Numerical Methods II, Courant Institute, Spring 2024
http://www.math.nyu.edu/faculty/goodman/teaching/NumericalMethodsII2024.html
Jonathan Goodman, January, 2024

Laplace equation

Discrete Laplace equation, accuracy, variational principle, Gauss Seidel

sec:intro

1 Introduction

The first few classes if this course are about solving elliptic partial differential equations (PDEs). This is a non-standard choice, but it seems to be a setting where many important ideas and methods used in the rest of the class can be introduced naturally. The textbook of LeVeque also puts elliptic PDE first.

This class discusses only one PDE, the *Laplace equation*. The purpose is not really to learn only about the Laplace equation. There are specialized methods such as integral equations methods for the Laplace equation, but which apply apply only to equations a lot like the Laplace equation. This class describes properties and methods related to the Laplace equation that have analogues in many or most (depending on the property) other elliptic PDEs. Things that you can learn about the Laplace equation by direct calculation are also true in other problems, but for reasons that take too long to explain for this course.

To say this differently, the Laplace equation in 2D is used here as a *model problem*. It is not the problem of interest. The methods described here probably are not the *methods of choice* for the Laplace equation in 2D in a square domain. Fourier series or integral equations give more accurate approximations. But Fourier series solutions do not directly apply to non-linear problems or domains with a complex shape. See later classes and homework for problems where the numerical methods described here are closer to a method of choice. A *model problem* is one that illustrates general features of actual target problems while being simple enough to understand directly.

sec:PDE

2 The Laplace equation

The Laplace equation is a partial differential equation (PDE) satisfied by a function u. We use notation $x = (x_1, \cdots, x_n)$ for a point in n dimensional space. This class is mostly about the case $n = 2$, so I write (x, y) instead of (x_1, x_2). In physical applications, $u(x, y)$ could be the equilibrium temperature or concentration of some diffusing chemical. The Laplace equation in two dimensions is

$$\triangle u = \partial_x^2 u + \partial_y^2 u = f(x, y) \; . \tag{1}$$ Le

The function f is called the *inhomogeneous term* or *forcing function* or *right hand side* (the latter even if the equation is written with f on the left as $f = \triangle u$). If u represents temperature, then f represents heat sources (adding heat) or heat

sinks (taking away heat). If u represents electric potential, then f is electric charge density. If u represents gravitational potential, then f represents mass density.

The function u is supposed to be defined for all (x, y) in a *domain* Ω. This class always takes Ω to be the unit square $0 < x < 1$ and $0 < y < 1$ but real applications usually have Ω as the whole space or some set with geometry more complicated than a square. Complex geometry and unbounded domains are crucial topics, but too advanced for this first class. A point (x, y) is in the *interior* of Ω if there is some $\epsilon > 0$ so that $(x', y') \in \Omega$ whenever

$$|(x, y) - (x', y')| = \sqrt{(x - x')^2 + (y - y')^2} < \epsilon \ .$$

The Laplace equation (1) is supposed to be satisfied for every x in the interior of Ω. The *boundary* of Ω is denoted by $\partial\Omega$ or Γ. A point (x, y) is on the boundary if for every $\epsilon > 0$ there is at least one (x', y') with $|(x, y) - (x', y')| < \epsilon$ with (x', y') in Ω and at least one not in Ω. For mathematicians, the term *domain* that Ω implies that Ω is equal to its interior, so that the points on Γ are not part of Ω. With these understandings, the boundary of the unit square consists of the left edge: $x = 0$, $0 \le y \le 1$, together with top, bottom and right edges.

The function u is determined by the PDE (1) that holds in the interior together with *boundary conditions*, which are extra conditions on u that apply at each boundary point. This class treats only the *Dirichlet* boundary value problem, which is that the value $u(x, y)$ is specified for each $(x, y) \in \Gamma$. A book on partial differential equations probably has a proof that if the right hand side and boundary values are reasonable functions (a term not defined here) then there is a unique u that satisfies the PDE in Ω and the boundary conditions on Γ.

$$u(x, y) = g(x, y) \ \text{ for all } (x, y) \in \Gamma \ . \tag{2}$$ Dbc

The *boundary value* function g need only be defined for $(x, y) \in \Gamma$. A shorthand way to express this is

$$u = g \text{ on } \Gamma \ .$$

3 Finite Difference Discretization

sec:disc

The PDE (1) can be thought of as an equation that is supposed to hold at every point $(x, y) \in \Omega$. The unknowns are the values $u(x, y)$ for every $(x, y) \in \Omega$. This is an infinite set of equations for an infinite set of unknowns. *Discretization* means creating a finite (but large) set of numbers U to approximate the infinite set of values of u, and determining U by solving a finite (but large) set of equations.

One way to discretize involves *finite difference* approximations of the partial derivatives in the PDE. *Finite differences* are approximations to derivatives,

such as

$$v'(x) \approx \frac{v(x+\Delta x) - v(x)}{\Delta x}$$
$$v''(x) \approx \frac{v(x+\Delta x) - 2v(x) + v(x-\Delta x)}{\Delta x^2} \ .$$

Such approximations apply to the x and y derivatives in the Laplace operator of (1). The result is

$$\begin{aligned} \triangle u(x,y) \approx & \quad \frac{u(x+\Delta x,y) - 2u(x,y) + u(x-\Delta x,y)}{\Delta x^2} \\ & + \ \frac{u(x,y+\Delta x) - 2u(x,y) + u(x,y-\Delta x)}{\Delta x^2} \ . \end{aligned} \tag{3}$$

These finite differences are applied on a finite (but large) *mesh* of points in the interior of the unit square.

To define the mesh, we start with points on the x and y axes with uniform spacing Δx

$$x_j = j\Delta x \ , \ \ y_k = k\Delta x \ . \tag{4}$$

These create a *grid* (also called *mesh* or *net*) of points in the interior of the unit square of the form (x_j, y_k). We choose Δx so that there are n of the x points in the interior of the interval $[0,1]$. The definition (4) makes $x_0 = 0$ on the boundary of the interval. We adjust Δx so that $x_{n+1} = 1$ is the other boundary point. This breaks the interval $[0,1]$ into $n+1$ pieces of size Δx, so

$$\Delta x = \frac{1}{n+1} \ .$$

This is a *uniform grid* of size $n \times n = n^2$ where the finite difference approximations (3) will be applied.

