KOBE

Kinetic Optimizer for Ballistic Engagements

Gregory Taylor
gregory.taylor.au@gmail.com

This report is a brief exploration of KOBE and its capabilities.

KOBE (Kinetic Optimizer for Ballistic Engagements), is a rust based library that calculates the required launch angle to intercept a moving target. To enable real-time usage, KOBE was developed to be as performant as physically possible, and can compute these outputs once every 70µs on most high end hardware. Currently, KOBE is at a ‘minimum capabilities stage’, where additional factors such as slewing speed or changing winds can be added in future developments to improve KOBE’s functionality.

Feature Set

Computing ballistic trajectories effected by:

- Bullet shape (G1/G7 drag)
- Spin drift
- Coriolis effects
- Environmental factors (air density and wind)

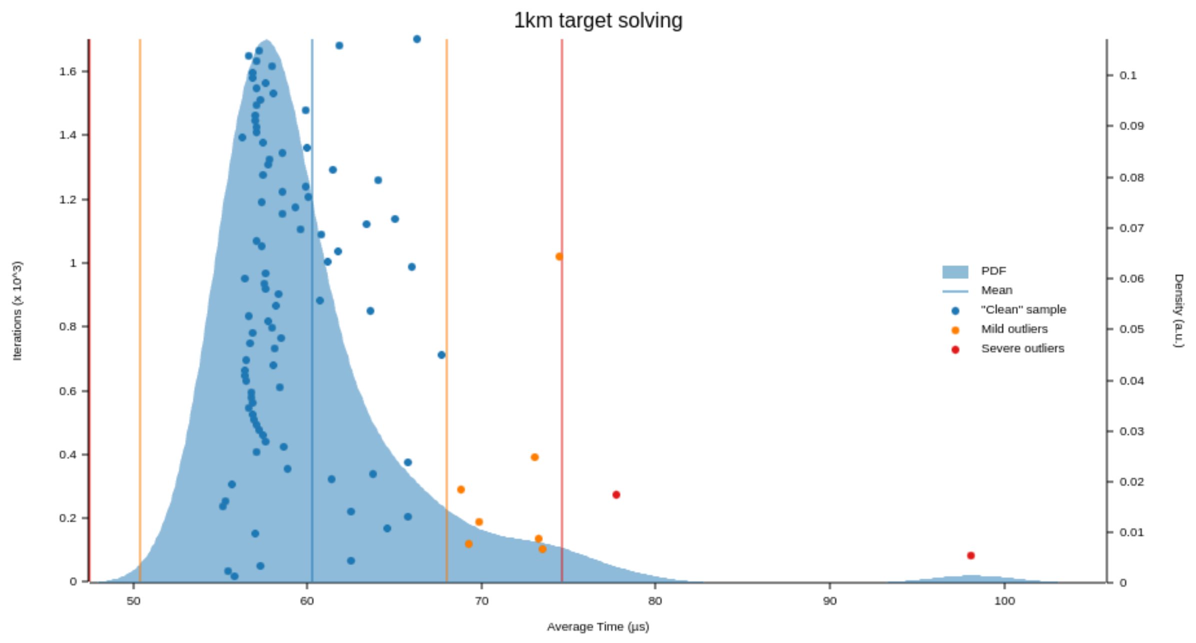
Identifying interception points for a given moving target

- Accommodating for flight time
- Maximising performance through gradient descent
- Guaranteed partial solutions, with most full solutions identified in < 4 iterations

Analysing impact characteristics

- Running 50,000+ flight paths on the GPU in parallel
- Simulating deviations in muzzle velocity, wind, and barrel condition
- Identifying probabilities and spread of trajectories

Targeting Performance



This chart shows the benchmarking performance in identifying and analysing a trajectory to a stationary target 1km away. During the average 70µs, approximately 20-40% of processing power is spent on identifying an ideal path that the projectile can take through gradient descent. The remaining time is spent on the GPU to simulate trajectory deviations and impact characteristics.

While being able to generate a result as quickly as possible is a major benefit of KOBE, it also allows for more head room for other processes to occur, such as drone vision detection or coordination with other platforms. It can also be purposefully rate limited to conserve power in low-energy environments or minimize cpu interrupts during mission critical events.

Another unique quirk of KOBE is the ability to generate partial solutions where an exact hit is not guaranteed or when computation is limited. This occurs when the projectile's path becomes difficult to predict within strong winds / high drag environments. Partial solutions are intended for 'worst case' scenarios where it is the responsibility of the operator or developer to handle such circumstances.

