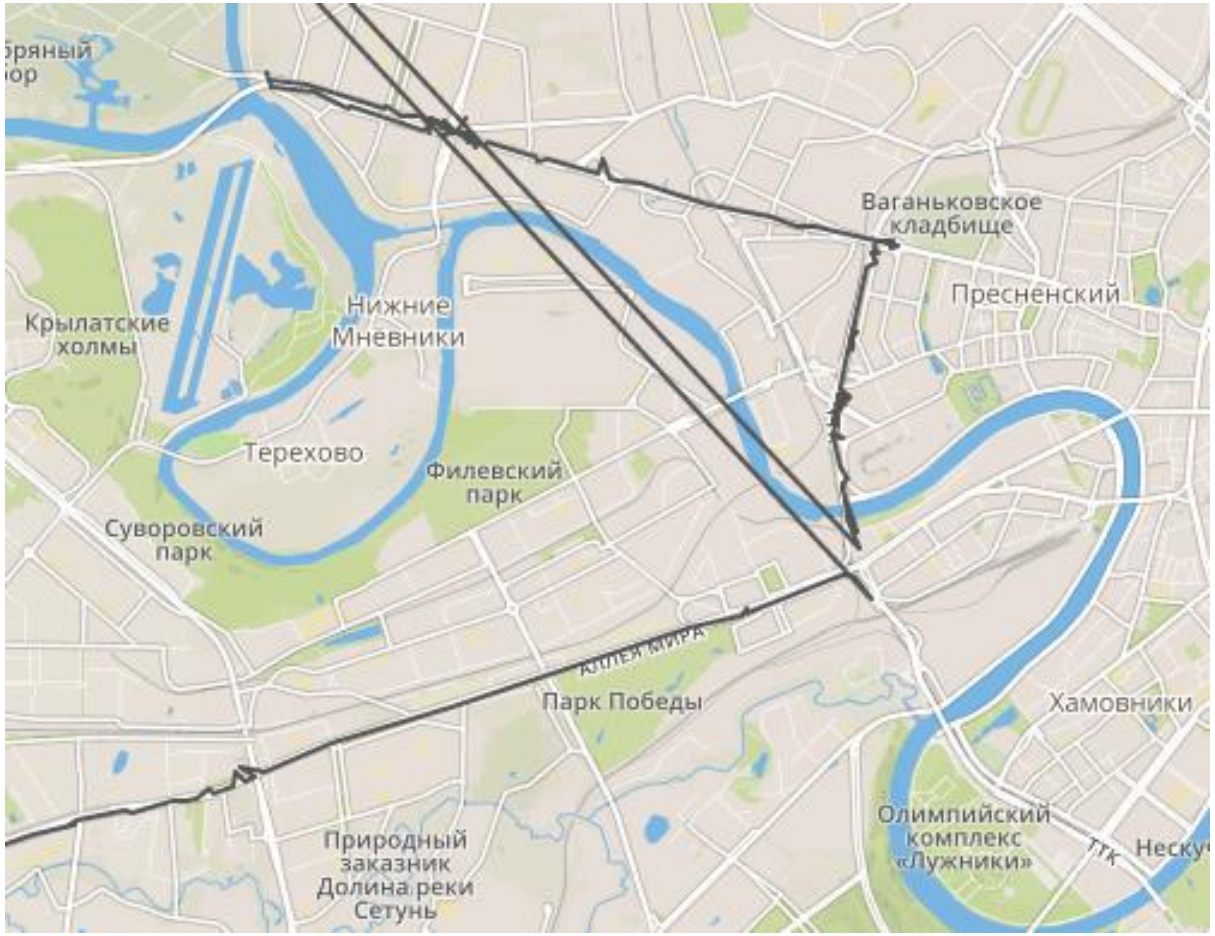
Сглаживание треков GPS на F#

Треки GPS, сферическая геометрия,
стабилизация и сглаживание,
фильтр Калмана, F#

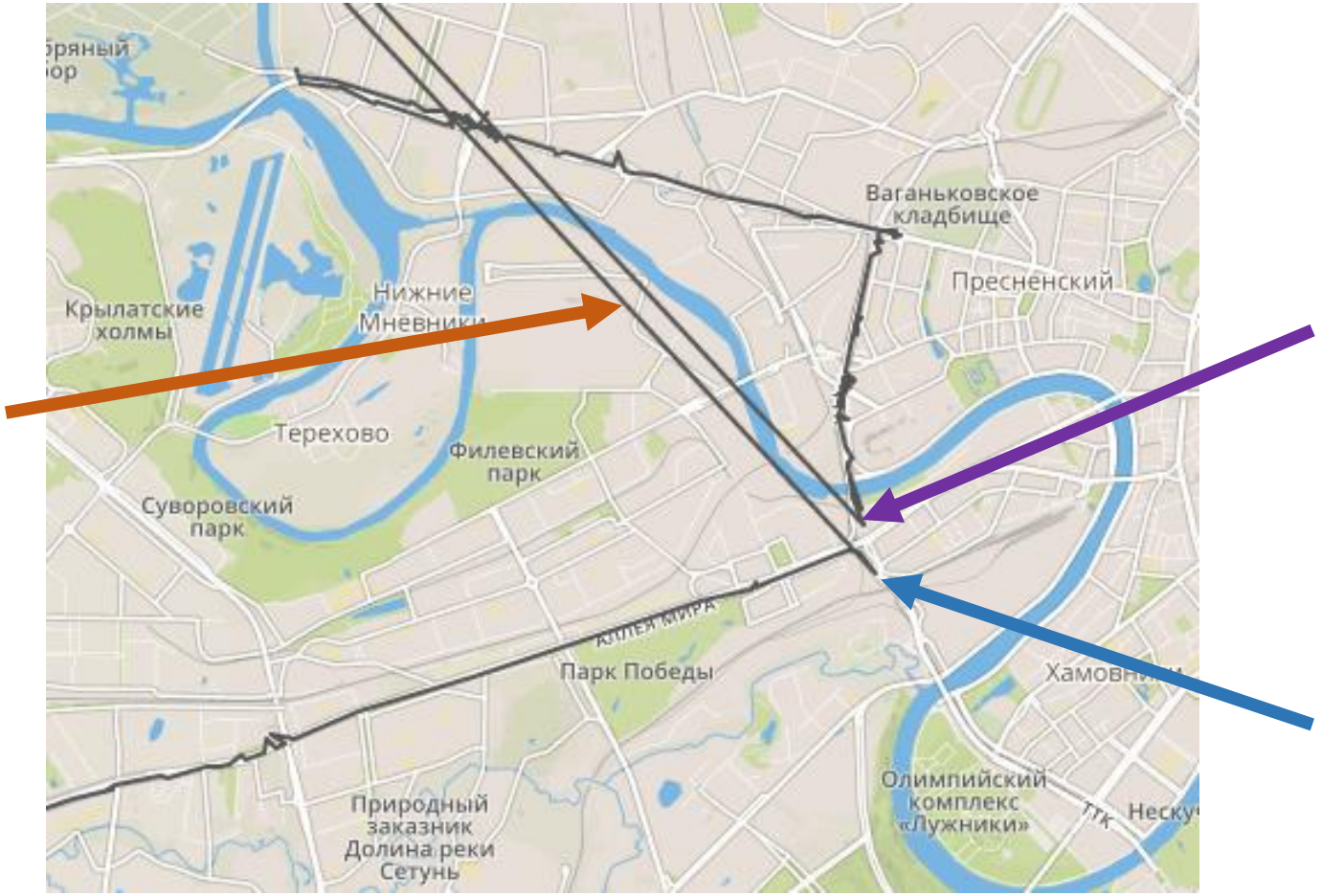
Московский клуб программистов
Марк Шевченко

Треки GPS



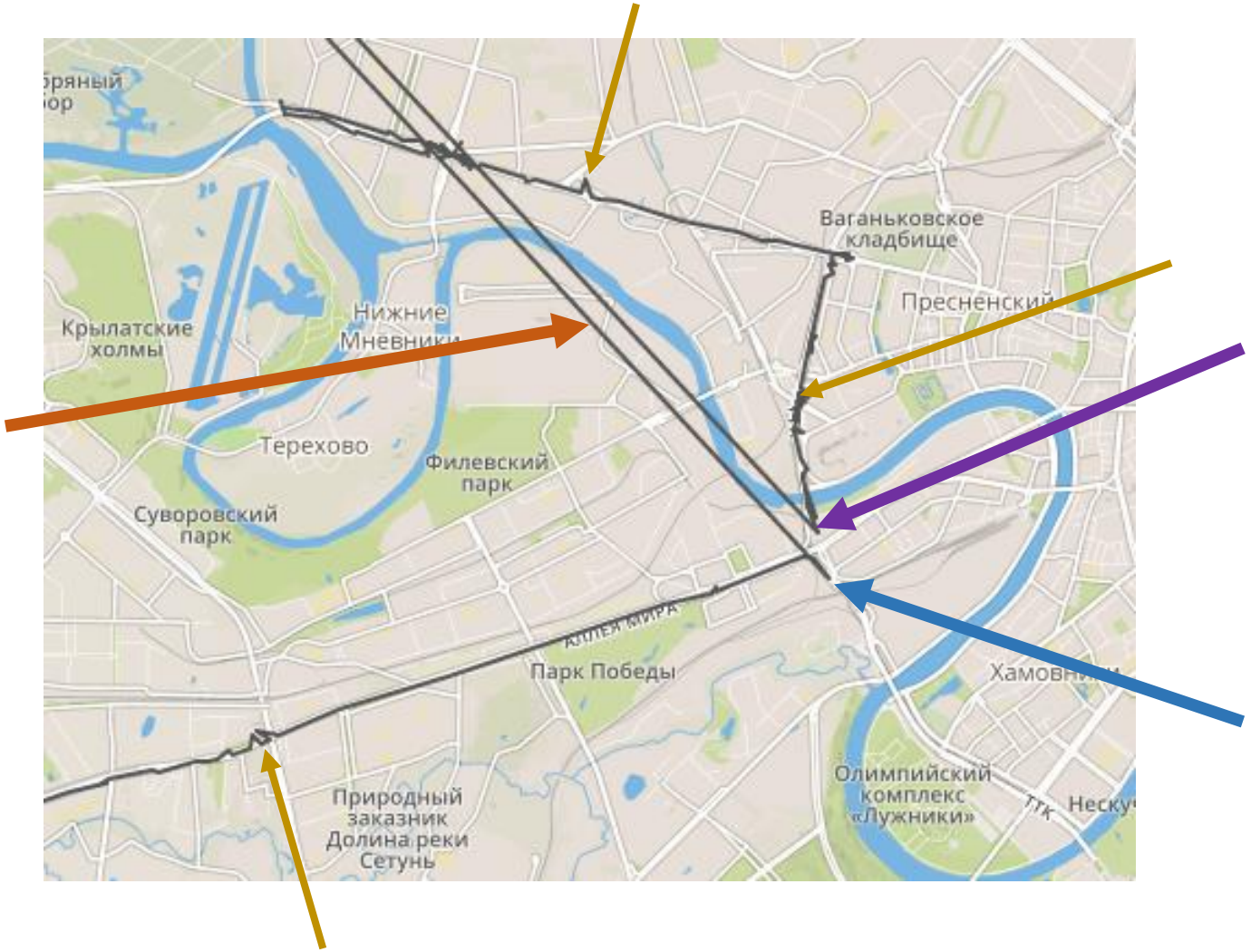
400 рублей

Треки GPS



~~400~~ 9000 рублей

Треки GPS



~~400~~ 9000 рублей

Данные GPS

```
type SensorItem(latitude: float, longitude: float, speed: float,
                heading: float, timestamp: DateTimeOffset) =
    member __.Latitude = latitude
    member __.Longitude = longitude
    member __.Timestamp = timestamp
    member __.Speed = speed
    member __.Heading = heading
```

Данные GPS

```
type SensorItem(latitude: float, longitude: float, speed: float,
                heading: float, timestamp: DateTimeOffset) =
    member __.Latitude = latitude
    member __.Longitude = longitude
    member __.Timestamp = timestamp
    member __.Speed = speed
    member __.Heading = heading

    override __.GetHashCode() =
        hash (latitude, longitude, timestamp, speed, heading)

    override __.Equals(other) =
        match other with
        | :? SensorItem as x -> (latitude, longitude, timestamp, speed, heading)
                              = (x.Latitude, x.Longitude, x.Timestamp, x.Speed, x.Heading)
        | _ -> false
```

Данные GPS

```
type SensorItem(latitude: float, longitude: float, speed: float,
                heading: float, timestamp: DateTimeOffset) =
    member __.Latitude = latitude
    member __.Longitude = longitude
    member __.Timestamp = timestamp
    member __.Speed = speed
    member __.Heading = heading

    override __.GetHashCode() =
        hash (latitude, longitude, timestamp, speed, heading)

    override __.Equals(other) =
        match other with
        | :? SensorItem as x -> (latitude, longitude, timestamp, speed, heading)
                              = (x.Latitude, x.Longitude, x.Timestamp, x.Speed, x.Heading)
        | _ -> false
```

Данные GPS

```
type SensorItem(latitude: float, longitude: float, speed: float,
                heading: float, timestamp: DateTimeOffset) =
    member __.Latitude = latitude
    member __.Longitude = longitude
    member __.Timestamp = timestamp
    member __.Speed = speed
    member __.Heading = heading

    override __.GetHashCode() =
        hash (latitude, longitude, timestamp, speed, heading)

    override __.Equals(other) =
        match other with
        | :? SensorItem as x -> (latitude, longitude, timestamp, speed, heading)
                              = (x.Latitude, x.Longitude, x.Timestamp, x.Speed, x.Heading)
        | _ -> false
```

Данные GPS

```
type SensorItem(latitude: float, longitude: float, speed: float,
                heading: float, timestamp: DateTimeOffset) =
    member __.Latitude = latitude
    member __.Longitude = longitude
    member __.Timestamp = timestamp
    member __.Speed = speed
    member __.Heading = heading

    override __.GetHashCode() =
        hash (latitude, longitude, timestamp, speed, heading)

    override __.Equals(other) =
        match other with
        | :? SensorItem as x -> (latitude, longitude, timestamp, speed, heading)
                              = (x.Latitude, x.Longitude, x.Timestamp, x.Speed, x.Heading)
        | _ -> false
```

Данные GPS

```
public class SensorItem
{
    public double Latitude { get; }

    public double Longitude { get; }

    public double Speed { get; }

    public double Heading { get; }

    public DateTimeOffset Timestamp { get; }

    public SensorItem(double latitude, double longitude, double speed, double heading, DateTimeOffset timestamp)
    {
        Latitude = latitude;
        Longitude = longitude;
        Speed = speed;
        Heading = heading;
        Timestamp = timestamp;
    }

    public override int GetHashCode()
    {
        return (Latitude, Longitude, Speed, Heading, Timestamp).GetHashCode();
    }

    public override bool Equals(object other)
    {
        if (other is SensorItem otherItem)
        {
            return Latitude == otherItem.Latitude
                && Longitude == otherItem.Longitude
                && Speed == otherItem.Speed
                && Heading == otherItem.Heading
                && Timestamp == otherItem.Timestamp;
        }

        return false;
    }
}
```

Что мы делаем с треками

1. Фильтр Калмана

Что мы делаем с треками

1. Сглаживание
    1. Фильтр Калмана

Что мы делаем с треками

1. Стабилизация
2. Сглаживание
    1. Фильтр Калмана
3. Прореживание

Что мы делаем с треками

1. Стабилизация
    1. Устранение нулевых и отрицательных интервалов
    2. Устранение всплесков скорости
    3. Устранение дребезга нулевой скорости
2. Сглаживание
    1. Фильтр Калмана
3. ~~Прореживание~~

Нулевые и отрицательные интервалы

55,75504290	37,58472160	09.06.2019 09:17:34
55,75491030	37,58458640	09.06.2019 09:17:39
55,75496650	37,58437700	09.06.2019 09:17:45
55,75546590	37,58444770	09.06.2019 09:17:51
55,75596390	37,58451480	09.06.2019 09:17:57
55,75639320	37,58449470	09.06.2019 09:18:03
55,75639320	37,58449470	09.06.2019 09:18:03
55,75673500	37,58457490	09.06.2019 09:18:07
55,75690160	37,58457010	09.06.2019 09:18:14
55,75698800	37,58456170	09.06.2019 09:18:19
55,75789030	37,58463780	09.06.2019 09:18:31
55,75726820	37,58453710	09.06.2019 09:18:25
55,75875630	37,58483550	09.06.2019 09:18:38
55,75954210	37,58527170	09.06.2019 09:18:44

Нулевые и отрицательные интервалы

55,75504290	37,58472160	09.06.2019 09:17:34
55,75491030	37,58458640	09.06.2019 09:17:39
55,75496650	37,58437700	09.06.2019 09:17:45
55,75546590	37,58444770	09.06.2019 09:17:51
55,75596390	37,58451480	09.06.2019 09:17:57
55,75639320	37,58449470	09.06.2019 09:18:03
55,75639320	37,58449470	09.06.2019 09:18:03
55,75673500	37,58457490	09.06.2019 09:18:07
55,75690160	37,58457010	09.06.2019 09:18:14
55,75698800	37,58456170	09.06.2019 09:18:19
55,75789030	37,58463780	09.06.2019 09:18:31
55,75726820	37,58453710	09.06.2019 09:18:25
55,75875630	37,58483550	09.06.2019 09:18:38
55,75954210	37,58527170	09.06.2019 09:18:44

Нулевые и отрицательные интервалы

55,75504290	37,58472160	09.06.2019 09:17:34
55,75491030	37,58458640	09.06.2019 09:17:39
55,75496650	37,58437700	09.06.2019 09:17:45
55,75546590	37,58444770	09.06.2019 09:17:51
55,75596390	37,58451480	09.06.2019 09:17:57
55,75639320	37,58449470	09.06.2019 09:18:03
55,75639320	37,58449470	09.06.2019 09:18:03
55,75673500	37,58457490	09.06.2019 09:18:07
55,75690160	37,58457010	09.06.2019 09:18:14
55,75698800	37,58456170	09.06.2019 09:18:19
55,75789030	37,58463780	09.06.2019 09:18:31
55,75726820	37,58453710	09.06.2019 09:18:25
55,75875630	37,58483550	09.06.2019 09:18:38
55,75954210	37,58527170	09.06.2019 09:18:44