If (x_j, y_k) is a grid point, then the finite difference approximation (3) involves the values of u only at other grid points. That is because, for example, $x_j - \Delta x = x_{j-1}$. We write $U_{jk} \approx u(x_j, y_k)$ for the approximate values on the grid points. We specify that U should satisfy a *finite difference approximation* to the Laplace equation (1) at the grid points. The *discrete Laplace operator* (discrete *laplacian*) is the right side of (3). We specify that U should satisfy the discrete Laplace equation, using the values of f at the grid points. This gives an equation at grid point (x_j, y_k) which takes the form (writing f_{jk} for $f(x_j, y_k)$)

$$\frac{U_{j+1,k} - 2U_{jk} + U_{j-1,k}}{\Delta x^2} + \frac{U_{j,k+1} - 2U_{jk} + U_{j,k-1}}{\Delta x^2} = f_{jk} \ . \tag{5}$$

An equivalent form of these equations shows that if $f = 0$ then each U_{jk} is the average of its four neighbors:

$$U_{jk} - \frac{U_{j+1,k} + U_{j-1,k} + U_{j,k+1} + U_{j,k-1}}{4} = \frac{-\Delta x^2}{4} f_{jk} \ . \tag{6}$$

These equations are supposed to apply for all n^2 grid points in the interior of the square. We assume that the "boundary" values are given their exact known values:

$$\left.\begin{array}{lcll} U_{0k} & = & u(0,y_k) & = & g(0,y_k) \\ U_{n+1,k} & = & u(x_{n+1},y_k) & = & g(1,y_k) \\ U_{j,0} & = & u(0,y_k) & = & g(0,y_k) \\ U_{j,n+1} & = & u(x_{n+1},y_k) & = & g(1,y_k) \end{array}\right\} . \tag{7}$$

bv

There are n^2 equations (5), one for each interior grid point. There are also n^2 unknowns U_{jk}. The boundary values that appear in some of the equations, such as $U_{0,k}$ and $U_{n+1,k}$, have known values. Thus the finite difference equations (5) are a system of linear equations with the same number of equations as unknowns. If the corresponding matrix is non-singular, then there is a unique solution. The solution is U, which is the finite difference approximation to u.

Ghost cells

Ghost cells are a programming trick that simplify the handling of boundary conditions. It seems natural, at first, to represent U in the computer with a two index array `U[j,k]`, where `j` and `k` are allowed to range from 1 to n (in Julia, Fortran, Matlab) or from 0 to $n-1$ (Python, C++, C). If you do this, then you have to check when you apply the finite difference formulas (6) whether the grid point (x_j, y_k) has a neighbor on the boundary (which corresponds to $j = 1$ or $j = n$ or $k = 1$ or $k = n$). Another way to do this is to allocate an array with dimensions $(n+2)\times(n+2)$ and put the boundary values g_{jk} into the boundary locations in the U array. For example, $U[0,k] = g_{0,k} = g(0,y_k)$, and $U[n+1,k] = g_{n+1,k} = g(1,y_k)$. This will give the right result whenever you apply the difference eqations (6) at an interior point without needing special code to hold the cases j or k being 1 or n. For example, when $j = 1$ the formula (6) accesses the value $U_{0,k}$, which would have the correct value $g(0,x_k)$. The values $U_{0,k}$, etc., are *ghost values*.

More complicated schemes with a higher order of accuracy (future classes) may use more than one row or column of ghost cells and ghost values.

4 Variational principle

sec:vp

A *variational principle* for u of U is a minimization problem so that u or U is the minimizer. Variational principles have many uses. In some cases, they help derive a PDE to describe a physical situation. For example, strains (displacements) in a material under stress minimize the total elastic energy, which is in integral involving first derivatives of the strains. Variational principles help in deriving good discretizations of a PDE. If the discretization has a discrete variational principle related to the continuous one satisfied by the PDE, then the discretization inherits stability properties of the PDE. *Finite element* methods are the method of choice for many elliptic PDE problems, and they are based on variational principles. Finally, variational principles lead to iteration

algorithms to solve the discrete equations that have guaranteed convergence properties. The *Gauss Seidel* algorithm described in Section 6.3 is an example. The plan of this Section is to explain a variational principle for the Laplace equation, then to derive a related discrete variational principle for the finite difference discrete equations (5) or (6).

Variational principles for linear problems involve *quadratic forms* and *linear forms* (often called linear *functionals*). If the function in the variational principle has a minimum, then the quadratic form is *positive definite*. A positive definite quadratic form defines an *inner product* that can be used to define orthogonality and a notion of distance (a *metric*). This inner product and metric lead to a simple geometric picture of the problem formulation and the effect of approximations.

4.1 For matrices

This subsection is about linear algebra, but it uses notation from Section 3 for continuity. The dimension is M; a generic vector is U or V. The right hand side is F. So we write $AU = F$ instead of $Ax = b$.

The $M \times M$ matrix A is *symmetric* if $A^T = A$. It is *positive definite* if $U^T AU > 0$ whenever $U \neq 0$. A matrix is *symmetric positive definite*, or *SPD*, if it has both properties. A *quadratic form*[1] is a function of $U \in \mathbb{R}^M$ of the form

$$Q(U) = \tfrac{1}{2}\, U^T AU \ .$$

We will usually assume, and often forget to say, that A is symmetric. If A is not symmetric, then there is a symmetric matrix $\widetilde{A}$ that has the same quadratic form. This is because $z = \frac{1}{2}\, U^T AU$ is a number, which may be thought of as as 1×1 matrix. Therefore $z^T = z$ implies that

$$U^T AU = z = z^T = U^T A^T (U^T)^T = U^T A^T U \ .$$

Therefore A and A^T give the same quadratic form. This implies that we can average and get

$$U^T AU = \tfrac{1}{2}\big(\, U^T AU + U^T A^T U \,\big) = U^T \widetilde{A} U \ , \quad \widetilde{A} = \tfrac{1}{2}\big(\, A + A^T \,\big) \ .$$

The symmetric matrix $\widetilde{A}$ is the *symmetric part*[2] of A. The quadratic form coming from A is the same as the quadratic form coming from the symmetric part of A.

[1] A polynomial in more than one variable is called a *form* if every term has the same degree. For example $C(U) = U_1^3 + U_2^2 U_3 - 2U_1 U_2 U_3$ is a cubic form. The quadratic form is a sum of quadratic monomials of the form $a_{jk} U_j U_k$.

