**МЕТОДИКА**

**ИЗНАЧАЛЬНЫЕ УСЛОВИЯ**

Изначальные технические(физические) условия: Существует сервер, имеющий память ??? ГБ и имеющий процессор марки - ???.

Планируется создание *микросервисного приложения*, состоящего из X = const сервисов, у каждого сервиса (пода) будет Ni, где i = [1, K] количество реплик (копий) сервиса, где K – заранее оговоренное некоторое число. Каждой реплике (копии) отводится некоторое пространство памяти. Ключевой особенностью сбора метрик заключается в анализе относительных, а не абсолютных значений. Введём новые термины:

*Нагрузочными тестами* будем называть упорядоченную конечную последовательность тестов L1 < …< Ln продолжительностью tсекунд, где n и t – заранее оговоренные числа.

*Статичными данными* будем называть данные, собранные экспериментальным путём с пары «система-сервер».

*Практическими данными* будем называть данные, полученные экспериментальным путём с пары «система-сервер» под действием нагрузочных тестов.

В предварительном этапе проводится измерение статичных данных:

* Статичная матрица задержки сети Mстат.сеть - симметричная матрица размерности N\*N, элементы отображают задержку сети между сервисами, по диагонали ставятся 0 или 1, так как задержка внутри отдельного сервиса минимальна.
* Относительная статичная загрузка памяти MEMстат (вектор значений размерности N, с элементами от 0 до 1);
* Относительная статичная загрузка процессора CPUстат (вектор значений размерности N, с элементами от 0 до 1);
* Абсолютное статичное энергопотребление системы Eстат (Квт\*ч).

Для системы в целом, и для отдельных сервисов (именно сервисов(подов), а не реплик), проводятся нагрузочные тесты, в результате которых логируются следующие величины:

* Нагрузочные тесты Li , где i = 1…n (вектор значений размерности N, единица измерения – запросы в секунду);
* Относительная загрузка памяти MEMi (вектор значений размерности N, с элементами от 0 до 1);
* Относительная загрузка процессора CPUi(вектор значений размерности N, с элементами от 0 до 1);
* Абсолютное энергопотребление системы Ei (Квт\*ч) //спорная величина;
* Матрица задержки сети Mсеть.i (симметричная матрица размерности N\*N, элементы отображают задержку сети между сервисами) P.S. по логике вещей должна быть такой же, как и статичная матрица задержки.
* Матрица времени отклика между отправкой запроса и получением ответа – Mload.i(матрица размерности N\*N, элементы которой отображают время между отправкой запроса и получением ответа).

Эксперимент проводится K раз, причем с каждым новым экспериментом количество реплик (копий сервисов) увеличивается.

Основными задачами данного эксперимента является ***выявления неявных закономерностей и определение оптимальной архитектуры*** микросервисного приложения.

Основными исследуемыми характеристиками являются:

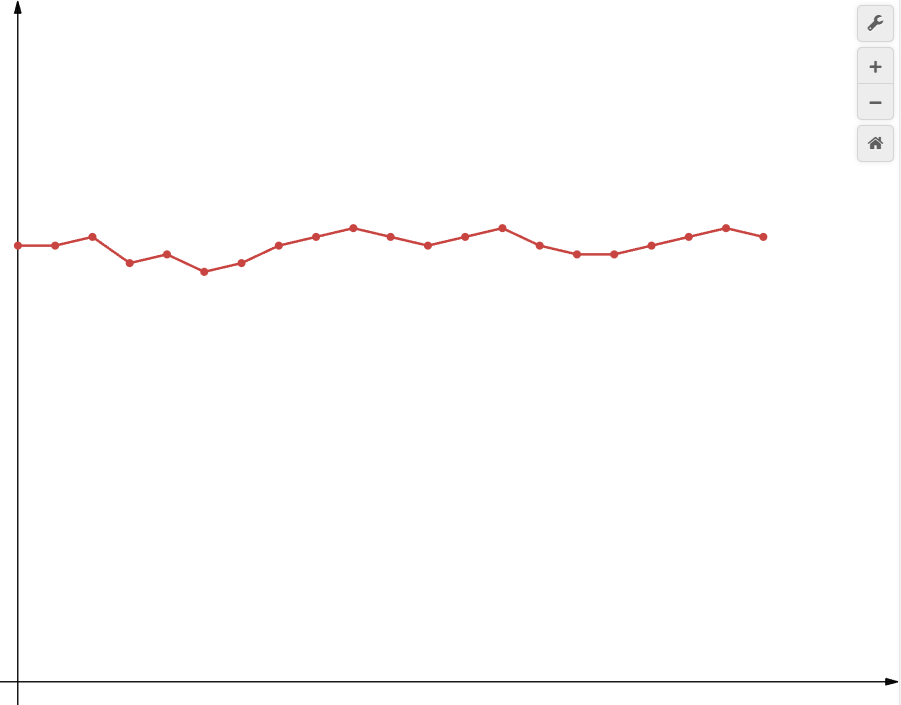
1. Потребление памяти.
2. Потребление ресурса процессора.
3. Энергопотребление.
4. Стабильность сети.
5. Производительность.

**ПОТРЕБЛЕНИЕ ПАМЯТИ**

Для заданной нагрузки Li каждые t/r мс считывается вектор относительной загрузки памяти MEMi.r. Получим матрицы

dMEMi.r = MEMi.r - MEMi.

Таким образом можно составить матрицу размерности N\*r, где строки матрицы отображают загруженность памяти при заданной нагрузке Li по времени t. Ожидается, что дисперсия по этим строкам окажется небольшой, что можно будет говорить о средней загруженности памяти при нагрузке Li по времени t.



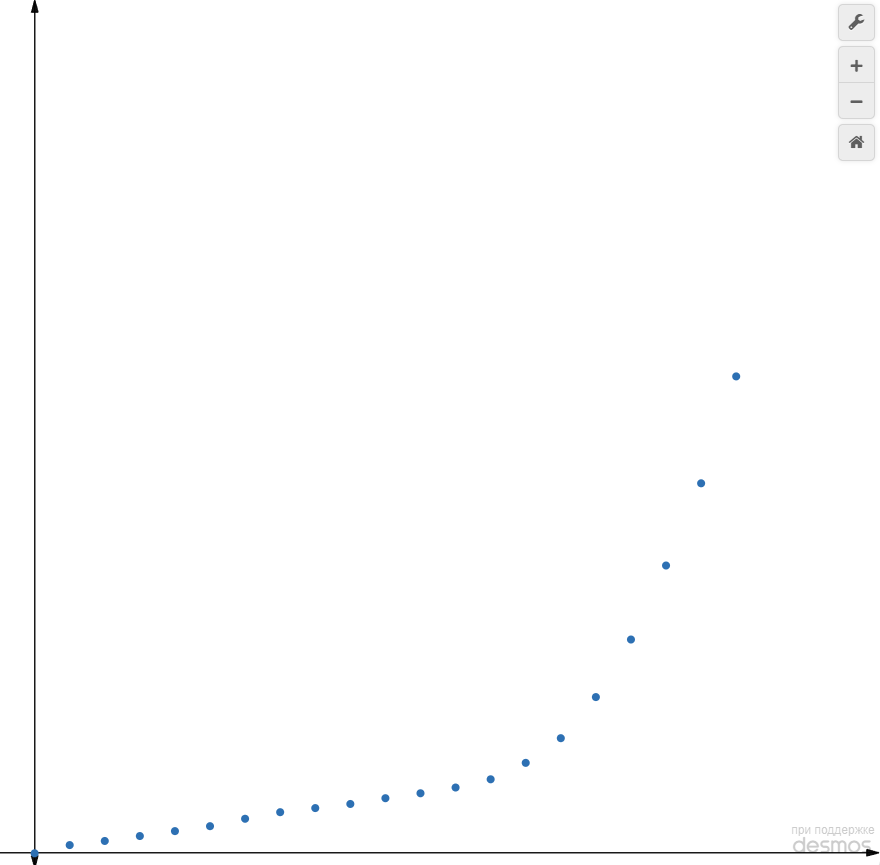
/\*Ожидаемый график зависимости дисперсии от времени \*/

PS Про методику вычисления среднего: имеет смысл при малой дисперсии использовать **среднее арифметическое**. Можно использовать эту величину в дискретном смысле, а можно и аппроксимировать дисперсии в непрерывную ф-ию, а затем вычислить среднее:

MEM = int(0, t)(f(x)dx)