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    points
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    points
```

```
[<Fact>]
let ``removeZeroOrNegativeTimespans - without points - returns empty list`` () =
    let source = []

    let actual = removeZeroOrNegativeTimespans source

    Assert.Empty(actual)
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    points
```

```
[<Fact>]
let ``removeZeroOrNegativeTimespans - with single point - returns single point`` () =
    let source = [SensorItem(0.0, 0.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:00+03:00"))]

    let actual = removeZeroOrNegativeTimespans source

    let expected = [SensorItem(0.0, 0.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:00+03:00"))]
    Assert.Equal<seq<SensorItem>>(expected, actual)
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    points
```

```
[<Fact>]
let ``removeZeroOrNegativeTimespans - with zero timespan - removes point`` () =
    let source = [SensorItem(0.0, 0.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:15+03:00"));
                  SensorItem(1.0, 1.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:15+03:00"));
                  SensorItem(2.0, 2.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:16+03:00"))]

    let actual = removeZeroOrNegativeTimespans source

    let expected = [SensorItem(0.0, 0.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:15+03:00"));
                    SensorItem(2.0, 2.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:16+03:00"))]
    Assert.Equal<seq<SensorItem>>(expected, actual)
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    match points with
    | [] -> []
    | [p] -> [p]
    | p1::_ -> let ps = points
                       |> List.pairwise
                       |> List.filter (fun (p1: SensorItem, p2) ->
                                       p2.Timestamp - p1.Timestamp > TimeSpan.Zero)
                       |> List.map (fun (_, p) -> p)

               p1::ps
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    match points with
    | [] -> []
    | [p] -> [p]
    | p1::_ -> let ps = points
                       |> List.pairwise
                       |> List.filter (fun (p1: SensorItem, p2) ->
                                       p2.Timestamp - p1.Timestamp > TimeSpan.Zero)
                       |> List.map (fun (_, p) -> p)

               p1::ps
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    match points with
    | [] -> []
    | [p] -> [p]
    | p1::_ -> let ps = points
                       |> List.pairwise
                       |> List.filter (fun (p1: SensorItem, p2) ->
                                       p2.Timestamp - p1.Timestamp > TimeSpan.Zero)
                       |> List.map (fun (_, p) -> p)

               p1::ps
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    match points with
    | [] -> []
    | [p] -> [p]
    | p1::_ -> let ps = points
                       |> List.pairwise
                       |> List.filter (fun (p1: SensorItem, p2) ->
                                           p2.Timestamp - p1.Timestamp > TimeSpan.Zero)
                       |> List.map (fun (_, p) -> p)

               p1::ps
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    match points with
    | [] -> []
    | [p] -> [p]
    | p1::_ -> let ps = points
                       |> List.pairwise
                       |> List.filter (fun (p1: SensorItem, p2) ->
                                       p2.Timestamp - p1.Timestamp > TimeSpan.Zero)
                       |> List.map (fun (_, p) -> p)

               p1::ps
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    match points with
    | [] -> []
    | [p] -> [p]
    | p1::_ -> let ps = points
                       |> List.pairwise
                       |> List.filter (fun (p1: SensorItem, p2) ->
                                       p2.Timestamp - p1.Timestamp > TimeSpan.Zero)
                       |> List.map (fun (_, p) -> p)

               p1::ps
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    match points with
    | [] -> []
    | [p] -> [p]
    | p1::_ -> let ps = points
                       |> List.pairwise
                       |> List.filter (fun (p1: SensorItem, p2) ->
                                       p2.Timestamp - p1.Timestamp > TimeSpan.Zero)
                       |> List.map (fun (_, p) -> p)

               p1::ps
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    match points with
    | [] -> []
    | [p] -> [p]
    | p1::_ -> let ps = points
                       |> List.pairwise
                       |> List.filter (fun (p1: SensorItem, p2) ->
                                       p2.Timestamp - p1.Timestamp > TimeSpan.Zero)
                       |> List.map (fun (_, p) -> p)

               p1::ps
```

$x \mid> f \mid> g \mid> h \leftrightarrow h(g(f(x)))$

Нулевые и отрицательные интервалы

```
public IReadOnlyList<SensorItem> Remove(IReadOnlyList<SensorItem> points)
{
    if (points.Count < 2)
        return points;

    var result = new List<SensorItem> { points[0] };

    for (int i = 1; i < points.Length; i++)
    {
        if (points[i].Timespan - points[i - 1].Timespan > TimeSpan.Zero)
            result.Add(points[i]);
    }

    return result;
}
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    match points with
    | [] -> []
    | [p] -> [p]
    | p1::_ -> let ps = points
                       |> List.pairwise
                       |> List.filter (fun (p1: SensorItem, p2) ->
                                           p2.Timestamp - p1.Timestamp > TimeSpan.Zero)
                       |> List.map (fun (_, p) -> p)

               p1::ps
```

Нулевые и отрицательные интервалы

```
[<Fact>]
let ``removeZeroOrNegativeTimespans - with positime timespans - returns same list`` () =
    let source = [SensorItem(0.0, 0.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:14+03:00"));
                  SensorItem(1.0, 1.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:15+03:00"));
                  SensorItem(2.0, 2.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:16+03:00"))]

    let actual = removeZeroOrNegativeTimespans source

    let expected = [SensorItem(0.0, 0.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:14+03:00"));
                    SensorItem(1.0, 1.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:15+03:00"));
                    SensorItem(2.0, 2.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:16+03:00"))]
    Assert.Equal<seq<SensorItem>>(expected, actual)
```


Внезапно

Нулевые и отрицательные интервалы

55,67005250	37,46812270	17.04.2019	11:07:26
55,67009476	37,46826623	01.09.1999	11:07:32
55,67008554	37,46821526	01.09.1999	11:07:42
55,66993610	37,46826690	17.04.2019	11:07:41

Нулевые и отрицательные интервалы

```
[<Fact>]
let ``removeZeroOrNegativeTimespans - with two negative timespans - removes both points`` () =
    let source = [SensorItem(0.0, 0.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:15+03:00"));
                  SensorItem(1.0, 1.0, 0.0, 0.0, DateTimeOffset.Parse("2018-11-07T16:38:16+03:00"));
                  SensorItem(2.0, 2.0, 0.0, 0.0, DateTimeOffset.Parse("2018-11-07T16:38:17+03:00"));
                  SensorItem(3.0, 3.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:18+03:00"))]

    let actual = removeZeroOrNegativeTimespans source

    let expected = [SensorItem(0.0, 0.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:15+03:00"));
                    SensorItem(3.0, 3.0, 0.0, 0.0, DateTimeOffset.Parse("2018-12-07T16:38:18+03:00"))]
    Assert.Equal<seq<SensorItem>>(expected, actual)
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    let rec filter (p1: SensorItem) points =
        match points with
        | (p2: SensorItem)::ps -> let Δtime = p2.Timestamp - p1.Timestamp
                                  if Δtime > TimeSpan.Zero
                                  then p2::filter p2 ps
                                  else filter p1 ps
        | _ -> points

    match points with
    | p1::ps -> p1::filter p1 ps
    | _ -> points
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    let rec filter (p1: SensorItem) points =
        match points with
        | (p2: SensorItem)::ps -> let Δtime = p2.Timestamp - p1.Timestamp
                                  if Δtime > TimeSpan.Zero
                                  then p2::filter p2 ps
                                  else filter p1 ps
        | _ -> points

    match points with
    | p1::ps -> p1::filter p1 ps
    | _ -> points
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    let rec filter (p1: SensorItem) points =
        match points with
        | (p2: SensorItem)::ps -> let Δtime = p2.Timestamp - p1.Timestamp
                                  if Δtime > TimeSpan.Zero
                                  then p2::filter p2 ps
                                  else filter p1 ps
        | _ -> points

    match points with
    | p1::ps -> p1::filter p1 ps
    | _ -> points
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    let rec filter (p1: SensorItem) points =
        match points with
        | (p2: SensorItem)::ps -> let Δtime = p2.Timestamp - p1.Timestamp
                                  if Δtime > TimeSpan.Zero
                                  then p2::filter p2 ps
                                  else filter p1 ps
        | _ -> points

    match points with
    | p1::ps -> p1::filter p1 ps
    | _ -> points
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    let rec filter (p1: SensorItem) points =
        match points with
        | (p2: SensorItem)::ps -> let Δtime = p2.Timestamp - p1.Timestamp
                                  if Δtime > TimeSpan.Zero
                                  then p2::filter p2 ps
                                  else filter p1 ps
        | _ -> points

    match points with
    | p1::ps -> p1::filter p1 ps
    | _ -> points
```

Нулевые и отрицательные интервалы

```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    let rec filter (p1: SensorItem) points =
        match points with
        | (p2: SensorItem)::ps -> let Δtime = p2.Timestamp - p1.Timestamp
                                  if Δtime > TimeSpan.Zero
                                  then p2::filter p2 ps
                                  else filter p1 ps
        | _ -> points

    match points with
    | p1::ps -> p1::filter p1 ps
    | _ -> points
```

Нулевые и отрицательные интервалы

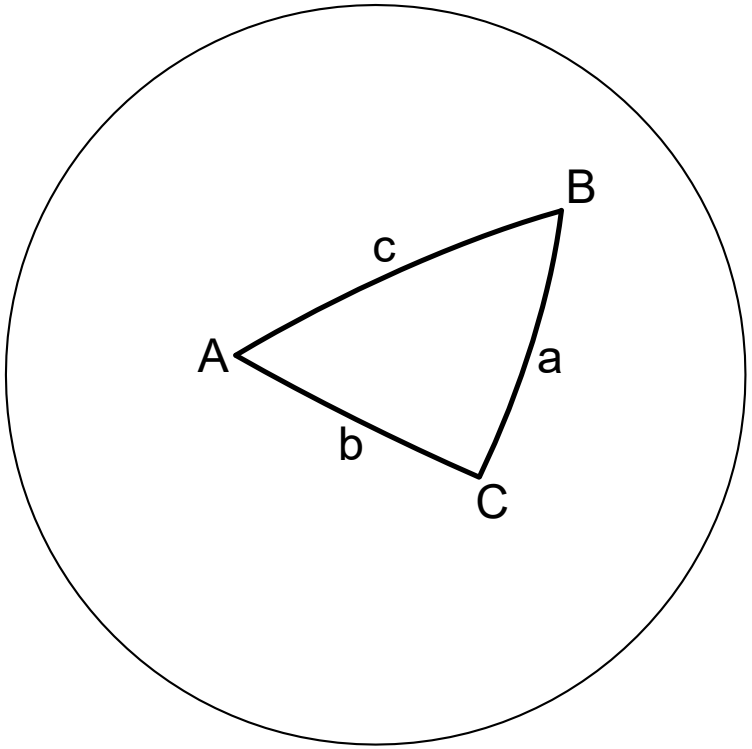
```
// <summary>
// Removes points with zero or negative time spans.
// </summary>
let removeZeroOrNegativeTimespans points =
    let rec filter (p1: SensorItem) points =
        match points with
        | (p2: SensorItem)::ps -> let Δtime = p2.Timestamp - p1.Timestamp
                                  if Δtime > TimeSpan.Zero
                                  then p2::filter p2 ps
                                  else filter p1 ps
        | _ -> points