[2] The *skew symmetric part* of A is $\frac{1}{2}\big(\, A - A^T \,\big)$. The matrix A is the sum of its symmetric and skew symmetric parts. If z is a complex number, its real part is $\frac{1}{2}(z + \overline{z})$ and its imaginary part is $\frac{1}{2}(z - \overline{z})$. The complex number is the sum of its real and imaginary parts. Taking the transpose of a matrix is analogous to taking the conjugate of a complex number. The symmetric part of a matrix has all real eigenvalues (being a symmetric matrix) and the skew symmetric part has all imaginary eigenvalues.

A positive definite matrix must be non-singular. Since A is square, if A is singular then there is a $U \neq 0$ with $AU = 0$. But taking the inner product with U then would give a non-zero vector with $U^T AU = U^T 0 = 0$, which contradicts the condition that A is positive definite.

An abstract *inner product* is a function of two vectors U and V and is written $\langle U, V \rangle$. To be a (real) inner product, it must be symmetric, linear in each argument, and positive definite:

$$\begin{aligned} \langle aU_1 + bU_2, V \rangle &= a\langle U_1, V \rangle + b\langle U_2, V \rangle \\ \langle U, aV_1 + bV_2 \rangle &= a\langle U, V_1 \rangle + b\langle U, V_2 \rangle \\ \langle U, V \rangle &= \langle V, U \rangle \\ \langle U, U \rangle &> 0 \text{ if } U \neq 0 \ . \end{aligned}$$

(Exercise for fun: show that an inner product has $\langle 0, 0 \rangle = 0$.) An SPD matrix defines an inner product through the formula

$$\langle U, V \rangle_A = U^T AV \ .$$

Conversely any inner product defines an SPD matrix. The entries are

$$A_{jk} = \langle e_j, e_k \rangle \ , \quad \text{the } e_j \text{ being the standard basis elements}$$

An inner product defines a *norm* through

$$\|U\|^2 = \langle U, U \rangle \ , \quad \text{or} \quad \|U\| = \sqrt{\langle U, U \rangle} \ .$$

The A inner product defines the A norm:

$$\|U\|_A^2 = \langle U, U \rangle_A = U^T AU \ .$$

There are norms not defined by an inner product, but any norm satisfies the *triangle inequality*

$$\|U + V\| \leq \|U\| + \|V\| \ .$$

A norm that comes from an inner product also satisfies the *Cauchy Schwarz inequality*

$$\langle U, V \rangle \leq \|U\| \cdot \|V\| \ .$$

There is an inequality about numbers: $xy \leq \frac{1}{2}x^2 + \frac{1}{2}y^2$. Combined with Cauchy Scshwarz, this gives

$$\langle U, V \rangle \leq \frac{1}{2} \|U\|^2 + \frac{1}{2} \|U\|^2 \ .$$

There is a simple but clever generalization of this that is used often in technical arguments. Let a be any positive number. Then[3]

$$\langle U, V \rangle = \langle \sqrt{a}U, \frac{1}{\sqrt{a}}V \rangle \leq \frac{a}{2} \|U\|^2 + \frac{1}{2a} \|V\|^2 \ .$$

[3]This is sometimes called the *Peter Paul* principle. To find out why, do a web search for the expression: "Rob from Peter to pay Paul".

This allows you to choose a small a and get an inequality that puts a small weight on $\|U\|$ (taking from Peter) if you compensate by putting a large weight on $\|V\|$ (paying Paul).

Warning: The matrix A that comes from the discretization (6) of the Laplace equation is symmetric and positive definite (we will see). The discretization (5) differs by a sign, so its matrix is symmetric and negative definite.

Any linear system of equations involving an SPD matrix has a *variational formulation*. Finding U to satisfy $AU = F$ is equivalent to

$$U = \arg\min_V \ \tfrac{1}{2} V^T A V - V^T F \ . \tag{8}$$ Avp

To see this you have to see that the function

$$E(V) = \tfrac{1}{2} V^T A V - V^T F \tag{9}$$ E(V)

has a minimum and a unique minimizer. Then you need to understand why that minimizer solves the linear equation system. Calculating the gradient of E solves the second issue and part of the first one. It gives a "weak form" of the variational principle:

$$\nabla E(U) = 0 \iff AU = F \ . \tag{10}$$ wvf

Since any positive definite matrix is non-singular, the solution U is unique and there is only one stationary point.[4]

We calculate the gradient of the quadratic and linear parts of E separately. The i component of $\nabla E(V)$ is

$$\frac{\partial}{\partial V_i} E \ .$$

The i component of $\nabla \left(V^T F \right)$ is

$$\frac{\partial}{\partial V_i} V^T F = \frac{\partial}{\partial V_i} \sum_j V_j F_j = F_i \ .$$

Since this is true for any index i, it implies the vector form

$$\nabla \, V^T F = F \ . \tag{11}$$ lg

Of course, $V^T F = F^T V$ so $\nabla F^T V = F$ also.

You can calculate $\nabla Q(V)$ in a similar way, but it may be approached abstractly as well. The abstract approach relies on a fact of calculus. If $f(x, y)$ is a function of two variables, then

$$\frac{d}{dx} f(x, x) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \ .$$

We apply this here using the *bilinear form* related to Q, which is

$$B(V, W) = \tfrac{1}{2} V^T A W = \frac{1}{2} \langle V, W \rangle_A \ .$$

[4] A *stationary point* of a function $f(x)$ is an x where $\nabla f(x) = 0$ Any minimizer is a stationary point but a stationary point need not be a minimizer, or even a local minimizer.

The quadratic form is $Q(V) = B(V,V) = \frac{1}{2}\,\|U\|_A^2$. The calculus fact applied to B implies that

$$\nabla Q = \nabla B(V,V) = \nabla_V B + \nabla_W B\ .$$

The notation is not ideal, but $\nabla_V B(V,W)$ is the gradient of B as a function of V with W held fixed, etc. Since $V^TAW = V^T(AW)$, we may take AW to be the F above and get

$$\nabla_V\, V^TAW = AW\ .$$

Similarly,

$$\nabla_W\, V^TAW = \nabla_W\, W^TA^TV = A^TV\ .$$

When A is symmetric, these calculations combine to give

$$\nabla B = \tfrac{1}{2}\,(AV + AW)\ .$$

When $V = W$, this leads to

$$\nabla Q(V) = AV\ . \tag{12}$$ qg

These results may be combined to give the gradient of the combined function (9) E(V)

$$\nabla E(V) = AV - F\ . \tag{13}$$ gE

This implies the weak variational formulation (10) wvf.