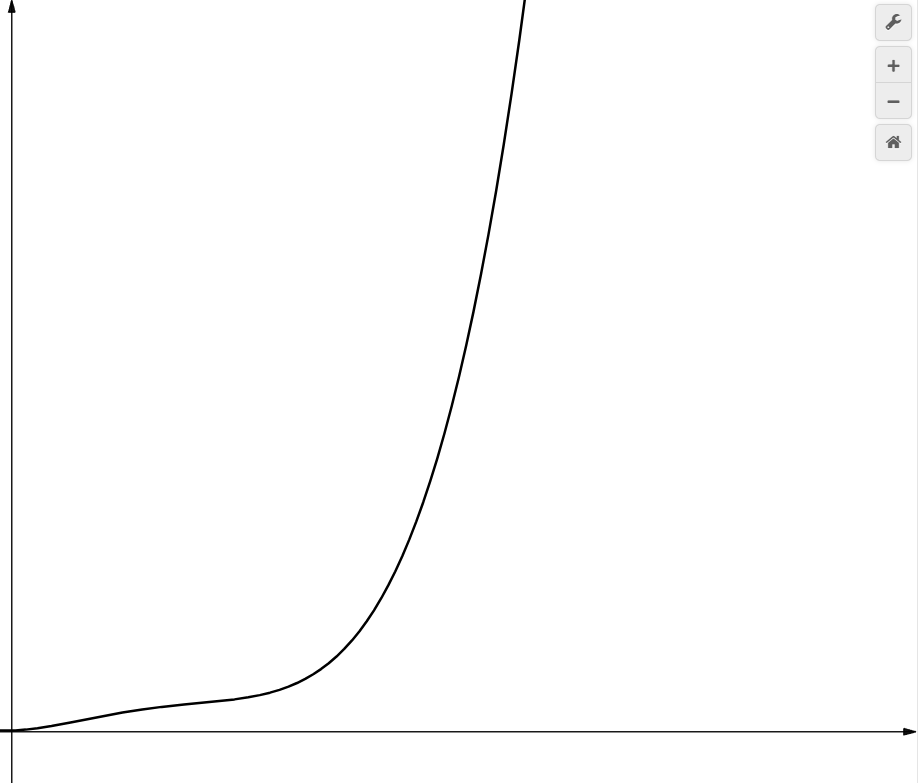
Таким образом получим дискретную функцию

fmem(L) = MEM

  
/\*Ожидаемый график дискретной функции fmem(L) \*/

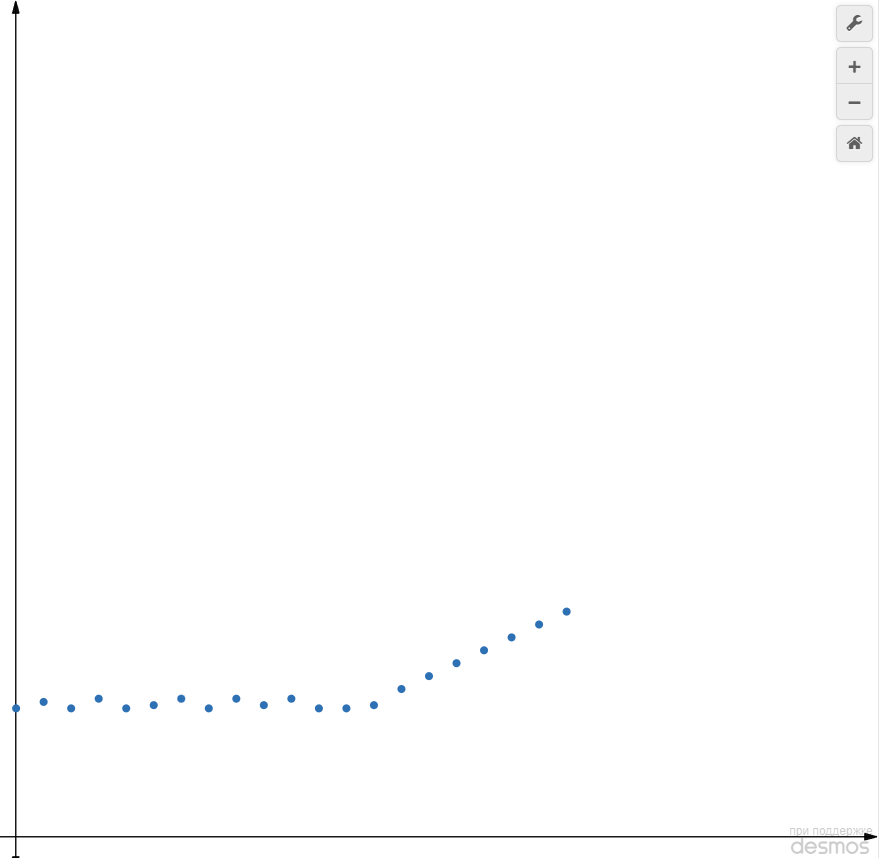
Которую мы аппроксимируем до непрерывной, всюду дифференцируемой функции

gmem(L) = MEM



/\*Ожидаемый график непрерывной функции gmem(L) \*/

а также получим функцию

g’mem(L) = dL/d

/\*Ожидаемый график непрерывной функции g’mem(L) \*/

**Оценка:**

* функция gmem(L) показывает зависимость памяти от заданной нагрузки. *Ожидается линейная зависимость, которая потом перейдёт в степенную или экспоненциальную.*
* функция g’mem(L) показывает рост загруженности памяти от нагрузки. *Ожидается константная зависимость, которая потом перейдёт в линейную, степенную или экспоненциальную.*

Проведем вышеуказанное исследование K раз, постепенно увеличивая количество реплик(копий) получим набор функций gmem.k(L), …, gmem.k(L) и набор значений Lкрит.1,… ,Lкрит.k, при достижении которых начинается нелинейный рост загруженности памяти (PS а может лучше при достижении X-персентиля???). Для каждого i: 1 <= i <= K вычислим следующие интегралы.

/\*Ожидаемые графики gmem.k(L), …, gmem.k(L) и отсечками Lкрит.1,… ,Lкрит.k с интегралами Ii \*/

Ii = int(0, Lкрит.i)(gmem.k(L)dL)

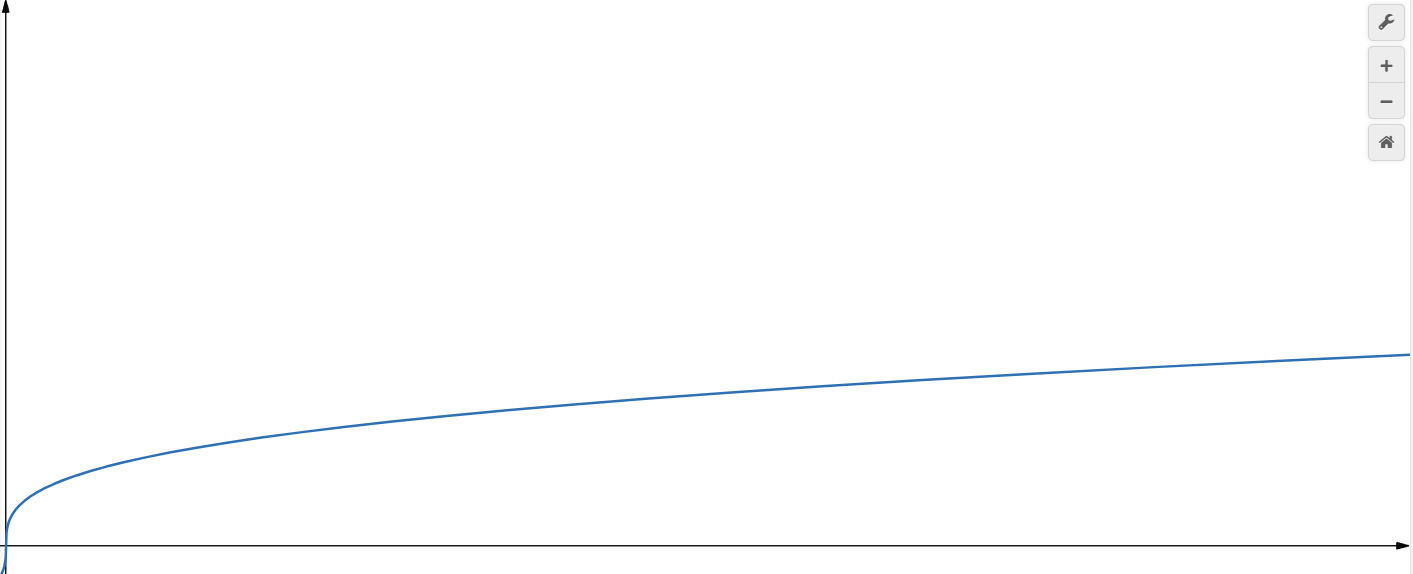
Данный интеграл показывает нагрузочно-загрузочную характеристику памяти до достижения критического значения нагрузки. Для более объективного оценивания вычислим относительную нагрузочно-загрузочную характеристику памяти

Chmem.i = Ii/i

Итоговая оценка заключается в Chmem.i. **Чем она меньше, тем лучше.**

А также получим **ключевую** зависимость Lкрит от количества n:

f(n) = Lкрит



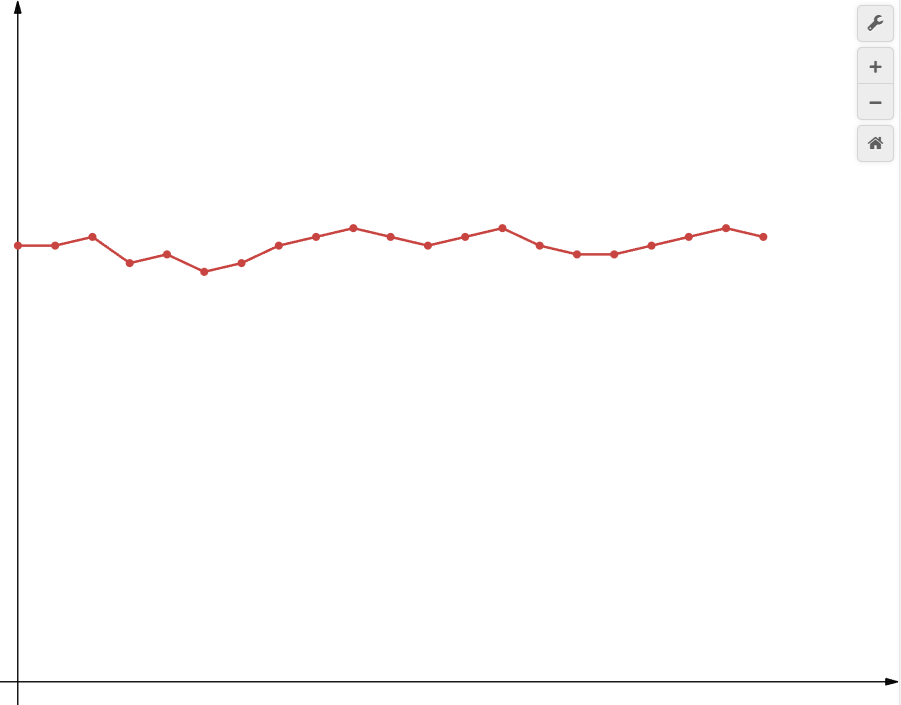
/\*Тут будет график f(n)\*/

**ПОТРЕБЛЕНИЕ РЕСУРСА ПРОЦЕССОРА**

*Аналогична оценке потребления памяти.* Для заданной нагрузки Li каждые t/r мс считывается вектор относительной загрузки процессора CPUi.r. Получим матрицы

dCPUi.r = CPUi.r - CPUi.