    match points with
    | p1::ps -> p1::filter p1 ps
    | _ -> points
```

Всплески и дребезг скорости

Наращивание программ с помощью больших блоков высокого уровня, созданных когда-то раньше или кем-то другим, помогает избежать целых уровней сложности.

Фредерик Брукс

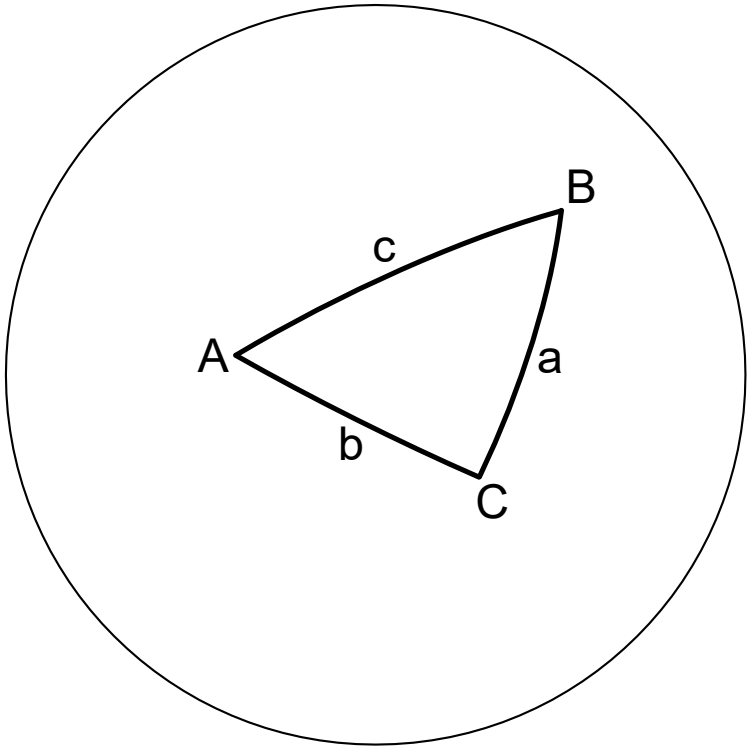
Всплески и дребезг скорости



Сферическая теорема косинусов

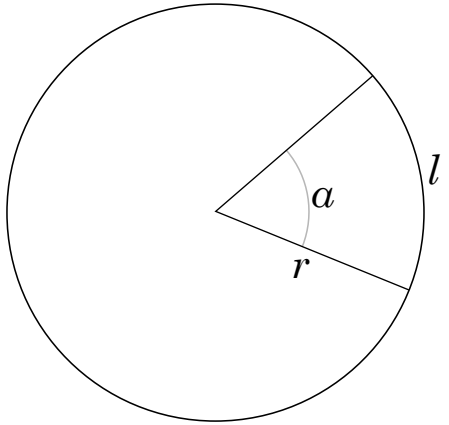
$$\cos c = \cos a \cos b + \sin a \sin b \cos C$$

Всплески и дребезг скорости



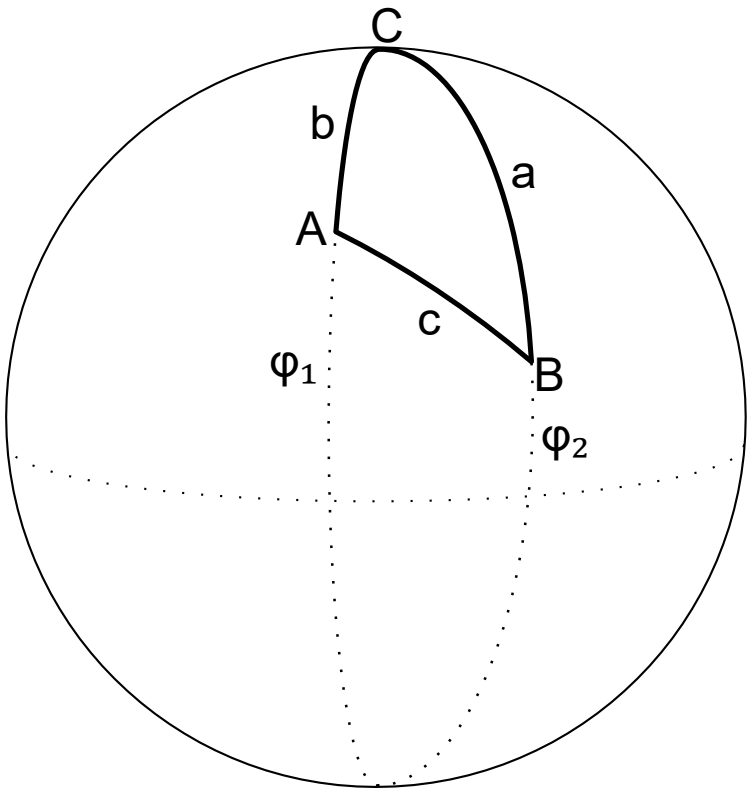
Сферическая теорема косинусов

$$\cos c = \cos a \cos b + \sin a \sin b \cos C$$



$$l = \alpha r$$

Всплески и дребезг скорости



$$C = \lambda_2 - \lambda_1$$

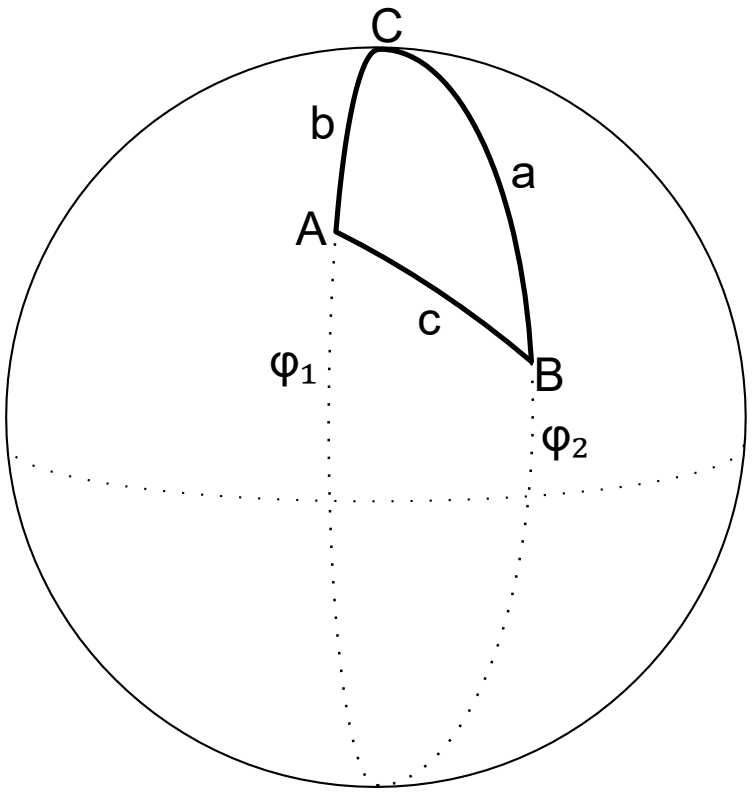
$$a = \frac{\pi}{2} - \varphi_2$$

$$b = \frac{\pi}{2} - \varphi_1$$

$$\cos c = \cos\left(\frac{\pi}{2} - \varphi_2\right)\cos\left(\frac{\pi}{2} - \varphi_1\right) +$$

$$+ \sin\left(\frac{\pi}{2} - \varphi_2\right)\sin\left(\frac{\pi}{2} - \varphi_1\right)\cos(\lambda_2 - \lambda_1)$$

Всплески и дребезг скорости



$$C = \lambda_2 - \lambda_1$$

$$a = \frac{\pi}{2} - \varphi_2$$

$$b = \frac{\pi}{2} - \varphi_1$$

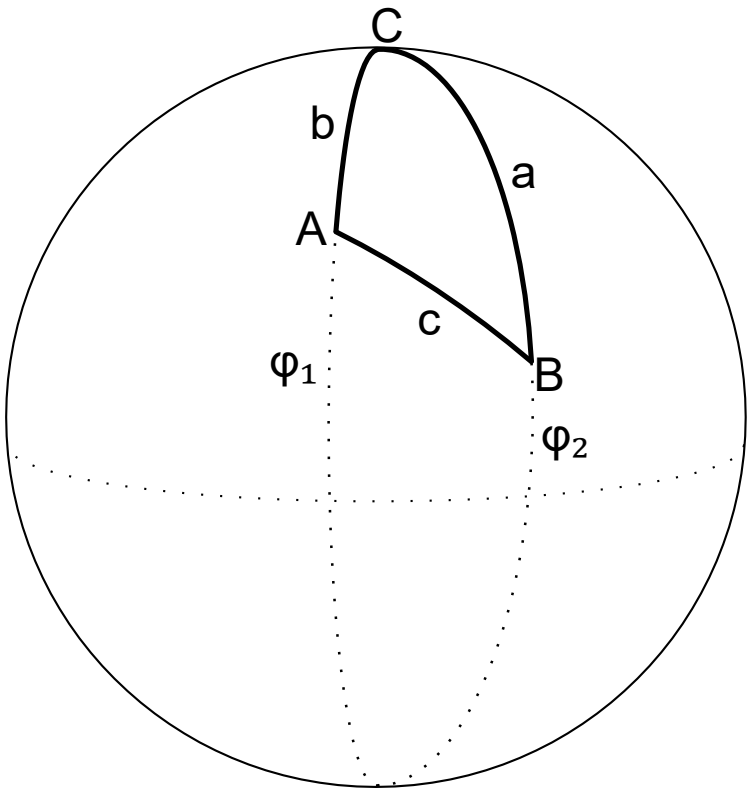
$$\cos\left(\frac{\pi}{2} - \alpha\right) = \sin\alpha$$

$$\sin\left(\frac{\pi}{2} - \alpha\right) = \cos\alpha$$

$$\cos c = \cos\left(\frac{\pi}{2} - \varphi_2\right)\cos\left(\frac{\pi}{2} - \varphi_1\right) +$$

$$+ \sin\left(\frac{\pi}{2} - \varphi_2\right)\sin\left(\frac{\pi}{2} - \varphi_1\right)\cos(\lambda_2 - \lambda_1)$$

Всплески и дребезг скорости



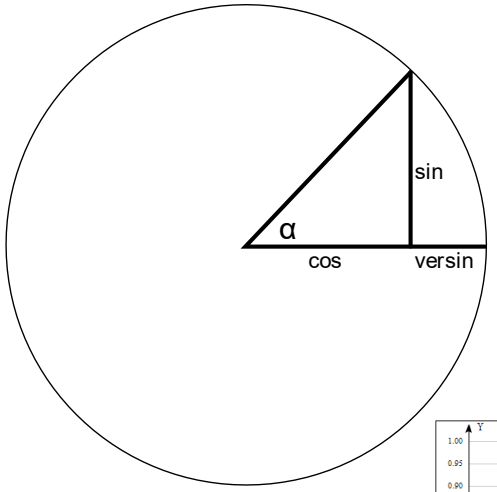
$$C = \lambda_2 - \lambda_1$$

$$a = \frac{\pi}{2} - \varphi_2$$

$$b = \frac{\pi}{2} - \varphi_1$$

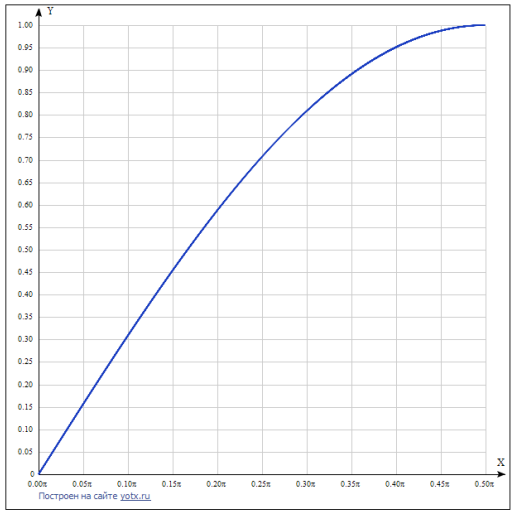
$$\cos c = \sin\varphi_2 \sin\varphi_1 + \cos\varphi_2 \cos\varphi_1 \cos(\lambda_2 - \lambda_1)$$

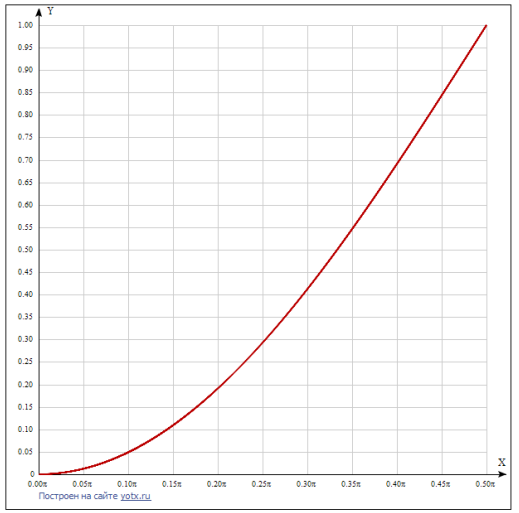
Всплески и дребезг скорости



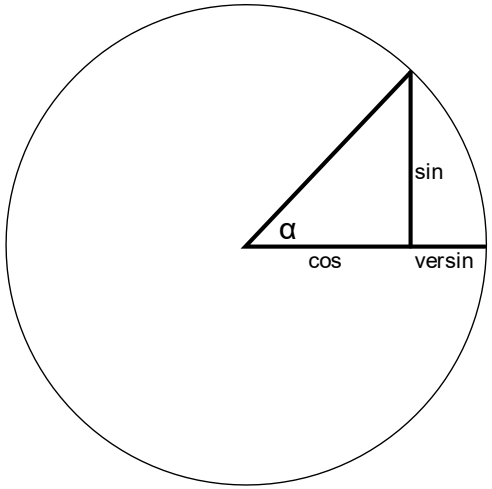
$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$





Всплески и дребезг скорости

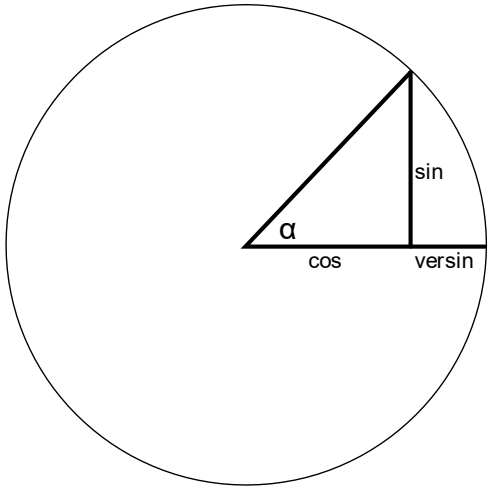


$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$\cos c = \cos a \cos b + \sin a \sin b \cos C$$

Всплески и дребезг скорости

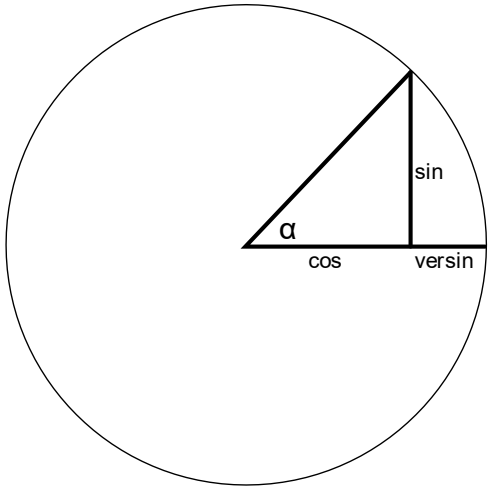


$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$\cos c = \cos a \cos b + \sin a \sin b - \sin a \sin b + \sin a \sin b \cos C$$

Всплески и дребезг скорости

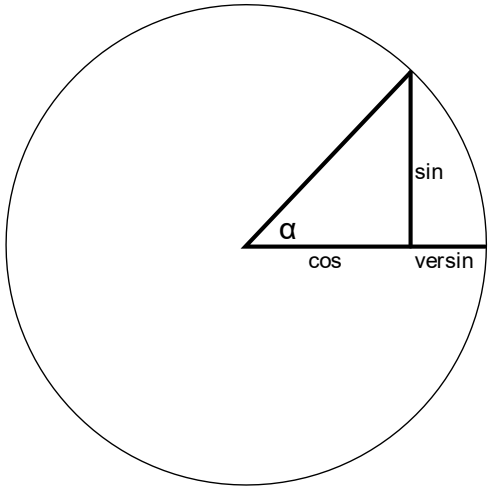


$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$\cos c = \cos(b-a) + \sin a \sin b\,(1-\cos C)$$

Всплески и дребезг скорости

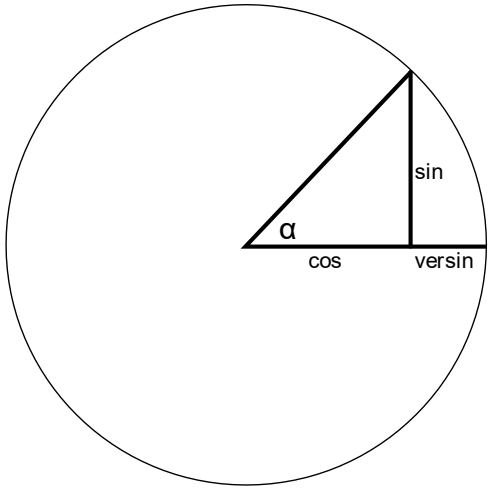


$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$1 - \cos c = 1 - \cos(b-a) + \sin a \sin b\,(1 - \cos C)$$

Всплески и дребезг скорости

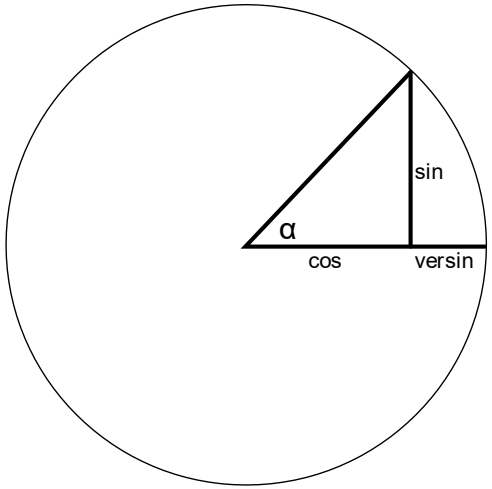


$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$\frac{1-\cos c}{2} = \frac{1-\cos(b-a)}{2} + \sin a \sin b \frac{1-\cos C}{2}$$

Всплески и дребезг скорости

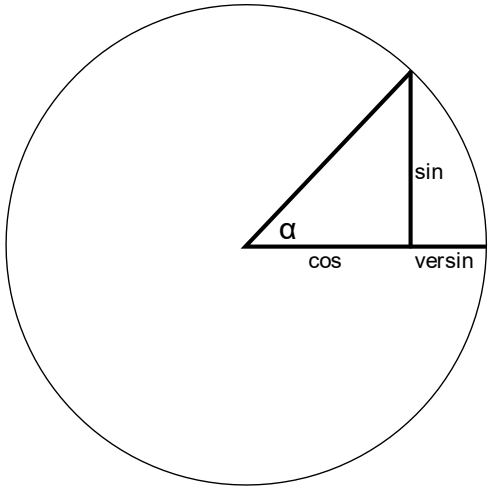


$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$\frac{1-\cos c}{2} = \frac{1-\cos(b-a)}{2} + \sin a \sin b \frac{1-\cos C}{2}$$

Всплески и дребезг скорости

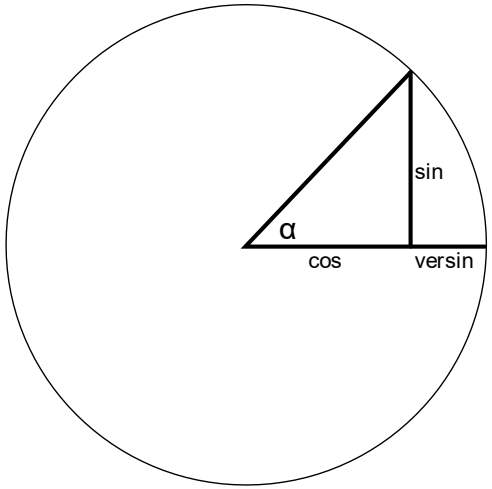


$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$\operatorname{hav} c = \operatorname{hav}(b-a) + \sin a \sin b \operatorname{hav} C$$

Всплески и дребезг скорости

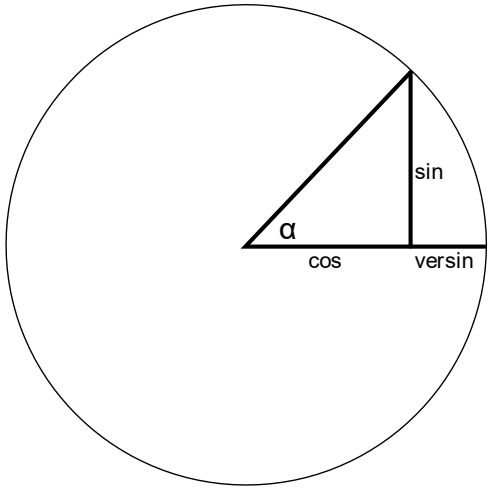


$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$\operatorname{hav} c = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos\varphi_2 \cos\varphi_1 \operatorname{hav}(\lambda_2 - \lambda_1)$$

Всплески и дребезг скорости

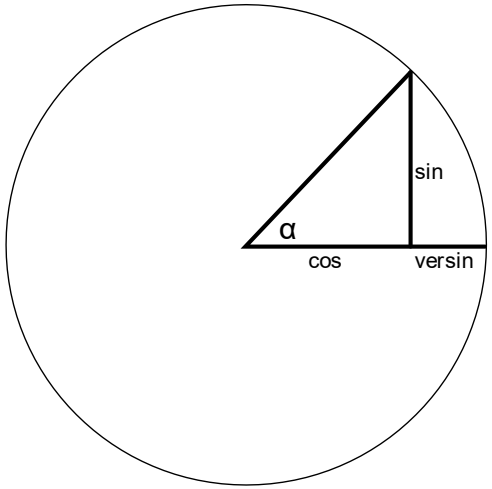


$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$\operatorname{hav} c = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos\varphi_2 \cos\varphi_1 \operatorname{hav}(\lambda_2 - \lambda_1)$$

Всплески и дребезг скорости

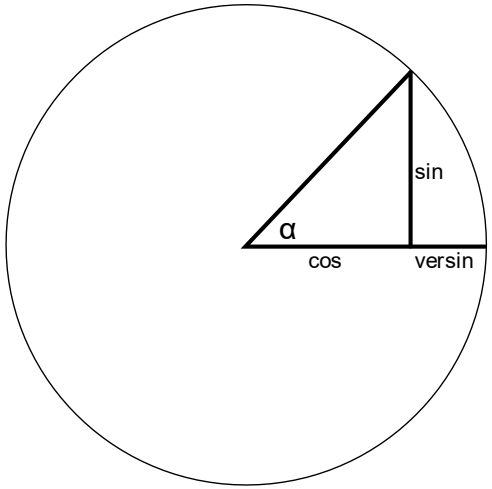


$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$\sin^2\frac{c}{2} = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos\varphi_2\cos\varphi_1\operatorname{hav}(\lambda_2 - \lambda_1)$$

Всплески и дребезг скорости



$$\operatorname{versin}\alpha = 1 - \cos\alpha$$

$$\operatorname{hav}\alpha = \sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$$

$$\sin^2\frac{c}{2} = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos\varphi_2\cos\varphi_1\operatorname{hav}(\lambda_2 - \lambda_1)$$

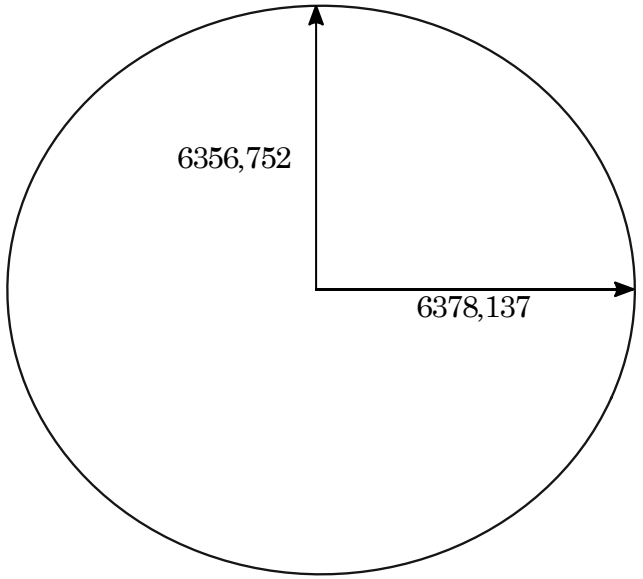
$$c = 2\arcsin\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos\varphi_2\cos\varphi_1\operatorname{hav}(\lambda_2 - \lambda_1)}$$

Всплески и дребезг скорости

$$c = 2 \arcsin \sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos \varphi_2 \cos \varphi_1 \operatorname{hav}(\lambda_2 - \lambda_1)}$$

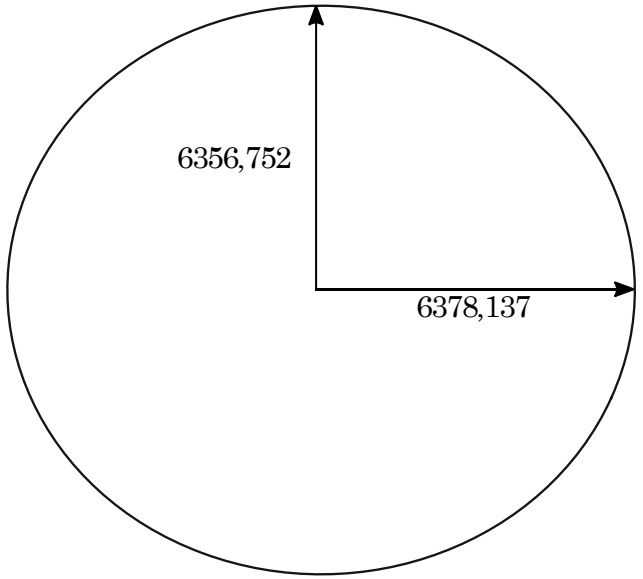
Всплески и дребезг скорости

$$c = 2 \arcsin \sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos \varphi_2 \cos \varphi_1 \operatorname{hav}(\lambda_2 - \lambda_1)}$$



Всплески и дребезг скорости

$$c = 2 \arcsin \sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos \varphi_2 \cos \varphi_1 \operatorname{hav}(\lambda_2 - \lambda_1)}$$



$$l = c \frac{6356{,}752 + 6378{,}137}{2}$$

Всплески и дребезг скорости