The *strong variational principle* states not only that the solution of the linear system is a stationary point, but also that U is the minimizer of E. This depends on the fact that A is positive definite. To see what's important here, consider a matrix that is symmetric but not positive definite, such as

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\ .$$

This matrix has

$$Q(V) = \tfrac{1}{2}\,\left(\,V_1^2 - V_2^2\,\right)\ .$$

The stationary point where $\nabla Q = 0$ is $U = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, but this U is not a minimizer because Q has no minimum.

The geometry of quadratic minimization problems involving $E(V)$ (see below) suggests the following proof that the solution of $AU = F$ is a minimizer. Let W be the difference between a generic vector V and the solution, U, of the linear system: $V = U + W$. Note that $AU = F$ implies that $U^TAU = U^TF$ and $W^TAU = W^TF$. Therefore,

$$\begin{aligned} E(V) = E(U+W) &= \frac{1}{2}\,(U+W)^TA\,(U+W) + (U+W)^TF \\ &= \frac{1}{2}U^TAU + W^TAU + \frac{1}{2}W^TAW + U^TF + W^TF \\ &= -\frac{1}{2}U^TAU + \frac{1}{2}W^TAW\ . \end{aligned}$$

If you look at this as a function of W alone, then U^TAU is a constant. Since A is positive definite, $W^TAW > 0$ unless $W = 0$. Thus, $E(V) > E(U)$ unless $W = 0$. If $W = 0$ then $E(V) = E(U)$. This shows U is a minimizer of E. We already saw that *a* minimizer is *the* minimizer, which also is the solution of $AU = F$.

Rather than this computational argument, you could argue that there is an r so that $E(V) > 0$ if $|V\| > r$ (details left out, but not too hard). Informally, this is because when V is large, the positive definite part of E "dominates" the linear part. Of course $E(\epsilon F) = \frac{\epsilon^2}{2} F^T AF - \epsilon \|F\|^2$, which is negative if ϵ is small enough. Therefore the minimum of $E(V)$ is the same as the minimum over a compact set, so it is attained. As before, a minimum is the unique solution of $AU = F$.

There is a geometric interpretation of the variational principle that will be useful later. For any number c there is a hyperplane of vectors with $V^TF = c$

$$H(c) = \left\{ V \text{ with } V^TF = c \right\} .$$

You seek vectors $V \in H(c)$ that are as close to the origin as possible in the A norm. This is equivalent to minimizing half of the square of the A norm

$$\min_{V \in H(c)} \frac{1}{2} \|V\|_A^2 = \min_{V^TF=c} \frac{1}{2} V^TAV . \tag{14}$$ gmp

The constrained minimizer may be found using a Lagrange multiplier λ, and we already have expressions for the gradients involved, so

$$AV = \lambda F . \tag{15}$$ Lm

We can find a formula for λ using $V = \lambda A^{-1}F$ (which may be transposed to give $V^T = \lambda F^T A^{-1}$) , and the constraint $V^TF = C$. The result is

$$\lambda = \frac{c}{F^TA^{-1}F} .$$

The geometric minimization problem (14) also may be used to solve the linear system. In fact the solution, V, of (14) satisfies $AU = F$ "up to a scale factor". That is, $U = mV$, for some scale factor m. Substituting in gives

$$AU = m\lambda F .$$

Thus, we solve the linear system with the scaling $U = \frac{1}{\lambda} V$. In retrospect, you could substitute $U = \frac{1}{\lambda} F$ into AU and get F.

4.2 For the Laplace equation

sec:vLe

One variational principle for the inhomogeneous Laplace equation (1) (also called the *Poisson* problem) involves the *Dirichlet integral* (also called *gradient energy* or *Dirichlet energy* or Dirichlet *form*)

$$\mathcal{D}(u) = \frac{1}{2} \int_\Omega |\nabla u(x,y)|^2 \, dxdy . \tag{16}$$ Di

The norm of the gradient is the euclidean norm

$$|\nabla u(x,y)|^2 = (\,\partial_x u(x,y))^2 + (\,\partial_y u(x,y))^2 \ .$$

The Dirichlet variational principle for the inhomogeneous Poisson problem is

$$\min_{u=g \text{ on } \Gamma} \ \frac{1}{2}\int_\Omega |\nabla u(x,y)|^2 \ dxdy \ + \ \int_\Omega f(x,y)\, u(x,y)\, dxdy \ . \qquad (17) \quad \boxed{\texttt{Dp}}$$

We will show (just below) that the u that minimizes this *functional* (a functional is a function of a function) satisfies the inhomogeneous Laplace equation (1) for $(x,y) \in \Omega$.

Here is a form of the traditional argument that a minimizer of (17) satisfies the Laplace equation and the boundary conditions. If u is a minimizer, over functions that satisfy the boundary conditions, then you lower the energy with any perturbation $u \leftarrow u + \epsilon v$ as long as $v(x,y) = 0$ on Γ. We can plug this in to get

$$\min \ \frac{1}{2}\int_\Omega |\nabla u(x,y) + \epsilon v(x,y)|^2 \ dxdy \ + \ \int_\Omega f(x,y)\, (u(x,y) + \epsilon v(x,y)\, dxdy \ .$$

You can see that this is

$$E(u) + \epsilon \left[\int_\Omega \nabla u(x,y) \cdot \nabla v(x,y)\, dxdy + \int_\Omega f(x,y) v(x,y)\, dxdy \right] + O(\epsilon^2) \ .$$

The order ϵ term must vanish for every v if u is a minimizer (why?) so if $v = 0$ on Γ, then

$$\int_\Omega \nabla u(x,y) \cdot \nabla v(x,y)\, dxdy + \int_\Omega f(x,y) v(x,y)\, dxdy = 0 \ .$$

The PDE comes in via integration by parts, which is sometimes called *Green's theorem*. If $v = 0$ on Γ, then

$$\int_\Omega \nabla u(x,y) \cdot \nabla v(x,y)\, dxdy = -\int_\Omega \triangle u(x,y)\, v(x,y)\, dxdy$$

This implies that for every v,

$$\int_\Omega [\,\triangle u(x,y) - f(x,y)\,]\, v(x,y)\, dxdy = 0 \ .$$

That is possible only if $\triangle u(x,y) = f(x,y)$ for all $(x,y) \in \Omega$.

