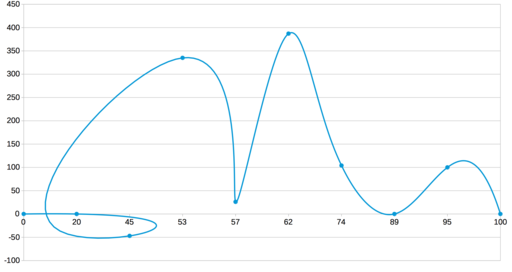
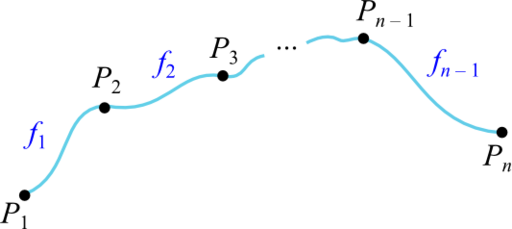
Как построить график по *n* точкам? Самое простое — отметить их маркерами на координатной сетке. Однако для наглядности их хочется соединить, чтобы получить легко читаемую линию. Соединять точки проще всего отрезками прямых. Но график-ломаная читается довольно тяжело: взгляд цепляется за углы, а не скользит вдоль линии. Да и выглядят изломы не очень красиво. Получается, что кроме ломаных нужно уметь строить и кривые. Однако тут нужно быть осторожным, чтобы не получилось вот такого:  


Немного матчасти

Восстановление промежуточных значений функции, которая в данном случае задана таблично в виде точек *P*1 ... *Pn*, называется интерполяцией. Есть множество способов интерполяции, но все они могут быть сведены к тому, что надо найти *n* – 1 функцию для расчёта промежуточных точек на соответствующих сегментах. При этом заданные точки обязательно должны быть вычислимы через соответствующие функции. На основе этого и может быть построен график:  
  
  
Функции *fi* могут быть самыми разными, но чаще всего используют полиномы некоторой степени. В этом случае итоговая интерполирующая функция (кусочно заданная на промежутках, ограниченных точками *Pi*) называется [сплайном](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%BB%D0%B0%D0%B9%D0%BD).  
  
В разных инструментах для построения графиков — редакторах и библиотеках — задача «красивой интерполяции» решена по-разному. В конце статьи будет небольшой обзор существующих вариантов. Почему в конце? Чтобы после ряда приведённых выкладок и размышлений можно было поугадывать, кто из «серьёзных ребят» какие методы использует.  
Ставим опыты

Самый простой пример — линейная интерполяция, в которой используются полиномы первой степени, а в итоге получается ломаная, соединяющая заданные точки.  
Давайте добавим немного конкретики. Вот набор точек (взяты *почти* с потолка):

0 0

20 0

45 -47

53 335

57 26

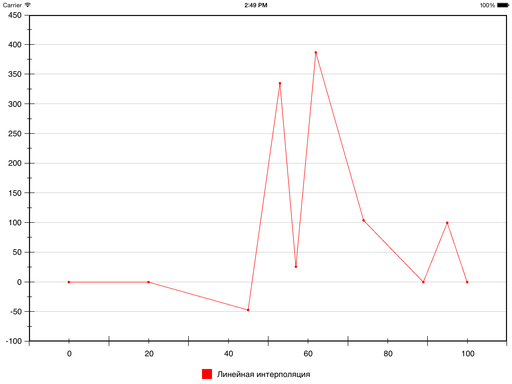
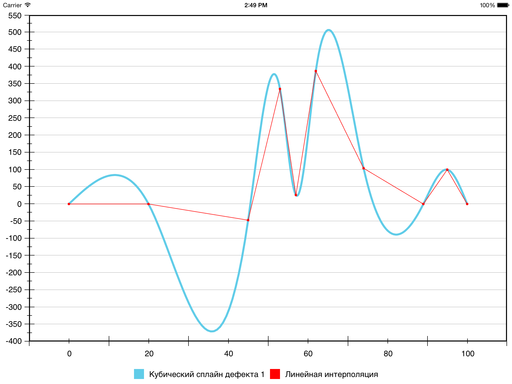
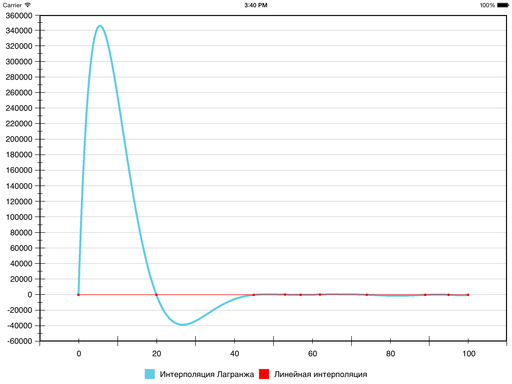
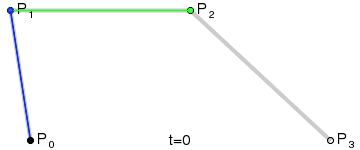
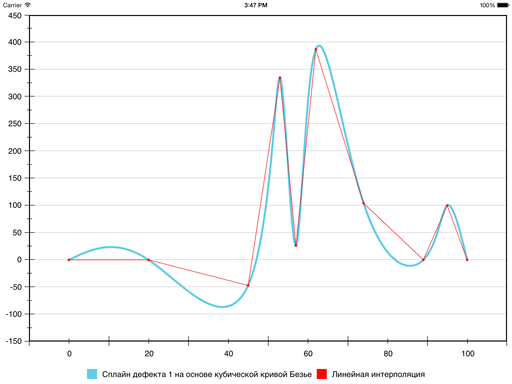
62 387