```
let distance latitude1 longitude1 latitude2 longitude2 =
    let hav x = sin(x/2.0) ** 2.0

    let earthEquatorialRadius = 6378.137
    let earthPolarRadius = 6356.752
    let averageEarthRadius = (earthEquatorialRadius + earthPolarRadius)/2.0

    let ϕ1 = radian latitude1
    let λ1 = radian longitude1

    let ϕ2 = radian latitude2
    let λ2 = radian longitude2

    let havc = hav (ϕ2 - ϕ1) + cos ϕ1 * cos ϕ2 * hav (λ2 - λ1)
    2.0 * asin (sqrt havc) * averageEarthRadius
```

Всплески и дребезг скорости

```
let distance latitude1 longitude1 latitude2 longitude2 =
    let hav x = sin(x/2.0) ** 2.0

    let earthEquatorialRadius = 6378.137
    let earthPolarRadius = 6356.752
    let averageEarthRadius = (earthEquatorialRadius + earthPolarRadius)/2.0

    let ϕ1 = radian latitude1
    let λ1 = radian longitude1

    let ϕ2 = radian latitude2
    let λ2 = radian longitude2

    let havc = hav (ϕ2 - ϕ1) + cos ϕ1 * cos ϕ2 * hav (λ2 - λ1)
    2.0 * asin (sqrt havc) * averageEarthRadius
```

Всплески и дребезг скорости

```
let distance latitude1 longitude1 latitude2 longitude2 =
    let hav x = sin(x/2.0) ** 2.0

    let earthEquatorialRadius = 6378.137
    let earthPolarRadius = 6356.752
    let averageEarthRadius = (earthEquatorialRadius + earthPolarRadius)/2.0

    let ϕ1 = radian latitude1
    let λ1 = radian longitude1

    let ϕ2 = radian latitude2
    let λ2 = radian longitude2

    let havc = hav (ϕ2 - ϕ1) + cos ϕ1 * cos ϕ2 * hav (λ2 - λ1)
    2.0 * asin (sqrt havc) * averageEarthRadius
```

Всплески и дребезг скорости

```
let distance latitude1 longitude1 latitude2 longitude2 =
    let hav x = sin(x/2.0) ** 2.0

    let earthEquatorialRadius = 6378.137
    let earthPolarRadius = 6356.752
    let averageEarthRadius = (earthEquatorialRadius + earthPolarRadius)/2.0

    let ϕ1 = radian latitude1
    let λ1 = radian longitude1

    let ϕ2 = radian latitude2
    let λ2 = radian longitude2

    let havc = hav (ϕ2 - ϕ1) + cos ϕ1 * cos ϕ2 * hav (λ2 - λ1)
    2.0 * asin (sqrt havc) * averageEarthRadius
```

Всплески и дребезг скорости

```
let distance latitude1 longitude1 latitude2 longitude2 =
    let hav x = sin(x/2.0) ** 2.0

    let earthEquatorialRadius = 6378.137
    let earthPolarRadius = 6356.752
    let averageEarthRadius = (earthEquatorialRadius + earthPolarRadius)/2.0

    let ϕ1 = radian latitude1
    let λ1 = radian longitude1

    let ϕ2 = radian latitude2
    let λ2 = radian longitude2

    let havc = hav (ϕ2 - ϕ1) + cos ϕ1 * cos ϕ2 * hav (λ2 - λ1)
    2.0 * asin (sqrt havc) * averageEarthRadius
```

Всплески и дребезг скорости

Расстояние от Москвы до Санкт-Петербурга

Всплески и дребезг скорости

Расстояние от Москвы до Санкт-Петербурга
Москва: 55,753960; 37,620393
Санкт-Петербург: 59,938630; 30,314130

Всплески и дребезг скорости

Расстояние от Москвы до Санкт-Петербурга
Москва: 55,753960; 37,620393
Санкт-Петербург: 59,938630; 30,314130
Расстояние: 634,37 км

Всплески и дребезг скорости

Расстояние от Москвы до Санкт-Петербурга
Москва: 55,753960; 37,620393
Санкт-Петербург: 59,938630; 30,314130
Расстояние: 634,37 км
Точность 0,5%
Вычисленное расстояние отличается от 634,37 не больше, чем на 634,37×0,005

Всплески и дребезг скорости

```
[<Fact>]
let ``distance - between Moscow and Saint Petersburg - equals 635km`` () =
    let mskLatitude = 55.753960
    let mskLongitude = 37.620393
    let spbLatitude = 59.938630
    let spbLongitude = 30.314130

    let actual = distance mskLatitude mskLongitude spbLatitude spbLongitude

    let expected = 634.37
    let epsilon = 0.005
    Assert.True(abs(actual - expected) < expected * epsilon)
```

Всплески и дребезг скорости

```
[<Fact>]
let ``distance - between Moscow and Saint Petersburg - equals 635km`` () =
    let mskLatitude = 55.753960
    let mskLongitude = 37.620393
    let spbLatitude = 59.938630
    let spbLongitude = 30.314130

    let actual = distance mskLatitude mskLongitude spbLatitude spbLongitude

    let expected = 634.37
    let epsilon = 0.005
    Assert.True(abs(actual - expected) < expected * epsilon)
```

Всплески и дребезг скорости

```
let velocity (p1: SensorItem) (p2: SensorItem) =
    let Δtime = (p2.Timestamp - p1.Timestamp).TotalHours
    let Δdistance = distance p1.Latitude p1.Longitude p2.Latitude p2.Longitude

    Δdistance/Δtime
```

Всплески и дребезг скорости

```
[<Fact>]
let ``velocity - with 1 grade per hour at equator - equals 111km per hour`` () =
    let startItem = new SensorItem(0.0, 0.0, 0.0, 0.0,
                    new DateTimeOffset(2019, 4, 26, 11, 00, 00, TimeSpan.Zero))
    let endItem = new SensorItem(0.0, 1.0, 0.0, 0.0,
                  new DateTimeOffset(2019, 4, 26, 12, 00, 00, TimeSpan.Zero))

    let actual = velocity startItem endItem

    let expected = 111.0
    Assert.Equal(expected, actual, 0)
```

Всплески и дребезг скорости

```
[<Fact>]
let ``velocity - with 1 grade per hour at equator - equals 111km per hour`` () =
    let startItem = new SensorItem(0.0, 0.0, 0.0, 0.0,
                    new DateTimeOffset(2019, 4, 26, 11, 00, 00, TimeSpan.Zero))
    let endItem = new SensorItem(0.0, 1.0, 0.0, 0.0,
                  new DateTimeOffset(2019, 4, 26, 12, 00, 00, TimeSpan.Zero))

    let actual = velocity startItem endItem

    let expected = 111.0
    Assert.Equal(expected, actual, 0)
```

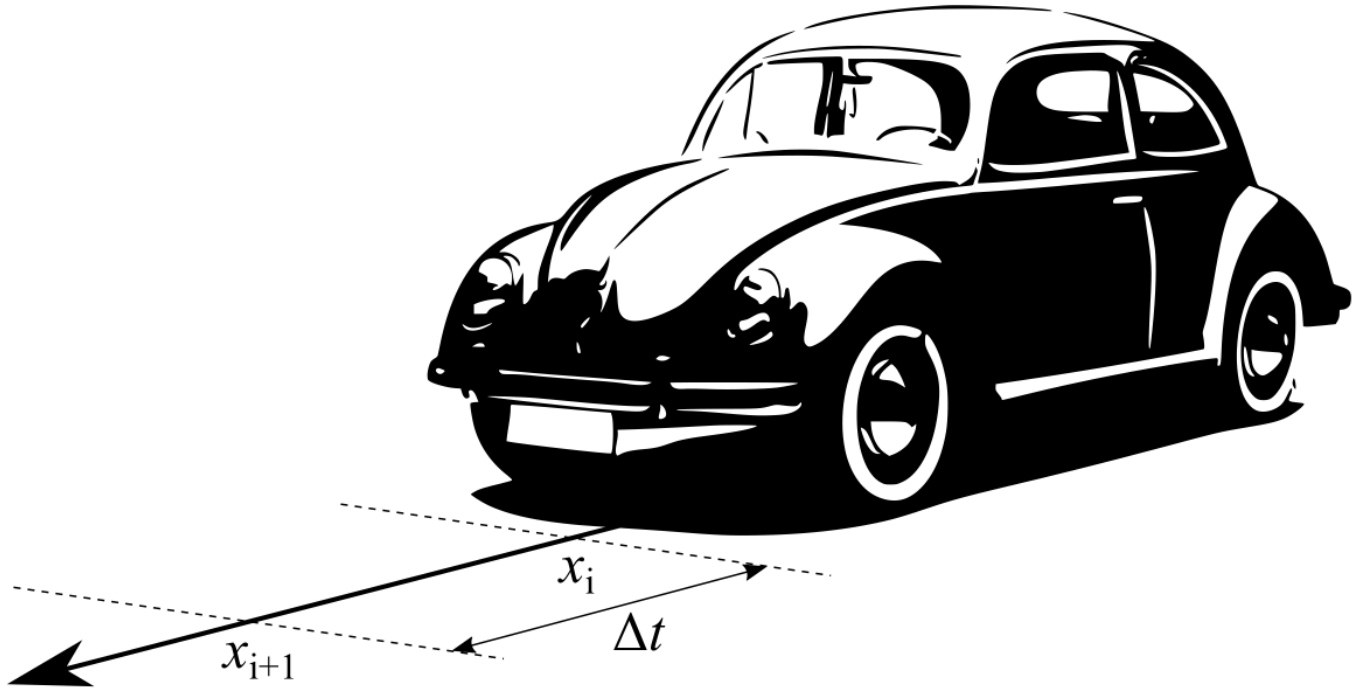
Всплески и дребезг скорости

```
let removeOutlineSpeedValues hiLimit points =
    let isOutlineSpeed p1 p2 =
        let velocity = velocity p1 p2

        velocity > hiLimit

    let rec filter p1 points =
        match points with
        | p2::ps -> if isOutlineSpeed p1 p2
                    then filter p1 ps
                    else p2::filter p2 ps
        | _ -> points

    match points with
    | p1::ps -> p1::filter p1 ps
    | _ -> points
```

Всплески и дребезг скорости

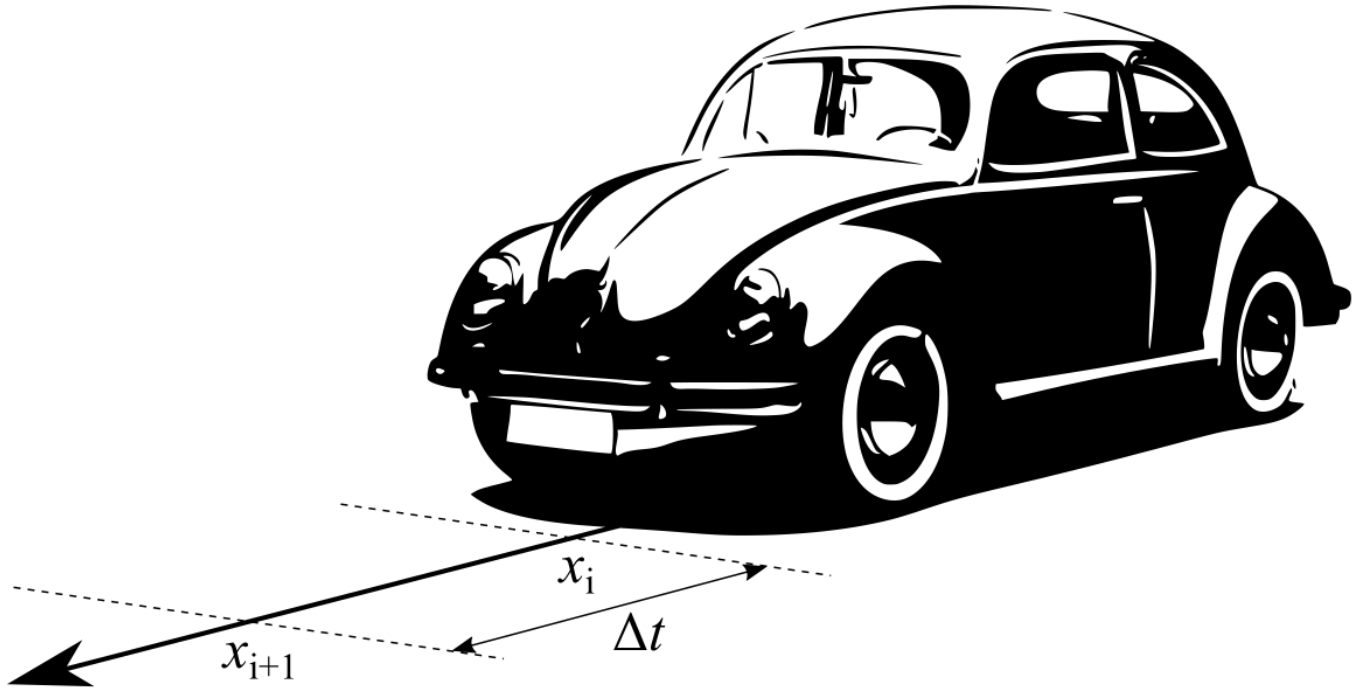
```
let removeOutlineSpeedValues hiLimit points =
    let isOutlineSpeed p1 p2 =
        let velocity = velocity p1 p2

        velocity > hiLimit

    let rec filter p1 points =
        match points with
        | p2::ps -> if isOutlineSpeed p1 p2
                    then filter p1 ps
                    else p2::filter p2 ps
        | _ -> points

