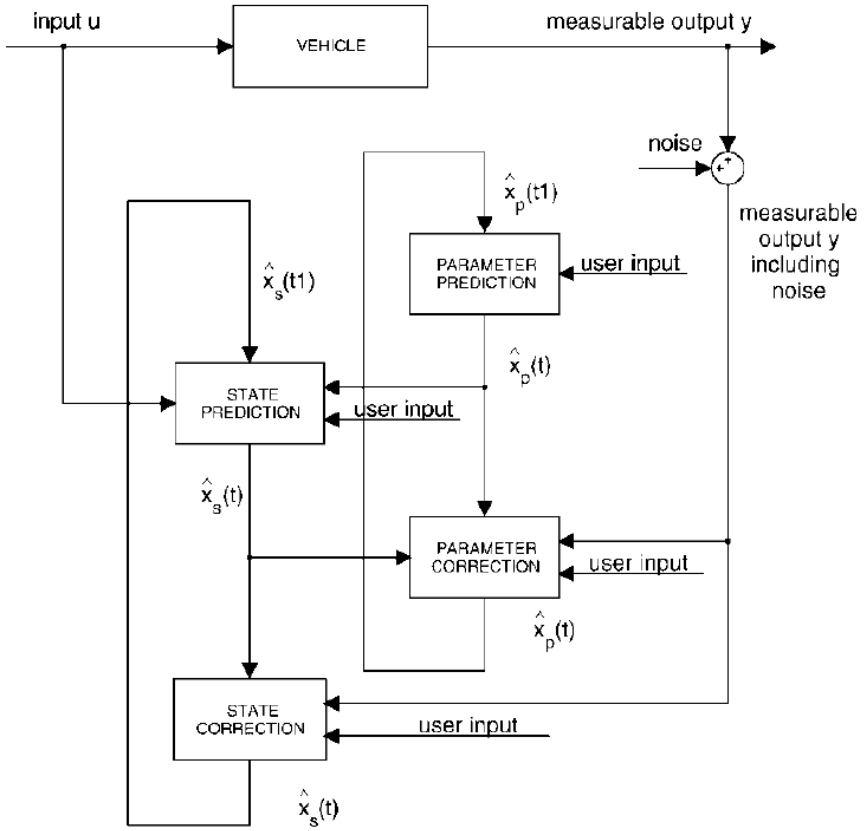


Figure 2. Scheme of DEKF.