Trajectory Performance, Accuracy, & Comparisons to Other Calculators

Here, we are comparing KOBE against existing tools/libraries to model the trajectory of a .308 Win round travelling for 1km with a maximum fixed time step of 0.0001 seconds (average of 10 rounds). Ideal drop is calculated using the AB Quantum output as the baseline measurement to minimise bias as it is the most popular option shown.

Name	AB Quantum (mobile app)	py-ballisticcalc	Ballistics-engine (wasm-cpu)	libballistics	KOBE	KOBE (*fast)
Time	~1200ms	2320ms	63ms	26ms	0.046ms	0.00038ms
Drop	11.13m	17.07m	6.91m	11.14m	11.19m	11.87m
Error	N/A (reference)	+53%	-38%	+0.08%	+0.5%	+5.4%

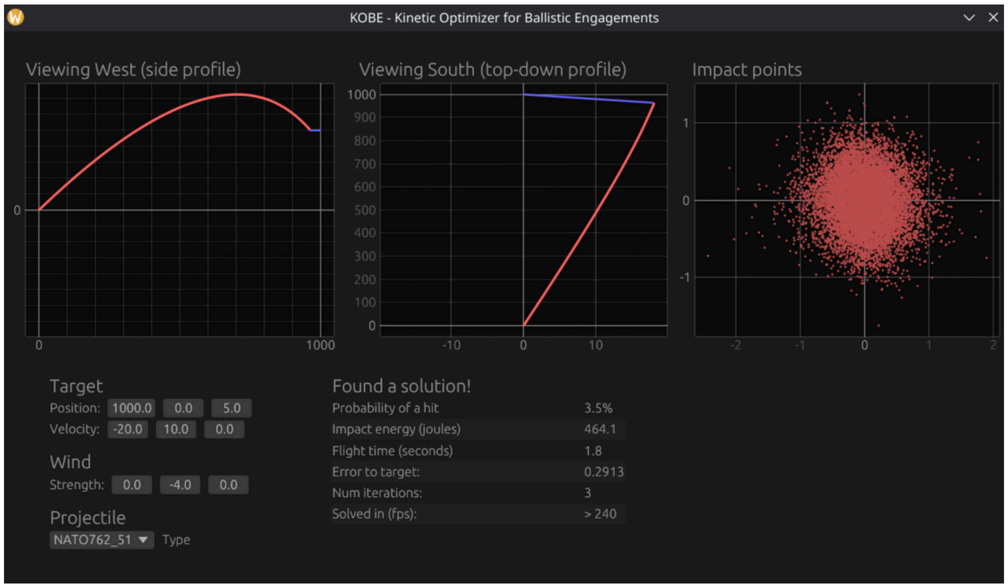
* The ‘fast’ setting for KOBE allows for additional acceleration in the physics engine via a predictive modeling algorithm at the cost of some accuracy. This is a more advanced technique that falls outside the scope of this report, but results in a 120x speed increase from the normal configuration and a 68,000x speed increase when compared to libballistics. Both KOBE and KOBE fast have a minimum overhead of 26ns.

In addition to the table provided, when compared against references in the Lapua Catalog, KOBE was able to maintain <1.0 MOA of drop and wind predictions within a 1km range, with a standard deviation in velocity of ~30m/s.

External

To practically explore the capabilities of KOBE, we supply two methods that can be utilised for your convenience. The files are hosted on Dropbox.

GUI Application - Executable: [Linux, Windows]


KOBE - Kinetic Optimizer for Ballistic Engagements
Viewing West (side profile)
Viewing South (top-down profile)
Impact points
0
0
1000
1000
900
800
700
600
500
400
300
200
100
0
-10
0
10
1
0
-1
-2
-1
0
1
2

Target
Position: 1000.0 0.0 5.0
Velocity: -20.0 10.0 0.0

Wind
Strength: 0.0 -4.0 0.0

Projectile
NATO762_51 Type

Found a solution!

Probability of a hit	3.5%
Impact energy (joules)	464.1
Flight time (seconds)	1.8
Error to target:	0.2913
Num iterations:	3
Solved in (fps):	> 240

Python Bindings - Wheels: [Linux, Windows] (Must use Python 3.14)

```python
from kobe_py import Vec3, Environment, Target, Physics, Projectile, Targeting
import time

physics = Physics(Projectile.WIN_308(), Environment(1013, 15, Vec3(0, 0, 0), 0))
targeting = Targeting(Target(lambda t: (100, 0, 0), 0.01), 10)

a = time.time()
zero, projectile = targeting.solve(physics)
print(f"solved in {(time.time() - a) * 1000:.2}ms") # 1.2ms (GIL bottleneck)

physics.init_projectile(zero.aiming_at)
while physics.projectile.position.x < 1000:
    physics.step()

print(physics.projectile.position.z) # drop = -11.1859
```