    match points with
    | p1::ps -> p1::filter p1 ps
    | _ -> points
```

Фильтр Калмана



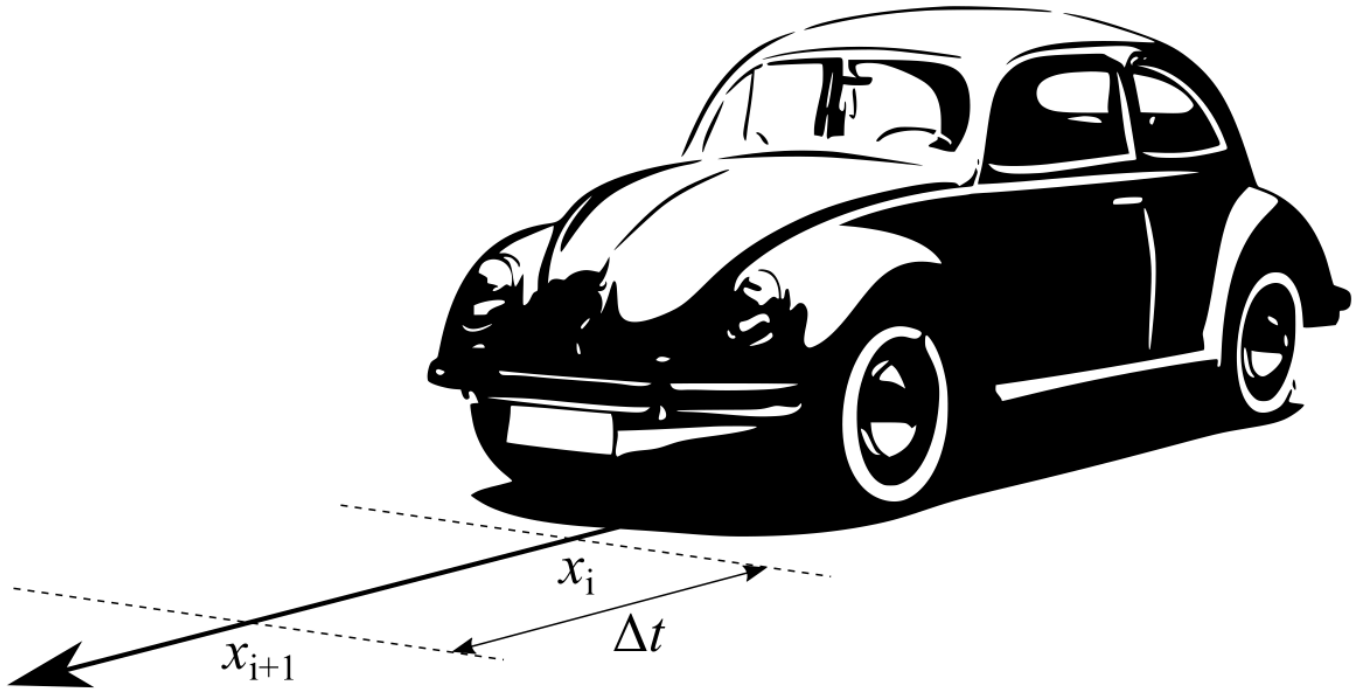
Фильтр Калмана

$$x_{i+1} = x_i + u_i \Delta t$$

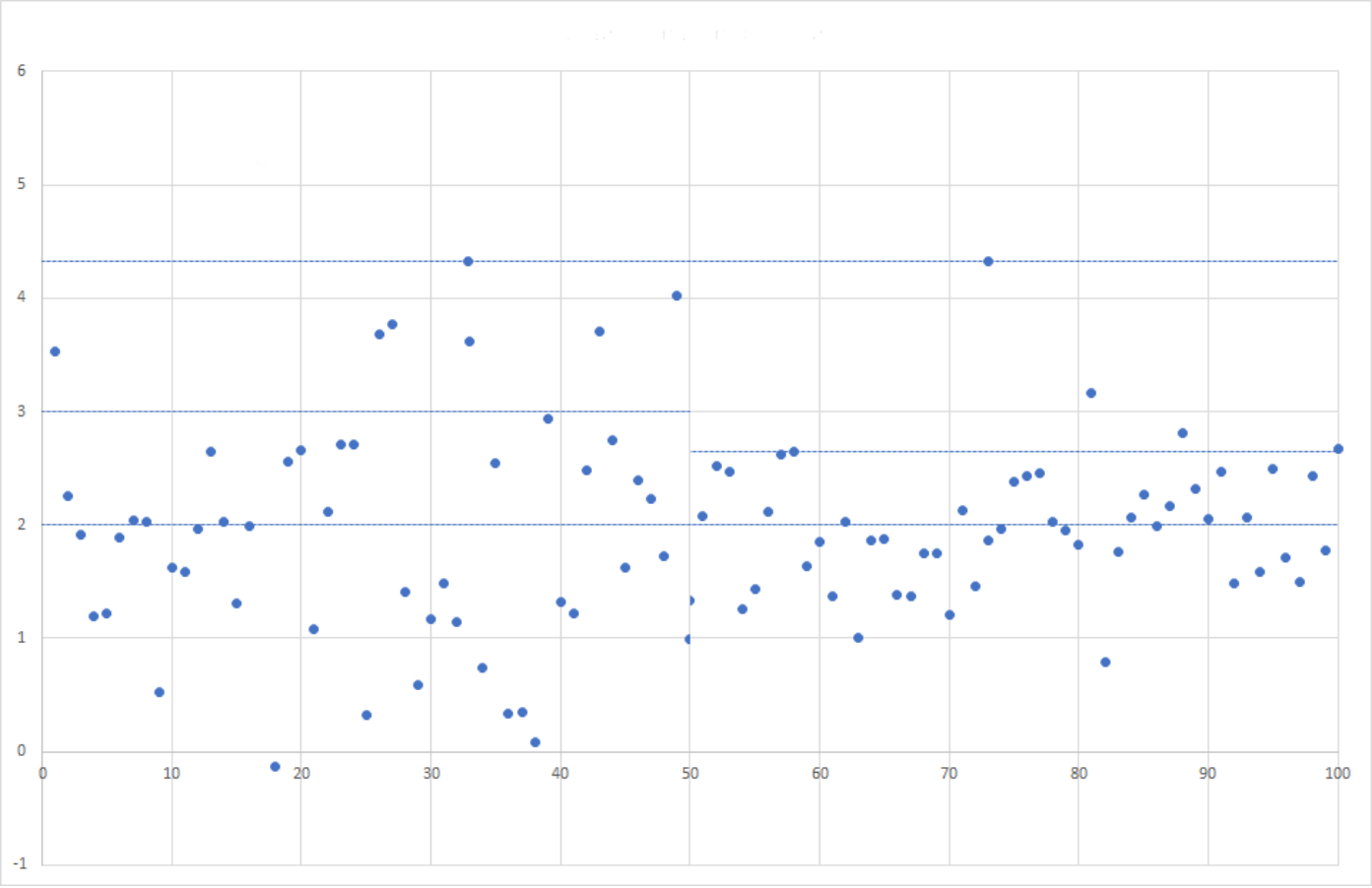


Фильтр Калмана

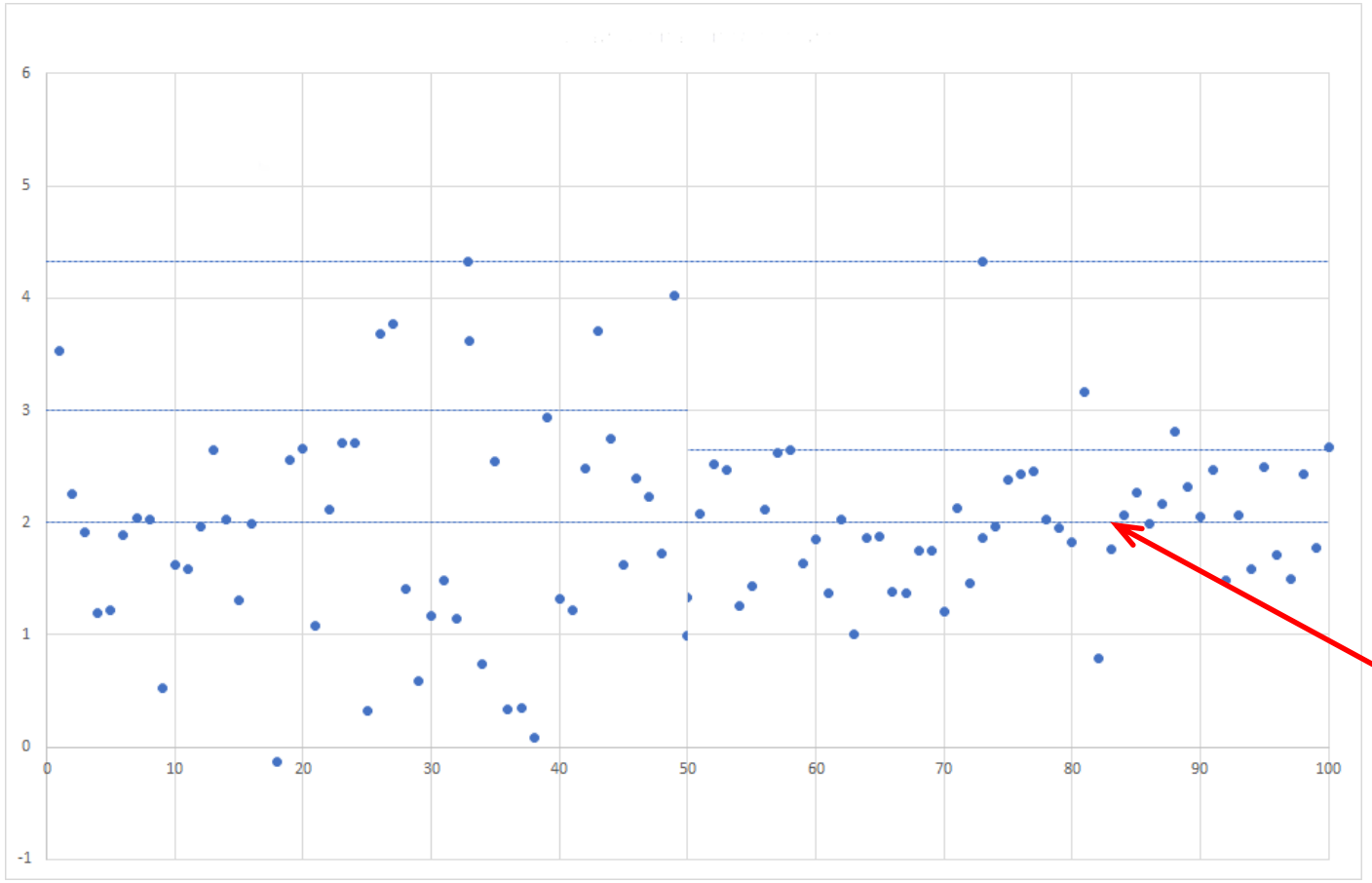
$$x_{i+1} = x_i + u_i \Delta t + \xi_i$$



Фильтр Калмана



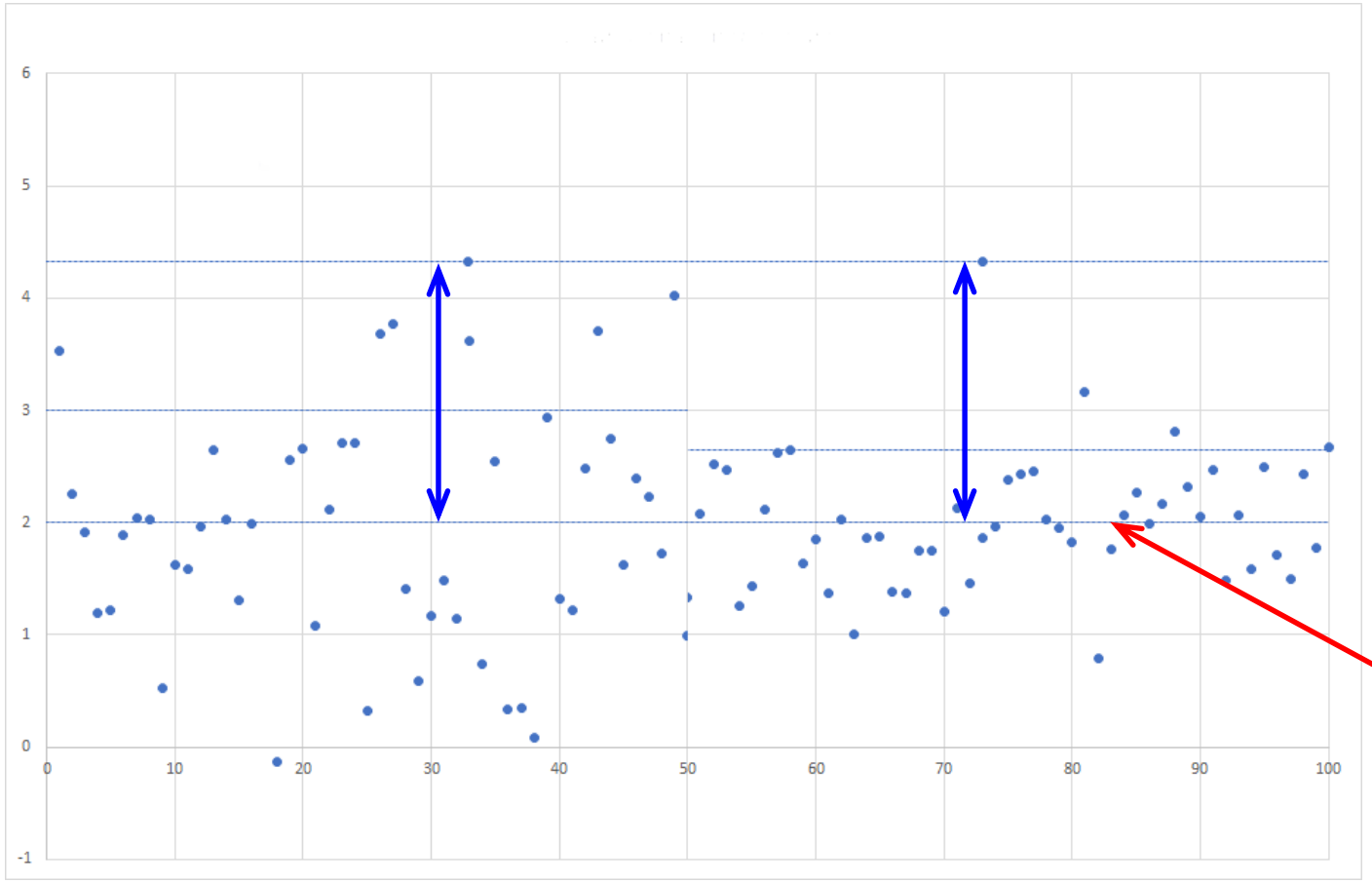
Фильтр Калмана



$$E\xi = 2$$

Среднее арифметическое
Математическое ожидание

Фильтр Калмана

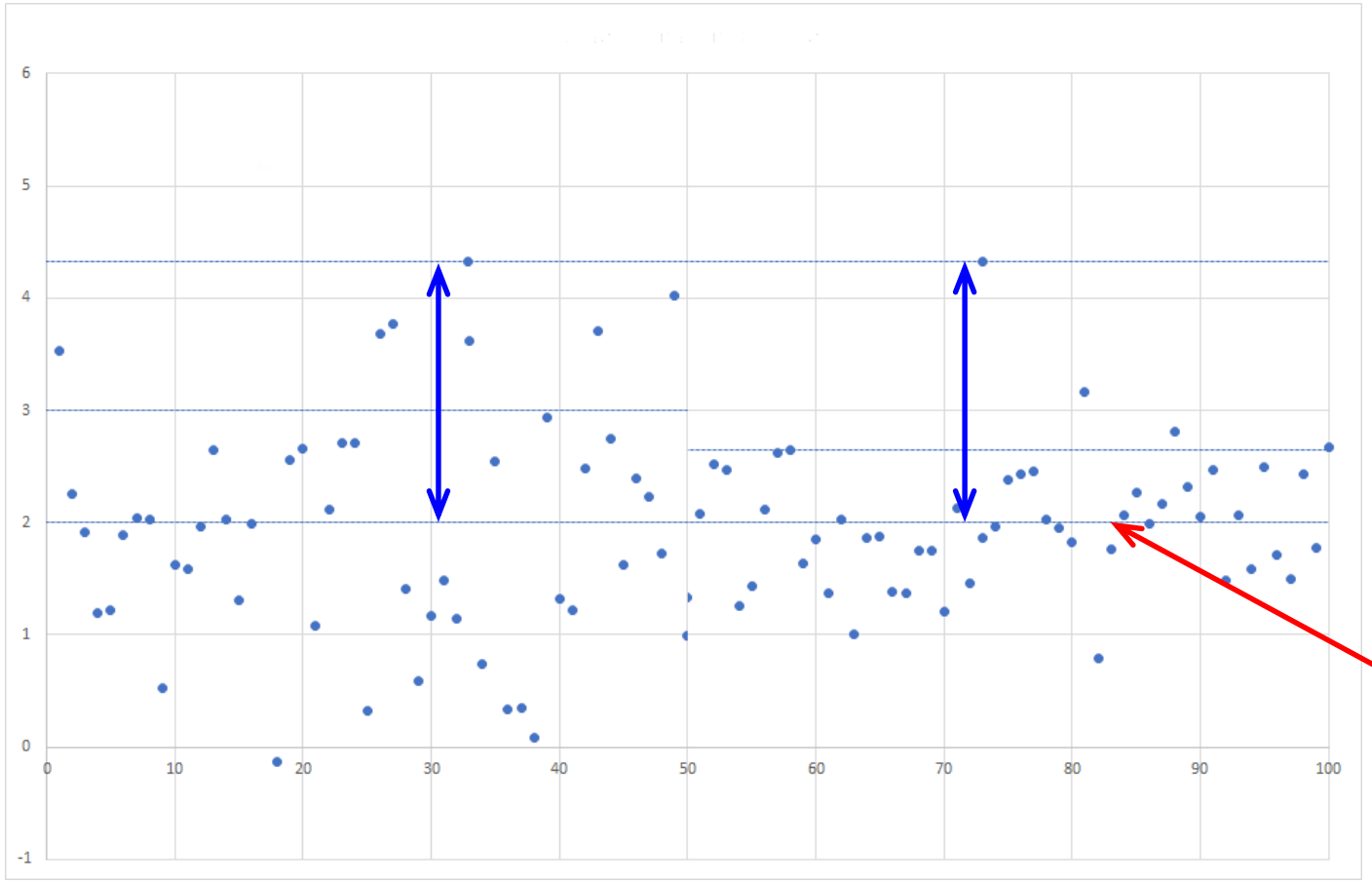


$$\xi - E\xi$$

$$E\xi = 2$$

Среднее арифметическое
Математическое ожидание

Фильтр Калмана

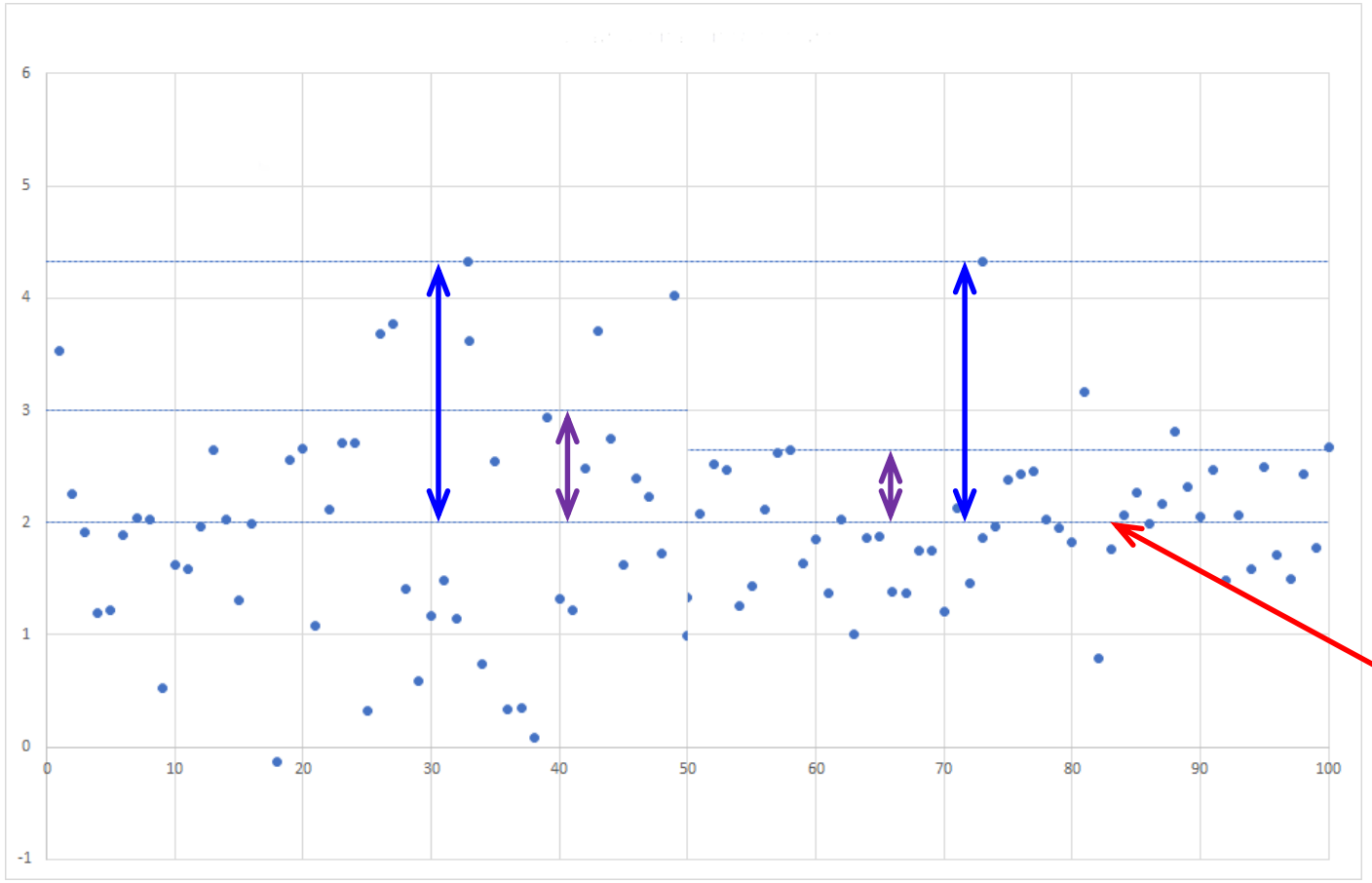


$$(\xi - E\xi)^2$$

$$E\xi = 2$$

Среднее арифметическое
Математическое ожидание

Фильтр Калмана



$$\sigma_\xi = \sqrt{E(\xi - E\xi)^2}$$

$$E(\xi - E\xi)^2$$

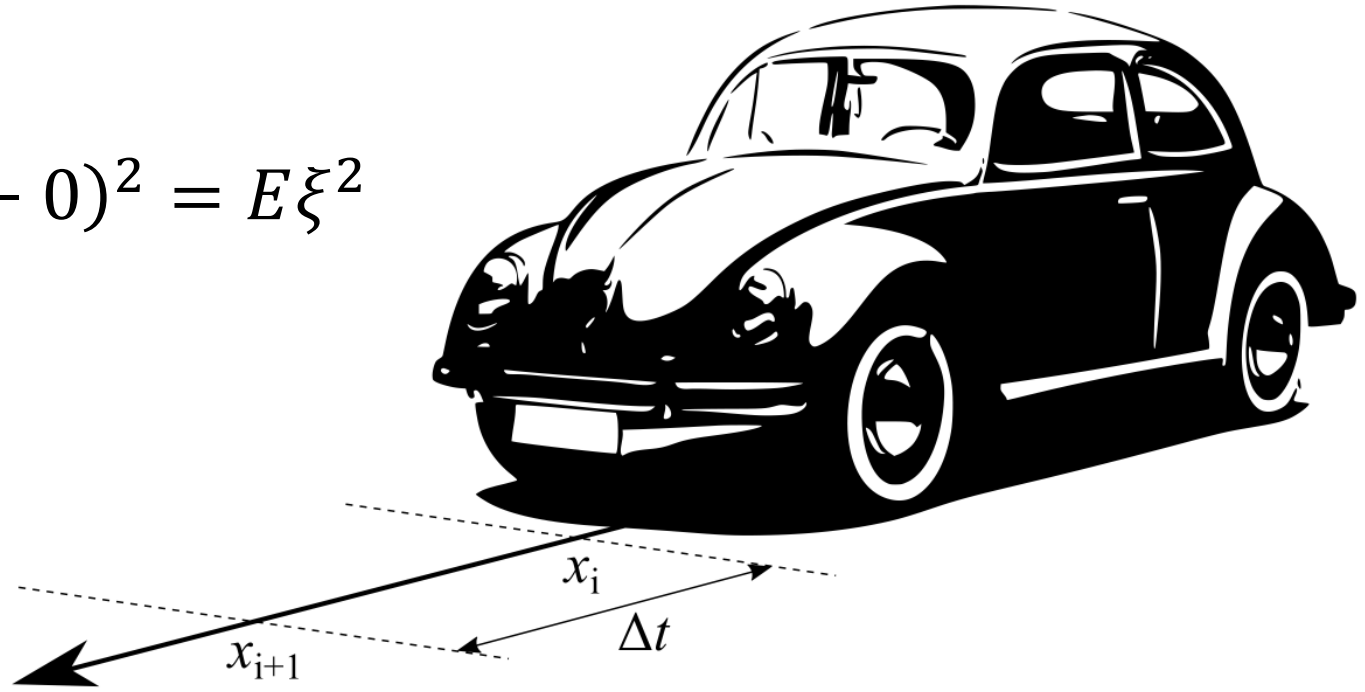
$$(\xi - E\xi)^2$$

$$E\xi = 2$$

Среднее арифметическое
Математическое ожидание

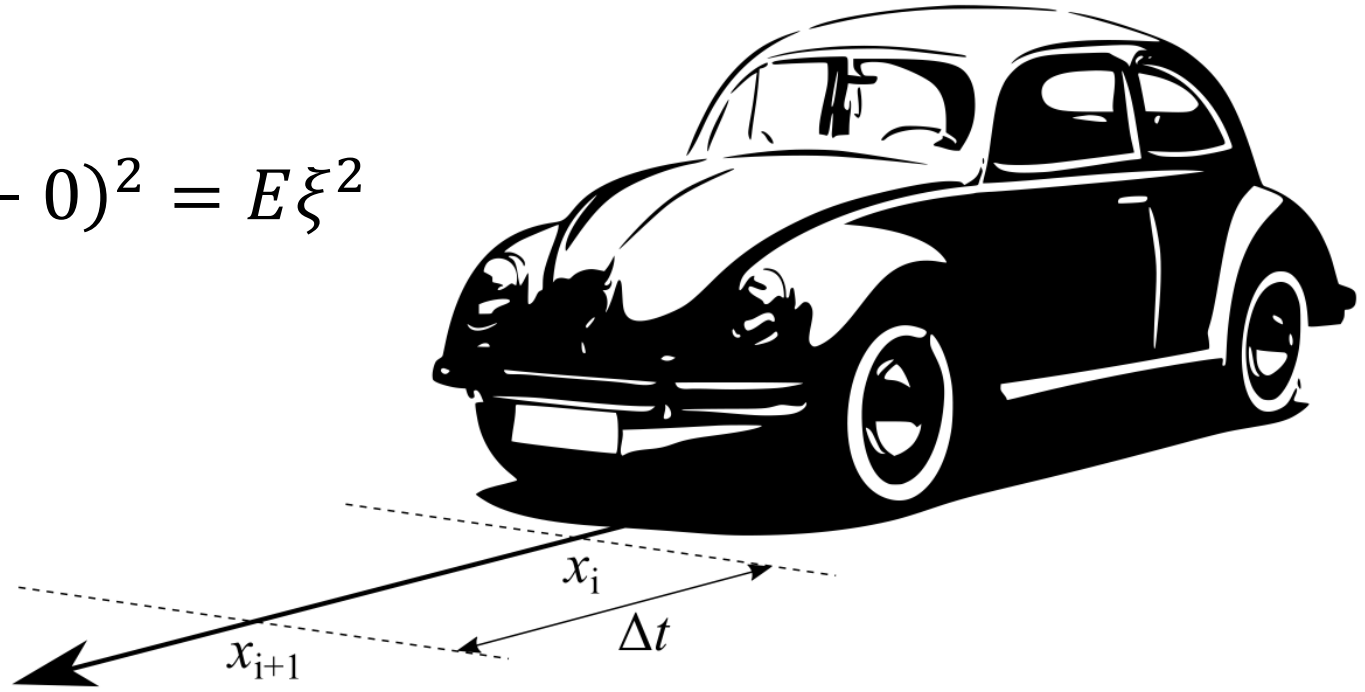
Фильтр Калмана

$$x_{i+1} = x_i + u_i \Delta t + \xi_i$$
$$E\xi = 0$$
$$\sigma_\xi^2 = E(\xi - E\xi)^2 = E(\xi - 0)^2 = E\xi^2$$



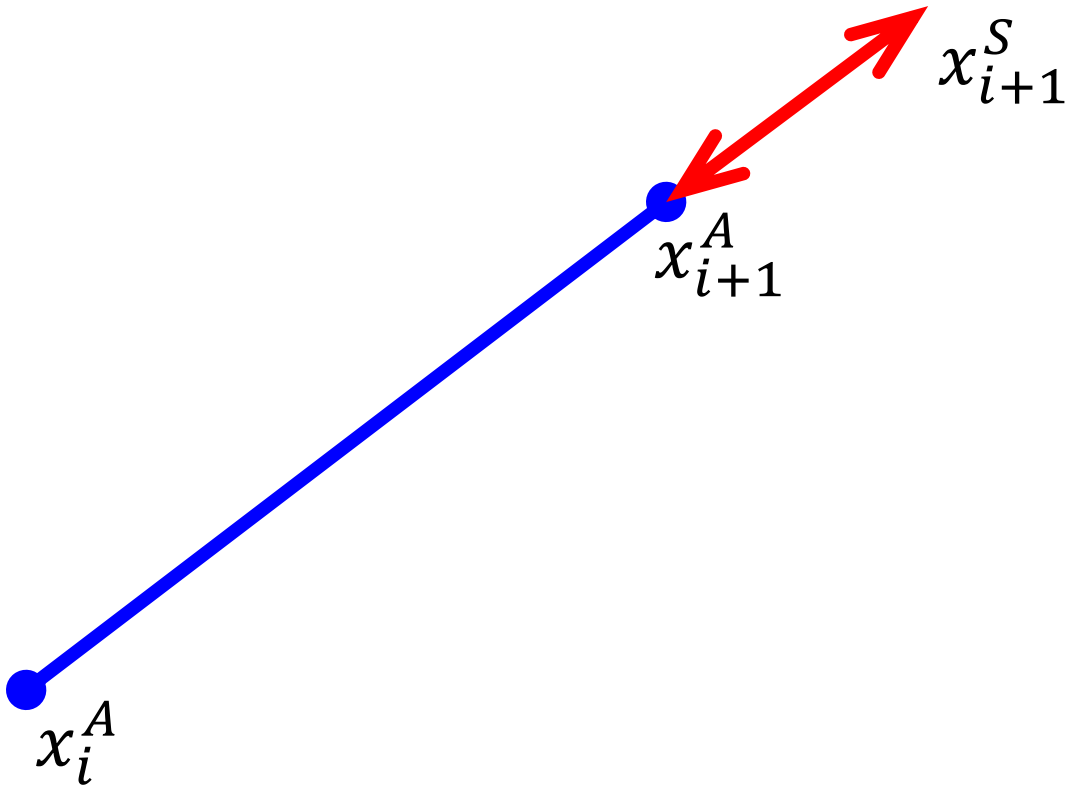
Фильтр Калмана

$$x_{i+1} = x_i + u_i \Delta t + \xi_i$$
$$E\xi = 0$$
$$\sigma_\xi^2 = E(\xi - E\xi)^2 = E(\xi - 0)^2 = E\xi^2$$

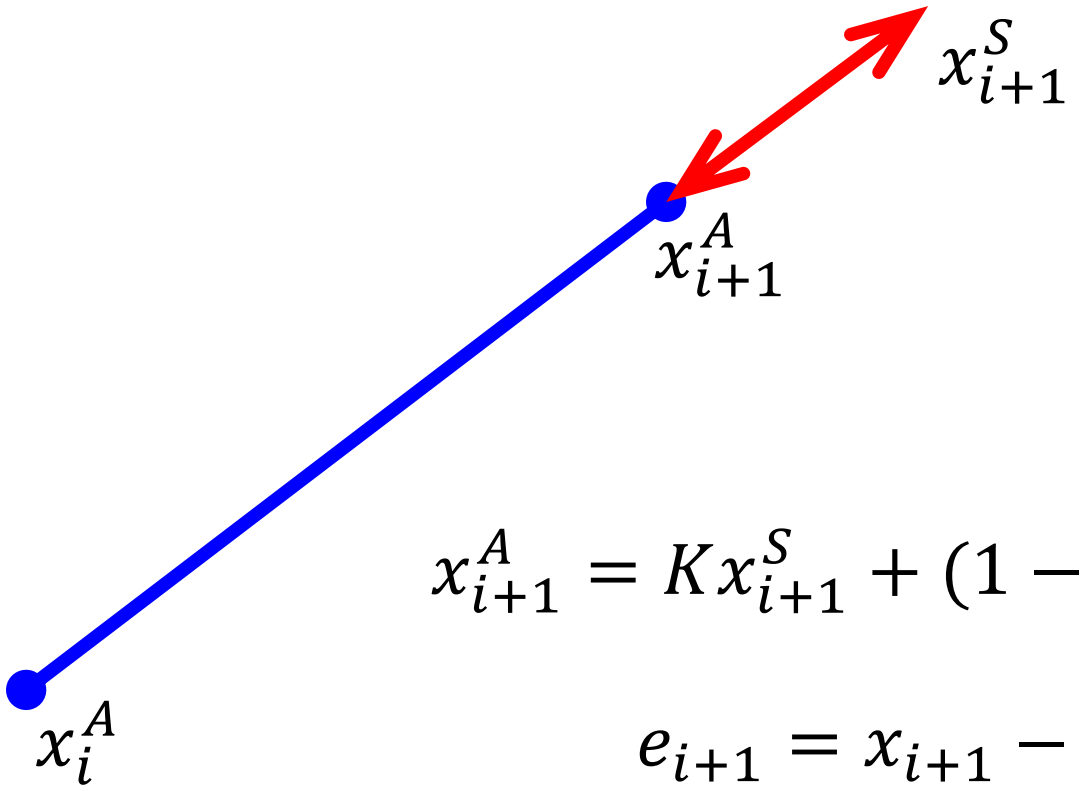


$$x_i^S = x_i + \eta_i$$
$$E\eta = 0$$
$$\sigma_\eta^2 = E(\eta - E\eta)^2 = E(\eta - 0)^2 = E\eta^2$$

Фильтр Калмана



Фильтр Калмана



$$x_{i+1}^{A} = K x_{i+1}^{S} + (1 - K)(x_{i}^{A} + u_{i}^{S} \Delta t)$$

$$e_{i+1} = x_{i+1} - x_{i+1}^{A}$$

Фильтр Калмана

$$e_{i+1} = x_{i+1} - x_{i+1}^{A}$$

$$x_{i+1}^{A} = K x_{i+1}^{S} + (1 - K)(x_i^{A} + u_i^{S} \Delta t)$$

$$x_{i+1} = x_i + u_i^{S} \Delta t + \xi_i$$

$$x_i^{S} = x_i + \eta_i$$

Фильтр Калмана

$$e_{i+1} = x_{i+1} - x_{i+1}^{A}$$

$$x_{i+1}^{A} = K x_{i+1}^{S} + (1 - K)(x_i^{A} + u_i^{S}\Delta t)$$

$$x_{i+1} = x_i + u_i^{S}\Delta t + \xi_i$$

$$x_{i+1}^{S} = x_{i+1} + \eta_{i+1}$$

Фильтр Калмана

$$e_{i+1} = x_{i+1} - x_{i+1}^{A}$$

$$x_{i+1}^{A} = K(x_{i+1} + \eta_{i+1}) + (1 - K)(x_i^{A} + u_i^{S}\Delta t)$$

$$x_{i+1} = x_i + u_i^{S}\Delta t + \xi_i$$

Фильтр Калмана

$$e_{i+1} = x_i + u_i^S \Delta t + \xi_i - x_{i+1}^A$$