Таким образом можно составить матрицу размерности N\*r, где строки матрицы отображают загруженность процессора при заданной нагрузке Li по времени t. Ожидается, что дисперсия по этим строкам окажется небольшой, что можно будет говорить о средней загруженности процессора при нагрузке Li по времени t.



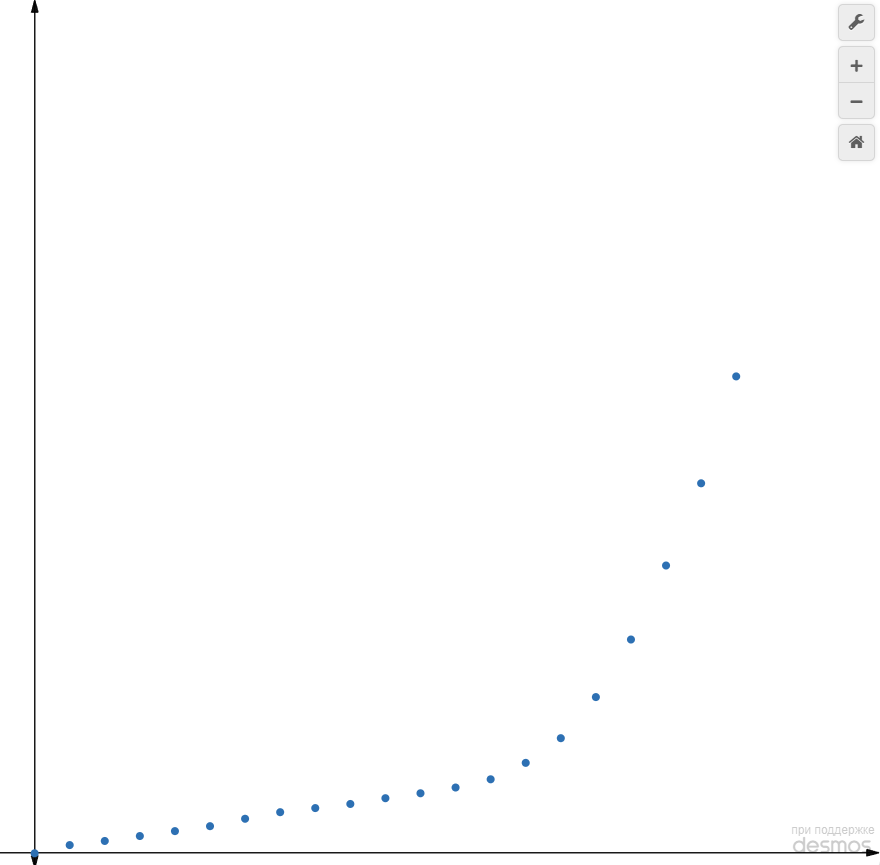
/\*Ожидаемый график зависимости дисперсии от времени \*/

PS Про методику вычисления среднего: имеет смысл при малой дисперсии использовать **среднее арифметическое**. Можно использовать эту величину в дискретном смысле, а можно и аппроксимировать дисперсии в непрерывную ф-ию, а затем вычислить среднее:

CPU = int(0, t)(f(x)dx)

Таким образом получим дискретную функцию

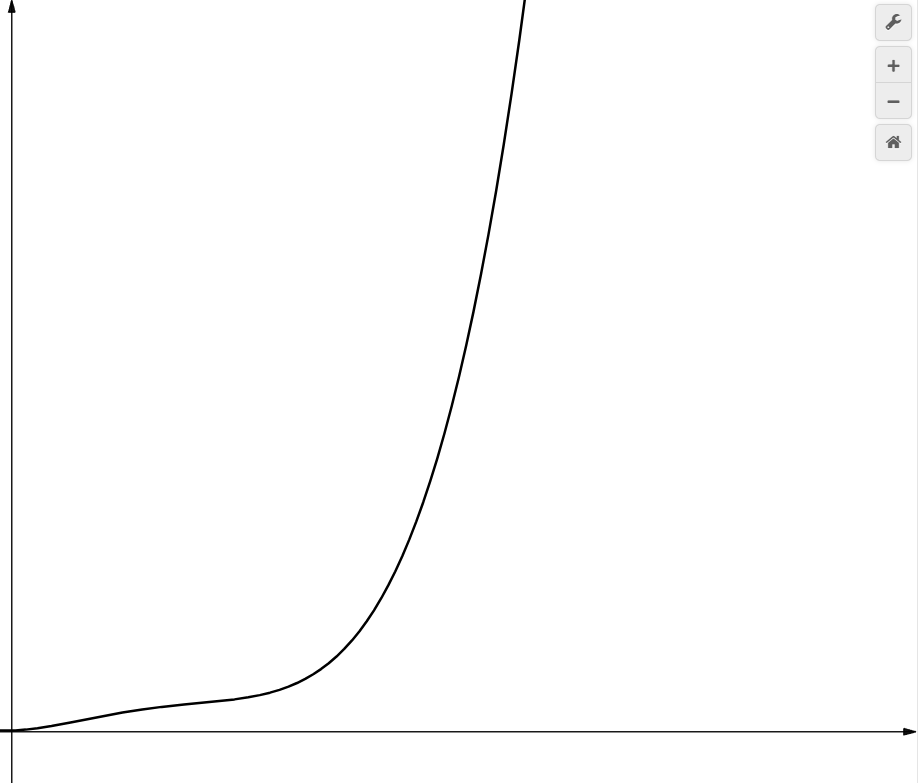
fCPU(L) = CPU



/\*Ожидаемый график дискретной функции fCPU(L) \*/

Которую мы аппроксимируем до непрерывной, всюду дифференцируемой функции

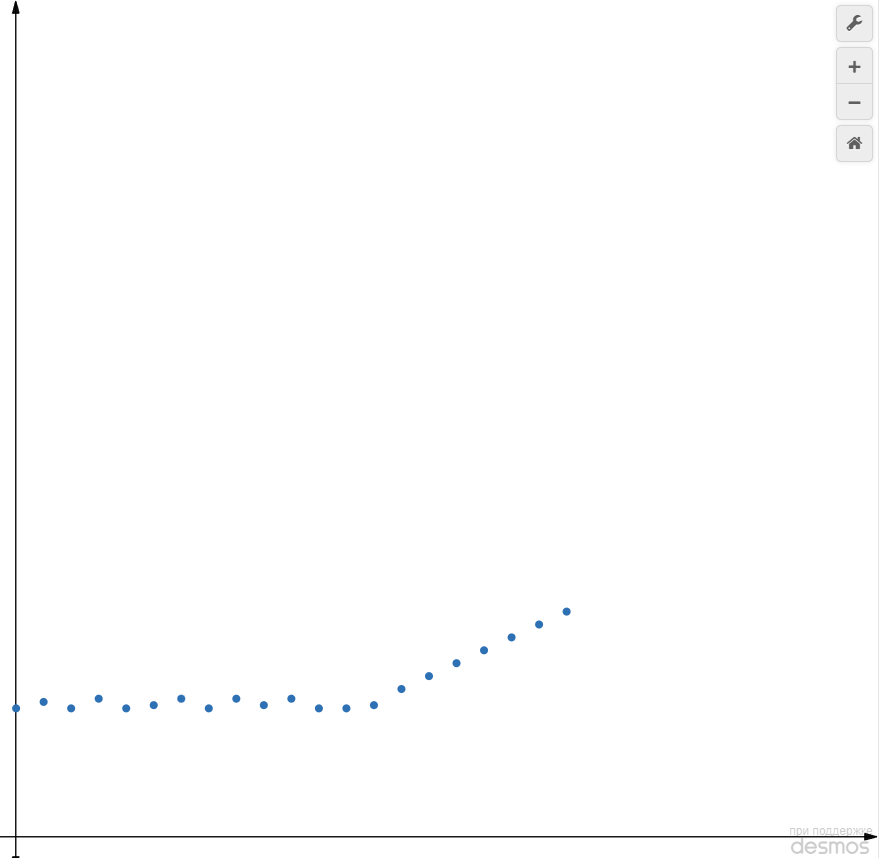
gCPU(L) = CPU



/\*Ожидаемый график непрерывной функции gCPU(L) \*/

а также получим функцию

g’CPU(L) = dL/dg



/\*Ожидаемый график непрерывной функции g’CPU(L) \*/

**Оценка:**

* функция gCPU(L) показывает зависимость загруженности процессора от заданной нагрузки. *Ожидается линейная зависимость, которая потом перейдёт в степенную или экспоненциальную.*
* функция g’CPU(L) показывает рост загруженности процессора от нагрузки. *Ожидается константная зависимость, которая потом перейдёт в линейную, степенную или экспоненциальную.*

Проведем вышеуказанное исследование K раз, постепенно увеличивая количество реплик(копий) получим набор функций gCPU.k(L), …, gCPU.k(L) и набор значений Lкрит.1,… ,Lкрит.k, при достижении которых начинается нелинейный рост загруженности процессора (PS а может лучше при достижении X-персентиля???). Для каждого i: 1 <= i <= K вычислим следующие интегралы.