This argument is not mathematically rigorous because we did not show that u and v are differentiable. A rigorous version of this argument was given by Hermann Weyl almost a century after Dirichlet gave this informal version. The discrete version below is rigorous because it is finite dimensional and involves only "ordinary" partial derivatives rather than variations in function space.

sec:vdLe

4.3 For the discrete Laplace equation

The linear system coming from the discretization (5) also have a variational principle coming from an SPD matrix A. You can guess that that this may be found using a discrete approximations to the integrals in (17). A natural discrete approximation to the integral fu is

$$\int_\Omega f(x,y)\,u(x,y)\,dxdy \approx \Delta x^2 \sum_{j=1}^{n}\sum_{k=1}^{n} f_{jk}U_{jk}\ .$$

The discrete analogue of the Dirichlet integral is the discrete Dirichlet sum we already found. Here is some reasoning to see how you could find it by looking for discrete analogues to continuous integrals. We use the one-sided approximate gradients

$$\partial_x u(x_j,y_k) \approx \frac{u(x_j+\Delta x,y_k)-u(x_j,y_k)}{\Delta x}$$
$$\partial_y u(x_j,y_k) \approx \frac{u(x_j,y_k+\Delta x)-u(x_j,y_k)}{\Delta x}$$

These approximations combine to make an estimate of the integral of the square gradient in a grid box

$$B_{jk} = \{\, x_j \le x \le x_{j+1}\ ,\ \ y_k \le y \le y_{k+1}\,\}\ .$$

$$\int\int_{B_{jk}} |\nabla u(x,y)|^2\,dxdy$$
$$\approx \Delta x^2\left[\left(\frac{u(x_j+\Delta x,y_k)-u(x_j,y_k)}{\Delta x}\right)^2+\left(\frac{u(x_j,y_k+\Delta x)-u(x_j,y_k)}{\Delta x}\right)^2\right]$$

The factors of Δx cancel on the right. We want to include all the grid boxes inside the square, even the ones that touch the boundary. When we sum over all boxes, j and k should run from 0 to n. Thus, the Dirichlet sum, which is the discrete approximation to the Dirichlet integral, should be

$$\mathcal{D}_n(U) = \frac{1}{2}\sum_{j=0}^{n}\sum_{k=0}^{n}\left[(U_{j+1,k}-U_{jk})^2+(U_{j,k+1}-U_{jk})^2\right]\ . \qquad (18)$$ dDs

This is clearly a quadratic form in the vector U. We take zero boundary conditions and "implement" them using the ghost cell idea of setting $U_{0,k}=0$, $U_{n+1,k}=0$, etc.

We find the matrix A corresponding to this quadratic form by computing

$$\frac{\partial \mathcal{D}_n(U)}{\partial U_{lm}} = (AU)_{lm}\ .$$

The quantity U_{lm} appears in four of the terms in the sum (22). The terms and corresponding derivatives are

$$
\begin{aligned}
j+1 = l, k = m \ : \quad & \frac{1}{2}\frac{\partial (U_{l,m} - U_{l-1,m})^2}{\partial U_{lm}} = U_{l,m} - U_{l-1,m} \\
j-1 = l, k = m \ : \quad & \frac{1}{2}\frac{\partial (U_{l+1,m} - U_{l,m})^2}{\partial U_{lm}} = U_{l,m} - U_{l+1,m} \\
j = l, k = m+1 \ : \quad & \frac{1}{2}\frac{\partial (U_{l,m} - U_{l,m-1})^2}{\partial U_{lm}} = U_{l,m} - U_{l,m-1} \\
j = l, k = m-1 \ : \quad & \frac{1}{2}\frac{\partial (U_{l,m+1} - U_{l,m})^2}{\partial U_{lm}} = U_{l,m} - U_{l,m+1}
\end{aligned}
$$

We add these terms to get the overall derivative

$$
\frac{\partial \mathcal{D}_n(U)}{\partial U_{lm}} = (U_{l,m} - U_{l-1,m}) + (U_{l,m} - U_{l+1,m}) + (U_{l,m} - U_{l,m-1}) + (U_{l,m} - U_{l,m+1})
$$

This gives

$$
(AU)_{lm} = 4U_{lm} - (U_{l-1,m} + U_{l+1,m} + U_{l,m-1} + U_{l,m+1}) \ .
$$

This is a form of the discrete Laplace operator with diagonal entries all equal to $+4$ and non-zero off-diagonal entries equal to -1.

5 Stability, Poincaré inequalities

sec:Pi

Poincaré inequalities[5] are inequalities that bound a norm of a function by a norm of its gradient. Inequalities of this kind are central to the theory of elliptic partial differential equations. They also address an issue that arises in infinite dimensional space but not in finite dimensions. In infinite dimensions it is possible that a quadratic form is positive in the sense that $Q(u) > 0$ if $u \neq 0$ and yet is not positive definite in the sense that there is a $\mu > 0$ so that $Q(u) \geq \mu \left\| u \right\|^2$. The continuous Poincaré inequality gives such a positive μ. The discrete version gives a positive μ that does not go to zero as $\Delta x \rightarrow 0$.

We use a discrete version of a Poincaré inequality here to show that the discretization (5) of the Laplace equation is *stable*. The discrete equations (5) are solvable in the sense that the number of equations is equal to the number of unknown values U_{jk}, both numbers being n^2. With a little more thinking, it can be shown that the discrete equations $AU = F$ are solvable because $AU = 0$ has no solutions except $U = 0$. This implies that for any n and for any norms $\left\| U \right\|_\alpha$ and $\left\| F \right\|_\beta$ (the subscripts α and β are only meant to denote that there is

[5] Henri Poincaré was a French mathematician active in the decades around 1900. He revolutionized many fields of mathematics, including topology (the Poincaré conjecture), dynamical systems (his book *Les Méthodes Nouvelles de la Méchanique Céleste* (new methods of celestial mechanics) was the beginning of modern chaos theory) and analytic number theory.

a choice of norms, and that the norms for U and F might be different) there is a $C_{n,\alpha,\beta}$ so that

$$\|U\|_\alpha \leq C_{n,\alpha,\beta}\|F\|_\beta \ . \tag{19}$$ gi

In numerical analysis it matters how $C_{n,\alpha,\beta}$ depends on n. We look for specific norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ so that $C_{n,\alpha,\beta}$ is independent of n as $n \rightarrow \infty$. We would say that the method is *stable* in the norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$.