$$x_{i+1}^A = K\left(x_i + u_i^S \Delta t + \xi_i + \eta_{i+1}\right) + (1 - K)(x_i^A + u_i^S \Delta t)$$

Фильтр Калмана

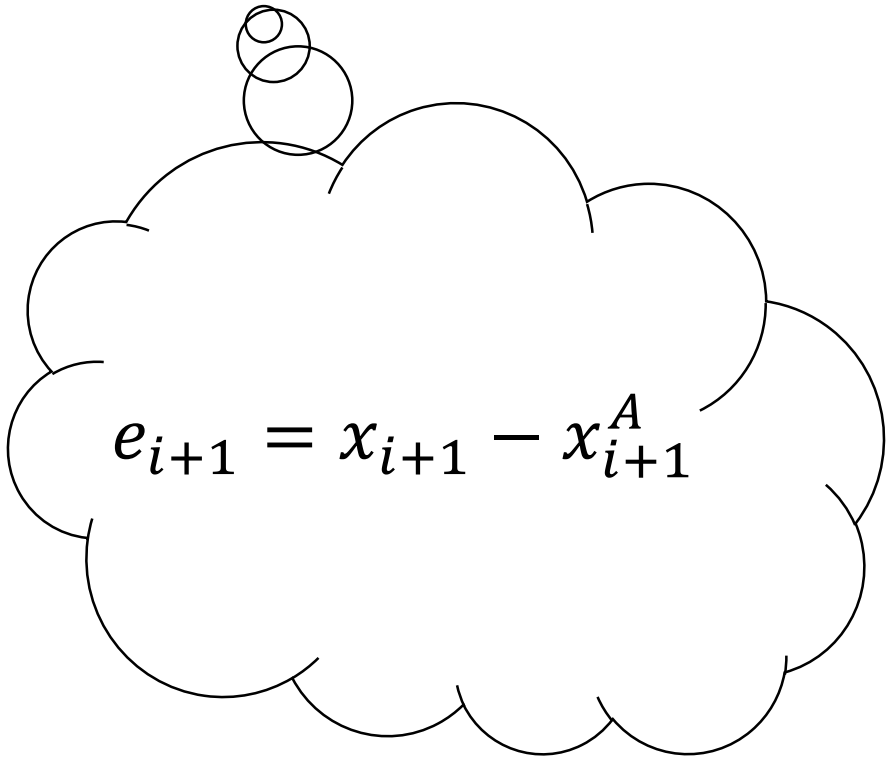
$$e_{i+1} = x_i + u_i^S \Delta t + \xi_i - K\left(x_i + u_i^S \Delta t + \xi_i + \eta_{i+1}\right) - \\ -(1-K)(x_i^A + u_i^S \Delta t)$$

Фильтр Калмана

$$e_{i+1} = (1 - K)\left(x_i + u_i^S \Delta t + \xi_i\right) - K\eta_{i+1} - (1 - K)(x_i^A + u_i^S \Delta t)$$

Фильтр Калмана

$$e_{i+1} = (1 - K)\left(x_i - x_i^A + \xi_i\right) - K\eta_{i+1}$$



Фильтр Калмана

$$e_{i+1} = (1 - K)(e_i + \xi_i) - K\eta_{i+1}$$

Фильтр Калмана

$$Ee_{i+1} = (1 - K)(Ee_i + E\xi_i) - KE\eta_{i+1}$$

Фильтр Калмана

$$Ee_{i+1}^2 = \left((1-K)(Ee_i + E\xi_i) - KE\eta_{i+1}\right)^2$$

Фильтр Калмана

$$Ee_{i+1}^2 = (1-K)^2(Ee_i + E\xi_i)^2 - 2(1-K)(Ee_i + E\xi_i)KE\eta_{i+1} + K^2E\eta_{i+1}^2$$

Фильтр Калмана

$$Ee_{i+1}^2 = (1-K)^2(Ee_i + E\xi_i)^2 - 2(1-K)(Ee_i + E\xi_i)KE\eta_{i+1} + K^2E\eta_{i+1}^2$$

Фильтр Калмана

$$Ee_{i+1}^2 = (1-K)^2(Ee_i + E\xi_i)^2 + K^2 E\eta_{i+1}^2$$

Фильтр Калмана

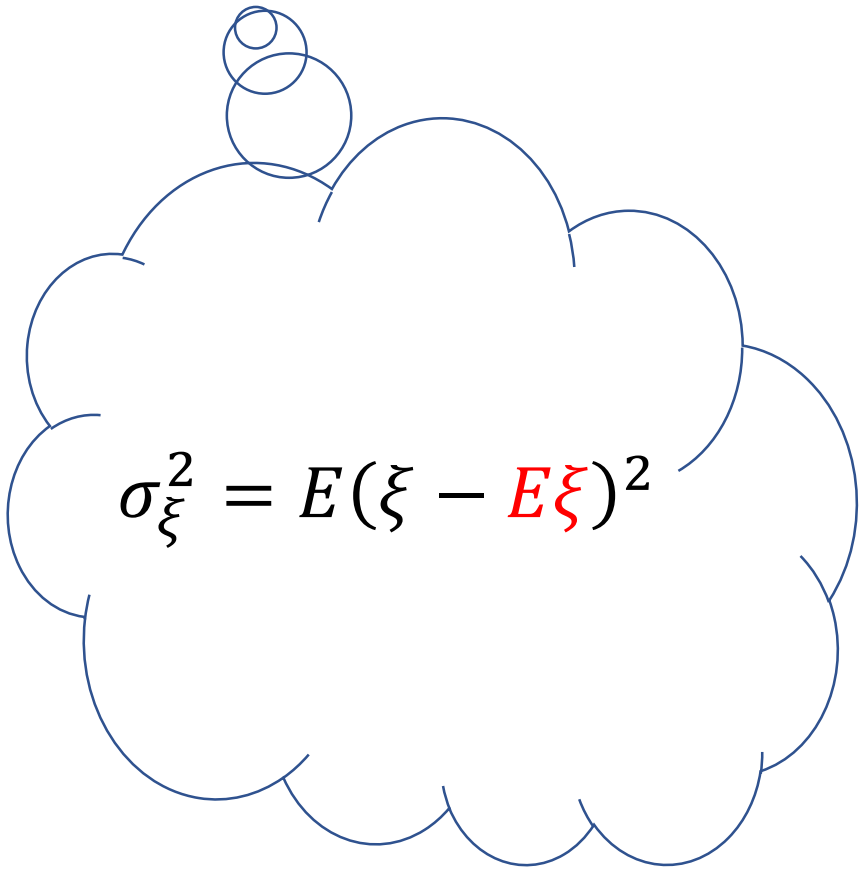
$$Ee_{i+1}^2 = (1-K)^2\left(Ee_i^2 + 2Ee_iE\xi_i + E\xi_i^2\right) + K^2E\eta_{i+1}^2$$

Фильтр Калмана

$$Ee_{i+1}^2 = (1-K)^2\left(Ee_i^2 + E\xi_i^2\right) + K^2 E\eta_{i+1}^2$$

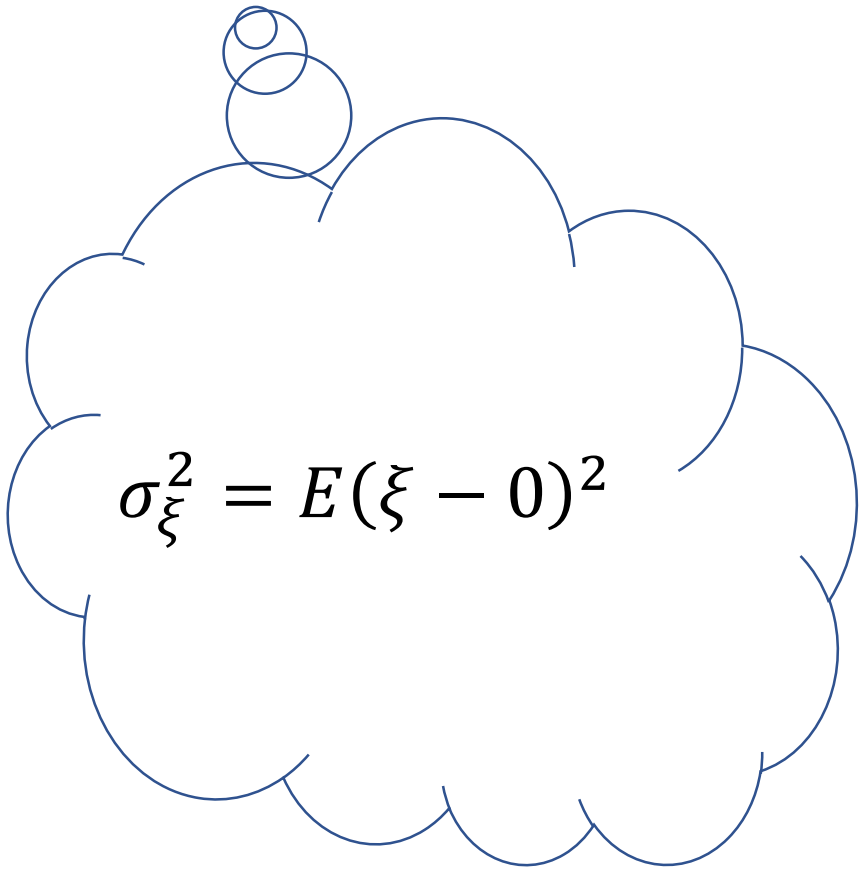
Фильтр Калмана

$$Ee_{i+1}^2 = (1-K)^2\left(Ee_i^2 + E\xi_i^2\right) + K^2 E\eta_{i+1}^2$$



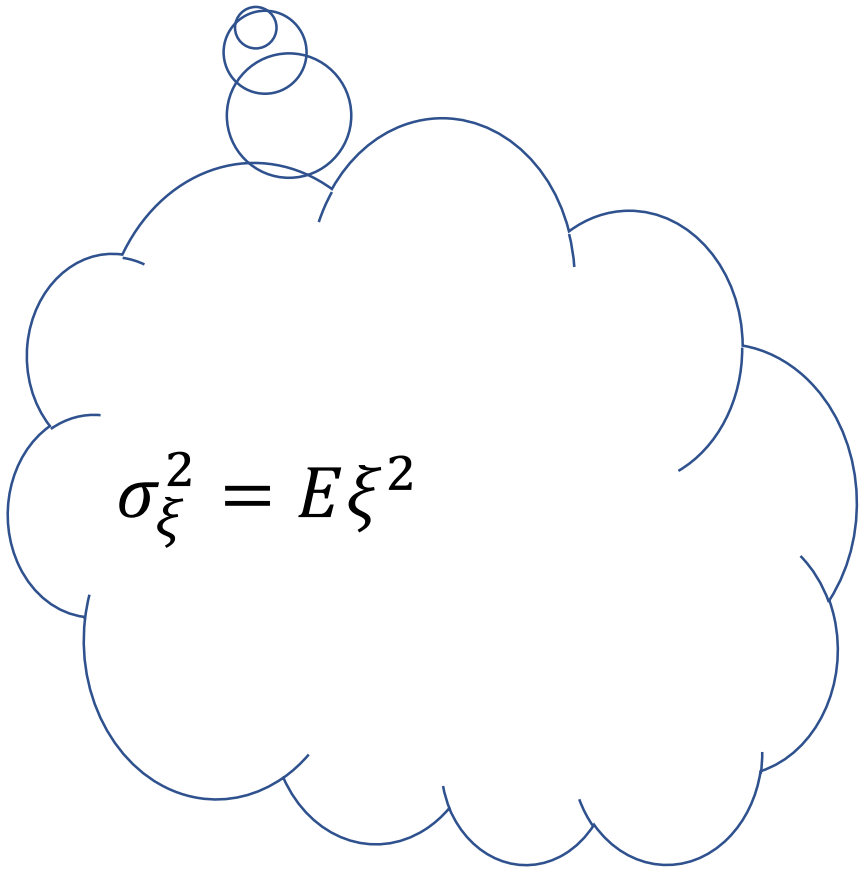
Фильтр Калмана

$$Ee_{i+1}^2 = (1-K)^2\left(Ee_i^2 + E\xi_i^2\right) + K^2 E\eta_{i+1}^2$$



Фильтр Калмана

$$Ee_{i+1}^2 = (1-K)^2\left(Ee_i^2 + E\xi_i^2\right) + K^2 E\eta_{i+1}^2$$



Фильтр Калмана

$$Ee_{i+1}^2 = (1-K)^2\left(Ee_i^2 + \sigma_\xi^2\right) + K^2\sigma_\eta^2$$

$$a = Ee_i^2 + \sigma_\xi^2$$

$$b = \sigma_\eta^2$$

Фильтр Калмана

$$Ee_{i+1}^2 = (1-K)^2 a + K^2 b$$

$$a = Ee_i^2 + \sigma_\xi^2$$

$$b = \sigma_\eta^2$$

Фильтр Калмана

$$Ee_{i+1}^2 = a - 2Ka + K^2a + K^2b$$

$$a = Ee_i^2 + \sigma_\xi^2$$

$$b = \sigma_\eta^2$$

Фильтр Калмана

$$Ee_{i+1}^2 = (a+b)K^2 - 2aK + a$$

$$a = Ee_i^2 + \sigma_\xi^2$$

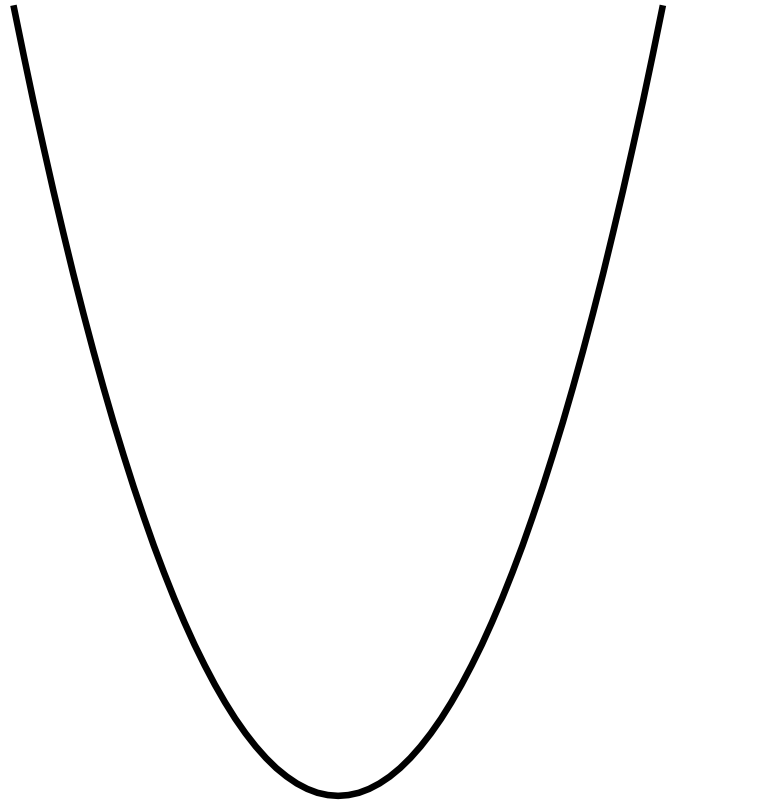
$$b = \sigma_\eta^2$$

Фильтр Калмана

$$Ee_{i+1}^2 = (a+b)K^2 - 2aK + a$$

$$a = Ee_i^2 + \sigma_\xi^2$$

$$b = \sigma_\eta^2$$