Ii = int(0, Lкрит.i)(gCPU.k(L)dL)

/\*Ожидаемые графики gcpu.k(L), …, gcpu.k(L) и отсечками Lкрит.1,… ,Lкрит.k с интегралами Ii \*/

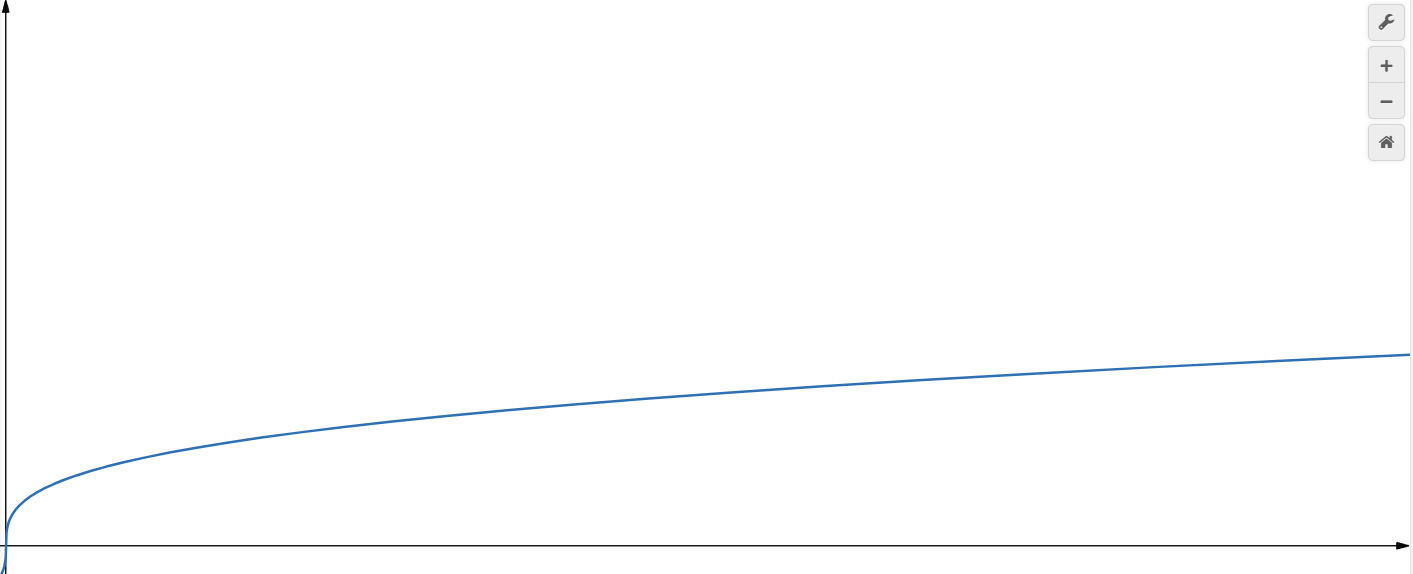
Данный интеграл показывает нагрузочно-загрузочную характеристику процессора до достижения критического значения нагрузки. Для более объективного оценивания вычислим относительную нагрузочно-загрузочную характеристику процессора

Chcpu.i = Ii/i

Итоговая оценка заключается в Chcpu.i. **Чем она меньше, тем лучше.**

А также получим **ключевую** зависимость Lкрит от количества n:

f(n) = Lкрит



/\*Тут будет график f(n)\*/

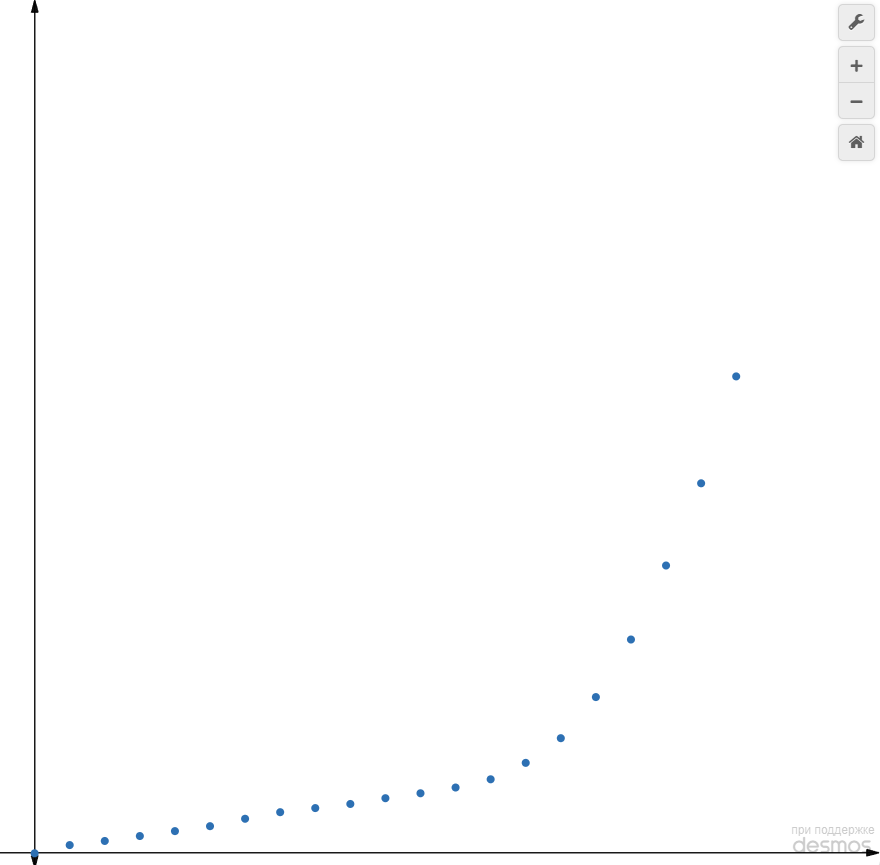
**ЭНЕРГОПОТРЕБЛЕНИЕ**

Для заданной нагрузки Li каждые t/r мс считывается вектор мощности Pi.r. Получим матрицы:

dPi.r = Pi.r - Pi.

В результате измерений получим ряд закономерностей f(t) = dPi.r который можно аппроксимировать до непрерывной функции

Fi(t) = dP



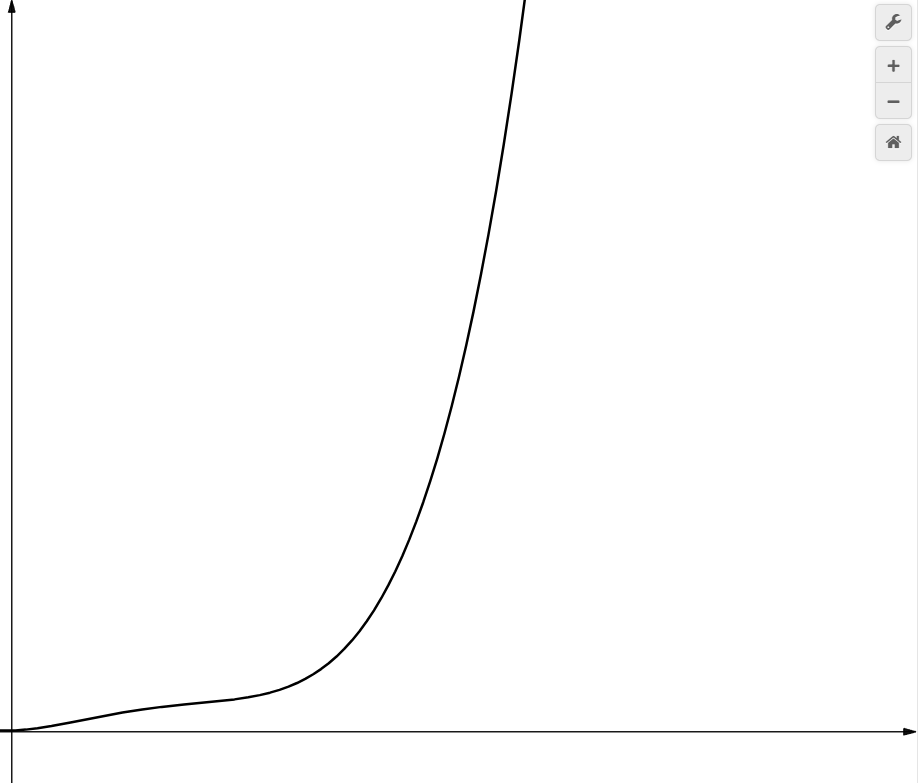
/\*График функции Fp(t)\*/

В результате мы получим ряд функций Fp.i(t), для всех i: 1 <= i <= n вычислим числовое значение *энергопотребления* системы:

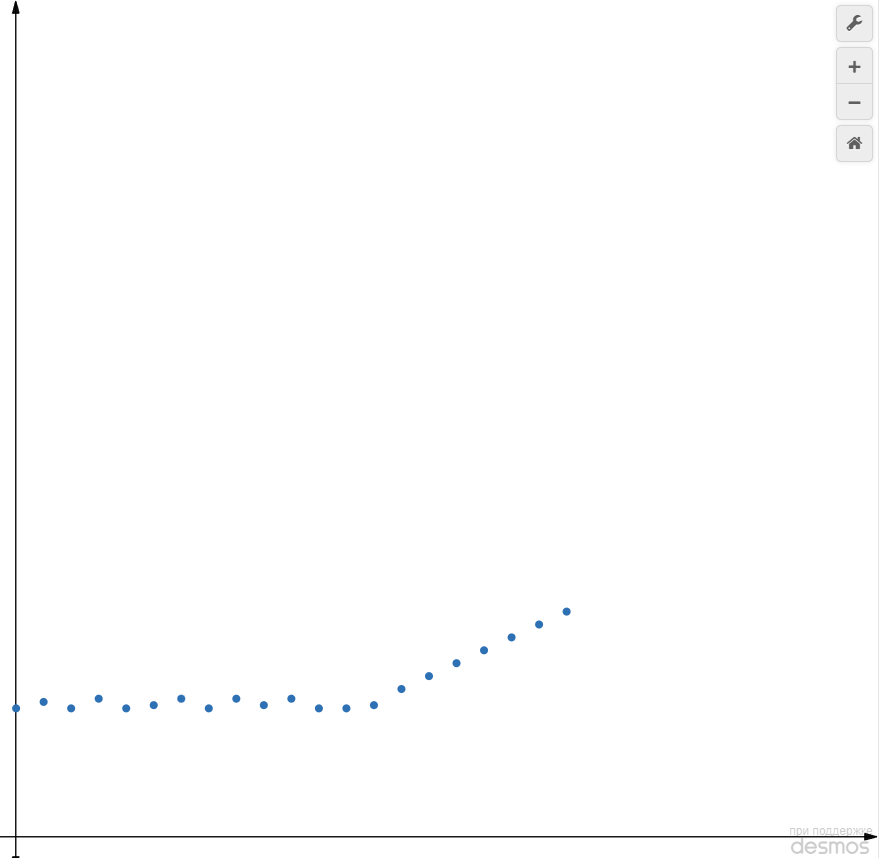
Ii = int(0,t) (Fi(t)dt)

Получим явную закономерность Ii от Li . Данная взаимосвязь является дискретной закономерностью, аппроксимируем её до непрерывной функции, а также сразу дифференцируем её:

gE(L) = E



g’E(L) = dL/dg



/\*Графики вышеприведённых функций\*/

**Оценка:**

* функция gE (L) показывает зависимость энергопотребления от заданной нагрузки. *Ожидается линейная зависимость, которая потом перейдёт в степенную или экспоненциальную.*
* функция g’E (L) показывает рост энергопотребления от нагрузки. *Ожидается константная зависимость, которая потом перейдёт в линейную, степенную или экспоненциальную.*

Проведем вышеуказанное исследование K раз, постепенно увеличивая количество реплик(копий) получим набор функций gE.k(L), …, gE.k(L) и набор значений Lкрит.1,… ,Lкрит.k, при достижении которых начинается нелинейный рост энергопотребления (PS а может лучше при достижении X-персентиля???). Для каждого i: 1 <= i <= K вычислим следующие интегралы.

Ii = int(0, Lкрит.i)(gE.k(L)dL)

/\*Ожидаемые графики gE.k(L), …, gE.k(L) и отсечками Lкрит.1,… ,Lкрит.k с интегралами Ii \*/

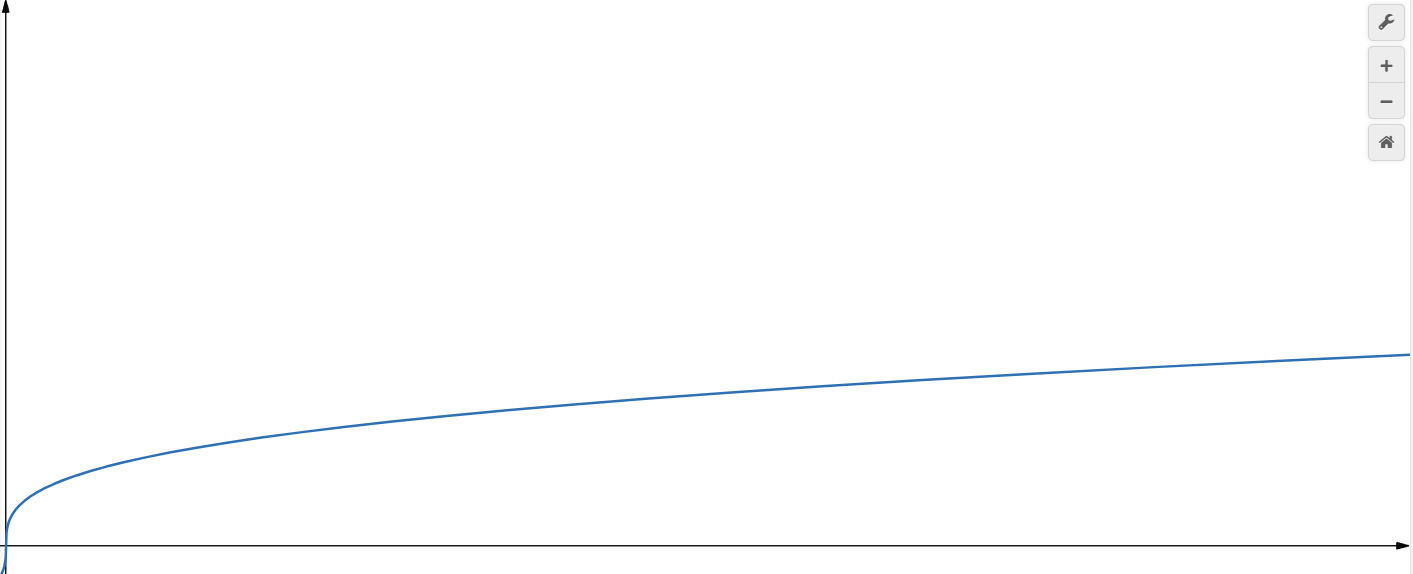
Данный интеграл показывает нагрузочно-энергетическую характеристику процессора до достижения критического значения нагрузки. Для более объективного оценивания вычислим относительную нагрузочно-энергетическую характеристику системы

ChE.i = Ii/i

Итоговая оценка заключается в ChE.i. **Чем она меньше, тем лучше.**

А также получим **ключевую** зависимость Lкрит от количества n:

f(n) = Lкрит



/\*Тут будет график f(n)\*/

**СТАБИЛЬНОСТЬ СЕТИ**

Для заданной нагрузки Li каждые t/r мс считывается вектор мощности Mсеть.i.r. Получим тензоры 3-го порядка размерности N\*N\*r:

dMсеть = Mсеть - Mстат.сеть

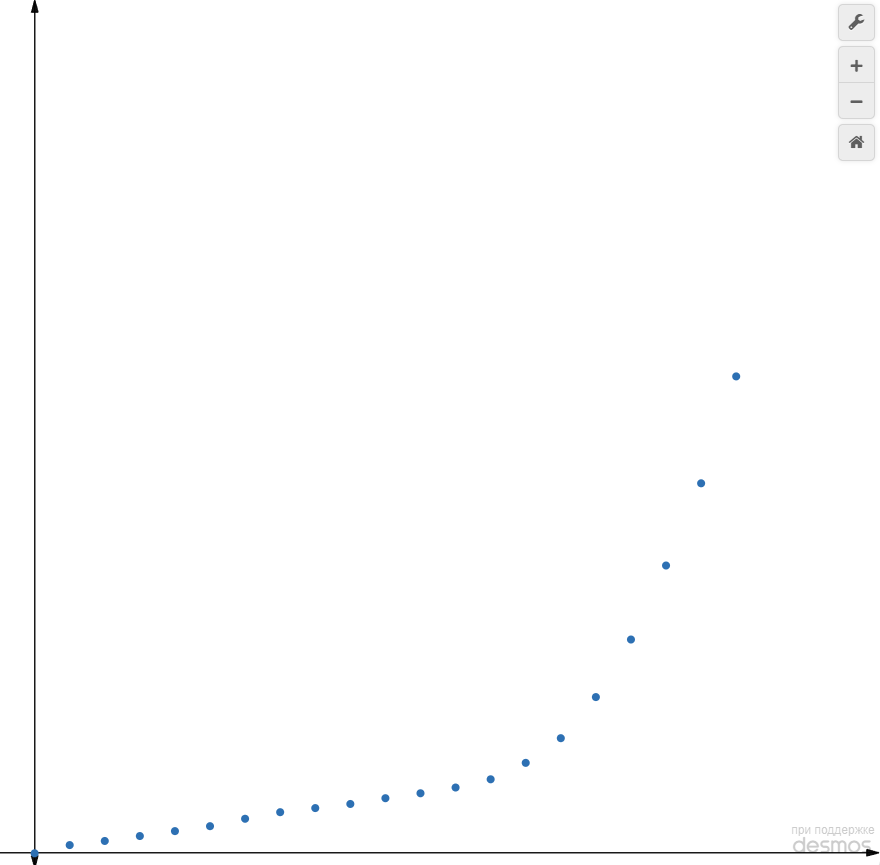
По строкам мы наблюдаем матрицы размерности N\*r которые отражают разницу, которая накладывает нагрузка Li во времени t. Ожидается, что дисперсия по этим строкам окажется небольшой, в связи со стабильностью нагрузки L. Следовательно можно говорить о *стабильности* сети при нагрузке Li по времени t.