74 104

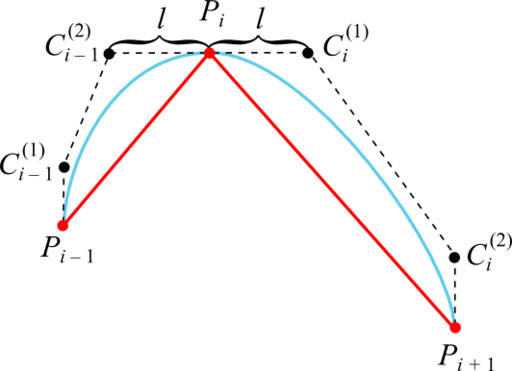
89 0

95 100

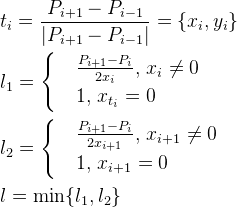
100 0

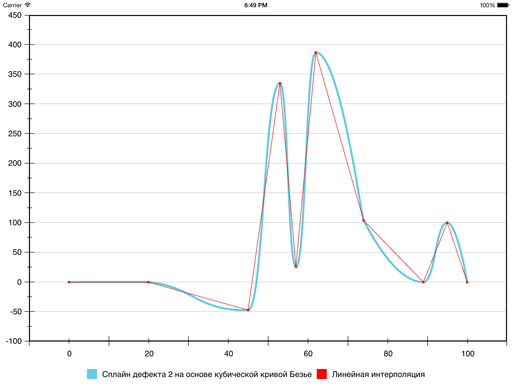
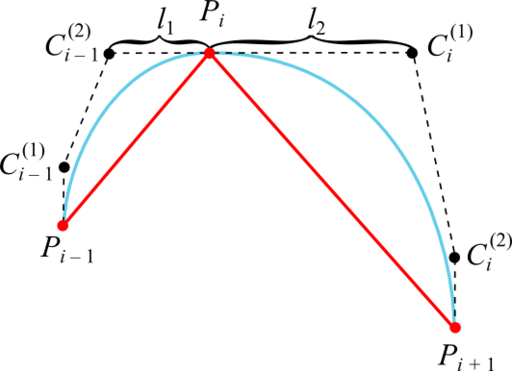
Результат линейной интерполяции этих точек выглядит так:  
  
  
Однако, как отмечалось выше, иногда хочется получить в итоге гладкую кривую.  
  