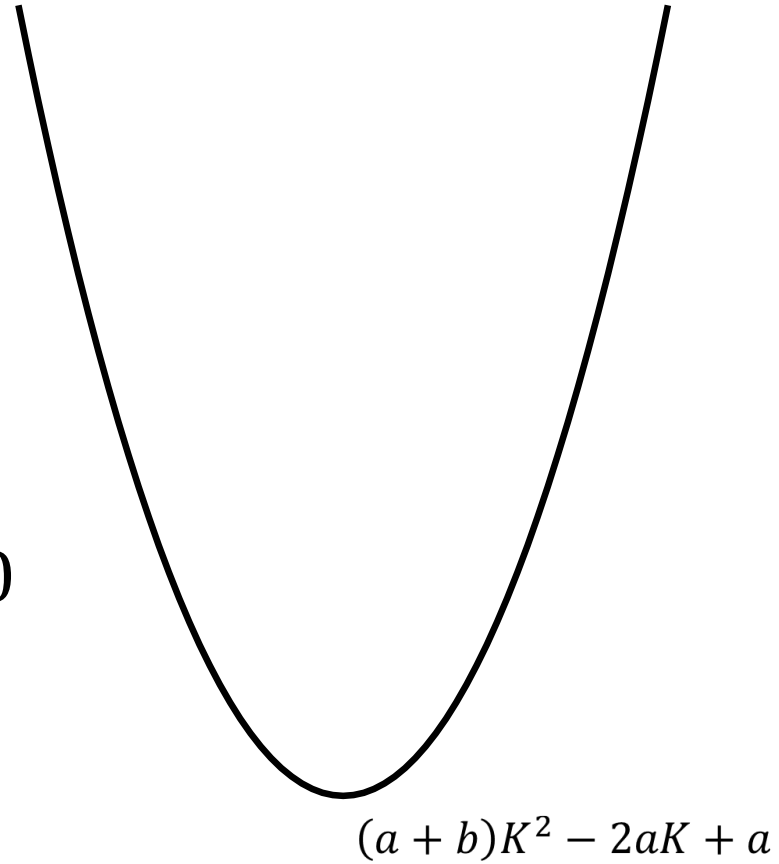
Фильтр Калмана

$$Ee_{i+1}^2 = (a+b)K^2 - 2aK + a$$

$$a = Ee_i^2 + \sigma_\xi^2$$

$$b = \sigma_\eta^2$$

$$\left((a+b)K^2 - 2aK + a\right)' = 0$$



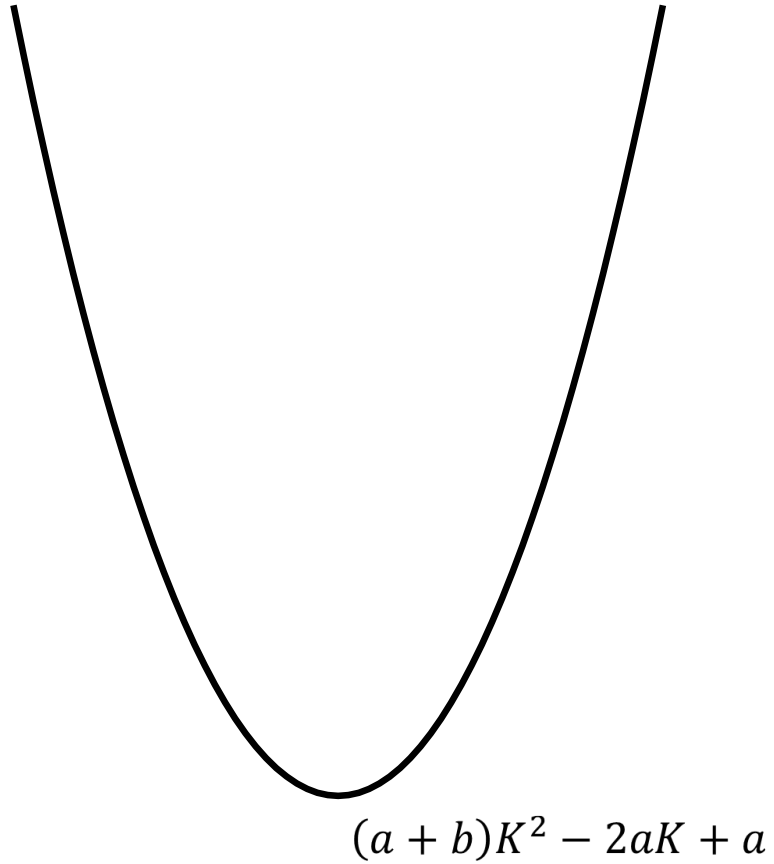
Фильтр Калмана

$$Ee_{i+1}^2 = (a+b)K^2 - 2aK + a$$

$$a = Ee_i^2 + \sigma_\xi^2$$

$$b = \sigma_\eta^2$$

$$2(a+b)K - 2a = 0$$



Фильтр Калмана

$$Ee_{i+1}^2 = (a+b)K^2 - 2aK + a$$

$$a = Ee_i^2 + \sigma_\xi^2$$

$$b = \sigma_\eta^2$$

$$K = \frac{a}{a+b}$$

Фильтр Калмана

$$Ee_{i+1}^2 = (a+b)K^2 - 2aK + a$$

$$a = Ee_i^2 + \sigma_\xi^2$$

$$b = \sigma_\eta^2$$

$$K = \frac{Ee_i^2 + \sigma_\xi^2}{Ee_i^2 + \sigma_\xi^2 + \sigma_\eta^2}$$

Фильтр Калмана

$$E e_{i+1}^2 = (a+b)K^2 - 2aK + a$$

$$K = \frac{a}{a+b}$$

Фильтр Калмана

$$Ee_{i+1}^2 = \frac{a^2(a+b)}{(a+b)^2} - 2\frac{a^2}{a+b} + a$$

Фильтр Калмана

$$Ee_{i+1}^2 = \frac{a^2}{a+b} - 2\frac{a^2}{a+b} + \frac{a(a+b)}{a+b}$$

Фильтр Калмана

$$Ee_{i+1}^2 = \frac{a^2 - 2a^2 + a^2 + ab}{a + b}$$

Фильтр Калмана

$$Ee_{i+1}^2 = \frac{ab}{a+b}$$

$$a = Ee_i^2 + \sigma_\xi^2$$

$$b = \sigma_\eta^2$$

Фильтр Калмана

$$Ee_{i+1}^2 = \frac{\left(Ee_i^2 + \sigma_\xi^2\right)\sigma_\eta^2}{Ee_i^2 + \sigma_\xi^2 + \sigma_\eta^2}$$

Фильтр Калмана

$$Ee_{i+1}^2 = \frac{\left(Ee_i^2 + \sigma_\xi^2\right)\sigma_\eta^2}{Ee_i^2 + \sigma_\xi^2 + \sigma_\eta^2}$$

$$Ee_0^2 = \sigma_\eta^2$$

Фильтр Калмана

$$Ee_{i+1}^2 = \frac{\left(Ee_i^2 + \sigma_\xi^2\right)\sigma_\eta^2}{Ee_i^2 + \sigma_\xi^2 + \sigma_\eta^2}$$

$$Ee_0^2 = \sigma_\eta^2$$

$$K = \frac{Ee_i^2 + \sigma_\xi^2}{Ee_i^2 + \sigma_\xi^2 + \sigma_\eta^2}$$

Фильтр Калмана

$$Ee_{i+1}^2 = \frac{\left(Ee_i^2 + \sigma_\xi^2\right)\sigma_\eta^2}{Ee_i^2 + \sigma_\xi^2 + \sigma_\eta^2}$$

$$Ee_0^2 = \sigma_\eta^2$$

$$K = \frac{Ee_i^2 + \sigma_\xi^2}{Ee_i^2 + \sigma_\xi^2 + \sigma_\eta^2}$$

$$x_{i+1}^A = Kx_{i+1}^S + (1 - K)(x_i^A + u_i^S \Delta t)$$

Фильтр Калмана

$$Ee_{i+1}^2 = \frac{\left(Ee_i^2 + \sigma_\xi^2\right)\sigma_\eta^2}{Ee_i^2 + \sigma_\xi^2 + \sigma_\eta^2}$$

$$Ee_0^2 = \sigma_\eta^2$$

$$K = \frac{Ee_i^2 + \sigma_\xi^2}{Ee_i^2 + \sigma_\xi^2 + \sigma_\eta^2}$$

$$x_{i+1}^A = Kx_{i+1}^S + (1 - K)(x_i^A + u_i^S \Delta t)$$

$$x_0^A = x_0^S$$

Фильтр Калмана