PS Про методику вычисления среднего: имеет смысл при малой дисперсии использовать **среднее арифметическое**. Можно использовать эту величину в дискретном смысле, а можно и аппроксимировать дисперсии в непрерывную ф-ию, а затем вычислить среднее:

NET = int(0, t)(f(x)dx)

Таким образом получим дискретную функцию

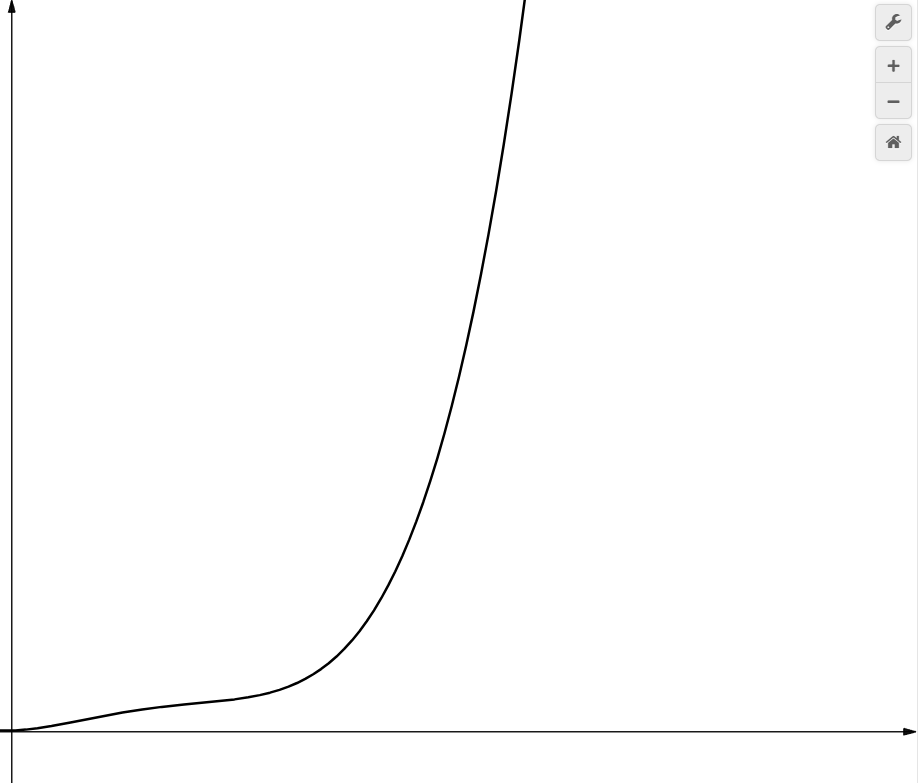
f NET(L) = NET



/\*Ожидаемый график дискретной функции fNET(L) \*/

Которую мы аппроксимируем до непрерывной, всюду дифференцируемой функции

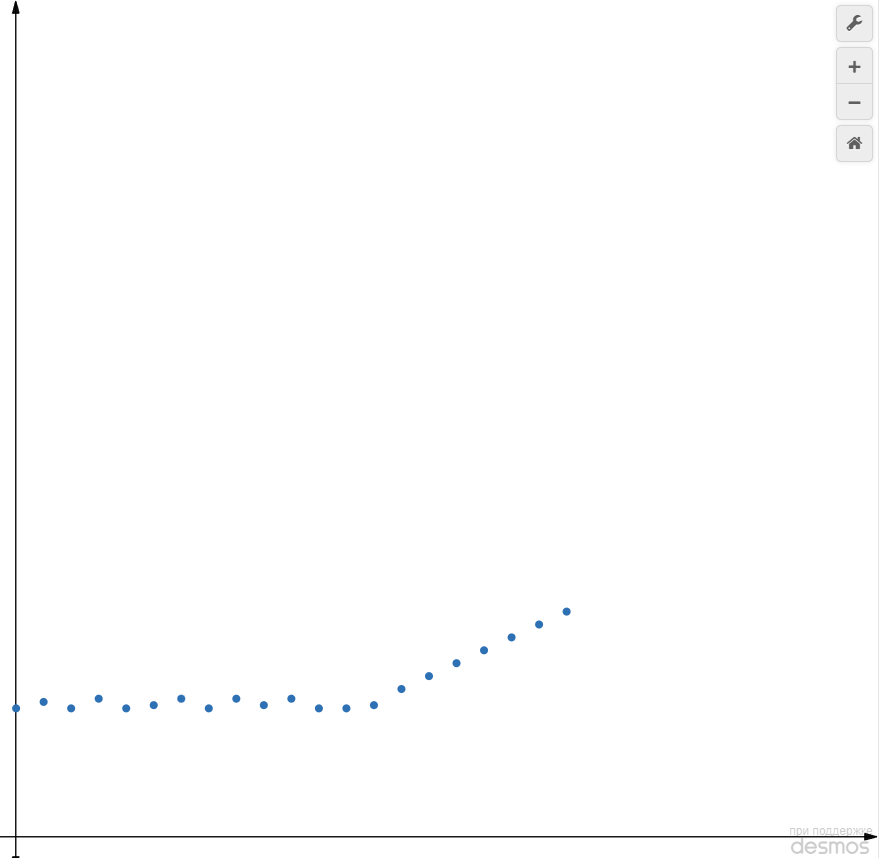
gNET(L) = NET



/\*Ожидаемый график непрерывной функции gNET (L) \*/

а также получим функцию

g’NET (L) = dL/dg

**

/\*Ожидаемый график непрерывной функции g’NET(L) \*/

**Оценка:**

* функция gNET (L) показывает зависимость стабильности сети от заданной нагрузки. *Ожидается линейная зависимость, которая потом перейдёт в степенную или экспоненциальную.*
* функция g’NET(L) показывает рост стабильности сети от нагрузки. *Ожидается константная зависимость, которая потом перейдёт в линейную, степенную или экспоненциальную.*

Проведем вышеуказанное исследование K раз, постепенно увеличивая количество реплик(копий) получим набор функций gNET.k(L), …, gNET.k(L) и набор значений Lкрит.1,… ,Lкрит.k, при достижении которых начинается нелинейный рост нестабильности сети (PS а может лучше при достижении X-персентиля???). Для каждого i: 1 <= i <= K вычислим следующие интегралы.

Ii = int(0, Lкрит.i)(gNET.k(L)dL)

/\*Ожидаемые графики gNET.k(L), …, gNET.k(L) и отсечками Lкрит.1,… ,Lкрит.k с интегралами Ii \*/

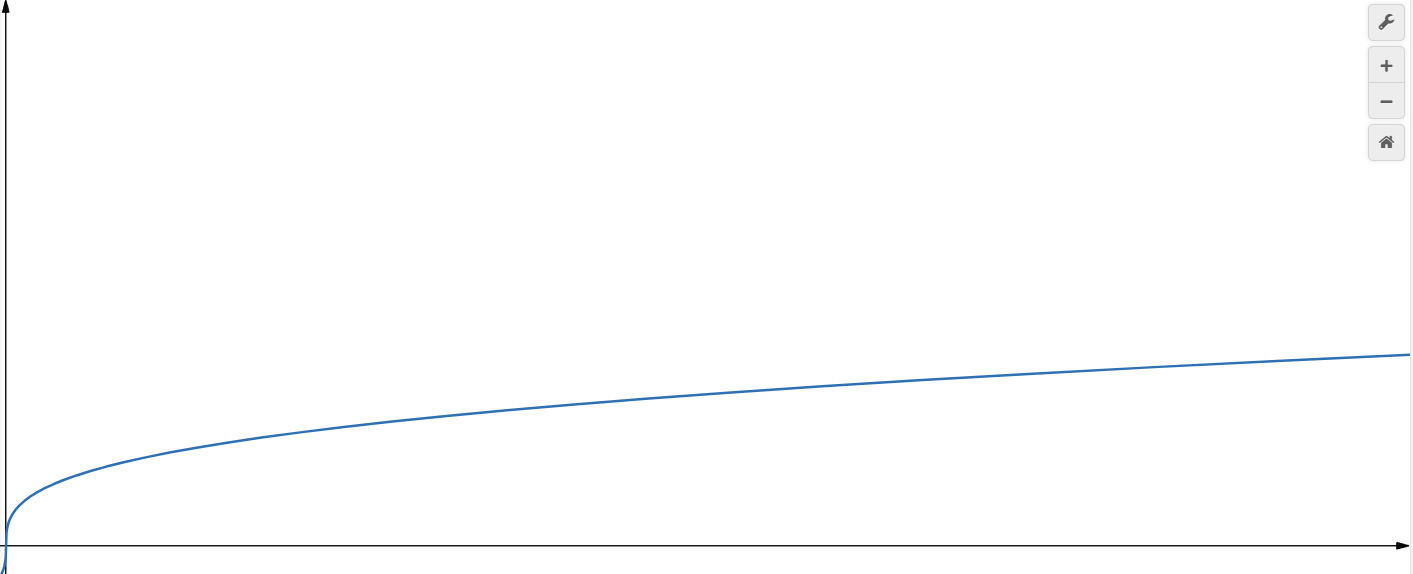
Данный интеграл показывает характеристику «стабильность сети-нагрузка» до достижения критического значения нагрузки. Для более объективного оценивания вычислим относительную характеристику