Что есть гладкость? Бытовой ответ: отсутствие острых углов. Математический: непрерывность производных. При этом в математике гладкость имеет порядок, равный номеру последней непрерывной производной, и область, на которой эта непрерывность сохраняется. То есть, если функция имеет гладкость порядка 1 на отрезке [*a*; *b*], это означает, что на [*a*; *b*] она имеет непрерывную первую производную, а вот вторая производная уже терпит разрыв в каких-то точках.   
У сплайна в контексте гладкости есть понятие дефекта. Дефект сплайна — это разность между его степенью и его гладкостью. Степень сплайна — это максимальная степень использованных в нём полиномов.  
Важно отметить, что «опасными» точками у сплайна (в которых может нарушиться гладкость) являются как раз *Pi*, то есть точки сочленения сегментов, в которых происходит переход от одного полинома к другому. Все остальные точки «безопасны», ведь у полинома на области его определения нет проблем с непрерывностью производных.  
Чтобы добиться гладкой интерполяции, нужно повысить степень полиномов и подобрать их коэффициенты так, чтобы в граничных точках сохранялась непрерывность производных.  
  
Традиционно для решения такой задачи используют полиномы третьей степени и добиваются непрерывности первой и второй производной. То, что получается, называют [кубическим сплайном дефекта 1](https://ru.wikipedia.org/wiki/%D0%9A%D1%83%D0%B1%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%BF%D0%BB%D0%B0%D0%B9%D0%BD). Вот как он выглядит для наших данных:  
  
  
Кривая, действительно, гладкая. Но если предположить, что это график некоторого процесса или явления, который нужно показать заинтересованному лицу, то такой метод, скорее всего, не подходит. Проблема в ложных экстремумах. Появились они из-за слишком сильного искривления, которое было призвано обеспечить гладкость интерполяционной функции. Но зрителю такое поведение совсем не кстати, ведь он оказывается обманут относительно пиковых значений функции. А ради наглядной визуализации этих значений, собственно, всё и затевалось.  
Так что надо искать другие решения.  
  
Другое традиционное решение, кроме кубических сплайнов дефекта 1 — полиномы Лагранжа. Это полиномы степени *n* – 1, принимающие заданные значения в заданных точках. То есть членения на сегменты здесь не происходит, вся последовательность описывается одним полиномом.  
Но вот что получается:  
  
  
Гладкость, конечно, присутствует, но наглядность пострадала так сильно, что… пожалуй, стоит поискать другие методы. На некоторых наборах данных результат выходит нормальный, но в общем случае ошибка относительно линейной интерполяции (и, соответственно, ложные экстремумы) может получаться слишком большой — из-за того, что тут всего один полином на все сегменты.  
  
В компьютерной графике очень широко применяются [кривые Безье](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%B2%D0%B0%D1%8F_%D0%91%D0%B5%D0%B7%D1%8C%D0%B5), представленные полиномами *k*-й степени.  
Они не являются интерполирующими, так как из *k* + 1 точек, участвующих в построении, итоговая кривая проходит лишь через первую и последнюю. Остальные *k* – 1 точек играют роль своего рода «гравитационных центров», притягивающих к себе кривую.  
Вот пример кубической кривой Безье:  
  
  
Как это можно использовать для интерполяции? На основе этих кривых тоже можно построить сплайн. То есть на каждом сегменте сплайна будет своя кривая Безье *k*-й степени (кстати, *k* = 1 даёт линейную интерполяцию). И вопрос только в том, какое *k* взять и как найти *k* – 1 промежуточную точку.  
Здесь бесконечно много вариантов (поскольку *k* ничем не ограничено), однако мы рассмотрим классический: *k* = 3.  
Чтобы итоговая кривая была гладкой, нужно добиться дефекта 1 для составляемого сплайна, то есть сохранения непрерывности первой и второй производных в точках сочленения сегментов (*Pi*), как это делается в классическом варианте кубического сплайна.  
Решение этой задачи подробно (с исходным кодом) рассмотрено [здесь](https://www.particleincell.com/2012/bezier-splines/).  
Вот что получится на нашем тестовом наборе:  
  
  
Стало лучше: ложные экстремумы всё ещё есть, но хотя бы не так сильно отличаются от реальных.

Думаем и экспериментируем

Можно попробовать ослабить условие гладкости: потребовать дефект 2, а не 1, то есть сохранить непрерывность одной только первой производной.  
Достаточное условие достижения дефекта 2 в том, что промежуточные контрольные точки кубической кривой Безье, смежные с заданной точкой интерполируемой последовательности, лежат с этой точкой на одной прямой и на одинаковом расстоянии:  
  
  
В качестве прямых, на которых лежат точки *Ci*– 1(2), *Pi и Ci*(1), целесообразно взять касательные к графику интерполируемой функции в точках *Pi*. Это гарантирует отсутствие ложных экстремумов, так как кривая Безье оказывается ограниченной ломаной, построенной на её контрольных точках (если эта ломаная не имеет самопересечений).  
  