Stability is the intuitive property that U does not "blow up" as $\Delta x \rightarrow 0$ and (this is the same thing) $n \rightarrow \infty$. We hope that U converges to the actual solution u as $\Delta x \rightarrow 0$, so $U \not\rightarrow \infty$ seems to be a prerequisite. Conversely, we will see that if the method is stable then U does converge to u as $\Delta x \rightarrow 0$. The hardest, most technical and most abstract parts of numerical analysis concern stability. To say that a method is stable, you find yourself saying something like: "There is a C and a norm $\|\cdot\|$ so that for all Δx sufficiently small, $\cdots$ (some inequality involving carefully chosen norms)." Stability inequalities for discrete approximations usually are discrete versions of an inequality that applies to solutions of the target PDE. To get there, you need to find the right PDE inequality and the right discrete analogue of it and then find a way to prove the discrete inequality.

Coming back to the Poincaré inequality, suppose Ω is the unit square ($0 \leq x \leq 1$ and $0 \leq y \leq 1$) and $u(x,y)$ is differentiable with $u = 0$ on Γ, then there is a constant C so that (proof below)

$$\int_\Omega u(x,y)^2 \, dxdy \ \leq \ C \int_\Omega |\nabla u(x,y)|^2 \, dxdy \ . \tag{20}$$ Pi

There is a similar inequality for functions U defined on a grid. To state this we need discrete analogues of both sides of (20) with the right powers of Δx. Suppose $U_{jk} = 0$ when $(x_j, y_k) \in \Gamma$ and that U_{jk} is defined for $j = 1, \cdots, n$ and $k = 1, \cdots, n$. A discrete approximation to the L^2 norm on the right side is

$$\|U\|_2^2 = \Delta x^2 \sum_{j=1}^{n} \sum_{k=1}^{n} U_{jk}^2 \ . \tag{21}$$ dU2

The prefactor Δx^2 makes the double sum consistent with the double integral, so that if $\widetilde{U}_{jk} = u(x_j, y_k)$, then

$$\left\| \widetilde{U} \right\|_2^2 \ \rightarrow \ \int_\Omega u(x,y)^2 \, dxdy \ , \quad \text{as } \Delta x \rightarrow 0 \ .$$

Powers of Δx only change the constant $C_{n,\alpha,\beta}$ in the generic inequality (19), but you have to get them right if you want an inequality where C stays bounded as $n \rightarrow \infty$. We are looking for a discrete inequality that is consistent with the continuous one we already know (from a previous PDE class, proof "reviewed" below). That makes it more likely that the discrete inequality might actually be true with a constant independent of n.

The discrete analogue of the Dirichlet integral is the discrete Dirichlet sum we already found. Here is some reasoning to see how you could find it by looking

for discrete analogues to continuous integrals. We use the one-sided approximate gradients

$$\partial_x u(x_j, y_k) \approx \frac{u(x_j + \Delta x, y_k) - u(x_j, y_k)}{\Delta x}$$
$$\partial_y u(x_j, y_k) \approx \frac{u(x_j, y_k + \Delta x) - u(x_j, y_k)}{\Delta x}$$

These approximations combine to make an estimate of the integral of the square gradient in a grid box

$$B_{jk} = \{\, x_j \le x \le x_{j+1} \;,\;\; y_k \le y \le y_{k+1} \,\} \;.$$

$$\int\int_{B_{jk}} |\nabla u(x,y)|^2 \, dxdy$$
$$\approx \Delta x^2 \left[\left(\frac{u(x_j + \Delta x, y_k) - u(x_j, y_k)}{\Delta x} \right)^2 + \left(\frac{u(x_j, y_k + \Delta x) - u(x_j, y_k)}{\Delta x} \right)^2 \right]$$

The factors of Δx cancel on the right. We want to include all the grid boxes inside the square, even the ones that touch the boundary. When we sum over all boxes, j and k should run from 0 to n. Thus, the Dirichlet sum, which is the discrete approximation to the Dirichlet integral, should be

$$\mathcal{D}_n(U) = \sum_{j=0}^{n} \sum_{k=0}^{n} \left[(U_{j+1,k} - U_{jk})^2 + (U_{j,k+1} - U_{jk})^2 \right] \;. \qquad (22) \quad \boxed{\texttt{dDs}}$$

The discrete Poincare inequality is that there is a C, which is *independent of* n, so that

$$\|U\|_2^2 \le C \, \mathcal{D}_n(U) \;. \qquad (23) \quad \boxed{\texttt{dPi}}$$

The factor Δx^2 in the definition of the 2−norm (21) is necessary for C to be independent of n.

We give a proof of the discrete Poincaré inequality is a discrete analogue of a part of a proof of the continuous inequality (20). The full proof of the continuous Poincaré inequality has technicalities related to what functions u are allowed. The discrete version (23) does not have this issue because the inequality holds for every vector U.

We give one of the proofs of the formal part of the continuous Poincaré inequality. By "formal" we mean that we assume that u has enough derivatives. Such an incomplete proof should be called "informal", but usually is called "formal" because it has the form of a proof without going through all the details. This proof illustrates a common trick in proving inequalities involving integrals: prove a simpler inequality and then integrate to get the desired result. Another trick is that if you add something positive to the right side of a true inequality, you get another true inequality. In this case, we start with a one dimensional Poincarë inequality that involves a function $v(x)$ with $v(0) = 0$ and $v(1) = 0$

and relates an integral involving v to an integral involving the derivative. Then we integrate over y to get an integral involving $u(x,y)$, then we add the y derivatives.

The one variable inequality is

$$\int_0^1 v(x)^2\,dx \le C\int_0^1 \left(v'(x)\right)^2\,dx\,. \tag{24}$$ ovPi

To prove this, we use the *Cauchy Schwarz inequality* (look it up if you don't know it). If f and g are "any" two functions, then

$$\int f(x)g(x)\,dx \le \sqrt{\int f(x)^2dx \int g(x)^2dx}\,.$$

If a is any number between 0 and 1, then (using the hypothesis that $v(0)=0$)

$$v(a) = \int_0^a v'(x)\,dx\,.$$

We use a trick involving Cauchy Schwarz:

$$\begin{aligned}
v(a) &= \int_0^a v'(x)\,dx \\
&= \int_0^a 1\cdot v'(x)\,dx \\
&\le \sqrt{\int_0^a 1^2\,dx \int_0^a \left(v'(x)\right)^2\,dx} \\
&= \sqrt{a}\sqrt{\int_0^a \left(v'(x)\right)^2\,dx} \\
v(a)^2 &\le a\int_0^a \left(v'(x)\right)^2\,dx \\
v(a)^2 &\le \int_0^1 \left(v'(x)\right)^2\,dx\,.
\end{aligned}$$