ChNET.i = Ii/i

Итоговая оценка заключается в ChNET.i. **Чем она меньше, тем лучше.**

А также получим **ключевую** зависимость Lкрит от количества n:

f(n) = Lкрит



/\*Тут будет график f(n)\*/

**ПРОИЗВОДИТЕЛЬНОСТЬ**

**Анализ производительности системы аналогичен анализу стабильности сети.** Для заданной нагрузки Li каждые t/r мс считывается вектор мощности Mload.i.r. Получим тензоры 3-го порядка размерности N\*N\*r:

dMload = Mload - Mсеть - Mстат.сеть

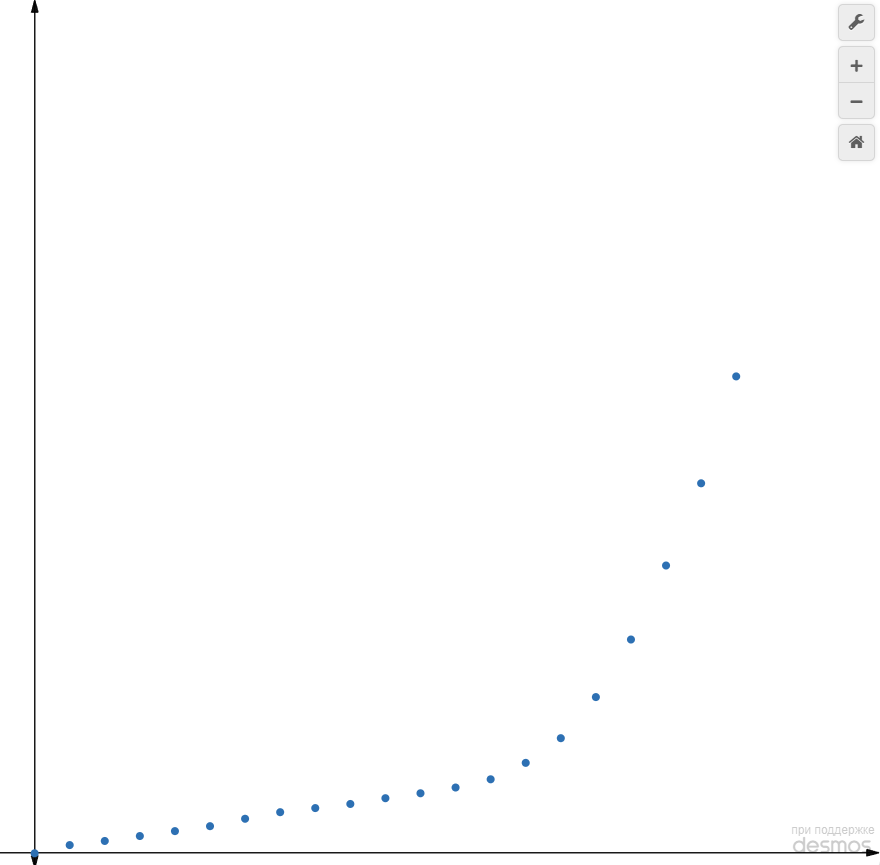
По строкам мы наблюдаем матрицы размерности N\*r которые отражают разницу, которая накладывает нагрузка Li во времени t. Ожидается, что дисперсия по этим строкам окажется небольшой, в связи со стабильностью нагрузки L. Следовательно можно говорить о *производительности и устойчивости системы* при нагрузке Li по времени t.

PS Про методику вычисления среднего: имеет смысл при малой дисперсии использовать **среднее арифметическое**. Можно использовать эту величину в дискретном смысле, а можно и аппроксимировать дисперсии в непрерывную ф-ию, а затем вычислить среднее:

LOAD = int(0, t)(f(x)dx)

Таким образом получим дискретную функцию

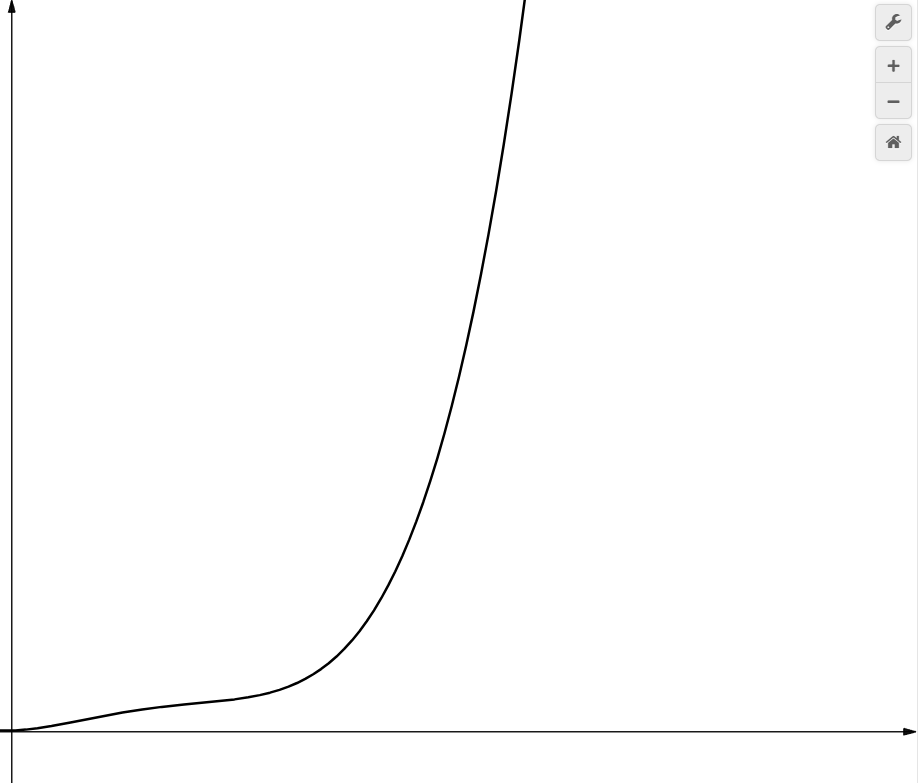
FLOAD(L) = LOAD



/\*Ожидаемый график дискретной функции fLOAD(L) \*/

Которую мы аппроксимируем до непрерывной, всюду дифференцируемой функции

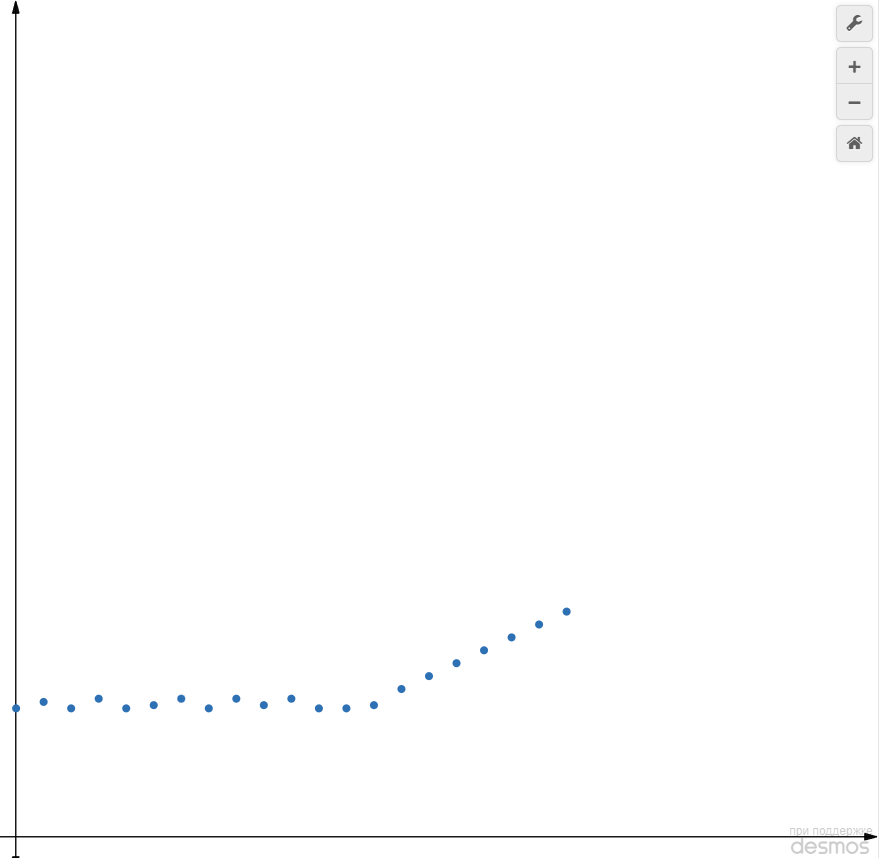
gLOAD(L) = LOAD



/\*Ожидаемый график непрерывной функции gLOAD (L) \*/

а также получим функцию

g’LOAD(L) = dL/dg

**

/\*Ожидаемый график непрерывной функции g’LOAD(L) \*/

**Оценка:**

* функция gLOAD (L) показывает зависимость производительности системы от заданной нагрузки. *Ожидается линейная зависимость, которая потом перейдёт в степенную или экспоненциальную.*
* функция g’LOAD(L) показывает падение производительности от нагрузки. *Ожидается константная зависимость, которая потом перейдёт в линейную, степенную или экспоненциальную.*

Проведем вышеуказанное исследование K раз, постепенно увеличивая количество реплик(копий) получим набор функций gLOAD.k(L), …, gLOAD.k(L) и набор значений Lкрит.1,… ,Lкрит.k, при достижении которых начинается нелинейное падение производительности приложения (PS а может лучше при достижении X-персентиля???). Для каждого i: 1 <= i <= K вычислим следующие интегралы.