```
let project latitude longitude speed heading =
    let cartesianAngleFromHeading =
        let flip angle = 360.0 - angle
        let rotateClockwise90 angle = (270.0 + angle) % 360.0

        flip >> rotateClockwise90

    let kilometersPerHour metersPerSecond = 3.6 * metersPerSecond

    let angle = radian (cartesianAngleFromHeading heading)
    let velocity = kilometersPerHour speed
    let velocityLongitude = velocity * cos angle
    let velocityLatitude = velocity * sin angle
    let kmpLongitude = distance latitude longitude latitude (longitude + 1.0)
    let kmpLatitude = distance latitude longitude (latitude + 1.0) longitude

    (velocityLatitude / kmpLatitude, velocityLongitude / kmpLongitude)
```

Фильтр Калмана

```
let project latitude longitude speed heading =
    let cartesianAngleFromHeading =
        let flip angle = 360.0 - angle
        let rotateClockwise90 angle = (270.0 + angle) % 360.0

        flip >> rotateClockwise90

    let kilometersPerHour metersPerSecond = 3.6 * metersPerSecond

    let angle = radian (cartesianAngleFromHeading heading)
    let velocity = kilometersPerHour speed
    let velocityLongitude = velocity * cos angle
    let velocityLatitude = velocity * sin angle
    let kmpLongitude = distance latitude longitude latitude (longitude + 1.0)
    let kmpLatitude = distance latitude longitude (latitude + 1.0) longitude

    (velocityLatitude / kmpLatitude, velocityLongitude / kmpLongitude)
```

Фильтр Калмана

```
let project latitude longitude speed heading =
    let cartesianAngleFromHeading =
        let flip angle = 360.0 - angle
        let rotateClockwise90 angle = (270.0 + angle) % 360.0

        flip >> rotateClockwise90

    let kilometersPerHour metersPerSecond = 3.6 * metersPerSecond

    let angle = radian (cartesianAngleFromHeading heading)
    let velocity = kilometersPerHour speed
    let velocityLongitude = velocity * cos angle
    let velocityLatitude = velocity * sin angle
    let kmpLongitude = distance latitude longitude latitude (longitude + 1.0)
    let kmpLatitude = distance latitude longitude (latitude + 1.0) longitude

    (velocityLatitude / kmpLatitude, velocityLongitude / kmpLongitude)
```

Фильтр Калмана

```
let project latitude longitude speed heading =
    let cartesianAngleFromHeading =
        let flip angle = 360.0 - angle
        let rotateClockwise90 angle = (270.0 + angle) % 360.0

        flip >> rotateClockwise90

    let kilometersPerHour metersPerSecond = 3.6 * metersPerSecond

    let angle = radian (cartesianAngleFromHeading heading)
    let velocity = kilometersPerHour speed
    let velocityLongitude = velocity * cos angle
    let velocityLatitude = velocity * sin angle
    let kmpLongitude = distance latitude longitude latitude (longitude + 1.0)
    let kmpLatitude = distance latitude longitude (latitude + 1.0) longitude

    (velocityLatitude / kmpLatitude, velocityLongitude / kmpLongitude)
```

Фильтр Калмана

```
let project latitude longitude speed heading =
    let cartesianAngleFromHeading =
        let flip angle = 360.0 - angle
        let rotateClockwise90 angle = (270.0 + angle) % 360.0

        flip >> rotateClockwise90

    let kilometersPerHour metersPerSecond = 3.6 * metersPerSecond

    let angle = radian (cartesianAngleFromHeading heading)
    let velocity = kilometersPerHour speed
    let velocityLongitude = velocity * cos angle
    let velocityLatitude = velocity * sin angle
    let kmpLongitude = distance latitude longitude latitude (longitude + 1.0)
    let kmpLatitude = distance latitude longitude (latitude + 1.0) longitude

    (velocityLatitude / kmpLatitude, velocityLongitude / kmpLongitude)
```

Фильтр Калмана

```
let smoothByKalman σξ ση (points: SensorItem list) =
    let iterateKalman ... = ...
    let toSensorItem ... = ...

    match points with
    | [] -> []
    | p1::ps -> let base = (p1.Latitude, p1.Longitude, p1.Timestamp, ση ** 2.0)
                let filtered = ps
                               |> List.scan iterateKalman base
                               |> List.map toSensorItem

                SensorItem(p1.Latitude, p1.Longitude, 0.0, 0.0, p1.Timestamp)::filtered
```

Фильтр Калмана

```
let toSensorItem (latitude, longitude, timestamp, _) =
    SensorItem(latitude, longitude, 0.0, 0.0, timestamp)
```

Фильтр Калмана

```
let iterateKalman (latitude, longitude, timestamp, errorSquare) p2 =
    let a = errorSquare + σξ ** 2.0
    let b = ση ** 2.0
    let nextErrorSquare = (a * b)/(a + b)
    let K = nextErrorSquare/b
    let Δt = (p2.Timestamp - timestamp).TotalHours
    let (vLatitude, vLongitude) = project latitude longitude p2.Speed p2.Heading
    let nextLatitude = K * p2.Latitude + (1.0 - K) * (latitude + vLatitude * Δt)
    let nextLongitude = K * p2.Longitude + (1.0 - K) * (longitude + vLongitude * Δt)

    (nextLatitude, nextLongitude, p2.Timestamp, nextErrorSquare)
```

Фильтр Калмана

```
let iterateKalman (latitude, longitude, timestamp, errorSquare) p2 =
    let a = errorSquare + σξ ** 2.0
    let b = ση ** 2.0
    let nextErrorSquare = (a * b)/(a + b)
    let K = nextErrorSquare/b
    let Δt = (p2.Timestamp - timestamp).TotalHours
    let (vLatitude, vLongitude) = project latitude longitude p2.Speed p2.Heading
    let nextLatitude = K * p2.Latitude + (1.0 - K) * (latitude + vLatitude * Δt)
    let nextLongitude = K * p2.Longitude + (1.0 - K) * (longitude + vLongitude * Δt)

    (nextLatitude, nextLongitude, p2.Timestamp, nextErrorSquare)
```

Фильтр Калмана

```
let iterateKalman (latitude, longitude, timestamp, errorSquare) p2 =
    let a = errorSquare + σξ ** 2.0
    let b = ση ** 2.0
    let nextErrorSquare = (a * b)/(a + b)
    let K = nextErrorSquare/b
    let Δt = (p2.Timestamp - timestamp).TotalHours
    let (vLatitude, vLongitude) = project latitude longitude p2.Speed p2.Heading
    let nextLatitude = K * p2.Latitude + (1.0 - K) * (latitude + vLatitude * Δt)
    let nextLongitude = K * p2.Longitude + (1.0 - K) * (longitude + vLongitude * Δt)

    (nextLatitude, nextLongitude, p2.Timestamp, nextErrorSquare)
```

Фильтр Калмана

```
let iterateKalman (latitude, longitude, timestamp, errorSquare) p2 =
    let a = errorSquare + σξ ** 2.0
    let b = ση ** 2.0
    let nextErrorSquare = (a * b)/(a + b)
    let K = nextErrorSquare/b
    let Δt = (p2.Timestamp - timestamp).TotalHours
    let (vLatitude, vLongitude) = project latitude longitude p2.Speed p2.Heading
    let nextLatitude = K * p2.Latitude + (1.0 - K) * (latitude + vLatitude * Δt)
    let nextLongitude = K * p2.Longitude + (1.0 - K) * (longitude + vLongitude * Δt)

    (nextLatitude, nextLongitude, p2.Timestamp, nextErrorSquare)
```

Фильтр Калмана

```
let iterateKalman (latitude, longitude, timestamp, errorSquare) p2 =
    let a = errorSquare + σξ ** 2.0
    let b = ση ** 2.0
    let nextErrorSquare = (a * b)/(a + b)
    let K = nextErrorSquare/b
    let Δt = (p2.Timestamp - timestamp).TotalHours
    let (vLatitude, vLongitude) = project latitude longitude p2.Speed p2.Heading
    let nextLatitude = K * p2.Latitude + (1.0 - K) * (latitude + vLatitude * Δt)
    let nextLongitude = K * p2.Longitude + (1.0 - K) * (longitude + vLongitude * Δt)

    (nextLatitude, nextLongitude, p2.Timestamp, nextErrorSquare)
```

Фильтр Калмана

```
[<Fact>]
let ``smoothBySimplifiedKalman - with points - filters coordinates`` () =
    let source = [SensorItem(45.0, 0.0, 0.0, 0.0,
                        DateTimeOffset.Parse("2018-12-07T16:38:14+03:00"));
                  SensorItem(45.5, 0.5, 0.0, 0.0,
                        DateTimeOffset.Parse("2018-12-07T16:38:15+03:00"));
                  SensorItem(45.0, 1.0, 0.0, 0.0,
                        DateTimeOffset.Parse("2018-12-07T16:38:16+03:00"))]

    let actual = List.toArray (smoothBySimplifiedKalman 1.0 1.0 source)

    Assert.Equal(45.1111111111111, actual.[1].Latitude, 1)
    Assert.Equal(0.111111111111111, actual.[1].Longitude, 1)
    Assert.Equal(45.0836111111111, actual.[2].Latitude, 1)
    Assert.Equal(0.331111111111111, actual.[2].Longitude, 1)
```

Интеграция с C#

```
[<Struct>]
type Location(latitude: float, longitude: float, timestamp: DateTimeOffset) =
    member __.Latitude = latitude
    member __.Longitude = longitude
    member __.Timestamp = timestamp

[<Struct>]
type DirectedLocation(latitude: float, longitude: float, speed: float,
                      heading: float, timestamp: DateTimeOffset) =
    member __.Latitude = latitude
    member __.Longitude = longitude
    member __.Timestamp = timestamp
    member __.Speed = speed
    member __.Heading = heading
```

Интеграция с C#

```
/// <summary>
/// Implements a few methods to fix bad GPS data.
/// </summary>
type GpsTrackFilter() =
    let mutable zeroSpeedDrift = 7.99
    let mutable outlineSpeed = 110.0
    let mutable modelPrecision = 2.13
    let mutable sensorPrecision = 0.77


    /// <summary>
    /// Gets or sets the minimal valid velocity.
    /// </summary>
    member __.ZeroSpeedDrift with get () = zeroSpeedDrift
                             and set value = zeroSpeedDrift <- value
```

Интеграция с C#

```
let toSensorItem x =
    SensorItem(x.Latitude, x.Longitude, x.Speed, x.Heading, x.Timestamp)

/// <summary>Fixes a GPS track.// <summary>
/// <param name="points">Source GPS track with possible bad data.</param>
/// <returns>Fixed track.</returns>
member __.Filter(points: seq<DirectedLocation>): IReadOnlyList<Location> =
    points
 |> Seq.map toSensorItem
 |> List.ofSeq
 |> removeZeroOrNegativeTimespans
 |> removeZeroSpeedDrift zeroSpeedDrift
 |> removeOutlineSpeedValues outlineSpeed
 |> smoothByKalman modelPrecision sensorPrecision
 |> List.map (fun x -> Location(x.Latitude, x.Longitude, x.Timestamp))
 :> IReadOnlyList<Location>
```

Интеграция с C#

```
let toSensorItem x =
    SensorItem(x.Latitude, x.Longitude, x.Speed, x.Heading, x.Timestamp)

/// <summary>Fixes a GPS track.// <summary>
/// <param name="points">Source GPS track with possible bad data.</param>
/// <returns>Fixed track.</returns>
member __.Filter(points: seq<DirectedLocation>): IReadOnlyList<Location> =
    points
 |> Seq.map toSensorItem
 |> List.ofSeq
 |> removeZeroOrNegativeTimespans
 |> removeZeroSpeedDrift zeroSpeedDrift
 |> removeOutlineSpeedValues outlineSpeed
 |> smoothByKalman modelPrecision sensorPrecision
 |> List.map (fun x -> Location(x.Latitude, x.Longitude, x.Timestamp))
 :> IReadOnlyList<Location>
```

Интеграция с C#

```
let toSensorItem x =
    SensorItem(x.Latitude, x.Longitude, x.Speed, x.Heading, x.Timestamp)

/// <summary>Fixes a GPS track.// <summary>
/// <param name="points">Source GPS track with possible bad data.</param>
/// <returns>Fixed track.</returns>
member __.Filter(points: seq<DirectedLocation>): IReadOnlyList<Location> =
    points
 |> Seq.map toSensorItem
 |> List.ofSeq
 |> removeZeroOrNegativeTimespans
 |> removeZeroSpeedDrift zeroSpeedDrift
 |> removeOutlineSpeedValues outlineSpeed
 |> smoothByKalman modelPrecision sensorPrecision
 |> List.map (fun x -> Location(x.Latitude, x.Longitude, x.Timestamp))
 :> IReadOnlyList<Location>
```

Интеграция с C#

```
var filter = new GpsTrackFilter();
var filteredLocations = filter.Filter(directedLocations);

var fileterdGeoJson = filteredLocations.ToGeoJson();
```

- Московский Клуб Программистов
  http://prog.msk.ru/
- Статья
  http://markshevchenko.pro/articles/fsharp-gps-tracks-filtration/
- GitHub
  https://github.com/binateq/gps-track-filter
- NuGet
  https://www.nuget.org/packages/Binateq.GpsTrackFilter/