The last uses the fact that the integrand $\left(v'(x)\right)^2$ is never negative, so integrating from 0 to 1 instead of from 0 to a cannot make the value smaller. We also use the assumption that $0 \le a \le 1$ so replacing a with 1 only makes right side bigger. Finally, integrate with respect to a from 0 to 1 and you get

$$\int_0^1 v(a)^2da \le \int_0^1 \left(\int_0^1 \left(v'(x)\right)^2\,dx\right)da = \int_0^1 \left(v'(x)\right)^2\,dx$$

This gives the one variable Poincaré inequality (24). ovPi

This can be integrated in y to give a two variable inequality. For any fixed y, the one variable inequality implies that

$$\int_0^1 u(x,y)^2\,dx \le \int_0^1 \left(\partial_x u(x,y)\right)^2\,dx\,.$$

Now integrate over y and get

$$\int_{y=0}^{y=1}\int_{x=0}^{x=1} u(x,y)^2\,dxdy \le \int_{y=0}^{y=1}\int_{x=0}^{x=1} \left(\partial_x u(x,y)\right)^2 dxdy\ . \qquad (25)$$ Pii

These integrals are over the unit square, which we have been calling Ω. The square gradient is

$$|\nabla u|^2 = (\partial_x u)^2 + (\partial_y u)^2\ .$$

We can add in the y derivative part to the right side of (25) and switch notation to get

$$\int_\Omega u(x,y)^2 dxdy \le \int_\Omega \left[(\partial_x u(x,y))^2 + (\partial_y u(x,y))^2\right] dxdy\ .$$

This is the Poincaré inequality (20).

6 Iterative solution methods

sec:im

Iterative methods are methods for solving discrete equation systems like (5). This section first explains why people use iterative methods, then it gives an example, the Gauss Seidel method. You will learn, if you code it, that Gauss Seidel is simple and reliable but very slow. Later classes will try to explain this slowness and use that understanding to make better iterative solvers.

6.1 Why iterative methods

sec:why

Most of the computer time in finding U goes into solving the discrete equations (5). These equations are linear, so they can be solved by *assembling* the appropriate matrix A (calculating and storing all its entries) and then using a linear algebra package to solve the corresponding equations

$$AU = F\ .$$

This direct approach is impractical for large n because A is too big. The unknowns U_{jk} form a vector $U \in \mathbb{R}^M$, with $M = n^2$. The matrix A has $M^2 = n^4$ entries. Direct Cholesky factorization for the linear system take about $\frac{1}{6}M^3 = \frac{1}{6}n^6$ operations. For $n = 100$, A has $100^4 = 100{,}000{,}000$ entries and the operation count for Cholesky is $\frac{1}{6}10^{12}$.

Sparse direct solvers lead to (much faster) algorithms with smaller powers of n. A matrix is *entrywise sparse* (often just called "sparse") if most of its entries are zero. The discrete Laplace matrix has at most 5 non-zero elements per row, because each of the discrete equations (5) involves at most 5 of the unknowns U_{jk}.

The matrix is also *banded*. Suppose you put the U_{jk} in *row major* order

$$U = U_{11}, \cdots, U_{1n}, U_{21}, \cdots, U_{2n} \cdots)\ .$$

Then U_{jk} is coupled only to adjacent entries $U_{j-1,k}$ and $U_{j+1,k}$ and to entries a distance away, $U_{j,k-1}$ and $U_{j,k+1}$. Thus, all the entries of A a distance more

than n from the diagonal are zero. The $M \times M$ matrix has *bandwidth* equal to n (or maybe $n+1$ or $2n+1$, depending on how you define "width"). The work for Cholesky factorization of such a matrix is (from Numerical Methods I)

$$(\text{size}) \cdot (\text{bandwidth})^2 = M \cdot n^2 = n^4 \ll n^6 \ .$$

A "band-solver" (Cholesky code for a banded matrix) can easily handle a 100×100 discrete Laplace problem. In 3D with an $n \times n \times n$ mesh, the bandwidth is n^2 and the work count is

$$(\text{size}) \cdot (\text{bandwidth})^2 = n^3 \cdot \left(n^2\right)^2 = n^7 \ .$$

The band-solver from LAPACK (or other top quality software) can do this on a laptop.

Band structure does not fully describe our sparsity. A matrix with bandwidth n could have $2n+1$ non-zeros per row, but ours has only 5. There is sophisticated *sparse matrix software* that stores only the non-zero entries of a matrix. Of course, it also has to store the locations of these non-zeros. It also tries to re-order the equations to reduce *fill-in*. The Cholesky factors of a sparse matrix are generally not sparse even when the matrix itself is sparse. For example, the factors of our banded matrix have non-zeros in every entry within the band (believe this, please). A zero in A that becomes non-zero in a factor is said to be *filled in*. There are clever algorithms and powerful heuristics that can reduce fill-in by big factors. But even these do not make the hardest problems practical.

6.2 Overview of iterative methods

sec:ov

Iterative methods

It is easy to write code to "apply" A without computing and storing its entries. A procedure that applies A takes a grid vector V as input and returns the vector $U = AV$. For example, if the equations are written in the form (6), dLes the entries of A are either 1 or $-\frac{1}{4}$. The code to compute AV would have in its inner loop

```
U[j,k] = V[j,k] - ( V[j-1,k] + V[j+1,k] + V[j,k-1]+V[j,k+1] )/4
```

Ghost cells are a programming trick to simplify the code that calculates AV. A *ghost cell* (more properly ghost "point") is a point on the boundary, corresponding to j or k being equal to 0 or $n+1$. The issue is that The trick is to define the arrays V and U so the indices j and k can run from 0 to $n+1$, which makes them $(n+1) \times (n+1)$ arrays in the computer.

6.3 Gauss Seidel iteration

sec:GS

The *Gauss Seidel* algorithm is an *iterative* algorithm for solving a system of equations

$$AU = F \ . \tag{26}$$ gls

One Gauss Seidel iteration involves "sweeping" through the components of the current *iterate* in some order. At each step, you solve equation j for component j, leaving the other components unchanged. You then replace the old component j with the new one and move on to component $j+1$. The component $j+1$ *update* uses the new component j value, which overwrote the old j value.