Ii = int(0, Lкрит.i)(gLOAD.k(L)dL)

/\*Ожидаемые графики gLOAD.k(L), …, gLOAD.k(L) и отсечками Lкрит.1,… ,Lкрит.k с интегралами Ii \*/

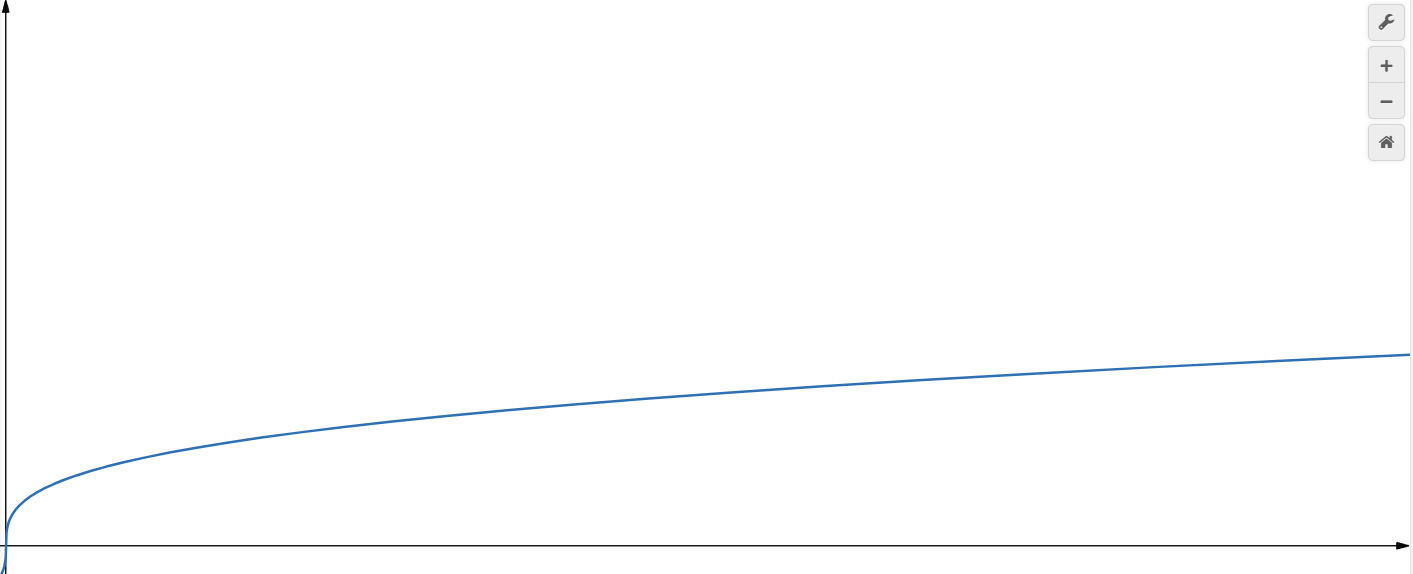
Данный интеграл показывает характеристику «производительность-нагрузка» до достижения критического значения нагрузки. Для более объективного оценивания вычислим относительную характеристику

ChLOAD.i = Ii/i

Итоговая оценка заключается в ChLOAD.i. **Чем она меньше, тем лучше.**

А также получим **ключевую** зависимость Lкрит от количества n:

f(n) = Lкрит



/\*Тут будет график f(n)\*/

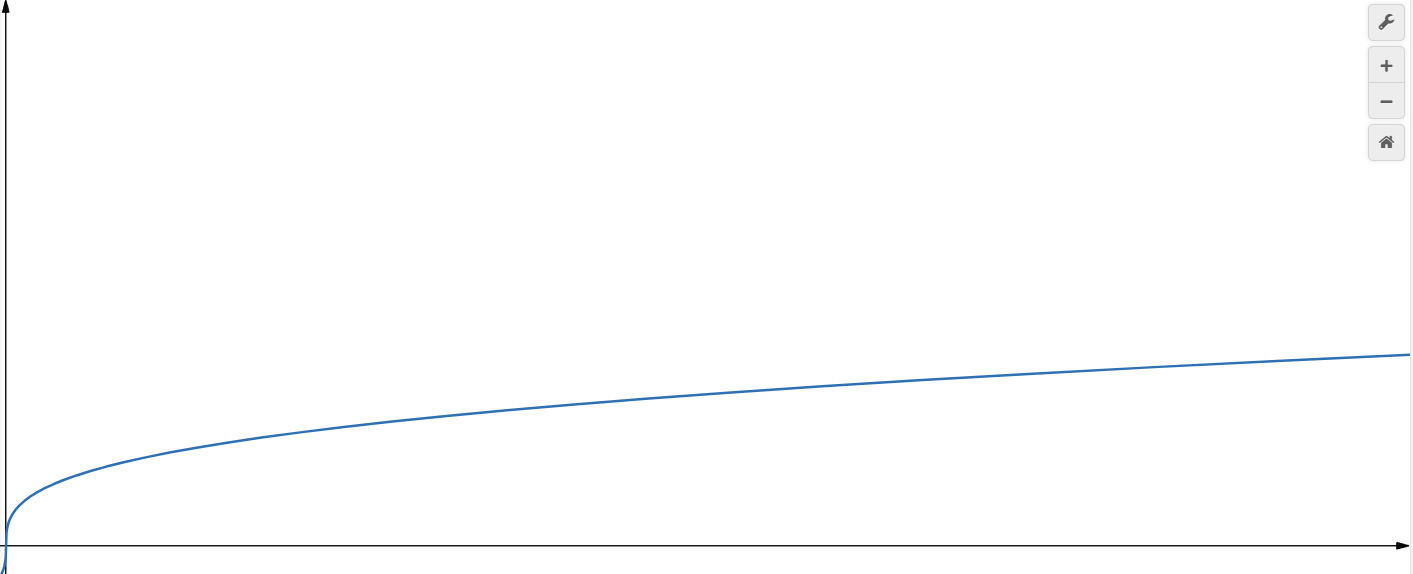
**ОБЩИЕ ВЗАИМОСВЯЗИ**

Выше мною был произведён анализ 5 метрик в отрыве друг от друга, т.е. на анализ не даёт комплексной оценки «система-сервер-нагрузка».

Разделим все метрики на две группы:

* *Ресурсные*: потребление памяти, потребление ресурса процессора, энергопотребление.
* *Производительные*: стабильность сети, производительность.

Ниже приведены ресурсные и производительные метрики (зависимости вида f(n) = Lкрит).



/\* Приведены пять графиков этих метрик\*/

Для начала займемся нормировкой метрик, для этого воспользуемся формулой:

Fnorm(n) = f(n) / Lкрит

* *Для ресурсных метрик*: Fnorm(n) показывает, какую *долю* от критического лимита мы потребляем. Идеал — значение близкое к 0, плохо — близкое к 1.
* *Для производительных метрик*: здесь интерпретация обратная. Fnorm(n) показывает, какую *долю* от максимально возможной (критической) производительности мы получаем. Идеал — значение близкое к 1, плохо — близкое к 0.

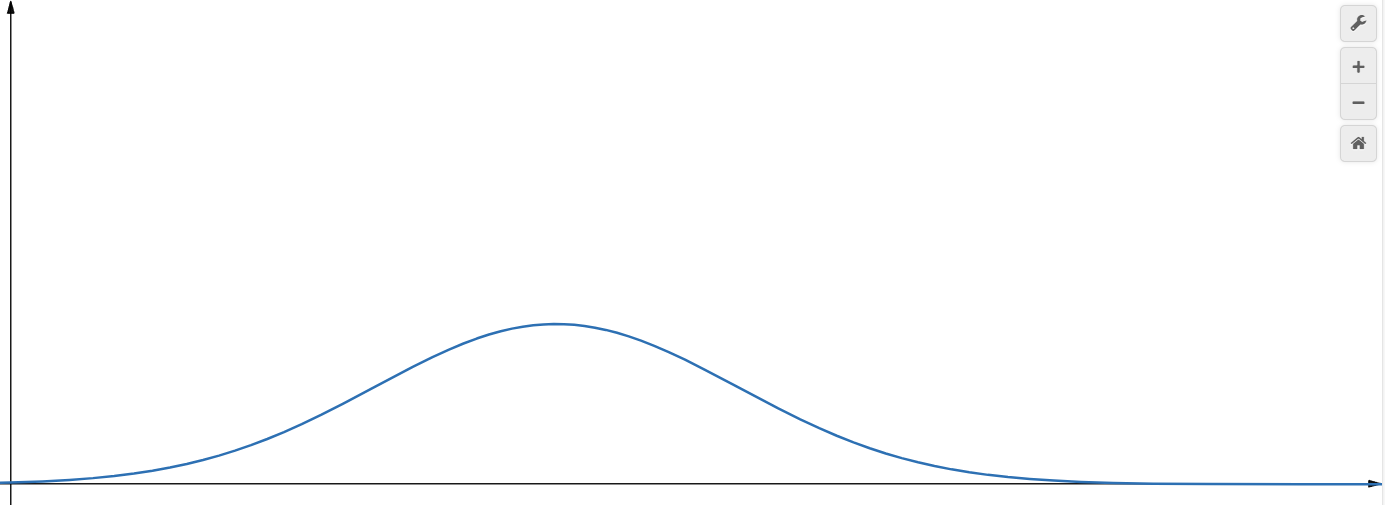
Теперь проведём группировку метрик: для всех значений:

R(n) = (CPUnorm(n) + MEMnorm(n) + Enorm(n)) / 3

P(n) = (NENTnorm(n) + LOADnorm(n)) / 2

Теперь после получений сводных кривых ресурсной нагрузки и производительности. Рассчитаем теперь *общую кривую эффективности*:

Effectivity = P(n) / R(n)



/\*тут график Effectivity = P(n) / R(n)\*/

Оптимальное количество реплик nopt будет там, где E(n) достигает максимума.