Методом проб и ошибок эвристика для расчёта расстояния от точки интерполируемой последовательности до промежуточной контрольной получилась такой:

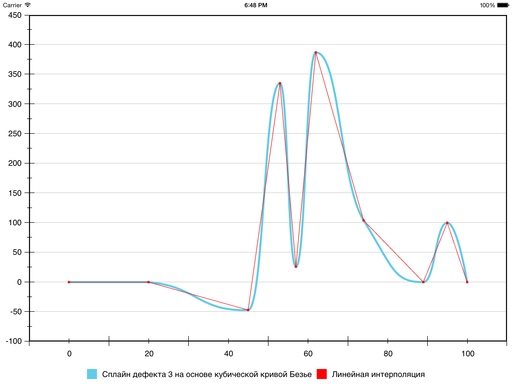
Эвристика 1



Первая и последняя промежуточные контрольные точки равны первой и последней точке графика соответственно (точки *C*1(1) и *Cn*– 1(2) совпадают с точками *P*1 и *Pn* соответственно).  
В этом случае получается вот такая кривая:  
  
  
Как видно, ложных экстремумов уже нет. Однако если сравнивать с линейной интерполяцией, местами ошибка очень большая. Можно сделать её ещё меньше, но тут в ход пойдут ещё более хитрые эвристики.  
  
К текущему варианту мы пришли, уменьшив гладкость на один порядок. Можно сделать это ещё раз: пусть сплайн будет иметь дефект 3. По факту, тем самым формально функция не будет гладкой вообще: даже первая производная может терпеть разрывы. Но если рвать её аккуратно, визуально ничего страшного не произойдёт.  
Отказываемся от требования равенства расстояний от точки *Pi* до точек *Ci*– 1(2) и C*i*(1), но при этом сохраняем их все лежащими на одной прямой:  
  
  
Эвристика для вычисления расстояний будет такой:

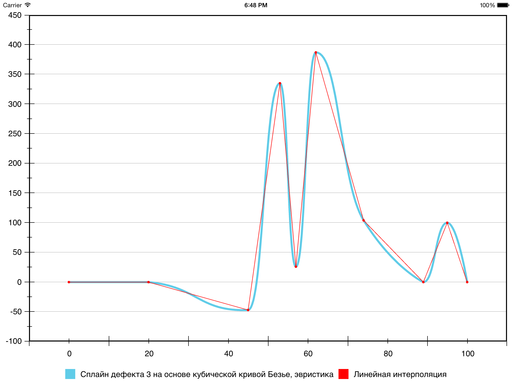
Эвристика 2

Расчёт *l*1 и *l*2 такой же, как в «эвристике 1».  
При этом, однако, стоит ещё проверять, не совпали ли точки *Pi* и *Pi*+ 1 по ординате, и, если совпали, полагать *l*1 = *l*2 = 0. Это защитит от «вспухания» графика на плоских отрезках (что тоже немаловажно с точки зрения правдивого отображения данных).

Результат получается такой:  
  
  
В результате на шестом сегменте ошибка уменьшилась, а на седьмом — увеличилась: кривизна у Безье на нём оказалась больше, чем хотелось бы. Исправить ситуацию можно, принудительно уменьшив кривизну и тем самым «прижав» Безье ближе к отрезку прямой, которая соединяет граничные точки сегмента. Для этого используется следующая эвристика:

Эвристика 3

Если абсцисса точки пересечения касательных в точках *Pi*(*xi*, *yi*) и *Pi*+ 1(*xi*+ 1, *yi*+ 1) лежит в отрезке [*xi*; *xi*+ 1], то *l*1 либо *l*2 полагаем равным нулю. В том случае, если касательная в точке *Pi* направлена вверх, нулю полагаем максимальное из *l*1 и *l*2, если вниз — минимальное.

Результат следующий:  
  
  
На этом было принято решение признать цель достигнутой.  
Может быть, кому-то пригодится [код](https://gist.github.com/anonymous/06f4104d93f6cef6f341).