A general *iterative algorithm* computes a sequence of iterates, $U^{(1)}$, $U^{(2)}$, $\cdots$. $U^{(k)}$, $\cdots$. The *residual* it iteration k is the amount by which $U^{(k)}$ fails to satisfy the equations:

$$R^{(k)} = AU^{(k)} - F \ . \tag{27}$$ Rk

This way, $R^{(k)} = 0$ implies that $U^{(k)}$ satisfies the system (26). The residual of equation j is (supposing there are M components in all, and that a_{ij} is the (i,j) component of A)

$$R_j^{(k)} = \sum_{i=1}^{M} a_{ji} U_i^{(k)} - F_j \ .$$

Suppose we are doing iteration k, sweeping through the components, and have reached component j. The values $U_i^{(k+1)}$ for $i < j$ have been computed and we seek $U_j^{(k+1)}$. The residual of equation j, keeping all components fixed except component j, is

$$\sum_{i<j} a_{ji} U_i^{(k+1)} + a_{jj} U_j^{(k+1)} + \sum_{i>j} a_{ji} U_i^{(k)} - F_j \ .$$

The first sum involves the components of $U^{(k+1)}$ that have already been computed. The second sum involves components of $U^{(k)}$ that have not yet been updates. We set this residual to zero by taking component j of the new iterate to be

$$U_j^{(k+1)} = \frac{1}{a_{jj}} \left(F_j \ - \ \sum_{i<j} a_{ji} U_i^{(k+1)} \ - \ \sum_{i>j} a_{ji} U_i^{(k)} \right) \ . \tag{28}$$ GSgm

This is the Gauss Seidel algorithm.

The code for the Gauss Seidel algorithm is simpler than (28) suggests. You need storage for only one vector U, because once $U_j^{(k+1)}$ is computed, the old value $U_j^{(k)}$ is never used. The code in Figure 1 illustrates this. The array (vector) `U` has entries `U[i]`, which represent either $U_i^{(k+1)}$ (if $i < j$) or $U_i^{(k)}$ (if $i > j$).

You might wonder why one sweep does not give the exact solution. After all, the new value $U_j^{(k+1)}$ is chosen to make the residual of equation equal to zero. Unfortunately, this makes the residual of equation $j-1$ not equal to zero. The residual calculation when U_{j-1} was updated used the old value $U_j^{(k)}$, not the new value $U_j^{(k+1)}$. With luck, the iterates converge as $k \to \infty$.

```
15
16  for k in range(iterations):
17      for j in range(M):
18          R = F[j]
19          for i in range(M):
20              if ( i != j ):
21                  R -= a[j,i]*U[i]
22              U[j] = (1/a[j,j]) * R
```

Figure 1: Code fragment (Python) to implement the Gauss Seidel iteration (GSgm 28). When the code gets to line 22, the variable `R` is the quantity inside parentheses there. The code has a single vector `U`. The values `U[i]` for $i < j$ represent $U_i^{(k+1)}$, and those with $i > j$ represent $U_i^{(k)}$. Only the value $i = j$, which would be $U_j^{(k)}$ is left out of the sum. fig:GS_loop

6.4 Variational structure and convergence

sec:GSv

The Gauss Seidel iteration might or might not converge, depending on the matrix A. We will see that it converges if A is SPD, because of a variational interpretation of the algorithm that is related to the variational formulation of the linear system. Even without the variational structure, you can see that A being SPD implies that the diagonals a_{jj} are not zero (because they are positive), so at least the equation (GSgm 28) makes sense. Another matrix might have zeros on the diagonal. For example

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} .$$

The linear system $AU = F$ for this A is equivalent to the equations $U_2 = F_1$ and $U_1 = F_2$. The linear system is easy to solve, but not by using Gauss Seidel iteration. A more nuanced example is

$$\begin{pmatrix} \epsilon & 1 \\ 1 & \epsilon \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} . \tag{29}$$ bex

The Gauss Seidel algorithm (GSgm 28) becomes (check this algebra)

$$\begin{aligned} U_1^{(k+1)} &= \frac{1}{\epsilon} \left(F_1 - U_2^{(k)} \right) \\ U_2^{(k+1)} &= \frac{1}{\epsilon} \left(F_2 - U_1^{(k+1)} \right) \\ &= \frac{1}{\epsilon} \left(F_2 - F_2 \right) + \frac{1}{\epsilon^2} U_2^{(k)} . \end{aligned}$$

If $|\epsilon| < 1$ and the initial guess $U_2^{(0)}$ is not exactly equal to the solution value for the linear system (bex 29), the iterates $U_2^{(k)}$ diverge as $k \to \infty$. Note that the matrix in (bex 29) is not positive definite. Indeed,

$$\begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \epsilon & 1 \\ 1 & \epsilon \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = 2U_1U_2 + \epsilon\left(U_1^2 + U_2^2 \right) .$$

If $U_2 = -U_1$, this is $2(\epsilon - 1)U_1^2$, which is negative if $\epsilon < 1$ and $U_1 \neq 0$.

If A is SPD, then the Gauss Seidel iteration may be understood as minimizing the functional $E(V)$ (see (9)) over component j. Suppose we are doing the sweep to compute $U^{(k+1)}$ from $U^{(k)}$ and have updated the components U_i with $i < j$. Consider a vector where component j is called W and the other components have fixed values either from iteration $k+1$ or from iteration k:

$$V(W) = (U_1^{(k+1)}, \cdots, U_{j-1}^{(k+1)},\ W,\ U_{j+1}^{(k)}, \cdots, U_M^{(k)}) \ .$$

Then the derivative of $E(V(W))$ with respect to W is component j of $\nabla E(V(W))$. According to the derivative calculation (13), this is component j of the residual $R(W) = AV(W) - F$. If we minimize E over W, this is equivalent (because the function is strictly convex) to setting the derivative with respect to W equal to zero. That (we just saw) is equivalent to setting component j of $A(V(W)) - F$ equal to zero, which is the Gauss Seidel algorithm.

A consequence of minimizing is that E decreases at every step of every iteration. When you replace $U_j^{(k)}$ without changing the other components, choosing the optimal $U_j^{(k+1)}$ ensures that E decreases. The difference between the right and left sides is only that $U_j^{(k)}$ on the right is replaced by $U_j^{(k+1)}$ on the left:

$$E(U_1^{(k+1)}, \cdots, U_j^{(k+1)}, U_{j+1}^{(k)}, \cdots, U_M^{(k)}) < E(U_1^{(k+1)}, \cdots, U_j^{(k)}, U_{j+1}^{(k)}, \cdots, U_M^{(k)})$$